Obrada informacija

Laboratorijska vježba 2

U ovoj vježbi upoznat ćete se s jednom primjenom tehnika obrade informacija u bioinformatici. Ova laboratorijska vježba nosi 4 boda. Izvješće s ove laboratorijske vježbe potrebno je predati u .pdf formatu na *Moodle*. Izvješće koje predajete se mora zvati *PrezimeIme.pdf*.

Osim biblioteka za rad s Fourierovom transformacijom (koristit ćemo samo numpy) koristit ćemo i biblioteku biopython koja sadrži puno korisnih alata iz područja bioinformatike. Mi ćemo je koristiti za jednostavnije baratanje bioinformatičkim tipovima podataka.

Biblioteka biopython dolazi s instalacijom Anaconde, ali ju je potrebno uključiti u okolinu (*environment*) koja se koristi.

Ako vježbu izvodite u Google Colab okruženju, morate instalirati biblioteku biopython. Instalaciju je potrebno izvršiti u sklopu prvog zadatka ove laboratorijske vježbe. Instalaciju izvodite sljedećim kodom:

```
try:
  import google.colab
  !pip install biopython
except ImportError:
  pass
```

Nakon izvođenja ovog koda, možete učitati biopython biblioteku.

1. Zadatak

Python biblioteke potrebne za laboratorijske vježbe su numpy i biopython. Uključite ih ("importirajte") i ispišite verziju svake od njih pomoću [ime_biblioteke].__version__.

UPUTA: Osnovna biopython biblioteka ima naziv Bio.

```
import numpy as np
import Bio

print("NumPy verzija:", np.__version__)
print("BioPython verzija:", Bio.__version__)
```

```
NumPy verzija: 2.0.2
BioPython verzija: 1.84
```

1. Zadatak

Uz laboratorijske vježbe dobili ste dvije datoteke s podacima. Datoteku koja sadrži referentni genom jednog soja bakterije Escherichia coli (escherichia_coli_reference.fasta) u FASTA formatu

i datoteku koja sadrži skup očitanja dobivenih sekvenciranjem (ecoli_ILL_small.fastq) u FASTQ formatu.

Datoteke možete učitati koristeći metodu *parse()* iz biblioteke Bio.SeqIO. Metoda *parse()* vraća iterator koji možete pretvoriti u Python listu na sljedeći način:

```
reads = list(parse("ime_datoteke", "tip_datoteke"))
```

Tip datoteke postavite na "fasta" ili "fastq".

Učitajte obje datoteke te ispišite broj zapisa u svakoj od njih (broj elemenata u listi). Datoteka koja sadrži referencu trebala bi imati samo jedan zapis, dok bi datoteka s očitanjima trebala sadržavati veći broj zapisa.

NAPOMENA: Ako niste sigurni kako pronaći datoteke na disku iz Jupyter notebook-a, uvijek možete provjeriti radni direktorij sljedećim naredbama:

```
import os
os.getcwd()
```

i promijeniti ga sa:

```
os.chdir()
```

Ako pak radite u Google Colab okruženju, koristite upute za učitavanje datoteka s Google diska iz prve laboratorijske vježbe.

```
from Bio.SeqIO import parse

referenca = list(parse("escherichia_coli_reference_nova.fasta",
"fasta"))
print(f"Broj zapisa u ocitanju reference: {len(referenca)}")

skup_ocitanja = list(parse("ecoli_ILL_small_nova.fastq", "fastq"))
print(f"Broj zapisa u skupu očitanja dobivenih sekvenciranjem:
{len(skup_ocitanja)}")
```

```
Broj zapisa u ocitanju reference: 1
Broj zapisa u skupu očitanja dobivenih sekvenciranjem: 38585
```

1. Zadatak

Svaki zapis koji ste učitali pomoću metode *Bio.SeqIO.parse()* sadrži Veći broj podataka od kojih su nam bitni samo neki. Naredbom print ispišite cijeli prvi zapis iz datoteke s očitanjima i iz datoteke s referencom.

