Docker - Beginner Biologist 3

Jean-Yves Sgro

2019

1 Learning objectives

- Select Docker containers from the docker hub
- Use multiple Docker container to accomplish related tasks
- Learn about Docker image and Docker files
- Bypass “Entrypoint” of Docker images

> *The main learning objective is to experience different methods of using dockers images and face (**and resolve**) challenges while doing so.*

In class these exercises will be run onto the classroom iMacs.

However, as best as I can I’ll provide Windows hints and instructions when possible, but a basic understanding of line-command under Windows would be more than useful for that (*e.g.* know what is **DOS** for example. See APPENDIX C.)

1.1 Requirements

- Be familiar with Docker or follow workshop 1 “Docker - Beginner Biologist 1” and workshop 2 “Docker - Beginner Biologist 2.”
- Docker will be used from a line-command terminal: **`Terminal`** on a Macintosh in the classroom. A rudimentary knowledge of `bash` command-line is necessary.
- If you are a Windows user: `PowerShell` can be used as a Terminal. However, setting Docker to run on Windows is more involved (not covered in class.)
- **Docker username**: downloads will require a (free) username, therefore registration is necessary in order to follow the tutorial. Go to https://hub.docker.com (https://hub.docker.com) and use the button “Sign up for Docker Hub” to register.

2 Set-up

Tutorials will be held in the Biochemistry classroom 201, and Docker has already be installed.

Instruction for installation can be found on the install link[1] of the Docker web site.

> *Note* HTML Version only:

If you are following this document in **HTML format** the code is shown with a colored background:

```
Green background: commands from local computer bash terminal
```

```
White background: standard output of programs.
```

```
Blue background: commands and output when WITHIN a bash container
```

```
Yellow background: commands or output for information. Do not run!
```

2.1 Getting started

To get started we need to open a text terminal as detailed below. In class we’ll use a Macintosh.

TASK:

Do one of the following:.

If you are on a Macintosh:

1. Find the `Terminal` icon in the `/Applications/Utilities` directory. Then double-click on the icon and `Terminal` will open.
2. **OR** use the top-right icon that looks like a magnifying glass (*Spotlight Search*,) start typing the word `Terminal` and press return. `Terminal` will open.

If you are on a PC:

1. Find `Power Shell` *e.g.* using Windows search or Cortana. This will open a suitable text-based terminal.

(*Note*: Windows `cmd` does not offer the appropriate commands.)

2.2 Version check

This ensures that Docker is properly installed. The exact running version itself is not very important.

At the `$` or `>` prompt within the window of `Terminal`, `cmd` or `PowerShell` type `docker --version` to check the version currently installed.

```
docker --version
```

```
Docker version 19.03.5, build 633a0ea
```

2.3 Docker login: Required!

Before going further, it is necessary now to login with your Docker Hub ID. You should already have created one before this or the previous workshop. If you need to create an ID now go to https://hub.docker.com (https://hub.docker.com) to register.

TASK:

Docker login:.

```
docker login
```

```
Login with your Docker ID to push and pull images from Docker Hub.
If you don't have a Docker ID, head over to https://hub.docker.com
to create one.
Username: YOUR_DOCKER_ID_HERE
Password:
Login Succeeded
$
```

Note: if you do not login first you will receive an error message when tryingt to start docker in the next steps.

3 Choosing a docker image

In the previous workshop we learned how to find, choose, and `pull` (download) a docker image from the Docker hub (https://hub.docker.com).

Today we'll download multiple images to access programs in each to accomplish preliminary tasks to genomic analysis. These images belong to a larger project called *Bioinformatics Docker Images Project*[2].

This group of docker images provides useful information on how to run the software contained within. However, there are unclear desciptions and errors that we'll overcome during the workshop.

We will use the following images:

- **clustalomega** (https://hub.docker.com/r/pegi3s/clustalomega/) (doc) (https://www.ebi.ac.uk/seqdb/confluence/display/THD/Clustal+Omega) - Sequence alignment
- **fastqc** (https://hub.docker.com/r/pegi3s/fastqc/) (doc) (https://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help/) - Sequence read quality assessment
- **sratoolkit** (https://hub.docker.com/r/pegi3s/sratoolkit) (doc) (https://trace.ncbi.nlm.nih.gov/Traces/sra/sra.cgi?view=toolkit_doc) - Operations on SRA database

We can get the appropriate `pull` request from the web pages on the Docker hub.

TASK:

pull images.

```
docker pull pegi3s/clustalomega

docker pull pegi3s/fastqc

docker pull pegi3s/sratoolkit
```

We can list images with:

```
docker image ls pegi3s/*
```

```
REPOSITORY           TAG         IMAGE ID          CREATED
SIZE
pegi3s/sratoolkit    latest      12fb29fcba17      2 months ago
306MB
pegi3s/fastqc        latest      5a439982c750      4 months ago
579MB
pegi3s/clustalomega  latest      ed9da1fc309e      4 months ago
290MB
```

> *Note* on size: Docker images are constructed in *layers* that can be common over multiple images. Therefore, the actual disk space to store multiple images is less than the sum of the `SIZE` that is reported in the list.

4 clustalomega

From web `clustalomega` documentation[3]: *Clustal Omega is a multiple sequence alignment program for aligning three or more sequences together in a computationally efficient and accurate manner. It produces biologically meaningful multiple sequence alignments of divergent sequences.*

A full description of the algorithms used by Clustal Omega is available in the Molecular Systems Biology paper *Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega* (Sievers et al. (2011).) Latest additions to Clustal Omega are described in *Clustal Omega for making accurate alignments of many protein sequences (Sievers and Higgins (2018).)

