#### Workshop on Genomics 2025





Mercè Montoliu Nerín

January 7th,2025



### What is UNIX?

powerful

#### **Operating system**

#### multi-user

multitasking



# Why is it important for bioinformatics?

Facilitates sharing and reproducing analyses

Handling large datasets and running analyses efficiently

Efficiency and speed

Access to powerful tools and applications

Using scripts to automate repetitive tasks



The terminal



Make it comfortable to work in

- Resize the window
- Change the font size
- Open multiple terminal windows (or tabs)
- Make sure you have the right combination of colours that work for you.





drwxr-xr-x@

drwxr-xr-x@ drwxr-xr-x@

drwxr-xr-x0 drwxr-xr-x0

drwxr-xr-x@

drwxr-xr-x@

drwxr-xr-x@

drwxr-xr-x@ 342 apple staff 11K 21 Au drwxr-xr-x@ 13 apple staff 416B 21 Au 8 apple staff

3 apple staff

3 apple staff

drwxr-xr-x@ 10 apple staff 320B 28 No 9 apple staff 288B 14 J

drwxr-xr-x@ 55 apple staff 1.7K 20 J 3 apple staff 96B 20 J

drwxr-xr-x@ 20 apple staff 640B 25 J drwxr-xr-x@ 29 apple staff

drwxr-xr-x0 22 apple staff 704B 5 Ap drwxr-xr-x@ 21 apple staff 672B 5 Ap drwxr-xr-x@ 11 apple staff 352B 7 Ap -rw-r--r--@ 2 apple staff 0B 4 Ma drwxr-xr-x0 33 apple staff 1.0K 7 Mag drwxr-xr-x0 430 apple staff 13K 15 Mag apples-MacBook-Pro.local@apple[Dropbox]\$

drwxr-xr-x0 4 apple staff

drwxr-xr-x0 3 apple staff

drwxr-xr-x0 38 apple staff

drwxr-xr-x0 11 apple staff

drwxr-xr-x@ 2891 apple staff

drwxr-xr-x@ 3 apple staff

drwxr-xr-x0 13 apple staff

drwxr-xr-x0 3 apple staff

drwxr-xr-x@ 43 apple staff 1.3K 21 Au drwxr-xr-x@ 36 apple staff 1.1K 21 Au drwxr-xr-x0 5 apple staff 160B 21 Au drwxr-xr-x@ 38 apple staff 1.2K 21 Au drwxr-xr-x@ 3 apple staff 96B 21 Au drwxr-xr-x@ 11 apple staff 352B 21 Au drwxr-xr-x0 19 apple staff 608B 24 Au 9 apple staff 288B 24 Au

4 apple staff 128B 21 Au

6 apple staff 192B 1 Oc

drwxr-xr-x0 11 apple staff

256B 21 Au

352B 21 Au

96B 21 Au

96B 12 No

928B 25 J

128B 26 J

96B 30 Ja

1.2K 3 Ma

352B 3 Ma

90K 6 Ma

96B 19 Ma

416B 22 Ma

96B 3 Ap

#### The terminal

Conder - bab 2 2822 Boda Joan i Josie 22 de desembre de 2826 2 2822 Ancliament_divulgacio 2 2822 Ancliament_divulgacio 2 2822 Hercules 2 2822 Vercules 2 2822 Concurs_TGN 2 2822 Concurs_TGN 2 2822 Concurs_LGN	-ploogle Dritre Jaaoh +	(basc) [rental.ub/ncb/mst annotati SCO27 anholdes: 11 barc.cl SCO27 anholdes: 12 barc.cl SCO27 anholdes: 12 barc.cl Charlow Construction and the score score anholdes: 12 barc.cl Dentification and the score anholdes: 12 barc.cl Dentification anholdes: 12 barc.cl Dentific	why 1% is inverse hosts cit miner opts cit main_ren cit protomer stiller notification of the still stiller notification of the still stiller of the still start of the still for the still start of the still for the still start of the still for the still start of the still start of the start of the still start of the start of the sta
2 2022 SatCells_2019 2 2022 TM 2 2022 Public 2 2022 Public 2 2022 Veus procel 2 2022 Veus procel 2 2022 Veus procel 2 2022 Veus proces 2 2020 Veus proces 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2		(a) (b) (b) (c) (c) (c) (c) (c) (c) (c) (c	all-of_sets/sets/sets.pr all-of_sets/sets/sets.pr all-of_sets/sets/sets.pr all-of_sets/sets.pr all-of_sets/sets.pr all-of_sets/sets.pr all-of_sets/sets.pr all-of_sets/sets/sets.pr all-of_sets/sets/sets all-of_sets/sets/sets all-of_sets/sets/sets all-of_sets/sets/sets all-of_sets/sets/sets all-of_sets/sets/sets all-of_sets/sets/sets all-of_sets/sets/sets all-of_sets/sets/sets all-of_sets/sets/sets/sets/sets/sets/sets/sets
14/18 GIZquierdo_PhD 20:13 seafan 12:05 Fotos_ocells_text_Borja 13:07 Lobsters_Josie 15:21 Camera Uploads 10:21 World bird list_JFerrer 20:59 Sequències-2 09:56 Prizes 12:09 Potadoo_Milan 12:29 Potadoo_Milan 09:28 papers_reviewed 11:24 Icon 11:24 Icon 11:24 Icon 11:24 Screenshots	1000000000000000000000000000000000000		
		NY AN To real 1.177 0.172	000002127, 47.81 00002127, 47.81 00002127, 47.81 00002127, 47.81 00002127, 47.81 00000217, 47.81 00000217, 47.81 00000017, 47.81 000000000000000000000000000000000000

