**Internet 🡪 Username: ffarahan Pass: Fall2022!**

https://accounts.jhsph.edu

**/dcl01/smart/data/UKBiobank/rs\_fMRI/4991388\_20227\_2\_0/fMRI/rfMRI.nii.gz**

**/dcl01/smart/data/UKBiobank/rs\_fMRI/4991388\_20227\_2\_0/fMRI/filtered\_func\_data\_clean\_MNI.nii.gz**

**Logging in (from Terminal):**

ssh [ffarahan@jhpce01.jhsph.edu](mailto:ffarahan@jhpce01.jhsph.edu)

(or [ffarahan@jhpce02.jhsph.edu](mailto:ffarahan@jhpce02.jhsph.edu))

graphicial interface: ssh -x [ffarahan@jhpce01.jhsph.edu](mailto:ffarahan@jhpce01.jhsph.edu) (or [ffarahan@jhpce02.jhsph.edu](mailto:ffarahan@jhpce02.jhsph.edu))

Verification code: [Google Authenticator]

Password: Fv.1369.Fv 🡪 Shell prompt

Logging out: exit

**General commands (navigating + looking):**

* ls ls -a ls -l ls -al
* pwd
* cd
* . ..
* man
* date
* echo
* hostname
* sleep
* control-C
* more
* nano
* cat
* vi/emacs
* rm 🡪 for directory: with r
* cp (or mv) -r original\_dir new\_dir 🡪 for file: w/o r
* ls | wc -l -> count files in a directory using wc

**Running programs on the cluster:**

* qsub - allows you to submit a batch job to the cluster
* qrsh - allows you establish an interactive session
* qstat - shows information about running jobs
* qacct - shows information about completed jobs
* qdel ID - delete job
* qhost - list of machines (nodes)

**The common abbreviations:**

* r: the job is running
* t: the job is being transferred to a cluster node.  Some jobs may remain in the "t" state during hte duration of their run.
* qw: the job is queued (and not running yet)
* Eqw: an error occurred with the job.  You will likely need to kill the job and verify that the job parameters are correct before resubmitting.  If you are certain the parameters are correct, please email bitsupport, and someone can clear the error state.

Note:

* The **qsub** command starts batch jobs at a later time (bash job). The **qrsh** command runs jobs interactively (interactive job).
* Never run a job on the login node!
* Always use "**qsub**" or **qrsh**" to make use of the compute nodes
* If you are going to be compiling programs, do so on a compute node via **qrsh**
* Even something as simple as copying large files should be done via **qrsh**

**Submitting a batch job (qsub):**

cd class-scripts

qsub -cwd script1

qstat (qstat -j ID)

* -cwd 🡪 To run the job in the current working directory (where qsub was executed) rather than the default (home directory)
* -o path/filename (-e path/filename) 🡪 To send standard output (error) stream to a different file or directory
* -m e -M john@jhu.edu 🡪 To receive notification via email when your job completes

You might use “-m a” instead of “-m e” to send you an email when the job is aborted

* qsub -l h\_fsize=50G myscript.sh 🡪 files larger than 10GB
* options to qsub in 2 ways:
  + command line (e.g., qsub -cwd -l h\_fsize=100G -m e -M john@jhu.edu script1.sh)
  + Setting them in the batch job script (lines that start with “#$”)
    - qsub script1-resource-request.sh
* Requesting additional RAM:
  + **qsub** -l mem\_free=10G,h\_vmem=10G job1.sh
  + **qrsh** -l mem\_free=10G,h\_vmem=10G
* Requesting additional cores:
  + A job: qsub -cwd -pe local 4 myscript.sh
  + A qrsh session: qrsh -pe local 6
* Example: A job which will use 10 cores and need 120GB of RAM
  + qsub -cwd -pe local 10 -l mem\_free=12G,h\_vmem=12G myscript2.sh

**Run using an interactive session:**

qrsh

**Modules (first, qrsh):**

* module list [e.g., type R; q() for exit]
* module avail
* module load
* module unload