Vidjet ćete da oba zapisa (među ostalim podacima) sadrže identifikator zapisa i sekvencu. Identifikator zapisa možete dohvatiti pomoću

```
zapis.id
```

dok sekvencu možete dohvatiti pomoću

```
zapis.seq
```

Ispišite identifikator i sekvencu za prvo očitanje te identifikator i prvih 200 znakova za referentni genom E.coli.

NAPOMENA: Referentni genom Escherichia coli je dugačak oko 4.5 milijuna slova

```
print("prvi zapis iz datoteke s očitanjima:")
print(skup_ocitanja[0])

print()
print("prvi zapis iz datoteke s referencom:")
print(referenca[0])

print()
print(f"1. očitanje - Identifikator: {skup_ocitanja[0].id}, sekvenca:
{skup_ocitanja[0].seq}")

print()
print(f"referenca - Identifikator: {referenca[0].id}, sekvenca:
{referenca[0].seq[:200]}")
```

```
prvi zapis iz datoteke s očitanjima:
ID: SRR2052522.671
Name: SRR2052522.671
Description: SRR2052522.671 HWI-EAS390_0001:4:1:6915:1123/1
Number of features: 0
Per letter annotation for: phred_quality
Seq('GATCTGGTGACCGGGTCGCGCAAAGTGATCATCGCCATGGAACATTGCGCCAAA...TGC')

prvi zapis iz datoteke s referencom:
ID: NC_000913.3
Name: NC_000913.3
Description: NC_000913.3 Escherichia coli str. K-12 substr. MG1655,
complete genome
Number of features: 0
Seq('AGCTTTTCATTCTGACTGCAACGGGCAATATGTCTCTGTGTGGATTAAAAAAAG...TTC')

1. očitanje - Identifikator: SRR2052522.671, sekvenca:
GATCTGGTGACCGGGTCGCGCAAAGTGATCATCGCCATGGAACATTGCGCCAAAGATGGTTCAGCAAAAA
TTTTGGGCCTCTGTATCATGCCACTCACTGCGCAATATCCGGATCAAATGC

referenca - Identifikator: NC_000913.3, sekvenca:
AGCTTTTCATTCTGACTGCAACGGGCAATATGTCTCTGTGTGGATTAAAAAAAGAGTGTCTGATAGCAGC
TTCTGAACTGGTTACCTGCCGTGAGTAAATTAAAATTTTATTGACTTAGGTCACTAAATACTTTAACCAA
TATAGGCATAGCGCACAGACAGATAAAAATTACAGAGTACACAACATCCATGAAACGCAT
```

1. Zadatak

Da bismo sekvence DNA analizirali metodama obrade signala, moramo pojednim nukleotidima (slovima) dodijeliti brojčane vrijednosti. Napišite funkciju u Pythonu koja će primiti slovo koje predstavlja nukleotid i vratiti odgovarajuću brojčanu vrijednost. Vrijednosti dodijelite na sljedeći način:

- A = 3
- G = 2
- C = -2
- T = -3

DNA sekvence mogu sadržavati i neke druge znakove (npr. 'N' koji označava da taj nuklotid nije poznat), njima dodijelite vrijednost 0. Također se može dogoditi da nukleotidi budu označeni i malim slovima, pa vodite računa da vaša funkcija mora vratiti ispravnu vrijendost i u tom slučaju.

```
def dna_nucleotide_value(nucleotide):
    nucleotide = nucleotide.upper()
    values = {
        'A': 3,
        'G': 2,
        'C': -2,
        'T': -3
    }

    return values.get(nucleotide, 0)
```