🗯 Spotify File E	dit View Playback Window H	elp		* 5* 🛆 🍘 🗉	1911 <b>?</b>	🕚 🔍 🚍 👫 Sat 13 May	15:29
eee 🗎 unix	_session — ssh -X montoliu@rackham	uppmax.uu.se — 115×30	To o o merceme	ontoliu — ssh -X merce@m	olev199.icm	.uu.se — 121×30	
	nts/evomics/unix/unix_session — ssh -X mo	ntollu@rackham.uppmax.uu.se	+	~ ssh -X merce@mole			
TTT AL ALVEL	/ ) )///   Surter: packhon1		D insularis busco	D suciona papar con buto		011 8-1-2 81 forte er	
	- /   User: montoliu		D_insularis.fna D_insularis.illumina_R1.fastq.gz D_insularis_illumina_R2.fastq.gz	D_sucinea_pepper_cor.fost D_sucinea_pepper_cor.fost D_sucinea_pilon1.fosta D_sucinea_pilon2.fosta		011_0-1-2_R2.fostq.gz 011_0-11_R1.fostq.gz 011_10-11_R2.fostq.gz	
*****	*******	*****	D_insularis_nextdenovo_busco	D_sucinea_pilon.fosta		011_18-20_R1.rustq.gz 011_18-20_R2.fastq.gz	
User Guides: http:// FAQ: http://www.uppm	www.uppmax.uu.se/support/user-guides ax.uu.se/support/faq		D_insularis_nextdenovo.fasta D_insularis_pl_busco D_insularis_p2_busco	D_sucinea_stats D_tropicalis_illumina_R1. D_tropicalis_illumina_R2.		011_3-4_R1.fastq.gz 011_3-4_R2.fastq.gz 011_illumina_R1.fastq.gz	
Write to support@upp	max.uu.se, if you have questions or comm	ents.	D_insularis_pepper_cor_busco D_insularis_pepper_cor.fasta D_insularis_pilon1.fasta	D_tropicalis_nextdenovo_b D_tropicalis_nextdenovo.f D_tropicalis_pepper_cor_b	usco usco	011_illumina_R2.fastq.gz 011_nanopore.fastq.gz 011_nextdenovo_busco	
base) [montoliu@rackham1 –] base) [montoliu@rackham1 me	\$ fly rcc]\$ ls	ecoffo1d	D_insularis_pilon2.fasta D_insularis_pilon3.fasta D_insularis_pilon.fasta D_insularis_pilon.fasta	D_tropicalis_ptpper_cor.f D_tropicalis_pilon1.fasta D_tropicalis_pilon2.fasta	asta	011_nextdenovo.Fasta 011_p1_busco 011_p2_busco 011_nextenses_com_busco	
ssemblies cactus base) [montoliu@rackham1 me base) [montoliu@rackham1 an	inspector polish run_assemblies rrcc]\$ cd annotation/ motation]\$ 1s		D_paulistorum_L06_busco D_paulistorum_L06.fna D_paulistorum_L06.ilumine_R1.fasta.ex	D_tropicalis_pilon.fasta D_tropicalis_busco D_tropicallis_fma		011_pepper_cor.fasta 011_pepper_cor.fasta 011_pilon1.fasta 011_pilon2.fasta	
02207_orthologues brak 02207_orthologues_all data issemblies gene base) [nontoliu@rackhaml an	ter_out maker_bopts.ctl maker_opts base maker_evm.ctl proteome nnotation maker_exe.ctl repeats motation]5 cd assemblies/ unablimit la	.ctl RNLdato STAR_index RNLmap tmp scripts	D_paulistorum_L06_tllumina_R2.fastq.gi D_paulistorum_L06_nextdenovo_busco D_paulistorum_L06_nextdenovo_fasta D_paulistorum_L06_p1_busco D_paulistorum_L06_p1_busco	E D_tropicallis_nanopore.fa D_willistoni_00_illumina_ D_willistoni_00_illumina_ D_willistoni_00_nanopore. D_willistoni_00_nanopore.	stq.gz R1.fastq.gz R2.fastq.gz fastq.gz	011_ptlon3.fasta 011_pilon.fasta 011_stats 01d_busco	
