Introduzione alla linea di commando e ai dati di sequenziamento

Fabrizio Mafessoni – Genomica Computazionale, Università di Trieste, Laurea Specialistica di Genomica Funzionale 2025/205

LINUX

Oggi

- Comandi di base della linea di commando
- Formato delle sequenze di DNA dal sequenziatore
- Comandi utili per manipolazione sequenze
- Scriveremo il primo programma complesso/utile

echo

stampa sullo schermo (standard output)

```
~$ echo Ciao Trieste
Ciao Trieste
~$ echo "Ciao Trieste"
Ciao Trieste
```

Definizione di variabili

possiamo assegnare uno specifico valore ad una variabile. Per "richiamare" il valore della variabile, usiamo $NOME_DELLA_VARIABILE o ${NOME_DELLA_VARIABILE}

```
~$ YOURNAME=Fabrizio
~$ echo YOURNAME
YOURNAME
~$ echo $YOURNAME
Fabrizio
~$ echo ${YOURNAME}
Fabrizio
```

Definizione di variabili

possiamo assegnare uno specifico valore ad una variabile. Per "richiamare" il valore della variabile, usiamo $NOME_DELLA_VARIABILE

```
~$ YOURNAME=Fabrizio
~$ echo YOURNAME
YOURNAME
~$ echo $YOURNAME
Fabrizio
~$ echo ${YOURNAME}
Fabrizio
```

Per evitare errori, spesso è utile usare le parentesi graffe ${NOME_DELLA_VARIABILE}

```
~$ echo ${YOURNAME}_${YOURNAME}
Fabrizio_Fabrizio
~$ echo $YOURNAME_$YOURNAME
Fabrizio
```

whoami

stampa lo usernameb

```
~$ whoami
fabrizio
```

whoami

stampa lo usernameb

```
~$ whoami
fabrizio
```

$(comando)

esegue il comando, cosi che possiamo assegnare il suo output ad una variabile

```
~$ YOURUSERNAME=$( whoami )
~$ echo $YOURUSERNAME
fabrizio
```

NB: ${VARIABILE} è diverso da $(COMANDO) !!

date

data di oggi

```
~$ date
Thu Sep 25 02:32:50 CEST 2025
```

date

data di oggi

```
~$ date
Thu Sep 25 02:32:50 CEST 2025
```

pwd

cartella in cui vi trovate

```
~$ pwd
/home/fabrizio
```

Possiamo come prima assegnarla ad una variabile

```
~$ CURRENTFOLDER=$( pwd )
~$ echo $CURRENTFOLDER
/home/fabrizio
```

>

redirige l'output verso un file

```
~$ echo $YOURNAME > testfile.txt
~$
```

ls

elenca file nella cartella

```
~$ ls -lh testfile.txt
-rw-r--r-- 1 fabrizio fabrizio 9 Sep 25 02:37 testfile.txt
```

cat

stampa e concatena contenuto di un file

```
~$ cat testfile.txt
Fabrizio
~$ cat testfile.txt testfile.txt
Fabrizio
Fabrizio
~$ echo LucioCorsi > testfile.txt
~$ cat testfile.txt
LucioCorsi
```

cat

stampa e concatena contenuto di un file

>>

redirige l'output verso un file appendendolo alla fine, e non sovrascrivendo come >

```
~$ cat testfile.txt
Fabrizio
~$ cat testfile.txt testfile.txt
Fabrizio
Fabrizio
~$ echo LucioCorsi > testfile.txt
~$ cat testfile.txt
LucioCorsi
~$ echo $YOURNAME > testfile.txt
~$ echo LucioCorsi >> testfile.txt
~$ cat testfile.txt
Fabrizio
LucioCorsi
```

rm

cancella un file

rm -r

cancella una cartella

```
~$ ls testfile.txt
testfile.txt
~$ rm testfile.txt
~$ ls testfile.txt
ls: cannot access 'testfile.txt': No such file or directory
```

```
~$ mkdir GenomicaComputazionale
~$ ls GenomicaComputazionale/
~$ rm -r GenomicaComputazionale
~$ ls GenomicaComputazionale/
ls: cannot access 'GenomicaComputazionale/': No such file or directory
```

cd

permette di cambiare cartella
. cartella attuale
.. Cartella madre/precedente
../.. Due cartelle precedenti
Notare la differenza tra path relativi e assoluti (preceduti da /).

```
~$ pwd
/home/fabrizio
~$ echo $HOME
/home/fabrizio
~$ mkdir GenomicaComputazionale
~$ cd GenomicaComputazionale
~/GenomicaComputazionale$ pwd
/home/fabrizio/GenomicaComputazionale
~/GenomicaComputazionale$ cd .
~/GenomicaComputazionale$ pwd
/home/fabrizio/GenomicaComputazionale
~/GenomicaComputazionale$ cd ..
~$ pwd
/home/fabrizio
~$ cd ../..
/$ pwd
/
/$ ls
bin                boot  etc   init  lib.usr-is-merged  lost+found  mnt  proc  run   sbin.usr-is-merged  srv  tmp  var
bin.usr-is-merged  dev   home  lib   lib64              media       opt  root  sbin  snap                sys  usr
```

cd

permette di cambiare cartella

. cartella attuale

.. Cartella madre/precedente

../.. Due cartelle precedenti

Notare la differenza tra path relativi e assoluti (preceduti da /).

```
~$ pwd
/home/fabrizio
~$ echo $HOME
/home/fabrizio
~$ mkdir GenomicaComputazionale
~$ cd GenomicaComputazionale
~/GenomicaComputazionale$ pwd
/home/fabrizio/GenomicaComputazionale
~/GenomicaComputazionale$ cd .
~/GenomicaComputazionale$ pwd
/home/fabrizio/GenomicaComputazionale
~/GenomicaComputazionale$ cd ..
~$ pwd
/home/fabrizio
~$ cd ../..
/$ pwd
/
/$ ls
bin                boot  etc   init  lib.usr-is-merged  lost+found  mnt  proc  run   sbin.usr-is-merged  srv  tmp  var
bin.usr-is-merged  dev   home  lib   lib64              media       opt  root  sbin  snap                sys  usr
/$ cd /home/fabrizio/GenomicaComputazionale/
~/GenomicaComputazionale$ pwd
/home/fabrizio/GenomicaComputazionale
~/GenomicaComputazionale$
```

`cd ~`

è come

`cd $HOME`

o semplicemente

`cd`

```
~/GenomicaComputazionale$ cd ~
~$ cd GenomicaComputazionale
mkdir Esercitazione1
~/GenomicaComputazionale$ cd Esercitazione1
~/GenomicaComputazionale/Esercitazione1$ echo "Questa cartella contiene l'esercitazione sulla basi di linux fatta il $( date )" > README.txt
```

`history`

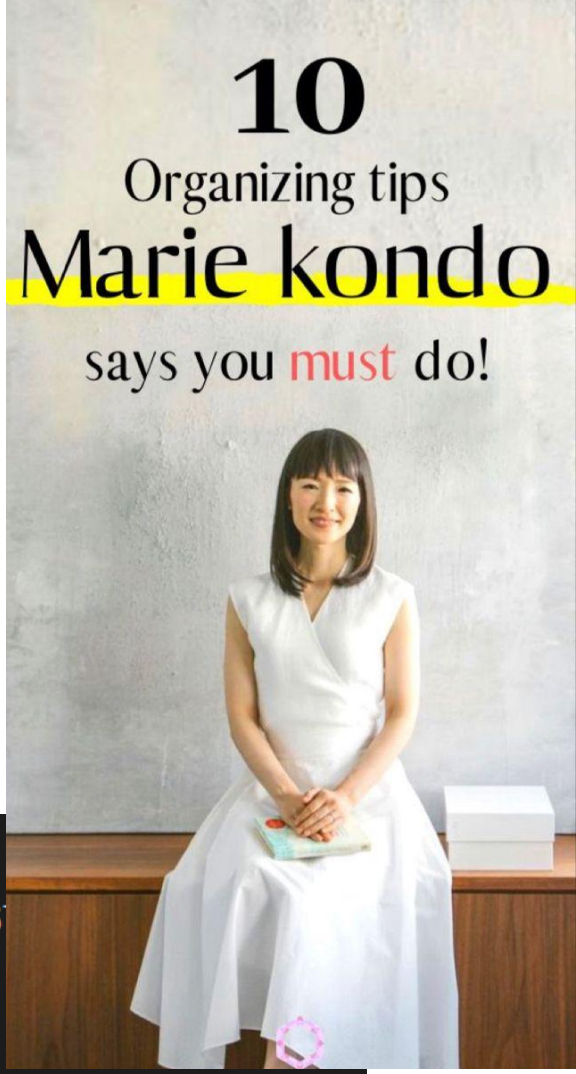
elenca la storia dei comandi.

Utilissimo per creare dei log o semplicemente ricordarci cosa abbiamo fatto.

```
~/GenomicaComputazionale/Esercitazione1$ history >> README.txt
~/GenomicaComputazionale/Esercitazione1$ head README.txt
Questa cartella contiene l'esercitazione sulla basi di linux fatta il Thu Sep 25 02:55:21 CEST 2025
 1321  ls -lht .. / | head
 1322  vim README_Barack
 1323  ls pca_results.eigenvec
 1324  less pca_results.eigenvec
 1325  vim README_Barack
 1326  mkdir results_barack_20250618
 1327  vim README_Barack
 1328  for phenotype in "gws" "leafarea" "quantomyield" "chlorophyllcontent" "stemheight" "dryweight" "grade"
; plink --bfile GWAS/chrall --pheno ../phenotype_${phenotype}_20250618.txt --pheno-name $phenotype --linear --
genvec --covar-number 1-5 --out results_barack_20250618/resPCA_$phenotype; done
 1329  vim README_Barack
```

`history`

elenca la storia dei comandi.
Utilissimo per creare dei log o semplicemente ricordarci cosa abbiamo fatto.
NB: creare dei README è bellissimo




```
~/GenomicaComputazionale/Esercitazione1$ history >> README.txt
~/GenomicaComputazionale/Esercitazione1$ head README.txt
Questa cartella contiene l'esercitazione sulla basi di linux fatta il Thu Sep 25 02:55:21 CES
 1321  ls -lht .. / | head
 1322  vim README_Barack
 1323  ls pca_results.eigenvec
 1324  less pca_results.eigenvec
 1325  vim README_Barack
 1326  mkdir results_barack_20250618
 1327  vim README_Barack
 1328  for phenotype in "gws" "leafarea" "quantomyield" "chlorophyllcontent" "stemheight" "dryweight" "grade"
; plink --bfile GWAS/chrall --pheno ../phenotype_${phenotype}_20250618.txt --pheno-name $phenotype --linear --
genvec --covar-number 1-5 --out results_barack_20250618/resPCA_$phenotype; done
 1329  vim README_Barack
```

`head` e `tail`
permettono di esplorare l'inizio e la fine dei files

```
~/GenomicaComputazionale/Esercitazione1$ head -3 README.txt
Questa cartella contiene l'esercitazione sulla basi di linux fatta il Thu Sep 25 02:55:21 CEST 2025
 1321  ls -lht .. / | head
 1322  vim README_Barack
~/GenomicaComputazionale/Esercitazione1$ tail -3 README.txt
 2318  echo "Questa cartella contiene l'esercitazione sulla basi di linux fatta il $( date )" > README.txt
 2319* history
 2320  history >> README.txt
```

wc

conto di righe, parole e caratteri

```
~/GenomicaComputazionale/Esercitazione1$ wc -l README.txt
1001 README.txt
~/GenomicaComputazionale/Esercitazione1$ wc -w README.txt
5118 README.txt
~/GenomicaComputazionale/Esercitazione1$ wc -c README.txt
49987 README.txt
```

