BLAST nədir və necə işləyir?

Bu proqram bir DNT və ya protein ardıcıllığını digər tanınmış ardıcıllıqlarla müqayisə etməyə imkan verir. Yəni bir genin və ya proteinin hansı orqanizmlərdə olduğunu və ya nə ilə oxşar olduğunu öyrənmək üçün çox faydalıdır.

Amma BLAST-ın bir çətinliyi var: çoxlu məlumat çıxır və bu məlumatlarla işləmək, analiz aparmaq və BLAST-ı avtomatik şəkildə (məsələn, skriptlərlə) işlətmək çətin ola bilər.

Burada Biopython köməyə gəlir. Biopython ilə BLAST-ı həm işlədə, həm də nəticələri oxuyub analiz edə bilərik.

BLAST-ı necə işlətmək olar?

BLAST-ı iki şəkildə:

1.Lokal olaraq – BLAST proqramı sizin kompüterinizdə işləyir.

2.Uzaqdan (online) – NCBI-nin (Amerikadakı bioloji məlumat mərkəzi) serverləri vasitəsilə işləyir.

Python ilə internet üzərindən BLAST işlətmək

Biopython kitabxanasında **qblast** adlı bir funksiya var. Bu funksiya ilə bir ardıcıllığı NCBI BLAST serverinə göndərib nəticələri geri ala bilərsiniz.

Amma NCBI bəzi **qaydalar** qoyub:

•Serverə hər 10 saniyədən tez sorğu göndərmək olmaz.

•Hər hansı bir sorğunun nəticəsini hər 1 dəqiqədən tez yoxlamaq olmaz.

•Əgər çoxlu sorğu göndərirsinizsə, email ünvanınızı göstərməlisiniz ki, problem olsa, sizinlə əlaqə saxlaya bilsinlər.

•50-dən çox sorğu göndərəcəksinizsə, bu işləri ya gecə, ya da həftə sonu edin.

Biopython-un **qblast** funksiyası bu qaydalara avtomatik əməl edir.

Onlayn BLAST versiyasını çağırmaq üçün Biopython-un Bio.Blast modulundakı qblast funksiyasından istifadə edirik.

```
from Bio import Blast
Blast.tool
'biopython'
Blast.email = "ulviyyemag@gmail.com"
```

qblast funksiyasının əsas arqumentləri

qblast funksiyası ilə internet üzərindən BLAST işlətmək üçün **3 əsas (vacib) arqument** mütləq verilməlidir:

1. İstifadə ediləcək BLAST proqramı

Bu, aşağıdakılardan biri olmalıdır və **kiçik hərflərlə** yazılır:

•"blastn" – DNT ardıcıllığına qarşı axtarış

•"blastp" – Protein ardıcıllığına qarşı axtarış

•"blastx" – DNT-ni proteinə tərcümə edib protein verilənlər bazası ilə müqayisə edir

•"tblastn" – Proteini DNT verilənlər bazasında axtarır

•"tblastx" – Həm sorğunu, həm də verilənlər bazasını proteinə tərcümə edib müqayisə edir.

2. Verilənlər bazası (database)

Hansı verilənlər bazasında axtarış etmək istəyirsinizsə, onun adını yazırsınız. Məsələn:

•"nt" – DNT verilənlər bazası

•"nr" – protein verilənlər bazası

•Digər verilənlər bazalarının adlarını NCBI-nin saytından tapa bilərik.

3. Sorğu ardıcıllığı (query sequence)

Bu, sizin axtarmaq istədiyiniz gen və ya protein ardıcıllığıdır. Bu formada ola bilər:

•Düz ardıcıllıq (məsələn: "ATGCGT...")

•FASTA formatında

•Və ya bir identifikasiya nömrəsi (məsələn: GI nömrəsi)

Əlavə (istəyə bağlı) parametrlər

Bundan əlavə, aşağıdakı **istəyə bağlı parametr**ləri də istifadə edə bilərsiniz:

url_base

İnternet üzərindən BLAST harada işlədiləcək. Adətən bu dəyişdirilmir, çünki standart olaraq NCBI serverlərinə bağlanır.

format_type

Nəticələrin hansı formatda alınacağını göstərir. Seçimlər:

•"XML" (standartdır və sonradan təhlil üçün ən uyğun formadır)

•"HTML" (brauzer üçün)

•"Text" (adi mətn formatı)

•"Tabular" (cədvəl şəklində)

•"JSON2" və "XML2" – daha yeni formatlardır

expect

Bu parametr **e-value** limitini təyin edir (BLAST-da uyğunluq ehtimalını göstərir). Kiçik dəyər qoysanız, nəticələr daha dəqiq olar.

Məsələn:

•expect=0.001 – Yalnız çox uyğun olan nəticələr çıxacaq.

E-value nədir və niyə vacibdir?

•**Tərifi:** E-value bir uyğunluğun (alignment-in) **təsadüfi olaraq verilən verilənlər bazasında neçə dəfə baş verə biləcəyini** göstərir.

