Linux III

Douglas Scofield
Evolutionary Biology Centre and UPPMAX
douglas.scofield@ebc.uu.se

1

Finding patterns: grep

- ‘grep’ is a very useful command that finds patterns
- Patterns can be literal (‘My name is Uppmax’) or can be regular expressions (‘My [a-z]\+ is Uppmax’)
- Wildcards are regular expressions too, but in grep you can do a lot more
  - many tutorials available online
  - same syntax is also used in ‘sed’ and ‘awk’ and (a slight variant) ‘perl’
  - we will only use a tiny bit of what is available

2

How many sequences in a sequence file?

- Get example Fasta and FastQ files

```
rackham5: ~/course $ cp -v /proj/introtouppmax/labs/linux2_additional-files/*.{fa,fq} .
'/proj/introtouppmax/labs/linux2_additional-files/ref.fa' -> './ref.fa'
'/proj/introtouppmax/labs/linux2_additional-files/reads.fq' -> './reads.fq'
```

- Looking at their formats…

```
rackham3: ~/course $ head -n 4 ref.fa reads.fq
==> ref.fa <==
>seq1
TGGCTCCTTTTGGTGTCAGTTGACTTGACTGGGCGGGTCCAATATCAATTGGGGCCTTTC
TGCCTTTTGGGCGGTCAGGTCTACCGGTTGTGAGGGGTGGCTTTCAACAATCTCAAAAGT
ATTTTCTGAAGACAGTTCTACTGGCTGGCTTCGCCGGCTGTAGACTGAATAACTAAAGAC

==> reads.fq <==
@UQNPK:00025:00052
GAAGAACGCAGCGAA
+
BB?BB@CCCCCCBCC
```

- Lines holding Fasta sequence names begin with '>'
- … FastQ … begin with '@'

3

How many sequences: name lines begin with > or @

- 'at beginning of line' is indicated with '^'
- 'at end of line' is indicated with '$'
- The regular expression should be single- or double-quoted so bash does not become confused (we will be using '>' !)

```
milou2: ~/course $ grep '^>' ref.fa
>seq1
>seq2
>seq3
>seq4
>seq5
>seq6
>seq7
milou2: ~/course $ grep '^>' ref.fa | wc -l
7
```

- How would you do this to count FastQ reads?
- Can you think of another way to count FastQ with 'wc -l'?

4

Extracting pieces of output or files

- What if we want just the sequence names?

```
milou2: ~/course $ grep '^>' ref.fa
>seq1
>seq2
>seq3
>seq4
>seq5
>seq6
>seq7
```

- ‘cut’ the sequence names out by (c)olumn!

```
milou2: ~/course $ grep '^>' ref.fa | cut -c2-
seq1
seq2
seq3
seq4
seq5
seq6
seq7
```

```
milou2: ~/course $ grep '^>' ref.fa | cut -c4-5
q1
q2
q3
q4
q5
q6
q7
```

5

Find the line of a specific sequence

- ‘grep -n’ includes line (n)umbers

```
milou2: ~/course $ grep -n '^>seq1$' ref.fa
1:>seq1
```

- for all matches

```
milou2: ~/course $ grep -n '^>' ref.fa
1:>seq1
11:>seq2
19:>seq3
26:>seq4
34:>seq5
43:>seq6
52:>seq7
```

- Only line numbers? ‘cut’ a (f)ield using a (d)elimiter

```
milou2: ~/course $ grep -n '^>' ref.fa | cut -f1 -d':'
1
11
19
26
34
43
52
```

What if we only want the line number for the last sequence?

```
milou2: ~/course $ grep -n '^>' ref.fa | cut -f1 -d':' | tail -n 1
52
```

6

Some differences with grep patterns

- '.' means any character, equivalent to '?' in the shell
- '*' means '0 or more of the *previous* character'
- '.*' is equivalent to '*' in the shell

Some grep patterns can be specified more simply by providing the –P option ('grep –P' for (P)erl style patterns)

- '+' means '1 or more of the *previous* character' ('\+' w/o –P)
- Terms are easier, too

```
rackham3: ~/course $ grep '^>\(seq1\|seq7\)$' ref.fa
>seq1
>seq7

rackham3: ~/course $ grep -P '^>(seq1|seq7)$' ref.fa
>seq1
>seq7
```