4.1 multiple sequences fasta file

As a follow-up to our EMBOSS sequence alignment we'll use this program to make a very small multiple sequence alignment of the short glucagon peptide family that was saved in a shared directory.

Note: If you need to create the shared directory and the sequence files see below in APPENDIX A.

The glucagon family sequence files are in individual fasta format. We first need to create a fasta file that contains them all with a simple `cat` command:

TASK:

Combine sequences.

```
cd $HOME/dockershare
cat *.fa > sequences.fasta
```

The final, multiple sequence fasta file contains all the data:

```
cat sequences.fasta
```

```
>GIP
YAEGTFISDYSIAMDKIRQQDFVNWLLAQ
>GLP-1
HAEGTFTSDVSSYLEGQAAKEFIAWLVKGRG
>GLP-2
HADGSFSDEMNTILDNLAARDFINWLIQTKITD
>glucagon
HSQGTFTSDYSKYLDSRRAQDFVQWLMNT
```

4.2 clustalomega image

We are now ready to apply what we learned in the previous workshop: we have downloaded (pulled) the docker image and we know how to share a directory… This should be a breeze right?!

We'll see!

First we need to explore the container to see where we can "attach" (map) the `dockershare` directory. In a previous example it was `/data` but we need to see if that directory actually exists in this image.

It seems that it *could* exist *if we trust the info* from the docker hub page for **`pegi3s/clustalomega`** :

You should adapt and run the following command:

```
docker run --rm -v /your/data/dir:/data pegi3s/clustalomega
-i /data/sequences.fasta  -o /data/sequences_aligned.fasta
```

In this command, you should replace:

- `/your/data/dir` to point to the directory that contains the FASTA file you want to align.
- `sequences.fasta` to the actual name of your FASTA file.
- `sequences_aligned.fasta` to the actual name of your aligned FASTA file.

So we should be good:

- `/your/data/dir` can be `$HOME/dockershare`
- We have created the multiple sequence files above and named it `sequences.fasta` .
- `sequences_aligned.fasta` would be the written output.

Since we would share the `/data` folder henceforth this should work OK.

4.3 clustalomega manual help

We can create a temporary container to request the help provided by the program itself. For this purpose we can add `-h` or `--help` . The program will print help informationon the screen.

Note: At this point we do not request a shared directory and the command can be given from within any directory we are at the moment on the local computer.

```
docker run -it --rm  pegi3s/clustalomega -h
```

For more details, the **`README`** file is available online[4].

Below we'll follow an example based on the sequences we have and inspired by the example on the docker hub page.

4.4 clustalomega container

We can now run `clustalomega` from a container. The example given on the web page adds the name of the files on the `docker run` command itself:

TASK:

Run command.

```
cd $HOME/dockershare
docker run -it --rm -v $HOME/dockershare:/data pegi3s/clustalomega
  -i /data/sequences.fasta -o /data/sequences_aligned.fasta
```

After completing the task the container exists and we are back to the host computer prompt.

We e can type the alignment file on the screen:

```
cat sequences_aligned.fasta
```

```
>GIP
YAEGTFISDYSIAMDKIRQQDFVNWLLAQ----
>GLP-1
HAEGTFTSDVSSYLEGQAAKEFIAWLVKGRG--
>GLP-2
HADGSFSDEMNTILDNLAARDFINWLIQTKITD
>glucagon
HSQGTFTSDYSKYLDSRRAQDFVQWLMNT----
```

The `-` represent the gaps. Therefore it worked.

However, if you re-run the same command (using the same file names) you'll get this error:

```
FATAL: Cowardly refusing to overwrite already existing file
'/data/sequences_aligned.fasta'. Use --force to force overwriting.
```

Therefore we learn that adding `--force` will fix that problem.

Other errors might suggest to look into the help:

```
For more information try: clustalo --help
```

The help page is rather long, but we can concentrate on the output file and format information:

```
Alignment Output:
   -o, --out, --outfile={file,-} Multiple sequence alignment output
  file (default: stdout)
   --outfmt={a2m=fa[sta],clu[stal],msf,phy[lip],selex,st[ockholm],v
  ie[nna]}
   MSA output file format (default: fasta)
```

We can re-reun the command with a different format which provides a better visual of an alignment. This would be true for the subset `clu[stal],msf,phy[lip],selex` .

EXERCISE:
Time permitting you can rerun the commands with one or more of these formats.
Do not forget to add `--force` to overwrite the output file.

For example: (written with line continuation mark `\`)

```
docker run -it --rm \
-v $HOME/dockershare:/data pegi3s/clustalomega \
-i /data/sequences.fasta -o \
/data/sequences_aligned.fasta \
--force \
--outfmt=clu

# check output
cat sequences_aligned.fasta
```

```
CLUSTAL O(1.2.4) multiple sequence alignment



GIP             YAEGTFISDYSIAMDKIRQQDFVNWLLAQ----
GLP-1           HAEGTFTSDVSSYLEGQAAKEFIAWLVKGRG--
GLP-2           HADGSFSDEMNTILDNLAARDFINWLIQTKITD
glucagon        HSQGTFTSDYSKYLDSRRAQDFVQWLMNT----
                :::*:* .: .  ::    ::*: **:
```

EXERCISE 2:
Time permitting create a file (named *e.g.* `sequences2.fasta` for longer protein sequence files as detailed in APPENDIX A for “Protein FASTA for clustalomega.”)

Note: There is very little difference between these files, and using the `clu` format for the output will make the results easier to read.