Yəni, bu uyğunluğun sadəcə təsadüf nəticəsində tapılma ehtimalını göstərir.

•**Kiçik dəyər → daha əhəmiyyətli nəticə:**

- Məsələn, əgər **e-value = 1**, bu o deməkdir ki, verilənlər bazasında bu cür uyğunluğun **təsadüfən 1 dəfə** baş vermə ehtimalı var.
- Əgər **e-value = 1e-5 (0.00001)** kimi çox kiçik bir dəyərdirsə, bu o deməkdir ki, bu uyğunluq çox güman ki, **təsadüfi deyil və əsl uyğunluqdur**.

BLAST proqramında **gap** anlayışı da çox mühüm parametrlərdəndir. Bu, iki ardıcıllıq (sequence) arasında uyğunluq (alignment) qurularkən **boşluqların** (nukleotid və ya amin turşusu olmayan yerlərin) necə idarə olunacağını təyin edir.

Əgər qblast funksiyasının daha çox parametrləri ilə maraqlanırsınızsa, NCBI-nin rəsmi sənədlərinə və ya Biopython-un daxili kömək sisteminə baxa bilərsiniz:

```
[7]: from Bio import Blast
     help(Blast.qblast)
```

```
Help on function qblast in module Bio.Blast:

qblast(program, database, sequence, url_base='https://blast.ncbi.nlm.nih.gov/Blast.cgi', auto_format=None, composition_based_statistics=None, db_genetic
_code=None, endpoints=None, entrez_query='(none)', expect=10.0, filter=None, gapcosts=None, genetic_code=None, hitlist_size=50, i_thresh=None, layout=No
ne, lcase_mask=None, matrix_name=None, nucl_penalty=None, nucl_reward=None, other_advanced=None, perc_ident=None, phi_pattern=None, query_file=None, que
ry_believe_defline=None, query_from=None, query_to=None, searchsp_eff=None, service=None, threshold=None, ungapped_alignment=None, word_size=None, short
_query=None, alignments=500, alignment_view=None, descriptions=500, entrez_links_new_window=None, expect_low=None, expect_high=None, format_entrez_query
=None, format_object=None, format_type='XML', ncbi_gi=None, results_file=None, show_overview=None, megablast=None, template_type=None, template_length=N
one, username='blast', password=None)
    BLAST search using NCBI's QBLAST server.

    Supports all parameters of the old qblast API for Put and Get.

    Please note that NCBI uses the new Common URL API for BLAST searches on
    the internet (https://blast.ncbi.nlm.nih.gov/doc/blast-help/urlapi.html).
    Thus, some of the parameters used by this function are not (or are no
    longer) officially supported by NCBI. Although they are still functioning,
    this may change in the future.

    Some useful parameters:

     - program        blastn, blastp, blastx, tblastn, or tblastx (lower case)
     - database       Which database to search against (e.g. "nr").
     - sequence       The sequence to search.
     - ncbi_gi        TRUE/FALSE whether to give 'gi' identifier.
     - descriptions   Number of descriptions to show.  Def 500.
     - alignments     Number of alignments to show.  Def 500.
     - expect         An expect value cutoff.  Def 10.0.
     - matrix_name    Specify an alt. matrix (PAM30, PAM70, BLOSUM80, BLOSUM45).
     - filter         "none" turns off filtering.  Default no filtering
     - format_type    "XML" (default), "HTML", "Text", "XML2", "JSON2",
                      or "Tabular".
     - entrez_query   Entrez query to limit Blast search - only applies when searching nucleotide BLASTDBs
```

Qeyd: NCBI-nin veb saytındakı BLAST ayarları ilə Biopython-un qblast funksiyasının standart ayarları **eyni deyil**. Əgər fərqli nəticələr alırsınızsa, o zaman **parametrləri (məsələn e-value və gap dəyərlərini)** yoxlamaq vacibdir.

Nümunə: BLASTN ilə nt verilənlər bazasında axtarış

GI nömrəsi ilə axtarış

Əgər sizdə axtaracağınız ardıcıllığın GI nömrəsi varsa (məsələn: 8332116), bu şəkildə istifadə edə bilərsiniz:

```
*]: from Bio import Blast
    result_stream = Blast.qblast("blastn", "nt", "8332116")
```

FASTA faylından ardıcıllığı oxumaq (sətir şəklində)

Əgər ardıcıllığınız bir .fasta faylındadırsa (məsələn: m_cold.fasta), əvvəlcə faylı oxuyub onu qblast funksiyasına verə bilərsiniz:

```
from Bio import Blast
fasta_string = open("C:\\Users\\Acer\\Downloads\\m_cold.fasta").read()
result_stream = Blast.qblast("blastn", "nt", fasta_string)
```

FASTA faylını SeqRecord kimi oxumaq və yalnız ardıcıllığı göndərmək

Əgər ardıcıllığı SeqRecord formatında oxuyursunuzsa və yalnız ardıcıllığın özünü göndərmək istəyirsinizsə:

```
[*]: from Bio import Blast
     from Bio import SeqIO
     record = SeqIO.read("C:\\Users\\Acer\\Downloads\\m_cold.fasta", "fasta")
     result_stream = Blast.qblast("blastn", "nt", record.seq)
```

Bu zaman BLAST sizin üçün avtomatik olaraq bir ID (identifikator) təyin edəcək.

