Aligning DNA sequences on compressed collections of genomes

Part 4. Practical session: Unix scripting

The CODATA-RDA Research Data Science Applied workshop on Bioinformatics

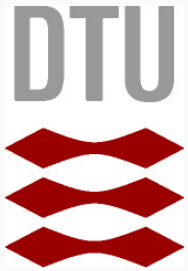
ICTP, Trieste - Italy

July 24-28, 2017

Nicola Prezza

Technical University of Denmark
DTU Compute
DK-2800 Kgs. Lyngby
Denmark

Slides adapted from
”Linux practical”, Cristian Del Fabbro



Today’s practical session

To start using bioinformatic alignment software, we have first to learn how to use **Unix bash scripting**

We will first learn how to ”communicate” commands in text format to a Unix system using a special powerful (and basic) interface: the Terminal

GNU/Linux

We work on a Unix system constituted by:

- An operating system (GNU)
- A kernel (Linux)
- A graphical interface (Gnome, KDE, Unity ...)

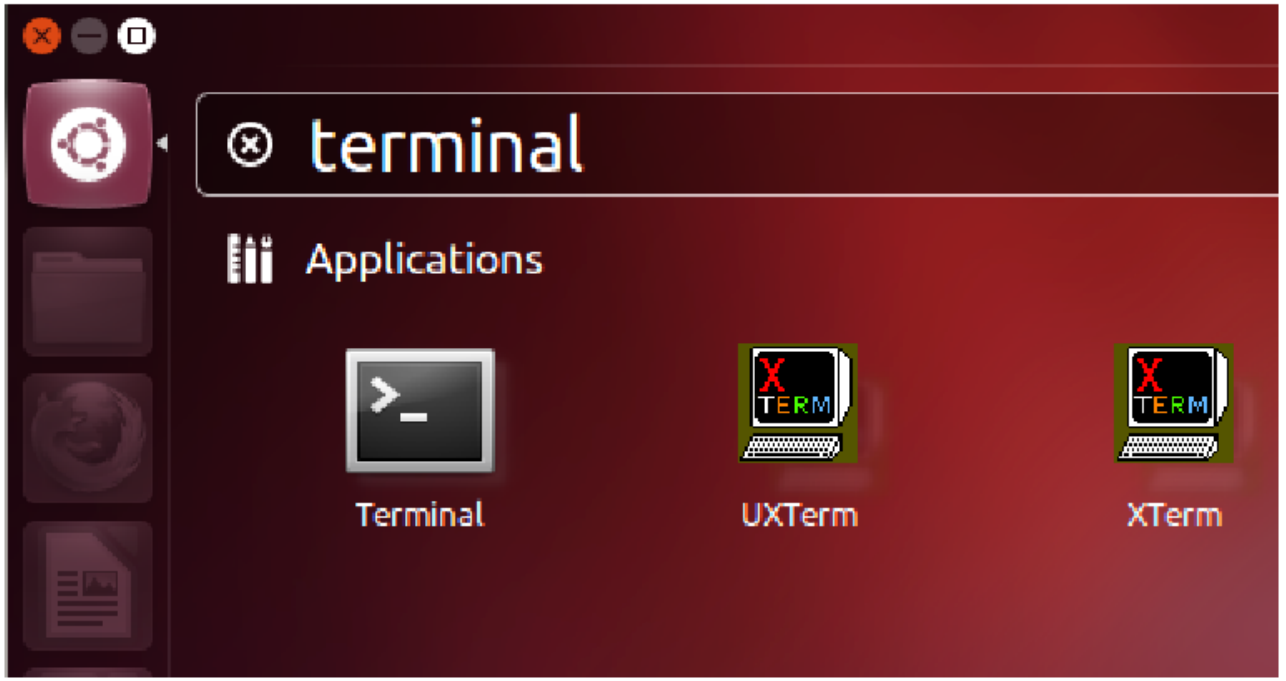
Graphical vs textual interface

- All the systems have a set of graphical applications (word processor, email reader, internet browser, ...) that can be controlled using mouse and keyboard
- All the system can be controlled using also a “textual” interface: the **terminal**

Interface	**pros**	**cons**
Graphical	easy to learn	- Cannot be automatized - Can manipulate only small files[1]
Textual	hard to learn	- Can be automatized - Manipulate huge files

[1] have you ever tried opening with excel a file of 10 GB?