Il simbolo * indica «qualsiasi carattere» ed è utilissimo per guardare a piu' file insieme

```
~/GenomicaComputazionale/Esercitazione1$ echo $YOURNAME > testfile.txt
~/GenomicaComputazionale/Esercitazione1$ ls -lh testfile.txt
-rw-r--r-- 1 fabrizio fabrizio 9 Sep 25 03:04 testfile.txt
~/GenomicaComputazionale/Esercitazione1$ ls -lh
total 56K
-rw-r--r-- 1 fabrizio fabrizio 49K Sep 25 02:57 README.txt
-rw-r--r-- 1 fabrizio fabrizio   9 Sep 25 03:04 testfile.txt
~/GenomicaComputazionale/Esercitazione1$ ls -lh *
-rw-r--r-- 1 fabrizio fabrizio 49K Sep 25 02:57 README.txt
-rw-r--r-- 1 fabrizio fabrizio   9 Sep 25 03:04 testfile.txt
~/GenomicaComputazionale/Esercitazione1$ ls -lh *.sh
ls: cannot access '*.sh': No such file or directory
~/GenomicaComputazionale/Esercitazione1$ ls -lh *.txt
-rw-r--r-- 1 fabrizio fabrizio 49K Sep 25 02:57 README.txt
-rw-r--r-- 1 fabrizio fabrizio   9 Sep 25 03:04 testfile.txt
~/GenomicaComputazionale/Esercitazione1$ ls -lh te*txt
-rw-r--r-- 1 fabrizio fabrizio 9 Sep 25 03:04 testfile.txt
~/GenomicaComputazionale/Esercitazione1$ wc -l *
 1001 README.txt
    1 testfile.txt
 1002 total
~/GenomicaComputazionale/Esercitazione1$ ls -lht *
-rw-r--r-- 1 fabrizio fabrizio   9 Sep 25 03:04 testfile.txt
-rw-r--r-- 1 fabrizio fabrizio 49K Sep 25 02:57 README.txt
~/GenomicaComputazionale/Esercitazione1$ cat *
```

Comandi utili per la bioinformatica

Comandi utili per la bioinformatica

grep

pipe |

just a fancy **grep**

grep PAROLA

trattiene tutte le righe con PAROLA

(o le esclude se con –v)

```
~$ grep date README.txt
 1966  sudo apt update && sudo apt -y upgrade
 1983  date
 2271  date
 2318  echo "Questa cartella contiene l'esercitazione sulla basi di linux fatta il $( date )" > README.txt
```

provate anche:

```
grep -v cd README.txt
grep . README.txt
grep "\." README.txt
```

Notate come per i caratteri speciali come il punto “.” sia necessario usare la backslash \ per evadere la loro funzione speciale e vederli come un semplice carattere (qui punto).

|

il simbolo «stanghetta» o «pipe»

`comando1 | comando2`

redirige l'output di `comando1` come input per `comando2`

|

il simbolo «stanghetta» o «pipe»

`comando1 | comando2`

redirige l'output di `comando1` come input per `comando2`

```
~$ echo "Olly" > sanremo.txt
echo "Lucio Corsi" >> sanremo.txt
echo "Fedez" >> sanremo.txt
echo "Brunori Sas" >> sanremo.txt
~$ cat sanremo.txt
Olly
Lucio Corsi
Fedez
Brunori Sas
~$ cat sanremo.txt | grep u
Lucio Corsi
Brunori Sas
~$ cat sanremo.txt | grep u | grep L
Lucio Corsi
```

|

il simbolo «stanghetta» o «pipe»

`comando1 | comando2`

redirige l'output di `comando1` come input per `comando2`

```
~$ echo "Olly" > sanremo.txt
echo "Lucio Corsi" >> sanremo.txt
echo "Fedez" >> sanremo.txt
echo "Brunori Sas" >> sanremo.txt
~$ cat sanremo.txt
Olly
Lucio Corsi
Fedez
Brunori Sas
~$ cat sanremo.txt | grep u
Lucio Corsi
Brunori Sas
~$ cat sanremo.txt | grep u | grep L
Lucio Corsi
```

```
~$ cat sanremo.txt | grep -v Fedez
Olly
Lucio Corsi
Brunori Sas
```

Se non siete fan di Sanremo (o se volete vedere utilizzi piu’ utilli)

```
~$ history | grep install
 1969  sudo apt -y install build-essential
 1970  sudo apt -y install git curl wget ca-certificates nano
 1971  sudo apt -y install vim less unzip tar
 1972  sudo apt -y install grep gawk sed coreutils diffutils
 1973  sudo apt -y install samtools bcftools bedtools tabix
 1974  sudo apt -y install fastqc bwa
 2350  history | grep install
```

La vostra prima «pipeline»

pipeline: serie di comandi eseguiti uno dopo l'altro in cui l'output del primo diventa l'input del secondo «senza file temporanei»

```
~$ wc -l README.txt
1001 README.txt
~$ cat README.txt | wc -l
1001
~$ cat README.txt | grep cd | wc -l
98
~$ cat README.txt | grep -v cd | grep = | wc -l
84
```

Scopo ultimo del corso: saper partire da sequenze di DNA fino ad assemblare interi genomi e caratterizzarli funzionalmente

Dal sequenziatore..

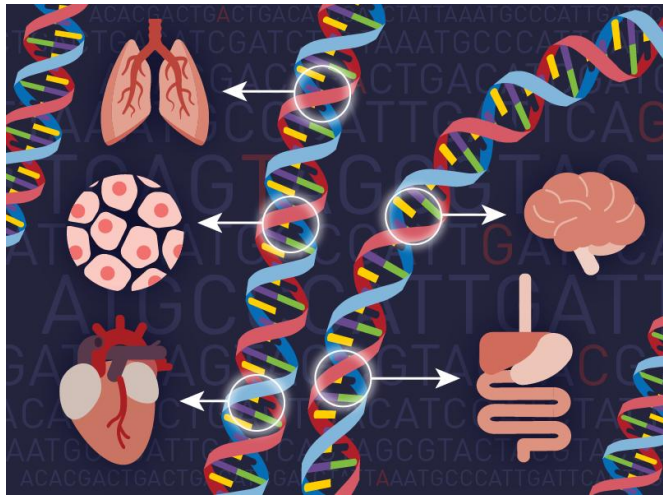
..al genoma..

..alla funzione

sequenze

assemblaggio

mappatura
e
genotipizzazione



Formati di file bioinformatici

.fastq

.fasta, .bam, .vcf

.bed, .gff

wget
scarica files dal web

```
wget https://raw.githubusercontent.com/fabrimafe/GenomicaComputazionale2025_2026/refs/heads/main/sequences1.fastq
wget https://raw.githubusercontent.com/fabrimafe/GenomicaComputazionale2025_2026/refs/heads/main/sequences2.fastq
wget https://raw.githubusercontent.com/fabrimafe/GenomicaComputazionale2025_2026/refs/heads/main/sequences3.fastq
wget https://raw.githubusercontent.com/fabrimafe/GenomicaComputazionale2025_2026/refs/heads/main/sequences4.fastq
wget https://raw.githubusercontent.com/fabrimafe/GenomicaComputazionale2025_2026/refs/heads/main/sequences5.fastq
```

Uso dell'* per nomi di files.

`ls -lt` ordina i file dal piu recente al piu vecchio

```
~$ ls *fastq
sequences1.fastq  sequences2.fastq  sequences3.fastq  sequences4.fastq  sequences5.fastq
~$ ls -lth | head
total 2.1G
-rw-r--r--  1 fabrizio fabrizio   35 Sep 25 07:56 sanremo.txt
-rw-r--r--  1 fabrizio fabrizio  47K Sep 25 07:43 README.txt
drwxr-xr-x  3 fabrizio fabrizio 4.0K Sep 25 02:54 GenomicaComputazionale
drwxr-xr-x  2 fabrizio fabrizio 4.0K Sep 25 02:36 empty
-rwxr-xr-x  1 fabrizio fabrizio  613 Sep 25 01:46 create_report_sequences.sh
-rw-r--r--  1 fabrizio fabrizio  575 Sep 25 01:45 report.txt
-rw-r--r--  1 fabrizio fabrizio  162 Sep 25 00:56 adaptors.txt
-rw-r--r--  1 fabrizio fabrizio  25K Sep 25 00:56 sequences.txt
-rw-r--r--  1 fabrizio fabrizio  10K Sep 25 00:56 readnames.txt
```

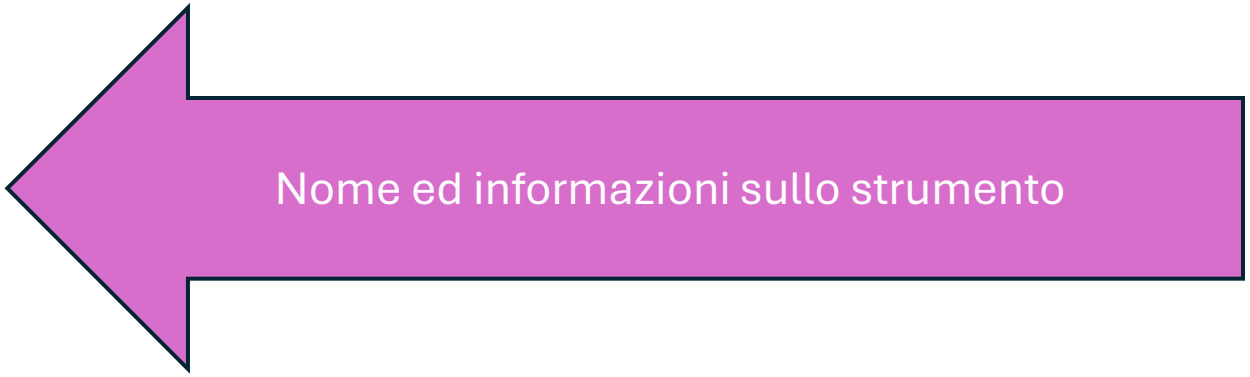
Le sequenze di DNA: il formato fastq

```
~$ head sequences1.fastq
@A00619:220:HYK7GDRX3:1:2101:13440:1000 1:N:0:ACTATAGA+TGCATACC
NCCCGTGAGATGATGCGCGTCATCCTCAACAAGGCCAAATCTGACCCCAAGCGACTCGTTCTCGCTGAGATCGGAAGAGCACACGTCTGAACTCCAGTCAC
+
#FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF:FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
@A00619:220:HYK7GDRX3:1:2101:18629:1000 1:N:0:ACTATAGA+TGCATACC
NGCTGTGGACTCCACCCGAGTTCCGGATCGAAACTGGTGAACTTATCCAGCAGCTCTCGGTCGATCCCCGGGAACTCCTCCAACACCACTGTTGGAAGACT
+
#FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
@A00619:220:HYK7GDRX3:1:2101:20745:1000 1:N:0:ACTATAGA+TGCATACC
NGGATATCAATCAACTCCACGCAACTCCAACTTCCCCTCAGATCGGAAGAGCACACGTCTGAACTCCAGTCACACTATAGAATCTCGGTTTGCGTCTTATG
```

- Riga1: @ riga con il nome della sequenza. Spesso (non obbligatoriamente) informazioni sul sequenziamento
- Riga2: Sequenza ACGT
- Riga3: +, un semplice separatore
- Riga4: Qualità delle sequenze

