

# Progetto Bioinformatica

## Davide Cantù - 869158

Tema 4

# Lettura dei parametri da console

```
if __name__ == "__main__":  
    fastq_file = sys.argv[1]  
    k = int(sys.argv[2])  
    f = float(sys.argv[3])  
    output_fasta = sys.argv[4]  
    if not os.path.exists(fastq_file):  
        sys.exit(f"Error: The file {fastq_file} does not exist.")  
    if not fastq_file.endswith(".fastq"):  
        sys.exit("Error: The input file must be a FASTQ file (with .fastq extension).")  
    if k < 1:  
        sys.exit("Error: The length of k-mer must be a positive integer.")  
    if f < 0 or f > 1:  
        sys.exit("Error: The frequency threshold must be between 0 and 1.")  
    if not output_fasta.endswith(".fasta"):  
        sys.exit("Error: The output file must be a FASTA file (with .fasta extension).")  
    main(fastq_file, k, f, output_fasta)
```

# Lettura dei parametri da console

```
fastq_file = sys.argv[1]
k = int(sys.argv[2])
f = float(sys.argv[3])
output_fasta = sys.argv[4]
```

Fī cōvā<sup>0</sup> cū<sup>0</sup> e<sup>0</sup> sgv̄ euf v8

- ĚĀ ū Āēzā/î sŕy°ā
- FēĀ ūbrî MĀĀ ū Āvgvzāysŕrî
- FēĀ ūĀĀ sŕy°î ūM
- ĚĀ ū Āî zāŕēM ū Āî yĀ°ā°ā

# Lettura dei parametri da console

```
if __name__ == "__main__":
    fastq_file = sys.argv[1]
    k = int(sys.argv[2])
    f = float(sys.argv[3])
    output_fasta = sys.argv[4]
    if not os.path.exists(fastq_file):
        sys.exit(f"Error: The file {fastq_file} does not exist.")
    if not fastq_file.endswith(".fastq"):
        sys.exit("Error: The input file must be a FASTQ file (with .fastq extension).")
    if k < 1:
        sys.exit("Error: The length of k-mer must be a positive integer.")
    if f < 0 or f > 1:
        sys.exit("Error: The frequency threshold must be between 0 and 1.")
    if not output_fasta.endswith(".fasta"):
        sys.exit("Error: The output file must be a FASTA file (with .fasta extension).")
    main(fastq_file, k, f, output_fasta)
```

# Lettura dei parametri da console

```
if not os.path.exists(fastq_file):
    sys.exit(f"Error: The file {fastq_file} does not exist.")
if not fastq_file.endswith(".fastq"):
    sys.exit("Error: The input file must be a FASTQ file (with .fastq extension).")
if k < 1:
    sys.exit("Error: The length of k-mer must be a positive integer.")
if f < 0 or f > 1:
    sys.exit("Error: The frequency threshold must be between 0 and 1.")
if not output_fasta.endswith(".fasta"):
    sys.exit("Error: The output file must be a FASTA file (with .fasta extension).")
```

[illegible]

# Lettura dei parametri da console

```
if __name__ == "__main__":
    fastq_file = sys.argv[1]
    k = int(sys.argv[2])
    f = float(sys.argv[3])
    output_fasta = sys.argv[4]
    if not os.path.exists(fastq_file):
        sys.exit(f"Error: The file {fastq_file} does not exist.")
    if not fastq_file.endswith(".fastq"):
        sys.exit("Error: The input file must be a FASTQ file (with .fastq extension).")
    if k < 1:
        sys.exit("Error: The length of k-mer must be a positive integer.")
    if f < 0 or f > 1:
        sys.exit("Error: The frequency threshold must be between 0 and 1.")
    if not output_fasta.endswith(".fasta"):
        sys.exit("Error: The output file must be a FASTA file (with .fasta extension).")
    main(fastq_file, k, f, output_fasta)
```

