

Table of Contents

- [1 Introduction to Unix, Part 3 - grep and regular expressions](#)
 - [1.1 Summary of the course today](#)
- [2 Searching patterns into files: grep](#)
 - - [2.0.1 Exercise: first grep](#)
 - [2.1 Accessing grep documentation](#)
 - [2.1.1 Exercise!](#)
 - [2.2 The Unix Philosophy and Approach to programming](#)
 - [2.3 Combining multiple Unix commands with the pipe symbol \(|\)](#)
 - [2.3.1 Exercise on Unix pipe: Searching multiple patterns with grep](#)
 - [2.3.2 the GENBANK format](#)
 - [2.4 Redirecting the output to a file, using the redirection symbol \(>\)](#)
 - [2.4.1 Note on file extensions](#)
 - [2.4.2 Warning on File Redirection and overwriting files](#)
 - [2.4.3 Appending contents to a file, using the double redirection symbol \(>>\)](#)
 - [2.5 Searching multiple files](#)
 - [2.6 Regular Expressions](#)
 - [2.6.1 The FASTA format](#)
 - [2.6.2 Exercise - search for patterns in FASTA files](#)
 - [2.7 Searching for special characters, such as ">"](#)
- [3 cut, sort and uniq](#)
 - - [3.0.1 Example cut, sort and uniq](#)
 - [3.1 the SAM and BAM formats](#)
 - [3.2 Example SAM file](#)
 - [3.3 Format of a SAM file](#)
 - [3.3.1 SAM Headers](#)
 - [3.4 SAM - Alignment section](#)
 - [3.4.1 Using cut to print the first columns of the SAM file](#)
 - [3.5 Using cut to extract columns from a file](#)
 - [3.5.1 Exercise! Fun with flags](#)
 - [3.5.2 Exercise! Extract chromosome and start position for this file](#)
- [4 Recap - grep, Unix piping, and cut](#)
 - [4.1 Example of Unix Approach](#)
- [5 Time for a break!](#)

Introduction to Unix, Part 3 - grep and regular expressions

Welcome to the Programming for Evolutionary Biology workshop!!

Giovanni M. Dall'Olio. Data Strategy and Design, GSK. March 2020.

Quick link to slides online: <https://tinyurl.com/evop-unix> (<https://tinyurl.com/evop-unix>)

All materials available here: https://github.com/dalloliogm/peb_unix_intro/archive/master.zip
(https://github.com/dalloliogm/peb_unix_intro/archive/master.zip).

Press space or down key to continue.

Summary of the course today

- Morning first half: Basic Terminal commands, First Login to Linux
- Morning second half: Login to a Remote Unix server, browsing file contents
- **Afternoon first half**: Finding patterns in a file with grep; piping commands; cut, sort and uniq
- Afternoon second half: awk and sed; makefiles

In [2]:

```
# Configuration - please ignore - this should not appear in the slideshow
alias grep='grep --color'
cd
cd workspace/peb_unix_intro/exercises/
# Please do not run this - it is only to format the slides
# if you did, type "unalias less"
alias less="head"
```

Searching patterns into files: grep

The exercises folder contains a file called `2_searching_patterns.txt`.

To see its contents, use the `less` command (or `head`).

What is going on? The contents of this file are nonsense.

In [3]:

```
less 2_searching_patterns.txt
```

```
P8mW50b1cJWpWZgKFJFwM0s7qC4Zyy      1fNoCACqioPMpafokl1qbLeviYz8PHay3R
G
Dtfe11liowE hxsksxdy95cmWsVPlFT      PvZ0A0olHEKKofWXaLFRjRMKGwceCEHg14
4
k8g99AzeTeKmOtxbg8Zqd9sPReQ4IV      OgJdwlpzORBuRr9VyMy0Dsqu9 ab743qVI
Z
cDfibnWrkZtLbJtUgQ6VBtu47Efutj      OPsz9joHgJVsMxXT4wKkrDKJ0K8R09Useg
E
dLWko0BZITViH835P88sVxcfdUBkEH      BD8J80icxMHXxfIgWuHkv5Gw1JajrE8bl
t
tD x0OwbI3BceM6OsnB5z30N7PhzjJ      3Jp8vADVnptAfc3T5DBGUUG1V7BJi4A3NP
F
AgUGZnR1NvC5kBKQF4fXakZKUJSnle      EBil5kDuu3DfN6Aiod0Ye3JVKlNyTQY3XE
7
T89m9 4lo5lrih3sleSIItTGRrOqGvH      lZ0G0New 03foaoTzDktlusrewqo6BauYx
X
FtyTsL9gi7I93hk08aEODXSngRUNkp      fzXUYW08 jdrc3pvBrRNi4gSSm4iNdwrJM
U
9hphNi41uXoyi4gXcgC6GhQGDTAbIR      qNGuxDKJ0n9EEyxgFuRxNUW5bDXLxUxCk
c
```

The file `2_searching_patterns.txt` contains some hidden messages.

We can find them with the `grep` Unix command, which allows to search for specific words or patterns into a file.

The basic syntax is:

```
grep PATTERN FILENAME
```

Exercise: first grep

Type the following command, to print all the lines containing the word `start` in the `2_searching_patterns.txt` file

```
grep start 2_searching_patterns.txt
```