SeqRecord-dan FASTA formatlı sətir yaratmaq

Əgər ardıcıllıq başqa bir formatdadır (məsələn, .gbk – GenBank faylı), bu zaman onu FASTA formatına çevirmək daha məntiqlidir:

```
[*]: from Bio import Blast
     from Bio import SeqIO
     records = SeqIO.parse("ls_orchid.gbk", "genbank")
     record = next(records)
     result_stream = Blast.qblast("blastn", "nt", format(record, "fasta"))
```

Bu üsul, FASTA olmayan fayllardakı ardıcıllıqları BLAST-da istifadə etmək üçün çox uyğundur.

Əgər ardıcıllığınız (**sequence**) FASTA formatında **deyilsə**, yəni məsələn .gbk (GenBank), .embl və s. kimi başqa bir formatdadırsa, **Bio.SeqIO** modulundan istifadə edərək bu fayldakı ardıcıllığı çıxarmaq daha uyğun olur.

Bu halda, həmin ardıcıllığı format(record, "fasta") ilə FASTA formatına çevirib **BLAST** axtarışı üçün istifadə edə bilərsiniz.

BLAST nəticələri internet üzərindən gəlir və **bir dəfə oxunduqdan sonra** ikinci dəfə istifadə oluna bilmir. Məsələn:

```
[27]: result_stream.read()  # Sadəcə bir dəfə oxuna bilər
```

Əgər bu nəticəni **təkrar-təkrar istifadə etmək** istəyirsinizsə, fayl şəklində saxlayın:

```
[*]: with open("my_blast.xml", "wb") as out_stream:
         out_stream.write(result_stream.read())

     result_stream.close()
```

Bu kod **my_blast.xml** adlı bir fayl yaradacaq və nəticəni ora yazacaq.

İstədiyiniz zaman faylı yenidən aça bilərsiniz:

```
[35]: result_stream = open("my_blast.xml", "rb")
```

İndi bu axın (result_stream) istifadə olunmağa hazırdır – məsələn, **parsinq** (yəni analiz etmək) üçün.

result_stream = NCBIWWW.qblast(...) sətri bizə BLAST nəticələrini bir axın (stream) şəklində verir. Bu nəticələr **XML formatında** olur.

Bu XML faylını biz sadəcə mətni kimi oxuya bilərik, amma analiz etmək (yəni "parsinq" etmək) üçün onu Bio.Blast.NCBIXML modulu ilə **strukturlaşdırılmış məlumat kimi oxumalıyıq**.

Bu, sadəcə XML mətni kimi oxumaqdır (parsinq etmədən):

```
print(result_stream.read())
```

Bu isə **parsinq edilmiş**, yəni BLAST nəticələrini **strukturlaşdırılmış obyekt kimi oxumaq** və üzərində işləməkdir:

```
from Bio.Blast import NCBIXML

# Fayldan və ya stream-dən parsinq
blast_record = NCBIXML.read(result_stream)

# Parsinqdən sonra məlumatı analiz edə bilərik:
for alignment in blast_record.alignments:
    print("Hit title:", alignment.title)
    for hsp in alignment.hsps:
        print("E-value:", hsp.expect)
```

•result_stream = BLAST nəticələri var, amma hələ **xam** (strukturlaşdırılmamış) haldadır.

•Parsinq = onu oxumaq və **anlayışlı, istifadəyə yararlı hala** salmaq.

•Parsinqdən sonra: nəticə üzərində analiz, süzgəc, statistik çıxarışlar və s. edə bilərsiniz**.**

BLAST nəticələrinin başqa formatlarda alınması

XML-dən əlavə JSON və ya TXT kimi formatlar da mümkündür

qblast funksiyasına format_type arqumentini əlavə etsəniz, fərqli nəticə formatları ala bilərsiniz.

