Assemblaggio del genoma di cloroplasto di patata dolce

Genomica Computazionale 2025/2026, Laurea Specialistica in Genomica Funzionale, Università di Trieste, Fabrizio Mafessoni, Tutor Daniela Eugenia Nerelli

In questo tutorial guarderemo all'assemblaggio di un genoma di cloroplato usando dati da un articolo scientifico (Zhou et al., 2018)

1. **Scaricare i dati**

Potete scaricare i dati (delle sequenze lunghe oxford nanopore) con

```
wget https://zenodo.org/record/3567224/files/sweet-potato-chloroplast-nanopore-reduced.fastq
```

Sono anche disponibili delle sequenze corte illumina (che non useremo qui, ma per le quali potete effettuare "polishing del genoma", non trattato a lezione ma secondo le indicazioni di questo tutorial online) scaricabili con

```
wget https://zenodo.org/record/3567224/files/sweet-potato-chloroplast-illumina-reduced.fastq
```

Queste sono reads fastq, che come visto a lezione è un formato caratterizzato da

```
@AFAFAHFA                nome sequenza e qualità
ACACTA                   sequenza
+                        un separatore visuale, +
#FFFFFFF : qualità       qualità base per base
```

Potete visualizzarle con

```
cat sweet-potato-chloroplast-nanopore-reduced.fastq
```

Notate come le sequenze nanopore siano ben piu' lunghe di quelle illumina

```
cat sweet-potato-chloroplast-illumina-reduced.fastq
```

Invece di visualizzare tutto il file potete usare il comando `head`, oppure less. Soprattutto per le sequenze nanopore, può essere comodo visualizzare le sequenze con l'opzione `less -S`, che non fa andare a capo la linea e vi permettere di scorrere a destra con le freccine.

```
less -S sweet-potato-chloroplast-nanopore-reduced.fastq
```

Opzionalmente potete esplorare i files fastq con fastqc. Il programma fastqc è installabile sia con conda

```
conda install -c bioconda fastqc
```

che con apt e brew.

Ubutu/windows: `sudo apt install fastqc`

Mac: `brew install fastqc`

Fastqc si utilizza soprattutto per esplorare reads illumina, generando un file html che potete aprire con qualsiasi browser. Nel nostro caso non le useremo, quindi questa esplorazione è opzionale. Per generare l'html:

```
fastqc sweet-potato-chloroplast-illumina-reduced.fastq -o
Illumina_fastq
```

2. Assemblare il genoma

2.1 Installazione programmi di assemblaggio

Per assemblare possiamo utilizzare vari assemblatori, con algoritmi leggermente differenti. Idealmente, installate Flye ed assemblate utilizzando questo programma. Nel caso otteniate errori durante l'installazione o durante l'esecuzione (potrebbe accadere per mancanza di memoria) usate raven o miniasm:

Flye:

approccio ibrido di De Brujin e Overlap-graphs;

in teoria piu' accurato ma richiede piu' tempo e memoria (potrebbe darvi errori std_alloc dovuti alla mancanza di memoria sul computer;

installabile con conda

```
conda install Flye
```

raven:

Basato su Overlap-graphs;

buona performace e piu' leggero di flye.

Installabile con conda:

```
conda install -c bioconda raven-assembler
```

miniasm:

Basato su Overlap-Graph;

molto rapido e leggero, ma richiede cura, "polishing" e non accuratissimo

Installabile con conda, ma anche direttamente con apt o brew. Nello specifico, con conda:

```
conda install -c bioconda minimap2 miniasm
```

Su windows o Ubuntu, usando apt:

`sudo apt install -y minimap2 miniasm` (ricordate la password quando usate `sudo`. Consiglio (non strettamente necessario) per questo anche l'installazione di racon:

```
sudo apt install -y racon
```

Usando invece brew sul mac:

```
brew install minimap2

minimap2 –version

brew tap brewsci/bio

brew install miniasm

miniasm 2>&1 | head -n1   # per controllare
```

Su brew potreste necessitare di dover prima eseguire:

```
xcode-select --install
```

2.2 Esecuzione dell'assemblagio

Con i rispettivi programmi potete usare i seguenti comandi:

Flye[1]

```
flye --nano-raw sweet-potato-chloroplast-nanopore-
reduced.fastq --genome-size 160000 -o Assembly_
```

Raven

```
raven -t 4 sweet-potato-chloroplast-nanopore-reduced.fastq >
raven.fasta
```

---

[1] Flye potrebbe darvi errori di memoria, del tipo

....

ERROR: Caught unhandled exception: std::bad_alloc [2025-10-09 15:21:34] ERROR: flye-modules(+0x3bf6d) [0x5b30aa884f6d] [2025-10-09 15:21:34] ERROR: /home/wsl--install/miniconda3/envs/Assembly/bin/../lib/libstdc++.so.6(+0xbb747) [0x7bde25876747]

....