# Lettura dei parametri da console

```
main(fastq_file, k, f, output_fasta)
```

Fèùgš Ā °űy°ì ĀĀ ēűgvűĀĀ ēűűì őűĀ ğ

# main

```
def main(fastq_file, k, f, output_fasta):
    sequences, qualities, identifiers = parse_fastq(fastq_file)
    if k > len(sequences[0]):
        sys.exit("Error: The length of k-mer is greater than the length of the reads.")
    kmer_counts = count_kmers(sequences, k)
    filtered_kmers = filter_kmers(kmer_counts, len(sequences), f)
    print_kmer_report(filtered_kmers)
    max_kmer, max_pos, max_count = find_max_kmer(filtered_kmers)
    if max_kmer is None:
        sys.exit("No k-mers found with the given frequency threshold.")
    print(f"Most frequent k-mer: {max_kmer} at position {max_pos} with {max_count} occurrences")
    extract_reads_with_kmer(sequences, qualities, identifiers, max_kmer, max_pos, output_fasta)
    plot_kmer_distribution(max_kmer, kmer_counts)
```



# main – parsing del file fastq

```
def main(fastq_file, k, f, output_fasta):
    sequences, qualities, identifiers = parse_fastq(fastq_file)
    if k > len(sequences[0]):
        sys.exit("Error: The length of k-mer is greater than the length of the reads.")
    kmer_counts = count_kmers(sequences, k)
    filtered_kmers = filter_kmers(kmer_counts, len(sequences), f)
    print_kmer_report(filtered_kmers)
    max_kmer, max_pos, max_count = find_max_kmer(filtered_kmers)
    if max_kmer is None:
        sys.exit("No k-mers found with the given frequency threshold.")
    print(f"Most frequent k-mer: {max_kmer} at position {max_pos} with {max_count} occurrences")
    extract_reads_with_kmer(sequences, qualities, identifiers, max_kmer, max_pos, output_fasta)
    plot_kmer_distribution(max_kmer, kmer_counts)
```

# parse\_fastq

```
def parse_fastq(file):
    sequences = []
    qualities = []
    identifiers = []
    for record in SeqIO.parse(file, "fastq"):
        sequences.append(str(record.seq))
        qualities.append(record.letter_annotations["phred_quality"])
        identifiers.append(record.id)
    return sequences, qualities, identifiers
```

Jì yǎ lǐ gōng yì ~~ǎ~~ǎ è yǎ zǐ bō i i ~~ǎ~~ǎ ù ~~ǎ~~ǎ zǎ yǎ r v ǎ i gǎ zǎ ~~ǎ~~ǎ ā u mē yǎ ~~ǎ~~ǎ gǎ yǎ yǎ ~~ǎ~~ǎ i yǎ ě gǎ i ~~ǎ~~ǎ yǎ yǎ ù i ù ~~ǎ~~ǎ ~~ǎ~~ǎ i sǎ i ~~ǎ~~ǎ i ~~ǎ~~ǎ i zǎ vǎ i ~~ǎ~~ǎ sǎ ù ~~ǎ~~ǎ sǎ i zǎ ~~ǎ~~ǎ yǎ v 9  
Q ù bǎ vǎ ~~ǎ~~ǎ ā ù ~~ǎ~~ǎ gǎ ě ù i ù ~~ǎ~~ǎ gǎ vǎ zǎ gǎ ~~ǎ~~ǎ i yǎ i ù mē yǎ ~~ǎ~~ǎ ě u ~~ǎ~~ǎ ~~ǎ~~ǎ i ù ~~ǎ~~ǎ gǎ vǎ yǎ gǎ i ~~ǎ~~ǎ vǎ ~~ǎ~~ǎ ě ~~ǎ~~ǎ ~~ǎ~~ǎ i ù ~~ǎ~~ǎ zǎ 9

ř g y z 899 ž s y l ě ř v ú 9 y p ě š ů s ů i y ě

# main – parsing del file fastq

```
def main(fastq_file, k, f, output_fasta):
    sequences, qualities, identifiers = parse_fastq(fastq_file)
    if k > len(sequences[0]):
        sys.exit("Error: The length of k-mer is greater than the length of the reads.")
    kmer_counts = count_kmers(sequences, k)
    filtered_kmers = filter_kmers(kmer_counts, len(sequences), f)
    print_kmer_report(filtered_kmers)
    max_kmer, max_pos, max_count = find_max_kmer(filtered_kmers)
    if max_kmer is None:
        sys.exit("No k-mers found with the given frequency threshold.")
    print(f"Most frequent k-mer: {max_kmer} at position {max_pos} with {max_count} occurrences")
    extract_reads_with_kmer(sequences, qualities, identifiers, max_kmer, max_pos, output_fasta)
    plot_kmer_distribution(max_kmer, kmer_counts)
```