28.fasta D.a 28.fasta D.a 28.masked.fasta D.s Lequinoxialis.fasta D.s Linsularis.fasta D_t _aquiistorum.106.fasta D_t	sawiistorum_L12.fasta p.fasta D.willistori_L63.f p.fasta EOU.masked.fasta acinea.fasta INS.masked.fasta illiston.le0.fasta UNS.masked.fasta	asta 011.fasta TRO.masked.fasta 011.new.fasta WMM.masked.fasta 01d.assemblies W.G3.masked.fasta SP_masked.fasta SP_masked.fasta	D_paulistorws_L06_pepper_cor_busco D_paulistorws_L06_pepper_cor_busco D_paulistorws_L06_pilon1, fosta D_paulistorws_L06_pilon2, fosta D_paulistorws_L06_pilon3, fosta	D_willistoni_00_nextdenov D_willistoni_00_pepper_co D_willistoni_00_pilon1.fo D_willistoni_00_pilon1.fo D_willistoni_00_pilon2.fo	o.fasta ir_busco ir_fasta ista ista	<pre>proteins quast_all_fna.sh quast_all.sh quast_new2W23.sh quast_new2W23.sh quast_new2W23.sh</pre>	
base) [montoliu@rackham1 as	semblies]\$		(base) merce@molev199:/space/no_backup	a/merce/poulistorum>			
	mito — -zsh — 115×	20	• • • • • mercemo	ontoliu — ssh -X merce@m	iolev199.icm	.uu.se — 121×20	
	-/Documents/postdoc/mito -	zsh -	+	~ — ssh -X merce@mole	v199.icm.uu.se	2	-
66 66, frozo, bed 66, frozo, bed 12, frozo, bed 12, frozo, bed 12, frozo, bed 5, frozo, bed 5, frozo, bed 5, teo, 21, frozo 110, providents, zip 110, providentos	011_cmd_beta7_fasts.nhr 011_cmd_beta7_fasts.nhr 011_cmd_beta7_fasts.nng 011_cmd_beta7_fasts.nng 011_cmd_beta7_fasts.nns 011_cmd_beta7_fasts.nns 011_cmd_beta7_fasts.nnf 011_cmd_beta7_fasts.nnf 011_cmd_beta7_fasts.nnf 011_cmd_beta7_fasts.nnf 011_cmd_beta7_fasts.nnf 011_cmd_beta7_fasts.nnf 011_cmd_beta7_fasts.nnf 011_cmd_beta7_fasts.nnf 011_cmd_beta7_fasts.nnf	TBD_fracbed TBD_artacles REQ_rest.bed REQ_rest.bed REG_fracbed REG_Fracbed REG	NS. fasts.mtf         I           NS. fasts.mtg         I           01dipbol.anno.05. one.tob         I           01dipbol.fasts.nb         I           01dipbol.fasts.nb<	Dilmitol2:/dsta.db Dilmitol2:/dsta.db Dilmitol2:/dsta.dn Dilmitol2:/dsta.nd Dilmitol2:/dsta.nds Dilmitol2:/dsta.nds Dilmitol2:/dsta.nds Dilmitol2:/dsta.nds Dilmitol2:/dsta.nds Dilmitol2:/dsta.nds Dilmitol2:/dsta.nds Dilmitol2:/dsta.nds Dimercifito/fitors di	011_mito4.fea 011_mito4.fea 011_mito5_enr 011_mito5_enr 011_mito5_enr 011_mito5_fea 011_mito5.fea 011_mito5.fea 011_mito5.fea	tat.mag 01_nito9.fasta tat.mft 01_nito9.fasta.mb tat.mto 01_nito9.fasta.mb 02_minos.tab 01_nito9.fasta.mi 02_minos.tab 01_nito9.fasta.me tat.mb 01_nito9.fasta.me tat.min 01_nito9.fasta.me tat.min 01_nito9.fasta.mo	
11.fasta 41.fasta.fai 11.fasta.ndb 11.fasta.nhr 11.fasta.nin	011_gypsy_rep.bed 011_hAT_rep.bed 011_helitron_rep.bed 011_mito_rep2.bed	visualize.for_meeting -Smito_selection.xlsx -Spsmudogenes.xlsx	anyotics black-results gover_and (base) mercefnolev193:/spoce/no_backup (base) mercefnolev193:/spoce/no_backup (base) mercefnolev193:/spoce/no_backup comands.oll.sh extract.	repeats mito_genomes old_a p/merce/mitos cd scripts/ a/merce/mito/scripts> ls 940.sh extract lisst_seq_12mito.sh mummer.	_blost_seq.sh	er_assemblies_results re-assembly ads scripts other_assemblies.sh pre-re-assemble_940.sh	
			enolev199:/space/no_backu	/merce/mito/scripts> []			
			) 💮 🏶 📀	😒 🖪 🚨 📝 🚳	ligv 🛛 🛃		
undoneundoneundone aut	M_ 000 200 20	9990.13 January av. 30394	L 98_ +				



