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.

```
try:
  import google.colab
  !pip install biopython
except ImportError:
  pass
import numpy as np
import Bio
print("Biopython verzija: ", Bio.__version__)
print("Numpy verzija: ", np.__version__)
```

```
Requirement already satisfied: biopython in /usr/local/lib/python3.6/dist-packages (1.78)
Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-packages (from biopython
```

```
Biopython verzija:  1.78
Numpy verzija:  1.18.5
```

2. 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 google.colab import drive
drive.mount('/content/drive')
from Bio import SeqIO
#from scipy import io
reference = list(SeqIO.parse("drive/My Drive/Colab Notebooks/escherichia_coli_reference.fasta", "fas
ocitanja = list(SeqIO.parse("drive/My Drive/Colab Notebooks/ecoli_ILL_small.fastq", "fastq"))
print("Broj referenci: ", len(reference)) #broj referenci, trebao bi biti 1
print("Broj očitanja: ", len(ocitanja)) #broj ocitanja
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/cor
Broj referenci:  1
Broj očitanja:  38585
```

3. 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("Identifikator i sekvenca za prvo očitanje:")
print(ocitanja[0].id)
print(ocitanja[0].seq)
print()
print("Identifikator i prvih 200 znakova za referentni genom E.coli:")
print(reference[0].id)
print(reference[0].seq[0:200])
```

```
Identifikator i sekvenca za prvo očitanje:
SRR2052522.671
GATCTGGTGACCGGGTCGCGCAAAGTGATCATCGCCATGGAACATTGCGCCAAAGATGGTTCAGCAAAAATTTTGGGCCTCTGTATCATGCCACT

Identifikator i prvih 200 znakova za referentni genom E.coli:
NC_000913.3
AGCTTTTCATTCTGACTGCAACGGGCAATATGTCTCTGTGTGGATTAAAAAAAGAGTGTCTGATAGCAGCTTCTGAACTGGTTACCTGCCGTGAG
```

4. 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 broj_nukl(slovo: str) -> int:
  if slovo == "A":
    return 3
  elif slovo == "G":
    return 2
  elif slovo == "C":
    return -2
  elif slovo == "T":
    return -3
  else:
    return 0
```

5. Zadatak

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

```
signal_oc = []
signal_ref = []
for znak in ocitanja[0].seq:
  signal_oc.append(broj_nukl(znak))
for znak in reference[0].seq:
  signal_ref.append(broj_nukl(znak))

len_oc = len(signal_oc)
len_ref = len(signal_ref)
k_arr = range(-len_oc+1,len_ref)

values =[-3, -2, 2, 3]
avg = np.average(values)

std = np.std(values)


x1 = signal_ref
x2 = signal_oc
x1 = [(x-avg)/std for x in x1]
x2 = [(x-avg)/std for x in x2]

padding1 = [0]*(len_oc-1)
padding2 = [0]*(len_ref-1)

X1 = np.fft.fft(padding1+x1)
X2 = np.fft.fft(x2+padding2)
```

```
Cor = np.conjugate(X2)*X1
cor = np.fft.ifft(Cor)

k = k_arr[cor.argmax()]

print("Korelacija koristeći FFT:")
print(np.real(cor))
#print("k={}".format(k))
```

```
Korelacija koristeći FFT:
[-0.92307692  0.30769231 -0.15384615 ... -1.38461538 -1.84615385
 -0.61538462]
```

6. 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("Duljina reference: ", len_ref)
duljine_oc = []

for ocitanje in ocitanja:
  duljine_oc.append(len(ocitanje.seq))

print("Srednja vrijednost očitanja: ", np.mean(duljine_oc))
print("Stand. devijacija očitanja: ", np.std(duljine_oc))
```

```
Duljina reference:  4641652
Srednja vrijednost očitanja:  121.0
Stand. devijacija očitanja:  0.0
```

7. 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.

```
signal_ref = []
```

```
for znak in reference[0].seq:
  signal_ref.append(broj_nukl(znak))

len_ref = len(signal_ref)

values =[-3, -2, 2, 3]
avg = np.average(values)

std = np.std(values)
x1 = signal_ref
x1 = [(x-avg)/std for x in x1]
padding2 = [0]*(len_ref-1)

for i in range(10):
  signal_oc = []
  for znak in ocitanja[i].seq:
    signal_oc.append(broj_nukl(znak))
  len_oc = len(signal_oc)
  k_arr = range(-len_oc+1,len_ref)

  x2 = signal_oc

  x2 = [(x-avg)/std for x in x2]

  padding1 = [0]*(len_oc-1)


  X1 = np.fft.fft(padding1+x1)
  X2 = np.fft.fft(x2+padding2)

  Cor = np.conjugate(X2)*X1
  cor = np.fft.ifft(Cor)

  k = k_arr[cor.argmax()]

  print("Korelacija koristeći FFT za očitanje br. ", i+1)
  print(np.real(cor))
  #print("k={}".format(k))
```

```
Korelacija koristeći FFT za očitanje br.  1
[-0.92307692  0.30769231 -0.15384615 ... -1.38461538 -1.84615385
 -0.61538462]
Korelacija koristeći FFT za očitanje br.  2
[ 1.38461538e+00 -6.66134596e-15 -2.92307692e+00 ... -1.38461538e+00
 -2.00000000e+00 -9.23076923e-01]
Korelacija koristeći FFT za očitanje br.  3
[-0.92307692 -2.         -1.69230769 ... -2.76923077 -1.53846154
 -0.61538462]
Korelacija koristeći FFT za očitanje br.  4
[ 1.38461538 -0.46153846 -0.92307692 ... -1.38461538 -0.76923077
 -0.92307692]
Korelacija koristeći FFT za očitanje br.  5
[ 1.38461538e+00 -8.43769769e-15 -2.46153846e+00 ...  1.53846154e-01
 -7.69230769e-01 -9.23076923e-01]
```

```
Korelacija koristeći FFT za očitanje br.  6
[ 0.92307692  1.53846154  1.38461538 ... -3.23076923 -2.
 -0.92307692]
Korelacija koristeći FFT za očitanje br.  7
[-9.23076923e-01  7.69230769e-01  2.46153846e+00 ... -1.38461538e+00
  2.22046709e-15  6.15384615e-01]
Korelacija koristeći FFT za očitanje br.  8
[ 0.92307692  2.         -1.07692308 ...  1.38461538  0.76923077
  0.92307692]
Korelacija koristeći FFT za očitanje br.  9
[-1.38461538 -2.30769231 -1.38461538 ... -1.84615385 -2.30769231
 -0.92307692]
Korelacija koristeći FFT za očitanje br.  10
[-0.92307692 -1.53846154  0.92307692 ... -0.92307692 -0.46153846
 -0.92307692]
```

8. 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(niz1: str, niz2: str) -> int:
  #nizovi znakova su po pretpostavci iste duljine
  output = 0
  for i in range(len(niz1)):
    if niz1[i] != niz2[i]:
      output += 1

  return output


signal_oc = []
signal_ref = []
for znak in ocitanja[0].seq:
  signal_oc.append(broj_nukl(znak))
for znak in reference[0].seq:
  signal_ref.append(broj_nukl(znak))

len_oc = len(signal_oc)
len_ref = len(signal_ref)
k_arr = range(-len_oc+1,len_ref)

values =[-3, -2, 2, 3]
avg = np.average(values)

std = np.std(values)
```

```
x1 = signal_ref
x2 = signal_oc
x1 = [(x-avg)/std for x in x1]
x2 = [(x-avg)/std for x in x2]

padding1 = [0]*(len_oc-1)
padding2 = [0]*(len_ref-1)

X1 = np.fft.fft(padding1+x1)
X2 = np.fft.fft(x2+padding2)

Cor = np.conjugate(X2)*X1
cor = np.fft.ifft(Cor)

k = k_arr[cor.argmax()]

#print("Korelacija koristeći FFT:")
#print(np.real(cor))
print("k={}".format(k))

ref1 = reference[0].seq[k:k+len_oc]
izlaz = hamming(ref1, ocitanja[0].seq)
print("Hammingova udaljenost: ", izlaz)
```

```
k=2324486
Hammingova udaljenost:  9
```

9. 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
tsv_file = open("drive/My Drive/Colab Notebooks/ecoli_ILL_small_aln.sam")
tsv_rows = csv.reader(tsv_file, delimiter="\t")
```

```
tsv_rows = csv.reader(tsv_file, delimiter='\t')
data = list(tsv_rows)
#print(data[2])
#print(ocitanja[0].id)

#naci id koji je isti kao ocitanja[0].id

for i in range(len(data)):
  if data[i][0] == ocitanja[0].id:
    print("Pozicija iz datoteke: ", data[i][3])
    break

print("Pozicija dobivena korelacijom: ", k)
```

```
Pozicija iz datoteke:  2324487
Pozicija dobivena korelacijom:  2324486
```

10. 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 metodom:

```
Bio.pairwise2.align.localxx(seq1, seq2)
```

Za prvih 100 očitanja 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.

```
signal_ref = []

for znak in reference[0].seq:
  signal_ref.append(broj_nukl(znak))

len_ref = len(signal_ref)
```

```
values =[-3, -2, 2, 3]
avg = np.average(values)
std = np.std(values)

x1 = signal_ref
x1 = [(x-avg)/std for x in x1]
padding2 = [0]*(len_ref-1)

tocni = 0

for i in range(100):
  signal_oc = []
  for znak in ocitanja[i].seq:
    signal_oc.append(broj_nukl(znak))
  len_oc = len(signal_oc)
  k_arr = range(-len_oc+1,len_ref)

  x2 = signal_oc

  x2 = [(x-avg)/std for x in x2]

  padding1 = [0]*(len_oc-1)


  X1 = np.fft.fft(padding1+x1)
  X2 = np.fft.fft(x2+padding2)

  Cor = np.conjugate(X2)*X1
  cor = np.fft.ifft(Cor)

  k = abs(int(k_arr[cor.argmax()]))


  #print("k={}".format(k))
  for j in range(len(data)):
    if data[j][0] == ocitanja[i].id: #nasli smo id
      #print("Razlika: ", abs(int(data[j][3]) - k))

      if abs(abs(int(data[j][3])) - k) <= 5:
        tocni += 1
        #print("Broj točnih očitanja: ", tocni)
      break

print("Broj točnih očitanja: ", tocni)
#print("Broj onih kojima smo našli id: ", broj)

     Broj točnih očitanja:  48
```

11. ZAKLJUČAK

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

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