This is your introduction to the RepeatMasker. Detailed instructions are available at https://www.animalgenome.org/bioinfo/resources/manuals/RepeatMasker.html and you should read them but for now, let's just do a simple run to get you started. For this exercise, you will be examining one butterfly genome. That of Heliconius melpomene. I chose this one because it's relatively small and therefore will run quickly, even on the most sensitive settings RepeatMasker has.

For this exercise, use a qlogin to spare the head node.

#------------------------------------------------------------------------#

# GETTING A REPEAT LIBRARY

#------------------------------------------------------------------------#

RepeatMasker depends on a library of known repeats that it uses to identify parts of the genome that match to them. Thus, you will need to provide such a library. Several resources exist. Eventually, you will be providing a custom library that you have made but for now, let's just use a library from RepBase, the most used repository of TEs.

1] Create a library of TEs to use from the queryRepeatDatabase utility script provided as part of the RepeatMasker package.

In your home directory, type: 'perl /lustre/work/daray/software/RepeatMasker/util/queryRepeatDatabase.pl -species heliconius >heliconius.lib'.

Note that this would be easier if you just added the RepeatMasker util folder to you path but for now, let's just deal with the extra typing.

Note that we could also be more comprehensive in our analysis by including all known elements from all hexapods. 'queryRepeatDatabase.pl -species hexapoda >hexapod.lib'.

2] Create a new folder for your RepeatMasker analysis and move your newly created library into it.

'cd /lustre/work/<your username>'

'mkdir rmaskertest'

'cd rmaskertest'

‘cp ~/heliconius.lib .’

#----------#

# GETTING A GENOME TO ANALYZE

#----------#

As mentioned, we will be examining the Heliconius melpomene genome. You could download the genome from various places but one is lepbase, a repository of lepidopteran genomes.

‘wget http://download.lepbase.org/v4/sequence/Heliconius_melpomene_helico3_-_scaffolds.fa.gz’

The genome is zipped and RepeatMasker can’t deal with that.

‘gunzip Heliconius_melpomene_helico3_-_scaffolds.fa.gz’

You should now have two files in your directory. Your library and the genome to examine. Check with ‘ls’.

#----------#

# RUNNING A BASIC ANALYSIS

#----------#

RepeatMasker has many, many options but we’re going to run a relatively simple analysis just to show you how.

Again, notice that it would be easier to put the RepeatMasker directory in your path but we’re skipping that for now.

The simplest way to run this analysis is to simply invoke RepeatMasker and use one processor.

Type ‘/lustre/work/daray/software/RepeatMasker/RepeatMasker -lib heliconius.lib Heliconius_melpomene_helico3_-_scaffolds.fa -dir .’

Open another terminal window or your FTP client and view the folder. See what changes and how it corresponds to what's happening on your screen.

This will run the analysis and output the files to our screen and to the current directory. It will also prevent you from doing anything else in this window while it runs.

Cancel the run by typing 'ctrl-C'.

There are two ways to get around this problem → output to a file while running the process in the background or use a qsub script. The latter is preferred. To output to a file and run in the background, you simply change your command to:

'/lustre/work/daray/software/RepeatMasker/RepeatMasker -lib heliconius.lib Heliconius_melpomene_helico3_-_scaffolds.fa -dir . >log.txt &'

Type 'tail -f log.txt' in your terminal to see what's happening.

Once you've seen what's happening, cancel this job by typing 'jobs' to see all running jobs (yours should be #1) and then type 'kill %1' to terminate the job. You can't just use ctrl-C because it's running in the background. There are other ways to terminate the job but this is one of the simplest.

Using a qsub is preferred because it will allow you to log out and come back whenever you're ready to view the results. It will also allow you to submit the job to a full queue and it will start whenever a set of processors opens up.

Open nano and type the following:

```
#! /bin/csh
#$ -V
#$ -cwd
#$ -N hMel_RM
#$ -o $JOB_NAME.o.$JOB_ID
#$ -e $JOB_NAME.e.$JOB_ID
#$ -q omni
#$ -pe sm 10
#$ -P quanah

cd /lustre/work/<your username>/rmaskertest

'/lustre/work/daray/software/RepeatMasker/RepeatMasker -lib heliconius.lib Heliconius_melpomene_helico3_-_scaffolds.fa -dir . -s -pa 10 >qsublog.txt &'
```

Save the new file as “rmasktest.sh’

You’ll notice a couple of changes to the command. We could have run them earlier just as easily but I’m trying to add things bit by bit.

-s invokes the slow/sensitive search option. It’s slower but it will be more sensitive and catch more difficult to find repeats.

To counteract this slowdown, -pa invokes using more processors. In this case, ten. Note that it matches the number of processors we asked to reserve in the -p line of the qsub script header.

Run the script by exiting your qlogin, navigate back to your working rmaskertest directory and submit the script to the queue by typing ‘qsub rmasktest.sh’

Check the queue to see that it’s running and use ‘tail’ to check the qsublog.txt.

It may take several hours to finish the analysis. There are other, even faster ways to run RepeatMasker using multithreading but for now, let’s just stick with this.

#--------------------------------------------------------------------------------#

# THE OUTPUT

#--------------------------------------------------------------------------------#

Depending on the options you invoke, you may get any of several output files but some are standard.

.align – details of each hit identified. This is a big file and may be useful for downstream analyses.

.out – the most often used output file. It’s basically a summary of each hit.

.tbl – a much briefer summary file.

Use ‘head’ to examine the .out file.

Use ‘less’ to examine the .tbl file.

Use ‘head -100’ to examine the .align file.