# main – parsing del file fastq

```
if k > len(sequences[0]):
    sys.exit("Error: The length of k-mer is greater than the length of the reads.")
```

Njueŋvɛ̃ŋlì g°eãṽŋŋỹzsr̃b̃î ì ɔ̃ ù Ǻ̃z̃ỹèì ɔ̃f ṽŋ°eũṽŋz̃sũṽŋp̃ũr̃ì Ǻ̃ Ǻ̃i y°ì ɔ̃m̃ŋũṽŋṽṽŋŋr̃ì Ǻ̃2â ɔ̃ũ Ǻ̃ṽũǺ̃ẽŋi ỹg̃eũf ṽŋ s̃ŋi ɔ̃w̃i ỹi Ǻ̃ũẽǺ̃z̃vgṽz̃ỗs̃ũb̃ẽǺ̃s̃ŋỹeũf ì Ǻ̃s̃y°ì z̃â2ũũr̃ì 9

# main – conta numero apparizioni k-mers

```
def main(fastq_file, k, f, output_fasta):
    sequences, qualities, identifiers = parse_fastq(fastq_file)
    if k > len(sequences[0]):
        sys.exit("Error: The length of k-mer is greater than the length of the reads.")
    kmer_counts = count_kmers(sequences, k)
    filtered_kmers = filter_kmers(kmer_counts, len(sequences), f)
    print_kmer_report(filtered_kmers)
    max_kmer, max_pos, max_count = find_max_kmer(filtered_kmers)
    if max_kmer is None:
        sys.exit("No k-mers found with the given frequency threshold.")
    print(f"Most frequent k-mer: {max_kmer} at position {max_pos} with {max_count} occurrences")
    extract_reads_with_kmer(sequences, qualities, identifiers, max_kmer, max_pos, output_fasta)
    plot_kmer_distribution(max_kmer, kmer_counts)
```

# count\_kmers

```
def count_kmers(sequences, k):  
    kmer_counts = defaultdict(lambda: defaultdict(int))  
    for seq in sequences:  
        for pos in range(len(seq) - k + 1):  
            kmer = seq[pos : pos + k]  
            kmer_counts[kmer][pos] += 1  
    return kmer_counts
```

FěÄ° ůM ůi Ägv° ůc ůü ï ýz yì yù ï g ï Ä sg ůc èy ï ÄÄ° ů ï yv Ä ščyy èy s ů ůš Ä ů ůg ï yv Ä ů ů ï yŇz è Ä ů Äyy yv ggs / Ä ů š ů Ä ů ů v È Ä ï ý Äy v z è y z s Ä° Ä ů ů Ä  
ži y° ï ů 149  
ü ï y z g v° ů ů z ï Ä ů Ä s ů ů è y s / Ä š Ä s ů ů è y s Ä ů ů ů ů ů ů Ä ů g Ä Ä ů yv Ä y° ï ů Ä ů Ä z ä yv Ä è Ä ů ů Ä ů ů v z è y z s Ä° Ä ů ů Ä  
ů ů Ä° Ä ů ů è y ï Ä ÄÄ° ů ï yv Ä ščv ů Ä ů ů è y z v Ä ů ů ï ů Ä y v z s ů ů ů 9

# count\_kmers

```
def count_kmers(sequences, k):
    kmer_counts = defaultdict(lambda: defaultdict(int))
    for seq in sequences:
        for pos in range(len(seq) - k + 1):
            kmer = seq[pos : pos + k]
            kmer_counts[kmer][pos] += 1
    return kmer_counts
```

CZi ù' y.ś/8

[illegible]