**Example:**

cd [where you want in remote directory]

1. Submit script file

qsub -cwd test1.sh

1. Run python commands interactively

qrsh -l gpu,mem\_free=300G,h\_vmem=300G,h\_fsize=300G

qrsh -l gpu,mem\_free=200G,h\_vmem=200G,h\_fsize=200G

qrsh -l gpu,mem\_free=100G,h\_vmem=100G,h\_fsize=100G

qrsh -l mem\_free=50G,h\_vmem=50G,h\_fsize=50G

qrsh -l mem\_free=100G,h\_vmem=100G,h\_fsize=100G ~~-pe local 5~~

module load conda

conda activate py3

python

qrsh -l mem\_free=30G,h\_vmem=31G

module load matlab

./batchname.sh

history | grep qrsh

qrsh -l gpu,mem\_free=50G,h\_vmem=50G,h\_fsize=50G

nvidia-smi (this is to see if any GPUs are in use)

conda create --name py2 python==2.7.9

conda create --name py3 python

conda activate py2

conda deactivate

conda env list

pip install --user package\_name

-----------------------------------------

Create new environment anaconda

-----------------------------------------

conda create -n hypertools python=3.7

conda activate hypertools

﻿!pip install nibabel nilearn sklearn hypertools

conda **env remove** --name myenv

There is one more issue in your test.sh.  You request gpu with 1G RAM. This is too lower. You will need more memory. Also, accessing GPU node is a little different. We only have 1 node with 3 GPUs, to access it, you need do the following

1. We only have 1 node with 3 GPU,qrsh -l gpu,mem\_free=50G,h\_vmem=50G
2. module load conda
3. source activate tensorflow-gpu-2.0
4. nvidia-smi   (this is to see if any GPUs are in use)
5. export CUDA\_VISIBLE\_DEVICES=0    or    export CUDA\_VISIBLE\_DEVICES=1   (to pick an unused GPU)
6. At this point you should be able to run R and Python programs that use tensorflow, and they will use the GPU you selected.

/dcl01/smart/data/hpc900/disk1/﻿116120/MNINonLinear/Results/rfMRI\_REST1\_LR/

**Transferring files to the cluster (get & put)**

<https://www.digitalocean.com/community/tutorials/how-to-use-sftp-to-securely-transfer-files-with-a-remote-server>

sftp [ffarahan@jhpce-transfer01.jhsph.edu](mailto:ffarahan@jhpce-transfer01.jhsph.edu)

pass: Fv.1369.Fv

cd /dcl01/smart/data/hpc900/sub-disk1]$ ls

100206 100307 100408 100610 101006 101107 101309 101410 101915 102008 102311 102513 102816 103111 103414 103515 103818 104012 104416 104820 105014 105115 105216 105620 105923 106016 106319 106521 107018 107220 107321 107422 107725 108121 108222 108323 108525 108828 109123 109325 109830 110007 110411 110613 111009 111312 111413 111514 111716 112112 112314 112516 112819 112920 113215 113619 113821 113922 114217 114318 114419 114621 114823 114924 115017 115219 115320 115825 116120 116221 116524 116726 117122 117324 117930 118023 118124 118225 118528 118730 118932 119126 119732 119833 120111 120212 120515 120717 121315 121416 121618 121820 121921 122317 122620 122822 123117 123420 123521 123824 123925 124220 124422 124624 124826 125525 126325 126628 126931 127327 127630 127933 128026

get /dcl01/smart/data/hpc900/sub-disk1/128026/MNINonLinear/Results/rfMRI\_REST\*\_\*/rfMRI\_REST\*\_\*\_Atlas\_MSMAll\_hp2000\_clean.dtseries.nii /Volumes/Elements/Hyperalignment/HCP/HCP900/sub-disk1/128026/MNINonLinear/Results