In quel caso potete provare a cambiare alcune opzioni aggiungendo la flag `(--asm-coverage 30` o meno) o evitare limiti preimpostanti di memoria che potreste avere sul vostro computer con `ulimit -v unlimited`, o forzando il sistema ad usare una cartella temporanea piu' grande (`export TMPDIR=$HOME/tmp; mkdir -p "$TMPDIR"`). Ma è possibile anche semplicemente che non riesca sul vostro computer!!! Usate a quel punto raven o miniasm senza rimorsi :D.

***Miniasm*[2]**

```
minimap2 -x ava-ont -t4 sweet-potato-chloroplast-nanopore-reduced.fastq   sweet-potato-chloroplast-nanopore-reduced.fastq > reads.paf
```

```
miniasm -f sweet-potato-chloroplast-nanopore-reduced.fastq reads.paf > asm.gfa
```

```
awk '/^S/{print ">"$2"\n"$3}' asm.gfa > asm.raw.fasta
```

3. Esplorazione dei risultati

Per i vari programmi di assemblaggio, avrete ottenuto dei risultati in formato fasta e (per Flye e miniasm in gfa). Il risultato principale sono le sequenze dei contig assemblati. Questi sono presentati in formato fasta, e potete semplicemente visualizzarli con `cat`, `head` o `less`. Il formato fasta rappresenta infatti semplicemente il nome di una sequenza (preceduto da >) e a seguire la sequenza, ed è tipicamente il formato utilizzato le sequenze genomiche. Spesso, se scaricate genomi da NCBI, troverete questi file con un'estensione di fatto equivalente (.fna). Un esempio di file in formato fasta (o .fna, per cui tipicamente le sequenze sono contigs or cromosomi interi):

```
>Nome_sequenza
ACGGGTTAA...
```

Quindi, potete ad esempio esplorare il numero di contigs in un filefasta chiamato filefata con

```
cat filefasta | grep '>' | wc -l
```

e la loro lunghezza totale con

```
cat filefasta | grep -v '>' | wc -c
```

---

[2] Sarebbe successivamente consigliabile con miniasm fare *polishing* delle reads per migliorare l'assemblaggio con:

```
minimap2 -x map-ont -t4 asm.racon1.fasta sweet-potato-chloroplast-nanopore-reduced.fastq > map2.paf
racon -t4 sweet-potato-chloroplast-nanopore-reduced.fastq map2.paf asm.racon1.fasta > asm.racon2.fasta
```

```
(Assembly) fabrizio@asusfabri:~/PraticaAssemblaggio$ head /home/fabrizio/PraticaAssemblaggio/Assembly_/assembly.fasta
>contig_1
CTTCGTTCGTTGTGTACTAATTAACAAAATTTTCGCTTTCTACAATTGATGAAAGGAAGG
AATCAAACAAATTTTCTTATACTTCTCATCCCTTTGAATCAGAAAATACTATGTTATTTC
TACAATTCTATTGTGTTTTGTAACTTTGTAATTGGATTCTTAGAATTCCCAAGGAGCTTG
GTTTCAATCTCGACATCTTAACCCAATGGTTACATCCTGCTATTTATCTTTGGATCACAC
TAATTCAACAGATTTAGCTGAGTTCTTAAAGCATACGGTAATTTCCGTGGGATTGCTTAT
TGCGGAATATTTATAGCATTTTATTATATAAACCCACCTATTCGTCTTTCAAAGTTTTCT
ATTAATTAATTCGTTTCATCAAAAAGCCTTAGAGAGTTATTCGTGACAAAATAGTCTTTG
TGATATATGACTGGACCTATAATCGTGCTTATATAGATACTTTTATAAGAATTTTTGTTT
GCCAAGTGCGAAGGTTTCTCAAATTATTCGTTGTTTTGATTTAAATTATTGATCTAATAT
```

```
(Assembly) fabrizio@asusfabri:~/PraticaAssemblaggio$ cat /home/fabrizio/PraticaAssemblaggio/Assembly_/assembly.fasta | grep '>'
>contig_1
>contig_2
>contig_3
>contig_6
```

```
(Assembly) fabrizio@asusfabri:~/PraticaAssemblaggio$ cat /home/fabrizio/PraticaAssemblaggio/Assembly_/assembly.fasta | grep -v '>' | wc -c
206203
```

