Docker - Beginner Biologist 2

Jean-Yves Sgro

2019

1 Learning objectives

- Select Docker containers from the docker hub
- Use a Docker container to accomplish tasks
- Review and use shared directories

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.)

1.1 Requirements

- Be familiar with Docker or follow workshop 1 "Docker - Beginner Biologist 1"
- 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 due time you will be able to create your own docker image. But for now we'll use images that are available on the Docker hub.

A docker image can contain a single useful software, or it can give access to a series of software. The more software the image contains the more disk space it is likely required. For example, the ORCA image (Jackman et al. (2019)) is close to 30Gb in size but contains over 600 the bioinformatics software and utilities.

3.1 EMBOSS

For this series of exercises we'll look for and use a docker image of the EMBOSS (Rice, Longden, and Bleasby (2000)) series of sequence analysis software.

"The European Molecular Biology Open Software Suite" (EMBOSS) is a free Open Source software analysis package specially developed for the needs of the molecular biology user community.[2]

EMBOSS contains a large number of sequence analysis tools, and we'll sample a few of them *via* a docker method.

The purpose of this tutorial is more about learning how to use a Docker container rather than learning EMBOSS itself. However, here are a few links for learning more about EMBOSS for reference:

EMBOSS	**Link**
Home page	http://emboss.sourceforge.net (http://emboss.sourceforge.net)
Tutorial	http://emboss.sourceforge.net/docs/emboss_tutorial/emboss_tutorial.html (http://emboss.sourceforge.net/docs/emboss_tutorial/emboss_tutorial.html)
Applications	http://emboss.sourceforge.net/apps/release/6.6/emboss/apps/index.html (http://emboss.sourceforge.net/apps/release/6.6/emboss/apps/index.html)
Grouped by functions	http://emboss.sourceforge.net/apps/release/6.6/emboss/apps/groups.html (http://emboss.sourceforge.net/apps/release/6.6/emboss/apps/groups.html)

3.2 EMBOSS on Docker

Any person with a Docker ID can create and upload images that are accessible to other users. Therefore we'll find a large number of Docker images available. However, they will not be constructed in the same way, may not contain the same version of the software, and might not have been updated in a long time. Therefore finding a suitable image might require some browsing before deciding which one(s) to download and test.

TASK:

Open we web browser.

- Go to https://hub.docker.com (https://hub.docker.com)
- Sign-in is optional
- In the top-left corner enter "EMBOSS" within the search field

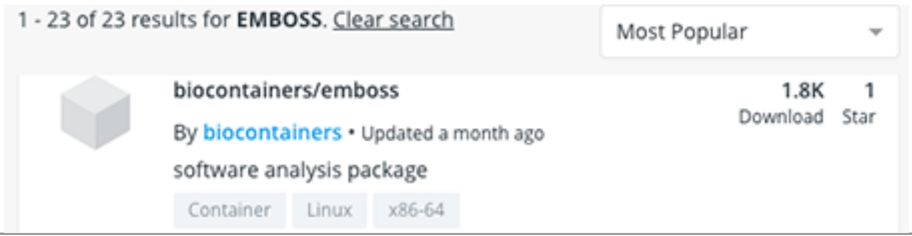
The resulting web page will show results. As of today (Oct 2019) there are **23** results.

How many results did you get?

____

The results are shown on a web page shown by “**Most Popular**” which may be the best option. The other option is “**Recently Updated**” which may be a better choice in certain cases.

For today we’ll chose the the first one named “**biocontainers/emboss**”


1 - 23 of 23 results for **EMBOSS**. Clear search

> *Important note*: the full name of the docker image is **biocontainers/emboss** containing 2 words. This complete name will need to be used later to activate it.

TASK:

Click on the biocontainers/emboss box.