Məsələn: JSON formatında nəticə almaq

```
from Bio import Blast
from Bio import SeqIO

record = SeqIO.read("m_cold.fasta", "fasta")
result_stream = Blast.qblast("blastn", "nt", record.seq, format_type="JSON2")
data = result_stream.read()
```

Bu nəticə **ZIP formatında** olacaq (sıxılmış fayl).

```
with open("myzipfile.zip", "wb") as out_stream:
    out_stream.write(data)
```

ZIP faylını açmaq və məlumatı oxumaq

```
import zipfile

myzipfile = zipfile.ZipFile("myzipfile.zip")
print(myzipfile.namelist())
```

Bu ZIP fayl içində olan faylların adlarını göstərəcək, məsələn:

```
['N5KN7UMJ013.json', 'N5KN7UMJ013_1.json']
```

Faylı aç və oxu:

```
stream = myzipfile.open("N5KN7UMJ013_1.json")
data = stream.read()
data = data.decode()  # Baytları mətnə çevir
print(data)  # JSON formatında nəticə
```

JSON (JavaScript Object Notation) məlumatını Python obyektinə çevirmək

```python
import json

d = json.loads(data)
print(d["BlastOutput2"]["report"]["program"])  # Məsələn: blastn
```

Bu sayədə **nəticələrə istədiyin kimi analiz** edə, proqramlaşdırma vasitəsilə filtrləmə və hesablama işləri apara bilərik.

```
>>> import json
>>> d = json.loads(data)
>>> print(d)
{'BlastOutput2': {'report': {'program': 'blastn', 'version': 'BLASTN 2.14.1+',
 'reference': 'Stephen F. Altschul, Thomas L. Madden, Alejandro A. Sch&auml;ffer,
 Jinghui Zhang, Zheng Zhang, Webb Miller, and David J. Lipman (1997),
 "Gapped BLAST and PSI-BLAST: a new generation of protein database search programs",
 Nucleic Acids Res. 25:3389-3402.',
 'search_target': {'db': 'nt'}, 'params': {'expect': 10, 'sc_match': 2,
 'sc_mismatch': -3, 'gap_open': 5, 'gap_extend': 2, 'filter': 'L;m;'},
 'results': {'search': {'query_id': 'Query_128889', 'query_len': 1111,
 'query_masking': [{'from': 797, 'to': 1110}], 'hits': [{'num': 1,
 'description': [{'id': 'gi|1219041180|ref|XM_021875076.1|', 'accession':
 'XM_021875076', 'title':
 'PREDICTED: Chenopodium quinoa cold-regulated 413 plasma membrane protein 2-like (LOC110697660), mRNA',
 'taxid': 63459, 'sciname': 'Chenopodium quinoa'}], 'len': 1173, 'hsps':
 [{'num': 1, 'bit_score': 435.898, 'score': 482, 'evalue': 9.02832e-117,
 'identity': 473, 'query_from'
...
```

BLAST-ı Lokalda İşə Salmaq

BLAST analizlərini internet üzərindən işlətmək mümkün olduğu kimi, kompüterində **lokal olaraq** da BLAST işlədə bilərik. BLAST-ı lokalda işlətməyin iki əsas üstünlüyü var:

1.Daha sürətlidir: BLAST sorğuları birbaşa kompüterində aparıldığı üçün internet üzərindən işləməyə nisbətən daha sürətli nəticə verir.

2.Öz şəxsi məlumat bazamızı yarada bilərik: Özümüz topladığımız və ya istifadə etdiyimiz xüsusi ardıcıllıqlarla BLAST axtarışları apara bilərik.

Dezavantajları

- BLAST-ı lokalda işlətmək üçün **komanda sətri proqramları** qurulmalıdır.
- Həmçinin BLAST üçün böyük ölçülü **verilənlər bazası** (məsələn, NR – non-redundant database) yüklənməli və qurulmalıdır.
- Bu verilənlər bazalarının **vaxtaşırı yenilənməsi** tövsiyə olunur.

NCBI BLAST+ (Yeni Nəsil BLAST Alətləri)

2009-cu ildən etibarən NCBI köhnə BLAST proqramlarının yerinə **BLAST+** adlı yeni nəsil proqram paketini istifadəyə verib. Bu proqramlar daha sürətli işləyir və daha çevikdir.

Python dili vasitəsilə bu alətləri işlətmək də olduqca asandır. Əsas ideya belədir: əvvəlcə komanda sətri üçün bir əmr cümləsi tərtib olunur, sonra isə bu əmr Python ilə işə salınır.

Nümunə: BLASTX ilə Lokal Axtarış

Tutaq ki, opuntia.fasta adlı bir faylda genetik nukleotid ardıcıllıqlarımız var. Bu ardıcıllıqları **BLASTX** aləti ilə **NR (protein) verilənlər bazasında** axtarmaq istəyirik. Əgər sistemə NR verilənlər bazası öncədən yüklənibsə, bu komandanı terminalda yazırıq:

```
blastx -query opuntia.fasta -db nr -out opuntia.xml -evalue 0.001 -outfmt 5
```

Bu əmr:

- opuntia.fasta faylındakı ardıcıllıqları
- nr adlı protein bazası ilə müqayisə edir
- Nəticələri opuntia.xml adlı XML faylına yazır
- E-value (gözlənilən uyğunluq) dəyəri kimi 0.001 təyin edir
- Nəticə formatı olaraq 5 nömrəli XML çıxışını istifadə edir

Bu əməliyyat bir neçə dəqiqə çəkə bilər. Amma nəticələr fayla yazıldığı üçün, lazım olsa təkrar analiz edə bilərik, və prosesi yenidən başlatmağa ehtiyac qalmaz.

