Analisi di k-mer in dati FASTQ

Progetto di Elementi di Bioinformatica

A.A.2024-2025

Anna Giulia Granziera, Mat.895402

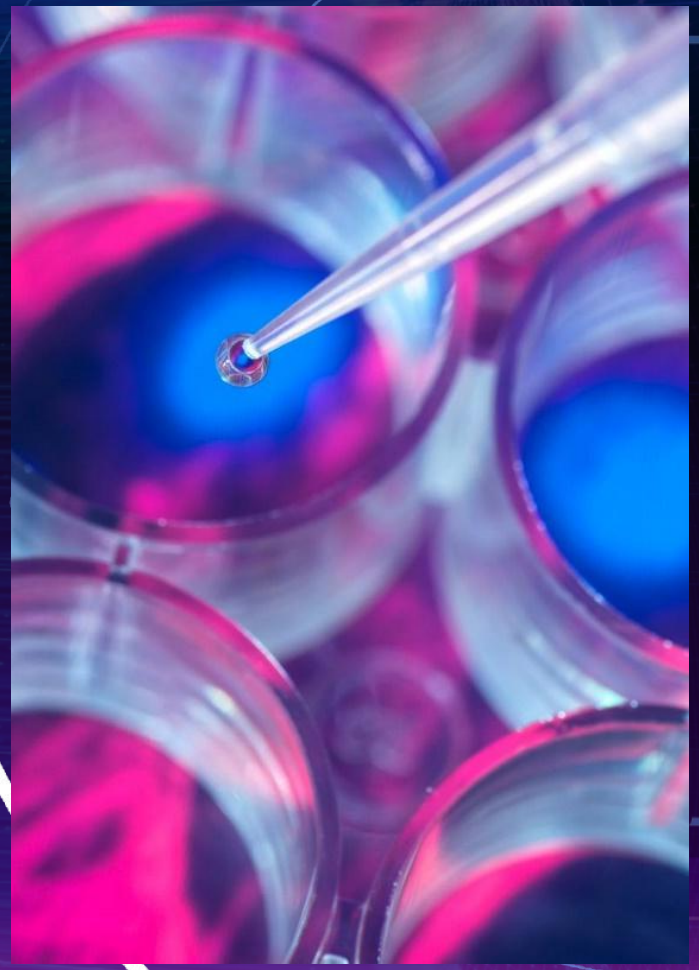
Introduzione

Il progetto si concentra sull'analisi di file FASTQ contenenti sequenze genomiche (**reads**) di pari lunghezza (**152 nucleotidi**)

L'obiettivo principale è esplorare come specifici **k-mer** , frammenti di sequenza di lunghezza definita, si distribuiscono lungo le reads, rivelando possibili pattern posizionali ricorrenti.

Obiettivi principali:

- Studiare la *distribuzione posizionale* dei k-mer
- Identificare *k-mer dominanti* nel dataset
- *Filtrare* le reads in base alla presenza del k-mer dominante e alla qualità
- Produrre *statistiche*, *visualizzazioni* e *output* in formato **FASTA**



Parametri di Input

Controlli iniziali:

Verifica dell'esistenza del file: {os.path.exists(file_fastq)}

Lettura e validazione dei dati FASTQ

Input principali:

- File FASTQ con sequenze di DNA → **file_fastq**
- Lunghezza del k-mer → **k = 6**
- Soglia di frequenza minima → **frequencyThreshold=0.002(0.2%)**

```
file_fastq = 'data/input.fastq'
print(f"Controllo se il file esiste: {os.path.exists(file_fastq)}")
k = 6
frequencyThreshold=0.002
```

```
Controllo se il file esiste: True
```

Strategie Computazionali e Implementazione

Lettura e parsing dei dati

parse_fastq()

Questa funzione gestisce il **parsing** del file FASTQ:

- verifica l'*esistenza* del file
- estrae *sequenze*, punteggi di *qualità phred* e *id* delle **reads**
- calcola la *lunghezza uniforme*
- restituisce tutti i *dati strutturati* per le **analisi successive**

```python
def parse_fastq(file_fastq):

    if not os.path.exists(file_fastq):
        raise FileNotFoundError(f"File '{file_fastq}' non trovato.")

    reads = list(SeqIO.parse(file_fastq, "fastq"))
    sequences = [str(r.seq) for r in reads]
    quality_scores = [r.letter_annotations["phred_quality"] for r in reads]
    ids = [r.id for r in reads]
    reads_len = len(sequences[0]) if sequences else 0

    len_diverse = any(len(seq) != reads_len for seq in sequences)
    if len_diverse:
        print("Attenzione: reads con lunghezze diverse!")

    print(f"Sono stati letti {len(sequences)} reads, ciascuno di {reads_len} basi")
    return reads, sequences, quality_scores, ids, reads_len

reads, sequences, quality_scores, ids, reads_len = parse_fastq(file_fastq)
```

```
Sono stati letti 5000 reads, ciascuno di 152 basi
```

```python
print("Esempio di read:")
print(reads[0])

print("\nEsempio di sequenza:")
print(sequences[0])

print("\nPunteggi di qualità della prima read:")
print(quality_scores[0])

print("\nEsempio di id della prima read:")
print(ids[0])
```

```
Esempio di read:
ID: SRR18961685.1
Name: SRR18961685.1
Description: SRR18961685.1 1 length=152
Number of features: 0
Per letter annotation for: phred_quality
Seq('TTAGGGTTAGGGTTAGGGTTAGGGTTAGGGTTAGGGTTAGGTTTAGGGTTAGGGTTACGGATAGGGTAACGTGTAGCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCATAACCCTAACCC...CAA')

Esempio di sequenza:
TTAGGGTTAGGGTTAGGGTTAGGGTTAGGGTTAGGGTTAGGTTTAGGGTTAGGGTTACGGATAGGGTAACGTGTAGCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCATAACCCTAACCC
AA

Punteggi di qualità della prima read:
[31, 32, 30, 32, 34, 35, 33, 36, 34, 36, 36, 36, 34, 36, 34, 36, 36, 36, 34, 36, 34, 36, 36, 29, 31, 31, 34, 36, 36, 36, 34, 36, 34, 30, 36, 36, 34, 3
0, 34, 36, 36, 14, 14, 36, 34, 15, 36, 36, 14, 14, 25, 15, 36, 13, 14, 14, 34, 12, 31, 28, 10, 35, 25, 15, 13, 13, 34, 8, 35, 12, 30, 14, 14, 14, 25,
16, 31, 31, 31, 29, 32, 33, 34, 34, 35, 33, 34, 35, 35, 35, 36, 34, 35, 35, 35, 35, 35, 34, 35, 35, 35, 35, 36, 34, 35, 35, 35, 35, 36, 34, 35, 35, 3
5, 35, 36, 34, 35, 27, 35, 11, 35, 34, 35, 35, 10, 35, 36, 34, 11, 26, 35, 11, 21, 34, 35, 27, 11, 9, 13, 34, 11, 12, 25, 11, 33, 34, 11, 12, 31, 12,
11, 31]

Esempio di id della prima read:
SRR18961685.1
```

Conteggio dei k-mer

count_kmers()

- **Algoritmo sliding window** implementato in **count_kmers()**
- Scansione di ogni read in tutte le *posizioni possibili*
- Memorizzazione in *dizionario*: **{k-mer: [conteggi_per_posizione]}**

```python
def count_kmers(sequences, k, reads_len):

    kmer_counts = {}

    for index, seq in enumerate(sequences):
        for pos in range(reads_len - k + 1):
            kmer = seq[pos : pos + k]

            if kmer not in kmer_counts:
                kmer_counts[kmer] = [0] * reads_len

            kmer_counts[kmer][pos] += 1
    return kmer_counts

kmer_dict = count_kmers(sequences, k, reads_len)

print("\nDistribuzione di due k-mer campione")
keys = list(kmer_dict.keys())
kmer_1 = keys[0]
print(f"\nK-mer 1: '{kmer_1}'")
print(f"Frequenze per posizione: {kmer_dict[kmer_1][:20]}...")

kmer_2 = keys[1]
print(f"\n K-mer 2: '{kmer_2}'")
print(f"Frequenze per posizione: {kmer_dict[kmer_2][:20]}...")

print(f"\nRiepilogo statistico:")
print(f"K-mer unici identificati: {len(kmer_dict)}")
print(f"Lunghezza distribuzioni posizionali: {len(kmer_dict[kmer_1])} posizioni")
```

```
Distribuzione di due k-mer campione

K-mer 1: 'TTAGGG'
Frequenze per posizione: [1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0]...

 K-mer 2: 'TAGGGT'
Frequenze per posizione: [2, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1]...

Riepilogo statistico:
K-mer unici identificati: 3876
Lunghezza distribuzioni posizionali: 152 posizioni
```

