

- Çıxış faylinda bir və ya bir neçə axtarış sorğusunun nəticələri ola bilər.
- Hər bir axtarış sorğusunda siz verilmiş axtarış verilənlər bazasından bir və ya bir neçə hit görəcəksiniz.
- Hər verilənlər bazası vuruşunda siz sorğu ardıcılığınız və verilənlər bazası ardıcılığı arasında faktiki ardıcılığın uyğunlaşdırılmasını ehtiva edən bir və ya bir neçə bölgəni görəcəksiniz.
- BLAT və ya Exonerate kimi bəzi programlar bu bölgələri bir neçə uyğunlaşdırma fragmentinə (yaxud BLAT-da bloklara) və ola bilsin ki, exonerate-dək eksonlara) daha da ayrıla bilər. Bu, həmişə gördüğünüz bir şey deyil, çünkü BLAST və HMMER kimi programlar bunu etmir.

Bu ümumiliyi dərk edərək, ondan Bio.SearchIO obyekt modelini yaratmaq üçünəsas kimi istifadə etmək qərarına gəldik. Obyekt modeli Python obyektlərinin iç-içəri yierarxiyasından ibarətdir, hər biri yuxarıda göstərilən bir konsepsiyanı təmsil edir. Bu obyektlər bunlardır:

- Tək axtarış sorğusunu təmsil etmək üçün QueryResult.
- Hit, vahid verilənlər bazasını təmsil etmək üçün vuruş. Hit obyektləri QueryResult daxilindədir və hər QueryResult-da sıfır və ya daha çox Hit obyekti var.
- HSP (yüksek kəxallı cüt üçün qısa), sorğu və hit ardıcılığı arasındaəhəmiyyətli uyğunlaşmaların region(larını) təmsil etmək üçün. HSP obyektləri Hit obyektlərinin içərisindədir və hər Hitin bir və ya daha çox HSP obyekti var.
- HSPFragment, sorğu və hit ardıcılığı arasında tək bitişik düzülüşü təmsil etmək üçün. HSPFragment obyektləri HSP obyektləri içərisindədir. BLAST və HMMER kimi ardıcıl axtarış vasitələrininəksəriyyəti HSP və HSPFragment obyektlərini birləşdirir, çünkü hər bir HSP yalnız bir HSPFragmentə malik olacaqdır. Bununla belə, çoxlu HSPFragment ehtiva edən HSP istehsal edən BLAT və Exonerate kimi alətlər var. Əgər bu, indi bir az çəşqin görünən, narahat olmayın, sonra bu iki obyekt haqqında dahaətraflı məlumat verəcəyik.

Bu dörd obyekt Bio.SearchIO-dan istifadə etdiyiniz zaman qarşılıqlıəlaqə quracağınız obyektləridir. Onlarəsas Bio.SearchIO metodlarından biri ilə yaradılmışdır: oxumaq, təhlil etmək, indeks və ya index_db. Bu əsulların təfərruatları sonrakibölmələrdə verilir. Bu bölmə üçün biz yalnız oxu və təhlildən istifadə edəcəyik. Bu funksiyalar öz Bio.SeqIO və Bio.AlignIO analoqları ilə eyni şəkildə davranır:

- oxu bir sorğu ilə çıxış fayllarının axtarışı üçün istifadə olunur və QueryResult obyekti qaytarır
- parse çoxlu sorğulu axtarış çıxış faylları üçün istifadə olunur və QueryResult verən generatoru qaytarır obyektlər

Bunu həll etdikdən sonra QueryResult ilə başlayaraq hər Bio.SearchIO obyekti araşdırmağa başlayaq.