Python ilə BLAST+ Əmrini İşlətmək

Yuxarıdakı əmri Python dilində subprocess modulu ilə işlədə bilərsən:

```
import subprocess

cmd = "blastx -query opuntia.fasta -db nr -out opuntia.xml"
cmd += " -evalue 0.001 -outfmt 5"
subprocess.run(cmd, shell=True)
```

Əgər əməliyyat uğurla icra olunubsa, opuntia.xml adlı bir fayl yaradılacaq. Bu fayl daha sonra Biopython kitabxanasının Bio.Blast.parse() funksiyası ilə analiz edilə bilər.

Digər BLAST Versiyaları

BLAST+ ilə yanaşı, əvvəlki illərdə istifadə olunan başqa BLAST proqramları da mövcuddur:

- **NCBI Legacy BLAST:** Köhnə BLAST versiyasıdır, artıq yenilənmir.
- **WU-BLAST (Washington Universiteti):** Alternativ BLAST versiyasıdır.
- **AB-BLAST (Advanced Biocomputing):** 2009-cu ildən istifadədədir, lakin pulsuz deyil.

Biopython bu köhnə alətləri birbaşa dəstəkləmir, amma əgər bu proqramların çıxış faylları NCBI standartına uyğundursa, onları da təhlil etmək mümkündür.

BLAST çıxışının təhlili

BLAST müxtəlif formatlarda nəticə verə bilər: XML, HTML və adi mətn (plain text). Əvvəllər Biopython yalnız BLAST-ın mətn və HTML formatlarını təhlil edə bilirdi, çünki o vaxtlar yalnız bu formatlar mövcud idi. Lakin, bu formatlar BLAST tərəfindən tez-tez dəyişdirildiyi üçün Biopython bu təhlil funksiyalarını silib.

Hazırda Biopython yalnız **XML**, **XML2** və **cədvəl (tabular)** formatlı BLAST nəticələrini təhlil edə bilir. Biz indi **Bio.Blast.parse** funksiyası vasitəsilə **XML** və **XML2** formatlarının necə təhlil edildiyi izah olunur. Funksiya avtomatik olaraq XML faylının hansı formatda olduğunu müəyyən edir.

Tabular formatdakı BLAST nəticələrini isə **Bio.Align.parse** funksiyası ilə təhlil etmək olar (bax: *Tabular output from BLAST or FASTA* bölməsi).

BLAST nəticələrini XML formatında necə əldə etmək olar?

- **İnternet üzərindən Biopython ilə BLAST işlətmək**
- **Biopython ilə lokal BLAST işlətmək**
- **BLAST nəticələrini NCBI saytından əldə edib XML formatında yadda saxlamaq**
- **Lokal olaraq BLAST işlədib nəticəni XML faylı şəklində saxlamaq**

Əsas odur ki, nəticə XML formatında olsun – necə əldə edilməsindən asılı olmayaraq, Biopython onu təhlil edə bilər.

XML formatlı BLAST faylını necə təhlil etməli?
Əgər nəticə axını (file-like object) varsa:

```
from Bio import Blast
result_stream = Blast.qblast("blastn", "nt", "8332116")
```

Yaxud lokal fayldan oxumaq üçün:

```
result_stream = open("my_blast.xml", "rb")
```

Tək sorğu üçün:

```
from Bio import Blast
blast_record = Blast.read(result_stream)
```

Çoxlu sorğular üçün:

```
from Bio import Blast
blast_records = Blast.parse(result_stream)
```

Iterator vasitəsilə məlumatı bir-bir oxumaq:

```
for blast_record in blast_records:
    # burada nəticələrlə istədiyiniz kimi işləyə bilərsiniz
```

Əgər nəticələrin hamısını yadda saxlamaq istəyirsinizsə:

```
blast_records = list(blast_records)
```

Yadda saxlayandan sonra indekslə işləmək mümkündür:

```
print(blast_records[2].query.description)
```

Faylı birbaşa fayl adı ilə oxumaq:

```
from Bio import Blast
with Blast.parse("my_blast.xml") as blast_records:
    for blast_record in blast_records:
        pass  # burada məlumatı emal edə bilərsiniz
```

Bu üsul faylı avtomatik açır və iş bitdikdən sonra bağlayır.

BLAST nəticələrinin xülasəsini çap etmək:

```
from Bio import Blast
with Blast.parse("my_blast.xml") as blast_records:
    print(blast_records)
```

```
Program: BLASTN 2.2.27+
     db: refseq_rna

  Query: 42291 (length=61)
         mystery_seq
   Hits: ----  -----  ------------------------------------------------------------
           #  # HSP  ID + description
         ----  -----  ------------------------------------------------------------
           0      1  gi|262205317|ref|NR_030195.1|  Homo sapiens microRNA 52...
           1      1  gi|301171311|ref|NR_035856.1|  Pan troglodytes microRNA...
           ...
```

BLAST Records, Record və Hit sinifləri

BLAST Records sinfi nədir?