The terminal



The shell

- A shell is a program that interprets and executes commands
- When you load the terminal, you interact with the shell with a *prompt*

The shell

A prompt includes several information:

`user@pc-name:~$`

Meaning

- User name: `user`
- computer name: `pc-name`
- position in filesystem: `~` (here, home directory)
- `@` and `$` are separators. We write commands after `$`

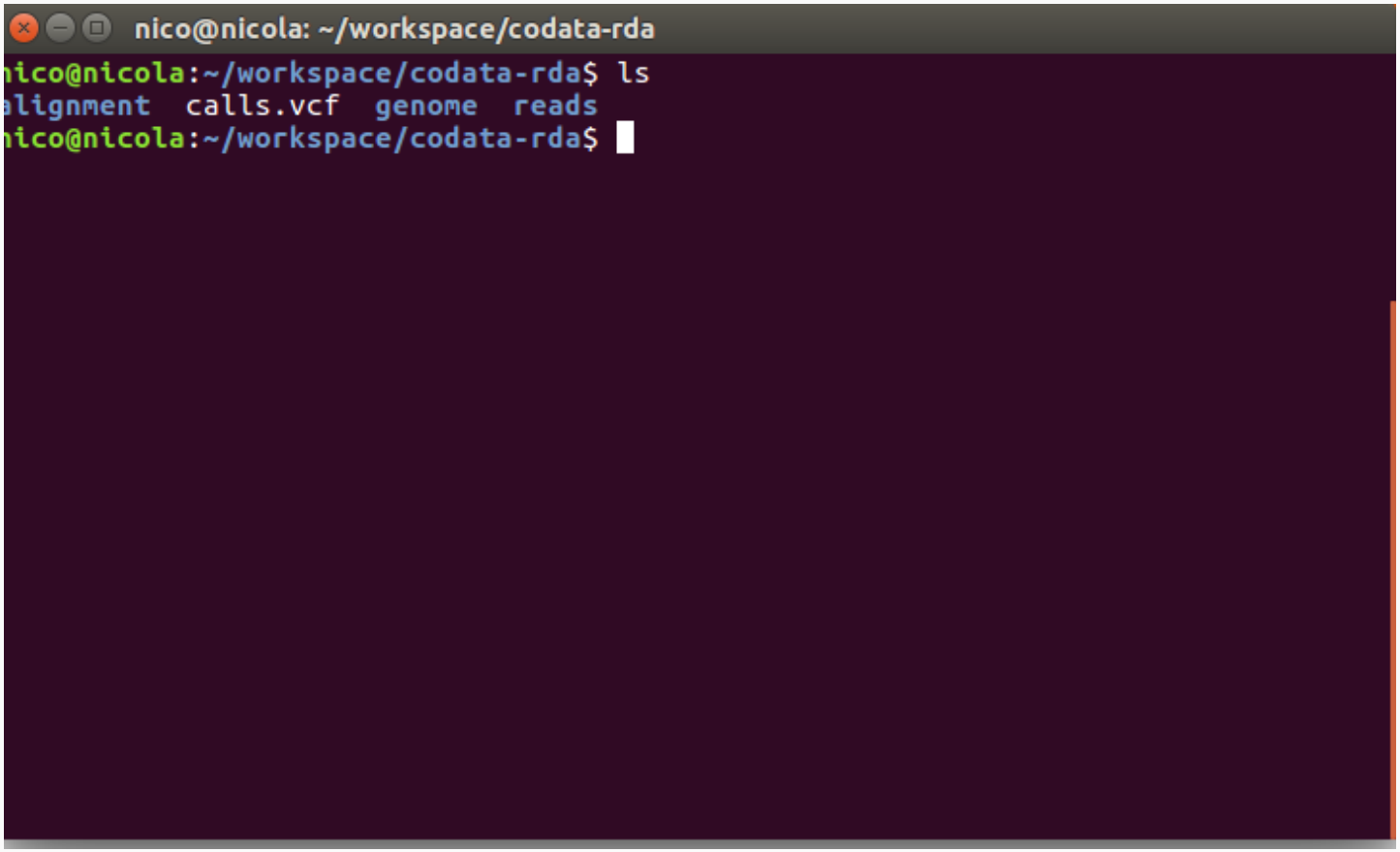
The filesystem

- A “filesystem” is a hierarchic representation (a tree) of a set of files
- Files are organized into folders (directories)
- Folders can be nested into sub-folders
- Each file and folder has a name and a path (the path from the root the the object)
- The “root” directory has no name and it is represented as / (slash)

Working directory

- The directory where we are (the prompt), is called “working directory” or “current directory”
- By default, the first working directory is the “home” (denoted by the symbol “ ”). Type the command `pwd` to discover in what folder you are.
- You can see the content of a folder (the list of files and directories) with the command `ls` (list).