Il formato gfa descrive invece la struttura completa del genoma assemblato, incluse le potenziali connessioni tra i vari contigs (indicati come *edges*). Ad esempio, nel caso di Flye, abbiamo possiamo vedere con

```
less assembly.fasta
```

le sequenze dei vari contigs ricostruiti, mentre nel file gfa dovreste notare una struttura del genere

```
cat assembly_graph.gfa | less -S
```

```
H       VN:Z:1.0
S       edge_1  CTTCGTTCGTTGTGTACTAATTAACAAAATTTTCGCTTTCTACAATTGATGAAAGGAAGGAA
S       edge_2  GCAATACTTGAAGCGGCGATCCAGAAAACCCCAATACTAAGATCGGCTAAACAAGCTTATAA
S       edge_3  TTTTTAATAGATGATAGATATATATGTATGATAGATATATATGAACAGTAAGAACTAGCATT
S       edge_4  AATACTCCTGGGTGACCGATAGCGAAGTAGTACCGTGAGGAAGGGTGAAAGAACCCCAGCGC
S       edge_5  AAGGCTAAATACTCCTGGGTGACCGATAGCGAAGTAGTACCGTGAGGGAAGGGTGAAAAGAA
S       edge_6  TCCCGTTGGACTTCTTATGTCCGGATATGGATCAAATAATAAATATTCCTTTTAGAGAGAAT
L       edge_2  +       edge_3  -       0M      RC:i:51
L       edge_2  +       edge_3  +       0M      RC:i:73
L       edge_2  -       edge_6  -       0M      RC:i:86
L       edge_2  -       edge_6  +       0M      RC:i:64
P       contig_1        edge_1+ *
P       contig_3        edge_2+,edge_3+,edge_2- *
P       contig_6        edge_2-,edge_6+,edge_2+ *
P       contig_2        edge_2+ *
P       contig_4        edge_4+ *
P       contig_5        edge_5+ *
(END)
```

Qui gli edges rappresentano delle sequenze assemblate. Le potenziali connessioni tra due edges sono rappresentate in *links* (L nella prima colonna) e quando organizzate in contigs, potenzialmente anche coinvolgendo piu' edges, come *paths* (P nella prima colonna). I links rappresentano potenziali connessioni tra due edges successivi (ad esempio per il primo link, edge_2 ed edge_3, il primo in direzione forward (+) ed il secondo reverse (-), con 51 sequencing reads in supporto di questo link. I paths corrispono ai contigs riportati nel file fasta, e descrivono "percorsi" tra edges.

Un modo carino per visualizzare questi files è con Bandage. Bandage può essere installato con conda o mamba

```
mamba install -c bioconda bandage
```

o su windows/ubuntu con apt:

```
sudo apt install bandage
```

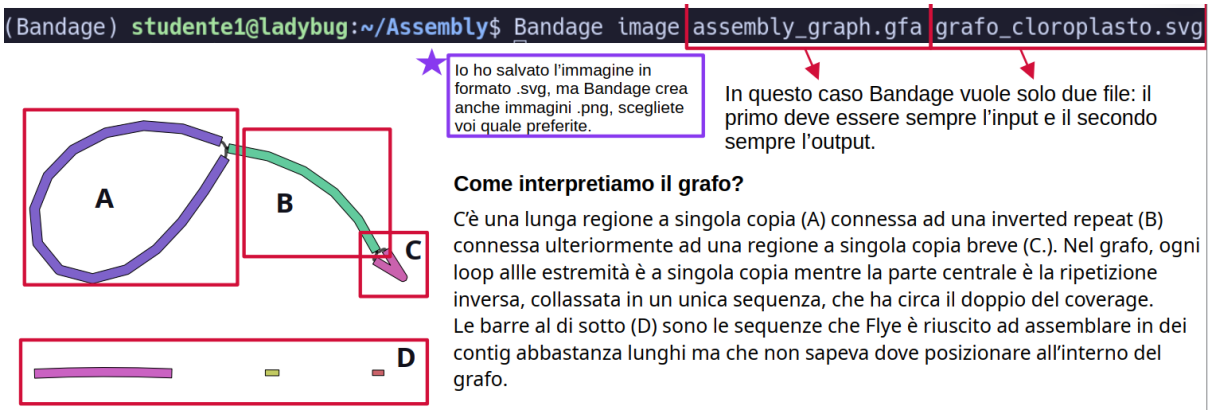
o su mac con brew

```
brew install --cask bandage
```

Come molti programmi grafici, potrebbe dare qualche problema nell'installazione e richiedere altre dipendenze. Nel caso abbiate problemi, molte delle librerie necessarie sono incluse in chrome[3].

Per eseguire Bandage

```
Bandage image assembly_graph.gfa grafo_cloroplasto.svg
```



[3] Per installare chrome:

```
wget -qO- https://dl.google.com/linux/linux_signing_key.pub | sudo gpg --dearmor -o
/usr/share/keyrings/google-chrome.gpg
echo "deb [arch=amd64 signed-by=/usr/share/keyrings/google-chrome.gpg]
http://dl.google.com/linux/chrome/deb/ stable main" | sudo tee /etc/apt/sources.list.d/google-
chrome.list
sudo apt update; sudo apt install -y google-chrome-stable
```

o su mac, con homebew

```
brew install --cask google-chrome
```

or download the .dmg from Google and install normally