### Quick Links

- HPC Home
- Getting an account
- Gentle Introduction to using HPC
- Getting started on Prince
- Prince How-to Articles
- Logging in
  - Windows
  - Mac / Linux
- Clusters and Storage
  - Prince (HPC)
  - Dumbo (Hadoop)
  - Brooklyn (OpenStack)
Dalmia (NYU Abu Dhabi)

Transferring data to/from the clusters
Transferring data to/from Prince cluster using Globus

Submitting jobs with sbatch

Available software
Licensed Software Available on the HPC Cluster

Building Software
Slurm Tutorial
Tutorials
FAQs
Scratch Area Cleanup
Programming for Biologists
Basic R jobs

Multiple R versions exist in HPC environment. To check what are available, on Prince:

```
$ module avail r
```

```
--------------------------------------------- /share/apps/modules/modulefiles
---------------------------------------------
 r/intel/3.0.3                           ray/openmpi/intel/20160114
 reprozip/intel/1.0.3                    rpy2/intel/2.5.6
 r/intel/3.1.2                           rdkit/intel/201409.2  requests/2.7.0
 rsem/intel/1.2.15                       rdp_classifier/2.11
 ribopicker/0.4.3                        rseqc/intel/2.3.9
 r/intel/3.2.2                           recon/intel/1.08      rmblast/2.2.28
 rstudio/0.98.1028                       rendertoolbox3/2.1-18
 randfold/intel/2.0                      rose/20151118
 rtax/0.984                              repeat_masker/4.0.5
 raxml/intel/7.3.0                       ruby/gnu/2.1.1
 rosettaintel/54167                      repeatmodeler/1.0.8
 raxml/intel/8.0.23                      repeatscout/intel/1.0.5
 rosetta/openmpi/intel/2014.35.57232
 raxml/intel/8.2.5                       rosetta/openmpi/intel/54167
```

Suppose we want to use 3.3.2, run these commands:
We first clean up the environment by doing 'module purge'. Then we load the R version selected, check what are available in current environment. We can see that R 3.3.2 is indeed loaded along with its dependency modules. Let's try this basic R example. We name it "example.R":

```r
df <- data.frame(x=c(1,2,3,1), y=c(7,19,2,2))
df
indices <- order(df$x)
order(df$x)
df[indices,]
df[rev(order(df$y)),]
```

Below is the screen scene while running it on Prince:
What is shown above is a simple demo case on login nodes. For real interactive analysis scenario, users are encouraged to run on compute nodes using the `srun` command to request dedicated resources, e.g.:

```
$ srun --x11 --nodes=1 --ntasks-per-node=4 --mem=4000 -t2:00:00 --pty /bin/bash
$ xterm
$ module load r/intel/3.3.2
$ R
```

Besides running our analysis interactively, long running and big data crunching jobs ought to be submitted to the batch system `slurm`. The "example.R" can be submitted to `slurm` to run in batch mode.

Copy example files to your newly created directory.
Below is how the example looks like:

```r
$ cat example.R
df <- data.frame(x=c(1,2,3,1), y=c(7,19,2,2))
df
indices <- order(df$x)
order(df$x)
df[indices,]
df[rev(order(df$y)),]
```

Then create a `sbatch` job script as:

```bash
$ cat run-R.sbatch
#!/bin/bash
#
#SBATCH --job-name=RTest
#SBATCH --nodes=1
#SBATCH --tasks-per-node=1
#SBATCH --mem=2GB
#SBATCH --time=01:00:00
module purge
module load r/intel/3.3.2

cd /scratch/$USER/example
R --no-save -q -f example.R > example.out 2>&1
```

Once the `sbatch` script file is ready, it can be submitted to the job scheduler using `sbatch`. After successful completion of job, verify output log file for detail output information.

```bash
sbatch run-R.sbatch
```