Working directory



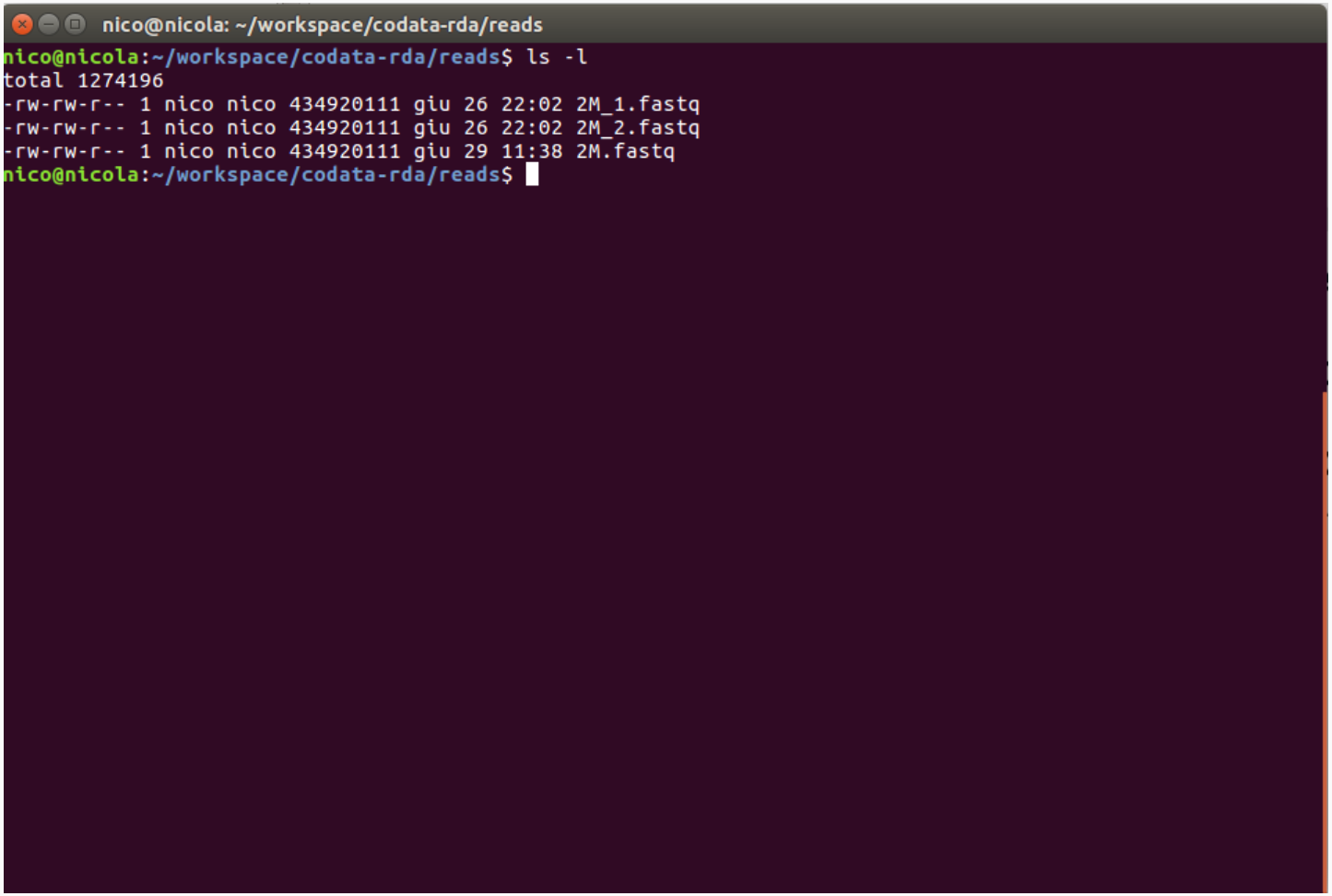
list documents

The “ls” command lists the contents of the current directory. When used from a terminal, it generally uses colors to differentiate between directories (blue), executable files (green), compressed file (red) or normal files (light gray).

list documents

- Like almost all commands in Linux, you can add options to the `ls` command to alter its output or influence its behavior
- An option is preceded by a dash or a double dash
- `ls -l` produces a “long format” directory listing; it also shows the permissions, owner, group, size, date and hour of modification
- `ls -a` lists all the files in the directory, including hidden ones

list documents



Moving in the filesystem

- You can move the current directory using the “cd” command (change directory):
  `cd codata-rda`.
  Note that prompt changes.
- you can move “one directory back” with the command
  `cd ..`
- you always return the home directory with
  `cd`

Where am I?

You always know where you are (in the filesystem):

1. reading the prompt information between ":" and "$"
2. using the command "pwd"

Absolute and relative paths

- An **absolute path** starts with a “/” (slash) and specifies the entire sequence of directories from the “root” directory (/) up to the specific file/directory being requested. Example:
  `/home/username/workspace/codata-rda/`
- A **relative path** does not starts with a “/” and is relative to the current directory. Example:
  `cd reads`
  works only IF the working directory is
  `~/codata-rda/`
  because folder `reads` is inside folder `~/codata-rda/`

Create and delete directories

- You can create a directory with
  `mkdir dir_name`
- You can delete an EMPTY directory with
  `rmdir dir_name`
- As a safety measure, the directory must be empty before it can be deleted

Remove content of a directory

- You can remove files (but not directories) with