4.5 Explore the container

Perhaps this should have been the first thing to do?!

Unless special instructions are given when the docker image is created, it shoud be possible to launch the container and explore its content as we have done in previous workshops.

The simplest way is to request a shell on the command *e.g.* adding `/bin/bash` or `/bin/sh` at the end of the `docker run` command. For example we ran `docker run -it alpine /bin/sh` in the first workshop.

In the same way the following should work, omitting sharing a directory as we only want to explore:

```
cd $HOME/dockershare
docker run -it --rm pegi3s/clustalomega /bin/sh
```

```
clustal-omega: unexpected argument "/bin/sh"
For more information try: clustalo --help
```

The container does not let us in… In this case we were able to run the `clustalo` software. But it is sometimes useful, or necessary. to dive inside the container… The section below explore these options.

5 Docker files and Entry points

The above problem poses the question of useability of a docker file. In the future we'll be able to create our own docker files and images but for now we have to rely on existing ones.

As we have done in a previous workshop, the best documentation is to explore the docker container to know what is available. In the above example we assume from the documentation that the `/data` directory exists… Since the command worked we can suppose that it exists.

As a quick reminder here is the process to create a container from the beginning, even though for now we only downloaded existing images.

1. Write a *docker file* containing instructions. This file is plain text.
2. Create a binary image with this information (with `docker build` .)
3. Upload the image on the hub (with `docker push` .)
4. Downloadable by other with `docker pull` .
5. Create a container from the image with `docker run`

In order to investigate (and learn useful information at the same time) we have to explore data provided on the Docker hub web page for this image. We will start with the *docker file* which contains instructions and therefore the *blueprint* information on the docker image and containers that are derived.

5.1 Entry point

TASK:

Explore clustalomega docker page.

- Go back to the `clustalomega` docker page: https://hub.docker.com/r/pegi3s

/clustalomega (https://hub.docker.com/r/pegi3s/clustalomega)
- Click on the *Dockerfile* tab

As of this writing this is what it contains:

```
FROM ubuntu:18.04

RUN apt-get update \
    && apt-get install -y wget make g++ libargtable2-dev

RUN wget http://www.clustal.org/omega/clustal-omega-1.2.4.tar.gz -
  O /tmp/clustalomega.tar.gz \
    && tar zxvf /tmp/clustalomega.tar.gz -C /opt/ && rm /tmp/clust
  alomega.tar.gz \
    && cd /opt/clustal-omega-1.2.4/ \
    && ./configure && make && make install

ENTRYPOINT ["clustalo"]
```

This docker file is rather "simple" in the sense that there are only a few lines. This is what is means (`\` is the line-continuation code to stipulate that this is a single line to execute.)

1. `FROM ubuntu:18.04` means that this "generic" version of *Ubuntu* is used as the starting point to which we add the libraries of software that are needed in the next lines.
2. `RUN apt-get update \` update Ubuntu and add a libray
3. `RUN wget` download and install `clustalomega` .
4. `ENTRYPOINT ["clustalo"]` when a container is activated, **immediately run** the `clustalo` program.

The last point is the critical one: as soon as the container is activated, the software `clustalo` is started and therefore all data provided on the `docker run` command line is passed on to the `clustalo` program. It follows that ou command **`docker run -it --rm pegi3s/clustalomega /bin/sh` failed** simply because the argument `/bin/sh` was given directly to the `clustalo` program which did not recognize this as a file name containing sequences as the program expects.

Therefore the problem is that we cannot, at this point, "enter and explore" the container itself because of the `ENTRYPOINT` command that is written "in stone" within the *docker file* and therefore within the *docker image* and subsequently the *docker container*.

5.2 Bypass Entry point

The solution to the problem is to **override** the `ENTRYPOINT` instruction by providing an alternate option on the `docker run` command itself. This is accomplished with the `--entrypoint` option.

To simply explore the `pegi3s/clustalomega` container without sharing a directory we can therefore use the following command:

```
docker run -it --rm  --entrypoint "/bin/bash"  pegi3s/clustalomega
```

```
root@faf2a0788794:/#
```

From this point we can now explore the inside of the docker container:

```
ls
```

```
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root
run  sbin  srv  sys  tmp  usr  var
```

5.2.1 Important note: arguments

(**SKIP this** section if not enough time.)

If we only want to enter the container with the shell command as above it is clear that in that case the `bash` shell (`/bin/bash`) should be installed. That is not always the case. The older, `sh` alternative (`bin/sh`) can be substituted but need also be installed.

Other options exist, for example simply asking to list the content of the default landing directory with `/bin/ls` as the entry point:

```
docker run --rm --entrypoint "/bin/ls"  pegi3s/clustalomega
```

However, there are times when an argument needs to be passed along. In that case it may not be obvious to know where to place the arguments within the `docker run` command[5].

For example, why the `bash` command `ls -l` works on the terminal, the following command would fail:

```
docker run --entrypoint "/bin/ls -l"  pegi3s/clustalomega
```

```
docker: Error response from daemon: OCI runtime create failed: con
tainer_linux.go:346: starting container process caused "exec: \"/b
in/ls -l\": stat /bin/ls -l: no such file or directory": unknown.
ERRO[0001] error waiting for container: context canceled
```

The reason is that the argument, in this case `-l` needs to be provided ***after*** the container is specified. Therefore the correct command would be:

```
docker run --entrypoint "/bin/ls"  pegi3s/clustalomega -l
```

```
total 64
drwxr-xr-x   2 root root 4096 May 15 14:07 bin
drwxr-xr-x   2 root root 4096 Apr 24  2018 boot
drwxr-xr-x   5 root root  340 Oct 24 19:34 dev
[...]
```

For `bash` commands, the arguments need also to be at the end, but the following command with `ls` at the end will fail:

```
docker run --entrypoint "/bin/bash"  pegi3s/clustalomega ls
```

```
/bin/ls: /bin/ls: cannot execute binary file
```

In this case it is necessary to add `-c` for the argument to be send to `bash` properly (see `man bash` for details.) Therefore the following command will work

```
docker run --entrypoint "/bin/bash"  pegi3s/clustalomega -c ls
```

EXERCISE: Try the following commands at the end:

- `-c ls -l`
- `-c "ls -l"`

What is the difference? ____

.