The user “*biocontainer*” is a provider of a large number of other docker images and has well organized pages. Once you get on that page you’ll see that there are different tabs named:

- **Overview**: details about all biocontainers
- **Tags**: important tag (see below)
- **Dockerfile**: How the docker image was created (will be useful in future workshop)
- **Builds**: not used here- some images can be automatically updated (built)

In the next step we’ll want to `pull` (donwload from the hub) the docker image. On the default (Overview) tab you can see (on the right) the command that you can copy to `pull` the image onto your computer. However, if you were to do this now you’d have an error:

```
docker pull biocontainers/emboss
```

```
Using default tag: latest
Error response from daemon: manifest for biocontainers/emboss:latest
not found: manifest unknown: manifest unknown
```

The error is apparently due to the fact that `docker` cannot find `biocontainers/emboss:latest`.

Tags:

This means that we need to talk about **tags**. The default tag is **`latest`** and is not required by default, just *assumed* since it's the default. This is true most of the time, unless the author(s) of the image decide that they want to use a specific tag. In that case `latest` does not exist and the specific tag has to be clearly mentioned on the `pull` request.

For example, in the previous workshop we pulled the image for the small linux distribution called `alpine`. The command was simply `docker pull alpine`. Then, when we asked to show the list of images with the command `docker image ls alpine` we could note that `latest` was entered under the column `TAG`:

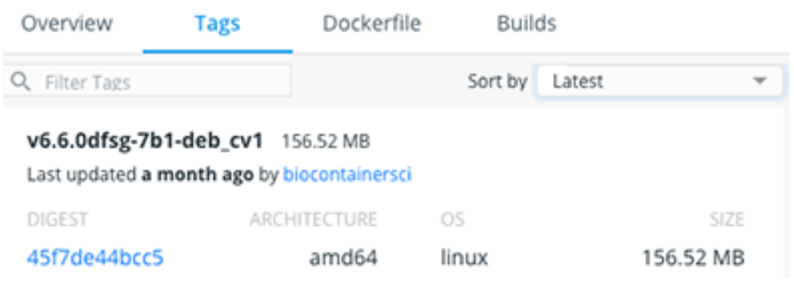
```
REPOSITORY     TAG           IMAGE ID        CREATED          SIZE
alpine         latest        11cd0b38bc3c    14 months ago    4.41MB
```

However, for the *biocontainers* images, it is necessary to use a specific tag which is listed under the `Tags` tab of the web page for the container.

TASK:

Click on the *Tags* tab of the biocontainers/emboss page.

You will note that by default the tags are shown sorted as "latest" (right hand side pull-down menu). As of today this looks like this:



v6.6.0dfsg-7b1-deb_cv1 156.52 MB
Last updated a month ago by biocontainersci

DIGEST	ARCHITECTURE	OS	SIZE
45f7de44bcc5	amd64	linux	156.52 MB

As of this writing the latest release of EMBOSS is 6.6.0[3] and the tag seems to reflect this within its first few characters `v6.6.0`.

> *Consequence*: The `pull` command ***must*** contain the complete specific tag.

TASK:

Pull biocontainers/emboss image.

From the above information it follows that the default `pull` command shown on the `Overview` Tag page ***will not work by default*** and a specific tag needs to be added to the request.

To that effect use the mouse to *Copy* the tag and add it to the pull request as shown below.

Note: The latest tag might change in the future and may be different than the one used below.

```
docker pull biocontainers/emboss:v6.6.0dfsg-7b1-deb_cv1
```

The tag will also need to be used later to activate the image (into a usable container.)

In a previous workshop we learned how to list docker images that are currently installed on the system. We can specifically list this one with the following command:

```
docker image ls biocontainers/emboss
```

```
REPOSITORY             TAG                      IMAGE ID        CREA
TED       SIZE
biocontainers/emboss   v6.6.0dfsg-7b1-deb_cv1   bc147a9dd825    5 we
eks ago   638MB
```

Note the `TAG` column.

4 EMBOSS container

Now that we have what seems to be an appropriate image with EMBOSS of the latest version, we can now activate the image and “**dive into it!**”

Reminder: To create a container from an image we use the command `docker run` that can also be altered by a number of modifiersIn the following command we’ll add the following modifiers as we have learned in the previous workshop:

- `-t` : “Allocate a pseudo-TTY” (*i.e.* a text terminal)
- `-i` : interactive
- `--rm` : “Automatically remove the container when it exits”
- see complete list with command `docker run --help`.