Filtraggio per frequenza

filter_kmer_report()

- Mantenimento solo dei k-mer con frequency ≥ *frequencyThreshold* ***(0.2%)***
- Calcola **frequency**: *occorrenze totali/posizioni analizzabili*
- Produce *dizionario filtrato* (**filtered_dict**) e *tabella riassuntiva(***table_filtered**) con **pandas**
- Produce *dizionario filtrato con i top 5 k-mer dominanti(***top5_table**)

```python
def filter_kmer_report(kmer_dict, frequencyThreshold, total_pos):

    filtered_dict = {}
    list_rows = []
    top_kmer_list = []

    for kmer, pos_counts in kmer_dict.items():
        total_occurrences = sum(pos_counts)
        frequency = total_occurrences / total_pos
        max_count = max(pos_counts) if pos_counts else 0
        best_pos = pos_counts.index(max_count) if pos_counts else 0
        included = frequency >= frequencyThreshold

        if included:
            filtered_dict[kmer] = pos_counts
            top_kmer_list.append((kmer, total_occurrences, best_pos, max_count))

        list_rows.append({
            "k-mer": kmer,
            "Occorrenze totali": total_occurrences,
            "Posizione max" : best_pos,
            "Occorrenze max": max_count,
            "Frequenza": round(frequency, 5),
            "Incluso": "Si" if included else "No"
        })

    table_filtered = pd.DataFrame(list_rows)
    table_filtered.sort_values(by="Frequenza", ascending=False, inplace=True)
    table_filtered.reset_index(drop=True, inplace=True)

    print("Tabella completa k-mer (con stato filtro):")
    display(table_filtered)


    if top_kmer_list:
        df_top = pd.DataFrame(top_kmer_list, columns=["k-mer", "Totale occorrenze", "Posizione max", "Occorrenze max"])
        df_top_sorted = df_top.sort_values(by="Occorrenze max", ascending=False).head(5)
        df_top_sorted.reset_index(drop=True, inplace=True)

        print("\nTop 5 k-mer filtrati (per occorrenze massime):")
        display(df_top_sorted)
    else:
        print("\nNessun k-mer ha superato il filtro di frequenza.")
        df_top_sorted = pd.DataFrame()

    return filtered_dict, list_rows, table_filtered, df_top_sorted


total_pos = (reads_len - k + 1) * len(sequences)

filtered_dict, list_rows, table_filtered, top5_table = filter_kmer_report(kmer_dict,frequencyThreshold, total_pos)

# Statistiche finali
print(f"\nStatistiche filtraggio:")
print(f"- K-mer totali analizzati: {len(kmer_dict)}")
print(f"- K-mer dopo filtraggio: {len(filtered_dict)}")
print(f"- Percentuale mantenuta: {len(filtered_dict)/len(kmer_dict)*100:.1f}%")
```

Tabella completa k-mer (con stato filtro):

	k-mer	Occorrenze totali	Posizione max	Occorrenze max	Frequenza	Incluso
0	TTTCCC	2106	66	32	0.00287	Si
1	CTGCCT	2041	84	40	0.00278	Si
2	CTCACT	2019	123	45	0.00275	Si
3	GGGCAG	1936	133	47	0.00263	Si
4	CTCTGG	1926	113	40	0.00262	Si
...	...	...	...	...	...	...
3871	TGCTAT	1	72	1	0.00000	No
3872	GTATAA	1	74	1	0.00000	No
3873	CCTCTA	2	71	1	0.00000	No
3874	ACCGTA	2	31	1	0.00000	No
3875	ATATAG	1	74	1	0.00000	No

3876 rows × 6 columns

Top 5 k-mer filtrati (per occorrenze massime):

	k-mer	Totale occorrenze	Posizione max	Occorrenze max
0	ACTTCT	1925	119	61
1	GCAGAG	1843	135	47
2	CAGAGG	1726	136	47
3	GGGCAG	1936	133	47
4	GAGCCT	1563	142	46

