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.

```
# Ovo je mjesto na kojem možete izvoditi svoj kod.
try:
  import google.colab
  !pip install biopython
except ImportError:
  pass

import numpy as np
print("Numpy v", np.__version__)
import Bio
print("Bio v", Bio.__version__)
```

```
Requirement already satisfied: biopython in /usr/local/lib/python3.7/dist-pack
```

```
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages
Numpy v 1.19.5
Bio v 1.79
```

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.

```
# Ovo je mjesto na kojem možete izvoditi svoj kod.

import os
os.getcwd()
# print(os.getcwd())
# iz Bio.Seql() importaj parse
from Bio.SeqIO import parse
from google.colab import drive
drive.mount('/content/drive')

fileFASTA = "/content/drive/My Drive/Colab Notebooks/escherichia_coli_reference.fas
fileFASTQ = "/content/drive/My Drive/Colab Notebooks/ecoli_ILL_small.fastq"
```

```
#parse je metoda za ucitavanje datoteke i na ovaj nacin dobijemo listu iz iteratora
FASTA = list(parse(fileFASTA, "fasta"))
FASTQ = list(parse(fileFASTQ, "fastq"))

#FASTA je referenca s kojom usporedujemo
print("Broj zapisa u FASTA datoteci: ", len(FASTA))
#FASTQ su ocitanja koja usporedujemo s referencom
print("Broj zapisa u FASTQ datoteci: ", len(FASTQ))
```

```
Mounted at /content/drive
Broj zapisa u FASTA datoteci:  1
Broj zapisa u FASTQ datoteci:  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

```
# Ovo je mjesto na kojem možete izvoditi svoj kod.
# FASTA i FASTQ su liste
print("Cijeli prvi zapis iz datoteke s očitanjima: \n", FASTQ[0])
print("_____________________________________________________")
print("Cijeli prvi zapis iz datoteke s referencom: \n", FASTA[0])
print("_____________________________________________________")
print("Identifikator za prvo očitanje: ", FASTQ[0].id)
print("_____________________________________________________")
print("Sekvenca za prvo očitanje: \n", FASTQ[0].seq)
print("_____________________________________________________")
print("Identifikator za referentni genom: ", FASTA[0].id)
print("_____________________________________________________")
print("Prvih 200 znakova za referentni genom E.Coli: \n",FASTA[0].seq[0:200:1])
```

```
Cijeli 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')
_____________________________________________________________
Cijeli 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 g
Number of features: 0
Seq('AGCTTTTCATTCTGACTGCAACGGGCAATATGTCTCTGTGTGGATTAAAAAAAG...TTC')
_____________________________________________________________
Identifikator za prvo očitanje:  SRR2052522.671
_____________________________________________________________
Sekvenca za prvo očitanje:
 GATCTGGTGACCGGGTCGCGCAAAGTGATCATCGCCATGGAACATTGCGCCAAAGATGGTTCAGCAAAAATTTTGGG
_____________________________________________________________
Identifikator za referentni genom:  NC_000913.3
_____________________________________________________________
Prvih 200 znakova za referentni genom E.Coli:
 AGCTTTTCATTCTGACTGCAACGGGCAATATGTCTCTGTGTGGATTAAAAAAAGAGTGTCTGATAGCAGCTTCTGAA
```

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.

```
# Ovo je mjesto na kojem možete izvoditi svoj kod.
def slovoBroj(slovo):
  if slovo == 'A' or slovo == 'a':
    return 3
  elif slovo == 'G' or slovo == 'g':
    return 2
  elif slovo == 'C' or slovo == 'c':
    return -2
  elif slovo == 'T' or 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.

```
# Ovo je mjesto na kojem možete izvoditi svoj kod.
# prvo ocitanje - FASTQ[0].seq
# prva referenca - FASTA[0].seq

#Sekvence nukleotida pretvaramo u sekvence brojeva tj. u signal
x1 = [slovoBroj(s) for s in FASTA[0].seq]
x2 = [slovoBroj(s) for s in FASTQ[0].seq]

#uzmi veličine
len1 = len(x1)
len2 = len(x2)
#ode napravis polje s vrdijednostima paramwtra k...
k_arr = range(-len2+1,len1)

#uzmi average
avg1 = np.mean(x1)
avg2 = np.mean(x2)

#uzmi standardnu devijaciju
std1 = np.std(x1)
std2 = np.std(x2)

#normaliziramo signal - po fomrmuli..
x1 = [(x-avg1)/std1 for x in x1]
x2 = [(x-avg2)/std2 for x in x2]

#dodajemo nule da bi popunili korelaciju
padding1 = [0]*(len2-1)
padding2 = [0]*(len1-1)

#računamo korelaciju pomoću FFT
#prvo DFT od jednog i drugog, zatim konjugiramo jedan, pomnozimo
# i onda Inverzni DFT tog medukoraka daje rjesenje!
#u prvi dodajes 0 na POCETAK, a u drugom ih dodajes na KRAJ
X1 = np.fft.fft(padding1 + x1)
X2 = np.fft.fft(x2 + padding2)

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

# ispisujemo k za koji je korelacija najveca
k = k_arr[cor.argmax()]
K = k
print("Correlation by FFT:")
#zanemarujemo imaginarni dio korelacije
```

```
print(cor.real)
print("k = ", k)
```

```
Correlation by FFT:
[-0.96628788  0.24537804 -0.19311781 ... -1.28947659 -1.7916225
 -0.59293484]
k =  2324486
```

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.

```
# Ovo je mjesto na kojem možete izvoditi svoj kod.
print("Duljina reference: ", len1)

# uzimamo u obzir sva očitanja i gledamo duljine
seq = [len(s.seq) for s in FASTQ]
print("Srednja vrijednost: ", np.mean(seq))
print("Standardna devijacija: ", np.std(seq))
```

```
Duljina reference:  4641652
Srednja vrijednost:  121.0
Standardna devijacija:  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.

```
# Ovo je mjesto na kojem možete izvoditi svoj kod.

#imamo jednu referencu (FASTA) i 10 očitanja (FASTQ)
referenca = [slovoBroj(s) for s in FASTA[0].seq]
len1 = len(referenca)

values = [3, 2, -2, -1]
avg = np.mean(values)
std = np.std(values)

for i in range(10):
  print("očitanje ", i + 1 ,':\n')
```

```
ocitanje = [slovoBroj(s) for s in FASTQ[i].seq]
len2 = len(ocitanje)
# normalizacija po formuli
referenca = [(x - avg) / std for x in referenca]
ocitanje = [(x - avg) / std for x in ocitanje]

Referenca = np.fft.fft(padding1 + referenca)
Ocitanje = np.fft.fft(ocitanje + padding2)

Cor = np.conjugate(Ocitanje) * Referenca
cor = np.fft.ifft(Cor)

k = k_arr[cor.argmax()]

#samo realni dio
print(np.real(cor))
```

```
očitanje  1 :

[-1.47058824  0.         -0.05882353 ... -1.23529412 -2.70588235
 -0.88235294]
očitanje  2 :

[ 0.41922243 -0.28533603 -1.72825552 ... -0.65802919 -1.89726838
 -1.00745772]
očitanje  3 :

[ 0.0907649   0.35624407  1.23071977 ... -1.8882641  -1.02241269
 -0.46968385]
očitanje  4 :

[-0.33814509  0.06812015 -0.30932542 ... -0.67383454 -0.06713789
 -0.67383454]
očitanje  5 :

[-0.45814211 -0.03256666  0.32787547 ...  0.62097541 -0.03256666
 -0.62097541]
očitanje  6 :

[-0.30980953 -0.62909735 -1.19283801 ... -1.57314635 -0.96833309
 -0.59533496]
očitanje  7 :

[ 0.54458381  0.00766274 -0.29609931 ... -0.58289752  0.00766274
  0.58289752]
očitanje  8 :

[-0.33496776 -0.89547754 -0.12652383 ...  1.27207543  0.46669535
  0.80761027]
očitanje  9 :

[ 0.79089224  1.58430868  2.3878219  ... -0.34796881 -1.14967904
 -0.57393802]
očitanje  10 :
```

```
[ 0.56814557  1.13716572  0.80065129 ... -0.34316125  0.22813281
 -0.57251848]
```

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.

```
# Ovo je mjesto na kojem možete izvoditi svoj kod.
from Bio.SeqIO import parse

def usporedba(prvi, drugi):
  brojRazlika = 0
  for i in range(len(prvi)):
    if prvi[i] != drugi[i]:
      brojRazlika = brojRazlika + 1
  return brojRazlika

#sekvenca prvog ocitanja
prvi = FASTA[0].seq
#skevenca reference
drugi = FASTQ[0].seq

#izjednaci duljine
Prvi = prvi[K : K + len(drugi)]
Drugi = drugi

print("Razlika je: ", usporedba(Prvi, Drugi))
```

```
Razlika je:  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).

```
# Ovo je mjesto na kojem možete izvoditi svoj kod.
import csv
tsv_file = open('/content/drive/My Drive/Colab Notebooks/ecoli_ILL_small_aln.sam')
# ucitavanje datoteke
tsv_rows = list(csv.reader(tsv_file, delimiter="\t"))

datoteka = list(tsv_rows[2])

# K je dobiveno prije kao pozicija s najboljim poravnanjem
print('Pozicija dobivena pomoću formule za korelaciju: ', K)
# u 4. stupcu se nalazi pozicija na kojoj je poravnanje najbolje
print("Pozicija dobivena iz datoteke: ", datoteka[3])
```

```
Pozicija dobivena pomoću formule za korelaciju:  2324486
Pozicija dobivena iz datoteke:  2324487
```

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.

```
# Ovo je mjesto na kojem možete izvoditi svoj kod.
```

```
from Bio.SeqIO import parse

FASTA = list(parse(fileFASTA, 'fasta'))
FASTQ = list(parse(fileFASTQ, 'fastq'))
zbroj = 0

referenca = [slovoBroj(s) for s in FASTA[0].seq]
referenca = [(x - avg) / std for x in referenca]
Referenca = np.fft.fft(padding1 + referenca)

#za prvih 100
for i in range(100):
  # procitaj i-to ocitanje
  ocitanje = [slovoBroj(j) for j in FASTQ[i].seq]
  # normalizacija i-tog ocitanja
  ocitanje = [(y - avg)/ std for y in ocitanje]
  # DFT od i-tog ocitanja (povecano nulama..)
  Ocitanje = np.fft.fft(ocitanje + padding2)
  Cor = np.conjugate(Ocitanje) * Referenca
  # i-ta korelacija
  cor = np.fft.ifft(Cor)
  # pozicija najvece slicnosti i-tog ocitanja reference
  k = k_arr[cor.argmax()]

  N = len(tsv_rows)
  for x in range(N):
    # lista s redcima i stupcima.. dvostruka
    if tsv_rows[x][0].startswith('@'):
      continue
    if tsv_rows[x][0] == FASTQ[i].id:
      #dvije pozicije se razlikuju za max 5 mjesta
      if abs(int(tsv_rows[x][3]) - k) <= 5:
        zbroj += 1

print ('Broj točno pozicioniranih očitanja: ', zbroj)
```

```
Broj točno pozicioniranih očitanja:  50
```

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.

✓ 6m 10s completed at 8:27 PM