In [4]:

```
grep start 2_searching_patterns.txt
```

7fJCrBDr7DLtFJWveW**start**hL74o1h
qhgW **start** hTRnNmXG6 kGBzklbr
8cd **start** ZTcMbuwb6uLSp5nelx1N
eeGNJy0Bpis07pPWPfVz4o **start** c
BOqAUjUyE**start**NxNmNMfffaCZBxz8
AmHTrLmexhknVNT0Y4a **start** 5tw0
SJVFTaWds1T0 **start** NKyluwBCkW7
hUUKV **start**LnJEPKVhgpcZBE03IyA
MktM1nrJN070b**start** 3gblKgoXGtT
m **start**QUB5dmMP7g8Qgj4Ddujwm6q
9**start**UxjflwcSvST44uLa5uj3EqK7
luD9dPnmva **start** wYKeYCxJPkW7e
AGldtgmouPUUkIQkDn8MS **start** AG
SF0gAZaOT0BF NRmYocCe4Al**start**t
jp8wri **start** nGtfuBd4Llqg6ffY
ctLAoukXdWPlw5**start** XQUe9nUuzY
VhR PxIHQtCH0q **start**KpfN MFVXw
YE ompcM7r07X **start** bHUQSQqRbf
PQZzoD7utluD9STr3T4 **start** APwT
Ll0tzzZfGLOerKfVcVrIbBJ **start**HA
6qSv NOG9**start**v6YoVLHJ7snR69tx
PLx5n7xFc3Hh5iwdfarA**start**YNL4A
Qd**start** ZIEshPHzz1usx3ALvEDkeZ
SMfbBaI0F**start** EHiw3ISlKRyE Sg
startskOYYZ1WOCQcN1os5fg9itzj9
x**start**CyNu6cHxBW0N6pYd9SLPDVuP
n.
SRw9OvAPFurRTtnfrTPtMik **start**l
kbmeUmSwdCCr3e81HJl**start**z4bFgs
e:
IPW6cn0R4iMTeqUp5lNFT**start** kaZ
SwybllovzEHStB**start**0Lr5 OzObgd
lhxeRXeOaEFFItDp9MxYvGnp**start**
4yN0AdOSIB8DG3 **start**GYAs3E5smu
hlm68FwK **start**3El5ZEpoQrgW9Raw
dEaUW VR0od **start** kKL4SZ3t7Jkj
5Q **start** 4lc3YSmRKwNvZ1mqgVw6L
g yGpLHWZWuL **start** BdjB3Z34m7U
DUIAnQiz AmLTtwb3sDtZK5 **start**
zq0B**start** lw40BCkw 70AwRVFZn60

```

/ Congrats!
|   You've used grep correctly,
\   and found a cow.
/

-----
      \      ^      ^
        (oo)\_____
          (__) \    )\ /\
                ||  ---w  ||
                ||      ||

```

The command `grep` allows to search for a pattern in a text file.

It will print all the matching lines to the screen.

=====

Next Exercise

=====

In the next exercise we will see how to access grep's documentatio

Grep the following word to continu

help

Accessing grep documentation

The `man` command allows to find documentation on a Unix command.

To find the documentation of grep:

```
$: man grep
```

Use the arrows keys to scroll up and down, and q to exit (as for the **less** command)

Exercise!

- Open the `grep` manual page
- Identify the parameter for doing case-insensitive search
- Do a case-insensitive search for the word `ignorecase` in the file `2_searching_patterns.txt`
- Count the number of matching lines (or use another `grep` option to count them)

In [5]:

```
grep help 2_searching_patterns.txt
```

```
B9cFVKf9uA3DkI1PnJChelpkTNm1dd
3Z0anUCaVbirdioBijhelp7kAVJfXy
8KKQR help VQsY47 sL9toQT8EY5D
iyGhelp6dhyXuyWHSaYNTeVoDex Cc
mcKWtNHTuilguXDA EKhelpPWuOIxH
UBBMDmOC8lhelpzkZEObgSXfjXNgrf
help 4M TGfdzHR8Yk9i9PwhNn 39
WlcSpeejraQCD1pu help WEXDH8 I
ription.
neCvTNSySVYpt0E8SGnXGIchelp9na
rfwiuKsJtjeHrHoBX9 help 0hicjc
aMUp4FGgcAHAMoj pzlNQ7Oz9helpo
7j help rXptYF5IRAXjdK7lukHJWd
YwauUPotIO help ivkaPxkiNvbJrt
FZE help lvPb0Yp11UT63FWV586j0
ecQayYhelp FWjb4HhwoHh7eek3yEX
WGj1R9df3ibXDalexNLhelpY7K9U00
OdrqEGoZakNhelp uEm4w7UkGYEkIG
JSKk13GFy help hBbPLplm7s5kQSg
g9zHZpIMsthWbx4pU help HuIH8GJ
ed to open
JBDak9BEjrgSBtOmDgjpwt9iphelp
two options:
LrYAhelp Uqy7pyCubW73LxjQcGof8
kl6pOXOgBlOWKEhOp8iKO help 06Y
searches
cMmgFjclS3tGXfAdOwaYu7xgfhelpS
help fmv KM614XJ TpZ7QvlfDByjP
kF help IzloBVA3hydzZ8cpW9TMRJ
screen.
iB5yneSru 5oAis nfaW helpZtknE
cIZkYsvp5rQN help rzRI9iwDL4a5
ions,
kX16x7a9PpjXQtoto9zVhelp 9HZij
is file for the word
Ryi5helpEZziizUlktMJnAEA4IMSnn
r of lines.
Up8sg ldoZcxZ3IaxA help IepzrB
4FdfFqlAqY93holNMkcmHd7Nihelp
s9UGvUvcQQxss1ETq21leshRaftsUQ
is
```

The documentation for `grep` can be accessed through `man`:

```
$: man grep
```

Scroll down to see all the parameters for `grep` and their desc

Use `/` to search for text.
Press the `q` key to exit.

```
=====
Next exercise
=====
```

For the next exercise, you will ne

`grep`'s documentation and identify

- the option for case-insensitive
- the option for counting the number of matching lines, instead of printing them to the

Once you have identified these opt

do a case-insensitive search on th

"ignorecase", then count the numbe

This will also help you reading th

In [6]:

```
# If we do a search for "ignorecase" without any option, we only get some of the
lines.
# You can notice that the cow is not properly displayed :-)
grep ignorecase 2_searching_patterns.txt
```

```
bp6vV3Fz6zWALIMHrignorecaseXsK
kJodjykbTRY5UsvkNignorecaselCJ
IQNrignorecaset3sSkwItlBqtJ97X
e exercise,
ignorecaseTiN1A5LE8JprmmIFPtly
search for the word
Ryi5helpEZziizUlkTmJnAEA4IMSnn
r of lines.
gBHJnHXdOKPCuNtlignorecase0Lhg
JpSx4YnQPBZYHignorecaseyAhXUnf
rSncrRqC3QweVignorecaseCJLnZUZ
```

Remember that, to continue with the exercise, you need to do a case-insensitive search for the word "ignorecase", then count the number of lines.

```
\ (oo)\_____
```