Riga 1: nome ed esempio di informazioni sulla sequenza

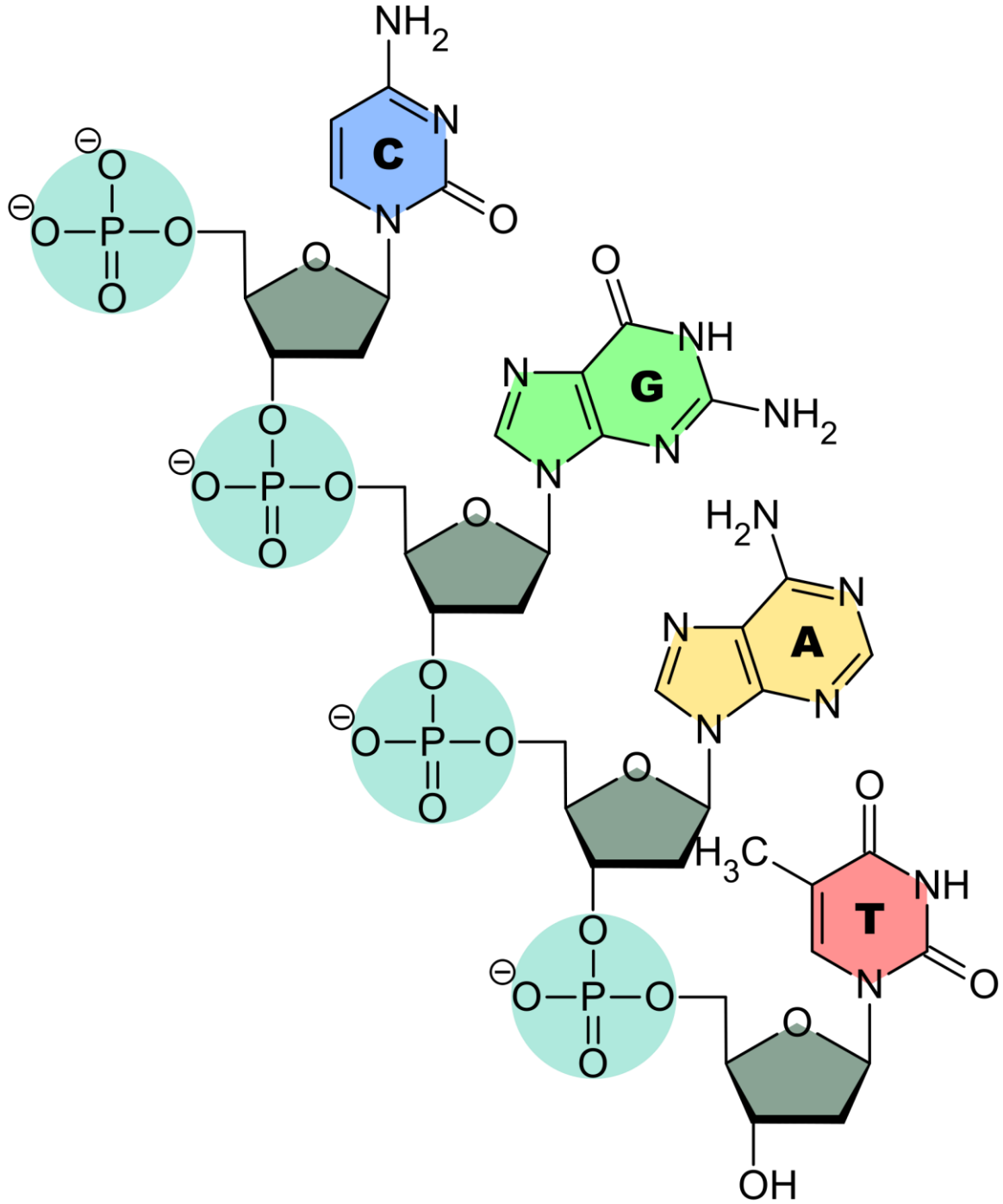
- @A00619:220:HYK7GDRX3:1:2101:13440:1000 1:N:0:ACTATAGA+TGCATACC
- A00619 → instrument ID
- 220 → run number
- HYK7GDRX3 → flowcell ID
- 1 → lane
- 2101 → tile
- 13440:1000 → x:y coordinates of the cluster -> **lo rende unico!!**



- Dopo lo spazio → **informazioni sulla read** (qui *Illumina CASAVA format*):
- 1 → tipo di read (1 = forward read; 2 = reverse read in paired-end sequencing)
- N → filtro (Y = filtro fallito, N = filtro passato)
- 0 → reads di controllo (0 = non una read di controllo)
- ACTATAGA+TGCATACC → sequenze di index/barcode (i7 + i5 adapters) usate per il demultiplexing (quando piu' campioni in uno stesso sequenziamento; ogni campione ha dei barcode diversi).

Riga2

- ACGGGAGGT……



Riga 4

NCCCGTGAGATGATGCGCGTCATCCTCAACAAGGCCAAATCTGACCCCAAGCGACTCGTTCTCGCTGAGATCGGAAGAGCACACGTCTGAACTCCAGTCAC

+

#FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF:FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

Qualità codificata come ASCII **Phred+33** → character → Q score → probabilità di erore.

N all'inizio → base **not confidently called** (incerta, "può essere qualsiasi cosa").

Esempi:

→ ASCII 35 → Q=2 → ~63% accuratezza (bassa, in pratica inutile).

F → ASCII 70 → Q=37 → error probability ≈ 0.0002 (~1 errore in 5000 basi, molto alta).

: → ASCII 58 → Q=25 → errore ≈ 0.003 (~1 in 316 basi).

sed

`sed "s/OLD/NEW/"` sostituisce il pattern OLD con NEW

sed

`sed "s/OLD/NEW/"` sostituisce il pattern OLD con NEW

- ESERCIZIO: Creiamo una lista dei nomi delle sequenze

sed

`sed "s/OLD/NEW/"` sostituisce il pattern OLD con NEW

- ESERCIZIO: Creiamo una lista dei nomi delle sequenze

```
cat sequences1.fastq | wc -l
cat sequences1.fastq | grep @ > readnames.txt
```

Ma cosi includiamo anche adattatori e altre informazioni, che vogliamo rimuovere. Quindi proviamo con:

```
cat sequences1.fastq | grep @ | sed 's/1:N:0:ACTATAGA+TGCATACC//'
```

Esistono piu' adattatori (verificate con cat)! Rimuoviamo allora tutto ciò che viene dopo "1:N:0:" usando .*:

```
cat sequences1.fastq | grep @ | sed "s/1:N:0:.*//"
cat sequences1.fastq | grep @ | sed "s/1:N:0:.*//" > readnames.txt
```

`[^X]` nega un carattere X dentro `sed`

- Esercizio: contiamo quante basi hanno qualita' F (alta qualita')

Rimuoviamo prima tutti i caratteri che non sono F:

```
cat sequences1.fastq | sed 's/[^F]//g'
```

```
cat sequences1.fastq | sed 's/[^F]//g' | wc –c
```

Problema: le nuove linee (\n) sono contate come un carattere. Quindi wc –c conta correttamente tutte le F, ma anche il numero di linee. Rimuoviamo le nuove linee con tr –d (-d per delete)

```
cat sequences1.fastq | sed 's/[^F]//g' | tr -d '\n' | wc -c
```

sort

ordina

uniq

estrae le righe uniche da un file sorted

sort

ordina

uniq

estrae le righe uniche da un file sorted

- ESERCIZIO: Creiamo una lista degli adattatori

```
cat sequences1.fastq | grep @ | sed "s/.*1:N:0://" | uniq
```

```
cat sequences1.fastq | grep @ | sed "s/.*1:N:0://" | sort | uniq > adaptors.txt
```

sort

ordina

uniq

estrae le righe uniche da un file sorted

- ESERCIZIO: Creiamo una lista degli adattatori

```
cat sequences1.fastq | grep @ | sed "s/.*1:N:0://" | uniq
```

```
cat sequences1.fastq | grep @ | sed "s/.*1:N:0://" | sort | uniq > adaptors.txt
```

Notate inoltre che naturalmente sed s/ rimuove solamente la prima istanza del carattere (per linea). Quindi se volessimo ad esempio sostituire tutte le A con delle N, dovremmo aggiungere /g (per sostituzione globale)

```
cat adaptors.txt | sed "s/A/N/"
```

```
cat adaptors.txt | sed "s/A/N/g"
```

Altre espressioni regolari complesse

Esercizio: estraiamo solo le sequenze di DNA

```
cat sequences1.fastq | grep -v @ | grep -v + | less -S
```

Possiamo cominciare con l'estrarre tutte le sequenze che iniziano (^carattere), finiscono con "carattere" (carattere$)

```
cat sequences1.fastq | grep -v @ | grep -v + | grep '^[ACGTN]'
```

```
cat sequences1.fastq | grep -v @ | grep -v + | grep '^[ACGTN][ACGTN]'
```

```
cat sequences1.fastq | grep -v @ | grep -v + | grep '[ACGTN][ACGTN]$'
```

o entrambi, senza interruzioni

```
cat sequences1.fastq | grep -E '^[ACGTN]+$' > sequences.txt
```

Oppure, possiamo ritenere tutte le sequenze che contengono o A,C,G,T o N ripetute esattamente 101 volte (come possiamo contare con wc –c)

```
cat sequences1.fastq | grep -v @ | grep -v + | grep '[ACGTN]\{101\}'
```

```
cat sequences1.fastq | tail -1 | wc -c
```

```
cat sequences1.fastq | grep '[ACGTN]\{101\}'
```

Altre espressioni regolari complesse

Esercizio: estraiamo solo le sequenze di DNA

```
cat sequences1.fastq | grep -v @ | grep -v + | less -S
```

Possiamo cominciare con l'estrarre tutte le sequenze che iniziano (^carattere), finiscono con "carattere" (carattere$)

```
cat sequences1.fastq | grep -v @ | grep -v + | grep '^[ACGTN]'
```

```
cat sequences1.fastq | grep -v @ | grep -v + | grep '^[ACGTN][ACGTN]'
```

```
cat sequences1.fastq | grep -v @ | grep -v + | grep '[ACGTN][ACGTN]$'
```

o entrambi, senza interruzioni

```
cat sequences1.fastq | grep -E '^[ACGTN]+$' > sequences.txt
```

Oppure, possiamo ritenere tutte le sequenze che contengono o A,C,G,T o N ripetute esattamente 101 volte (come possiamo contare con wc –c)

```
cat sequences1.fastq | grep -v @ | grep -v + | grep '[ACGTN]\{101\}'
```

```
cat sequences1.fastq | tail -1 | wc -c
```

```
cat sequences1.fastq | grep '[ACGTN]\{101\}'
```

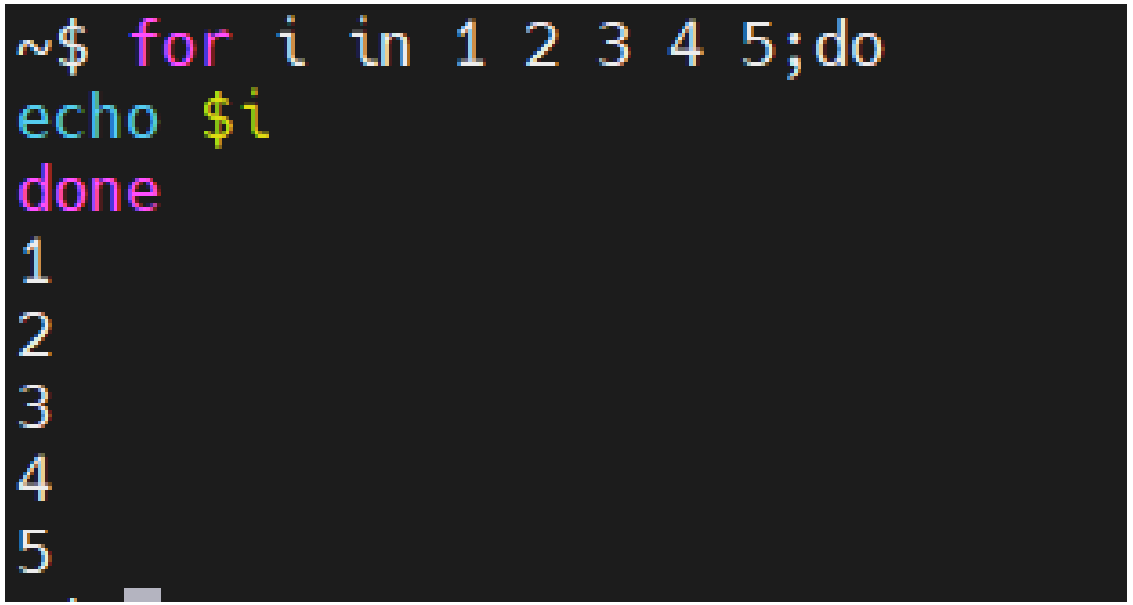
```
cat sequences1.fastq | grep -E '[ACGTN]{101}'
```

```
..o tutte le sequenze che iniziano e fino alla fine hanno solo
```

Loops e condizionali: la vera programmazione

`for VARIABILE in ….;do comando; done`

esegue un comando per ogni valore in …



`for VARIABILE in ….;do comando; done`

esegue un comando per ogni valore in …

Il **comando** puó essere uno o piu comandi, anche complessi

```
~$ for i in 1 2 3 4 5;do
echo sequences${i}.fastq
ls -lh sequences${i}.fastq
done
sequences1.fastq
-rw-r--r-- 1 fabrizio fabrizio 66K Sep 25 00:44 sequences1.fastq
sequences2.fastq
-rw-r--r-- 1 fabrizio fabrizio 66K Sep 25 00:44 sequences2.fastq
sequences3.fastq
-rw-r--r-- 1 fabrizio fabrizio 66K Sep 25 00:44 sequences3.fastq
sequences4.fastq
-rw-r--r-- 1 fabrizio fabrizio 66K Sep 25 00:44 sequences4.fastq
sequences5.fastq
-rw-r--r-- 1 fabrizio fabrizio 67K Sep 25 00:44 sequences5.fastq
```