6 Two containers for genomics

In this section we'll use two containers that harbor software that is very useful in the genomics, next generation sequencing (NGS). This will be a short example of using 2 separate docker images to accomplish a task as detailed below.

Task to accomplish:

- Download "paired-end library" of *E.Coli* Illumina sequencing from the NCBI archives
- Check the quality of this NGS sequence reads data

For this we'll use two diferent containers from the pegi3s series (https://pegi3s.github.io/dockerfiles/) that harbor the following software:

- sratoolkit (https://ncbi.github.io/sra-tools/): collection of tools and libraries for using data in the INSDC Sequence Read Archives. We will need the `fastq-dump` utility.
- fastqc (https://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help/) - Sequence read quality assessment.

We already pulled these images earlier. In case you have not done so use the following commands:

```
docker pull pegi3s/fastqc
docker pull pegi3s/sratoolkit
```

6.1 Data

The Sequence Read Archive (SRA, previously known as the Short Read Archive) is a bioinformatics database that provides a public repository for DNA sequencing data, especially the "short reads" generated by high-throughput sequencing (Leinonen, Sugawara, and Shumway (2011)).

SRA

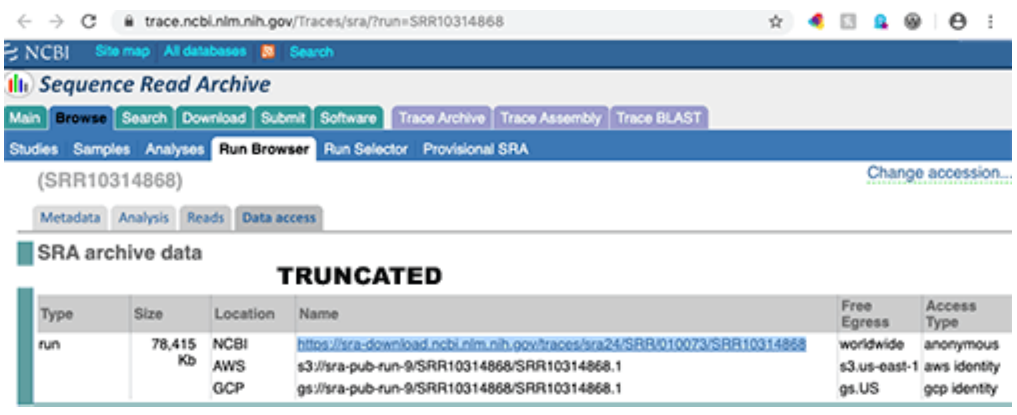
Sequence Read Archive (SRA) makes biological sequence data available to the research community to enhance reproducibility and allow for new discoveries by comparing data sets. The SRA stores raw sequencing data and alignment information from high-throughput sequencing platforms, including Roche 454 GS System®, Illumina Genome Analyzer®, Applied Biosystems SOLiD System®, Helicos Heliscope®, Complete Genomics®, and Pacific Biosciences SMRT®.

The data in the SRA archive is organized by experiment and runs

The data will be from run **SRR10314868 (https://trace.ncbi.nlm.nih.gov/Traces/sra/?run=SRR10314868)** of experiment **SRX7025998 (https://www.ncbi.nlm.nih.gov/sra/SRX7025998)** (see NIH book Understanding SRA Search Results (https://www.ncbi.nlm.nih.gov/books/NBK56913/) for more details.)

On the page for SRR10314868 (https://trace.ncbi.nlm.nih.gov/Traces/sra/?run=SRR10314868) the following cascading tabs show the link to download the data

- Level 1: `Browse` Tab
- Level 2: `Run Browser` Tab
- Level 3" `Data access` Tab



Type	Size	Location	Name	Free Egress	Access Type
run	78,415 Kb	NCBI	https://sra-download.ncbi.nlm.nih.gov/traces/sra24/SRR/010073/SRR10314868	worldwide	anonymous
		AWS	s3://sra-pub-run-9/SRR10314868/SRR10314868.1	s3.us-east-1	aws identity
		GCP	gs://sra-pub-run-9/SRR10314868/SRR10314868.1	gs.US	gcp identity

The web address we'll use later is:
https://sra-download.ncbi.nlm.nih.gov/traces/sra24/SRR/010073/SRR10314868 (https://sra-download.ncbi.nlm.nih.gov/traces/sra24/SRR/010073/SRR10314868)

6.1.1 FastQ

Ultimately we'll want the data in the `fastq` format.

The `fastq` format ia *a text-based format for storing both a biological sequence (usually nucleotide sequence) and its corresponding quality scores. Both the sequence letter and quality score are each encoded with a single ASCII character for brevity.*[6]

See APPENDIX B for more details.

7 sratoolkit

The first task is to download Illumina sequence data from the Sequence Read Archives (SRA (https://www.ncbi.nlm.nih.gov/sra).)

