<table>
<thead>
<tr>
<th>Dalmata (NYU Abu Dhabi)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transferring data to/from the clusters</td>
</tr>
<tr>
<td>Transferring data to/from Prince cluster using Globus</td>
</tr>
<tr>
<td>Submittin g jobs with sbatch</td>
</tr>
<tr>
<td>Available software</td>
</tr>
<tr>
<td>Licensed Software Available on the HPC Cluster</td>
</tr>
<tr>
<td>Building Software</td>
</tr>
<tr>
<td>Slurm Tutorial</td>
</tr>
<tr>
<td>Tutorials</td>
</tr>
<tr>
<td>FAQs</td>
</tr>
<tr>
<td>Scratch Area Cleanup</td>
</tr>
<tr>
<td>Programming for Biologists</td>
</tr>
</tbody>
</table>
Running jobs on the Prince Cluster

Accessing the Prince Cluster

- From Windows workstation
- From Mac workstation

Software and Environment Module

Job script and resource request

Introduction to job scheduling

Submitting jobs with sbatch

Requesting resources
Pulling it all together - Preparing, submitting and monitoring a job on Prince

In this section we will prepare, submit and monitor a small R job. Our test case comes from the NYU Data Services "Introduction to R" tutorial

Exercise
Start a terminal session on Prince and replicate this example in it.

Choose your own example
After - or instead of - following this example through, prepare and submit a run of something genuinely relevant to your research. This way, if you are doing this tutorial in a classroom, the presenter will be available should you have questions or strike difficulties

We're using R, so first we'll look for available modules. On Prince:
There's a few modules starting with r, and a couple of versions of R. We'll use the latest version, 3.1.2.

```bash
$ module purge
$ module list
No Modulefiles Currently Loaded.
$ module load r/intel/3.3.2
```

Take a look at what it did:

```bash
$ module list
Currently Loaded Modulefiles:
   1) jdk/1.8.0_111  2) intel/17.0.1  3) openmpi/intel/2.0.1  4) r/intel/3.3.2
```

... clearly, R uses a lot of other packages. The modulefile has looked after loading the correct ones.

```bash
$ module show r/intel/3.3.2
```

For our example, we'll get some code and data from /share/apps/examples:

```bash
$ mkdir /beegfs/$USER/R-example
$ cd !$
$ cp /share/apps/examples/r/basic/* .
```

Take a look at the job script:

```
$ module avail r
--- /share/apps/modulefiles
---
gstreamer/intel/1.10.2  mothur/intel/1.35.1  r/intel/3.3.2
```

---

**Hint:** there are usage examples for a few common packages here

```bash
$ mkdir /beegfs/$USER/R-example
$ cd !$
$ cp /share/apps/examples/r/basic/* .
```
There are a few steps we can try here:

1. Start an interactive batch session, and run the example.R script interactively
2. Close the interactive session, and submit the batch script as a job:

   $ sbatch my_R_job.s

You'll get a job id returned.

Is it running yet?

   $ squeue -u $USER

You could watch the output in the run directory:

   $ ls -l ${SCRATCH}/R-example

Finally, when the job finishes, you should see a .out file in the directory you submitted from.

**Exercise**

Experiment with sbatch options for the job name, output and error file merging and location, resource limits.