1 Using the Cluster

The Mathematics and Statistics department has a cluster for computations. You should submit jobs that need extensive computation to this cluster. The cluster runs Linux. All jobs must be submitted as shell scripts, so you need some familiarity with the Bourne shell. There are a number of online resources you can use to look up necessary commands. Here are some useful commands to help you

ls This lists the files in the current directory. A very useful option is `ls -l` which lists additional information about each file.

cd This changes to a directory given on the command line.

mkdir This creates a new directory with the given name. Sorting your files into different directories makes it easier to find the files you need.

chmod This changes file permissions. You can use this to allow other users to access certain files in your account, but since the directory is usually not accessible, it won't be very useful. A more important use is to make a script executable. The command

```
chmod a+x script.sh
```

makes `script.sh` executable, allowing you to submit it. If it is not executable, you can't run it.

cat This displays the file to the screen.

less This displays the file to the screen, but allows forward and backward scrolling, so that you can read the whole file.

emacs This is a powerful text editor. You can use this to edit your scripts on the cluster. Since the menu bar won't work on a terminal, you'll need to know the following two commands: `ctrl+x, ctrl+s` saves the file. `ctrl+x, ctrl+c` closes emacs. (There are many other powerful keyboard shortcuts if you use emacs a lot.)

To run a script you have written, with name `script.sh` in the current directory, first make sure the script is executable using the `chmod` command (see above), then type `./script.sh` The "`.`" indicates the current directory.

1.1 Here Documents

Often you want to submit a large number of jobs with slightly different parameters — e.g. simulations with different random seeds or different scenarios. One approach I use for this is called a here document.

In the script a here document looks like

```
R --no-save <<EOF

# YOUR R-CODE HERE
# use the source command to run your main script.

EOF
```

The string `<<` starts the document. The following characters `EOF` define a string that (put at the start of the line) will end the document. Anything between these lines will be passed as input to `R`. (I use the `--no-save` option, since I may have multiple `R` jobs running at once, so the saves will all clash anyway. If I want to save the `R` state, I use the `save` command in `R`.) However, before these lines are sent to `R`, variables are expanded. Variables in the Bourne shell are accessed by preceeding the variable name with the dollar sign, `$` and ended with whitespace characters or quotation marks. For example, if the variable `n` has value `"17"`, then the text

```
hello$n <- 5
```

will be changed to

```
hello17 <- 5
```

before sending it to `R`. A few important points about this:

- This means that if your `R`-code contains dollar signs, or backslash signs, you need to preceed them with a backslash. That is, `\$` or `\\`.
- Since variable names are terminated by whitespace or quotation marks, if you want your variable name to be followed by a letter, a number or an underscore, you need to insert an empty string. For example `$n""29` will be expanded as 1729, by `$n29` will be expanded as an empty string (since the variable `n29` is not defined).

There is one set of special variables that is important here. The variables `$1`, `$2`, ..., refer to the arguments on the line that called the script. That is, if your script is called `script.sh`, and you run it with the command `./script.sh hello world` (or if you submit it with the `qsub` command) then inside the script, `$1` will be replaced by `hello` wherever it occurs, and `$2` will be replaced by `world` wherever it occurs.

The example script `Cluster_Submission_Here_Document_template.sh` uses a here document to pass its command line argument into the `R` script.

1.2 Using a Script to Submit Multiple Jobs

If I need to submit a large number of jobs, I usually write a script that runs through a loop to submit them all. The example file `Submission_Script.sh` is an example of this. For loops in the Bourne shell are typically of the form:

```
for VARIABLE in LIST_OF_VALUES
do

        SOME_COMMANDS

done
```

If you want to run over a long list of values, you can use code to generate the list of values. The backtick character ` (found below the `Esc` key on most keyboards) is used to enclose code that produces the command line arguments. For example, the easiest way to create a loop from 1 to 1000, is

```
for VARIABLE in `seq 1 1000`
do

        SOME_COMMANDS

done
```

Inside the loop, you can use `qsub` to submit the job to the cluster. When you run the script, it will go through the loop and submit all your jobs. The example script `Submission_Script.sh` gives an example of this type of script.