`for VARIABILE in ….;do comando; done`

esegue un comando per ogni valore in …

```
~$ for i in $( seq 1 5);do
```

Il **comando** puó essere uno o piu comandi, anche complessi

```
~$ for i in 1 2 3 4 5;do
echo sequences${i}.fastq
ls -lh sequences${i}.fastq
done
sequences1.fastq
-rw-r--r-- 1 fabrizio fabrizio 66K Sep 25 00:44 sequences1.fastq
sequences2.fastq
-rw-r--r-- 1 fabrizio fabrizio 66K Sep 25 00:44 sequences2.fastq
sequences3.fastq
-rw-r--r-- 1 fabrizio fabrizio 66K Sep 25 00:44 sequences3.fastq
sequences4.fastq
-rw-r--r-- 1 fabrizio fabrizio 66K Sep 25 00:44 sequences4.fastq
sequences5.fastq
-rw-r--r-- 1 fabrizio fabrizio 67K Sep 25 00:44 sequences5.fastq
```

`for VARIABILE in ….;do comando; done`

esegue un comando per ogni valore in …

I valori della variabile possono anche essere specificati come liste di files, o liste di nomi

```
~$ ls sequences*.fastq
sequences1.fastq  sequences2.fastq  sequences3.fastq  sequences4.fastq  sequences5.fastq
~$ for i in sequences*.fastq;do echo $i; done
sequences1.fastq
sequences2.fastq
sequences3.fastq
sequences4.fastq
sequences5.fastq
```

`for VARIABILE in ….;do comando; done`

esegue un comando per ogni valore in …

I valori della variabile possono anche essere specificati come liste di files, o liste di nomi

```
~$ ls sequences*.fastq
sequences1.fastq  sequences2.fastq  sequences3.fastq  sequences4.fastq  sequences5.fastq
~$ for i in sequences*.fastq;do echo $i; done
sequences1.fastq
sequences2.fastq
sequences3.fastq
sequences4.fastq
sequences5.fastq
```

```
~$ for i in topolino paperino;do
echo $i | sed "s/o$/a/"
done
topolina
paperina
```

```
if [condizione];then
comando1
else
comando2
fi
```

> «**g**reater **t**han»

*Esegue **comando1** se condizione è vera, **comando2** se falsa*

```
~$ for i in 1 2 3 4 5;do
echo $i
done
1
2
3
4
5
```

```
for i in 1 2 3 4 5;do
    if [ $i -gt 3 ]; then
    echo $i
    fi
done
4
5
```

```
if [condizione];then
comando1
else
comando2
fi
```

*Esegue **comando1** se condizione è vera, **comando2** se falsa*

```
~$ for i in 1 2 3 4 5;do
echo $i
done
1
2
3
4
5
```

< «less than»

```
~$ for i in 1 2 3 4 5;do
    if [ $i -lt 3 ]; then
    echo $i
    fi
done
1
2
```

```
if [condizione];then
comando1
else
comando2
fi
```

```
~$ VINCITORE="LucioCorsi"
if [ $VINCITORE == "Fedez" ];then
echo "Noooo!"
else
echo "Yay!"
fi
Yay!
```

```
$(( OPERAZIONE MATEMATICA ))

totale=$(( 2 + 2))
echo $totale
```

```
$(( OPERAZIONE MATEMATICA ))

totale=$(( 2 + 2))
echo $totale
```

$((OPERAZIONE)) diverso da ${VARIABILE} diverso da $(comando)

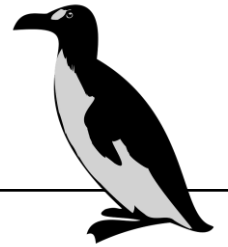
while loop

esegue finche' una condizione è verificata

```
~$ i=1
while [ $i -lt 3 ];do
echo $i
i=$(( i + 1))
done
1
2
```

Loops e condizionali: la vera programmazione

Stesso concetto in tutti i linguaggi



Concetto	**Bash**	**Python**	**R**	**AWK**

FOR	`for i in 1 2 3; do`
	`echo $i`
	`done`
		`for i in [1,2,3]:`
		`print(i)`
			`for(i in 1:3) {`
			`print(i)`
			`}`
				`for(i=1;i<=3;i++)`
				`{`
				`print i`
				`}`

IF	`if [ $x -eq 5 ];`
	`then`
	`echo "Yes"`
	`fi`
		`if x == 5:`
		`print("Yes")`
			`if(x == 5) {`
			`print("Yes")`
			`}`
				`if(x == 5) {`
				`print "Yes"`
				`}`

WHILE	`while [ $x -lt 3 ];`
	`do`
	`echo $x`
	`x=x+1`
	`done`
		`while x < 3:`
		`print(x)`
		`x += 1`
			`while(x < 3) {`
			`print(x)`
			`x <- x + 1`
			`}`
				`while(x < 3) {`
				`print x`
				`x++`
				`}`

gestire files e programmi

programmazione interpretata

statistica e analisi dati

uso rapido di stringhe e tabelle

Stesso concetto in tutti i linguaggi

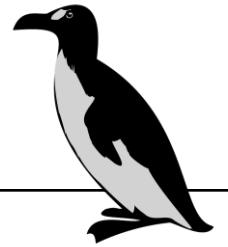
Concetto	Bash	Python	Julia	R	AWK	C	C++	Java	Rust

FOR	for i in 1 2 3; do
	echo $i
	done
		for i in [1,2,3]:
		print(i)
			for i in [1,2,3]
			println(i)
			end
				for(i in 1:3) {
				print(i)
				}
					for(i=1;i<=3;i++) {
					print i
					}
						for(int i=1;i<=3;i++) {
						printf("%d",i);
						}
							for(int i=1;i<=3;i++) {
							cout << i;
							}
								for(int i=1;i<=3;i++) {
								System.out.println(i);
								}
									for i in 1..=3 {
									println!("{}", i);
									}

IF	if [$x -eq 5];
	then
	echo "Yes"
	fi
		if x == 5:
		print("Yes")
			if x == 5
			println("Yes")
			end
				if(x == 5) {
				print("Yes")
				}
					if(x == 5) {
					print "Yes"
					}
						if(x == 5) {
						printf("Yes");
						}
							if(x == 5) {
							cout << "Yes";
							}
								if(x == 5) {
								System.out.println("Yes");
								}
									if x == 5 {
									println!("Yes");
									}

WHILE	while [$x -lt 3]; do
	echo $x
	((x++))
	done
		while x < 3:
		print(x)
		x += 1
			while x < 3
			println(x)
			x += 1
			end
				while(x < 3) {
				print(x)
				x <- x + 1
				}
					while(x < 3) {
					print x
					x++
					}
						while(x < 3) {
						printf("%d",x);
						x++;
						}
							while(x < 3) {
							cout << x;
							x++;
							}
								while(x < 3) {
								System.out.println(x);
								x++;
								}
									while x < 3 {
									println!("{}", x);
									x += 1;
									}

Stesso concetto in tutti i linguaggi



Concetto	**Bash**	**Python**	**R**	**AWK**

FOR	`for i in 1 2 3; do`
	`echo $i`
	`done`
		`for i in [1,2,3]:`
		`print(i)`
			`for(i in 1:3) {`
			`print(i)`
			`}`
				`for(i=1;i<=3;i++)`
				`{`
				`print i`
				`}`

IF	`if [ $x -eq 5 ];`
	`then`
	`echo "Yes"`
	`fi`
		`if x == 5:`
		`print("Yes")`
			`if(x == 5) {`
			`print("Yes")`
			`}`
				`if(x == 5) {`
				`print "Yes"`
				`}`

WHILE	`while [ $x -lt 3 ];`
	`do`
	`echo $x`
	`x=x+1`
	`done`
		`while x < 3:`
		`print(x)`
		`x += 1`
			`while(x < 3) {`
			`print(x)`
			`x <- x + 1`
			`}`
				`while(x < 3) {`
				`print x`
				`x++`
				`}`

gestire files e programmi

programmazione interpretata

statistica e analisi dati

uso rapido di stringhe e tabelle

Dollari e parentesi

ESEMPIO

- ${VARIABILE} — richiama il valore di una variabile

```
$ X=Trieste; echo $X
Trieste
$ echo ${X}
Trieste
```

- $(COMANDO) — restituisce l'output di un comando

```
$ date
Wed Oct  1 14:03:10 CEST 2025
$ echo date
date
$ echo $( date )
Wed Oct 1 14:03:20 CEST 2025
```

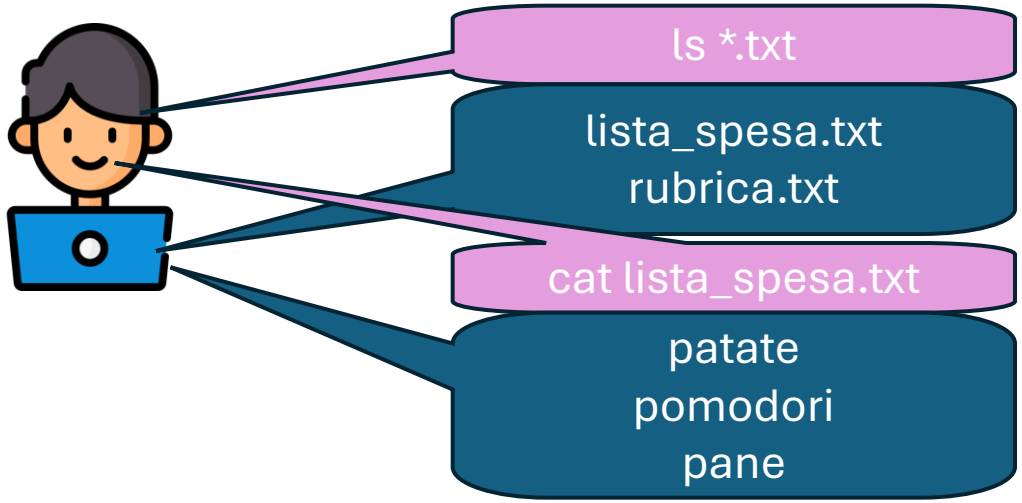
- $((OPERAZIONE)) — esegue un'operazione matematica

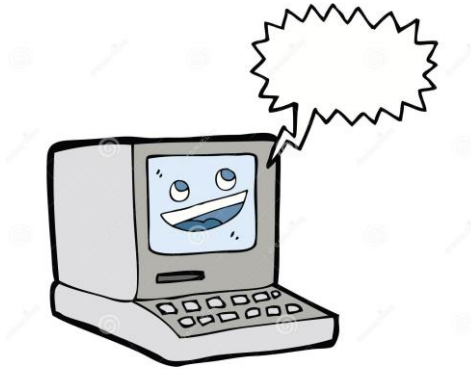
```
$ echo $(( 2 + 2 ))
4
```

- [CONDIZIONE] — valuta se CONDIZIONE è vera

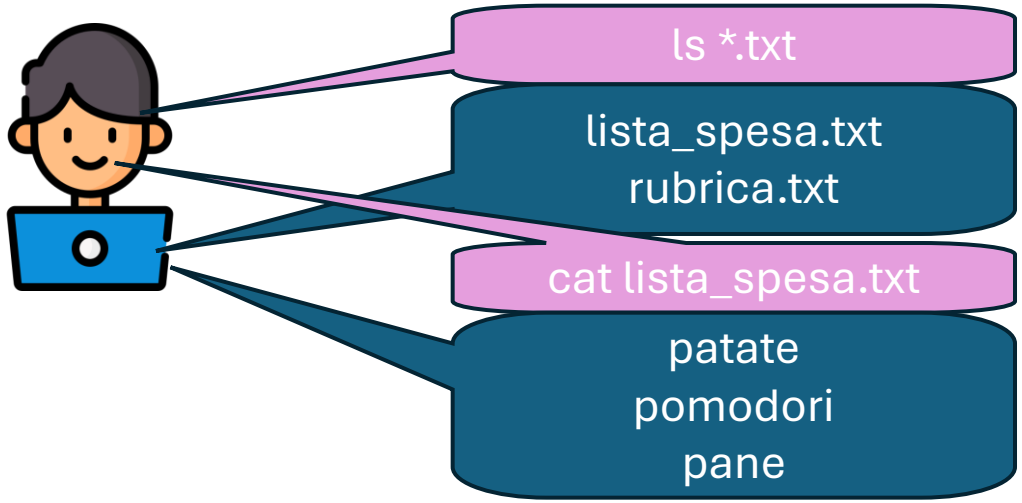
```
$ if [ 1 != 2 ]; then echo "sono diversi!"; fi
sono diversi!
```

