BMRC Slurm Notes

- `srun` gives us dedicated resources to run code interactively:

```
srun −p short −−pty bash
```

  One can use `exit` to logout.

- `sbatch` can be used to submit non-interactive jobs. Using the following command to submit a job

```
sbatch −p short hello.sh
```

  When the job is finished, a default file ending with `.out` is produced that reports both the output (results of `stdout`) and the error (results of `stderr`).

  - `-D` or `--chdir` can be used to tell Slurm what the workng directory should be:
    - One can either pass this to the command line:

```
sbatch −D /well/project/users/username/slurm_test hello.sh
```

    - One can also add an extra parameter to a script file

```
#SBATCH −D /well/project/users/username/slurm_test
```

    - Alternatively, one may want to alter the `.bashrc` file so that it works every time:

```
export SBATCH_ACCOUNT=project.prj
```

- It is suggested to use `module purge` at the start of our own script file first so that Slurm does not inherit from our environment when submitting the job. Then, we can load required environments in the script file.

- Instead of typing parameters in the shell explicitly to alter `sbatch` behavior, it is recommended to include configuration parameters directly in the script file:

```
#SBATCH −A project.prj
#SBATCH −J my−job
#SBATCH −o my−job−%j.out
#SBATCH −e my−job−%j.err
#SBATCH −p short
```

  - `-A` specifies our project account name, which is usually `ansari.prj`. However, this can be check with `id -gn`.
  - `-J` specifies a job name, which is usually the name of the script file.
  - `-o` (`-e`) specifies an output (error) file with the job-id in its name represented by `%J`.
  - `-p` specifies the job is a short run.

A copy of a template file can be found:

```
cp /apps/scripts/slurm.template.sh ~/myjobscript.slurm.sh
```

- To see the status of all queued or running jobs, one can use:

```
squeue –u <username>
```

  Or one can see the status of a specific job by

```
squeue –j <job_id>
```

  One can also view jobs currently running/pending by adding `-t RUNNING` / `-t PENDING`, or jobs in a particular partition by adding `-p <parition>`.

- To show detailed information about a job in the queue, one can use

```
scontrol show job <jobid>
```

  The `-dd` option will cause the command to show additional details. If `<jobid>` is not specified then `scontrol` will show statistics for all jobs.

- Type `sinfo` to see Slurm partitions information[1].

- `sacct` allows us to see finished jobs as well:

```
sacct –j <jobid> ––format=JobID,JobName,MaxRSS,Elapsed
```

  Or

```
sacct –u <username> ––format=JobID,JobName,MaxRSS,Elapsed
```

  depending on whether one wants to see a specific job or all jobs. What can be reported can be check by `sacct --helpformat`.

- To see a status report of running jobs only, use

```
sstat ––format=AveCPU,AvePages,AveRSS,AveVMSize,JobID –j <jobid> ––allsteps
```

- To cancel a job, type

```
scancel <jobid> command
```

- Note that though lines starting with `#` are bash comments, those followed by SBATCH can be interpreted by the scheduler.

- If one wants to run a same code multiple times on different sets of input data where they are all independent, it is recommended to submit an array job that

  – Made up of a number of tasks, with each task having a specific task id.

  – Has a single submission script.

  – Has a single job id.

[1] For more information, refer to: https://www.medsci.ox.ac.uk/for-staff/resources/bmrc/using-the-bmrc-cluster-with-slurm.

See the following example for an example submission:

```
sbatch −−array 2−10:2 myscript.sh
```

where tasks are numbered 2,4,6,8,10. One can also specify a comma-separated subset of tasks after the `--array` to achieve execution on these tasks. More information on array jobs can be found on the BMRC cluster page.

- To submit dependent jobs, one can use

```
JOBA_ID=$(sbatch −−parsable −p short jobA.sh)
sbatch −p short −d afterok:$JOBA_ID jobB.sh
```

so that job B won't start until job A is finished. If both jobs are array jobs that have one-to-one correspondence, simply use

```
JOBA_ID=$(sbatch −−parsable −p short −−array 1−100 jobA.sh)
sbatch −p short −−array 1−100 −d aftercorr:$JOBA_ID jobB.sh
```