What is covered

After working through these exercises you should be able to:

1. Run R jobs from a file rather than interactively and monitor their progress.
2. Use the **ssh** command to access remote systems.
3. Use the **screen** command to manage access to remote systems.
4. Use the **scp** command to copy files between systems.
5. Checkpoint your jobs.

1 Running R jobs from a file

Command	Purpose
R CMD BATCH *file*	Run an R job from a file, *file*
tail -f *file*	Read a file, *file*, from the end, watching as data is appended.

Table 1: The R BATCH command

If an R job lasts more than a few minutes or will be running on a remote system, use the **R Batch** command.

Here is some code – with thanks to Tom Jin – to compute points from the Mandelbrot set. It should be stored in a file **mandel.R**.

```
#! /usr/bin/env Rscript

source("mandelbrot.R")

gridSize <- 4000

# Compute fractal
y <- iterate.until.escape(immin = -1.5, immax = 1.5, remin = -2,
                          remax = 1, delta = 3/gridSize,
                          trans = function(z,c)z^2+c,
                          cond = function(z)abs(z) <= 2, max = 127)
# Output the fractal.
if(interactive()) {
  image(y, col = topo.colors(128), useRaster = TRUE)
} else {
  library(png)
  writePNG(y, target = "mandelbrot.png")
}
```

Note the test for an interactive session **`+if(interactive()) {`**. If R is running from the terminal, the plot is displayed on the screen, if run from a script, then the plot is saved to a file, `mandelbrot.png`.

An additional file is needed (**`source("mandelbrot.R")`**. Either download **`mandelbrot.R`** from **`http://www.stats.ox.ac.uk/pub/susan/cdt/Command_Line/`** or copy and paste the following into a file of the same name.

It should be stored in the same directory as your script.

```
# Function based on http://rosettacode.org/wiki/Mandelbrot_set#R
iterate.until.escape <- function(remin, remax, immin, immax,
 delta, trans, cond, max=50, response=dwell) {
  #we iterate all active points in the same array operation,
  #and keeping track of which points are still iterating.

  re <- seq(remin, remax, delta)[-1]
  im <- seq(immin, immax, delta)[-1]
  c <- outer(re, im, function(x,y) complex(real=x, imaginary=y))

  z <- array(0, dim(c))
  active <- seq_along(z)
  dwell <- z
  dwell[] <- 0
  for (i in 1:max) {
    z[active] <- trans(z[active], c[active]);
    survived <- cond(z[active])
    dwell[active[!survived]] <- i
    active <- active[survived]
    if (length(active) == 0) break
  }
  eval(substitute(response))
}
```

Assuming the R commands are saved in a file, **`mandel.R`**, and that it is stored in the current directory, then the command

`R CMD BATCH mandel.R &`

would run the command. Remember to run the job "in the background" by appending an **`&`** to the command so that you keep control of the command line.

The output that would usually appear on the screen will be sent to a file, **`mandel.Rout`** by default. The contents of the **`mandel.Rout`** will look something like this:

```
R version 3.2.2 (2015-08-14) -- "Fire Safety"
Copyright (C) 2015 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
```

```
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> #! /usr/bin/env Rscript
> source("mandelbrot.R")
> gridSize <- 4000
> # Compute fractal
> y <- iterate.until.escape(immin = -1.5, immax = 1.5, remin = -2,
+         remax = 1, delta = 3/gridSize,
+         trans = function(z,c)z^2+c,
+         cond = function(z)abs(z) <= 2, max = 127)
> # Output the fractal.
> if(interactive()) {
+ image(y, col = topo.colors(128), useRaster = TRUE)
+ } else {
+ library(png)
+ writePNG(y, target = "mandelbrot.png")
+ }
>
> proc.time()
   user  system elapsed
 29.945   2.282  32.462
```

1.1 Monitoring jobs

It is possible to watch the output from the job as it is written to the output file. For this job use

```
tail -f mandel.Rout
```

This is a particularly useful command. For more information about **`tail`** use **`man tail`**.