Statistiche filtraggio:
- K-mer totali analizzati: 3876
- K-mer dopo filtraggio: 32
- Percentuale mantenuta: 0.8%

Identificazione del k-mer dominante

find_max_kmer()

- Individuazione del *k-mer più frequente*(**max_count**) e della *sua posizione*(**max_pos**)
- *Confronto dei conteggi* per trovare il **massimo assoluto(max_kmer)**
- *Visualizzazione* del risultato in *tabella*(**table_max_kmer**) per un'immediata lettura

```
def find_max_kmer(filtered_dict):
    max_kmer, max_pos, max_count = None, None, 0

    for kmer, pos_counts in filtered_dict.items():
        for pos, count in enumerate(pos_counts):
            if count > max_count:
                max_kmer, max_pos, max_count = kmer, pos, count

    if max_kmer is None:
        print("Nessun k-mer dominante trovato.")
    return max_kmer, max_pos, max_count

max_kmer, max_pos, max_count = find_max_kmer(filtered_dict)

table_max_kmer = pd.DataFrame({
    "k-mer dominante": [max_kmer],
    "Posizione max": [max_pos],
    "Occorrenze max": [max_count]
})

display(table_max_kmer)
```

	k-mer dominante	Posizione max	Occorrenze max
0	ACTTCT	119	61

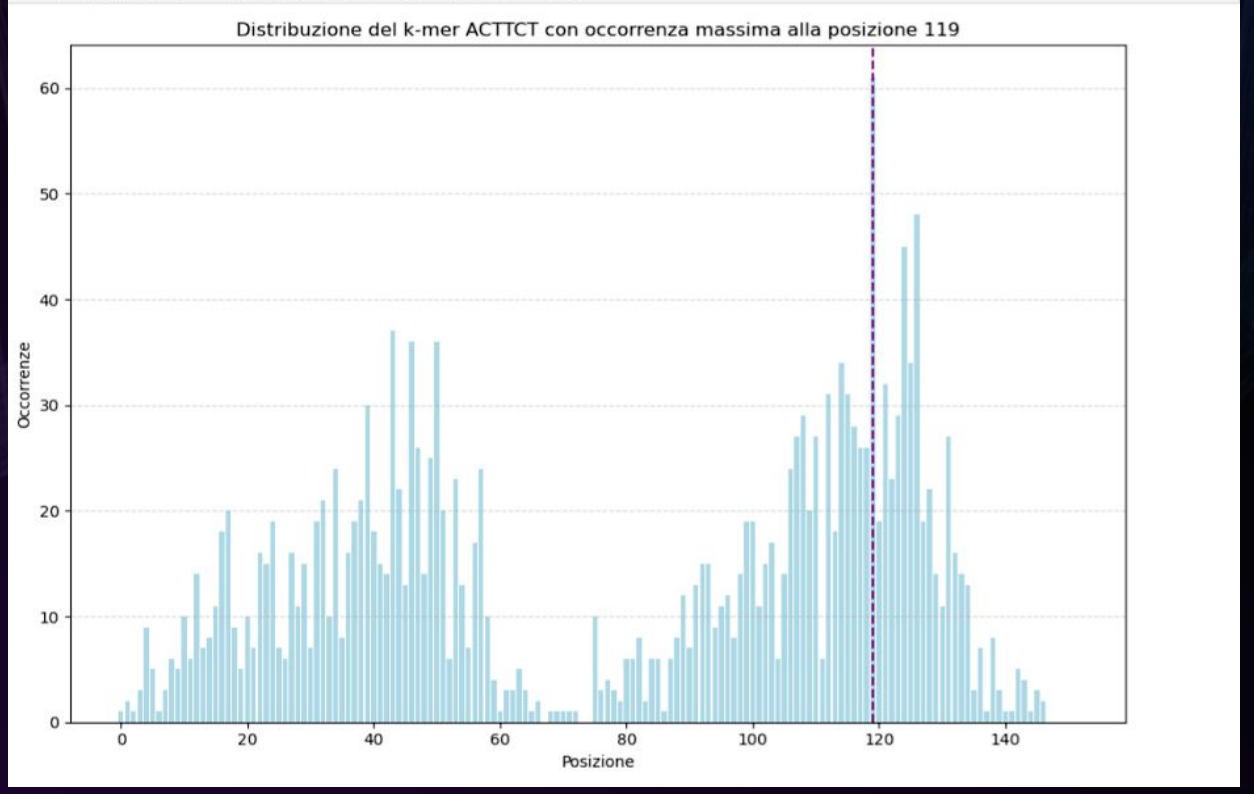
Visualizzazione Finale

plot_kmer_distribution()