Eseguire comandi vs scrivere un programma

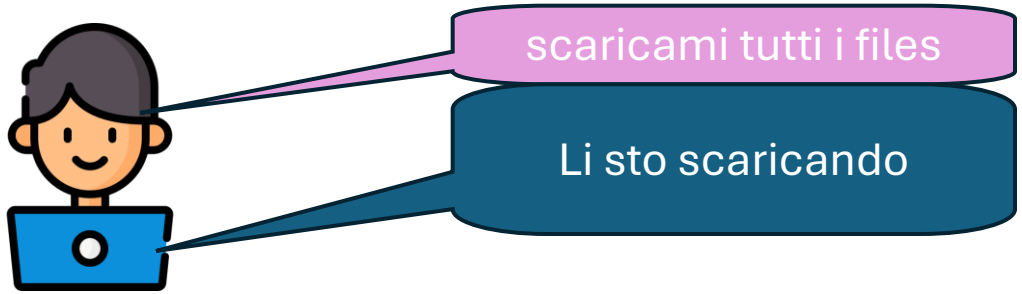




Eseguire comandi vs scrivere un programma



Eseguire comandi vs scrivere un programma

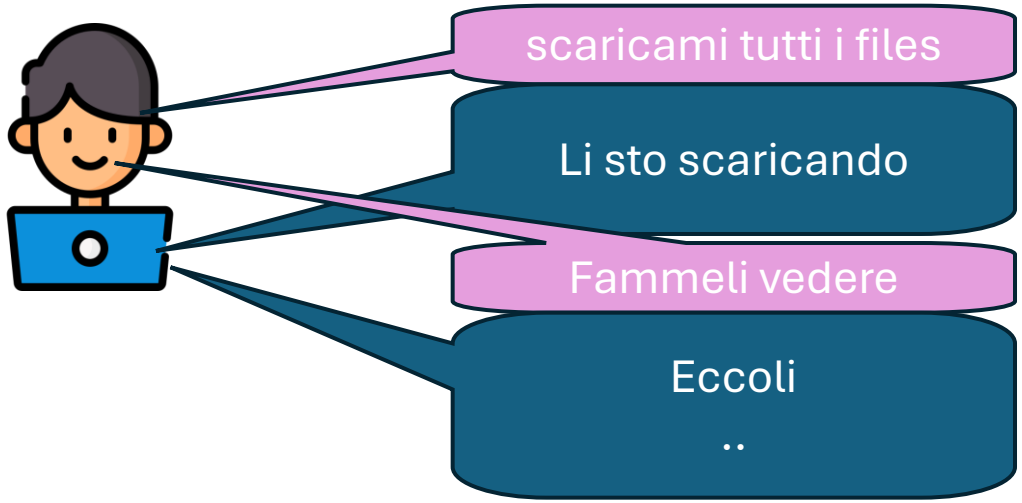



```
$ for i in $( seq 1 5 );do
> wget https://raw.githubusercontent.com/fabrimafe/GenomicaComputazionale2025_2026/refs/heads/main/sequences${i}.fastq
> done
--2025-10-02 06:07:34--  https://raw.githubusercontent.com/fabrimafe/GenomicaComputazionale2025_2026/refs/heads/main/sequences1.fastq
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.109.133, 185.199.108.133, 185.199.111.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.109.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 67422 (66K) [text/plain]
Saving to: ‘sequences1.fastq.1’

sequences1.fastq.1                  100%[=============================================================================>]  65.84K  --.-KB/s    in 0.06s

2025-10-02 06:07:35 (1020 KB/s) - ‘sequences1.fastq.1’ saved [67422/67422]
```

Eseguire comandi vs scrivere un programma

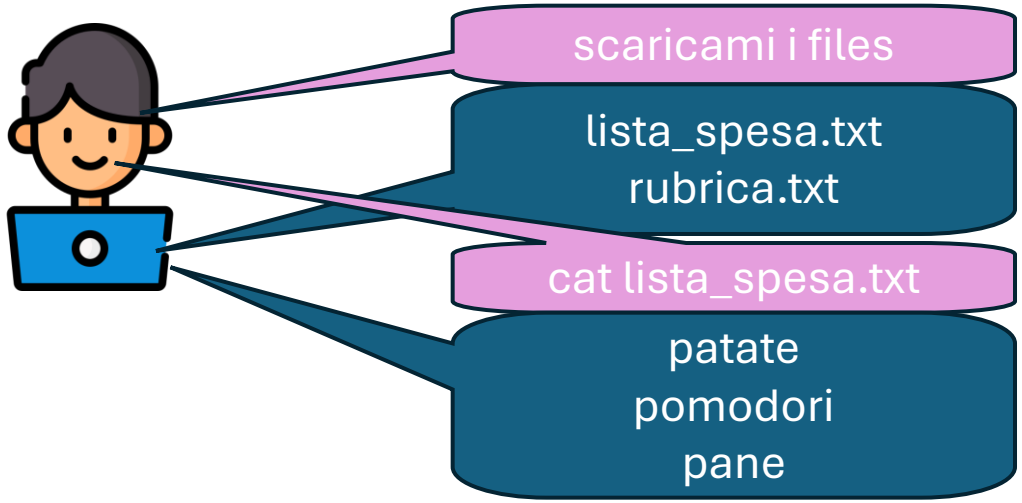



```
$ for i in $( seq 1 5 );do
> wget https://raw.githubusercontent.com/fabrimafe/GenomicaComputazionale2025_2026/refs/heads/main/sequences${i}.fastq
> done
--2025-10-02 06:07:34--  https://raw.githubusercontent.com/fabrimafe/GenomicaComputazionale2025_2026/refs/heads/main/sequences1.fastq
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.109.133, 185.199.108.133, 185.199.111.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.109.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 67422 (66K) [text/plain]
Saving to: ‘sequences1.fastq.1’

sequences1.fastq.1                  100%[=============================================================================>]  65.84K  --.-KB/s    in 0.06s

2025-10-02 06:07:35 (1020 KB/s) - ‘sequences1.fastq.1’ saved [67422/67422]
$ ls *fastq
sequences1.fastq  sequences2.fastq  sequences3.fastq  sequences4.fastq  sequences5.fastq
```

Eseguire comandi vs scrivere un programma



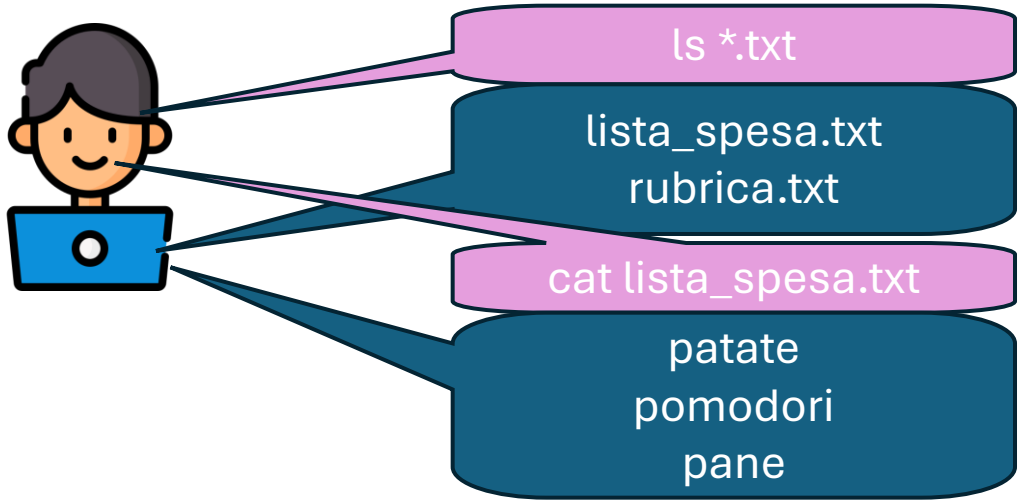

```
$ for i in $( seq 1 5 );do
> wget https://raw.githubusercontent.com/fabrimafe/GenomicaComputazionale2025_2026/refs/heads/main/sequences${i}.fastq
> done
--2025-10-02 06:07:34--  https://raw.githubusercontent.com/fabrimafe/GenomicaComputazionale2025_2026/refs/heads/main/sequences1.fastq
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.109.133, 185.199.108.133, 185.199.111.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.109.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 67422 (66K) [text/plain]
Saving to: ‘sequences1.fastq.1’

sequences1.fastq.1                  100%[=============================================================================>]  65.84K  --.-KB/s    in 0.06s

2025-10-02 06:07:35 (1020 KB/s) - ‘sequences1.fastq.1’ saved [67422/67422]
$ ls *fastq
sequences1.fastq  sequences2.fastq  sequences3.fastq  sequences4.fastq  sequences5.fastq
```

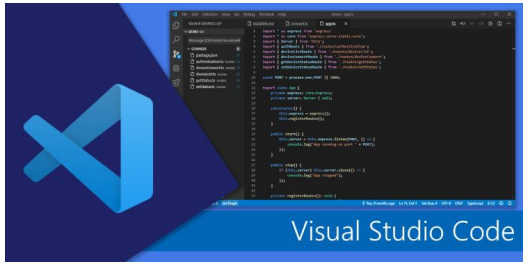
Eseguire comandi vs scrivere un programma

Interattivamente



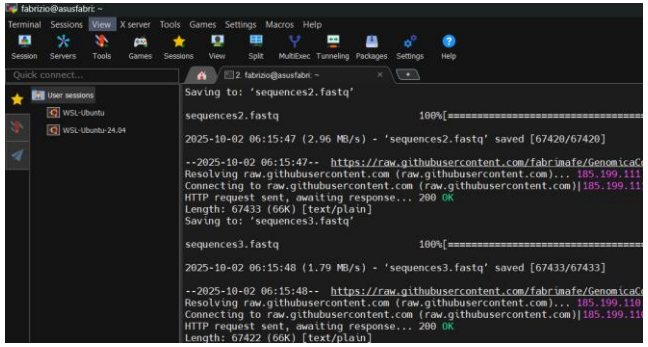
Usando un editor

Editor: VScode, word, etc.



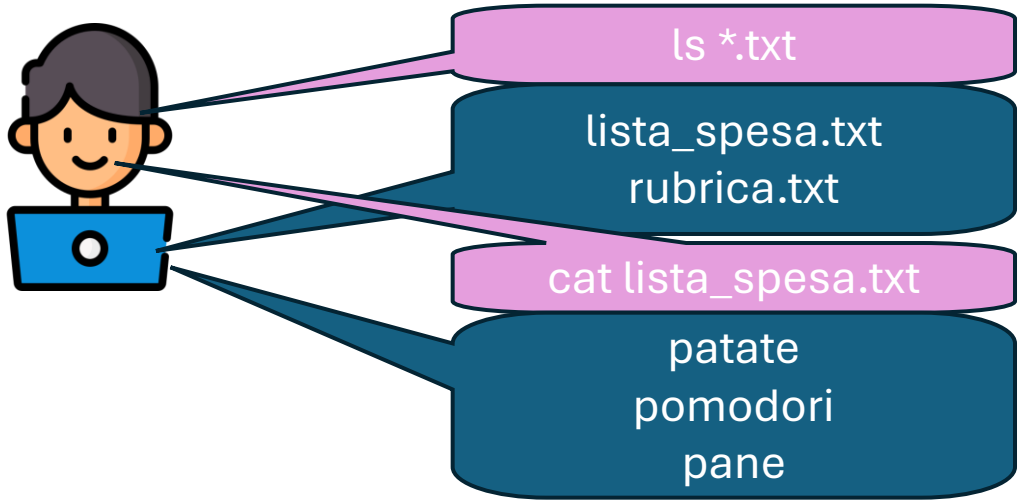
copia e incolla

Terminale: mobaXterm, iTerm



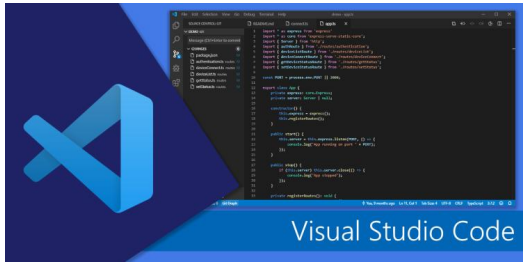
Eseguire comandi vs scrivere un programma

Interattivamente



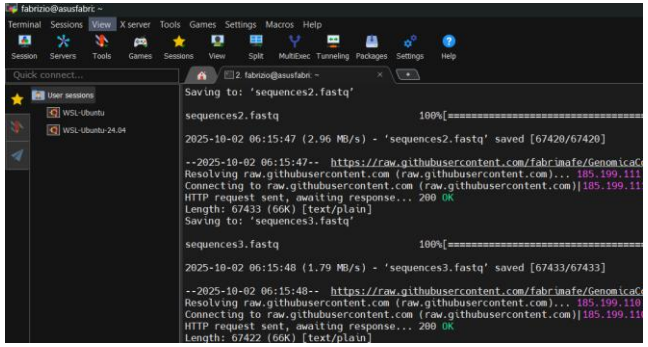
Usando un editor

Editor: VScode, word, etc.



copia e incolla

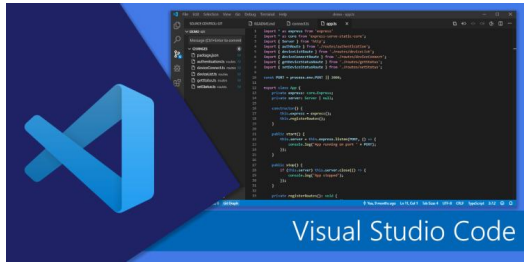
Terminale: mobaXterm, iTerm



Eseguire comandi vs scrivere un programma

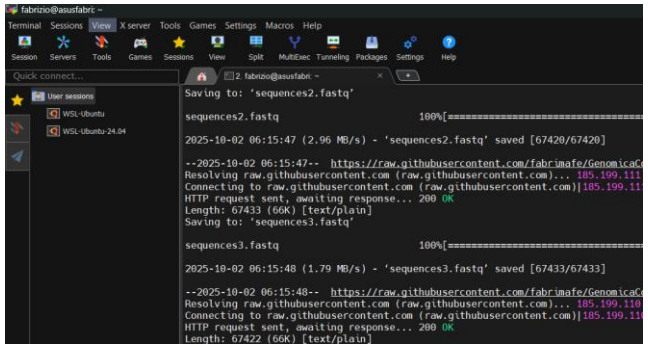
Usando un editor

Editor: VScode, word, etc.

