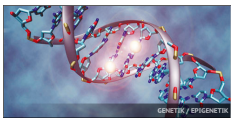
Epigenetic Data Processing Course, WS 2019/20
Prof. Dr. Jörn Walter
Abdulrahman Salhab, Michael Scherer, Kathrin Kattler
Department of Genetic/Epigenetics & Saarland University



UNIVERSITÄT DES SAARLANDES

General task sheet

1 Get to know the system

1. Be sure to always start/resume a screen session (see section 3). It might happen that the connection to the cluster is lost, and you wouldn't want to loose everything you worked on.
2. How many CPUs and how much RAM? is available Run `htop` (`q` to quit)
3. Where are the disks? Run `df -h` and `df -h /vol/*`
   (In order for everything to work, you must work out of /vol)
4. Where can I write beyond the home folder?
   a) Run `groups` to find all groups you belong to.
   b) Find your working directory: `find /vol -group <groupname>`

2 Working with tables on the command line, or awk

It is deceptively easy to work with table formatted data directly on the command line. Instead of loading into another software, like `R` or a spreadsheet program, especially filtering and reformatting comes with a couple of simple commands.

`cut` : A simple tool to extract particular columns from a file. The most important flags are:

- `-f` List of column numbers to extract. Ranges and comma separated lists are allowed. Example: `2-4,8` extracts column 2 to 4 and 8.
- `-d` the delimiter that separates the columns. Defaults to tab (`\t`), but this allows you to handle comma-separated files and others as well.

For more options, see the manual (`man cut`)

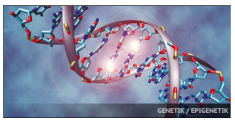
`awk` : A full-blown programming language, complete with for loops, if clauses etc, just to handle table data. Can it be any better? `awk` works slightly different compared to other linux commands in that you write a small "program" to describe what you want to do. To extract a columns 2-4 and 8 we can do this:

```
awk -vOFS='\t' '{print $2,$3,$4,$8}' file.csv
```

To break it down:

- `-vOFS` `awk` works with variables. The flag `-v` allows us to define those from outside the program. `OFS` is a special variable. It is one of many built-in `awk` variables[1], and `OFS` in particular control the output field separator. Here, we set it to tab.
- `{....}` This is the code block, that is the function that will be applied to each line of the input file. It is possible to have multiple blocks. In that case, the blocks will be executed in order, before advancing to the next line.
- `print` Self-explanatory? prints the string defined by the following argument. The `,` is used to separate columns and will be replaced by the content of the `OFS` variable
- `$2` In `awk`, the `$` sign is the code word for column. So, in the case of `$2`, it refers to the value of the second column. It can also be used to make assignments; `$2="chr"$2` will add a "chr"as prefix to `$2`.

[1] https://www.tutorialspoint.com/awk/awk_built_in_variables.htm



Where it becomes interesting is when you consider that you can make code blocks conditional, like `<condition> {...}`. The code block will then only be executed for lines fulfilling the condition. This is very useful, for instance when filtering files. Conditions can be arbitrarily complex, but here are a couple of examples:

`$2>5 && $6=="exon"` Column two larger than five and column six is equal to exon. Normal boolean operators. Strings need to be quoted, or `awk` will interpret them as variable names.

`/<regex>/` line matching the regular expression

`$3~/<regex>/` Column three matching the regular expression

`$5+$7-2 > 54` Arithmetic operations are no problem

`NR>1` The built-in variable `NR` contains the line number. This statement would skip the first line (maybe a header) of the file.