7

Other grep options

- grep -i : (i)gnore case in expression

```
milou2: ~/course $ grep -i 'SEQ1' ref.fa
>seq1
```

- grep -v : in(v)ert match, lines that do not match expression

```
milou2: ~/course $ grep -i 'SEQ[1-5]' ref.fa | grep -v '[457]'
>seq1
>seq2
>seq3
```

- grep -F : (F)ixed expression, ignore wildcards

```
milou2: ~/course $ ls
*  a  b  c  d  e  ee  f  ff  reads.fq  ref.fa
milou2: ~/course $ ls -l | grep -F '*'
-rw-rw-r-- 1 douglas douglas    0 Aug 25 15:11 *
milou2: ~/course $ ls -l "*"
-rw-rw-r-- 1 douglas douglas 0 Aug 25 15:11 *
```

- grep --color : use color in output

```
milou2: ~/course $ grep -i --color 'SEQ[^2-6]' ref.fa
>seq1
>seq7
```

8

Just a few more grep options

- grep -c : only print a (c)ount of the matching lines

```
milou2: ~/course $ grep -c '^>' ref.fa
7
milou2: ~/course $ grep -cv '^>' ref.fa
52
```

- grep -m *N* : stop output after *N* (m)atches

```
milou2: ~/course $ grep -m 1 'q[367]' ref.fa
>seq3
```

- grep -H : include the filename (default with >1 file)

```
milou2: ~/course $ grep -Hn --color 'q[14]' ref.fa
ref.fa:1:>seq1
ref.fa:26:>seq4
milou2: ~/course $ cat ref.fa | grep -Hn --color 'q[14]'
(standard input):1:>seq1
(standard input):26:>seq4
```

sorry, no mnemonic

- grep -l, -L : only print fi(l)enames containing/(L)acking match

```
milou2: ~/course $ grep -l 'seq1$' ref.fa reads.fq
ref.fa
milou2: ~/course $ grep -L 'seq1$' ref.fa reads.fq
reads.fq
```

9

The last grep options, seriously

- grep -B *N* : include *N* lines (B)efore the match in output
- grep -A *N* : include *N* lines (A)fter the match in output

```
milou2: ~/course $ grep -B 1 '^>seq2$' ref.fa
TGTGCAGGACGCC
>seq2
milou2: ~/course $ grep -A 3 '^@UQNPK:00685:00805$' reads.fq
@UQNPK:00685:00805
GAAGGATCATTGAATCTATCGTGCA
+
1=<>;??9895442444:4444999
```

- Just the sequence of that read? The quality string? The name of the next read?

```
milou2: ~/course $ grep -A 1 '^@UQNPK:00685:00805$' reads.fq | tail -n 1
GAAGGATCATTGAATCTATCGTGCA
milou2: ~/course $ grep -A 3 '^@UQNPK:00685:00805$' reads.fq | tail -n 1
1=<>;??9895442444:4444999
milou2: ~/course $ grep -A 4 '^@UQNPK:00685:00805$' reads.fq | tail -n 1
@UQNPK:01060:00786
```

```
milou2: ~/course $ grep '^>' ref.fa | grep -A 1 '^>seq3$' | tail -n 1
>seq4
```

10

Bash $(…)

- $(< **file**) replaces the whole $(…) with the contents of **file**
- $(*command*) replaces $(…) with the output of *command*

```
milou-b: ~/course $ cat > filelist
c
e
ff
milou-b: ~/course $ grep -n 'longer' $(< filelist)
ff:1:this file is a little longer
ff:3:this file is a little longer
milou-b: ~/course $ grep -n 'longer' $(cat filelist)
ff:1:this file is a little longer
ff:3:this file is a little longer
milou-b: ~/course $ grep -n 'longer' $(grep '[ef]' filelist)
ff:1:this file is a little longer
ff:3:this file is a little longer
```

```
milou-b: ~/course $ for F in $(cat filelist) ; do
> grep -Hn 'longer' "$F"
> done
ff:1:this file is a little longer
ff:3:this file is a little longer
```