### File system organization





### Paths - Absolute vs Relative



. refers to our current location

. . refers to the location above us











### Paths - Absolute vs Relative





### Paths - Absolute vs Relative





### File system navigation

#### pwd - where am I?



cd - change directory





> cd ./Genomics





#### Key shortcuts

Ctrl + C	halts current command				
Ctrl + Shift + C	copy (linux)	-	Cmd + C (mac		
Ctrl + Shift + V	paste (linux)	-	Cmd + V (mac		
Ctrl + W	erases one word in current line				
Ctrl + U	erases whole line				
Ctrl + A	go to begining of line				
Ctrl + E	go to end of line				
Typeexit	log out of current session				



# Create, copy, move, and remove files and folders

mkdir - create new directory

cp - copy file

mv - move file or directory

rm - remove file

"Unix was not designed to stop its users from doing stupid things, as that would also stop them from doing clever things." - Doug Gwyn







man command - manual of the command



- Is -I formatted list
- Is -h "human" formatted list
- Is -Ih combination of flags



#### Inputs and outputs

- *stdin* It stands for standard input, and is used for taking text as an input.
- *stdout* It stands for standard output, and is used to text output of any command you type in the terminal, and then that output is stored in the stdout stream.
- *stderr* It stands for standard error. It is invoked whenever a command faces an error, then that error message gets stored in this data stream.



## stdin, stdout, stderr

command stdin

if it works: prints in our terminal the *stdout* if it fails: prints in our terminal the *stderr* 

command *stdin* > *stdout* 

if it works: *stdout* is redirected to a file if it fails: prints in our terminal the *stderr* 

command2 stdin2 > stdout

*stdout* is redirected to a file and rewrites its contents

command2 *stdin2* >> *stdout* 

stdout is redirected to a file and appended after its contents



## stdin, stdout, stderr

command stdin

if it works: prints in our terminal the *stdout* if it fails: prints in our terminal the *stderr* 

command file1 > output.txt *stdin stdout* 

if it works: *stdout* is redirected to a file if it fails: prints in our terminal the *stderr* 

command2 file2 > output.txtstdout is redirected to a file and rewrites itsstdinstdoutcontents

command2 file2 >> output.txtstdout is redirected to a file and appendedstdinstdoutafter its contents



# stdin, stdout, stderr

command file1 2> errors.txt if stdin stderr if

if it works: prints in our terminal the *stdout* if it fails: *stderr* is redirected to a file

command file1 &> output.txt redirects both stdout and stderr to a file
 stdin stdout&stderr

command file1 > output.txt 2> errors.txtredirects both stdout and stderr to astdinstdoutstderrseparate file each.



## Explore file content

- wc word count (-I lines, -c characters, -w words)
- less visualize file contents in your terminal screen (press q to exit)
- cat prints contents of your file as *standard output* in your terminal
- head visualize the first 10 lines of a file
- tail visualize the last 10 lines of a file

**The character | (pipe)** is used to concatenate commands, so that we can run one command after the other, avoiding the creation of intermediate files.

command1 input | command2 > output

#### Instead of :

command1 input > output1
command2 output1 > output2

Using pipe, the output of running *command1* on a given input gets directly piped into *command2*, and we obtain an output of these two consecutive commands, generating only one output.



cat - prints contents of your file as standard output in your terminal

#### redirect to a command

cat fileA | command > output.txt

concatenate files

cat fileA fileB >> fileC



cat fileA > fileC

cat fileB >> fileC



sort - puts in certain order a series of lines in our file

sort -r fileA sorts in reverse order

sort -n fileA sorts lines in fileA numerically

sort -k 2 fileA sort fileA by column 2

sort -k 2nr fileA sort fileA by column 2, numerically and in reverse order

sort -V fileA sort lines in fileA numerically natural.