To stop tail type **CTRL-C** – that is hold down the Ctrl key and press the letter c.

Command	Purpose
`ssh` *host*	Log on to a different system, named *host*.
`ssh` *user*`@`*host*`.stats.ox.ac.uk`	Log on to a different system from outside the department, named *host*.

Table 2: Logging on to a different system

2 Logging on to a remote machine

During the course, you will be using servers to run jobs. You will need to login to these servers to create and start your jobs; and you will need to be able to transfer data between your desktop systems and these servers.

There are five CDT servers. They are

greyheron, greywagtail : 758GB RAM, 96 (hyperthreaded) processors, 16TB /data directory

greypartridge, greyplover : 758GB RAM, 48 (not hyperthreaded) processors, 16TB /data directory

greyostrich : 758GB RAM, 48 (not hyperthreaded) processors,22TB /data directory, four NVIDIA TESLA K80 GPU cards (eight GPUs in total). This server should be used only for GPU work.

I'll refer to these as the `grey*` servers from now on.

From any Statistics computer the short form of the host name can be used. So

```
ssh greywagtail
```

would be used to log on the CDT server `greywagtail`. For all the following examples `greywagtail` has been used, but this can be replaced by any other server.

On each `grey*` server you should find two directories where you can store data:

`/data/`*host*`/oxwasp/oxwasp16/`*user*
`/data/`*host*`/not-backed-up/oxwasp/oxwasp16/`*user*

Data in the first directory is a backed up daily, data in the second, never. There is a system-wide limit of 300GB changed data per day for backups so please, if you are moving a lot of data around, check with other members of the group to make sure they are not doing the same thing.

Note that it is possible to set up ssh keys so that you are not prompted for a password each time

Command	Purpose
`scp` *file host*`:`*location*	Copy a single file, *file*, to a remote system, *host*.
`scp -r` *directory user@host*`.stats.ox.ac.uk:`*location*	From outside the department, copy a directory, *directory*, to location, *location*, on a remote system, *host*.
`scp` *user@host*`.stats.ox.ac.uk:`*location/file* `.`	From outside the department, copy a single file *file*, from location *location*, to the the current directory on your local computer, using the same file name.

Table 3: The `scp` command

3 Copying files between systems

You will often need to move files and directories between systems. Here are some examples.

3.1 Copy a single file from your desktop to a grey* server

- Make sure you are in the directory where the file you want to copy is stored.
- Make sure you know the location on the grey* server where you will copy to file to.

In this example the file we will copy is called `serial.R`, with username `jones`, and the location is the directory `/data/greywagtail/oxwasp/oxwasp16/jones/R-scripts/` on the server, `greywagtail`.

```
scp serial.R greywagtail:/data/greywagtail/oxwasp/oxwasp16/jones/R-scripts/.
```

Note the final space and dot "." in the location. This means that the file will have the same name as the version that is being copied. If you want to give a different name use

```
scp serial.R greywagtail:/data/greywagtail/oxwasp/oxwasp16/jones/R-scripts/MySerial.R
```

for example.

3.2 Copy a directory file from your desktop to a grey* server

- Make sure you are in the directory above the directory you want to copy is stored.
- Make sure you know the location on the grey* server where you will copy to directory to.

In this example, the directory to be copied is **Project** and is to be copied to a directory of the same name **/data/greywagtail/oxwasp/oxwasp16/jones/** on **greywagtail**.

scp -r Project greywagtail:/data/greywagtail/oxwasp/oxwasp16/jones/

3.3 Copy a single file from a system outside the department to a grey* server

If you need access to Statistics servers from anywhere other than a Statistics desktop computer use

ssh *user***@gate.stats.ox.ac.uk**

from a terminal window (or PuTTY on Windows). Replace *user* with your Statistics username, and then

ssh greywagtail

to connect to the server of your choice.

Alternatively, connect to the Statistics VPN, and use **ssh** *user***@greywagtail.stats.ox.ac.uk**.

Once on a remote system you will have access to your files in the **/homes** directory but *not* files in your **/data/***host***/***user* directory on your desktop computer.

