Putting all pieces together - an R example

<table>
<thead>
<tr>
<th>Quick Links</th>
</tr>
</thead>
<tbody>
<tr>
<td>HPC Home</td>
</tr>
<tr>
<td>Getting an account</td>
</tr>
<tr>
<td>Gentle Introduction to using HPC</td>
</tr>
<tr>
<td>Getting started on Prince</td>
</tr>
<tr>
<td>Prince How-to Articles</td>
</tr>
<tr>
<td>Logging in</td>
</tr>
<tr>
<td>Windows</td>
</tr>
<tr>
<td>Mac / Linux</td>
</tr>
<tr>
<td>Clusters and Storage</td>
</tr>
<tr>
<td>Prince (HPC)</td>
</tr>
<tr>
<td>Dumbo (Hadoop)</td>
</tr>
<tr>
<td>Brooklyn (OpenStack)</td>
</tr>
</tbody>
</table>
Dalm a (NYU Abu Dhab i)

Transferring data to/from the clusters
Transferring data to/from Prince cluster using Globus
Submiting jobs with sbatch
Available software
Licensed Software Available on the HPC Cluster
Building Software
Slurm Tutorial
Tutorials
FAQs
Scratch Area Cleanup
Programming for Biologists
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:

```
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:

```bash
$ sbatch my_R_job.s
```

You'll get a job id returned.

Is it running yet?

```bash
$ squeue -u $USER
```

You could watch the output in the run directory:

```bash
$ 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.