  `rm file1 file2 file3`
- you can remove files and directory (recursively) with
  `rm -r file1 file2 file3 dir1 dir2`
- Be careful:
  - the files are DELETED PERMANENTLY
  - with -r you can destroy ALL your data

Exercise

1. create the directory “test” in your home directory
2. enter in “test” directory and create the “inside” directory
3. remove “inside” directory
4. remove the “test” directory

History and tab completion

- It does not take long before the thought of typing the same command over and over becomes unappealing. One solution is to use the command line history
- How? By scrolling with the [Up] and [Down] arrow keys, you can find your previously typed commands
- Another time-saving tool is known as command completion. If you type part of a file or pathname and then press the [Tab] key, the shell presents you with the remaining portion of the available file/path.

Changing a name and moving a file

With the command `mv` (move) you can:

- rename a file:
  `mv old_filename new_filename`
- move a file inside a directory:
  `mv filename ~/codata-rda/alignment`
  Note: `alignment` is an existing directory
- move AND rename:
  `mv old_filename ~/codata-rda/new_filename`
  Note: in this case, `new_filename` did not exist or it was a file (not a directory) before typing the command. Warning: if the `new_filename` exists, it will be silently overwritten

Copying files and directories

With the command “cp” you can make a copy of a file or a directory

- `cp old_name new_name`
- `cp file dir_name`
- `cp old_name dir_name/new_name`
- `cp -r file1 file2 dir1 dir_out/`

Warning: if the destination file exists, it will be silently overwritten

Display file content

Note: today our files are inside directory

`/scratch/`

Display file content

To display the contents of the specified file into the screen:

`less filename`

You can use arrows keys and page up/down keys to navigate up and down. Hit “q” key to quit.

Exercise

use `less` to see the content of

`/scratch/2M.fastq`

First and last lines

Show the first 10 and last 10 lines:

```
head filename
tail filename
```

Show the first “n” (e.g., 20) and last “n” lines:

```
head -n 20 filename
tail -n 20 filename
```

Exercise

see the first 5 and last 5 lines of the file

`/scratch/2M.fastq`

Write to output: echo

Command to write character strings to standard output:

`echo string`

Example: `echo hello world`

Redirect output to file

To redirect the standard output to a file, use the redirection operator ”>”:

```
echo hello world > test.txt
```

The above command writes ”hello world” in the file test.txt

The cat command

Another way to see file contents is using the cat command:

```
cat filename
```

This command displays the entire file, so it is not convenient to use it with big files. It can be used to concatenate files:

```
cat file1.txt file2.txt > file3.txt
```

Exercise

Create a single file concatenating `2M_1.fastq` and `2M_2.fastq`

The cat command

Exercise

1. In your home directory, create a new directory called “exercise” (mkdir)
2. Change your directory to the directory exercise (cd)
3. Write your name in the file name.txt (echo)
4. Write your surname in the file surname.txt
5. Concatenate files name.txt and surname.txt in the new file student.txt (cat)
6. Visualize the content of the file student.txt (less or cat)

Select lines (search): the grep command

To select lines matching a specified “PATTERN” in a file:

```
grep PATTERN filename.txt
```

Example: to select all the lines that contains the DNA sequence “CCGATTGT” from the file `2M_1.fastq`:

```
grep CCGATTGT 2M_1.fastq
```

Note: we are not specifying the path of the file so the working directory must contain `2M_1.fastq`

Select lines (search): the grep command

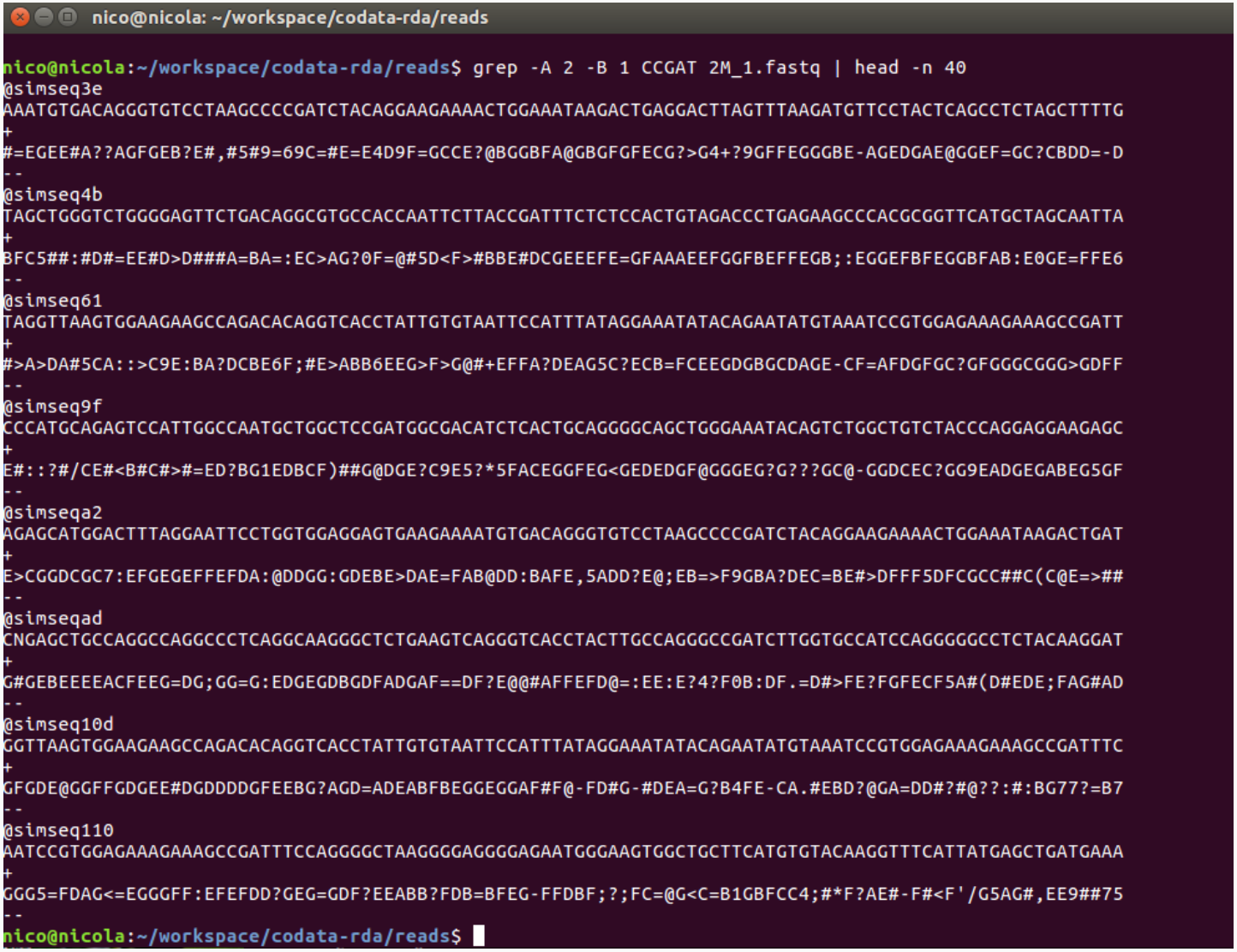
To select lines matching a specified “PATTERN” in a file, and also output x lines before and y lines after:

```
grep -B x -A y PATTERN filename.txt
```

Select lines (search): the grep command

Example: select all the lines that contains the DNA sequence "CCGATTGT" from the file `2M_1.fastq`, and also output the following 3 lines and preceding line:

```
grep -A 2 -B 1 CCGATTGT 2M_1.fastq
```


nico@nicola: ~/workspace/codata-rda/reads
nico@nicola:~/workspace/codata-rda/reads$ grep -A 2 -B 1 CCGAT 2M_1.fastq | head -n 40
@simseq3e
AAATGTGACAGGGTGTCCTAAGCCCCGATCTACAGGAAGAAAACTGGAAATAAGACTGAGGACTTAGTTTAAGATGTTCCTACTCAGCCTCTAGCTTTTG
+
#=EGEE#A??AGFGEB?E#,#5#9=69C=#E=E4D9F=GCCE?@BGGBFA@GBGFGFECG?>G4+?9GFFEGGGBE-AGEDGAE@GGEF=GC?CBDD=-D
--
@simseq4b
TAGCTGGGTCTGGGGAGTTCTGACAGGCGTGCCACCAATTCTTACCGATTTCTCTCCACTGTAGACCCTGAGAAGCCCACGCGGTTCATGCTAGCAATTA
+
BFC5##:#D#=EE#D>D###A=BA=:EC>AG?0F=@#5D<F>#BBE#DCGEEEFE=GFAAAEEFGGFBEFFEGB;:EGGEFBFEGGBFAB:E0GE=FFE6
--
@simseq61
TAGGTTAAGTGGAAGAAGCCAGACACAGGTCACCTATTGTGTAATTCCATTTATAGGAAATATACAGAATATGTAAATCCGTGGAGAAAGAAAGCCGATT
+
#>A>DA#5CA::>C9E:BA?DCBE6F;#E>ABB6EEG>F>G@#+EFFA?DEAG5C?ECB=FCEEGDGBGCDAGE-CF=AFDGFGC?GFGGGCGGG>GDFF
--
@simseq9f
CCCATGCAGAGTCCATTGGCCAATGCTGGCTCCGATGGCGACATCTCACTGCAGGGGCAGCTGGGAAATACAGTCTGGCTGTCTACCCAGGAGGAAGAGC
+
E#::?#/CE#<B#C#>#=ED?BG1EDBCF)##G@DGE?C9E5?*5FACEGGFEG<GEDEDGF@GGGEG?G???GC@-GGDCEC?GG9EADGEGABEG5GF
--
@simseqa2
AGAGCATGGACTTTAGGAATTCCTGGTGGAGGAGTGAAGAAAATGTGACAGGGTGTCCTAAGCCCCGATCTACAGGAAGAAAACTGGAAATAAGACTGAT
+
E>CGGDCGC7:EFGEGEFFEFDA:@DDGG:GDEBE>DAE=FAB@DD:BAFE,5ADD?E@;EB=>F9GBA?DEC=BE#>DFFF5DFCGCC##C(C@E=>##
--
@simseqad
CNGAGCTGCCAGGCCAGGCCCTCAGGCAAGGGCTCTGAAGTCAGGGTCACCTACTTGCCAGGGCCGATCTTGGTGCCATCCAGGGGGCCTCTACAAGGAT
+
G#GEBEEEEACFEEG=DG;GG=G:EDGEGDBGDFADGAF==DF?E@@#AFFEFD@=:EE:E?4?F0B:DF.=D#>FE?FGFECF5A#(D#EDE;FAG#AD
--
@simseq10d
GGTTAAGTGGAAGAAGCCAGACACAGGTCACCTATTGTGTAATTCCATTTATAGGAAATATACAGAATATGTAAATCCGTGGAGAAAGAAAGCCGATTTC
+
GFGDE@GGFFGDGEE#DGDDDDGFEEBG?AGD=ADEABFBEGGEGGAF#F@-FD#G-#DEA=G?B4FE-CA.#EBD?@GA=DD#?#@??:#:BG77?=B7
--
@simseq110
AATCCGTGGAGAAAGAAAGCCGATTTCCAGGGGCTAAGGGGAGGGGAGAATGGGAAGTGGCTGCTTCATGTGTACAAGGTTTCATTATGAGCTGATGAAA
+
GGG5=FDAG<=EGGGFF:EFEFDD?GEG=GDF?EEABB?FDB=BFEG-FFDBF;?;FC=@G<C=B1GBFCC4;#*F?AE#-F#<F'/G5AG#,EE9##75
--
nico@nicola:~/workspace/codata-rda/reads$

Select lines (search): the grep command

Note: if we use -A and -B commands with `grep`, in the output the matching lines are separated with ”- -”

In a few slides we will see how to remove ”- -” from the output (if this is not desired)

Select lines (search): the grep command

For now, let's see how to select only lines that **do not contain** a pattern: option -v

```
grep -v CCGATTGT 2M_1.fastq
```

Lines that do not **start** with a pattern:

```
grep -v ^CCGATTGT 2M_1.fastq
```

Pipeline

The character ”|” allows to use the output of a command as input for another program, example:

```
grep CCGATTGT 2M_1.fastq | head
```

returns the first ten lines that contains CCGATTGT

Pipeline

The character ”|” allows to use the output of a command as input for another program, example:

```
grep CCGATTGT 2M_1.fastq | head
```

returns the first ten lines that contain CCGATTGT

grep + pipe + wc

Exercise

1. Select all the lines that contain the DNA sequence “CCGATTGT” from the file `2M_1.fastq`, and also output the following 3 lines and preceding line
2. Remove from the output lines starting with ”- -”
3. Save the resulting output to a file named `CCGATTGT.txt`
4. Count the number of lines in `CCGATTGT.txt`

File compression

As seen in the previous lectures, files can often be reduced in size using compression. Several compression programs available in our system:

- `gzip file` (Lempel-Ziv)
- `7z a out.7z file` (Lempel-Ziv)
- `bzip2 file` (Burrows-Wheeler transform)

To decompress use, respectively:

- `gunzip file.gz`
- `7z e out.7z`
- `bunzip2 file`

File compression

Exercise

1. Create a file `base` containing the first 20 lines of `2M.fastq`
2. Create a file `repetitive` containing 32 copies of the file `base`[2]
3. Compress `repetitive` using `gzip`, `bzip2`, and `7z`. Who compresses better?
4. Decompress the files created in the previous step
5. Delete the uncompressed files.

[2] Hint: you could use `cat` 5 times doubling the number of copies at each time

Counting lines, characters, words

To print the number of lines, words, and bytes in a file: command **wc** (word count)

```
wc filename
```

or

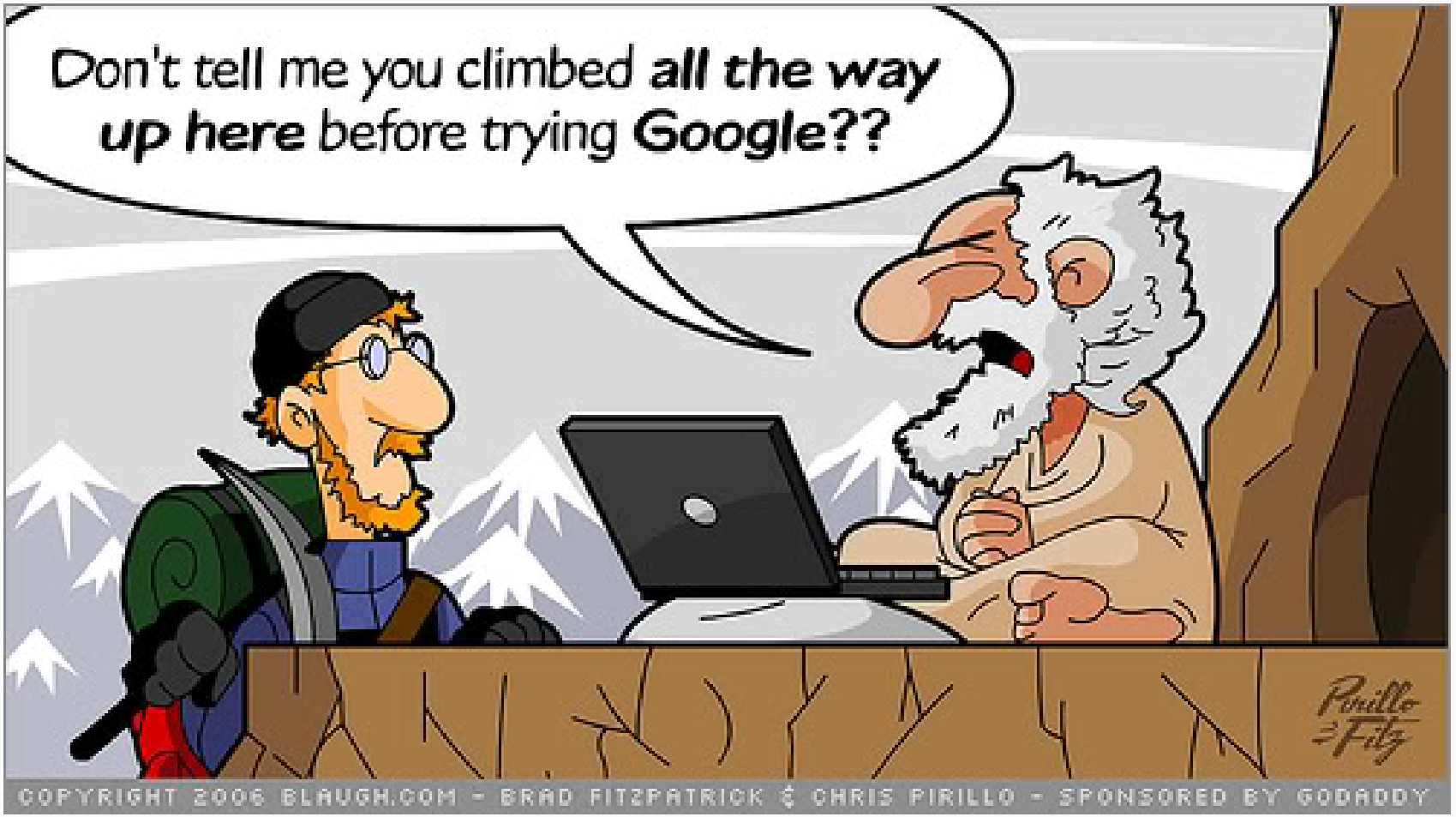
```
cat filename.gz | wc
```

To print only the number of lines:

```
wc -l filename
```

Exercise

Count how many lines contain the pattern CCGATTGT in the file `2M_1.fastq`


Don't tell me you climbed all the way up here before trying Google??
Pirillo Fitz
COPYRIGHT 2006 BLAUGH.COM - BRAD FITZPATRICK & CHRIS PIRILLO - SPONSORED BY GODADDY