To download the data we'll use a utility called `wget` (*web get*) that *should be present* within the container.

While most NGS data is typically available as "fastq" files (sequence and quality) these files, even compressed with the standard utilities (`.gz` files) are larger thant the format used to archive the data on the SRA.

Therefore, special utility software is necessary to decompressed the files downloaded from the SRA. For this purpose we'll use the `fastq-dump` utility.

7.1 Explore sratoolkit container

Let’s first quickly explore the contents of the `sra` pegi3s/sratoolkit` container. We can already add the shared directory as well:

```
cd $HOME/dockershare

docker run -it --rm -v $HOME/dockershare:/home  pegi3s/sratoolkit
```

We can now ask if the 2 programs we want to use are available with the `which` command:

```
which wget
which fastq-dump
```

```
/usr/bin/wget
/opt/sratoolkit.2.9.6-ubuntu64/bin//fastq-dump
```

We can conclude that indeed these 2 utilities are present and we can now continue.

7.2 Download data

We’ll work within the `home` directory which now maps alls files that are in the `dockershare` directory from the computer host.

TASK:

Donwload data with `wget`.

We’ll use the `wget` utility to download the dat from the web addressed determined above within the SRA web site.

```
cd /home

wget https://sra-download.ncbi.nlm.nih.gov/traces/sra24/SRR/010073
  /SRR10314868
```

```
[...]
(sra-download.ncbi.nlm.nih.gov)|130.14.250.27|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 80296784 (77M) [application/octet-stream]
Saving to: 'SRR10314868'

SRR10314868      100%[============================>]  76.58M  36.2
MB/s    in 2.1s

2019-10-24 23:03:58 (36.2 MB/s) - 'SRR10314868' saved [80296784/80
296784]
```

This created a file called simply `SRR10314868` and we can see how big it is:

```
ls -l SRR10314868
```

```
-rw-r--r--  1 jsgro  AD\Domain Users    77M Oct 20 19:34 SRR103148
68
```

The file is 77 Megabytes. We can compare this later to the size of the fastq or compressed fastq file that will result from the extraction from this SRA archive.

7.3 Unarchive data

Now we can use `fastq-dump` to unarchive `SRR10314868` in order to extract the sequence information into `.fastq` formated file(s).

The `SRR10314868` archive is that of a "paired-end" run *i.e.* each fragment was sequence from both *forward* and *reverse* directions. Most software will prefer that these be stored in two separate files. From the help we can learn that the modifier `--split-file` will "*Dump each read into separate file. Files will receive suffix corresponding to read number.*"[7].

We are now looking within the container. Since the shared directory is `/home` we'll change to this location before running the command.

```
cd /home
fastq-dump --split-files  SRR10314868
```

```
Read 850953 spots for SRR10314868
Written 850953 spots for SRR10314868
```

This may take a one or two minutes. We can then list the files to see how big they are:

```
ls -lh SRR10314868*
```

```
-rw-r--r-- 1 root root  77M Oct 21 00:34 SRR10314868
-rw-r--r-- 1 root root 229M Oct 24 23:31 SRR10314868_1.fastq
-rw-r--r-- 1 root root 229M Oct 24 23:31 SRR10314868_2.fastq
```

We can therefore remark that about 500Mb were compressed into 77Mb within the archive.

Note: to see the content of these text files simply use the `more` command (`q` to quit) or alternatively use the `head` command to see the first 10 lines (default.)

To save space we can optionally delete the `SRR10314868` file (now or later) and we can also compress the fastq files. The software that we'll use in the next step (`fastqc`) can read compressed files directly.

```
gzip SRR10314868_*.fastq
ls -lh SRR10314868_*.gz
```

```
-rw-r--r-- 1 root root 55M Oct 24 23:31 SRR10314868_1.fastq.gz
-rw-r--r-- 1 root root 58M Oct 24 23:31 SRR10314868_2.fastq.gz
```

Thefore, while the `gzip` compression is very good, the SRA compression had achieved a stronger compression level. Both files amount to 113Mbytes while the SRA file was only 77Mbytes.

Conclusion: we now have 2 files from a *paired-end* experiment that we'll test for quality with the `fastqc` program in the next section. The `fastqc` program will be accessed in a different container.

Since we are now done with this container we can exit. Since we started it with the `--rm` option it will be automatically be removed from the system:

```
exit
```

```
$
```

Note: `fastq-dump` could also be used without running inside the container. In this case it would be necessary to also share a directory, but also specify the output directory because the default directory in the container is `/` when starting it. The following command would therefore work as well and produce the same `.fastq` files:

```
docker run -it --rm -v $HOME/dockershare:/home pegi3s/sratoolkit f
  astq-dump  --split-files SRR10314868 ---outdir /home
```

In the next section we'll analyse the quality of the sequence contained in those sequening files.

8 FastQC

From the FastQC web site[8]:

"*FastQC aims to provide a simple way to do some quality control checks on raw sequence data coming from high throughput sequencing pipelines. It provides a modular set of analyses which you can use to give a quick impression of whether your data has any problems of which you should be aware before doing any further analysis.*"

FastQC (Andrews (2010)) can be run in command-line but can also be started with a graphical interface (GUI) in the `X11` environment. This presumes that `X11` is available and running, and that "sharing" communication be established between the container and the host. This is possible on a Linux or a Macintosh system, perhaps with more difficulty on Windows, but is not trivial for a beginner user. We'll explore these options in the next workshops.