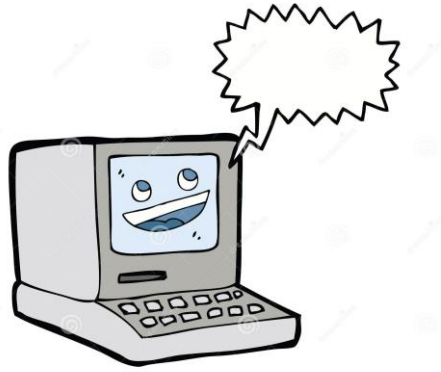


copia e incolla

Terminale: mobaXterm, iTerm

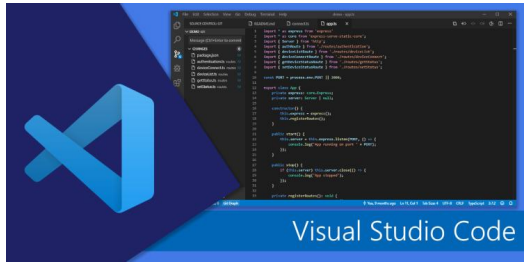


Eseguire comandi vs scrivere un programma



Esegui (su terminale)

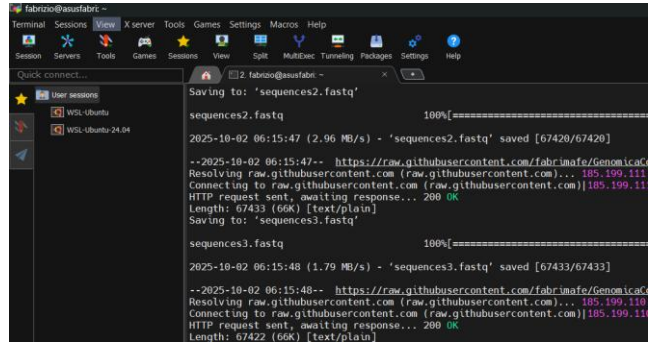
Editor: VScode, word, etc.



copia e incolla o salva in un file

file

Terminale: mobaXterm, iTerm

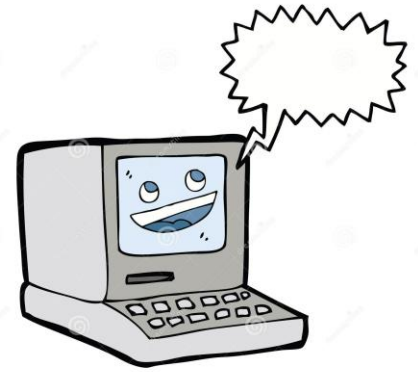


Eseguire comandi vs scrivere un programma

Rendete il programma eseguibile con `chmod +x`

```
$ chmod +x issunday.sh
$ ls -lh issunday.sh
-rwxr-xr-x 1 fabrizio fabrizio 181 Oct  2 06:59 issunday.sh
```

Owner Group Others



Editor: VScode, word, etc.

```
### Programma che dice se e' Domenica
#X is 1 if Sun, 0 otherwise
X=$( date | grep Sun | wc -l )
if [ $X == 1 ];then
        echo "E' Domenica!"
else
        echo "No, svegliati!"
fi
~
-- INSERT --
```

copia e incolla o salva in un file

file

Esegui (su terminale)

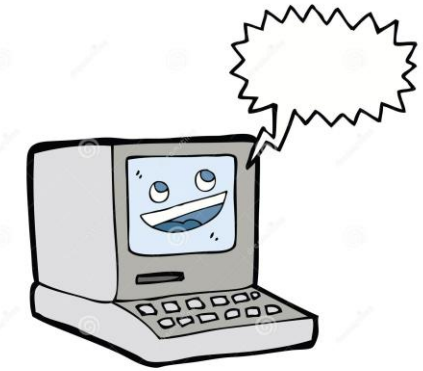
```
./issunday.sh
No, svegliati!
```

Eseguire comandi vs scrivere un programma

Rendete il programma eseguibile con `chmod +x`

```
$ chmod +x issunday.sh
$ ls -lh issunday.sh
-rwxr-xr-x 1 fabrizio fabrizio 181 Oct  2 06:59 issunday.sh
```

Owner Group Others



Editor: VScode, word, etc.

```
### Programma che dice se e' Domenica
#X is 1 if Sun, 0 otherwise
X=$( date | grep Sun | wc -l )
if [ $X == 1 ];then
        echo "E' Domenica!"
else
        echo "No, svegliati!"
fi
~
-- INSERT --
```

copia e incolla o salva in un file

file

Esegui (su terminale)

```
./issunday.sh
No, svegliati!
```

Eseguire comandi vs scrivere un programma

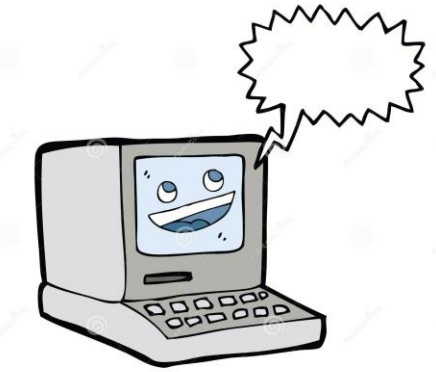
Specificare il linguaggio usato con la *shebang*, *una riga iniziale che inizia con #!*
(per bash #!/bin/bash)

```
#!/bin/bash
### Programma che dice se e' Domen
#X is 1 if Sun, 0 otherwise
X=$( date | grep Sun | wc -l )
if [ $X == 1 ];then
        echo "E' Domenica!"
else
        echo "No, svegliati!"
fi
```

Rendete il programma eseguibile con `chmod +x`

```
$ chmod +x issunday.sh
$ ls -lh issunday.sh
-rwxr-xr-x 1 fabrizio fabrizio 181 Oct  2 06:59 issunday.sh
```

Owner Group Others



Editor: VScode, word, etc.

```
### Programma che dice se e' Domenica
#X is 1 if Sun, 0 otherwise
X=$( date | grep Sun | wc -l )
if [ $X == 1 ];then
        echo "E' Domenica!"
else
        echo "No, svegliati!"
fi
~
-- INSERT --
```

copia e incolla o salva in un file

file

Esegui (su terminale)

```
./issunday.sh
No, svegliati!
```

Eseguire comandi vs scrivere un programma

Specificare il linguaggio usato con la *shebang*, *una riga iniziale che inizia con #!*
(per bash #!/bin/bash) LINUX

```
#!/bin/bash
### Programma che dice se e' Domen
#X is 1 if Sun, 0 otherwise
X=$( date | grep Sun | wc -l )
if [ $X == 1 ];then
        echo "E' Domenica!"
else
        echo "No, svegliati!"
fi
```

```
#!/bin/sh
#!/usr/bin/python
#!/usr/bin/env Rscript
```

POSIX, un "Linux" piu' basico

Editor: VScode, word, etc.

```
### Programma che dice se e' Domenica
#X is 1 if Sun, 0 otherwise
X=$( date | grep Sun | wc -l )
if [ $X == 1 ];then
        echo "E' Domenica!"
else
        echo "No, svegliati!"
fi
~
-- INSERT --
```

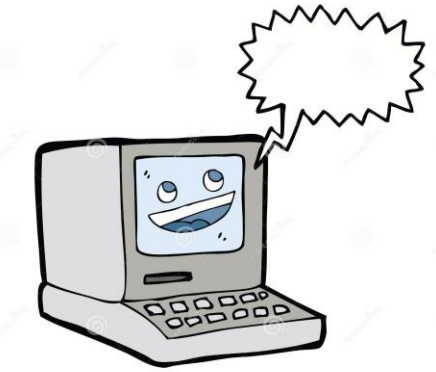
copia e incolla o salva in un file

file

Rendete il programma eseguibile con `chmod +x`

```
$ chmod +x issunday.sh
$ ls -lh issunday.sh
-rwxr-xr-x 1 fabrizio fabrizio 181 Oct  2 06:59 issunday.sh
```

Owner Group Others



Esegui (su terminale)

```
./issunday.sh
No, svegliati!
```

ESERCIZI

- Scrivete un piccolo programmino con un `for` e/o un `if`

Mettiamo tutto insieme e scriviamo un programma complesso e utile!

Esercizio:
Scriviamo un programma che scriva un report che, per ogni file **fastq** nella nostra cartella, scriva
1) il numero delle reads
2) il numero delle basi ad alta qualità (F)
3) il numero di adattatori

Per ottenere i files:

```
wget https://raw.githubusercontent.com/fabrimafe/GenomicaComputazionale2025_2026/refs/heads/main/sequences1.fastq
wget https://raw.githubusercontent.com/fabrimafe/GenomicaComputazionale2025_2026/refs/heads/main/sequences2.fastq
wget https://raw.githubusercontent.com/fabrimafe/GenomicaComputazionale2025_2026/refs/heads/main/sequences3.fastq
wget https://raw.githubusercontent.com/fabrimafe/GenomicaComputazionale2025_2026/refs/heads/main/sequences4.fastq
wget https://raw.githubusercontent.com/fabrimafe/GenomicaComputazionale2025_2026/refs/heads/main/sequences5.fastq
```

Le sequenze di DNA: il formato fastq

```
~$ head sequences1.fastq
@A00619:220:HYK7GDRX3:1:2101:13440:1000 1:N:0:ACTATAGA+TGCATACC
NCCCGTGAGATGATGCGCGTCATCCTCAACAAGGCCAAATCTGACCCCAAGCGACTCGTTCTCGCTGAGATCGGAAGAGCACACGTCTGAACTCCAGTCAC
+
#FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF:FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
@A00619:220:HYK7GDRX3:1:2101:18629:1000 1:N:0:ACTATAGA+TGCATACC
NGCTGTGGACTCCACCCGAGTTCCGGATCGAAACTGGTGAACTTATCCAGCAGCTCTCGGTCGATCCCCGGGAACTCCTCCAACACCACTGTTGGAAGACT
+
#FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
@A00619:220:HYK7GDRX3:1:2101:20745:1000 1:N:0:ACTATAGA+TGCATACC
NGGATATCAATCAACTCCACGCAACTCCAACTTCCCCTCAGATCGGAAGAGCACACGTCTGAACTCCAGTCACACTATAGAATCTCGGTTTGCGTCTTATG
```

- Riga1: @ riga con i nomi delle sequenze ed informazioni sul sequenziamento
- Riga2: Sequenza ACGT
- Riga3: +, un semplice separatore
- Riga4: Qualità delle sequenze

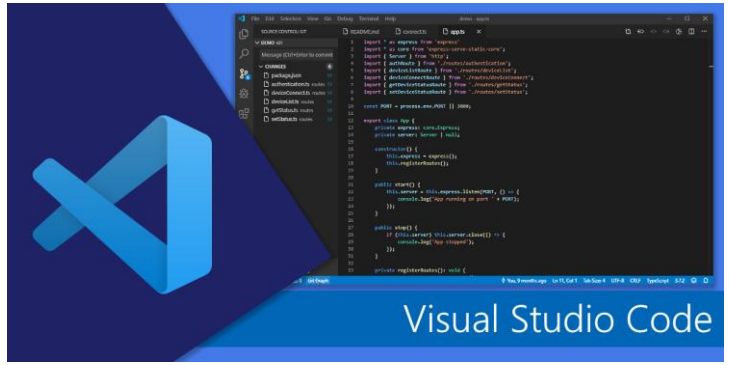
Mettiamo tutto insieme e scriviamo un programma complesso e utile!