1. NĀĀ1 8Ů 8Ā 16ŮĀĀŮĒyyeyĭ sĭĀvzšM/ũt / ĀiĀĀi yŋi ũM  
1. ĀĀĀ1 8Ů 8Ā 8# 16ŮĀĀŮĒyyeyĭ sĭĀvzšM/ũt | ĀiĀĀi yŋi ũM i ĀiĀvzšM/ũt - ĀiĀĀi yŋi ũM  
1. ĀĀĀ1 8Ů 8Ā 16ŮĀĀŮĒyyeyĭ sĭĀvzšM/ũt - ĀiĀĀi yŋi ũM  
1. ĀĀĀ1 8Ů 8# 16ŮĀĀŮĒyyeyĭ #ĀvĕĀvĕĀiĀvzšM/ũt -  
1. NĀD1 8Ů 8# 16ŮĀDŮĒyyeyĭ #ĀvĕĀvĕĀiĀvzšM/ũt /  
1. ĀDĀ1 8Ů 8# 16ŮĀDŮĒyyeyĭ #ĀvĕĀvĕĀiĀvzšM/ũt |  
1. DĀĀ1 8Ů 8# 16ŮĀĀŮĒyyeyĭ #ĀvĕĀvĕĀiĀvzšM/ũt -

# main – filtraggio k-mers

```
def main(fastq_file, k, f, output_fasta):
    sequences, qualities, identifiers = parse_fastq(fastq_file)
    if k > len(sequences[0]):
        sys.exit("Error: The length of k-mer is greater than the length of the reads.")
    kmer_counts = count_kmers(sequences, k)
    filtered_kmers = filter_kmers(kmer_counts, len(sequences), f)
    print_kmer_report(filtered_kmers)
    max_kmer, max_pos, max_count = find_max_kmer(filtered_kmers)
    if max_kmer is None:
        sys.exit("No k-mers found with the given frequency threshold.")
    print(f"Most frequent k-mer: {max_kmer} at position {max_pos} with {max_count} occurrences")
    extract_reads_with_kmer(sequences, qualities, identifiers, max_kmer, max_pos, output_fasta)
    plot_kmer_distribution(max_kmer, kmer_counts)
```



# filter\_kmers

```
def filter_kmers(kmer_counts, total_reads, f):
    threshold = f * total_reads
    filtered_kmers = defaultdict(int)
    for k, v in kmer_counts.items():
        if sum(v.values()) ≥ threshold:
            filtered_kmers[k] = v
    return filtered_kmers
```

FeĀ° ūM ūi Ā ūi yzūi ĭ yz ĭ ūyeĀŪi ĭ yz ġr ĭ Āyy.ēs ūv Āēu ĭ ūv Āēu Āēv ū Ā° ēu Āēv ū ēĀ ĭ ġzēĀ ēu ēĀ ūā ūā 8ĭ Āēu Ā9 Āēv ēēy ĭ ē z Ġ° ūi ĭ yv Ā ĭ sĭ y° ĭ ūM Ā ġ Āv ūv Ā° ? 6ēu yē Ā yēu ūv Āy ĭ zēĀ ġv ūz ĭ yēM ūi Āv ūv ĀŪi ĭ yz ġr ĭ Āv ūv Āyy.ēs ūv Āēu ĭ ūv Ā \ ? Āēu Āēv ēēy ĭ ē z 9

# main – stampa report testuale

```
def main(fastq_file, k, f, output_fasta):
    sequences, qualities, identifiers = parse_fastq(fastq_file)
    if k > len(sequences[0]):
        sys.exit("Error: The length of k-mer is greater than the length of the reads.")
    kmer_counts = count_kmers(sequences, k)
    filtered_kmers = filter_kmers(kmer_counts, len(sequences), f)
    print_kmer_report(filtered_kmers)
    max_kmer, max_pos, max_count = find_max_kmer(filtered_kmers)
    if max_kmer is None:
        sys.exit("No k-mers found with the given frequency threshold.")
    print(f"Most frequent k-mer: {max_kmer} at position {max_pos} with {max_count} occurrences")
    extract_reads_with_kmer(sequences, qualities, identifiers, max_kmer, max_pos, output_fasta)
    plot_kmer_distribution(max_kmer, kmer_counts)
```

# print\_kmer\_report

```
def print_kmer_report(filtered_kmers):
    with open("report.dat", "w") as f:
        count = 0
        f.write("K-mer report:\n\n")
        for kmer, pos_counts in filtered_kmers.items():
            positions_and_counts = [f"[{pos}, {count}]" for pos, count in sorted(pos_counts.items())]
            f.write(f"{kmer}: {' '.join(positions_and_counts)}\n")
            count += 1
        if count == 0:
            f.write("No k-mers found with the given frequency threshold.")
    print(f"K-mers report successfully written to report.txt")
```