11.1.1 Sorğu Nəticəsi

QueryResult obyekti tək axtarış sorğusunu təmsil edir və sıfır və ya daha çox Hit obyektlərini ehtiva edir. Əlimizdə olan BLAST faylından istifadənin necə görünüşünə baxaq:

```
>>> Bio idxaldan SearchIO >>> blast_qresult
= SearchIO.read ("my_blast.xml", "blast-xml") >>> print(blast_qresult)
```

```
Program: blastn (2.2.27+)
Sorğu: 42291 (61)
      mystery_seq
Hədəf: refseq_rna
Baxışlar: -----
      # # HSP ID + təsviri
```

```

0 1 gi|262205317|ref|NR_030195.1 | Homo sapiens mikroRNT 52...
1 1 gi|301171311|ref|NR_035856.1 | Pan troglodytes mikroRNT...
2 1 gi|270133242|ref|NR_032573.1 | Macaca mulatta mikroRNT ...
3 2 gi|301171322|ref|NR_035857.1 | Pan troglodytes mikroRNT...
4 1 gi|301171267|ref|NR_035851.1 | Pan troglodytes mikroRNT...
5 2 gi|262205330|ref|NR_030198.1 | Homo sapiens mikroRNT 52...
6 1 gi|262205302|ref|NR_030191.1 | Homo sapiens mikroRNT 51...
7 1 gi|301171259|ref|NR_035850.1 | Pan troglodytes mikroRNT...
8 1 gi|262205451|ref|NR_030222.1 | Homo sapiens mikroRNT 51...
9 2 gi|301171447|ref|NR_035871.1 | Pan troglodytes mikroRNT...
10 1 gi|301171276|ref|NR_035852.1 | Pan troglodytes mikroRNT...
11 1 gi|262205290|ref|NR_030188.1 | Homo sapiens mikroRNT 51...
12 1 gi|301171354|ref|NR_035860.1 | Pan troglodytes mikroRNT...
13 1 gi|262205281|ref|NR_030186.1 | Homo sapiens mikroRNT 52...
14 2 gi|262205298|ref|NR_030190.1 | Homo sapiens mikroRNT 52...
15 1 gi|301171394|ref|NR_035865.1 | Pan troglodytes mikroRNT...
16 1 gi|262205429|ref|NR_030218.1 | Homo sapiens mikroRNT 51...
17 1 gi|262205423|ref|NR_030217.1 | Homo sapiens mikroRNT 52...
18 1 gi|301171401|ref|NR_035866.1 | Pan troglodytes mikroRNT...
19 1 gi|270133247|ref|NR_032574.1 | Macaca mulatta mikroRNT ...
20 1 gi|262205309|ref|NR_030193.1 | Homo sapiens mikroRNT 52...
21 2 gi|270132717|ref|NR_032716.1 | Macaca mulatta mikroRNT ...
22 2 gi|301171437|ref|NR_035870.1 | Pan troglodytes mikroRNT...
23 2 gi|270133306|ref|NR_032587.1 | Macaca mulatta mikroRNT ...
24 2 gi|301171428|ref|NR_035869.1 | Pan troglodytes mikroRNT...
25 1 gi|301171211|ref|NR_035845.1 | Pan troglodytes mikroRNT...
26 2 gi|301171153|ref|NR_035838.1 | Pan troglodytes mikroRNT...
27 2 gi|301171146|ref|NR_035837.1 | Pan troglodytes mikroRNT...
28 2 gi|270133254|ref|NR_032575.1 | Macaca mulatta mikroRNT ...
29 2 gi|262205445|ref|NR_030221.1 | Homo sapiens mikroRNT 51...
~~~
```

97 1 gi|356517317|ref|XM_003527287.1 | PROQNOZ EDİLİR: Glycine ma...
98 1 gi|297814701|ref|XM_002875188.1 | Arabidopsis lyrata su...
99 1 gi|397513516|ref|XM_003827011.1 | TƏ QDİM EDİLƏ N: Panisc...

Biz obyekt modelinin saə thini yenicə cızmağa başlamışıq, lakin siz artıq bə zilə rinin olduğunu görə bilərsiniz faydalı məlumat. QueryResult obyektində çapı çağrımaqla siz görə bilərsiniz:

- Proqramın adı və versiyası (blastn version 2.2.27+)
- Sorğunun ID-si, təsviri və onun ardıcılığının uzunluğu (ID 42291, təsvir "sirr ardıcılığı"dır və 61 nukleotid uzunluğunda)
- Axtarış üçün hədəf verilənlər bazası (refseq rna) -
- Nəticə hitlərin qısa icmali. Sorğu ardıcılığımız üçün 100 potensial hit (nömrələnmmiş cədvəldə 0-99). Hər bir hit üçün biz onun nəçər HSP-ni, ID-sini və bir parçasını da görə bilərik. onun təsviri. Diqqət yetirin ki, Bio.SearchIO yalnız hitləri göstərməklə hit cədvəlinin icmalını keşir 0-29, sonra isə 97-99.

İndi yuxarıdakı prosedurdan istifadə edərək BLAT nəticələrimizi yoxlayaq:

```
>>> blat_qresult = SearchIO.read("my_blat.psl", "blat-psl") >>> print(blat_qresult)
```

```
Program: blat (<namə lum versiya>
Soru: mystery_seq (61) <namə lum
tə svir> Hə də f: <namə lum
hə də f>
Baxışlar: -----
# # HSP ID + tə sviri
-----
0      17 chr19 <namə lum tə svir>
```

Bə zi fə rqlə rin olduğunu də rhal görə cə ksınız. Bunlardan bə zilə ri, gördüğünüz kimi, PSL formatının onun tə fə rrüatlarını saxlama üsulu ilə ə laqə dardır. Qalanları BLAST və BLAT arxivşalarımız arasında hə qiqi program və hə də f verilə nlə r bazası fə rqlə ri ilə ə laqə dardır:

- Programın adı və versiyası. Bio.SearchIO programın BLAT olduğunu bilir, lakin çıkış faylında program versiyası ilə bağlı mə lumat yoxdur, ona görə də o, standart olaraq 'namə lum versiya' olaraq tə yin olunur.
- Sorğunun ID-si, tə sviri və onun ardıcılığının uzunluğu. Diqqə t yetirin ki, bu detallar BLAST-da gördüyüümüzdə n bir qə də r fə rqlidir. ID 42991 ə və zinə 'sirr ardıcılığı'dır, mə lum tə svir yoxdur, lakin sorğunun uzunluğu hə lə də 61-dir. Bu, ə slində fayl formatlarının özəl ri tə rə fində n tə qdim edilə n fə rqdir.
- BLAST bə zə n öz sorğu identifikatorlarını yaradır və ardıcılığın tə sviri kimi orijinal ID-də n istifadə edir.
- BLAT çıkış faylında göstə rilmə diyi üçün hə də f verilə nlə r bazası mə lum deyil.
- Və nə hayə t, ə limizdə olan hitlə rin siyahısı tamam başqadır. Burada biz sorğu ardıcılığının yalnız 'chr19' verilə nlə r bazası girişinə toxunduğuunu görürük, lakin orada 17 HSP bölgə sini görürük. Bununla belə, hə r birinin öz hə də f verilə nlə r bazası olan fə rqlı bir programdan istifadə etdiyimizi nə zə rə alsaq, bu tə ə ccübü olmamalıdır.

Çap metoduna müraciə t edə rkə n gördüğünüz bütün tə fə rrüatlar Python-un obyekt atributuna giriş notasiyasından (aka nöqtə notasiyası) istifadə etmə klə fə rdi olaraq ə ldə edilə bilə r. Eyni üsuldan istifadə edə rə kə ldə edə bilə cə yiniz digə r formata xas attributlar da var.

```
>>> print("%s %s" % (blast_qresult.program, blast_qresult.version)) blastn 2.2.27+ >>> print("%s %s" % (blat_qresult.program,
blat_qresult.version)) blat
<namə lum versiya> >>> blast_qresult.paramhold.0ml xüsusi #blast_qresult.paramhold.01
```

Ə ləctən attributların tam siyahısı üçün hə r bir formata aid sə nə dlə ri yoxlaya bilə rsiniz. Budur **BLAT üçün** olanlar və **BLAT üçün**.

QueryResult obyektlə rində çapdan istifadə ni nə zə rdə n keçirdikdə n sonra gə lin daha də rinə inə k. QueryResult tam olaraq nə dir? Python obyektlə ri baxımından QueryResult siyahı və lügə t arasında hibriddir. Başqa sözlə , siyahıların və lügə tlə rin bütün rahat xüsusiyyə tlə rinə malik konteyner obyektidir.

Python siyahıları və lügə tlə ri kimi QueryResult obyektlə ri də tə krarlanır. Hə r iterasiya Hit obyektini qaytarır:

```
>>> blast_qresult -da hit üçün :
```

```
...
vurmaq
...
Hit(id='gi|262205317|ref|NR_030195.1|', query_id='42291', 1 hsps)
Hit(id='gi|301171311|ref|NR_035856.1|', query_id='42291', 1 hsps)
Hit(id='gi|270133242|ref|NR_032573.1|', query_id='42291', 1 hsps)
Hit(id='gi|301171322|ref|NR_035857.1|', query_id='42291', 2 hsps)
Hit(id='gi|301171267|ref|NR_035851.1|', query_id='42291', 1 hsps)
...
```

QueryResult-da neçə element (hit) olduğunu yoxlamaq üçün sadə cə olaraq Python-un len metodunu işə sala bilərsiniz:

```
>>> len(blast_qresult)
100
>>> len(blast_qresult) 1
```

Python siyahıları kimi, dilim notasiyasından istifadə edərək QueryResult-dan elementləri (xitləri) əldə edə bilərsiniz:

```
>>> blast_qresult[0] # ən yaxşı hiti əldədir (id='gi|262205317|
ref|NR_030195.1|', query_id='42291', 1 hsp) >>> blast_qresult[-1] # son hiti alır Hit(id='gi|
397513516|ref|XM_003827011.1|', query_id='42291', 1 hsp)
```

Çoxsaylı xitləri əldə etmək üçün siz QueryResult obyektlərinin dilim notasiyasından istifadə edərək dilimləyə bilərsiniz. Bu halda, dilim yalnız dilimlənmış hitləri ehtiva edən yeni QueryResult obyektini qaytaracaq:

```
>>> blast_slice = blast_qresult[:3] # ilk üç vuruşu dilimləyir >>> print(blast_slice)
```

Program: blastn (2.2.27+)

Sorğu: 42291 (61)

 mystery_seq

Hədəf: refseq_rna

Baxışlar: -----

 # # HSP ID + təsviri

0	1 gi 262205317 ref NR_030195.1 Homo sapiens mikroRNA 52... 1 gi 301171311 ref
1	NR_035856.1 Pan trogloditlər mikroRNT... 1 gi 270133242 ref NR_032573.1 Macaca
2	mulatta mikroRNT ...

Python lügətləri kimi, siz də hit identifikatorundan istifadə edərək hitləri əldə edə bilərsiniz. Bu, axtarış sorğusunun nəticələrində verilmiş hit ID-nin mövcud olduğunu bildiyiniz halda xüsusilə faydalıdır:

```
>>> blast_qresult["gi|262205317|ref|NR_030195.1 |"]
Hit(id='gi|262205317|ref|NR_030195.1|', query_id='42291', 1 hsp)
```

Siz həmçinin hit_keys istifadə edərək Hit obyektlərinin tam siyahısını və Hit ID-lərinin tam siyahısını əldə edə bilərsiniz:

```
>>> blast_qresult.hits [...] # bütün
xitlərin siyahısı >>> blast_qresult.hit_keys [...]
```

 # bütün hit ID-lərin siyahısı

Sorğu nəticələrində müəyyən bir vuruşun olub olmadığını yoxlamaq istəyirsizsə nə etməli? Siz in açar sözündən istifadə edərək sadə Python üzvlük testi edə bilərsiniz:

```
>>> "gi|262205317|ref|NR_030195.1|" blast_qresult -da Doğrudur
```

```
>>> "gi|262205317|ref|NR_030194.1|" blast_q nəticə sində
Yalan
```

Bəzən bir vuruşun olub olmadığını bilmək kifayət deyil; siz də hitin dərəcəsini bilmək istəyirsiniz. Burada indeks metodunu xülasətmə üçün gəlir:

```
>>> blast_qresult.index("gi|301171437|ref|NR_035870.1|") 22
```

Unutmayın ki, biz burada Python-un sıfır ə saslı indekslə şdirmə stilində n istifadə edirik. Bu, yuxarıdakı vuruşumuz deməkdir №-də yer alır. 22 yox, 23.

Hə mçinin qeyd edək ki, burada gördüğünüz hit dərəcəsi orijinal axtarış çıkış faylında mövcud olan yerli hit sıralamasına əsaslanır. Fərqli axtarış vasitələri bu hitləri müxtəlif meyarlara əsasən sıfariş edə bilər.

Əgər yerli hit sıfarişi zövqünüzü uyğun gəlmirsə, siz QueryResult obyektinin çeşidləmə metodundan istifadə edə bilərsiniz. O, Python-un list.sort metoduna çox bənzəyir, yeni çeşidlənmmiş QueryResult obyekti yaratmaq və ya yaratmamaq üçün seçim əlavə edilir.

QueryResult.sort istifadə edərək, hər bir hitin tam ardıcılıq uzunluğu əsasında hitləri çeşidləmək üçün burada bir nümunə verilmişdir. Bu xüsusi növ üçün biz in_place bayrağını False olaraq təyin edəcəyi ki, çeşidləmə yeni QueryResult obyekti qaytarsın və ilkin obyektimizi çeşidlənməmiş buraxsın. Biz həmçinin əks bayrağı True olaraq təyin edəcəyi ki, azalan qaydada sıralayaq.