Esercizio:
Scriviamo un programma che scriva un report che, per ogni file **fastq** nella nostra cartella, scriva
1) il numero delle reads
2) il numero delle basi ad alta qualità (F)
3) il numero di barcodes (anche chiamati indici o i5 i7 adapters)

```
NOME_REPORT=report.txt
for i in *fastq; do
cat $i | grep @ | wc -l > $NOME_REPORT
cat $i | sed 's/[^F]//g' | tr -d '\n' | wc -c >> $NOME_REPORT
cat $i | grep @ | sed "s/.*1:N:0://" | sort | uniq | wc -l >> $NOME_REPORT
done
```

Come creare un piccolo programma eseguibile

- Aprite il vostro editor, scrivete il vostro codice e salvatelo (possibilmente come file .sh)



- In alternativa, aprite vim e digitatelo/copiatelo dopo essere entrati in modalità INSERT (premere *i*)

```
~$ vim create_report_sequences.sh
```

- Infine uscite e rendete eseguibile il vostro file con

```
chmod +x ILVOSTROFILE
```

```
$ chmod +x create_report_sequences.sh
$ ls -lh create_report_sequences.sh
-rwxr-xr-x 1 fabrizio fabrizio 605 Sep 25 08:44 create_report_sequences.sh
```

- Potete verificare che il vostro file esiste ed è eseguibile con

```
ls -lh
```

create_report

```
~$ vim create_report_sequences.sh
```

o sul vostro editor

```
#!/bin/bash

NOME_REPORT="report.txt"
if [ -f $NOME_REPORT ];then
rm $NOME_REPORT
else
touch $NOME_REPORT
fi

for i in *fastq; do
echo "================" >> $NOME_REPORT
echo $i >> $NOME_REPORT
echo "----------------" >> $NOME_REPORT
#numero reads
echo "number of reads" >> $NOME_REPORT
cat $i | grep @ | wc -l >> $NOME_REPORT
#numero basi buone
echo "number of high quality (F) bases" >> $NOME_REPORT
cat $i | sed 's/[^F]//g' | tr -d '\n' | wc -c >> $NOME_REPORT
#numero adattatori unici
echo "number of barcodes" >> $NOME_REPORT
cat $i | grep @ | sed "s/.*1:N:0://" | sort | uniq | wc -l >> $NOME_REPORT
done
```

```
~$ ./create_report_sequences.sh report.txt
~$ cat report.txt
================
sequences1.fastq
----------------
number of reads
250
number of high quality (F) bases
22881
number of barcodes
9
================
sequences2.fastq
----------------
number of reads
250
number of high quality (F) bases
23064
number of barcodes
3
================
```

create_report

```
~$ vim create_report_sequences.sh
```

o sul vostro editor

```
#!/bin/bash

NOME_REPORT="report.txt"
if [ -f $NOME_REPORT ];then
rm $NOME_REPORT
else
touch $NOME_REPORT
fi

for i in *fastq; do
echo "================" >> $NOME_REPORT
echo $i >> $NOME_REPORT
echo "----------------" >> $NOME_REPORT
#numero reads
echo "number of reads" >> $NOME_REPORT
cat $i | grep @ | wc -l >> $NOME_REPORT
#numero basi buone
echo "number of high quality (F) bases" >> $NOME_REPORT
cat $i | sed 's/[^F]//g' | tr -d '\n' | wc -c >> $NOME_REPORT
#numero adattatori unici
echo "number of barcodes" >> $NOME_REPORT
cat $i | grep @ | sed "s/.*1:N:0://" | sort | uniq | wc -l >> $NOME_REPORT
done
```

Questo programma non è molto flessibile. Ad esempio usa sempre lo stesso output file (potremmo non voler sovrascriverlo ogni volta, e invece aggiornare periodicamente il nostro report con i fastq che sequenziamo nuovamente). Proviamo a renderlo piu' flessibile!

Argomenti posizionali in bash

- Per rendere flessibile un programma dobbiamo poter cambiare i suoi input e le sue funzionalità senza ogni volta dover modificare il programma.
- Per questo possiamo definire degli argomenti: dei valori che il nostro programma utilizzerà per le sue funzioni.
- In bash, possiamo definire argomenti posizionali semplicemente con $POSIZIONE_DELL_ARGOMENTO, ad esempio $1 per il primo argomento.

```
#!/bin/bash
#----Programma che conta le lettere in una parola---
WORD=$1 #L argomento 1 definisce una parola di input
NUMERO_CARATTERI=$( echo $WORD | wc -c )
echo "La tua parola ha $NUMERO_CARATTERI caratteri"
```

ESEMPIO: contalettere.sh

```
./contalettere.sh panamabananas
La tua parola ha 14 caratteri
```

- Posso anche definire files di input, output, numeri, etc.
- Per piu' argomenti, semplicemente posso usare $2, $3 per il secondo e terzo argomento, rispettivamente (e a seguire)
- E' SEMPRE utile **commentare** i programmi con informazioni utili usando # (come nell'esempio), soprattutto per definire gli argomenti, cosi che non dobbiate decifrare il vostro codice a posteriori

create_report

```
~$ vim create_report_sequences.sh
```

o sul vostro editor

```
#!/bin/bash

NOME_REPORT=$1
if [ -f $NOME_REPORT ];then
rm $NOME_REPORT
else
touch $NOME_REPORT
fi

for i in *fastq; do
echo "================" >> $NOME_REPORT
echo $i >> $NOME_REPORT
echo "----------------" >> $NOME_REPORT
#numero reads
echo "number of reads" >> $NOME_REPORT
cat $i | grep @ | wc -l >> $NOME_REPORT
#numero basi buone
echo "number of high quality (F) bases" >> $NOME_REPORT
cat $i | sed 's/[^F]//g' | tr -d '\n' | wc -c >> $NOME_REPORT
#numero adattatori unici
echo "number of barcodes" >> $NOME_REPORT
cat $i | grep @ | sed "s/.*1:N:0://" | sort | uniq | wc -l >> $NOME_REPORT
done
```

Prende il primo ***argomento*** ($1)

```
./create_report_sequences.sh ilmioreport_20ttobre2025.txt
```

```
~$ ./create_report_sequences.sh report.txt
~$ cat report.txt
================
sequences1.fastq
----------------
number of reads
250
number of high quality (F) bases
22881
number of barcodes
9
================
sequences2.fastq
----------------
number of reads
250
number of high quality (F) bases
23064
number of barcodes
3
================
```

create_report

```
~$ vim create_report_sequences.sh
```

o sul vostro editor

```
#!/bin/bash

NOME_REPORT=$1
if [ -f $NOME_REPORT ];then
rm $NOME_REPORT
else
touch $NOME_REPORT
fi

for i in *fastq; do
echo "================" >> $NOME_REPORT
echo $i >> $NOME_REPORT
echo "----------------" >> $NOME_REPORT
#numero reads
echo "number of reads" >> $NOME_REPORT
cat $i | grep @ | wc -l >> $NOME_REPORT
#numero basi buone
echo "number of high quality (F) bases" >> $NOME_REPORT
cat $i | sed 's/[^F]//g' | tr -d '\n' | wc -c >> $NOME_REPORT
#numero adattatori unici
echo "number of barcodes" >> $NOME_REPORT
cat $i | grep @ | sed "s/.*1:N:0://" | sort | uniq | wc -l >> $NOME_REPORT
done
```

Prende il primo ***argomento*** ($1)

```
./create_report_sequences.sh ilmioreport_20ttobre2025.txt
```

Un create_report migliore

```
cp create_report_sequences.sh create_report_sequences2.sh
vim create_report_sequences2.sh
```

```
#!/bin/bash

INPUTPATH=$1
NOME_REPORT=$2

echo "------------------------------------------------------------
Script that summarize a set of fastq files given an input path with the selected fastq (argument 1) and an output report (argument 2).
Example usage: create_report_sequences2.sh fastq_folder/sequences report.txt
Writing $NOME_REPORT for files:
$( ls ${INPUTPATH}*fastq )"
echo "------------------------------------------------------------"

if [ -f $NOME_REPORT ];then
rm $NOME_REPORT
else
touch $NOME_REPORT
fi

for i in ${INPUTPATH}*fastq; do
echo "================" >> $NOME_REPORT
echo $i >> $NOME_REPORT
echo "---------------" >> $NOME_REPORT
#numero reads
echo "number of reads" >> $NOME_REPORT
cat $i | grep @ | wc -l >> $NOME_REPORT
#numero basi buone
echo "number of high quality (F) bases" >> $NOME_REPORT
cat $i | sed 's/[^F]//g' | tr -d '\n' | wc -c >> $NOME_REPORT
#numero adattatori unici
echo "number of barcodes >> $NOME_REPORT
cat $i | grep @ | sed "s/.*1:N:0://" | sort | uniq | wc -l >> $NOME_REPORT
done
```

Il primo argomento è il «path» o percorso (nome del file/s, inclusa la cartella in cui si trovano)
Il secondo è il nome del report

Messaggio di aiuto che ci dice come usare lo script e cosa farà

./Nome_dello_script

Argomento 1 (nome dei files)

Argomento 2 (nome del report)

Utilizzo:

```
$ ./create_report_sequences2.sh sequences report.txt
```

Un create_report migliore

```
$ pwd
/home/fabrizio
$ ./create_report_sequences2.sh sequences report.txt
-----------------------------------------------------------
Script that summarize a set of fastq files given an input path with the selected fastq (argument 1) and an output report (argument 2).
Example usage: create_report_sequences2.sh fastq_folder/sequences report.txt
Writing report.txt for files:
sequences1.fastq
sequences2.fastq
sequences3.fastq
sequences4.fastq
sequences5.fastq
```

Messaggio di aiuto

Un create_report migliore

```
$ pwd
/home/fabrizio
$ ./create_report_sequences2.sh sequences report.txt
-----------------------------------------------------------
Script that summarize a set of fastq files given an input path with the selected fastq (argument 1) and an output report (argument 2).
Example usage: create_report_sequences2.sh fastq_folder/sequences report.txt
Writing report.txt for files:
sequences1.fastq
sequences2.fastq
sequences3.fastq
sequences4.fastq
sequences5.fastq
-----------------------------------------------------------
$ ./create_report_sequences2.sh sweet_potato/ sweet_potato/report.txt
-----------------------------------------------------------
Script that summarize a set of fastq files given an input path with the selected fastq (argument 1) and an output report (argument 2).
Example usage: create_report_sequences2.sh fastq_folder/sequences report.txt
Writing sweet_potato/report.txt for files:
sweet_potato/sweet-potato-chloroplast-illumina-reduced.fastq
sweet_potato/sweet-potato-chloroplast-nanopore-reduced.fastq
-----------------------------------------------------------
```

Messaggio di aiuto

Così possiamo usare lo script su ogni cartella, o selezionare alcuni file specifici, anche senza spostarlo o modificarlo

Un create_report migliore

```
$ pwd
/home/fabrizio
$ ./create_report_sequences2.sh sequences report.txt
-----------------------------------------------------
Script that summarize a set of fastq files given an input path with the selected fastq (argument 1) and an output report (argument 2).
Example usage: create_report_sequences2.sh fastq_folder/sequences report.txt
Writing report.txt for files:
sequences1.fastq
sequences2.fastq
sequences3.fastq
sequences4.fastq
sequences5.fastq
-----------------------------------------------------
$ ./create_report_sequences2.sh sweet_potato/ sweet_potato/report.txt
-----------------------------------------------------
Script that summarize a set of fastq files given an input path with the selected fastq (argument 1) and an output report (argument 2).
Example usage: create_report_sequences2.sh fastq_folder/sequences report.txt
Writing sweet_potato/report.txt for files:
sweet_potato/sweet-potato-chloroplast-illumina-reduced.fastq
sweet_potato/sweet-potato-chloroplast-nanopore-reduced.fastq
-----------------------------------------------------
$ mkdir scripts
$ mv create_report_sequences2.sh scripts/
$ scripts/create_report_sequences2.sh sequences report.txt
scripts
$ scripts/create_report_sequences2.sh sequences report.txt
-----------------------------------------------------
Script that summarize a set of fastq files given an input path with the selected fastq (argument 1) and an output report (argument 2).
Example usage: create_report_sequences2.sh fastq_folder/sequences report.txt
Writing report.txt for files:
sequences1.fastq
sequences2.fastq
sequences3.fastq
sequences4.fastq
sequences5.fastq
-----------------------------------------------------
```

Messaggio di aiuto

Cosi possiamo usare lo script su ogni cartella, o selezionare alcuni file specifici, anche senza spostarlo o modificarlo

E’ cosi utile e flessibile che potremmo pernsare di salvarlo in una cartella scripts cosi poi da poterlo riusare ogni volta

Tenete da conto il lavoro fatto quando può tornare utile!