1. Zadatak

Upotrebite napisanu funkciju da bi od prvog očitanja i od reference kreirali signal. Izračunajte korelaciju pomoću Fourierove transformacije. Zanemarite imaginarne vrijednosti.

```
prvo_ocitanje_signal = list(map(dna_nucleotide_value,
skup_ocitanja[0].seq))
referenca_signal = list(map(dna_nucleotide_value, referenca[0].seq))

def calculate_correlation(signal1, signal2):
    # Dovesti oba signala na istu duljinu (zero-padding) za korelaciju
    length = max(len(signal1), len(signal2))
    signal1 = np.pad(signal1, (0, length - len(signal1)))
    signal2 = np.pad(signal2, (0, length - len(signal2)))

    # Fourierova transformacija signala
    fft1 = np.fft.fft(signal1)
    fft2 = np.fft.fft(signal2)

    # Inverzna Fourierova transformacija umnoska (jedan signal je
konjugiran)
    correlation = np.fft.ifft(fft1 * np.conj(fft2))

    # Zanemarujemo imaginarni dio; korelacija realnih signala mora
biti realna
```

```
    return np.real(correlation)

korelacija_prvog = calculate_correlation(referenca_signal,
prvo_ocitanje_signal)
print("Korelacija između referetnog genoma i prvog očitanja je
sljedeća:")
print(korelacija_prvog)
max_index_prvog = np.argmax(korelacija_prvog)
max_value_prvog = korelacija_prvog[max_index_prvog]
print(f"Najevca vrijendost korelacije se pronalazi za pomak:
{max_index_prvog}, i ona iznosi: {max_value_prvog}")
```

```
Korelacija između referetnog genoma i prvog očitanja je sljedeća:
[210. 110.   2. ... -17.  22.  50.]
Najevca vrijendost korelacije se pronalazi za pomak: 2324486, i ona
iznosi: 692.9999999999997
```

1. Zadatak

Ispišite duljinu reference. Koristeći metode biblioteke *numpy*, izračunajte srednju vrijednost i standardnu devijaciju duljine očitanja (uzmite u obzir sva očitanja).

Primijetit ćete da su sva očitanja jednake duljine.

```
print(f"Duljina reference: {len(referenca[0])}")

cista_ocitanja = [list(ocitanje.seq) for ocitanje in skup_ocitanja]
duljine_ocitanja = [len(ocitanje) for ocitanje in cista_ocitanja]

mean_ocitanja = np.mean(duljine_ocitanja)
print(f"Srednja vrijednost duljine ocitanja: {mean_ocitanja}")
std_ocitanja = np.std(duljine_ocitanja)
print(f"Standardna devijacija duljine ocitanja: {std_ocitanja}")
```

```
Duljina reference: 4641652
Srednja vrijednost duljine ocitanja: 121.0
Standardna devijacija duljine ocitanja: 0.0
```

1. zadatak

Što ako želimo izračunati korelaciju za veći broj očitanja i istu referencu? To je tipičan slučaj u bioinformatici jer uređaji za sekvenciranje proizvode tisuće i desetke tisuća očitanja koja se potom mapiraju na istu referencu.

Ako korelaciju računamo izravno, potrebno ju je svaki put izračunati iz početka. Ako korelaciju računamo pomoću FFT-a, transformaciju za referencu potrebno je napraviti samo jednom.