In [7]:

```
# The -i option allows to do a case-insensitive search.
# As you can see, some lines contain upper case characters:
grep -i ignorecase 2_searching_patterns.txt
```

```
bp6vV3Fz6zWALIMHrignorecaseXsK
kJodjykbTRY5UsvkNignorecaselCJ
IQNrignorecaset3sSkwItlBqtJ97X
e exercise,
ignorecaseTiN1A5LE8JprmmIFPtly
search for the word
Ryi5helpEZziizUlkTmJnAEA4IMSnn
r of lines.
uRpeIGNORECASEwt6VeQSyNEyd7cgV
G19VrSqlignOrecaseF14RySa6iLJi
hlWDPz4C7IgnorEcasevfD0uiSnmXw
BxTunYQnlgiOZdBUuhGIgnorEcase4
ignOrecaseSWU8Gagvfy vt9SdNj8j
ignOrecaseyrPZmYffv1 M9djtfz9k
3O54VIX8fd3rWIGNORECASEuqpGiSR
ACo8r0nvOfg5iee 7IGNORECASE cF
GajDz7WpdCXaxYCIgnorEcasenHiHl
gBHJnHXdOKPCuNtlignorecase0Lhg
PHqOtUJpIgnorEcasebiNPUz9HQCDp
t7wIgnorEcasepUUcgUelfQSD KGDC
HtdRU3U0slignOrecaseqNi0E0von5
JpSx4YnQPBZYHignorecaseyAhXUnf
rSncrRqC3QweVignorecaseCJLnZUZ
94ignOrecaseaWpKKR1ND0i5h7bejh
```

Remember that, to continue with the exercise, you need to do a case-insensitive search for the word "ignorecase", then count the number of lines.

```
/ Good Job! \
| You did a |
| case-insens |
| itive |
\ search /
-----
\ ^__^
 \ (oo)\_____
  (_____) \
           ||----w |
           ||
```

In [8]:

```
# To solve the exercise, we also have to count the number of output lines.
# This can be done with the "-c" option:
grep -i -c ignorecase 2_searching_patterns.txt
```

21

In [9]:

```
# solution: how to find the instructions for the next exercise
grep 21 2_searching_patterns.txt
```

```
CtDB RLr3x3do921Squ8iOvohs7vk4
4kUkCECIiM21OY8EcY4TIsPkKLB0mj
Qjk21MQpAeF4VyA8cAlEXfl8wWJzDY
B363B5byCj121rzXGoQiRahUq9zqz
La8Ku9mjSIpSfVrAgi 21Db5puhK65
Gc9kKOst21ABdPUU5Y6yxvMFbmBEr5
X9ihXj3yFmi21YQ1tRg6pnqhTLWclL
Zax213YlOJbXitxdyPlXFu0hnNoi6i
tAWUDnEbN1stleW21Jmzgrlmwcr9Ok
efc219TYASVKMmFWPOPMXaYCYnN5Nu
kkzt8YaRyVAtaXiQC7Q9LlFLzFj21a
eELR21QbUmlwLGgQQGN8F A 6HJTq3
BEN sRlcDIOQPyADwCy3217naBh8TU
saR SuoPzf5eV21w5v6sccKkqTJy9U
iYVLEcmJqF21MF1n lvFrFqXfQulK
4yQdcMZY97LVuLc9Y4217zYEzC4ceK
NZdPx216fVVJ6tc3WMzeIR6f4cHDad
Q07zhV21z1OTJazhH03k4aCpGydFEn
21PaFNZKqvVZq6bwr6FGI4TA9cy5Zr
xGKrwF3tpNVzq8ipAPxPD21pkSJ3XU
3ulFTmdrtZUFB5TAcM9lxbKgdBM21m
JsBQ21j6EGfYMsVrWtFmD Kv8wDJuI
ix
8ay3N5uRdYncy0B8RHV4XcP21sNvak
s9UGvUvcQQxss1ETq21leshRaftsUQ
is
217SnZXPGgtIIwQPeyGQ6nYZgZq4wr
nto
xtSyHNFsQZijEC21ckUdzDjaJDboxQ
or
21dPvDB6tpt7c5eKNz3voR7HbqvVEG
8CRaMViZ9m5hQXko21pwgHT6wurX39
QR21gbuUOy67xAVKYyUOeXasfj3L9O
iLgB tGHDkYZ21 iLhgjbFj9GE thp
t
sabcPdsaGZ21ccvLBDgjbFj9GE thp
```

```
/ Congrats! Yes \
| the answer to the |
| case-insensitive  |
| and count question|
| is 21.            |
\ ----- /
      \   ^   ^
       \ (oo)\_____
        (__) \       )\/\
              ||----w |
              ||     ||
```

=====

Next topic

=====

We are going to make a break from the exercises to talk about the Un

ix

pipng system.

This will also help you reading th

output more easily, by piping it i

another Unix command such as less

head.

For example, try

```
$: grep 21 2_searching_patterns.tx
```

to read this output easily.

The Unix Philosophy and Approach to programming

Earlier this morning we mentioned the Unix Philosophy:

- Make each program do one thing well.
- Expect the output of every program to become the input to another, as yet unknown, program.
- Work on file streams, reading one line at a time.

All the commands we saw today follow the Unix philosophy:

- `ls` is for listing files
- `grep` is for searching patterns in files
- `less`, `head`, and others are for seeing the contents of a file

So, they all do one single thing, and they do it well.

Combining multiple Unix commands with the pipe symbol (|)

Unix commands can be combined together using the pipe symbol, `|`.

For example, we can combine the output of a `grep` search with `head` or `less`:

```
$: grep (first pattern) myfile.txt | head
$: grep (first pattern) myfile.txt | less
```

In [10]:

```
#
grep 21 2_searching_patterns.txt | less
#grep 21 2_searching_patterns.txt | less
```

```
CtDB RLr3x3do921Squ8iOvohs7vk4
4kUkCECIiM210Y8EcY4TIsPkKLB0mj
Qjk21MQpAeF4VyA8cAlEXfl8wWJzDY
B363B5byCj121rzXGoQIiRahUq9zqz
La8Ku9mjSIpSfVrAgi 21Db5puhK65
Gc9kKOst21ABdPUU5Y6yxvMFbmBEr5
X9ihXj3yFmi21YQ1tRg6pnqhTLWclL
Zax213Y10JbXitxdyPlXFu0hnNoi6i
tAWUDnEbN1stlEw21Jmzgrlmwcr9Ok
efc219TYASVKMmFWPOpMXaYCYnN5Nu
```

```
/ Congrats! Yes \
| the answer to the |
| case-insensitive  |
| and count question|
| is 21.            |
/
-----
\   ^__^
```

Exercise on Unix pipe: Searching multiple patterns with `grep`

One way to `grep` searches is to combine them with the Unix Piping system.

This can be done using the pipe `|` symbol:

```
$: grep (first pattern) myfile.txt | grep (second pattern)
```

Press space or the down key for some examples.

the GENBANK format

The following exercises are based on the GENBANK format, used to store gene annotations.

There is an example GENBANK file at `data/genes/mgat_genes.gb`.