BLAST nəticə faylı bir neçə BLAST sorğusunun nəticələrini ehtiva edə bilər. Biopython-da bu məlumatlar **Bio.Blast.Records** obyektində saxlanılır. Bu obyekt bir iterator (döngü ilə irəliləyən obyekt) kimi işləyir və hər BLAST sorğusu üçün bir Bio.Blast.Record obyekti verir.

Əsas atributlar:

•**source:** BLAST faylının haradan yükləndiyini göstərir (faylın adı və ya fayl obyekti).

•**program:** İstifadə olunan BLAST proqramı (məsələn, blastn).

•**version:** BLAST proqramının versiyası (məsələn, BLASTN 2.2.27+).

•**reference:** Ədəbiyyat istinadı (məsələn, Altschul və b. 1997).

•**db:** Sorğunun qarşılaşdırıldığı verilənlər bazası (məsələn, refseq_rna).

•**query:** SeqRecord obyekti — sorğuya aid məlumat (id, təsvir, ardıcıllıq).

•**param:** BLAST parametrlərinin saxlandığı lüğət (məsələn, expect, sc-match, gap-open və s.).

•**mbstat:** Mega BLAST statistik məlumatları (yalnız köhnə versiyalarda olur).

BLAST Record sinfi nədir?

Bio.Blast.Record obyekti bir sorğu üçün BLAST nəticələrini təmsil edir. Bu sinif, əslində **Hit** obyektlərindən ibarət bir siyahıdır və **list** sinifindən miras alır.

Əsas atributlar:

•**query:** BLAST sorğusunun SeqRecord obyekti.

•**stat:** Statistik göstəricilərdən ibarət lüğət (məsələn, db-num, db-len, kappa, lambda, entropy).

Hit sinfi nədir?

Hər bir **Hit** obyekti bir uyğunluğu göstərir — yəni BLAST sorğusuna uyğun gələn bir sekans. Bu obyektlər **blast_record** siyahısının elementləridir.

BLAST nəticələrinin işlənməsi (misallarla):

```
for hit in blast_record:
    print(hit.target.id)
```

Yuxarıdakı kod sorğu üçün uyğun gələn bütün hitləri çap edir.

```
len(blast_record)
```

Bu kod nəticədə neçə hit olduğunu göstərir.

```
blast_record[0]  # birinci hit
blast_record[-1]  # sonuncu hit
```

```
"gi|262205317|ref|NR_030195.1|" in blast_record  # bu ID-li hit mövcuddurmu?
```

```
blast_record.index("gi|301171437|ref|NR_035870.1|")  # bu hitin sıralamadakı yeri
```

BLAST (Basic Local Alignment Search Tool) proqramında **"hit"** dedikdə, **bizim verdiyimiz sorğu ardıcıllığına (query sequence) verilənlər bazasında oxşarlıq göstərən bir seqment (və ya sekans)** nəzərdə tutulur.

Hit dedikdə nə daxildir?

Hər bir **hit** içində aşağıdakı məlumatlar olur:

Hissə	İzahı
ID	Tapılan uyğun ardıcıllığın identifikatoru
Title	Tapılan ardıcıllığın adı və təsviri
Alignments (HSP)	Sənin ardıcıllığınla tapılan ardıcıllığın hansı hissələri uyğun gəlir
Score	Uyğunluğun "gücü" (nə qədər yaxşı uyğun gəlir)
E-value	Təsadüfi uyğunluq ehtimalı (kiçik olsa, daha yaxşıdır)

Məsələn:

Sən ACTGGTCA... ardıcıllığını axtarırıq.

BLAST:

•Verilənlər bazasında NC_000001.11 Homo sapiens chromosome 1 adlı bir DNT hissəsi **sənin ardıcıllığınla 98% oxşarlıq göstərir**.

Bu nəticə **bir "hit"dir.**

Əgər başqa bir DNT hissəsi də uyğun gəlsə — bu da **ikinci "hit"** olacaq.

BLAST Hit sinfi nədir?

BLAST proqramı ilə analiz etdikdən sonra əldə etdiyimiz nəticələrdə hər uyğunluq (yəni uyğun gələn sekanslar) bir **"Hit"** olaraq qeyd olunur. Python kitabxanasında bu nəticələr **Bio.Blast.Hit** obyektləri ilə təmsil olunur.

hit = blast_record[0] nədir?

Bu, BLAST nəticələri siyahısının (blast_record) **ilk uyğunluğunu** (yəni ilk tapılan uyğun sekansı) seçir.

hit.target nəyi göstərir?