11

Bash <(…) : anonymous named pipe

Also called *process substitution*

- <(*somecommand*) creates a temporary file containing the output of *somecommand*

- Useful for creating temporary files that don't use extra space
  - for example, do some preliminary processing before use, such as sort:

```
diff <(sort file1.txt) <(sort file2.txt)
```

  - removing blank and whitespace-only lines before processing:

```
somecommand <(grep -v '^\s*$' file.txt)
```

  - decompressing files for commands that don't handle compressed files

```
bwa mem ref.fa <(xzcat r1.fq.xz) <(xzcat r2.fq.xz) | ...
```

12

Augmenting your environment: .bashrc

- Wherever you are, save your position with 'pushd .' and cd to your home directory. See the directory stack with 'dirs'

```
milou2: ~/course $ pushd .
~/course ~/course
milou2: ~/course $ cd
milou2: ~ $ dirs
~ ~/course
milou2: ~ $ dirs -v
 0  ~
 1  ~/course
```

- Edit the '.bashrc' configuration file with nano, add the line

```
alias rm='rm -i'
```

A similar line may already be there, check first!

- Move back to previous location with 'popd'

```
milou2: ~ $ popd
~/course
milou2: ~/course $ dirs
~/course
```

13

Load an UPPMAX module with some tools

- the tinyutils module provides several useful tools
- search for module versions with **module spider**

```
rackham3: ~/course $ module spider tinyutils
```

- load the module with **module load**

```
rackham3: ~/course $ module load tinyutils/1.4
rackham3: ~/course $ which hist
/sw/apps/tinyutils/1.4/rackham/hist
rackham3: ~/course $ which table
/sw/apps/tinyutils/1.4/rackham/table
```

14

How long are my fasta sequences?

- Use the 'fastalength' tool from the exonerate module
  - `module load bioinfo-tools`
  - `module load exonerate`

```
milou-b: ~/course $ module load bioinfo-tools exonerate
milou-b: ~/course $ fastalength ref.fa
493 seq1
368 seq2
356 seq3
364 seq4
461 seq5
468 seq6
383 seq7
```

16

What is the total length? Mean? Median?

- That's what tinyutils are for

```
milou-b: ~/course $ fastalength ref.fa | cut -f1 -d' '
493
368
356
364
461
468
383
milou-b: ~/course $ fastalength ref.fa | cut -f1 -d' ' | sum
2893
milou-b: ~/course $ fastalength ref.fa | cut -f1 -d' ' | mean
413.286
milou-b: ~/course $ fastalength ref.fa | cut -f1 -d' ' | median
368
milou-b: ~/course $ fastalength ref.fa | cut -f1 -d' ' | max
493
milou-b: ~/course $ fastalength ref.fa | cut -f1 -d' ' | min
356
```

17

What is the length distribution of my reads?

- With a bit of awk (or the `len` tinyutil) to get lengths of lines

```
milou-b: ~/course $ grep '^[ACGTN]\+$' reads.fq | head -n 3
GAAGAACGCAGCGAA
GAAGAACGCAGCGAA
GAAGGATCATTGAATCTATCGTGCATCGATGAAGAACGCAGCGAA
milou-b: ~/course $ grep '^[ACGTN]\+$' reads.fq | awk '{ print length($0) }' | head -n 3
15
15
45
milou-b: ~/course $ grep '^[ACGTN]\+$' reads.fq | len | head -n 3
15
15
45
milou-b: ~/course $ grep '^[ACGTN]\+$' reads.fq | len | table
45      7
46      1
19      1
37      1
47      2
22      1
15      25
25      1
milou-b: ~/course $ grep '^[ACGTN]\+$' reads.fq | len | table | sort
15      25
19      1
22      1
25      1
37      1
45      7
46      1
47      2
```

18

Manipulating name values in bash

- *name*=**value** assigns **value** to *name*
- $*name* and ${*name*} produce the value of *name*
- ${*name*} can be useful in some contexts
  - ${*name*}_suffix prefixes the value of *name* to '_suffix'
  - $name_suffix looks in *name_suffix* for a value
- ${*name*%**pattern**} removes **pattern** from end of *name*
  - F=file.fa; echo ${F%.fa} produces 'file'
  - F=f.file.fa; echo ${F%%.*} produces 'f'
- ${*name*#**pattern**} removes **pattern** from beginning of *name*
  - F=/home/douglas/file.fa; echo ${F#*/} produces 'home/douglas/file.fa'
  - F=/home/douglas/file.fa; echo ${F##*/} produces 'file.fa'
- How might one get just the directory part?