- **Diagramma a barre (bar plot)** che mostra la frequenza di un k-mer in tutte le posizioni possibili lungo le **reads**
- Una *linea verticale* evidenzia la posizione con il **massimo numero di occorrenze**, rivelando eventuali pattern di localizzazione preferenziale

```python
def plot_kmer_distribution(kmer, pos_counts, max_pos):
    plt.figure(figsize=(10, 7))
    plt.bar(range(len(pos_counts)), pos_counts, color='lightblue')
    plt.axvline(x=max_pos, color='purple', linestyle='--')
    plt.xlabel("Posizione")
    plt.ylabel("Occorrenze")
    plt.grid(axis="y", linestyle="--", alpha=0.5)
    plt.title(f"Distribuzione del k-mer {kmer} con occorrenza massima alla posizione {max_pos}")
    plt.tight_layout()
    plt.show()

if not filtered_dict:
    print("Nessun k-mer supera la soglia di frequenza da analizzare. Cambiare i parametri di configurazione.")
else:
    plot_kmer_distribution(max_kmer, filtered_dict[max_kmer], max_pos)
```



File Output FASTA Finale

output_fasta()

- Seleziona **reads** contenenti il *k-mer dominante*
- Calcola la **qualità media** di ciascuna *read selezionata*
- Scrive le *reads filtrate* in un **nuovo file FASTA** con la *qualità media nell'ID*

```
def output_fasta(reads, quality_scores, max_kmer,max_pos, k, output_fasta):

    output = []

    for i, r in enumerate(reads):
        sequence = str(r.seq)
        q = quality_scores[i]

        if sequence[max_pos:max_pos + k] == max_kmer:
            average = round(stats.mean(q), 2)
            newRecord = SeqRecord(Seq(sequence), id=f"{r.id} {average}", description="")
            output.append(newRecord)

    write(output, output_fasta, "fasta")
    print(f"Sono stati salvati {len(output)} reads in '{output_fasta}'.")

if max_kmer is not None and max_pos is not None:
    output_fasta(reads, quality_scores, max_kmer, max_pos, k, "output.fasta")
else:
    print("Impossibile generare output: k-mer dominante non identificato.")
```

```
Sono stati salvati 61 reads in 'output.fasta'.
```

Scelte Progettuali

- **Parsing FASTQ** -Estrazione sequenze e punteggi Phred
- **Estrazione k-mer** -Finestra scorrevole per il conteggio completo e distribuzione posizionale
- **Filtro k-mer** -Rimozione delle sottosequenze troppo rare
- **K-mer dominante** -Identificato tramite picco massimo nella distribuzione posizionale
- **Visualizzazione** -Diagramma a barre della distribuzione del k-mer dominante
- **Esportazione FASTA** -Sequenze contenenti il k-mer dominante nella posizione di picco, con qualità media in header

Librerie Utilizzate

- **import pandas as pd:** *per gestione dati tabellari (**table_filtered = pd.DataFrame(list_rows)**; **df_top_sorted = df_top.sort_values(by="Occorrenze max", ascending=False).head(5)**; **table_max_kmer = pd.DataFrame ({ "k-mer dominante": [max_kmer],…})**)*
- **import os as os:** *esistenza file fastq prima del parsing (**os.path.exists(file_fastq)**)*
- **import matplotlib.pyplot as plt:** *per visualizzazione (diagramma a barre)*
- **from Bio import SeqIO:** *lettura/scrittura formati biologici (**reads = list(SeqIO.parse(file_fastq, "fastq"))**)*
- **from Bio.Seq import Seq** */* **from Bio.SeqRecord import SeqRecord:** *record sequenza con metadata e rappresentazione sequenze biologiche (**newRecord = SeqRecord(Seq(sequence), id=f"{r.id} {average}", description="")**)*
- **from Bio.SeqIO import write:** *scrittura file di sequenze (**write(output, output_fasta, "fasta")**)*
- **import statistics as stats:** *per calcolo qualità media (**average = round(stats.mean(q), 2)**)*

File FASTA finale (output.fasta)

Il progetto produce come risultato un **file FASTA** contenente tutte le sequenze del dataset che includono il **k-mer dominante** nella posizione di **massima frequenza**.