Izračunajte korelacije za prvih 10 očitanja.

```
fft_reference = np.fft.fft(referenca_signal)
```

```python
# Funkcija za izračun korelacije pomoću FFT-a
def calculate_correlation_fft(fft_ref, signal):
    length = max(len(fft_ref), len(signal))
    signal = np.pad(signal, (0, length - len(signal)))

    fft_signal = np.fft.fft(signal)

    # Inverzna Fourierova transformacija umnoška (jedan signal je
konjugiran)
    correlation = np.fft.ifft(fft_ref * np.conj(fft_signal))

    return np.real(correlation)

for i in range(10):
    ocitanje_signal = list(map(dna_nucleotide_value,
list(skup_ocitanja[i].seq)))
    korelacija = calculate_correlation_fft(fft_reference,
ocitanje_signal)

    # Pronalazak najveće vrijednosti korelacije i indeksa
    max_index = np.argmax(korelacija)
    max_value = korelacija[max_index]

    print(f"Korelacija za očitanje {i+1}:")
    print(korelacija)
    print(f"Najveća vrijednost korelacije: {max_value} za pomak:
{max_index}\n")
```

```
Korelacija za očitanje 1:
[210. 110.   2. ... -17.  22.  50.]
Najveća vrijednost korelacije: 692.9999999999997 za pomak: 2324486

Korelacija za očitanje 2:
[   9.   -3.   50. ... -122.   45.  -23.]
Najveća vrijednost korelacije: 710.9999999999995 za pomak: 1877260

Korelacija za očitanje 3:
[ 58. -75.  78. ... 104.  31.  65.]
Najveća vrijednost korelacije: 733.9999999999997 za pomak: 557777

Korelacija za očitanje 4:
[-19.  55.   5. ... 100.  62. -16.]
Najveća vrijednost korelacije: 362.9999999999997 za pomak: 1144877

Korelacija za očitanje 5:
[ 153. -137.   21. ...    3.  -14.   86.]
Najveća vrijednost korelacije: 662.9999999999997 za pomak: 3583639

Korelacija za očitanje 6:
[ 56.  46. -72. ... -34. -90. -85.]
Najveća vrijednost korelacije: 781.9999999999995 za pomak: 4051518
```

```
Korelacija za očitanje 7:
[ 83.  -2.  17. ... 134.  46.  12.]
Najveća vrijednost korelacije: 358.9999999999997 za pomak: 2293706

Korelacija za očitanje 8:
[ -25.   -1.   37. ...   70.   71. -157.]
Najveća vrijednost korelacije: 800.9999999999997 za pomak: 1011323

Korelacija za očitanje 9:
[111. 134. -42. ...  37. -29. -82.]
Najveća vrijednost korelacije: 636.9999999999995 za pomak: 628546

Korelacija za očitanje 10:
[ -3.  46. -78. ... -81.  17. -13.]
Najveća vrijednost korelacije: 676.9999999999995 za pomak: 1497921
```

1. zadatak

Na temelju najveće vrijednosti korelacije između reference i prvog očitanja pronađite poziciju na referenci koja je najsličnija očitanju. Pozicija odgovara vrijednosti parametra k za koji je korelacija najveća.

Napišite metodu koja će primiti dva niza znakova jednake duljine, usporediti znakove na istim pozicijama i vratiti broj razlika (Hammingova udaljenost).

"Izrežite" dio reference koji je najsličniji očitanju (iste duljine kao i očitanje) i usporedite ga s očitanjem pomoću napisane funkcije.

```
def hamming_distance(seq1, seq2):
    if len(seq1) != len(seq2):
        raise ValueError("Sekvence moraju biti jednake duljine.")

    return sum(el1 != el2 for el1, el2 in zip(seq1, seq2))

# Prvo očitanje
prvo_ocitanje_signal = list(map(dna_nucleotide_value,
list(skup_ocitanja[0].seq)))
korelacija = calculate_correlation_fft(fft_reference,
prvo_ocitanje_signal)

# Pronalazak maksimalne vrijednosti korelacije i njenog indeksa
(pomaka)
max_index = np.argmax(korelacija)
max_value = korelacija[max_index]

# Izdvajanje dijela reference koji je najbliži očitanju
reference_segment_length = len(prvo_ocitanje_signal)
reference_segment_signal = referenca_signal[max_index:max_index +
```

```
reference_segment_length]

reference_segment = list(referenca[0].seq)[max_index:max_index +
reference_segment_length]
prvo_ocitanje = list(skup_ocitanja[0].seq)

# Izračunavanje Hammingove udaljenosti
hamming_dist = hamming_distance(reference_segment_signal,
prvo_ocitanje_signal)

print(f"Najveća vrijednost korelacije između reference i prvog
očitanja: {max_value} za pomak: {max_index}")
print(f"Dio reference: {''.join(reference_segment)}")
print(f"Očitanje:      {''.join(prvo_ocitanje)}")
print(f"Hammingova udaljenost između referentnog segmenta i očitanja:
{hamming_dist}")
```

```
Najveća vrijednost korelacije između reference i prvog očitanja:
692.9999999999997 za pomak: 2324486
Dio reference:
GATCTGGTGACCGGGTCGCGCAAAGTGATCATCGCCATGGAACATTGCGCCAAAGATGGTTCAGCAAAAA
TTTTGCGCCGCTGCACCATGCCACTCACTGCGCAACATGCGGTGCATATGC
Očitanje:
GATCTGGTGACCGGGTCGCGCAAAGTGATCATCGCCATGGAACATTGCGCCAAAGATGGTTCAGCAAAAA
TTTTGGGCCTCTGTATCATGCCACTCACTGCGCAATATCCGGATCAAATGC
Hammingova udaljenost između referentnog segmenta i očitanja: 9
```

1. zadatak

U datoteci "ecoli_ILL_small_aln.sam" dana su već izračunata poravnanja svih očitanja na referencu u SAM formatu. SAM je tekstualni "tab separated" format. U prvom stupcu se nalati identifikator očitanja, dok se u četvrtom stupcu nalazi pozicija na referenci na koju je očitanje najbolje poravnato (ostali stupci nas ne zanimaju). Također, datoteka s poravnanjima sadrži i nekoliko *header* readaka kojima prvi stupac počinje sa znakom '@', njih također možete zanemariti.

Otvorite datoteku s poravnanjima i pronađite poravnanje za prvo očitanje (identifikator očitanja u datoteci s očitanjima i datoteci s poravnanjima mora biti jednak). Usporedite poziciju u datoteci sa pozicijom koju ste dobili pomoću korelacije.

UPOUTA: TSV datoteke možete otvoriti na sljedeći način:

```
tsv_file = open("file_name")
tsv_rows = csv.reader(tsv_file, delimiter="\t")
```

Varijabla *tsv_rows* će sadržavati listu redaka, a svaki redak biti lista vrijednosti (po jedna za svaki stupac).

```
import csv
```

```
prvo_ocitanje_id = skup_ocitanja[0].id

with open("ecoli_ILL_small_aln_nova.sam") as tsv_file:
    tsv_rows = csv.reader(tsv_file, delimiter="\t")

    for row in tsv_rows:
        if row[0].startswith('@'):
            continue

        if row[0] == prvo_ocitanje_id:
            zapisan_korak = int(row[3])

            print(f"Pozicija iz SAM datoteke za prvo očitanje
({prvo_ocitanje_id}): {zapisan_korak}")
            break

print(f"Pozicija dobivena putem korelacije: {max_index}")

# Usporedba dviju pozicija
if zapisan_korak == max_index:
    print("Pozicije su jednake.")
else:
    print("Pozicije se razlikuju.")
```

```
Pozicija iz SAM datoteke za prvo očitanje (SRR2052522.671): 2324487
Pozicija dobivena putem korelacije: 2324486
Pozicije se razlikuju.
```

1. zadatak

Za prvo očitanje pozicija dobivena pomoću korelacije trabala bi biti 2324486, dok je pozicija u datoteci s poravnanjima 2324487. Razlikuju se samo za 1 pa možemo zaključiti da nam je korelacija dala dobru poziciju za poravnanje.

Prisjetimo se da korelacija ne računa točno poravnanje već ju koristimo samo da bi našli kandidatne pozicije za točno računanje. Tek onda na takvim pozicijama možemo točno poravnanje izračunati pomoću algoritama dinamičkog programiranja. Ako bi primijenili dinamičko programiranje za računanje poravnanja očitanja s cijelom referencom, postupak bi bio znatno sporiji i zahtijevao bi veliku količinu radne memorije.

Ako želite to možete isprobati pomoću algoritama za poravnanje u biblioteci *bioparser*. Lokalno poravnanje možete izračunati na sljedeći način:

```
from Bio import Align
aligner = Align.PairwiseAligner()
aligner.mode = "local"
alignments = aligner.align(reference[0].seq[pos-20:pos+len(read_sig)
+20], reads[1].seq)
```

Za **drugu** skupinu od 100 očitanja (101. - 200.) izračunajte korelaciju te pomoću korelacije poziciju najveće sličnosti očitanja i reference. Usporedite rezultat sa podacima u datoteci s poravnanjima. Ispišite broj očitanja za koja se dvije pozicije razlikuju za najviše 5 mjesta.

```
alignment_positions = {}
with open("ecoli_ILL_small_aln_nova.sam") as tsv_file:
    tsv_rows = csv.reader(tsv_file, delimiter="\t")
    for row in tsv_rows:
        if row[0].startswith('@'):
            continue
        read_id = row[0]
        position = int(row[3])
        alignment_positions[read_id] = position

matching_count = 0

for i in range(100, 200):
    ocitanje_signal = list(map(dna_nucleotide_value,
list(skup_ocitanja[i].seq)))
    korelacija = calculate_correlation_fft(fft_reference,
ocitanje_signal)

    max_index = np.argmax(korelacija)
    predicted_position = max_index + 1  # Korekcija jer je 0-
indeksirano

    read_id = skup_ocitanja[i].id
    sam_position = alignment_positions.get(read_id, None)

    if sam_position:
        if abs(predicted_position - sam_position) <= 5:
            matching_count += 1

print(f"Broj očitanja za koja se pozicija dobivena korelacijom
razlikuje od SAM pozicije za najviše 5 mjesta: {matching_count}")
```

```
Broj očitanja za koja se pozicija dobivena korelacijom razlikuje od
SAM pozicije za najviše 5 mjesta: 41
```

1. zadatak

Očekivani broj točno pozicioniranih očitanja je otprilike 50, jer smo za sada uspješno radili samo s očitanjima na jednom lancu DNA.

Odaberite jedno očitanje koje NIJE uspješno pozicionirano. Ono bi se trebalo nalaziti na drugom (reverzno-kompleentarnom) DNA lancu. Napravite reverzni komplement odabranog očitanja. Možete koristiti metodu reverse_complement klase Seq.

```
rc_seq = seq.reverse_complement()
```

Izračunajte pomoću korelacije poziciju poravnanja za reverzno komplementarno očitanje, te je usporedite s pozicijom iz datoteke. Sada bi se trebale poklapati!

```
misaligned_read = None
misaligned_index = -1
for i in range(100, 200):
    ocitanje_signal = list(map(dna_nucleotide_value,
list(skup_ocitanja[i].seq)))
    korelacija = calculate_correlation_fft(fft_reference,
ocitanje_signal)

    max_index = np.argmax(korelacija)
    predicted_position = max_index + 1

    read_id = skup_ocitanja[i].id
    sam_position = alignment_positions.get(read_id, None)

    if sam_position and abs(predicted_position - sam_position) > 5:
        misaligned_read = skup_ocitanja[i]
        misaligned_index = i
        break

if misaligned_read:
    print(f"Neporavnano očitanje pronađeno s ID-jem:
{misaligned_read.id}")

    rc_seq = misaligned_read.seq.reverse_complement()
    rc_signal = list(map(dna_nucleotide_value, list(rc_seq)))

    rc_korelacija = calculate_correlation_fft(fft_reference,
rc_signal)

    rc_max_index = np.argmax(rc_korelacija)
    rc_predicted_position = rc_max_index + 1

    print(f"Originalna pozicija u SAM datoteci: {sam_position}")
    print(f"Pozicija za reverzni komplement pomoću korelacije:
{rc_predicted_position}")
else:
    print("Nije pronađeno neporavnano očitanje koje zadovoljava
kriterije.")
```

```
Neporavnano očitanje pronađeno s ID-jem: SRR2052522.55677
Originalna pozicija u SAM datoteci: 1668019
Pozicija za reverzni komplement pomoću korelacije: 1668019
```

1. ZAKLJUČAK

Prolaskom kroz zadatke u ovoj vježbi dobili ste kratak uvod u rad s bioinformatičkim podacima i tehnikama obrade signala.