19

Manipulating name values in bash: example

- Save a result to a filename with a modified suffix

```
milou-b: ~/course $ F=ref.fa; grep -c '^>' "$F" > ${F%.fa}.count
milou-b: ~/course $ cat ref.count
7
```

- basename and dirname can also be helpful to get filenames and directory names

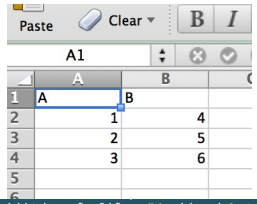
```
milou-b: ~/course $ F=/home/douglas/file.fa
milou-b: ~/course $ basename $F
file.fa
milou-b: ~/course $ dirname $F
/home/douglas
```

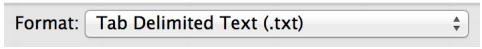
- 'man basename' and 'man dirname'

20

File conversions

- Mac, Windows and Linux text files use different line endings
  - Linux: Linefeed
  - Mac: Carriage-return
  - Windows: Carriage-return + Linefeed






```
rackham5: ~/course $ cp -v /proj/introtouppmax/labs/linux2_additional-files/Workbook1.txt .
'/proj/introtouppmax/labs/linux2_additional-files/Workbook1.txt' -> './Workbook1.txt'
rackham3: ~/course $ cat Workbook1.txt
3       6rackham3: ~/course $
rackham3: ~/course $ dos2unix Workbook1.txt
dos2unix: converting file Workbook1.txt to Unix format ...
rackham3: ~/course $ cat Workbook1.txt
3       6rackham3: ~/course $
rackham3: ~/course $ mac2unix Workbook1.txt
mac2unix: converting file Workbook1.txt to Unix format ...
rackham3: ~/course $ cat Workbook1.txt
A       B
1       4
2       5
3       6rackham3: ~/course $
```

21

Collecting multiple files into one: `tar`

- a cluster of options specifies the mode, compression, other options, and the output file into which files are collected.
  - -c creates a file; no compression, or z j J to specify compression format

```
rackham3: ~/course $ tar -cvf t.tar ?
B
C
D
a
b
c
d
e
f
t
rackham3: ~/course $ tar -czf t.tar.gz ?
rackham3: ~/course $ tar -cjf t.tar.bz2 ?
rackham3: ~/course $ tar -cJf t.tar.xz ?
rackham3: ~/course $ ls -l t.tar*
-rw-rw-r-- 1 douglas douglas 10240 Jan 20 12:29 t.tar
-rw-rw-r-- 1 douglas douglas   259 Jan 20 12:29 t.tar.bz2
-rw-rw-r-- 1 douglas douglas   243 Jan 20 12:29 t.tar.gz
-rw-rw-r-- 1 douglas douglas   272 Jan 20 12:29 t.tar.xz
```

`tar -cvf t.tar ?`
- `-c` create new
- `v` verbose option
- `f t.tar` create the file t.tar
- `?` list of files to be included in the created file

