HPC use notes

Communicating with the cluster and managing your remote machine

Use SSH: from a shell window type …

- `ssh -l username login.hpc.imperial.ac.uk`
- You will be asked for your standard cluster password
- Now it's as though you were sitting with a shell open at the login node
- `ls` (will list the files in $HOME)
- `mkdir foldername` (make a new folder)
- `mv filename $HOME/foldername` (move a file to another place)
- `cd foldername` (change directory)
- `cat filename` (see your file to check its contents)

File transfer between local and remote machines

Use SFTP: from the directory of your code in a shell window type …

- `sftp username@login.hpc.imperial.ac.uk`
- You will be asked for your standard cluster password
- `put filename.R` (will copy filename.R from your current **local** working directory to your current **remote** working directory)
- `get filename.R` (will copy filename.R from your current **remote** working directory to your current **local** working directory)
- `exit`

Or use SCP: from a shell window type …

- `scp path/to/file.txt username@login.hpc.imperial.ac.uk:/home/username/`

  This works in a similar way to the cp command in standard directory management is run directly from the terminal (rather than logging in and then using "put" and "get").

Running a job on the cluster

1. Make a script which performs the desired tasks. This should be a .R or .py file which takes in job number and uses that to determine which task to perform. (See HPC_script.R, HPC_script_sources_file.R, and HPC_script_python.py for examples). When sourced this file must also save the outputs to the current working directory, and all file names should be distinct (otherwise the different jobs will overwrite each other). Ensure that you use an absolute file path when you source any .R file; e.g.:

   ```
   source("/rds/general/user/abc123/home/filename_to_source.R")
   ```

2. (Using SFTP or SCP from the terminal of <u>your local machine</u>): Copy your task script to an area on the remote machine.
   - E.g., $HOME/run_files/HPC_script.R

3. (Using SSH to access the remote machine): To write a .sh file to provide the cluster with instructions to run the task script. This will reference your task script so you will need to be mindful of the relative file path. If you wish, you can include instructions within this file to move all the output files from the cluster run into a specific folder on the remote machine. (See run_script.sh and run_script_python.sh for examples).
    - E.g., $HOME/run_files/run_script.sh
    - From the terminal you can write this directly to file by typing:
      `cat > run_script.sh`
      and then typing out the desired code line-by-line.

4. (Using SSH to access the remote machine): From the terminal, tell the cluster to run the jobs, and how many to run.
    - `qsub -J 1-32 run_script.sh` (tells the cluster to do 32 runs based on the .sh file – the job number will be from 1 to 32)
    - `qstat` (tells you the job ID and tells you the current status)
    - `qdel jobID[]` (deletes/cancels the job – insert the correct job ID in place of "`jobID`" based on what `qstat` tells you)

5. (Using SSH to access the remote machine): To check that your job is running correctly: Wait 5-10 minutes then check that nothing has gone wrong.
    - `qstat` (is your job running still?)
    - `ls` (are output files as expected?)
    - `cat filename.sh.ejob-id.index` (are error files empty?)
    - `cat filename.sh.ojob-id.index`
      (are standard output files as expected?)
    - `qstat` (is your job running still?)
    - `exit` (you're done for now; come back later)

6. (Using SSH to access the remote machine): Once the entire job is definitely complete (check using qstat), you will want to get the files back onto your local machine.
    - `ls` (output files as expected?)
    - `cat output filename` (contents as expected?)
    - `cat filename.sh.ejob-id.index` (error files empty?)
    - `cat filename.sh.ojob-id.index` (standard output files as expected?)
    - `tar czvf filename.tgz *`
    - `mv filename.tgz $HOME`
    - `exit`

    (Using SFTP, from a new directory on your own computer of where you want the results to be):
    - `sftp username@login.cx1.hpc.ic.ac.uk`
    - You will be asked for your standard cluster password
    - `get filename.tgz`
    - `exit`
      Your results are now all on your own computer
    - `tar xzvf filename.tgz`
      Your results are now complete uncompressed and ready for use

Advanced options

- Rsync for uploads and downloads

For mass uploads and downloads, it may be good to switch over from SCP to using the rsync command instead. Whilst these nominally do the same job, rsync will (with the right flags) check whether files have changed and only transfer files which require updating. If we wished to use rsync to copy the whole of our home directory to the current local directory, we would need to use:

```
rsync -chavzP --stats <USERNAME>@login.
hpc.imperial.ac.uk:/home/<USERNAME>/ .
```

A distinct downside of rsync is in its reliance on arcane flags to modify behaviour. This can lead to unexpected results, even with only minor spelling mistakes.

- Mounting your partition as a network drive

There is a possibility for mounting your HPC partition as a network drive, using the mount command.

```
sudo apt install cifs-utils
cd /media/
mkdir HPC
sudo mount -t cifs -o
username=<USERNAME>//rds.imperial.ac.uk/rds/user/<USERNAME> HPC
```

Note: this might take a while to work, or it may not work at all. In the end it's likely not worth worrying about greatly, but if it works it can make things that little bit easier.

- Checking jobs on the HPC without logging on

Checking the progress of jobs on the HPC can be done using the qstat command on the HPC bash terminal. This is a little annoying however as it's a good few keystrokes to connect, enter your password, enter qstat then disconnect. The process also fills your screen with a lot of unhelpful text, e.g., the logon banner text. With the following command, we can pass a call to qstat through our secure shell:

```
ssh <USERNAME>@login.hpc.imperial.ac.uk "(/opt/pbs/bin/qstat)"
```

- Authorising our SSH key (so no password is needed)

We can also reduce the amount of typing we need to do even further by adding our SSH keys to the `/home/<USERNAME>/.ssh/authorized_keys` file, removing the need to enter a password. Here is a useful tutorial for this:

https://www.ssh.com/academy/ssh/copy-id

Many of the operations described in this document can be scripted as either aliases or little shell tools to save you time going forwards.