The left-most part of a GENBANK file contain keywords, such as `LOCUS`, `DEFINITION`, etc. This is ideal for `grep` searching.

In [11]:

```
head genes/mgat_genes.gb
```

```
LOCUS      HUMUDPCNA                      4705 bp    DNA      linear    PRI
19-SEP-1995
DEFINITION Human alpha-1,3-mannosyl-glycoprotein beta-1,
            2-N-acetylglucosaminyltransferase (MGAT) gene, complete
cds.
ACCESSION  M61829
VERSION    M61829.1  GI:340075
KEYWORDS   alpha-1,3-mannosyl-glycoprotein beta-1,2-N-acetylglucosa
minyltrae.
SOURCE     Homo sapiens (human)
   ORGANISM Homo sapiens
            Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Eute
leostomi;
            Mammalia; Eutheria; Euarchontoglires; Primates; Haplorrh
ini;
```

Let's search all the lines where "ORGANISM" is "Homo sapiens".

We need two grep commands:

```
grep ORGANISM genes/mgat_genes.gb | grep 'Homo sapiens'
```

Notice that searching for "Homo sapiens" alone would not be enough, as there are other lines where the word "Homo sapiens" is present.

In [12]:

```
grep ORGANISM genes/mgat_genes.gb | grep 'Homo sapiens'
```

```
ORGANISM  Homo sapiens
ORGANISM  Homo sapiens
ORGANISM  Homo sapiens
ORGANISM  Homo sapiens
ORGANISM  Homo sapiens
ORGANISM  Homo sapiens
ORGANISM  Homo sapiens
ORGANISM  Homo sapiens
```

The file contains sequences from two other organisms apart from Homo sapiens, Mus musculus and Bos taurus.

One of these is the keyword to access the next exercise - can you guess which one it is?

```
# Solution: grep for "bos taurus":
grep ORGANISM genes/mgat_genes.gb | grep taurus
```

$$\backslash \quad \begin{array}{c} \wedge \quad \wedge \\ \hline (oo) \end{array} \quad \backslash$$

() \

||--

11

=====

Next Exercise

=====

Let's talk about
the Unix Piping syst
before searching mul

```
# Note that no output is shown to the terminal.
grep ORGANISM genes/mgat genes.gb | grep taurus > bos_taurus.txt
```

10/26

```
# The -t option in ls sorts the output by modification date
ls -l -t
```

Use `less` , `head` or `cat` to see the contents of this new file:

```
# cat (concatenate) prints the whole contents to the screen:
cat bos taurus.txt
```

$$\begin{array}{c} \diagup \quad \diagdown \\ \diagdown \quad \diagup \\ \diagup \quad \diagdown \end{array}$$

() \

||--

=====

Next Exercise

=====

Let's talk about
the Unix Piping syst
before searching mul

Note on file extensions

Unix is agnostic to file extensions (e.g. .txt, .png, .exe).

In the last exercises we added the .txt extension to the file, however we only did it for convention.

You can run `grep` and any unix command on any file type, regardless of the extension; if the file is in binary format, the output will be made of random characters.

Warning on File Redirection and overwriting files

Important - the redirection symbol will overwrite any existing file content, without asking confirmation. There is no back-up and no Trash folder - once a file is overwritten, the previous contents are gone.

The next command uses `echo` to print a simple message, which is then redirected to the `bos_taurus.txt` files created previously.

In [17]:

```
# The echo command prints a message to the screen - or wherever you redirect it
echo "Simple Text message"
```

Simple Text message

In [18]:

```
echo "All previous contents have been deleted" > bos_taurus.txt
```

In [19]:

```
head bos_taurus.txt
```

All previous contents have been deleted

There is now way to retrieve the previous contents of the file, except if you used a version control system (e.g. git) to store them.

Appending contents to a file, using the double redirection symbol (>>)

We can append content to a file, without deleting the previous contents. This is done with the double redirection symbol, `>>` .

This will store the new output at the end of the file. Let's try it:

In [20]:

```
echo "My first message" > redirect_me_here.txt
ls
cat redirect_me_here.txt # cat is similar to head, but prints the whole contents
```

```
1_browsing_textfiles.txt  chr8          ngs
2_searching_patterns.txt  exercise_sequences.fasta  old_files
bos_taurus.txt            genes         redirect_me_her
e.txt
chr20                     multiplefiles
My first message
```

Let's keep adding messages:

In [21]:

```
echo "My second message" >> redirect_me_here.txt
echo "My third message" >> redirect_me_here.txt
echo "My fourteenth message" >> redirect_me_here.txt
```

In [22]:

```
cat redirect_me_here.txt
```

```
My first message
My second message
My third message
My fourteenth message
```

Searching multiple files

Let's go back to our grep exercises.

Grep can also search for patterns across multiple files, using in a single command.

The folder `data/multiplefiles/` contains 50 randomly generated files. You can see their contents with `head data/multiplefiles/*` or with `less`.

One of these files contains the word "regex" in it. Are you able to find it?

In [23]:

```
# solution: you can use the "*" character to specify multiple files:
grep 'regex' multiplefiles/*
```

```
multiplefiles/file32.txt:5gsumFTKbKEJv9dD8W94FhoEQU8qf8RMUcregexR
multiplefiles/file32.txt:YgDiqKA Clorregex9giqI66c3sOwfLirOsgPpSuq
multiplefiles/file32.txt:IsXSnP 8U8pKR0LsVuKregexO5GFegOtV4GW4fNQ
Good! You've found the
multiplefiles/file32.txt:l 4px8KhPRmfEJgi5uTuV01XahG3HlsYregex4wt
file containing the word "regex"
multiplefiles/file32.txt:yz8P5 HC6N5D XRHPncZjTAeMregexT9bQUoZdsh
multiplefiles/file32.txt:lKjregexMHQbp zJEV xF4EvzMhyCrdYJlUHD4ol
The next lesson will focus on using
multiplefiles/file32.txt:zd6C 9regex sviPg977VRvG cSC3TPv6E0PD18
Unix piping system.
multiplefiles/file32.txt:eWUd18s0MVx5YYrEK KCKeF5hvOregexIiZbIGUX
To continue,
multiplefiles/file32.txt:MLXiKZJ8KyHMOu9lYsz4ZjFYJSfB 14tregextpJ
grep file32 2_searching_patterns.txt
multiplefiles/file32.txt:veFQUregexfnQxwQw6POJRNvvAeYwToX6ptvN39m
multiplefiles/file32.txt:chONvregexiGjHkmptPVTjOzvWVGbrGoHoywV4Vy
```

Regular Expressions

Regular expressions allow to search for more complex patterns.