Finally remember that **the image tag is mandatory**, otherwise you’ll have an error that says:

```
Unable to find image 'biocontainers/emboss:latest' locally
docker: Error response from daemon: manifest for biocontainers/embos
s:latest not found: manifest unknown: manifest unknown.
See 'docker run --help'.
```

We’ll now explore the inside of the container…

TASK:

Run the folowing command and those that follow:.

```
docker run -it --rm biocontainers/emboss:v6.6.0dfsg-7b1-deb_cv1
```

```
biodocker@bb321bea813b:/data$
```

We are now looking within the container as indicated by the long prompt (with blue background in the HTML version of this document.) we can now explore the inside of the container (directories and software.) Later we'll restart the container again with a shared directory to actually accomplish some analysis. For now we'll explore the structure of the container.

We can now check where we are within the container system:

```
pwd
```

```
/data
```

We can also check if there are any files contained within theis directory:

```
ls
```

We conculde that the directory is empty. We can remember that fact as useful information to later share this directory when we restart.

4.1 Linux

The underlying structure of the container is a general Linux system.

Images (and therefore containers) can be created from various verions of Linux, such as the popular Ubuntu.

The word "Linux" is very generic and it may be useful to find out what is the actual version that is running within the container. In fact "Linux" should be called "GNU/Linux".

The next 3 commands will ask information about the Linux version running under the hood. They are here for information but are not critical to using the container or the EMBOSS software.

The command `uname -a` shows all the information it knows about the system:

```
uname -a
```

```
Linux fdd1a7fa9677 4.9.184-linuxkit #1 SMP Tue Jul 2 22:58:16 UTC 20
19 x86_64 GNU/Linux
```

This tells us that we are running an Intel compatible (`x86`) 64bit system based on *linuxkit* "*a toolkit for building custom minimal, immutable Linux distributions*."[4]

The next 2 commands show that the type of Linux is based on one of the 2 major families called *Debian* (the other is part of the "*red hat*" family.)

```
cat /etc/issue
```

```
Debian GNU/Linux 10 \n \l
```

```
cat /etc/os-release
```

```
PRETTY_NAME="Debian GNU/Linux 10 (buster)"
NAME="Debian GNU/Linux"
VERSION_ID="10"
VERSION="10 (buster)"
VERSION_CODENAME=buster
ID=debian
HOME_URL="https://www.debian.org/"
SUPPORT_URL="https://www.debian.org/support"
BUG_REPORT_URL="https://bugs.debian.org/"
```

4.2 User

One question that arises on occasion is “who is the user” of the container. Some containers are by default running as “root” *i.e.* administrator level. We have a hint here that we are not running as “root” because the prompt ends with a `$` sign, while the “root” user would show a hash/pound sign `#`.

We can ask what is the username with the command:

```
whoami
```

```
biodocker
```

We can also ask what is the defaul shell running:

```
echo $SHELL
```

```
/bin/bash
```

Therefore we are logged in as user `biodocker` running the `bash` shell and that is all good.

4.3 Where is EMBOSS?

EMBOSS consists in a series of software for the analysis of protein or nucleic acid DNA and RNA sequences (but not Next Gen sequencing.)

We can figure out where the software is located with a few commands, just knowing the name of at least one of the programs. For example, the program `needle` is an implementation of the Needleman-Wunsch global alignment of two sequences (Needleman and Wunsch (1970).)

The `bash` command `which` shows the location of a given software:

```
which needle
```

```
/usr/bin/needle
```

Then we can list the location with a long list:

```
ls -l /usr/bin/needle
```

```
lrwxrwxrwx 1 root root 20 Jan 28  2019 /usr/bin/needle -> ../lib/emb
oss/needle
```

This tells us that the program is located in another directory (`->` `..` indicated a "symbolic link," sometimes known as "shortcut") and we can list the complete directory with the understanding that `..` represents the parent directory. Therefore, we can first list the `emboss` directory found in `/usr/lib` a suggested by the symbolic link:

```
ls /usr/lib/
```

```
apt  dpkg  emboss  gcc  locale  mime  os-release  sasl2  tc  tmpfile
s.d  x86_64-linux-gnu
```

We can now list the entire `emboss` directory with:

```
ls /usr/lib/emboss
```

Here are the first few lines for the 261 programs that are included in this version:

```
aaindexextract    drfindformat    megamerger       seqret
abiview           drfindid        merger           seqretsetall
acdc              drfindresource  msbar            seqretsplit
acdgalaxy         drget           mwcontam         seqxref
acdlog            drtext          mwfilter         seqxrefget
...
...
```

Within the list you should be able to spot the `needle` program.