```
>>> blast_qresult[:5] -də hit üçün : # id və ilk beş vuruşun ardıcılıq uzunluğu
```

```
...     çap ("%s %i" % (hit.id, hit.seq_len))
...
```

```
gi|262205317|ref|NR_030195.1| 61 gi|
301171311|ref|NR_035856.1| 60 gi|
270133242|ref|NR_032573.1| 85 gi|
301171322|ref|NR_035857.1| 86 gi|
301171267|ref|NR_035851.1| 80
```

```
>>> sort_key = lambda hit: hit.seq_len >>>
```

```
sorted_qresult = blast_qresult.sort(key=sort_key, reverse=True) >>> sorted_qresult[:5] -də vuruş
üçün : print("%s %i" % (hit.id, hit.seq_len))
...
```

```
gi|397513516|ref|XM_003827011.1| 6002 gi|
390332045|ref|XM_776818.2| 4082 gi|
390332043|ref|XM_003723358.1| 4079 gi|
356517317|ref|XM_003527287.1| 3251 gi|
356543101|ref|XM_003539954.1| 2936
```

Burada in_place bayrağının olmasının üstünlüyü ondan ibarətdir ki, biz yerli sıfarişi qoruyuruq, ona görə də ondan sonra yenidən istifadə edə bilərik. Qeyd etmək lazımdır ki, bu QueryResult.sort-un defolt davranışını deyil, buna görə də in_place bayrağını açıq şəkildə True olaraq təyin etməli olduq.

Bu nöqtədə QueryResult obyektləri haqqında kifayət qədər məlumatınız var ki, onun sizin üçün işləməsi mümkün olsun. Lakin Bio.SearchIO modelindəki növbəti obyektə keçməzdən nəvvəl QueryResult obyektləri ilə işləməyi daha da asanlaşdırıbiləcək daha iki üsul dəstindən nəzər salaq: filtr və xəritə üsulları.

Əgər Python-un siyahı anlayışları, generator ifadələri və ya daxili filtr və xəritə funksiyaları ilə tanışınızsa, onların siyahıya bənzər obyektlərlə işləmək üçün nəqədər faydalı olduğunu biləcəksiniz (əgər deyilsinizsə, onları yoxlayın!). QueryResult obyektlərinin manipulyasiya etmək üçün bu quraşdırılmış üsullardan istifadə edə bilərsiniz, lakin siz adı Python siyahıları ilə nəticələnəcək və daha maraqlı manipulyasiyalar etmək qabiliyyətinizi itirəcəksiniz.

Buna görə QueryResult obyektləri filtr və xəritə metodlarının özəlləzzətinə min edir. Süzgəcən analoqu kimi hit_filter və hsp_filter üsulları var. Adından da göründüyü kimi, bu üsullar QueryResult obyekti ya Hit obyektlərinə, ya da HSP obyektlərinəsəsizdir. Eynilə, xəritənin analoqu kimi, QueryResult obyektləri də hit_map və hsp_map metodlarını təmin edir. Bu üsullar verilmiş funksiyani müvafiq olaraq QueryResult obyektiindəki bütün hitlərə və ya HSP-lərə tətbiq edir.

Gəlin hit_filter ilə başlayan bu üsulları işlək vəziyyətdə görək. Bu üsul, verilmiş Hit obyektiinintəyin etdiyiniz şərti keçib-keçmədiyini yoxlayan geri çağırış funksiyasını qəbul edir. Başqa sözlə, funksiya öz argumenti kimi tək Hit obyektiini qəbul etməli və ya True və ya False qaytarmalıdır.

Yalnız bir HSP olan Hit obyektlə rini süzgə cdə n keçirmə k üçün hit_filter istifadə nümunə sidir:

```
>>> filter_func = lambda hit: len(hit.hsps) > 1 # geri çağırış funksiyası >>> len(blast_qresult) # no. 100 filtrlə nə və l hitlə r

>>> filtered_qresult = blast_qresult.hit_filter(filter_func) >>> len(filtered_qresult) # no. süzgə cdə n sonra
hitlə rin sayı 37