Here are some simple regular expression examples:

regex	description
.	matches any character
[A-Za-z]	matches any of the characters within parenthesis
.*	matches any character, any number of times

The FASTA format

Open the file `data/genes/sequences.fasta`

This is a FASTA file, a format used for DNA/RNA and protein sequences.

In [24]:

```
less genes/sequences.fasta
```

```
>seq000 sequence description
das$myvarCGNTTTNTAATTATATANNTAGCGTGATCC
>seq001 sequence description
NNCGNANAGCTNGACCTAGTTGAAATTGTG
>seq002 sequence description
CGGTATGCAGCCCNNGCGTNGCNTNAATNA
>seq003 sequence description
CGCNTCNTNCTTCAACGACGCTTTTANC
>seq004 sequence description
CGTNGCGNNGGAGCCATTANTAGNNCCT
```

Exercise - search for patterns in FASTA files

Use grep to identify all the sequences containing three As, followed by any two characters, followed by three Ts.

- AAA AA TTT
- AAA AG TTT
- ..
- AAA zz TTT

In [25]:

```
grep 'AAA..TTT' genes/sequences.fasta
```

```
TCGACCTGCNNANGGCTNTGTAAAccTTTG
TNCGTCCGGCCAAAAAAgTTTNGNGCTTG
NAAGTTTAAGAAAaTTTCATGAGCNCGAG
NCTGCGCNGCGAGCCACAACNAAGtTTTG
TAGNGTACTCCTTNTNAAAaTTTtTCTCG
ACTNGACGCTTCNCTNAACcTTTGCNTNT
GGGNAAAaTTTCGGCTGNNTGCNCNNTNN
TNGAACCCNGTNCGGTCAACcTTTTCNTA
NGGTACAAAGCNCCANNNTTAAAgTTTC
AAAcTTTNCNGCCNTNCTCANNCGTGNTT
NTTCACTCAAActTTTNGNGNATTNGGAAT
GGATGAAAaTTTNCCAGCCAANNCTTGNA
```

Bonus: use the `-B 1` grep option to retrieve the names of these sequences:

In [26]:

```
grep -B1 'AAA..TTT' genes/sequences.fasta
```

```
>seq009 sequence description
TCGACCTGCNNANGGCTNTGTAAAccTTTG
--
>seq012 sequence description      _____
TNCGTCCGGCCAAAAAAgTTTNGNGCTTG
--
>seq024 sequence description      / Congrats! This \
NAAGTTTAAGAAAaTTTCATGAGCNCGAG
--
>seq025 sequence description      | was the last      |
NCTGCGCNGCGAGCCACAACNAAGtTTTG
--
>seq026 sequence description      \ grep exercise /
TAGNGTACTCCTTNTNAAAAaTTTTCCTCG
--
>seq030 sequence description      -----
ACTNGACGCTTCNCTNAAAccTTTGCNTNT
--
>seq032 sequence description      \  ^__^
GGGNAAAaTTTCGGCTGNNTGCNCNNTNN
--
>seq033 sequence description      \ (oo)\_____
TNGAACCCNGTNCGGTCAAAccTTTTCNTA
--
>seq038 sequence description      (__) \       )\/\
NGGTACAAAGCNCCANNNTTAAAgTTTC
--
>seq043 sequence description      ||----w  |
AAAccTTTNCNGCCNTNCTCANNCGTGNTT
--
>seq046 sequence description      ||       ||
NTTCACTCAAActTTTNGNGNATTNGGAAT
--
>seq049 sequence description
GGATGAAAaTTTNCCAGCCAANNCTTGNA
```

Searching for special characters, such as ">"

Look at the output of the previous exercise. Is there a way to print only the lines that contain the > character?

We need to be careful because > is also the redirection symbol - we may accidentally overwrite some file.

The solution in this case to use single or double quotes: '>' to search for a literal > :

In [27]:

```
# To search for a literal >, remember to quote it correctly:
grep -B1 'AAA..TTT' genes/sequences.fasta | grep ">"
```

```
>seq009 sequence description
>seq012 sequence description
>seq024 sequence description / Congrats! This \
>seq025 sequence description | was the last |
>seq026 sequence description \ grep exercise /
>seq030 sequence description -----
>seq032 sequence description \ ^ ^
>seq033 sequence description \ (oo)\ _____
>seq038 sequence description ( )\ )\ /\
>seq043 sequence description ||----w |
>seq046 sequence description ||      ||
>seq049 sequence description
```

cut, sort and uniq

`cut`, `sort` and `uniq` are three Unix commands frequently used to work with tabular files.

- `cut` : Extract columns from a TSV or CSV file
- `sort` : Sort a file alphabetically or numerically
- `uniq` : Remove duplicated lines in a sorted file

Example cut, sort and uniq

The previous exercises printed the header lines of all the sequences in a FASTA file matching `AAA..TTT` :

```
grep -B1 'AAA..TTT' genes/sequences.fasta | grep ">"
```

Let's add a `cut` statement to extract the first column.

In [28]:

```
# cut options: -f 1 (first column), -d " " (specify that the field delimiter is
an empty space)
grep -B1 'AAA..TTT' genes/sequences.fasta | grep ">" | cut -f 1 -d" "
```

```
>seq009
>seq012
>seq024
>seq025
>seq026
>seq030
>seq032
>seq033
>seq038
>seq043
>seq046
>seq049
```

the SAM and BAM formats

The exercises on cut, sort and uniq are based on the **SAM** format, used for storing NGS data.

SAM stands for Sequence Alignment/Map format (<https://samtools.github.io/hts-specs/SAMv1.pdf> (<https://samtools.github.io/hts-specs/SAMv1.pdf>)). SAM files contains the output of the alignment of sequences or short reads to a reference genome.

The Introduction to NGS session of this course will explore the SAM format in details, and teach you how it is generated from sequencing data.