TASK:
Terminate the container.

```
exit
```

This will echo the word `exit` and return us to the host computer prompt.

```
$
```

5 Set-up EMBOSS container with shared data folder

In the previous workshop we learned various methods for using software from a container. The easiest is to work *within* the container after a folder from the host computer has been shared in order to share and save data files more easily.

Therefore we'll restart the container with a shared folder named *e.g.* `dockershare`.

5.1 Create dockershare shared folder

- If you created `dockershare` during the last workshop you can skip to the next task.
- If you need to create the `dockershare` directory follow these commands:

1. go back to your *home* directory simply by typing `cd`
2. verify that you are in the *home* directory
3. create the directoruy with command `mkdir dockershare`
4. verify that it has been successfully created with `ls -d` to list only directories

```
cd
pwd
mkdir dockershare
ls -d dockershare
```

```
/Users/jsgro
dockershare
```

The location of the `dockershare` directory will reflect YOUR username on the computer you are using.

A useful generic method to designate this directory is `$HOME/dockershare`. This has the advantage of using a *variable* (`$HOME`) that does not implicitly use your name. Therefore it can be used within a script, or simply as a *Copy/Paste* method. We'll use this in the command below.

5.2 Add sequences to dockershare dir

We can now populate the directory with a few sequence that we can use later with the EMBOSS program `needle` as a test example.

We'll use very short peptide sequences to keep the output simple: the glucagon family.

Glucagon is the principal hyperglycemic hormone, and acts as a counterbalancing hormone to insulin. Glucagon is a peptide hormone of 29 amino acids that shares the same precursor molecule, proglucagon, with GLP-1 and GLP-2. By tissue-specific posttranslational processing, glucagon is secreted from pancreatic α cells whereas GLP-1 and GLP-2 are secreted from intestinal L cells. All these peptides have considerable sequence similarity (Park (2015).)

GIP, a related member of the glucagon peptide superfamily, also regulates nutrient disposal via stimulation of insulin secretion (Brubaker and Drucker (2002).)

We will now add the sequences for these small peptides with a `bash` "trick". In brief we send the sequence text to a file which closes when the "end-of-file" signal (`EOF`) is seen. The file format is *fasta* which requires that the first line is `>` immediately followed by a sequence name.

Sequences are taken from Brubaker and Drucker (2002) Table 1[5] (page 180.)

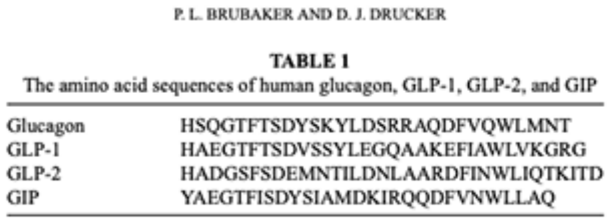

P. L. BRUBAKER AND D. J. DRUCKER

TABLE 1
The amino acid sequences of human glucagon, GLP-1, GLP-2, and GIP

Glucagon	HSQGTFTSDYSKYLDSRRAQDFVQWLMNT
GLP-1	HAEGTFTSDVSSYLEGQAAKEFIAWLVKGRG
GLP-2	HADGSFSDEMNTILDNLAARDFINWLIQTKITD
GIP	YAEGTFISDYSIAMDKIRQQDFVNWLLAQ

Simply *Copy/Paste* the following code, that starts by getting into the `dockershare` directory.