Può essere utile salvare tutti i vostri scripts in una cartella «scripts», «bin» o «comandi» che potrete riusare sempre

```
$ pwd
/home/fabrizio/sweet_potato
$ ~/scripts/create_report_sequences2.sh ./ report.txt
```

oppure (con lo stesso significato)

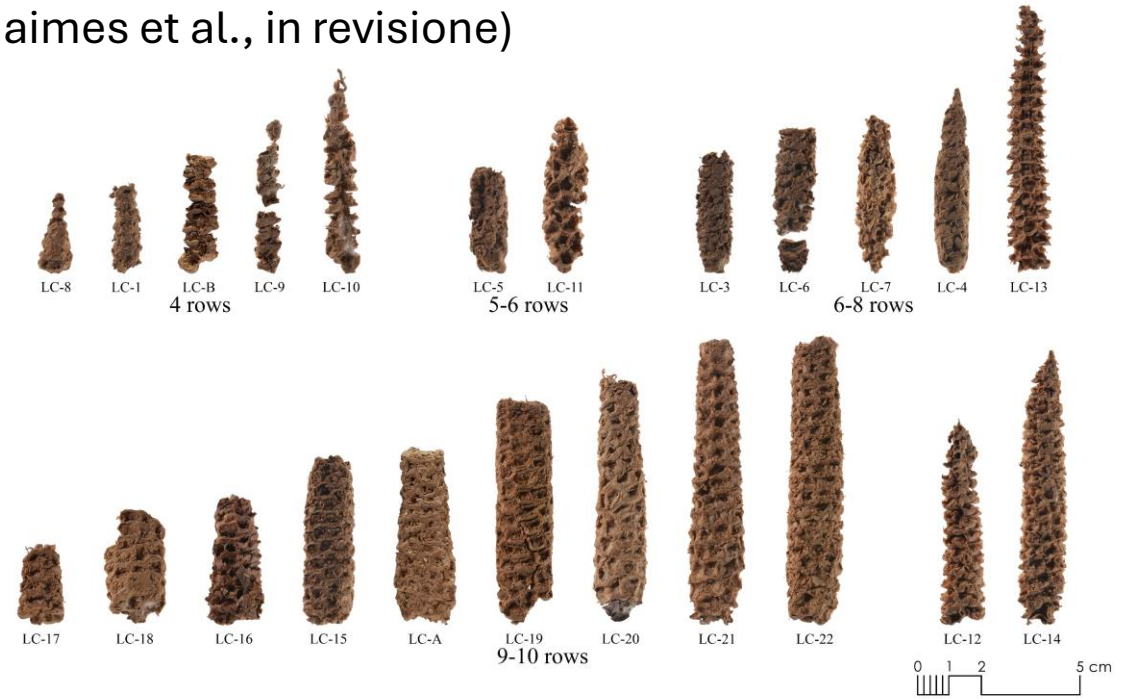
```
$ ${HOME}/scripts/create_report_sequences2.sh ./ report.txt
```

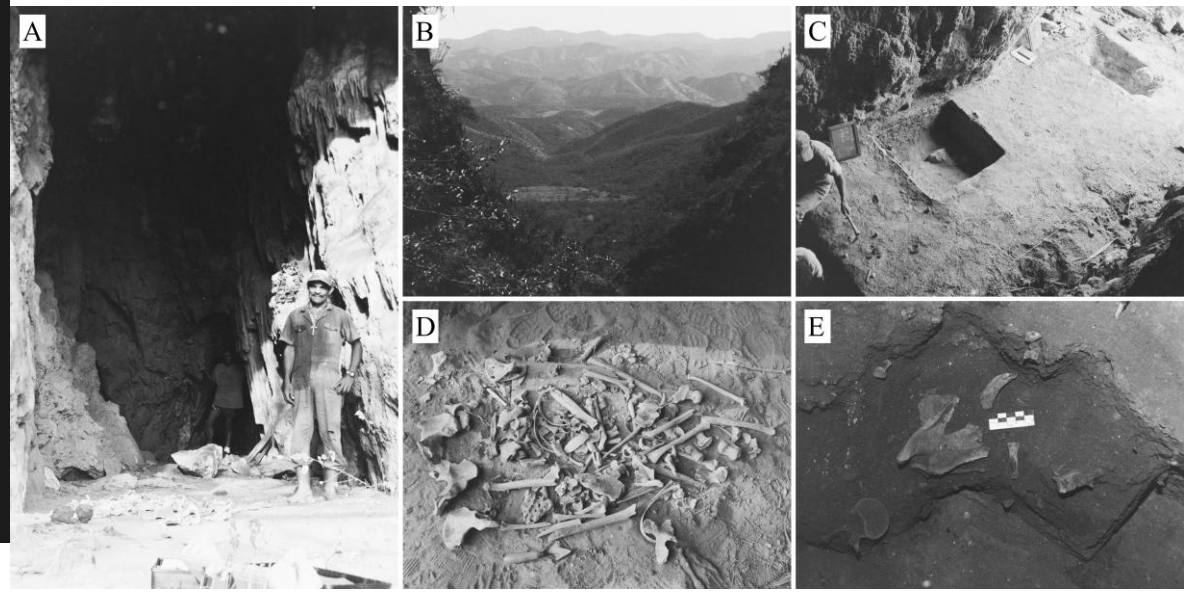
E' cosi utile e flessibile che potremmo pernsare di salvarlo in una cartella scripts cosi poi da poterlo riusare ogni volta

```
$ mkdir scripts
$ mv create_report_sequences2.sh scripts/
$ scripts/create_report_sequences2.sh sequences report.txt
scripts
$ scripts/create_report_sequences2.sh sequences report.txt
------------------------------------------------------------
Script that summarize a set of fastq files given an input path with the selected fastq (argument 1) and an output report (argument 2).
Example usage: create_report_sequences2.sh fastq_folder/sequences report.txt
Writing report.txt for files:
sequences1.fastq
sequences2.fastq
sequences3.fastq
sequences4.fastq
sequences5.fastq
------------------------------------------------------------
```

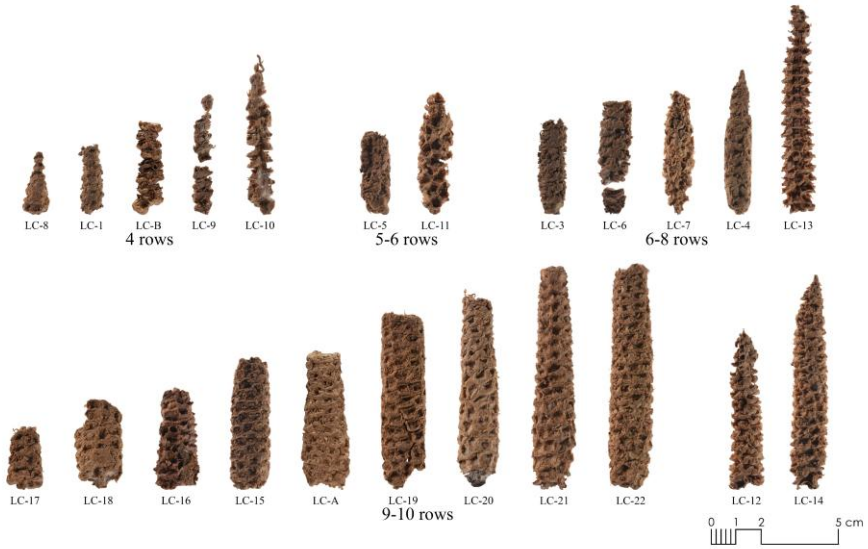
```
~$ ./create_report_sequences.sh report.txt
~$ cat report.txt
================
sequences1.fastq
----------------
number of reads
250
number of high quality (F) bases
22881
number of barcodes
9
================
sequences2.fastq
----------------
number of reads
250
number of high quality (F) bases
23064
number of barcodes
3
================
```

Mais di circa 2000 anni fa dal Venezuela
(Jaimes et al., in revisione)





Esercizi



- Il genoma del mais è ripetuto all'85%
- Tra le ripetizioni piu' abbondanti troviamo le ripetizioni tandem (microsatelliti) di AT e CAC
- Esercizio 1: *Trovate tutte le ripetizioni di almeno due volte di AT e CAC*
- Esercizio 2: *Calcolare la distribuzione di contenuto GC (la proporzione di C e G sul totale delle basi) per ogni sequenza*

Il formato **FASTA**

- Generalmente utilizzato per rappresentare genomi e genomi di referenza. Il risultato finale di un assemblaggio!
- Il piu' semplice dei formati

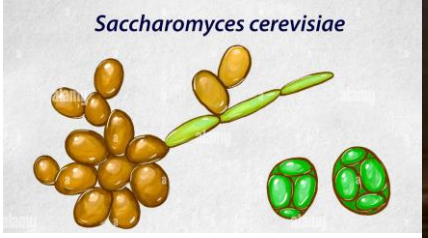
```
>nome_cromosoma_o_sequenza1
ATGGAATAAGGTATANATTTTAAACCCTTAATATAAT…..
>nome_cromosoma_o_sequenza2
GGTTATCCTATATCATTTTTAACCTAGGGGGGGGA…..
```

Il formato **FASTA**

- Generalmente utilizzato per rappresentare genomi e genomi di referenza
- Il piu' semplice dei formati
- Non ha informazione su qualità (come i fastq) quindi alcuni caratteri possono essere soft masked (lettere minuscole) o hard-masked (con delle N):

```
>nome_cromosoma_o_sequenza1
atggaataaggtNNNNNNNNNNATTTTAAACCTCT…..
>nome_cromosoma_o_sequenza2
GGTTATCCTATATCATTTTTAACCtaggggggGGA…..
```

Esercizio



- Potete scaricare il primo cromosoma eucariotico sequenziato (1996)

```
wget http://hgdownload.soe.ucsc.edu/goldenPath/sacCer3/chromosomes/chrI.fa.gz
```

- Il piu' piccolo. Una volta fatti scaricate in un loop gli altri 15 fino a:

```
wget http://hgdownload.soe.ucsc.edu/goldenPath/sacCer3/chromosomes/chrXVI.fa.gz
```

- Potete poi decomprimerli con

```
gunzip chrI.fa.gz
```

- oppure visualizzarli con la version di cat per files zippati

```
zcat chrI.fa.gz
```

TIPS

- **Ricordate tr -d '\n' e sed**
- **Può esservi utile (non necessariamente) una opzione di grep, grep –o (stampa solo il pattern trovato e non tutta la riga)**

- Esercizi:
  1. Mascherate con delle N (hard-masking) i microsatelliti con 3 o piu' ripetizioni di TA e AAT per il primo cromosoma
  2. Mascherate soft (rendendo le lettere minuscole) i microsatelliti con 3 o piu' ripetizioni di TA e AAT per il primo cromosoma
  3. Calcolate la percentuale mascherata per cromosoma e per il genoma
  4. Ripetete per tutti i cromosomi in un loop e trovate il cromosoma con piu' microsatelliti
  5. Effettuate la stessa operazione per il cromosoma 21 umano:

```
wget http://hgdownload.soe.ucsc.edu/goldenPath/hg38/chromosomes/chr21.fa.gz
```

Domande


Per la pratica e lo studio consiglio di consultare la «Guida ai comandi Linux» su Moodle, partendo dall’ «Un esempio istruttivo»

Errata corrige

- In una versione precedente delle slides avevo utilizzato il termine «adaptors» al posto di «barcodes». Questo perché si usano vari termini per indicare i «barcodes» usati per il demultiplexing, tra cui piu' specificamente «i5-i7 adapters», e a volte «indici». Per evitare confusione con gli adattatori usati in altri contesti ho sostituito il termine con «barcodes».
- Notate che le informazioni sul sequenziamento trovate nel nome della read/prima riga dei file fastq non sono obbligatorie nel formato fastq, e possono variare a seconda della piattaforma di sequenziamento.