`tar -czf t.tar.gz ?`
- `z` created file is gzip-compressed

`tar -cjf t.tar.bz2 ?`
- `j` created file is bzip2-compressed

`tar -cJf t.tar.xz ?`
- `J` created file is xz-compressed

Verbose and compression options can both be included: `tar -cvzf t.tar.gz ?`

22

List the contents of a tarfile: `tar -t`

- compression format is autodetected

```
rackham3: ~/course $ tar -tf t.tar
B
C
D
a
b
c
d
e
f
t
rackham3: ~/course $ tar -tvf t.tar.bz2
-rw-rw-r-- douglas/douglas   0 2020-08-26 09:23 B
-rw-rw-r-- douglas/douglas   0 2020-08-26 09:23 C
-rw-rw-r-- douglas/douglas   0 2020-08-26 09:23 D
-rw-rw-r-- douglas/douglas   0 2020-08-26 09:23 a
-rw-rw-r-- douglas/douglas   0 2020-08-26 09:23 b
-rw-rw-r-- douglas/douglas   0 2020-08-26 09:23 c
-rw-rw-r-- douglas/douglas   0 2020-08-26 09:23 d
-rw-rw-r-- douglas/douglas  21 2020-08-26 09:23 e
-rw-rw-r-- douglas/douglas  29 2020-08-26 09:23 f
-rw-rw-r-- douglas/douglas  21 2020-08-26 09:23 t
rackham3: ~/course $ tar -tvf t.tar.xz B a z
-rw-rw-r-- douglas/douglas   0 2020-08-26 09:23 B
-rw-rw-r-- douglas/douglas   0 2020-08-26 09:23 a
tar: z: Not found in archive
tar: Exiting with failure status due to previous errors
```

`tar -tf t.tar.bz2`
- `-t` list contents
- `v` detailed listing
- `f t.tar.bz2` use tarfile t.tar.bz2

If a name or names are given after the name of the tarfile, only those files are shown in the list

23

Extract the contents of a tarfile: `tar -x`

- compression format is autodetected

```
rackham3: ~/course $ mkdir extract
rackham3: ~/course $ cd extract
rackham3: ~/course/extract $ tar -xvf ../t.tar.gz
B
C
D
a
b
c
d
e
f
t
rackham3: ~/course/extract $ ls
B  C  D  a  b  c  d  e  f  t
rackham3: ~/course/extract $ tar -xvvf ../t.tar.gz
-rw-rw-r-- douglas/douglas   0 2020-08-26 09:23 B
-rw-rw-r-- douglas/douglas   0 2020-08-26 09:23 C
-rw-rw-r-- douglas/douglas   0 2020-08-26 09:23 D
-rw-rw-r-- douglas/douglas   0 2020-08-26 09:23 a
-rw-rw-r-- douglas/douglas   0 2020-08-26 09:23 b
-rw-rw-r-- douglas/douglas   0 2020-08-26 09:23 c
-rw-rw-r-- douglas/douglas   0 2020-08-26 09:23 d
-rw-rw-r-- douglas/douglas  21 2020-08-26 09:23 e
-rw-rw-r-- douglas/douglas  29 2020-08-26 09:23 f
-rw-rw-r-- douglas/douglas  21 2020-08-26 09:23 t
```

`tar -xvf t.tar.gz`

- `-x` extract contents
- `v` show list of files as extracted
- `f t.tar.gz` extract from t.tar.gz

If a name or names are given after the name of the tarfile, only those files are extracted from the tarfile

Using two v characters 'vv' shows a detailed list of files while extracting

24

Computing and verifying checksums

- A 'short' number calculated while reading the contents of a file
- Checksums differ by a lot when files differ by a little
- If a downloaded file has a checksum, check it!

```
rackham3: ~/course $ cat e
this is a short file
rackham3: ~/course $ cat e1
this is a shirt file
rackham3: ~/course $ md5sum e > e.md5
rackham3: ~/course $ cat e.md5
a7499c996564a448b368fe716d8e9dec  e
rackham3: ~/course $ md5sum e1 > e1.md5
rackham3: ~/course $ cat e1.md5
e5fe8beffb5de4ea0b9ad7e0a002a9c1  e1
rackham3: ~/course $ md5sum -c e.md5 e1.md5
e: OK
e1: OK
```

`md5sum file`

calculates MD5 (32-byte) checksum for `file` (without file, reads stdin)

`md5sum -c file.md5`

verifies MD5 checksums for files and checksums contained in `file.md5`

Other programs calculate other checksums: SHA256, SHA512, etc.

```
rackham3: ~/course $ sha256sum e
fe70698e2af77a74a77321bed21cdf02f67d1edbcaf10fc25ea3a6a4743bf432  e
rackham3: ~/course $ sha512sum e
8bb1ea7ca3810f375835b78bba40980c94ddcc2e9234e78682ee6466339ae5087a7f667bcf45c26a1cf1daef6a264e45fe986680a004124f9fdb9cb53eb19cbc  e
```

25