>>> filtered_qresult[:5] -də hit üçün : # hit uzunluqlarını tez yoxlayın print("%s %i" % (hit.id,
...     len(hit.hsps)))
...
gi|301171322|ref|NR_035857.1| 2 gi|
262205330|ref|NR_030198.1| 2 gi|
301171447|ref|NR_035871.1| 2 gi|
262205298|ref|NR_030190.1| 2 gi|
270132717|ref|NR_032716.1| 2
```

hsp_filter hit_filter ilə eyni işlə yir, yalnız Hit obyektlə rinə baxmaq ə və zinə , hə r vuruşda HSP obyektlə rində filtrlə mə hə yata keçirir.

Xə ritə metodlarına gə ldikdə , onlar da öz arqumentlə ri kimi geri çağırış funksiyasını qə bul edirlə r. Bununla belə , True ə ya False qaytarmaq ə və zinə , geri çağırış funksiyası də yişdirilmiş Hit ə ya HSP obyekti qaytarmalıdır (hit_map ə ya hsp_map istifadə etmə yinidə n asılı olaraq).

Hit identifikatorlarının adını də yişmə k üçün hit_map istifadə etdiyimiz bir nümunə yə baxaq:

```
>>> def map_func(hit): # "gi|301171322|
...     ref|NR_035857.1|" adını də yışır "NR_035857.1" hit.id = hit.id.split(" | ")[3] nöqtə sinə qayıt
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
................................................................
```

Yenə hsp_map hit_map ilə eyni işlə yir, lakin Hit obyektlə ri ə və zinə HSP obyektlə rində işlə yir.

11.1.2 Vurma

Hit obyektlə ri bir verilə nlə r bazası girişində n bütün sorğu nə ticə lə rini tə msil edir. Onlar Bio.SearchIO obyekt iyerarxiyasında ikinci sə viyyə li konteynerdir. Gördünüz ki, onlar QueryResult obyektlə rində dir, lakin onların özlə rində HSP obyektlə ri var.

BLAST axtarışımızdan başlayaraq onların necə göründüğünü görə k:

```
>>> Bio import SearchIO -dan >>>
blast_qresult = SearchIO.read("my_blast.xml", "blast-xml") >>> blast_hit =
blast_qresult[3] sorğu nə ticə sində n # dördüncü hit >>> print(blast_hit)
```

Sorğu: 42291

sirli_seq
 Hit: gi|301171322|ref|NR_035857.1| (86)
 Pan troglodytes microRNA mir-520c (MIR520C), microRNT
 HSP-lər:

# Edə yər Bit hesabi	Aralıq	Sorğu diapazonu	Hit diapazonu
0 8.9e-20	100.47	60	[1:61] [13:73]
1 3.3e-06	55.39	60	[0:60] [13:73]

Görürsünüz ki, burada əsas şeylər var:

- Sorğunun ID-si və təsviri mövcuddur. Hit həmişə sorğuya bağlıdır, ona görə də biz izləmək istəyirik mənşəli sorğunun da. Bu dəyərlərlə rəsədy_id və istifadə edərək hitdən daxil olmaq olar query_description atributları.
- Bizdə həmçinin unikal hit ID-si, təsviri və tam ardıcılıq uzunluğu var. Onlara id istifadə edərək daxil olmaq olar, təsvir və müvafiq olaraq seq_len.
- Nəhayət, bu hitin ehtiva etdiyi HSP-lər haqqında qısa məlumatı ehtiva edən cədvəl var. Hər cərgədə, Biz siyahıda vacib HSP təfərruatlarını əldə etdik: HSP indeksi, onun e-dəyəri, bit hesabı, span (boşluqlar daxil olmaqla hizalanma uzunluğu), sorğu koordinatları və vuruş koordinatları.

İndi bunu BLAT axtarışı ilə müqayisə edək. Unutmayın ki, BLAT axtarışında bir vuruşumuz var 17 HSP.