For today we'll stick with the command-line option…

8.1 Starting FastQC

We already pulled the docker image earlier. Run this command if you need to download it now:

```
docker pull pegi3s/fastqc
```

We could now explore the container briefly to see which directory could be shared. Preferably an empty directory on the container. In the previous section it was `/home` .

Start the container with the command:

```
docker run -it --rm pegi3s/fastqc
```

```
Exception in thread "main" java.awt.HeadlessException:
No X11 DISPLAY variable was set, but this program performed an ope
ration which requires it.
[truncated output...]
    at uk.ac.babraham.FastQC.FastQCApplication.<init>(FastQCApplic
ation.java:63)
    at uk.ac.babraham.FastQC.FastQCApplication.main(FastQCApplicat
ion.java:332)
```

Should we say “**OOPS?**”

The error indicated that the container will start immediately with FastQC running in graphical mode for `X11` but we are not set for that…

To understand the problem we can go onto the docker page for this image: hub.docker.com/r /pegi3s/fastqc (https://hub.docker.com/r/pegi3s/fastqc)

The paragraph “Using the FastQC image in Linux” has a similar command to start the program that would appear to be what we need, while the next paragraph “Running the FastQC GUI in Linux” details what to do to start the program with `X11` graphical display.

In reality this information is misleading as we just found out.

To understand why we can check the tab named `Dockerfile` as we did earlier with another image. There we’ll see that the ***last line*** says:

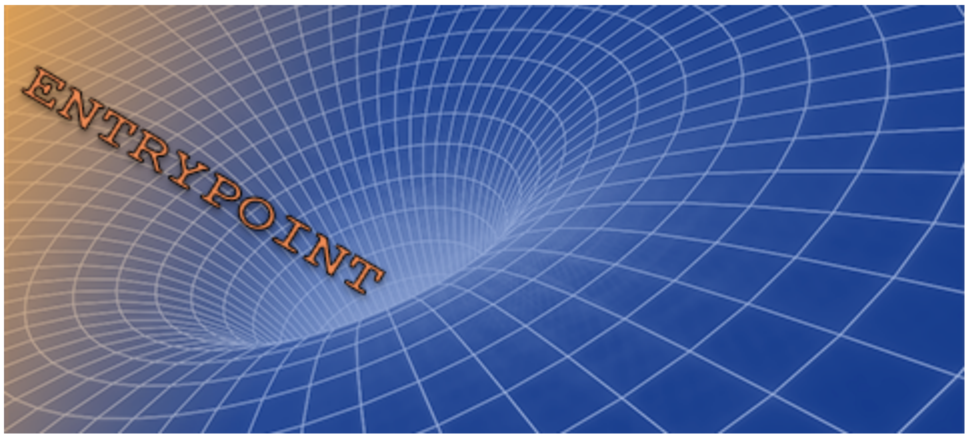
```
ENTRYPOINT ["fastqc"]
```

This means that as soon as the container start the program `fastqc` will be activated, and since its default is to run as GUI this will cause an error unless `X11` is available and running.

As we learned earlier, we’ll have to **bypass** the entry point, and then we’ll be able to work from within the container.

8.2 Bypass Entry Point

We can circumvent the entrypoint command from the Docker image by specifying an alternate command. If we use `/bin/bash` as we did before we'll be able to start the container and obtain an interactive terminal.



TASK:

Start container.

We bypass entry point and also specify shared directory as before:

```
docker run -it --rm  --entrypoint "/bin/bash" -v $HOME/dockershar
  e:/home pegi3s/fastqc
```

```
root@06d472dffb7c:/#
```

We can now start working, remembering that we are working within the container. We first change into the shared directory and verify that the compressed archived are present.

```
cd /home
ls *.gz
```

```
SRR10314868_1.fastq.gz  SRR10314868_2.fastq.gz
```

We can now run the program on one or more files. `fastqc` can read compressed files and therefore we do not need to uncompress them before use.

To run `fastqc` on all files the command ` `fastqc *.gz` would do. For today we'll run it on one of them:

```
fastqc SRR10314868_1.fastq.gz
```

```
Started analysis of SRR10314868_1.fastq.gz
Approx 5% complete for SRR10314868_1.fastq.gz
Approx 10% complete for SRR10314868_1.fastq.gz
[... truncated]
Approx 95% complete for SRR10314868_1.fastq.gz
Analysis complete for SRR10314868_1.fastq.gz
```

Two files will be created for each sequence file analyzed.

```
ls SRR10314868_1*fastqc*
```

```
SRR10314868_1_fastqc.html  SRR10314868_1_fastqc.zip
```

The information is the same in both and visible through a web browser. Only the HTML format and organization is different.

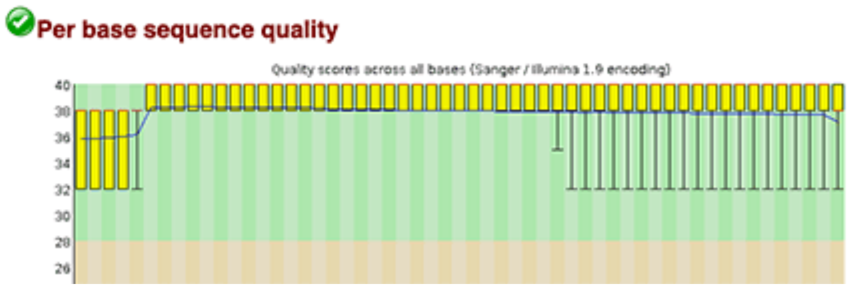
TASK:
Open file in web browser.

To see the results simply go to the host computer and open the file `SRR10314868_1_fastqc.html` with a web browser.

The top of the page will show a summary information table

Measure	Value
Filename	SRR10314868_1.fastq.gz
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	850953
Sequences flagged as poor quality	0
Sequence length	35-101
%GC	50

The quality score is reported in the form of a “box plot” that here show very high quality:



8.3 Exit container

If you are finished, exit the container and return to the host prompt:

```
exit
```

```
$
```

9 Summary of commands learned or reviewed

Docker Commands	Comment
`docker --version`	Short output of version
`docker login`	Required. Register at docker.com
`docker pull`	download a docker image from hub.docker.com
`tag`	some docker images require a specific tag
`docker image ls`	list docker image. Equiv command: `docker images`
`docker run -it --rm -v $HOME/dockershare:/data`	run shell in container, share `dockershare` directory
`docker container ls -a`	list all containers, same as command above
`docker run -it --rm  --entrypoint`	bypass ENTRYPOINT command
`/bin/bash` , `/bin/ls`	common “entrypoint” options

Shell Commands	Comment
`which`	shell command to find program location
`$HOME`	shell variable designated the default home folder
`cd $HOME/dockershare`	change directory to `dockershare` located in `$HOME`
`cat > GIP.fa  <<- EOF`	create a file from *stdin* until EOF
`gzip`	Compress file(s) with GNU zip format

Software within containers	Comment
`clustalo`	multiple sequence alignemnt of 3 or more sequences
`fastq-dump`	Extract archived sequence data into fastq files.
`fastqc`	check quality of sequence data.

10 APPENDIX A

10.1 Create shared directory

```
cd $HOME
mkdir dockershare
```

10.2 Create glucagon family files

Simply *Copy/Paste* the following code into the Terminal:

```
# This ensures that we go to dockershare directory
cd $HOME/dockershare

# Then we create fasta files one by one

cat  > glucagon.fa  <<- EOF
>glucagon
HSQGTFTSDYSKYLDSRRAQDFVQWLMNT
EOF

cat > GLP-1.fa  <<- EOF
>GLP-1
HAEGTFTSDVSSYLEGQAAKEFIAWLVKGRG
EOF

cat > GLP-2.fa  <<- EOF
>GLP-2
HADGSFSDEMNTILDNLAARDFINWLIQTKITD
EOF

cat > GIP.fa  <<- EOF
>GIP
YAEGTFISDYSIAMDKIRQQDFVNWLLAQ
EOF
```

10.3 Protein FASTA for clustalomega

Simply *Copy/Paste* the following code into the Terminal:

```
# This ensures that we go to dockershare directory
cd $HOME/dockershare

# Then we create fasta file sequences2.fa

cat  > sequences2.fasta  <<- EOF
>tr|O13169|O13169_CYPCA Alpha-globin OS=Cyprinus carpio GN=No.3 al
  pha PE=3 SV=1
MSLSDKDKAAVKALWAKISPKADDIGAEALGRMLTVYPQTKTYFAHWDDLSPGSGPVKKH
GKVIMGAVADAVSKIDDLVGGLASLSELHASKLRVDPANFKILAHNVIVVIGMLFPGDFP
PEVHMSVDKFFQNLALALSEKYR
>sp|P69905|HBA_HUMAN Hemoglobin subunit alpha OS=Homo sapiens GN=H
  BA1 PE=1 SV=2
MVLSPADKTNVKAAWGKVGAHAGEYGAEALERMFLSFPTTKTYFPHFDLSHGSAQVKGHG
KKVADALTNAVAHVDDMPNALSALSDLHAHKLRVDPVNFKLLSHCLLVTLAAHLPAEFTP
AVHASLDKFLASVSTVLTSKYR
>sp|P01942|HBA_MOUSE Hemoglobin subunit alpha OS=Mus musculus GN=H
  ba PE=1 SV=2
MVLSGEDKSNIKAAWGKIGGHGAEYGAEALERMFASFPTTKTYFPHFDVSHGSAQVKGHG
KKVADALASAAGHLDDLPGALSALSDLHAHKLRVDPVNFKLLSHCLLVTLASHHPADFTP
AVHASLDKFLASVSTVLTSKYR
EOF
```

Data from https://www.ebi.ac.uk/seqdb/confluence/display/JDSAT/Multiple+Sequence+Alignment+Tool+Input+Examples (https://www.ebi.ac.uk/seqdb/confluence/display/JDSAT/Multiple+Sequence+Alignment+Tool+Input+Examples)

11 APPENDIX B

11.1 FastQ format

Fastq files (`.fq`) are usually the “deliverable” data that is given to users, more often in a gnuzip (`gzip`) compressed form (`.fq.gz`) as `fastq` is a plain text format.

Here are the first 4 lines of one of the files we’ll unarchive today:

```
@SRR10314868.1.1 1 length=101
TGGGAGCAGATGGTTATGCCTTACACAGACAACCGCTGGAACGAAACCTATTATCGTTCAACGATG
ATTATTGGTCTGGCACCTGAAGGGAAGGTAAGAGT
+SRR10314868.1.1 1 length=101
GGGGGIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIGIIIIIIIGIIGIIIIIIIGGIGIIIGGGG
GGIIGIGGGIGIIIIIIIIIGGIIGIIIIIGIIIG
```

Fastq files contain 4 lines per recorded sequence:

- Line 1 begins with a `@` character and is followed by a sequence identifier and an optional description (*like a FASTA* e.g.* a title line).
- Line 2 is the raw sequence letters (the base calls: A, C, T, G and N).
- Line 3 begins with a `+` character as a separator
- Line 4 encodes the quality values for the sequence in Line 2, and must contain the same number of symbols as letters in the sequence.

The quality values are Phred scores +33 (if it is Illumina data, hence Phred+33) encoded using ASCII (http://drive5.com/usearch/manual/quality_score.html) characters to represent the numerical quality scores.

There have been a few encoding schemes detailed on the Wiki FastQ format (https://en.wikipedia.org/wiki/FASTQ_format) page (scroll down or see below.)

Note: *PHRED* was the name of a program developped by Philip Palmer Green (https://en.wikipedia.org/wiki/Philip_Palmer_Green) and acronym for "*Phil's read editor*" often used in conjunction with "*Phil's revised assembly program*" or *PHRAP* (Moody (2004).)

See also:

- https://en.wikipedia.org/wiki/FASTQ_format (https://en.wikipedia.org/wiki/FASTQ_format)
- https://support.illumina.com/bulletins/2016/04/fastq-files-explained.html (https://support.illumina.com/bulletins/2016/04/fastq-files-explained.html)
- https://help.basespace.illumina.com/articles/descriptive/fastq-files/ (https://help.basespace.illumina.com/articles/descriptive/fastq-files/)
- https://www.illumina.com/science/technology/next-generation-sequencing/plan-experiments/quality-scores.html (https://www.illumina.com/science/technology/next-generation-sequencing/plan-experiments/quality-scores.html)
- http://drive5.com/usearch/manual/quality_score.html (http://drive5.com/usearch/manual/quality_score.html)
- https://en.wikipedia.org/wiki/Phred_quality_score (https://en.wikipedia.org/wiki/Phred_quality_score)

11.1.1 Qualtity scores

The quality scores from the Wiki FastQ format (https://en.wikipedia.org/wiki/FASTQ_format) page is reproduced here (HTML version of this document only):

```
  SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS.......................
  ..........................XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  ...............................IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
  .................................JJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJ
  LLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL......................
  !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`ab
  |                         |    |        |
 33                        59   64       73
  0........................26...31.......40
                           -5....0........9.......................
                                 0........9.......................
                                    3.....9.......................
  0.2......................26...31........41

 S - Sanger        Phred+33,  raw reads typically (0, 40)
 X - Solexa        Solexa+64, raw reads typically (-5, 40)
 I - Illumina 1.3+ Phred+64,  raw reads typically (0, 40)
 J - Illumina 1.5+ Phred+64,  raw reads typically (3, 41)
     with 0=unused, 1=unused, 2=Read Segment Quality Control Indica
     (Note: See discussion above).
 L - Illumina 1.8+ Phred+33,  raw reads typically (0, 41)
```

12 APPENDIX C

Windows users may run into more difficulties depending on set-up and admin privileges. Docker with fancy variable commands will only run in PowerShell.

Here are useful links:

- PowerShell: Environment Variables: https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_environment_variables (https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_environment_variables)
- Get started with Docker for Windows: https://docs.docker.com/docker-for-windows/ (https://docs.docker.com/docker-for-windows/)
- Unable to share drives:
  - https://github.com/docker/for-win/issues/2946 (https://github.com/docker/for-win/issues/2946)
  - https://github.com/docker/for-win/issues/1352 (https://github.com/docker/for-win/issues/1352)

REFERENCES

Andrews, S. 2010. “FASTQC. A Quality Control Tool for High Throughput Sequence Data.” https://www.bioinformatics.babraham.ac.uk/projects/fastqc/ (https://www.bioinformatics.babraham.ac.uk/projects/fastqc/).

Leinonen, R., H. Sugawara, and M. Shumway. 2011. “The sequence read archive.” *Nucleic Acids Res.* 39 (Database issue): 19–21.

Moody, Glyn. 2004. *Digital Code of Life: How Bioinformatics Is Revolutionizing Science, Medici*. John Wiley & Sons.

Sievers, F., and D. G. Higgins. 2018. “Clustal Omega for making accurate alignments of many protein sequences.” *Protein Sci.* 27 (1): 135–45.

Sievers, F., A. Wilm, D. Dineen, T. J. Gibson, K. Karplus, W. Li, R. Lopez, et al. 2011. “Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega.” *Mol. Syst. Biol.* 7 (October): 539. https://www.ncbi.nlm.nih.gov/pubmed/21988835 (https://www.ncbi.nlm.nih.gov/pubmed/21988835).

---

1. https://docs.docker.com/install/ (https://docs.docker.com/install/)↩
2. http://pegi3s.github.io/dockerfiles (http://pegi3s.github.io/dockerfiles)↩
3. https://www.ebi.ac.uk/seqdb/confluence/display/THD/Clustal+Omega (https://www.ebi.ac.uk/seqdb/confluence/display/THD/Clustal+Omega)↩
4. http://www.clustal.org/omega/README (http://www.clustal.org/omega/README)↩
5. https://oprea.rocks/blog/how-to-properly-override-the-entrypoint-using-docker-run/ (https://oprea.rocks/blog/how-to-properly-override-the-entrypoint-using-docker-run/)↩
6. https://en.wikipedia.org/wiki/FASTQ_format (https://en.wikipedia.org /wiki/FASTQ_format)↩
7. https://trace.ncbi.nlm.nih.gov/Traces/sra/sra.cgi?view=toolkit_doc&f=fastq-dump (https://trace.ncbi.nlm.nih.gov/Traces/sra/sra.cgi?view=toolkit_doc&f=fastq-dump)↩
8. https://www.bioinformatics.babraham.ac.uk/projects/fastqc/ (https://www.bioinformatics.babraham.ac.uk/projects/fastqc/)↩