(*Note*: the files are also available for download - short URL: tiny.cc/docker01)

```
# 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
```

We can verify that the files have been created.

```
ls
```

```
GIP.fa      GLP-1.fa    GLP-2.fa    glucagon.fa
```

We can also type one of sequences the screen to verify that the format and content are as expected:

```
cat glucagon.fa
```

```
>glucagon
HSQGTFTSDYSKYLDSRRAQDFVQWLMNT
```

To see them all on screen simply use `cat *.fa` instead.

5.3 Restart container with shared directory

We are now ready to restart the container and modify the previous `docker run` command so that we can access the shared directory.

We are now going to "dive into" the container.

Remember that we noted above that the directory `/data` within the container is itself empty. We can therefore safely use this directory to "map" the `dockershare` directory into the container.

TASK:
Restart the container with shared folder.

The command below adds `-v` to map the `dockershare` directory on the host computer to the `data` directory on the container.

```
docker run -it --rm -v $HOME/dockershare:/data biocontainers/emboss:
        v6.6.0dfsg-7b1-deb_cv1
```

```
biodocker@bb321bea813b:/data$
```

We can now verify that indeed, we can see and use the glucagon family sequence files that were just created:

```
ls
```

```
GIP.fa  GLP-1.fa  GLP-2.fa  glucagon.fa
```

Success! We can now use the EMBOSS programs! Since this is a shared directory, results will be saved on the host computer even when we terminate the container.

5.4 Using EMBOSS within container

Now that we have a shared directory and access to EMBOSS programs we can use the programs and save the results.

5.5 Needle global alignment

We can use the EMBOSS the program `needle` (global alignment of two sequences, Needleman and Wunsch (1970)) as a first test.

From the EMBOSS documentation[6]: *`needle` reads two input sequences and writes their optimal global sequence alignment to file. It uses the Needleman-Wunsch alignment algorithm to find the optimum alignment (including gaps) of two sequences along their entire length.*

As an example we can align glucagon and GLP-1.

```
needle glucagon.fa GLP-1.fa
```

Simply press `return` or `Enter` 3 times to keep the proposed default for the gap penalty and extension, and to use the suggested name for the output file:

```
Needleman-Wunsch global alignment of two sequences
Gap opening penalty [10.0]:
Gap extension penalty [0.5]:
Output alignment [glucagon.needle]:
```

We can now visualize the output file on the screen:

```
cat glucagon.needle
```

```
########################################
# Program: needle
# Rundate: Tue 22 Oct 2019 16:38:08
# Commandline: needle
#    [-asequence] glucagon.fa
#    [-bsequence] GLP-1.fa
# Align_format: srspair
# Report_file: glucagon.needle
########################################

#=======================================
#
# Aligned_sequences: 2
# 1: glucagon
# 2: GLP-1
# Matrix: EBLOSUM62
# Gap_penalty: 10.0
# Extend_penalty: 0.5
#
# Length: 31
# Identity:      14/31 (45.2%)
# Similarity:    22/31 (71.0%)
# Gaps:           2/31 ( 6.5%)
# Score: 88.0
#
#
#=======================================

glucagon           1 HSQGTFTSDYSKYLDSRRAQDFVQWLMNT--     29
                     |::||||||.|.||:.:.|::|:.||:..
GLP-1              1 HAEGTFTSDVSSYLEGQAAKEFIAWLVKGRG     31


#---------------------------------------
#---------------------------------------
```

Note: all the mandatory parameters can also be given on the command line at once, therefore removing the need to confirm at each step:

```
needle glucagon.fa GLP-1.fa -outfil glucagon.needle -gapopen 10 -gap
extend 0.5
```

5.6 Programs with graphics

Some EMBOSS programs can output graphical output but as the container is currently set-up we can only access the created files only from the host computer side.

Here are 2 very short examples that illustrate this:

- `pepwheel` : Draw a helical wheel diagram for a protein sequence
- `pepinfo` : Plot amino acid properties of a protein sequence in parallel

The default graphic output in EMBOSS is an *interactive* window called `x11` that we'll change to *e.g.* `PNG` to save the results into graphics file(s) instead.