cd /dcl01/smart/data/hpc900/disk2]$ ls

# I made these commas, and sorting myself for using in python

128127, 128329, 128632, 128935, 129028, 129129, 129331, 129432, 129533, 129634, 129937, 130013, 130316, 130417, 130619, 130821, 130922, 131217, 131419, 131621, 131722, 131823, 131924, 132017, 132118, 133019, 133625, 133827, 133928, 134021, 134223, 134324, 134425, 134728, 134829, 135225, 135528, 135730, 135932, 136227, 136732, 136833, 137027, 137128, 137229, 137633, 137936, 138231, 138534, 138837, 139233, 139637, 139839, 140117, 140319, 140420, 140824, 140925, 141119, 141422, 141826, 142424, 142828, 143325, 143426, 143527, 144125, 144226, 144428, 144731, 144832, 145127, 145531, 145834, 146129, 146331, 146432, 146533, 146634, 146937, 147030, 147737, 148032, 148133, 148335, 148436, 148840, 148941, 149236, 149337, 149539, 149741, 149842, 150019, 150423, 150524, 150625, 150726, 150928, 151223, 151425, 151526, 151627, 151728, 151829, 152831, 153025, 153227, 153429, 153631, 153732, 153833, 154229

get /dcl01/smart/data/hpc900/disk2/153631/MNINonLinear/Results/rfMRI\_REST\*\_\*/rfMRI\_REST\*\_\*\_Atlas\_MSMAll\_hp2000\_clean.dtseries.nii /Volumes/Elements/Hyperalignment/HCP/HCP900/disk2/153631/MNINonLinear/Results

cd /dcl01/smart/data/hpc900/disk3]$ ls

﻿179245, 179346, 179548, 179952, 180129, 180432, 180735, 180836, 180937, 181131, 181232, 181636, 182032, 182436, 182739, 182840, 183034, 183337, 185139, 185341, 185442, 185846, 185947, 186141, 186444, 187143, 187345, 187547, 187850, 188347, 188448, 188549, 188751, 189349, 189450, 190031, 190132, 191033, 191336, 191437, 191841, 191942, 192035, 192136, 192439, 192540, 192641, 192843, 193239, 193441, 194140, 194645, 194746, 194847, 195041, 195445, 195647, 195849, 195950, 196144, 196346, 196750, 197348, 197449, 197550, 197651, 198249, 198350, 198451, 198653, 198855, 199150, 199251, 199453, 199655, 199958, 200008, 200109, 200210, 200311, 200614, 200917, 201111, 201414, 201515, 201717, 201818, 202113, 202719, 203418, 203721, 203923, 204016, 204319, 204420, 204521, 204622, 205119, 205220, 205725, 205826, 206222, 207123, 207426, 207628, 208024, 208125, 208226, 208327, 208428, 209127, 209228, 209329

get /dcl01/smart/data/fvfarahani/task\_hcp/disk3/**wm**/\*.npy /Volumes/Elements/Chris/task\_glasser/disk3/**wm**

get /dcl01/smart/data/fvfarahani/task\_hcp/disk2/**gambling**/\*.npy /Volumes/Elements/Chris/task\_glasser/disk2/**gambling**

get /dcl01/smart/data/fvfarahani/task\_hcp/disk3/**motor**/\*.npy /Volumes/Elements/Chris/task\_glasser/disk3/**motor**

get /dcl01/smart/data/fvfarahani/task\_hcp/disk3/**language**/\*.npy /Volumes/Elements/Chris/task\_glasser/disk3/**language**

get /dcl01/smart/data/fvfarahani/task\_hcp/disk3/**social**/\*.npy /Volumes/Elements/Chris/task\_glasser/disk3/**social**

get /dcl01/smart/data/fvfarahani/task\_hcp/disk2/**relational**/\*.npy /Volumes/Elements/Chris/task\_glasser/disk2/**relational**

get /dcl01/smart/data/fvfarahani/task\_hcp/disk3/**emotion**/\*.npy /Volumes/Elements/Chris/task\_glasser/disk3/**emotion**