Bu, **tapılmış uyğun hədəf sekansı** göstərir. Məsələn:

```
SeqRecord(seq=Seq(None, length=61), id='gi|262205317|ref|NR_030195.1|', name='NR_030195', description='Homo sapiens microRNA 520b (MIR520B), microRNA')
```

Bu o deməkdir ki, tapılan uyğun seqans **insan mikroRNA 520b**-dir və uzunluğu 61 nukleotiddir.

HSP (High-scoring Segment Pair) nədir?

HSP - BLAST tərəfindən müəyyən edilən uyğunluq bölgəsidir. Yəni sorğu seqansı ilə hədəf seqans arasında yüksək uyğunluğu olan hissədir.

Məsələn:

E-value	Bit score	Span	Query range	Hit range
8.9e-20	100.47	60	[1:61]	[13:73]

Burada:

•**E-value**: Nə qədər təsadüfi ola biləcəyini göstərir. Kiçik olduqca, uyğunluq **əhəmiyyətlidir**.

•**Bit score**: Uyğunluğun keyfiyyətini göstərir.

•**Query range və Hit range**: Uyğunluğun başladığı və bitdiyi yerləri göstərir (istifadəçi və verilənlər bazasında olan sekansda).

Bir "Hit" içində neçə uyğunluq ola bilər?

•Bir hit bir və ya bir neçə uyğunluğu (HSP-ləri) özündə saxlayır.

•Məsələn: bəzi hallarda **bir genetik sekans** müxtəlif xromosom bölgələrinə uyğun gələ bilər – belə olan halda bir hit içində bir neçə Alignment olur.

alignment obyektləri və xüsusiyyətləri

```
alignment = hit[0]
```

Bu, ilk alignment-i (uyğunluğu) göstərir. alignment obyekti Bio.Blast.HSP sinfindəndir və aşağıdakı məlumatları saxlayır:

•**alignment.target:** uyğun gələn hədəf sekansı (bazadan gələn)

•**alignment.query:** istifadəçi tərəfindən sorğu verilən sekans

•**alignment.coordinates:** uyğunluğun başlanğıc və son nöqtələri

Filtrasiya nümunəsi:

```
E_VALUE_THRESH = 0.04
for alignments in blast_record:
    for alignment in alignments:
        if alignment.annotations["evalue"] < E_VALUE_THRESH:
            print("****Alignment****")
            print("sequence:", alignment.target.id)
            print("score:", alignment.score)
            print("e value:", alignment.annotations["evalue"])
```

Bu kod parçacığı E-value'si **0.04-dən az olan** bütün uyğunluqları siyahıya alır və məlumatlarını çap edir.

BLAST qeydlərinin yazılması

Bio.Blast modulunda **write** funksiyasından istifadə edərək BLAST qeydlərini XML faylı kimi yadda saxlaya bilərik. Standart olaraq (DTD-əsaslı) XML formatı istifadə olunur; write funksiyasına fmt="XML2" arqumentini əlavə etməklə qeydləri (sxem-əsaslı) XML2 formatında da saxlamaq mümkündür.

```
from Bio import Blast
stream = Blast.qblast("blastn", "nt", "8332116")
records = Blast.parse(stream)
Blast.write(records, "my_qblast_output.xml")
```

və ya

```
Blast.write(records, "my_qblast_output.xml", fmt="XML2")
```

Bu nümunədə **Blast.qblast** funksiyasından alınan məlumatı birbaşa XML faylına da yazmaq olar. Lakin **qblast** funksiyasından əldə olunan məlumatı qeydlərə çevirməklə, onları yazmazdan əvvəl çeşidləyə və ya filtrləyə bilərik. Məsələn, yalnız müsbət balı 400-dən yüksək olan BLAST HSP-ləri ilə maraqlana bilərik:

```
filter_func = lambda hsp: hsp.annotations["positive"] >= 400
for hit in records[0]:
    hit[:] = filter(filter_func, hit)

Blast.write(records, "my_qblast_output_selected.xml")
```

Blast.write funksiyasının ikinci arqumenti olaraq fayl adından başqa fayl axını da istifadə etmək olar. Bu halda, axın yazmaq üçün binary formatda açılmalıdır:

```
with open("my_qblast_output.xml", "wb") as stream:
    Blast.write(records, stream)
```

PSI-BLAST ilə işləmək

PSI-BLAST proqramını (psiblast) birbaşa əmr sətrindən və ya Python-un subprocess modulundan istifadə edərək işə sala bilərsiniz.

Hazırda, NCBI tərəfindən PSI-BLAST axtarışlarının internet üzərindən aparılması dəstəklənmir.

Qeyd edək ki, **Bio.Blast parser-i PSI-BLAST-ın** mövcud versiyalarının XML çıxışlarını oxuya bilir, lakin hər iterasiyada hansı ardıcıllığın yeni və ya təkrar istifadə edildiyi kimi məlumat XML faylında yer almır.

RPS-BLAST ilə işləmək

RPS-BLAST proqramını (rpsblast) da birbaşa əmr sətrindən və ya subprocess modulundan istifadə edərək işə sala bilərik.