These are just a bare minimal to get started. As per ususal, Internet provides more documentation and tutorials. Here is one example:

https://www.tutorialspoint.com/awk/

3 Tips & tricks

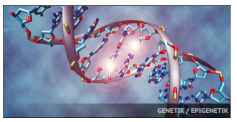
- `screen`[2] is very convenient when working in shell. It allows you to start multiple windows from the same SSH session and resume your work after de-attaching the screen. Don't create multiple screen instances. Resume and create multiple windows instead. Here are common commands for `screen`:

```
#start a screen with a given name
screen -S screen_name
#list your screens
screen -ls
#de-attach from a screen
Ctrl-a d
#Force resume screen
screen -rd screen_name
#create a new window inside the screen
Ctrl-a c
#Name screen window
Ctrl-a A
#list all screen windows
Ctrl-a "
#switch between next/previous windows
Ctrl-a n and Ctrl-a p
#terminate a screen
screen -XS screen_name quit
```

- Pause `R` to do something else (to install more packages) by sending it to the background:

```
#Send process to background
Ctrl-z
#list background processes
jobs
#resume first process
fg
#resume third process
```

[2] https://www.rackaid.com/blog/linux-screen-tutorial-and-how-to/




```
fg 3
```

- If something (R) complains about the X11 connection.
  1. First make sure you are connected with X forwarding and that VxSrv is running
  2. If this is an old screen and a new ssh connection, the settings might have changed. Detach the screen and run: `echo $DISPLAY`. Do the same inside the session. If the two are different, this is the problem. On the command line, `export DISPLAY="<correct string>"` should work. To rescue a running R session, run:

     ```
     Sys.setenv(DISPLAY="<correct string>")
     ```

4 Conda and BioConda

While a container is an excellent executor when all steps are known, it adds complexity which can be cumbersome in a more exploratory phase. Conda is a partly OS agnostic package manager that also allows you to create transferable virtual environments. BioConda[3] is an effort to provide as many bioinformatics tools as possible. Installing Conda is easy and there is a good guide on the BioConda page. Here, we will use an already installed instance.

1. Conda comes with a base environment. Normally you install it to your home folder. For general access it is located under `/vol`. You can load it with:

   ```
   source /vol/data/conda/base/bin/activate
   ```

   You now have direct access to the `conda` executable (check with `which`).

2. A conda environment can be viewed as a subsytem which comprises different software packages. We provide a pre-compiled environment (called `core`) with many important software packages. However, some packages will have to be installed by you. Clone the core environment at `/vol/data/conda/envs/core` into a **subfolder of the conda folder in your group folder**. See the documentation for help:

   https://conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html

   The command will look like this:

   ```
   conda create -clone /vol/data/conda/envs/core -p /col/<yourdisk>/<yourgroup>/conda/<envname>
   ```

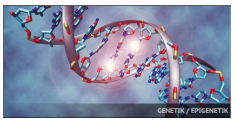
   And finally, activate your new environment.

3. Make sure the BioConda channel is properly configured (`conda info`). If not, follow the instructions on the BioConda page.

4. You can now:
   - List installed packages – `conda list`
   - Search for new packages – `conda search`
   - Install new packages – `conda install`
   - Update old packages – `conda update`

   Install bedtools version 2.22, and then update it to the latest version.

[3] https://bioconda.github.io/



5 SLURM

The capacity of a computer is in many ways limited by its components. Instead of creating special components in order to increase capacity, a common strategy is to connect many smaller computers and distribute the load. You might have been surprised not to find more processes after starting the pipeline. As a matter of fact, the only task of the process you started was to orchestrate and submit jobs/tasks to the worker nodes in the background. Here we work with a workload scheduler called SLURM[4].

1. Create a new work folder and enter it
2. Run `squeue` and `squeue -u <username>`. Are your jobs running?
3. Run `sinfo -N -l`
4. Submit a job:
   `sbatch -o job.out --wrap="sleep 60 && hostname"`
5. Check the job queue again
6. When the job is done, compare the output to that of a local execution of `hostname`

This is just scratching the surface. Some more commands to test can be found here:
https://www.rc.fas.harvard.edu/resources/documentation/convenient-slurm-commands/

[4] https://slurm.schedmd.com/overview.html