Here we will have a quick preview to it, to help you familiarize with the Unix tools used to handle NGS data

Example SAM file

The `exercises/ngs` folder contains a sample SAM file, `1000G_HG00154_sample.sam` :

In [29]:

```
head ngs/1000G_HG00154_sample.sam
```

```
@HD      VN:1.0   GO:none SO:coordinate
@SQ      SN:1     LN:249250621   AS:NCBI37       UR:ftp://ftp.1000gen
omes.ebi.ac.uk/vol1/ftp/technical/reference/human_g1k_v37.fasta.gz
M5:1b22b98cdeb4a9304cb5d48026a85128   SP:Human
@SQ      SN:2     LN:243199373   AS:NCBI37       UR:ftp://ftp.1000gen
omes.ebi.ac.uk/vol1/ftp/technical/reference/human_g1k_v37.fasta.gz
M5:a0d9851da00400dec1098a9255ac712e   SP:Human
@SQ      SN:3     LN:198022430   AS:NCBI37       UR:ftp://ftp.1000gen
omes.ebi.ac.uk/vol1/ftp/technical/reference/human_g1k_v37.fasta.gz
M5:fdfd811849cc2fadebc929bb925902e5   SP:Human
@SQ      SN:4     LN:191154276   AS:NCBI37       UR:ftp://ftp.1000gen
omes.ebi.ac.uk/vol1/ftp/technical/reference/human_g1k_v37.fasta.gz
M5:23dccd106897542ad87d2765d28a19a1   SP:Human
@SQ      SN:5     LN:180915260   AS:NCBI37       UR:ftp://ftp.1000gen
omes.ebi.ac.uk/vol1/ftp/technical/reference/human_g1k_v37.fasta.gz
M5:0740173db9ffd264d728f32784845cd7   SP:Human
@SQ      SN:6     LN:171115067   AS:NCBI37       UR:ftp://ftp.1000gen
omes.ebi.ac.uk/vol1/ftp/technical/reference/human_g1k_v37.fasta.gz
M5:1d3a93a248d92a729ee764823acbbc6b   SP:Human
@SQ      SN:7     LN:159138663   AS:NCBI37       UR:ftp://ftp.1000gen
omes.ebi.ac.uk/vol1/ftp/technical/reference/human_g1k_v37.fasta.gz
M5:618366e953d6aaad97dbe4777c29375e   SP:Human
@SQ      SN:8     LN:146364022   AS:NCBI37       UR:ftp://ftp.1000gen
omes.ebi.ac.uk/vol1/ftp/technical/reference/human_g1k_v37.fasta.gz
M5:96f514a9929e410c6651697bded59aec   SP:Human
@SQ      SN:9     LN:141213431   AS:NCBI37       UR:ftp://ftp.1000gen
omes.ebi.ac.uk/vol1/ftp/technical/reference/human_g1k_v37.fasta.gz
M5:3e273117f15e0a400f01055d9f393768   SP:Human
```

Format of a SAM file

A SAM file is composed by a header section (beginning of the file, lines starting with "@") and an alignment section (rest of the file)

SAM Headers

To extract the headers from a SAM file, we can use grep with the `^@` regular expression.

This expression `^@` matches all the lines that start with a `@` character. Note that simply searching for `@` would not be enough, because the alignment section also contains this character.

In [30]:

```
grep "@" ngs/1000G_HG00154_sample.sam | less
```

```
@HD      VN:1.0  GO:none SO:coordinate
@SQ      SN:1    LN:249250621   AS:NCBI37      UR:ftp://ftp.1000gen
omes.ebi.ac.uk/vol1/ftp/technical/reference/human_g1k_v37.fasta.gz
M5:1b22b98cdeb4a9304cb5d48026a85128      SP:Human
@SQ      SN:2    LN:243199373   AS:NCBI37      UR:ftp://ftp.1000gen
omes.ebi.ac.uk/vol1/ftp/technical/reference/human_g1k_v37.fasta.gz
M5:a0d9851da00400dec1098a9255ac712e      SP:Human
@SQ      SN:3    LN:198022430   AS:NCBI37      UR:ftp://ftp.1000gen
omes.ebi.ac.uk/vol1/ftp/technical/reference/human_g1k_v37.fasta.gz
M5:fdfd811849cc2fadebc929bb925902e5      SP:Human
@SQ      SN:4    LN:191154276   AS:NCBI37      UR:ftp://ftp.1000gen
omes.ebi.ac.uk/vol1/ftp/technical/reference/human_g1k_v37.fasta.gz
M5:23ccd106897542ad87d2765d28a19a1      SP:Human
@SQ      SN:5    LN:180915260   AS:NCBI37      UR:ftp://ftp.1000gen
omes.ebi.ac.uk/vol1/ftp/technical/reference/human_g1k_v37.fasta.gz
M5:0740173db9ffd264d728f32784845cd7      SP:Human
@SQ      SN:6    LN:171115067   AS:NCBI37      UR:ftp://ftp.1000gen
omes.ebi.ac.uk/vol1/ftp/technical/reference/human_g1k_v37.fasta.gz
M5:1d3a93a248d92a729ee764823acbbc6b      SP:Human
@SQ      SN:7    LN:159138663   AS:NCBI37      UR:ftp://ftp.1000gen
omes.ebi.ac.uk/vol1/ftp/technical/reference/human_g1k_v37.fasta.gz
M5:618366e953d6aaad97dbe4777c29375e      SP:Human
@SQ      SN:8    LN:146364022   AS:NCBI37      UR:ftp://ftp.1000gen
omes.ebi.ac.uk/vol1/ftp/technical/reference/human_g1k_v37.fasta.gz
M5:96f514a9929e410c6651697bde59aec      SP:Human
@SQ      SN:9    LN:141213431   AS:NCBI37      UR:ftp://ftp.1000gen
omes.ebi.ac.uk/vol1/ftp/technical/reference/human_g1k_v37.fasta.gz
M5:3e273117f15e0a400f01055d9f393768      SP:Human
```