Hazırda, NCBI tərəfindən RPS-BLAST axtarışlarının internet üzərindən aparılması da dəstəklənmir.

Bio.Blast parser-i RPS-BLAST-ın mövcud versiyalarının XML çıxışlarını oxuya bilir.

Qeyd: Aşağıdakı fəsil – *BLAST və digər ardıcıllıq axtarış alətləri* – Bio.SearchIO modulunu təsvir edir. Bu modulun gələcəkdə Bio.Blast modulunun yerini alması nəzərdə tutulur, çünki daha ümumi bir çərçivə təqdim edir və digər oxşar ardıcıllıq axtarış alətləri ilə işləmək imkanı verir. Lakin hələlik həm Bio.SearchIO, həm də köhnə Bio.Blast modulu ilə NCBI BLAST nəticələri üzərində işləyə bilərik.

İnternet üzərindən BLAST işlətmək

İnternet üzərindən BLAST işlətmək üçün Bio.Blast.NCBIWWW modulundakı qblast() funksiyasından istifadə olunur. Bu funksiyanın üç mütləq arqumenti var:

1.İlk arqument: Axtarış üçün istifadə olunacaq BLAST proqramının adı (kiçik hərflərlə yazılır). Mövcud seçimlər: blastn, blastp, blastx, tblast, tblastx.

Daha ətraflı məlumat üçün: https://blast.ncbi.nlm.nih.gov/Blast.cgi

2.İkinci arqument: Axtarışın ediləcəyi verilənlər bazası. Məsələn, nt (nükleotid verilənlər bazası). Ətraflı: https://blast.ncbi.nlm.nih.gov/doc/blast-help/

3.Üçüncü arqument: Sorğu ardıcıllığınız (nümunəniz) – bu ardıcıllığın özü, FASTA formatında ardıcıllıq və ya GI nömrəsi ola bilər.

NCBI qaydaları (https://blast.ncbi.nlm.nih.gov/doc/blast-help/developerinfo.html#developerinfo):

•Serverə 10 saniyədən tez-tez müraciət etmirik.

•Eyni RID üçün sorğu 1 dəqiqədən tez-tez edilməməlidir.

•email və tool parametrlərindən istifadə ederək – problem yarandıqda NCBI bizimlə əlaqə saxlaya bilir.

•50-dən çox sorğu göndəririksə, skriptləri həftəsonu və ya iş günləri 21:00 – 05:00 arasında işlədirik.

```
from Bio.Blast import NCBIWWW
NCBIWWW.email = "A.N.Other@example.com"
result_handle = NCBIWWW.qblast("blastn", "nt", "8332116")
```

FASTA faylından oxumaq üçün:

```
from Bio.Blast import NCBIWWW
fasta_string = open("m_cold.fasta").read()
result_handle = NCBIWWW.qblast("blastn", "nt", fasta_string)
```

FASTA faylını SeqIO ilə oxuyub yalnız ardıcıllığı istifadə etmək:

```
from Bio.Blast import NCBIWWW
from Bio import SeqIO
record = SeqIO.read("m_cold.fasta", format="fasta")
result_handle = NCBIWWW.qblast("blastn", "nt", record.seq)
```

Yerli BLAST işlətmək

Üstünlüklər:

•Yerli BLAST internet BLAST-dan daha sürətli ola bilər.

•Özünüzə məxsus verilənlər bazası yarada bilərsiniz.

•Məxfi və ya dərc olunmamış məlumatları istifadə etmək mümkündür (NCBI-yə göndərmək olmaz).

Çətinliklər:

•BLAST-ın komanda sətri vasitəsilə işləyən alətləri quraşdırılmalıdır.

•BLAST verilənlər bazası yüklənməli və yenilənməlidir.

•Müxtəlif BLAST paketləri mövcuddur; BLAT kimi bənzər proqramlar da var amma onlar saxta BLAST nəticələri yarada bilər.

BLAST (köhnə)

Böyük BLAST analizlərində yaranan məlumat həcmilə işləmək və BLAST işlərini avtomatlaşdırmaq çətin ola bilər.

Biopython BLAST-dakı problemləri yaxşı bilir və BLAST ilə işləməyi asanlaşdırmaq üçün bir çox alətlər hazırlayıb.

BLAST ilə işləmək iki əsas mərhələyə bölünə bilər, hər ikisini Biopython daxilində həyata keçirmək olar. Əvvəlcə, sorğu ardıcıllığınız üçün BLAST-ı işə salmaq və nəticə almaq. Daha sonra isə BLAST nəticələrini Python-da analiz üçün parse etmək. BLAST-ı işə salmağın bir çox yolu var və bunlar müxtəlif kateqoriyalara bölünə bilər. Ən vacib fərqlərdən biri BLAST-ı lokal (öz kompüterinizdə) işlətmək və BLAST-ı uzaqdan (məsələn, NCBI serverlərində) işlətməkdir.