sort -u fileA sort lines and removes duplicates -> sort fileA | uniq



#### Are these two files different?

diff - can tell us if there are differences between two files

diff -q fileA fileB

"Files fileA and fileC differ"

diff fileA fileB

prints differences





#### Splitting a file

**split -** split a given file into multiple files (default 1000)

split -I 20 fileA

produce x number of files from fileA, each containing 20 lines.

cut - extract specific parts of a file

cut -c 2 fileA

extract specific columns from a file



**Nano -** The simpler option of text editor. All commands within the nano text editor are given by pressing the Control-key, usually represented as ^

- ^S save current file
- ^O save to (a different file)
- ^X exit from nano



**Vim** - a highly configurable text editor built to make creating and changing any kind of text very efficient

- i start insert mode (you can start typing after where your cursor is)
- ESC exits insert mode (also Ctrl + C)
- :w save file without exiting
- :q exit file (if there are unsaved changes, it fails)
- :wq save and exit
- :q! exit without saving changes



**emacs** - a text editor characterized by its extensibility and configurability. Some essential commands commands get activated by typing Control + X, then the command (while holding the control key), but there is a wide range of key combinations to be used to move and edit the text

Ctrl + x + s save file

Ctrl + x + c exit editor (if not saved, it ask if you want to save, then type "yes")





#### Bash scripts

Shell scripts often have the suffix . sh

Shell scripts must be executable chmod 755 or chmod +x

Comments can be written in scripts with a #

Variables can be used to shorten long paths

Shell loops can be used to process lots of files

\ can be used to wrap long commands across multiple lines

#!/bin/bash must be the first line, it specifies interpreter


#### Bash scripts - Variables

We can save variables under almost any name.

Variables can be string type:

```
evomics="Workhop_on_genomics_2024"
```

```
data="genome_assembly_file.fasta"
```

path="/home/genomics/workshop\_materials/unix"

Integer type:

num=5

or float type:

pi=3.14

We can refer to the variables using a dollar sign: \$evomics \${evomics}



### Bash scripts - for loops

Loop over files inside a directory:

done

Loop over files that we stored inside a variable:

files="file1 file2 file3 file4" for file in \$files do commands \$file done



#### Bash scripts - while loops

#### while loops

while read line do

command \$line

done

We can pipe the command ls -l to this script to run the command on each of the files listed.



### Bash scripts - Stay informed!

If we want to print messages to the standard output while the script is running we can do that using the echo command. This is specially useful when running a long pipeline of multiple commands, so that we can keep track of the stage that is currently running.

```
for file in ./unix/working_directory/*fastq
do
        echo "Command 1 is running on $file"
        command1 $file
        echo "Command 2 is running on $file"
        command2 $file
done
```



#### Bash scripts - Some tricks!

```
for file in ./unix/working_directory/*fastq
```

do

done

```
file_name=$(basename $file ".fastq")
```

```
command1 $file -out ${file_name}_command1.fastq
```

It is the same as:

```
for file in ./unix/working_directory/*fastq
do
    file_name=$(basename $file ".fastq")
    command1 $file -out "$file_name"_command1.fastq
done
```





What my family and friends think I do







What my supervisor thinks I do

ChatGPT





#### Cheat-sheet

#### Workshop on Genomics 2024

pwd	show current path / directory		
ls	list directory		
cd dir	change to directory dir		
cd	change to home		
cd ~	change to home		
cd -	change to previous working directory		
	current directory		
	parent directory		
mkdir <i>dir</i>	create directory dir		
cp file1 file	e2 copy file1 to file2		
mv file1 fil	e2 move <i>file1</i> to <i>file2</i> or rename		
	file1 to file2		
rm file1	delete file1		
In -s file lir	nk create symbolic link		
	-		

wc	count (-I line	s, -w words, -c	characters)
tail <i>file</i>	output last 1	0 lines of file	
head <i>file</i>	output first 1	0 lines of file	
less <i>file /</i>	more file vi	isualize content	ts of file
cat <i>file</i>	o	utput file to star	ndard output
sort sort	rows	diff <i>fileA fileB</i>	differences?
uniq keej	o unique rows	cut -c 2	cut column 2

#### unix cheatsheet

man command chmod +x <i>file</i>	man mak	ual for command es <i>file</i> executable	
command1   co	mmand2	runs command2 on output of command1 download file into current dir	
ssh user@server		connect to cluster	
tar -xzf archive. tar -czf archive. gzip compress	tar.gz tar.gz <i>arch</i> .gz	uncompress tar.gz ive compress <i>archive</i> to .tar.gz gunzip uncompress .gz	z
Ctrl + C	halts curre	ent command	
Tab Arrow up	autocomplete current line		
Anow up	previous (	Sommanus	
Ctrl + Shift + C Ctrl + Shift + V	copy (linu	x) Cmd + C (mac) (x) Cmd + V (mac)	
	paste (int		
Ctrl + W	erases one word in current line		44
Ctrl + U	erases whole line		
Ctrl + A	go to begining of line		
Ctrl + E	go to end	of line	
exit	log out of current session		#0^1



#### Cheat-sheet





#### Rubber duck









# Bioinformatic Files

## Parsing and Editing



Regular expressions (or RegEx) are like special codes used to search for patterns in text.