[illegible]

CZi ù' y.ś/8

[illegible][illegible]

# main – trova k-mer con maggiore frequenza

```
def main(fastq_file, k, f, output_fasta):
    sequences, qualities, identifiers = parse_fastq(fastq_file)
    if k > len(sequences[0]):
        sys.exit("Error: The length of k-mer is greater than the length of the reads.")
    kmer_counts = count_kmers(sequences, k)
    filtered_kmers = filter_kmers(kmer_counts, len(sequences), f)
    print_kmer_report(filtered_kmers)
    max_kmer, max_pos, max_count = find_max_kmer(filtered_kmers)
    if max_kmer is None:
        sys.exit("No k-mers found with the given frequency threshold.")
    print(f"Most frequent k-mer: {max_kmer} at position {max_pos} with {max_count} occurrences")
    extract_reads_with_kmer(sequences, qualities, identifiers, max_kmer, max_pos, output_fasta)
    plot_kmer_distribution(max_kmer, kmer_counts)
```

# find\_max\_kmer

```
def find_max_kmer(filtered_kmers):
    max_kmer, max_pos, max_count = None, None, 0
    for kmer, pos_counts in filtered_kmers.items():
        for pos, count in pos_counts.items():
            if count > max_count:
                max_kmer, max_pos, max_count = kmer, pos, count
    return max_kmer, max_pos, max_count
```

FeA p u M u i A u i 2 u r e l 2 u u i y z e u e A u i y o e A v z s M u i A A u u i y A E v u g i g v u y e i A u i y A y y o i u d 9

# main – trova k-mer con maggiore frequenza

```
def main(fastq_file, k, f, output_fasta):
    sequences, qualities, identifiers = parse_fastq(fastq_file)
    if k > len(sequences[0]):
        sys.exit("Error: The length of k-mer is greater than the length of the reads.")
    kmer_counts = count_kmers(sequences, k)
    filtered_kmers = filter_kmers(kmer_counts, len(sequences), f)
    print_kmer_report(filtered_kmers)
    max_kmer, max_pos, max_count = find_max_kmer(filtered_kmers)
    if max_kmer is None:
        sys.exit("No k-mers found with the given frequency threshold.")
    print(f"Most frequent k-mer: {max_kmer} at position {max_pos} with {max_count} occurrences")
    extract_reads_with_kmer(sequences, qualities, identifiers, max_kmer, max_pos, output_fasta)
    plot_kmer_distribution(max_kmer, kmer_counts)
```

# main – trova k-mer con maggiore frequenza

```
if max_kmer is None:
    sys.exit("No k-mers found with the given frequency threshold.")
print(f"Most frequent k-mer: {max_kmer} at position {max_pos} with {max_count} occurrences")
```

Lì ÁeÁ° ùM ùí Á ùí zù éVzùùì y ùv ùÁy EeÁ ùÁr ézzù vÁ vyvÁE ÿ Á ùeÁ ùÁ Áì y°ì ùM Ásùù gèÁrì Áv ùÁv ùv ÁcèÁÁy EeÁ Áì ÿ gñì Áv ùv Á éy.éy zù ùÁy y°ì ùM Á°yì ysvì ÁEÁ°ì ùeÁs yv zècèÁ ÁÁùì éÁ sèv ù èuf vÁ°ì zà Áv ùv ùv Ás ùì Á lì g°éÁ Á°gñì zzzEùrì ùÁ ÁeùÁyì èM ùì Áì ùÁ yì yv yÁyì Áv ùÁ yÁv ù°ùy°ì Ágys ÿ ÁeùÁ ùv Áì Áy yv yÁgñì Áv ùÁv ùv ÁcèÁÁy EeÁ Áì ÿ gñì Ás yì gèùv Áì Áv ùì sÁ ùs Á ùs ùì ùÁcèù yéÁ°Á gñùz ùì ÁÁì ÿ y.ÁÁy y°ì ùÁ ÁEÁ° éÁv zzz ùì Á ÁÁ° ùì yv Á sÁ gñyì ùM 9