5.6.1 pepwheel

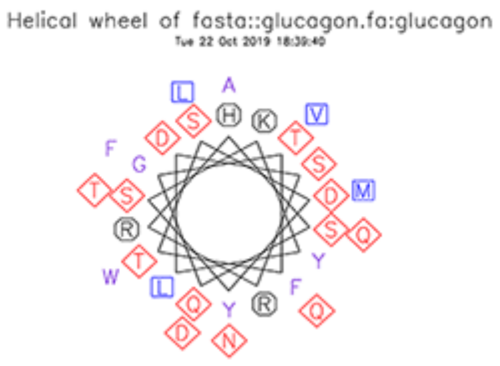
From the EMBOSS documentation[7]: *`pepwheel` draws a helical wheel diagram for a protein sequence. This displays the sequence in a helical representation as if looking down the axis of the helix. It is useful for highlighting amphipathicity and other properties of residues around a helix. By default, aliphatic residues are marked with squares, hydrophilic residues are marked with diamonds, and positively charged residues with octagons, although this can be changed.*

```
pepwheel glucagon.fa
```

Now change the output default from `x11` to `png` or you can try some other useful format: `gif` , `pdf` , `svg` .

```
Draw a helical wheel diagram for a protein sequence
Graph type [x11]: png
Created pepwheel.1.png
```

Since this file is a graphics file, we cannot look at it from within the container. However, it is simple to access the file from the graphical interface of the host computer!



5.6.2 pepinfo

From the EMBOSS documentation[8]: *`pepinfo` plots various amino acid properties in parallel for an input protein sequence. The types of plot available are i. Hydrophobicity plots using the method of Kyte & Doolittle, the optimal matching hydrophobicity scale*

(OHM) of Sweet & Eisenberg, or consensus parameters (Eisenberg et al). ii. Histogram of the presence of residues with the physico-chemical properties: Tiny, Small, Aliphatic, Aromatic, Non-polar, Polar, Charged, Positive, Negative. The data are also written out to an output file.

```
pepinfo glucagon.fa
```

Now change the output default from `X11` to `pdf` or you can try some other useful format: `gif`, `png`, `svg`.

```
Plot amino acid properties of a protein sequence in parallel.
Graph type [x11]: pdf
Output file [glucagon.pepinfo]:
Created pepinfo.pdf
```

Results: you can check the content of the text ouput written in `glucagon.pepinfo` with the `more` command that will show one screen at a time. Then press `space bar` for the next screen and `q` to end and return to the prompt.

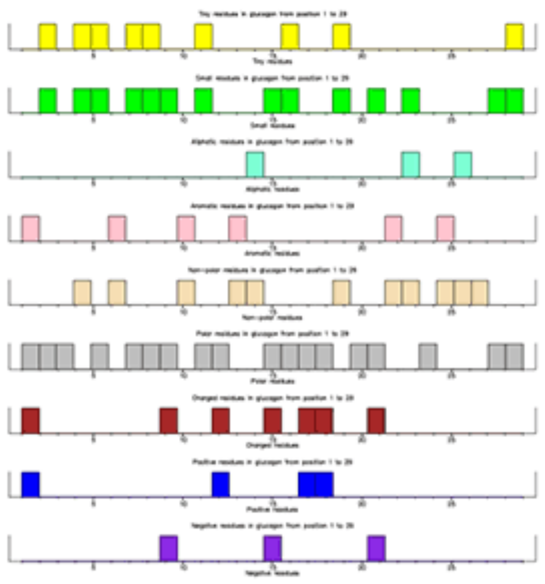
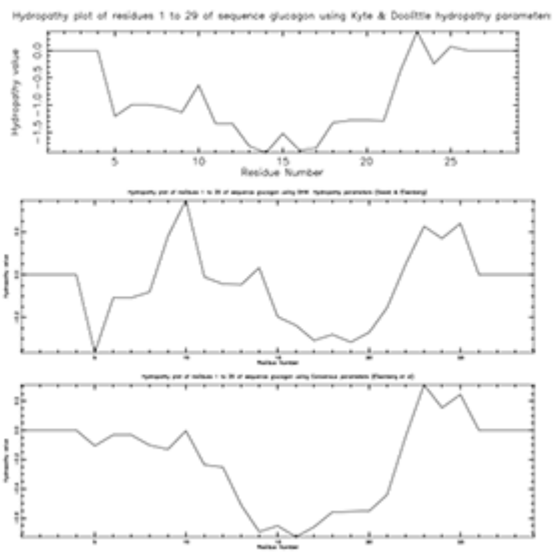
```
more glucagon.pepinfo
```

```
Printing out Tiny residues in glucagon from position 1 to 29

Position  Residue           Result
       1       H                             0
       2       S                             1
       3       Q                             0
       4       G                             1
       5       T                             1
       6       F                             0
[...]
```

You can also look at the image file(s) from the host computer side finding then within the `dockershare` directory.