Special characters in RegEx

We can match any character using regular expressions, except those that have a special meaning in RegEx.

The below listed characters are special characters in RegEx:





The dot (.) represents a single character, any single character. It can be a digit, a letter, a symbol, and even a space.

If we want to match our sequence "ATG", but this time we want to include the next nucleotide, we can do that with "ATG.", in this case, the character after 'ATG' was a 'T', so we find 'ATGT':

ATAGCATCAA<u>ATG</u>TAGCATTTACGTAGTAGCTATAGCTATTACGTAGGGCTACTTTATAGCATCAAATCTAGCATCTACG TAGCATCAAATCTAGCACGTACGTAGTAGCTC<u>ATG</u>CTATTACGTAGCGCAACAGCTCAACCTCAGGCTACTTTATAGCAT CAAATCTAGCATTA



The star (\*) means "zero or more occurrences of the previous character."

We can combine it with the dot in the following way ".\*". So we match zero or more occurrences of any character.

If you wanted to find all sequences that start with "ATG" and end with "TAA", you could use a regular expression like "ATG.\*TAA", which means "find 'ATG', followed by zero or more of any character, followed by 'TAA'":

ATAGCATCAA<u>ATG</u>TAGCATTTACGTAGTAGCTATAGCTATTACGTAGGGCTACTTTATAGCATCAAATCTAGCATCTACGTAGCATC AAATCTAGCACGTACGTAGTAGCTCATGCTATTACGTAGCGCAACAGCTCAACCTCAGGCTACTTTATATCTAGCAT<u>TAA</u>CGTAGTA



The plus (+) means "one or more occurrences of the previous character".

We can combine it with the dot in the following way ".+". So we match one or more occurrences of any character.

If you wanted to find all sequences that start with "ATG" and end with "TAA", you could use a regular expression like "ATG.+TAA", which means "find 'ATG', followed by one or more of any character, followed by 'TAA'":

ATAGCATCAA<u>ATG</u>TAGCATTTACGTAGTAGCTATAGCTATTACGTAGGGCTACTTTATAGCATCAAATCTAGCATCTACGTAGCATC AAATCTAGCACGTACGTAGTAGCTCATGCTATTACGTAGCGCAACAGCTCAACCTCAGGCTACTTTATATCTAGCAT<u>TAA</u>CGTAGTA



The plus (+) means "one or more occurrences of the previous character".

But, what would happen if we have this sequence? Can we use **ATG.+TAA** to match it? **ATAGCATCAA<u>ATGTAA</u>CATTTACGTAGTAGCTATAGCTATTACGTAGGGCTACTTTATAGCATCAAATCTAGCATCTACGTAGCATC AAATCTAGCACGTACGTAGTAGCTCATGCTATTACGTAGCGCAACAGCTCAACCTCAGGCTACTTTATAGCATCAAATCTAGCATC** 

"ATG.\*TAA" would be able to match it, but not "ATG.+TAA", as it requires that there is at least one character in between 'ATG' and 'TAA'



The question mark (?) matches zero or one time the previous character

If we want to match our sequence starting with "ATG" and ending with TAA, and we know sometimes there is a T after ATG, but sometimes not, we can do that with "ATGT?TAA", in this case, the character after 'ATG' can be a T, or can be nothing, and both the following sequences would be matched:

ATAGCATCAA<u>ATGTAA</u>CATTTACGTAGTAGCTATAGCTATTACGTAGGGCTACTTTATAGCATCAAATCTAGCATCTACGTAGCATC AAATCTAGCACGTACGTAGTAGCTCATGCTATTACGTAGCGCAACAGCTCAACCTCAGGCTACTTTATAGCATCAAATCTAGCATTC

ATAGCATCAA<u>ATG</u>T<u>TAA</u>CATTTACGTAGTAGCTATAGCTATTACGTAGGGCTACTTTATAGCATCAAATCTAGCATCTACGTAGCAT CAAATCTAGCACGTACGTAGTAGCTCATGCTATTACGTAGCGCAACAGCTCAACCTCAGGCTACTTTATAGCATCAAATCTAGCATT



The curly brackets ({}) can reference the amount of times we expect the previous character to

occur. It has three main configurations:

{m} - previous character exactly m number of times

{m,n} - previous character m to n number of times

{m,} - previous character m or more number of times

If we want to find 3 'A' in a row, we could use "A{3}"

#### ATAGCATCATAATGTAGCATTTACGTAGTAGCTATAGCTATTACGTAGGGCT<u>AAA</u>AATAGCATCATATCTAG

We can also specify that we want to find 'A' a minimum of 3 times and a maximum of 5 with "A{3,5}", then we would match:

#### ATAGCATCATAATGTAGCATTTACGTAGTAGCTATAGCTATTACGTAGGGGCT<u>AAAAA</u>TAGCATCATATCTAG

(We could get the same result in this specific sequence, by using "A{3,}")



The caret (^) is used to match the beginning of the line.

So that if we want to match sequences that start with "ATG", we can use **^ATG**.

For example, in the following sequences:

**ATG**ATAGCTTAACATTTACGTAGTAGCTATAGCTATT

GTCATGAGCTATTAGCATCACATCTAGCACGTTCATG

**ATG**CTATGAAGTCTACTTTATAGCATCAAATCTAGTA

The regular expression ^ATG matches ATG only in the first and third lines because they begin with "ATG"



The dollar sign (\$) is used to match the end of a line or string.

If we want to match sequences that end with "TAA", we can use "TAA\$".

For example, in the following sequences:

TATAGCTAAAGTCTACTTTATAATCAATGATAGCT<u>TAA</u> ATGAGCTATTAGCATCACATCTAGCAGTCATGAGCTAT GTAGCATTTACGTAGTAGCTATAGCTATGCTAAGAAGT

The regular expression TAA\$ matches TAA only in the first line because it ends with "TAA"



Square brackets ([]) are used to define a set of characters to match.

For example, [ACGT] matches any single character that is A, C, G, or T.

If we want to find sequences where "ATG" is followed by a C or G, we can use ATG[CG].

In the sequence:

ATAGCATCAAATGCTAACATTTACGTAGTAGCTATAGCTATTACGTATGGCTACTTTATAGCATCAAATCT

The regular expression ATG[CG] matches both ATGC and ATGG.



The pipe (|) means "or" and allows you to specify alternative patterns.

For instance, ATG | TAA matches either "ATG" or "TAA"

In the sequences:

ATGACGACGTAGCGCAACAGCTCAACCTCAGGCTACTTTATAGCATCAAATCTAGCATT<u>TAA</u>ATAG TCTAGC<u>ATG</u>ACGACGTAGCGCAACAGCTCAACCTCAATAGCTATTACGTAGTGCAATGTACTATTA ACCTCAGGCTACTTTATATAGCTATTACGTAGAGCATCAAATCTAGCATT<u>TAA</u>ATAGCCCGTATCC

The regular expression ATG|TAA matches ATG or TAA when found.



Parentheses (()) are used for grouping and capturing.

If we want to capture sequences that follow the pattern "ATG" followed by any two characters and "TAA," we can use "(ATG..TAA)" to match and capture the sequence "ATGCTTAA":

ATGCTTAAATGCCCAGTAA

Captured groups can be referenced later in the same regular expression or used in programming languages. To find repeated sequences like "ATGTACTAA", you can use:

(ATGTACTAA).\*\1

It would match in the following sequence:

GTAA<u>ATGTACTAA</u>CAGTAACGTAGCG<u>ATGTACTAA</u>ACCTCAATAG



The backslash (\) is used as an escape character to treat special characters literally.

For example, if you want to match a literal dot, use "\." instead of "."

If we want to find "A.T" as it appears (with the dot), we can use "A\.T" <u>A.T</u>GACTTAAG.<u>A.T</u>

The regular expression "A\.T" matches A.T twice in the sequence.



The command grep will print the lines matching a given pattern.

grep PATTERN file
grep -e PATTERN file (Pattern uses regex)

Understanding grep with a simple fasta file: >contig1

AATCTAGCATTTACGTAGTAGCTAAAGCTAAACCTCAGGGGGCTACTTTAT

>contig2





Find a specific sequence within our sequences:

grep "AGGGG" file.fasta

-> will print only the first sequence:

AATCTAGCATTTACGTAGTAGCTAAAGCTAAACCTC<u>AGGGGG</u>CTACTTTAT



Find sequence headers only:

grep ">" file.fasta

-> will print all fasta headers:

>contig1

>contig2



Count number of sequences:

```
grep ">" file.fasta | wc -l
```

-> will count how many lines contain ">", which will match with the number of sequences: 2

we can also use the flag -c in grep to do the same:

```
grep -c ">" file.fasta
```



Print the DNA sequences with no headers:

grep -v ">" file.fasta

-> will print all lines that do not contain ">":