**Transferring Local Files to the Remote System**

put localFile

bye

spyder --new-instance option

Submitting Parallel Jobs on a Cluster (A Job with Multiple Tasks) https://blog.albertkuo.me/post/submitting-parallel-jobs-on-a-cluster/

## 1. Submit A Job with Multiple Tasks

To run things in parallel, we are going to submit a job consisting of multiple tasks. Each task will correspond to a different simulation. To do this, type the following in the terminal:

qsub -t 1:10 your\_script.sh

Here, we are submitting one job with 10 tasks. Each task is assigned an ID from 1 to 10, stored in the SGE\_TASK\_ID environment variable.[2](https://blog.albertkuo.me/post/submitting-parallel-jobs-on-a-cluster/#fn2)

SGE\_TASK\_ID gets automatically defined by the -t option. So, when you submit an array job with 10 tasks like this "qsub -t 1:10 script.sh," each of the tasks get their own SGE\_TASK\_ID ranging from 1 to 10.

## 2. Use the Task ID to Set Parameters

In order to change the parameters for each simulation, you need to be able to “read in” the SGE\_TASK\_ID variable. In Python, you can do this with the following code chunk:

**import** os

task\_id = int(os.getenv("SGE\_TASK\_ID"))

Then once you have the task ID, you can easily set the parameters of your simulation as needed. For example:

# Parameter options

parameters\_arr = np.array([50, 100, 200])

# Select parameter

parameter = parameters\_arr[task\_id - 1]

# Run simulation

runSimulation(parameter = parameter)

@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @

@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!

Someone could be eavesdropping on you right now (man-in-the-middle attack)!

It is also possible that a host key has just been changed.

The fingerprint for the ECDSA key sent by the remote host is

SHA256:7nJrfDJCFwqzbxc98J+pjBFzrct3GZRwjmlfKAzP96c.

Please contact your system administrator.

Add correct host key in /Users/Farzad/.ssh/known\_hosts to get rid of this message.

Offending ECDSA key in /Users/Farzad/.ssh/known\_hosts:6

ECDSA host key for jhpce02.jhsph.edu has changed and you have requested strict checking.

Host key verification failed.

Response: You need use a text editor to open the file and remove the line 6 before running the ssh command

ls -a -> cd .ssh -> nano known\_hosts -> remove the line 6

Oracle Cloud

<https://cloud.oracle.com/?region=us-ashburn-1>

Username: fvfarahani

Pass: @Fv@1369@Fv@

Onedrive link:

<https://livejohnshopkins-my.sharepoint.com/personal/psadil1_jh_edu/_layouts/15/onedrive.aspx?csf=1&web=1&e=h0dvAn&cid=98c5f902%2D2004%2D4d19%2D92ca%2D8a6ad3bd81e5&id=%2Fpersonal%2Fpsadil1%5Fjh%5Fedu%2FDocuments%2Fprojects%2Fbrain%2Dbehavior%2Drelationship&FolderCTID=0x012000E035BCA3C457F04596C46BC0D36802E6>

Submit a bash script to work on the files (e.g., rename):

1. Log in to the JHPCE cluster using your JHED ID and password.
2. Once you are logged in, navigate to the directory where your npy files are located using the **cd** command. For example, if your files are in a directory called **data**, you would run **cd data**.
3. Open a text editor (e.g., **nano**, **vim**, or **emacs**) and paste the script provided in the previous answer. Save the file with a **.sh** extension (e.g., **rename\_files.sh**).
4. Make the script executable by running the command **chmod +x rename\_files.sh**.

#!/bin/bash

# navigate to the directory where your npy files are located

cd /dcs05/ciprian/smart/farahani/SL-CHA/ts/REST1\_LR\_aligned/

# loop through each npy file in the directory and rename it

for file in \*.npy

do

# extract the subject ID from the file name

sbj\_ID=$(echo $file | grep -oE '[0-9]+')

# rename the file with the desired format

mv $file "REST1\_LR\_aligned\_${sbj\_ID}.npy"

done

1. Run the script by typing **./rename\_files.sh** and pressing Enter.