Ciascuna entry del file presenta:

- L'***ID*** originale del read
- La ***qualità media*** della sequenza, calcolata come media dei punteggi phred
- La ***sequenza nucleotidica*** composta dalle basi adenina (A), timina (T), citosina (C) e guanina (G)

```
(bioinfo) anna03@Laptop-VVA0O0NI-ANNA:~/bioinfo_lab$ cat output.fasta
>SRR18961685.193 34.82
TGATCTTCCACCTGCTCTCCCAGGGCCAAAGCTAGACCTGCTGAGCCCCTCCCTCCAGCC
GGCTGGTCTGAGCAGTAGGCATAGGGGAAAGATTGGAGGAAAGATGAGTGACAGCATCAA
CTTCTCTCACAACCTAGGCCAGTAAGTAGTGC
>SRR18961685.4577 34.91
CATAAAATAGTGGTTTTCAAACTGTAGAGCTCTGGACTTCTCACTTCTAGGGCAGAGGGA
GCCTGAACAAGTGAGGGAGGCCGGACCTTTGGAGACTGTGTGGGGGGGCCTGGGCACTGA
CTTCTGCAACCACCTGAGCGCGGGCATCCTGT
>SRR18961685.4580 34.72
TGAAAAATGTGTTGCTGTAGTTTGTTATTAGACCCCTTCTTTCCATTGGTTTAATTAGGA
ACGGGGAACCCAGAGCTCATAAAATAGTGGTTTTCAAACTGTAGAGCTCTGGACTTCTCA
CTTCTAGGGCAGAGGGAGCCTGAACAAGTGAG
>SRR18961685.4583 34.91
TTAGACCCCTTCTTTCCATTGGTTTAATTAGGAATGGGGAACCCAGAGCCTCACTTGTTC
AGGCTCCCTCTGCCCTTCATAAAATAGTGGTTTTCAAACTGTAGAGCTCTGGACTTCTCA
CTTCTAGGGCAGAGGGAGCCTGAACAAGTGAG
>SRR18961685.4584 34.61
TGCTGTAGTTTGTTATTAGACCCCTTCTTTCCATTGGTTTAATTAGGAACGGGGAACCCA
GAGCCTCACTTGTTCATCATAAAATAGTGGTTTTCAAACTGTAGAGCTCTGGACTTCTCA
CTTCTAGGGCAGAGGGAGCCTGAACAAGTGAG
>SRR18961685.4585 34.85
TTTCCATTGGTTTAATTAGGAATGGGGAACCCAGAGCCTCACTTGTTCAGGCTCCCTCTG
CCCTAGAAGTGAGAAGTCATAAAATAGTGGTTTTCAAACTGTAGAGCTCTGGACTTCTCA
CTTCTAGGGCAGAGGGAGCCTGAACAAGTGAG
>SRR18961685.4590 34.84
CTTCTTTCCATTGGTTTAATTAGGAACGGGGAACCCAGAGCCTCACTTGTTCAGGCTCCC
TCTGCCCTAGAAGTGATCATAAAATAGTGGTTTTCAAACTGTAGAGCTCTGGACTTCTCA
CTTCTAGGGCAGAGGGAGCCTGAACAAGTGAG
>SRR18961685.4591 34.91
CTTTCCATTGGTTTAATTAGGAATGGGGAACCCAGAGCCTCACTTGTTCAGGCTCCCTCT
GCCCTAGAAGTGAGAATCATAAAATAGTGGTTTTCAAACTGTAGAGCTCTGGACTTCTCA
CTTCTAGGGCAGAGGGAGCCTGAACAAGTGAG
>SRR18961685.4595 34.97
GTTTTGTGCCACTTCTGGATGCTAGGGTTACACTGGGAGATACAGCAGTGAAGCTGAAAT
GAAAAATGTGTTGCTGTCATAAAATAGTGGTTTTCAAACTGTAGAGCTCTGGACTTCTCA
CTTCTAGGGCAGAGGGAGCCTGAACAAGTGAG
```

Grazie

Anna Giulia Granziera

Mat.895402

a.granziera@campus.unimib.it