AATCTAGCATTTACGTAGTAGCTAAAGCTAAACCTCAGGGGGCTACTTTAT





sed ("stream editor") is a tool that can parse a file line by line, and transform text, using a compact programming language that can fit in one line. Sed is a powerful tool with a big array of possible commands, but the most common one is the substitution, in which we find a pattern and substitute it for another string.

sed 's/patternA/patternB/' file.txt





Modify the fasta header to contain "sequence" instead of "contig":

```
sed 's/contig/sequence/' file.fasta
```

We will obtain the entire file with the replacement:

>sequence1 assembled 2025 a

AATCTAGCATTTACGTAGTAGCTAAAGCTAAACCTCAGGGGCTACTTTAT

>sequence2 assembled 2025 b



Substitute spaces for underscores, in order to avoid problems with other programs:

In this case we add the flag "g" at the end, to make sure it replaces each occurrence even if there is multiple within the same line:

```
sed 's/ /_/g' file.fasta
```

We will obtain the entire file with the replacements:

```
>sequence1_assembled_2025_a
```

```
AATCTAGCATTTACGTAGTAGCTAAAGCTAAACCTCAGGGGGCTACTTTAT
```

```
>sequence2_assembled_2025_b
```



Simplify a fasta header:

```
sed 's/ .*//' file.fasta
```

We use regular expressions to match the first space we find in a line, followed by any character (.), any number of times (\*)

We will obtain the entire file with the replacement:

>contig1

AATCTAGCATTTACGTAGTAGCTAAAGCTAAACCTCAGGGGGCTACTTTAT

>contig2



Simplify a fasta header:

```
sed 's/ .*//' file.fasta
```

We use regular expressions to match the first space we find in a line, followed by any character (.), any number of times (\*)

We will obtain the entire file with the replacement:

>contig1

AATCTAGCATTTACGTAGTAGCTAAAGCTAAACCTCAGGGGGCTACTTTAT

>contig2



AWK is a language designed for text processing, like sed and grep. AWK is a standard feature of most Unix-like operating systems. AWK reads one line at a time, searching for a specific pattern to execute the desired action. It requires a condition, and an action:

awk condition {action} file.txt

AWK is a language field aware (column aware):

\$0 refers to the whole line

\$1, \$2, \$3 ... refers to columns 1, 2, 3 ...




Print only the lines containing genes in contig1: awk '\$1="contig1" {print}' file.bed

We would get the following printed out:

contig1	20	1305	gene1	•	+
contig1	467	4 8563	gene4	•	-

Understand	ding av	wk with	a simple	BED	file:
contig1	20	1305	gene1	•	+
contig1	4674	8563	gene4	•	-
contig2	1239	5387	gene6	•	-
contig3	546	3524	gene9	•	+



 Understanding awk with a simple BED file:

 contig1
 20
 1305
 gene1
 +

 contig1
 4674 8563
 gene4

 contig2
 1239 5387
 gene6

 contig3
 546
 3524
 gene9
 +

Count how many genes we have in our file:

```
awk '$1="contig"' file.bed | wc -1
```

or we can create a counter after each condition is met, in this case finding the word contig in column 1. And we use the function END to mark that an extra action is done when all lines are finished being parsed:

```
awk '$1="contig" {count++} END {print count}' file.bed
```

both these commands will print: 4



Understanding awk with a simple BED file:

contig1	20	1305	gene1	•	+
contig1	4674	8563	gene4	•	-
contig2	1239	5387	gene6	•	-
contig3	546	3524	gene9	•	+

We can also use the function BEGIN to add an action before we start parsing the lines in our file:

awk 'BEGIN {print "We have these many genes:"} \$1="contig" {count++} END
{print count}' file.bed

This command will print:

We have these many genes:

4



 Understanding awk with a simple BED file:

 contig1
 20
 1305
 gene1
 +

 contig1
 4674 8563
 gene4

 contig2
 1239 5387
 gene6

 contig3
 546
 3524
 gene9
 +

Finally, we can combine information in multiple columns to create our conditions.

Print out the gene names of all genes that are larger than 2000 bp:

We need can use the information in column 2 and 3, which marks the start and end of each gene, and we will print the information in column 4 (gene name) if column 3 - column 2 is larger than 2000:

```
awk '($3 - $2 > 2000) {print $4}' file.bed
```

It will print:

gene4

gene6



## Time to put into practice everything you learned!

Inside the folder /home/genomics/workshop\_materials/unix\_tutorial, you will find a folder called data. It contains a dataset that includes:

- genome\_assembly.fasta
- sequencing\_reads.fastq
- variant\_analysis.vcf
- annotation.gff