`pepinfo` graphics output 1 **`pepinfo` graphics output 2**

`pepinfo` graphics output 1	**`pepinfo` graphics output 2**
	

5.7 Limitations to this EMBOSS container

This particular container has specific limitations, that might also extend to other EMBOSS containers.

- There are no databases. These are very large and therefore not present.
- Some EMBOSS programs depend on other programs that are not installed: example is `emma` for multiple sequence analysis that is an EMBOSS wrapper for the `clustalX` program which is not installed. Therefore `emma` is not functional.
- There are no manual documentations. This is not very critical since the documentation exist in many copies on the Internet.

However, it is possible to get the minimal help by adding `-h` after an EMBOSS program, for example `needle -h`.

6 EMBOSS from outside container

We have just used the EMBOSS programs from wihtin the container while sharing a directory with the host computer. In this way it is easy to call each EMBOSS program simply by name directly: `needle`, `pepwheel`, etc.

However, as we have alluded in the previous workshop, we can also call each of the program from the host computer itself providing we give the proper container information. This may be useful in cases where only one or two programs need to be access while working on a project.

An analogy for using a container from the outside could be using a remote control to control actions within a black-box…

As an example we'll create a new alignment from sequences located in the `dockershare` directory without entering the docker container itself. (Hence, on the HTML version of this document the background will be green -on the host- rather than blue - within container.)

However, it is still necessary to specify the shared directory on the command line as this is the only way that the programs within the container (here `neele`) can "see" the files that are on the host computer.

The sequences are located within the `dockershare` directory which is mapped to `/data` within the container. As such we actually do not need to be within `dockershare` on the host computer terminal for these commands to work. However, it is still useful to place the focus on the shared directory to quickly explore any output result:

```
cd $HOME/dockershare
```

Then we run the EMBOSS `needle` program by activating the container. When the `needle` program is finished, the container will exit and be removed (`--rm` .)

```
 docker run -it --rm -v$HOME/dockershare:/data  biocontainers/embos
       s:v6.6.0dfsg-7b1-deb_cv1 needle
```

The name of the sequence have to be typed, while the default options on the next line can simply be accepted by pressing return.

```
Input sequence: GLP-1.fa
Second sequence(s): GLP-2.fa
Gap opening penalty [10.0]:
Gap extension penalty [0.5]:
Output alignment [glp-1.needle]:
```

Alternatively the command could also be written as a single line:

```
docker run -it --rm -v$HOME/dockershare:/data  biocontainers/emboss:
        v6.6.0dfsg-7b1-deb_cv1 needle GLP-1.fa GLP-2.fa -outfil glp-
        1.needle -gapopen 10 -gapextend 0.5
```

```
Needleman-Wunsch global alignment of two sequences
```

Note that for clarity we can use the “line continuation symbol” `\` to break the long line into more readable clode:

```
docker run -it --rm -v$HOME/dockershare:/data  \
biocontainers/emboss:v6.6.0dfsg-7b1-deb_cv1 needle \
GLP-1.fa GLP-2.fa -outfil glp-1.needle -gapopen 10 -gapextend 0.5
```

We can check that the alignment worked by looking at the last few lines of the output file with the `ail` command:

```
tail glp-1.needle
```

```
#
#=======================================


GLP-1              1 HAEGTFTSDVSSYLEGQAAKEFIAWLVKGRG--      31
                     ||:|:|:.::::.|:..||::||.||::.:.
GLP-2              1 HADGSFSDEMNTILDNLAARDFINWLIQTKITD      33



#---------------------------------------
#---------------------------------------
```

7 Docker containers

Before we finish we want to make sure that no contaier is in halted mode (*e.g.* when `--rm` is forgotten) as these can accumulate over time and use a large amount of space.

```
docker container ls -a
```

```
CONTAINER ID   IMAGE   COMMAND   CREATED   STATUS     PORTS   NAMES
```

If you have any halted container delete them using the `CONTAINER ID` name with *e.g.* changing the value below to that of your container(s).

```
docker rm 461346b98938
```

8 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

Shell Commands	Comment
`cat`	print entire file on screen
`cat > filename`	send input to `filename`
`more`	print text file content on screen. `space bar` for 1 more screen. `q` to quit.
`uname -a`, `cat /etc/issue`, `cat /etc/os-release`	shell commands to view Linux version
`which`	shell command to find program location

Shell Commands	Comment
`$HOME`	shell variable designated the default home folder
`cd $HOME/dockershare`	change directory to `dockershare` located in `$HOME`

EMBOSS Commands	Comment
`needle`	global pairwise alignemnt of 2 sequences
`pepwheel`	draws a helical wheel diagram for a protein sequence.
`pepinfo`	plots various amino acid properties in parallel.

REFERENCES

Brubaker, P. L., and D. J. Drucker. 2002. “Structure-function of the glucagon receptor family of G protein-coupled receptors: the glucagon, GIP, GLP-1, and GLP-2 receptors.” *Recept. Channels* 8 (3-4): 179–88. https://doi.org/10.3109/10606820213687 (https://doi.org/10.3109/10606820213687).

Jackman, S. D., T. Mozgacheva, S. Chen, B. O’Huiginn, L. Bailey, I. Birol, and S. J. M. Jones. 2019. “ORCA: A Comprehensive Bioinformatics Container Environment for Education and Research.” *Bioinformatics*, April. https://doi.org/10.1093/bioinformatics/btz278 (https://doi.org/10.1093/bioinformatics/btz278).

Needleman, S. B., and C. D. Wunsch. 1970. “A general method applicable to the search for similarities in the amino acid sequence of two proteins.” *J. Mol. Biol.* 48 (3): 443–53. https://doi.org/10.1016/0022-2836(70)90057-4 (https://doi.org/10.1016/0022-2836(70)90057-4).

Park, Min Kyun. 2015. “Subchapter 17A - Glucagon.” In *Handbook of Hormones: Comparative Endocrinology for Basic and Clinical Research*, 129–31. Academic Press. https://doi.org/10.1016/B978-0-12-801028-0.00138-0 (https://doi.org/10.1016/B978-0-12-801028-0.00138-0).

Rice, P., I. Longden, and A. Bleasby. 2000. “EMBOSS: the European Molecular Biology Open Software Suite.” *Trends Genet.* 16 (6): 276–77. https://doi.org/10.1016/S0168-9525(00)02024-2 (https://doi.org/10.1016/S0168-9525(00)02024-2).

1. https://docs.docker.com/install/ (https://docs.docker.com/install/)↩
2. http://emboss.sourceforge.net (http://emboss.sourceforge.net)↩

3. http://emboss.sourceforge.net/apps/ (http://emboss.sourceforge.net/apps/)↩
4. https://github.com/linuxkit/linuxkit (https://github.com/linuxkit/linuxkit)↩
5. http://www.glucagon.com/pdfs/Receptors%20and%20Channels.pdf (http://www.glucagon.com/pdfs/Receptors%20and%20Channels.pdf)↩
6. http://emboss.sourceforge.net/apps/release/6.6/emboss/apps/needle.html (http://emboss.sourceforge.net/apps/release/6.6/emboss/apps/needle.html)↩
7. http://emboss.sourceforge.net/apps/release/6.6/emboss/apps/pepwheel.html (http://emboss.sourceforge.net/apps/release/6.6/emboss/apps/pepwheel.html)↩
8. http://emboss.sourceforge.net/apps/release/6.6/emboss/apps/pepinfo.html (http://emboss.sourceforge.net/apps/release/6.6/emboss/apps/pepinfo.html)↩