# main – creazione file fasta

```
def main(fastq_file, k, f, output_fasta):
    sequences, qualities, identifiers = parse_fastq(fastq_file)
    if k > len(sequences[0]):
        sys.exit("Error: The length of k-mer is greater than the length of the reads.")
    kmer_counts = count_kmers(sequences, k)
    filtered_kmers = filter_kmers(kmer_counts, len(sequences), f)
    print_kmer_report(filtered_kmers)
    max_kmer, max_pos, max_count = find_max_kmer(filtered_kmers)
    if max_kmer is None:
        sys.exit("No k-mers found with the given frequency threshold.")
    print(f"Most frequent k-mer: {max_kmer} at position {max_pos} with {max_count} occurrences")
    extract_reads_with_kmer(sequences, qualities, identifiers, max_kmer, max_pos, output_fasta)
    plot_kmer_distribution(max_kmer, kmer_counts)
```



# extract\_reads\_with\_kmer

```
def extract_reads_with_kmer(sequences, qualities, identifiers, kmer, position, output_fasta):
    count = 0
    with open(output_fasta, "w") as fasta_out:
        for i, seq in enumerate(sequences):
            if seq[position : position + len(kmer)] == kmer:
                avg_quality = sum(qualities[i]) / len(qualities[i])
                count += 1
                fasta_out.write(f">{identifiers[i]} avg_quality={avg_quality:.2f}\n{seq}\n")
    print(f"Exported {count} reads with k-mer {kmer} in position {position}.")
```

[illegible]

Avg°ùrì ùcēMv'úi ĀĀ°š'vĀęgvĀđi ys'í ũâ8  
řgyž899 ūŔ sūs'í î ś9'yc@ ē sūđĀLNĀ2kvũ êā

# main – visualizzazione diagramma k-mer

```
def main(fastq_file, k, f, output_fasta):
    sequences, qualities, identifiers = parse_fastq(fastq_file)
    if k > len(sequences[0]):
        sys.exit("Error: The length of k-mer is greater than the length of the reads.")
    kmer_counts = count_kmers(sequences, k)
    filtered_kmers = filter_kmers(kmer_counts, len(sequences), f)
    print_kmer_report(filtered_kmers)
    max_kmer, max_pos, max_count = find_max_kmer(filtered_kmers)
    if max_kmer is None:
        sys.exit("No k-mers found with the given frequency threshold.")
    print(f"Most frequent k-mer: {max_kmer} at position {max_pos} with {max_count} occurrences")
    extract_reads_with_kmer(sequences, qualities, identifiers, max_kmer, max_pos, output_fasta)
    plot_kmer_distribution(max_kmer, kmer_counts)
```

# extract\_reads\_with\_kmer

```
def plot_kmer_distribution(kmer, kmer_counts):  
    positions = sorted(kmer_counts[kmer].keys())  
    counts = [kmer_counts[kmer][pos] for pos in positions]  
    plt.bar(positions, counts, color="blue", edgecolor="black")  
    plt.xlabel("Position")  
    plt.ylabel("Occurrences")  
    plt.grid(axis="y", linestyle="--", alpha=0.5)  
    plt.title(f"Occurrences of k-mer '{kmer}' per position")  
    plt.show()
```

FeÅ° ùM/ùì Ąwâzùrì ý2î szôg°äüvÿyŵî °gì ĄŁyēl ġvĄġēyŷi Ąžēuf vĄĄġyŷi yēĄ ēōy wâgġi ĄřvzōēĄġy y°ì ùMĄ ĄĄvzšM/ùì Ąì ŰĄrì ý  
ġvŰĄ ēōōsŷi Ąy y°ì ùMĄi Ącūsŷi ġvŷi 9Ą° ŰĄžgšzēġšvŰĄ Ą ĄvzšM/ùì Ąġ° ŰĄy šřēēĄĄ° Űì ýĄ Ąġġvŷi ŰM9

Ąvġ° Űì ŰēM/ùì ĄĄ° šĄ vĄġvĄšĄ yšŰì ŰĄ8

[řgyž89Ą ēōy wâgġi yōġēġŰ](#)

# Esempio di utilizzo

```
py bioinfo.py input.fastq 5 0.1 output.fasta
```

[illegible]