In [31]:

```
# Grepping for "@" is not enough, because some of the Alignment lines contain the same character
grep "^@" ngs/1000G_HG00154_sample.sam | less
```

```
@HD      VN:1.0   GO:none SO:coordinate
@SQ      SN:1    LN:249250621   AS:NCBI37      UR:ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/technical/reference/human_g1k_v37.fasta.gz
M5:1b22b98cdeb4a9304cb5d48026a85128      SP:Human
@SQ      SN:2    LN:243199373   AS:NCBI37      UR:ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/technical/reference/human_g1k_v37.fasta.gz
M5:a0d9851da00400dec1098a9255ac712e      SP:Human
@SQ      SN:3    LN:198022430   AS:NCBI37      UR:ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/technical/reference/human_g1k_v37.fasta.gz
M5:fdfd811849cc2fadebc929bb925902e5     SP:Human
@SQ      SN:4    LN:191154276   AS:NCBI37      UR:ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/technical/reference/human_g1k_v37.fasta.gz
M5:23dccd106897542ad87d2765d28a19a1     SP:Human
@SQ      SN:5    LN:180915260   AS:NCBI37      UR:ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/technical/reference/human_g1k_v37.fasta.gz
M5:0740173db9ffd264d728f32784845cd7     SP:Human
@SQ      SN:6    LN:171115067   AS:NCBI37      UR:ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/technical/reference/human_g1k_v37.fasta.gz
M5:1d3a93a248d92a729ee764823acbbc6b     SP:Human
@SQ      SN:7    LN:159138663   AS:NCBI37      UR:ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/technical/reference/human_g1k_v37.fasta.gz
M5:618366e953d6aaad97dbe4777c29375e     SP:Human
@SQ      SN:8    LN:146364022   AS:NCBI37      UR:ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/technical/reference/human_g1k_v37.fasta.gz
M5:96f514a9929e410c6651697bded59aec     SP:Human
@SQ      SN:9    LN:141213431   AS:NCBI37      UR:ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/technical/reference/human_g1k_v37.fasta.gz
M5:3e273117f15e0a400f01055d9f393768     SP:Human
```

SAM - Alignment section

The Intro to NGS session of this workshop will explain the headers section more in detail.

Let's focus on the Alignment section of the SAM file. Which grep option can be used to print the lines that DO NOT start with a @ character?

In [32]:

```
grep -v "^@" ngs/1000G_HG00154_sample.sam | less
```

```
ERR018419.14029113      147      21      33031576      60      6S70
M      =      33031178      -467      CAAGACAAAAAATGAAAAC TACAAAAG
CATCCATCTTGGGGCGTCCCAATTGCTGAGTAACAAATGAGACGCTGT      #####CACC
C:34ACCA7470BBBB?<@7@@@>=BA@@@@8=A@@?BABA@>A@B?@B@?BBAB@A?@7?@?><
X0:i:1  X1:i:0  XC:i:70 MD:Z:70 RG:Z:ERR018419  AM:i:37 NM:i:0 SM:
i:37    MQ:i:60 XT:A:U  BQ:Z:@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
ERR018419.19261557      83      21      33031576      60      76M
=      33031160      -491      AAAAAAATGAAAAC TACAAAAGCATCCTTCTTGGGG
CGTCCCAATTGCTGAGTAACGAATGAGACGCTGTGGCCAA      ?CCCCBB@CBCA@A@@BBC
B46639/%53)@@@@?8?B@@@@BA@B??B@@4<;?5#<=:0A@A8@:B?BA?<?B>      X0:
i:1      X1:i:0  MD:Z:27A28A19  RG:Z:ERR018419  AM:i:37 NM:i:2 SM:
i:37    MQ:i:60 XT:A:U  BQ:Z:BEEDBA@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@DC
ERR018418.15483761      99      21      33031580      60      71M5
S      =      33032025      486      AAATGAAAAC TACAAAAGCATCCATCTT
GGGGCGTCCCAATTGCTGAGTAACAAATGAGACGCTGTGGCCAACTC      ;AAA@BBAB?B
A?BBCB@@AA@@AA@BB@8=<5@?@@: B9B@:B@=@@AB?ABBA;>:==2<B:6733?#####
X0:i:1  X1:i:0  XC:i:71 MD:Z:70A0      RG:Z:ERR018418  AM:i:37 NM:
i:1      SM:i:37 MQ:i:60 XT:A:U  BQ:Z:@FC@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
ERR018420.32040701      83      21      33031606      60      76M
=      33031211      -470      TTGGGGCGTCCCAATTGCTGAGTAACAAATGAGACG
CTGTGGCCAAACTCAGTCATAACTAATGACATTTCTAGAC      @A@A@A8<AA@?B@AA@@A@
A??B@@BBAA?A?@8?@A>A??@@BB@@B@A>B@AAB@@ABAA@@@ABBBABB?@@      X0:
i:1      X1:i:0  MD:Z:76 RG:Z:ERR018420  AM:i:37 NM:i:0 SM:i:37 MQ:
i:60    XT:A:U  BQ:Z:@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
ERR018420.24362571      147      21      33031608      60      14S6
2M      =      33031194      -475      AACGCATCCGTCTTGGGGCGTCCCAATT
GCTGAGTAACAAATGAGACGCTGTGGCCAAACTCAGTCATAACTAATG      #####
###<>;7;;@??A@8>@@?A;?B??A@='A;66?>9>A??=?B?@A>A>B@@AB@@AB@A<
X0:i:1  X1:i:0  XC:i:62 MD:Z:62 RG:Z:ERR018420  AM:i:37 NM:i:0 SM:
i:37    MQ:i:60 XT:A:U  BQ:Z:@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
ERR018418.8551258      147      21      33031642      60      76M
=      33031240      -477      CTGTGGCCAAACTCAGTCATAACTAATGACATTTCT
AGACAAAGTGACTTCAGATTTTCAAAGCGTACCCTGTTTA      BB9B@=@?BBAAA@B>B@AA
B@@A@AA?@@ABBB@AB@@@BBA=B?@@BB@B@A@BAB@BBA@8?A?AA@A>BAA<      X0:
i:1      X1:i:0  MD:Z:76 RG:Z:ERR018418  AM:i:37 NM:i:0 SM:i:37 MQ:
i:60    XT:A:U  BQ:Z:@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
ERR018418.12789017      99      21      33031654      60      76M
=      33032063      484      TCAGTCATAACTAATGACATTTCTAGACAAAGTGAC
TTCAGATTTTCAAAGCGTACCCTGTTTACATCATTTTGCC      ??A@A@B@AB?BABA@B?AA
BB@BA@B>ABB@@@B>BB?A@BABBB@ABBA@8@@?@@B@AABB?BB@BBCCA<B      X0:
i:1      X1:i:0  MD:Z:76 RG:Z:ERR018418  AM:i:37 NM:i:0 SM:i:37 MQ:
i:60    XT:A:U  BQ:Z:@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
ERR018418.9132383      99      21      33031729      60      36M4
0S      =      33032168      503      CAATTTTCGCGTACTGCAACCGGCGGGGCC
ACGCCCCCGTGGAAGAAGGGTGTTTTCTCCCCCTTTCGGGGGTTTCGG      ?AAACB@7@9A
A?B@@AB?@8@@8AA<@+855<<<#####
X0:i:1  X1:i:0  XC:i:36 MD:Z:36 RG:Z:ERR018418  AM:i:37 NM:i:0 SM:
i:37    MQ:i:60 XT:A:U  BQ:Z:@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
ERR018419.8913816      83      21      33031868      60      41S3
5M      =      33031428      -474      GGGGAGCCCCCGGCGCCCGCGCCCCCTT
GGCCCCGCCCCCGGTCTTTCCCGGCCACTCGCGACCCGAGGCTGCCGC      #####
#####;4;=<;8@<:198
X0:i:1  X1:i:0  XC:i:35 MD:Z:3A31      RG:Z:ERR018419  AM:i:37 NM:
```

```

i:1      SM:i:37 MQ:i:60 XT:A:U  BQ:Z:@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
ERR018419.25794159      147      21      33031877      60      41S3
5M      =      33031430      -481      CGGGCGCCCGGCCCCCTTGCNCCAGCC
CGAGGCCATTCCCGGCCACTCGCGGCCCGCGGCTGCCGCAGGGGGCGG      #####
#####!#####73-)2.05;</;=41:;;>+;;
X0:i:1  X1:i:0  XC:i:35 MD:Z:11A4A18      RG:Z:ERR018419  AM:i:25 NM:
i:2      SM:i:25 MQ:i:60 XT:A:U  BQ:Z:@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