- Make sure you are in the directory where the file you want to copy is stored.
- Make sure you know the location on the **grey*** server where you will copy to file to.

The simplest approach is to first connect the VPN. To copy a single file use

scp *file user***@***host***.stats.ox.ac.uk:***location*

for example to copy the file **serial.R**, keeping the same name, from your home directory to the directory **/data/greywagtail/oxwasp/oxwasp16/jones/** on **greywagtail** as the user **jones** use

scp serial.R jones@greywagtail.stats.ox.ac.uk:/data/greywagtail/oxwasp/oxwasp16/jones/.

If you cannot connect to the VPN then you will need to copy the file or directory first to your home directory on **gate.stats.ox.ac.uk** and then to the appropriate server.

3.4 Copy a single file from a server to your local computer outside the department

- Make sure you are in the directory where the file you want to copy will be stored.
- Make sure you know the location on the grey* server where the file you want to copy is located.

The simplest approach is to first connect the VPN.

To copy the file **serial.R** in the directory **/data/greywagtail/oxwasp/oxwasp16/jones/** on **greywagtail** as the user **jones** and using same file name on the local computer use

```
scp jones@greywagtail.stats.ox.ac.uk:/data/greywagtail/oxwasp/oxwasp16/jones/serial.R
.
```

Note the final space and dot "." which will preserve the name of the file that is being copied. To change the name replace the dot by a file name.

If you cannot connect to the VPN then you will need to copy the file or directory first to your home directory on `gate.stats.ox.ac.uk` and then to the appropriate server.

4 Running jobs on a remote machine

Command	Purpose
`screen`	Connect and disconnect from a session from multiple locations and allow long-running processes to persist without an active shell session.

Table 4: The `screen` command

Once you have the `R` script and any associated files on the server you are ready to submit the job.

On the remote system you should use the `screen` command. This allows you to submit `R` (and other) jobs, then disconnect from your session. Your desktop computer can then be switched off or rebooted, without interrupting or stopping the `R` job on the remote system. To check the process of the your job you simply `ssh` again to the same server, and start the `screen` command again.

An example session would look like this.

```
ssh greywagtail
screen
R CMD BATCH mandel.R &
```

Don't forget run the job in the background. If you want to check that the job is running use `tail -f mandel.Rout`

Once you are happy the job is running use the sequence

CTRL-a `d`

to detach from the screen process. You should see a message like:

```
screen
[detached from 6422.pts-0.greywagtail]
```

You can then logout. To reattach the screen session log back into the server and use

```
screen -r
```

If you have multiple `screen` sessions on a server, then the command

```
screen -list
```

will display all your screen sessions. For example:

```
There are screens on:
        7375.pts-0.greywagtail        (Detached)
        6422.pts-0.greywagtail        (Detached)
2 Sockets in /var/run/screen/S-jones.
```

To attach a particular session use

```
screen -r 7375.pts-0.greywagtail
```

Once you have finished with a screen session reattach the session and type in

```
exit
```

You can use `screen -list` to check that it has closed. As ever, use `man screen` for full details.

There is a longer `screen` tutorial here: http://www.rackaid.com/blog/linux-screen-tutorial-and-how-to/.

There is an alternative to the screen command, `tmux` which is installed on all `grey*` servers.

5 Checkpointing your job

Command	Purpose
`dmtcp_launch`	Checkpoint your script

Table 5: The `dmtcp` command

To further protect your jobs against both unexpected events such as power failures, or scheduled reboots use

```
dmtcp_launch R CMD BATCH mandel.R &
```

This means that in the event of a reboot, your job will start at the point at which is was interrupted.

6 Further help and advice

6.1 Remote access from Windows

For Windows users there is two useful applications

PuTTY ssh client for Windows used for a command line connection to Linux/Unix systems.

WinSCP a graphical user interface (GUI) to manage your copies.

6.2 Advice for sharing servers

Our general rules for shared server usage are here:

http://www.stats.ox.ac.uk/about_us/it_information/restrictedaccess/use_of_compute_servers

There are various links on that page which should help.