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]
```

[illegible]

- [illegible]
- [illegible]
- [illegible]
- [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

```
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).")
```

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)
```

Fẻửĝśv Ă ᵒ ửyᵒ ì ĂśẠửẢử ẻśử ĝv ửĂśĂỵ ẻyẻửì ởyśẠì 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
```

[illegible]

[illegible]

[illegible]

[illegible]

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.")
```

[illegible]

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
```

[illegible]

[illegible]

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
```

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

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
```

[illegible]

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]

[illegible]

[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
```

FẻĂ̦º ửM̀ŝvửỉ Ặ̦ửî 2ừ ẻŽ2ừừ ì ỷźẻụEẻĂŝĂ̦ụ4ừ ì ỷ6Ăẻ̦Ăỵvźŝ Mŝvửỉ Ă ĂŝĂ̦ụº ừ ì ỷv Ă ŝÆvụ̂ Ăĝř ì Ăĝvừ ỵẻỳ ĂŝĂ̦ụ4ừ ì ỷỵŝĂĂ̦ỵì ỵº ì ửữ̂ 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")
```

[illegible]

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}.")
```

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()
```

[illegible]

[illegible]

[illegible]

Esempio di utilizzo

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

[illegible]