```

Using cut to print the first columns of the SAM file

The SAM file is very wide - it doesn't fit into the screen.

We can use cut to print only a subset of columns:

In [33]:

```
grep -v "^@" ngs/1000G_HG00154_sample.sam | cut -f 1,2,3,4,5,6,7 | less
```

```

ERR018419.14029113      147      21      33031576      60      6S70
M      =
ERR018419.19261557      83      21      33031576      60      76M
=
ERR018418.15483761      99      21      33031580      60      71M5
S      =
ERR018420.32040701      83      21      33031606      60      76M
=
ERR018420.24362571      147      21      33031608      60      14S6
2M      =
ERR018418.8551258      147      21      33031642      60      76M
=
ERR018418.12789017      99      21      33031654      60      76M
=
ERR018418.9132383      99      21      33031729      60      36M4
OS      =
ERR018419.8913816      83      21      33031868      60      41S3
5M      =
ERR018419.25794159      147      21      33031877      60      41S3
5M      =

```

Using *cut* to extract columns from a file

The alignment section of a SAM file contains information on the mapping of short sequencing reads to a reference genome. Here is a description of each column, taken from <https://seqan.readthedocs.io/en/seqan-v1.4.2/Tutorial/BasicSamBamIO.html> (<https://seqan.readthedocs.io/en/seqan-v1.4.2/Tutorial/BasicSamBamIO.html>).

Col	Field	Type	N/A Value	Description
1	QNAME	string	mandatory	The query/read name.
2	FLAG	int	mandatory	The record's flag.
3	RNAME	string	*	The reference name.
4	POS	32-bit int	0	1-based position on the reference.
5	MAPQ	8-bit int	255	The mapping quality.
6	CIGAR	string	*	The CIGAR string of the alignment.
7	RNEXT	string	*	The reference of the next mate/segment.
8	PNEXT	string	0	The position of the next mate/segment.
9	TLEN	string	0	The observed length of the template.
10	SEQ	string	*	The query/read sequence.
11	QUAL	string	*	The ASCII PHRED-encoded base qualities.

SAM FLAGS are explained in this page: <https://broadinstitute.github.io/picard/explain-flags.html> (<https://broadinstitute.github.io/picard/explain-flags.html>).

In [34]:

```
# Printing the chromosome and position for each alignment
grep -v "^@" ngs/1000G_HG00154_sample.sam | cut -f 3,4 | less
```

```
21      33031576
21      33031576
21      33031580
21      33031606
21      33031608
21      33031642
21      33031654
21      33031729
21      33031868
21      33031877
```


Exercise! Fun with flags

The FLAG column allows to identify properties of a short-read alignment.

How many distinct FLAG values there are in this file?

- Use `grep` to print the alignment section of `ngs/1000G_HG00154_sample.sam` (remove the headers)
- Use `cut` to print the FLAG column (#2)
- Use `uniq` to eliminate duplicate lines. Add the `-c` option to `uniq` to count the number of duplicate lines
- If the output does not look correct, add a `sort` statement between `cut` and `uniq`

In [35]:

```
grep -v "^@" ngs/1000G_HG00154_sample.sam | cut -f 2 | uniq -c | head
```

```
1 147
1 83
1 99
1 83
2 147
2 99
1 83
3 147
1 83
3 147
```

In [39]:

```
grep -v "^@" ngs/1000G_HG00154_sample.sam | cut -f 2 | sort -n | uniq -c | head
```

```
16 0
12 16
5 73
3 81
296 83
6 97
314 99
3 113
1 117
3 121
```

Exercise! Extract chromosome and start position for this file

This file contains data from multiple chromosomes. How to extract which chromosomes are included, and the starting position?

- Use `grep` to print the alignment section of `ngs/1000G_HG00154_sample.sam` (remove the headers)
- Use `cut` to print the chromosome and position
- Sort the output by the second column. Use the `-n` option to specify that this is a numeric sort.
- Use `uniq` on the output. Check `man uniq`, and find the option to compare no more than 3 characters per row

In [37]:

```
grep -v "^@" ngs/1000G_HG00154_sample.sam | cut -f 3,4 | sort -n -k 2 | uniq -w 3
```

17	7512372
21	33031576

In [38]:

```
grep -v "^@" ngs/1000G_HG00154_sample.sam | cut -f 3,4 | sort -n -r -k 2 | uniq -w 3
```

21	33050212
17	7513450

In []:

Recap - grep, Unix piping, and cut

This afternoon we learned:

- **grep** extracts lines in a file that match a specific pattern or expression
- **the Unix Piping system** allows to concatenate commands together with `|`, `>` and `>>`
- **Genbank**, **FASTA** and **SAM** file formats
- **cut**, **sort** and **uniq** allow to extract columns and calculate unique values.

Example of Unix Approach

In the following days, when you will learn the basics of Python and R programming, you will be faced with the dilemma of how to organize your scripts.

You can use the Unix Philosophy as a guideline to have more versatile files and scripts.

Let's imagine we want to plot the CG content of a genome:

- **Approach 1 (non-Unix)**: write a single script that downloads the genome, calculates the GC content, and draws a plot
- **Approach 2 (Unix)**: write three separate scripts or functions, then pipe them together using the Unix Pipe or with a wrapper script.

The Unix Approach requires a bit of extra work, but it is more versatile in case you want to reuse the same scripts.

Time for a break!

In []: