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
  import numpy as np
  import Bio
  from scipy import signal
except ImportError:
  pass
```

```
print("mumpy version: " + np.__version__)
print("biopython version: " + Bio.__version__)
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: biopython in /usr/local/lib/python3.7/dist-packages (1.79)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from biopython) (1.21.6)
mumpy version: 1.21.6
biopython version: 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.

```
from google.colab import drive
drive.mount('/content/drive')
from Bio import SeqIO

referenca = list(SeqIO.parse("/content/drive/My Drive/Colab Notebooks/escherichia_coli_reference.fasta", "fasta"))
ocitanja  = list(SeqIO.parse("/content/drive/My Drive/Colab Notebooks/ecoli_ILL_small.fastq", "fastq"))

print("Broj elemenata u datoteci s referencom (FASTA file) je: " + str(len(referenca)) )
print("Broj elemenata u datoteci s očitanjima (FASTQ file) je: " + str(len(ocitanja)) )
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", forc
Broj elemenata u datoteci s referencom (FASTA file) je: 1
Broj elemenata u datoteci s očitanjima (FASTQ file) je: 38585
```

Double-click (or enter) to edit

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.

```
#print(str(referenca[0]) + "\n")
#print(str(ocitanja[0]) + "\n")

print("ID 1. očitanja:" + "\n" + ocitanja[0].id + '\n \n' + "Sekvenca 1. očitanja:" + "\n" + str(ocitanja[0].seq) + "\
print("prvih 200 znakova za referentni genom E.coli:" + "\n" + referenca[0].seq[:200])
```

```
ID 1. očitanja:
SRR2052522.671

Sekvenca 1. očitanja:
GATCTGGTGACCGGGTCGCGCAAAGTGATCATCGCCATGGAACATTGCGCCAAAGATGGTTCAGCAAAAATTTTGGGCCTCTGTATCATGCCACTCACTGCGCAATATCCGG

prvih 200 znakova za referentni genom E.coli:
AGCTTTTCATTCTGACTGCAACGGGCAATATGTCTCTGTGTGGATTAAAAAAAGAGTGTCTGATAGCAGCTTCTGAACTGGTTACCTGCCGTGAGTAAATTAAAATTTTATT
```

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 kodiranje(sekvenca):
  for i in sekvenca:
    if (i.upper() == "A"):
      return(3)
    elif (i.upper() == "G"):
      return(2)
    elif (i.upper() == "C"):
      return(-2)
    elif (i.upper() == "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.

```
ref_signal = list()
ocitanje_signal = list()

for slovo in referenca[0]:
```

```
  ref_signal.append(kodiranje(slovo))

for slovo in ocitanja[0]:
  ocitanje_signal.append(kodiranje(slovo))

# print(ref_signal[0:10])       --> provjera
# print(ocitanje_signal[0:10])  --> provjera

# vektor_korr = signal.correlate(ocitanje_signal, ref_signal, 'full', method='fft')
# print(korelacija)
# nadena metoda correlate i u scipy libraryju koja radi istu stvar

vektor_korr = np.correlate(ocitanje_signal, ref_signal, 'full')
print("Vektor korelacije: " + str(vektor_korr) )
```

```
Vektor korelacije: [ -4 -12  -9 ...  -1   2  -6]
```

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: " + str(len(referenca[0].seq) ) )
# moglo bi se računati i pomoću "ručnog" zbrajanja i dijeljenja, ali numpy library ima korisne metode mean i std nad a

a = list()

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

a = np.array(a)
# print(a)
print("Srednja vrijednost (mean, μ) duljine očitanja: " + str( a.mean() ) )
print("Standardna devijacija (σ) duljine očitanja: " + str( a.std() ) ) #?
```

```
Duljina reference: 4641652
Srednja vrijednost (mean, μ) duljine očitanja: 121.0
Standardna devijacija (σ) duljine 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.

```
B = []
d = dict()  # rječnik za prvih 10 ocitanja pretvorena u signale
i = 1

for oc in ocitanja[0:10]:
  A = [kodiranje(slovo) for slovo in oc.seq]
  d.update( {str(i): A} )
  i += 1
```

```
#print(d["1"])

for slovo in referenca[0].seq:
  B.append(kodiranje(slovo))

B = np.array(B)

B = np.fft.fft( np.pad( B, (120, 0), 'constant' ) )   # transformacija za referencu samo jednom

for i in range(10):
  # print( str(np.correlate(d[str(i+1)], B, "full")) ) --> isto tocno, ali transformacija za referencu n puta

  popunjen_signal = np.pad( d[str(i + 1)], (0, 4641651), 'constant' )
  A = np.fft.fft( popunjen_signal )

  C = np.conjugate( A ) * B
  C = np.fft.ifft(C)[::-1]  # iz nekog razloga je naopako

  print("korelacija za " + str(i+1) + ". očitanje: " +  str(C.real.round(2)) )
  print()
  if ( not i ):
    korelacija_za_1_ocitanje = C  # spremanje u varijablu za sljedeci zadatak
```

```
korelacija za 1. očitanje: [ -4. -12.  -9. ...  -1.   2.  -6.]

korelacija za 2. očitanje: [ -6. -13.  -9. ... -19.  -0.   9.]

korelacija za 3. očitanje: [ -4. -10. -18. ... -11. -13.  -6.]

korelacija za 4. očitanje: [-6. -5. -9. ... -6. -3.  9.]

korelacija za 5. očitanje: [ -6.  -5.   1. ... -16.  -0.   9.]

korelacija za 6. očitanje: [ -6. -13. -21. ...   9.  10.   6.]

korelacija za 7. očitanje: [ 4. -0. -9. ... 16.  5. -6.]

korelacija za 8. očitanje: [ 6.  5.  9. ... -7. 13.  6.]

korelacija za 9. očitanje: [ -6. -15. -12. ...  -9. -15.  -9.]

korelacija za 10. očitanje: [ -6.  -3.  -6. ...   6. -10.  -6.]
```

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_2_niza (niz1, niz2):
  max_len = max(len(niz1), len(niz2))
  zbroj = 0
  for i in range(max_len):
    if (niz1[i] != niz2[i]):
      zbroj += 1
  return zbroj
```

```
print("korelacija za 1. ocitanje: " + str(korelacija_za_1_ocitanje.real.round(2)) + "\n" )
print( "maksimalna vrijednost korelacije: " + str (max( korelacija_za_1_ocitanje.real.round(2) ) ) )

k = np.argmax(korelacija_za_1_ocitanje)
print("pozicija maksimalne korelacije: ", str(k) )

pocetak = len( korelacija_za_1_ocitanje ) - k - 121 # hardkodirano jer svako ocitanje je duljine 121
print("pocetna pozicija: ", str(pocetak) )
print("\n" + "- - - - - - - - - - - - - ispis apstrahiranih vrijednosti - - - - - - - - - - - - - - - - - -")


print( str(ocitanja[0].seq) + "\n" + str(referenca[0].seq[pocetak:pocetak+121] ) )
hammingova_udaljenost = hamming_2_niza( ocitanja[0].seq, referenca[0].seq[pocetak:121 + pocetak] )
print("Hammingova udaljenost najmanjih: ", str(hammingova_udaljenost) )
print("\n" + "- - - - - - - - - - - - - ispis ''originalnih signala''  - - - - - - - - - - - - - - - - - -")

B = []
for slovo in referenca[0].seq:
  B.append(kodiranje(slovo))

print( str(d["1"]) + "\n" + str( B[pocetak:121+pocetak] ) )
hammingova_udaljenost2 = hamming_2_niza( d["1"], B[pocetak:121+pocetak] )
print("Hammingova udaljenost najmanjih: ", str(hammingova_udaljenost) )
```

```
korelacija za 1. ocitanje: [ -4. -12.  -9. ...  -1.   2.  -6.]

maksimalna vrijednost korelacije: 693.0
pozicija maksimalne korelacije:  2317165
pocetna pozicija:  2324486

- - - - - - - - - - - - - ispis apstrahiranih vrijednosti - - - - - - - - - - - - - - - - - -
GATCTGGTGACCGGGTCGCGCAAAGTGATCATCGCCATGGAACATTGCGCCAAAGATGGTTCAGCAAAAATTTTGGGCCTCTGTATCATGCCACTCACTGCGCAATATCCGG
GATCTGGTGACCGGGTCGCGCAAAGTGATCATCGCCATGGAACATTGCGCCAAAGATGGTTCAGCAAAAATTTTGCGCCGCTGCACCATGCCACTCACTGCGCAACATGCGG
Hammingova udaljenost najmanjih:  9

- - - - - - - - - - - - - ispis ''originalnih signala''  - - - - - - - - - - - - - - - - - -
[2, 3, -3, -2, -3, 2, 2, -3, 2, 3, -2, -2, 2, 2, 2, -3, -2, 2, -2, 2, -2, 3, 3, 3, 2, -3, 2, 3, -3, -2, 3, -3, -
[2, 3, -3, -2, -3, 2, 2, -3, 2, 3, -2, -2, 2, 2, 2, -3, -2, 2, -2, 2, -2, 3, 3, 3, 2, -3, 2, 3, -3, -2, 3, -3, -
Hammingova udaljenost najmanjih:  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 nalazi 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("/content/drive/My Drive/Colab Notebooks/ecoli_ILL_small_aln.sam")
```

```
tsf_rows = list (csv.reader(tsv_file, delimiter="\t") )

prvi_stupac = dict()
cetvrti_stupac = dict()

broj = 0
for element in tsf_rows:
  if (element[0][0] != '@'):
    prvi_stupac.update({str(element[0]): broj })
    if (len(element) > 3):
      cetvrti_stupac.update({str(element[0]): element[3]})
  broj += 1

print("ID prvog ocitanja: " + str(ocitanja[0].id) + "\nRedak prvog očitanja u tsv datoteci: " + str(prvi_stupac[ocitan

print( "Pozicija max. poravnanja u tablici: " + str(cetvrti_stupac[ocitanja[0].id]) )
print( "Pozicij max. poravnanja prema izračunu: ", str(pocetak) )
print("- - - - - - - - - - - - - - - - - - - - - - - - - - - - - ")
print("Razlika pozicija je samo 1 mjesto")
```

```
ID prvog ocitanja: SRR2052522.671
Redak prvog očitanja u tsv datoteci: 2
Pozicija max. poravnanja u tablici: 2324487
Pozicij max. poravnanja prema izračunu:  2324486
- - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Razlika pozicija je samo 1 mjesto
```

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.

```
from Bio import pairwise2

print("Primjer uporabe libraryja: ")
for a in pairwise2.align.globalxx("ACCGT", "ACG"):
    print(pairwise2.format_alignment(*a))

print("- - - - - - - - - - - - - - - - - - - - - - - - - - - - - ")


for a in pairwise2.align.localxx(ocitanja[0].seq, referenca[0].seq[pocetak:121 + po
  print(pairwise2.format_alignment(*a))

for ocitanje in ocitanja[0:100] :
  print(ocitanje.id, len(C - np.argmax(C) - 121) )
```

```
SRR2052522.23349 4641772
SRR2052522.23911 4641772
SRR2052522.24191 4641772
SRR2052522.24597 4641772
SRR2052522.24840 4641772
SRR2052522.25270 4641772
SRR2052522.27349 4641772
SRR2052522.27706 4641772
SRR2052522.27894 4641772
SRR2052522.28063 4641772
SRR2052522.28640 4641772
SRR2052522.31407 4641772
SRR2052522.31832 4641772
SRR2052522.31934 4641772
SRR2052522.32057 4641772
SRR2052522.33263 4641772
SRR2052522.33290 4641772
SRR2052522.34810 4641772
SRR2052522.35251 4641772
SRR2052522.35421 4641772
SRR2052522.35654 4641772
SRR2052522.36181 4641772
SRR2052522.36517 4641772
SRR2052522.36536 4641772
SRR2052522.36563 4641772
SRR2052522.36890 4641772
SRR2052522.39237 4641772
SRR2052522.39281 4641772
SRR2052522.41456 4641772
SRR2052522.41719 4641772
SRR2052522.41879 4641772
SRR2052522.42159 4641772
SRR2052522.43596 4641772
SRR2052522.43779 4641772
SRR2052522.43918 4641772
SRR2052522.44065 4641772
SRR2052522.44668 4641772
SRR2052522.47508 4641772
SRR2052522.48239 4641772
SRR2052522.48346 4641772
SRR2052522.48700 4641772
SRR2052522.49210 4641772
SRR2052522.49984 4641772
SRR2052522.50490 4641772
SRR2052522.51677 4641772
SRR2052522.52344 4641772
SRR2052522.52571 4641772
SRR2052522.53437 4641772
SRR2052522.53469 4641772
SRR2052522.53847 4641772
SRR2052522.53882 4641772
SRR2052522.54155 4641772
SRR2052522.54286 4641772
SRR2052522.54371 4641772
SRR2052522.55060 4641772
SRR2052522.55158 4641772
```

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.

✓ 6s completed at 23:59