```
>>> blat_qresult = SearchIO.read("my_blat.psl", "blat-psl")
>>> blat_hit = blat_qresult[0] # yeganə hit
```

Sorğu: mystery_seq

<naməlum təsvir>

Hit: chr19 (59128983)

<naməlum təsvir>

HSP-lər:

# Edə yər Bit hesabi	Aralıq	Sorğu diapazonu	Hit diapazonu
0	?	?	[0:61] [54204480:54204541]
1	?	?	[0:61] [54233104:54264463]
2	?	?	[0:61] [54254477:54260071]
3	?	?	[1:61] [54210720:54210780]
4	?	?	[0:60] [54198476:54198536]
5	?	?	[0:61] [54265610:54265671]
6	?	?	[0:61] [54238143:54240175]
7			[0:60] [54189735:54189795]
8			[0:61] [54185425:54185486]
9			[0:60] [54197657:54197717]
10	????	????	[0:61] [54255662:54255723]
11	?	?	[0:61] [54201651:54201712]
12	?	?	[8:60] [54206009:54206061]
13	?	?	[10:61] [54178987:54179038]
14	?	?	[8:61] [54212018:54212071]
15	?	?	[8:61] [54234278:54234321]
16	?	?	[8:61] [54238143:54238196]

Burada, `a və llə r` gördüğümüz BLAST hitində olduğu kimi, oxşar sə viyyə də tə fə rrüat `ə ldə etdik`. `Bə zi fə rqə r` var izah etmə yə də yə r, lakin:

- e-də yə r və bit xal sütun də yə rlə ri. BLAT HSP-lə rin e-də yə rlə ri və bit balları olmadığı için ekran defoltları '?'.
- Aralıq sütunu haqqında nə demə k olar? Aralıq də yə rlə ri tam hizalanma uzunluğunu göstə rmə k üçün nə zə rdə tutulub, bütün qaliqlardan və mövcud ola bilə cə k boşluqlardan ibarə tdir. PSL formatında bu yoxdur mə lumat asanlıqla mövcuddur və Bio.SearchIO bunun nə olduğunu tə xmin etmə yə cə hd etmir, ona görə də biz `ə ldə edirik` '?' e-də yə r və bit hesabı sütunlarına bə nzə yır.

Python obyektlə ri baxımından Hit, demə k olar ki, Python siyahıları ilə eyni davranış, lakin HSP obyektlə rini ehtiva edir. eksklüziv olaraq. Ə gə r siyahılarla tanışsınızsa, Hit obyekti ilə işlə mə kdə heç bir çə tinliklə qarşılaşmamalısınız.

Python siyahıları kimi, Hit obyektlə ri də tə karrlana bilir və hə r iterasiya özündə bir HSP obyektini qaytarır:

```
>>> blast_hit -də hsp üçün :
...
      hsp
...
HSP(hit_id='gi|301171322|ref|NR_035857.1|', query_id='42291', 1 fragment)
HSP(hit_id='gi|301171322|ref|NR_035857.1|', query_id='42291', 1 fragment)
```

Neçə HSP obyektinin olduğunu görmə k üçün Hit-də len-i çağırı bilə rsiniz:

```
>>> len(blast_hit)
2
>>> len(blat_hit)
17
```

Tə k HSP və ya çoxlu HSP obyektlə rini `ə ldə etmə` k üçün Hit obyektlə rində dilim notasiyasından istifadə edə bilə rsiniz. QueryResult kimi, birdə n çox HSP üçün dilimlə sə niz, yalnız dilimlə nmışlə ri ehtiva edə n yeni Hit obyekti qaytarılacaq. HSP obyektlə ri:

```
>>> blat_hit[0] # tə k elementlə ri ə ldə edin
HSP(hit_id='chr19', query_id='mystery_seq', 1 fragment)
>>> sliced_hit = blat_hit[4:9] # çoxlu elementlə ri ə ldə edin
>>> lent(dilimlə nmış_vur)
5
>>> çap(dilimlə nmış_vur)
Sorgu: mystery_seq
<namə lum tə svir>
Hit: chr19 (59128983)
<namə lum tə svir>
```

HSP-lə r: -----

# E-də yə r Bit hesabı	Aralıq	Sorgu diapazonu	Hit diapazonu
0		[0:60]	[54198476:54198536]
1	??	[0:61]	[54265610:54265671]
2	?	[0:61]	[54238143:54240175]
3	?	[0:60]	[54189735:54189795]
4	?	[0:61]	[54185425:54185486]

Siz hə mçinin HSP-ni Hit daxilində çeşidlə yə bilə rsiniz, burada gördüğünüz çeşidlə mə metodu kimi eyni arqumentlə rdə n istifadə edə rə k QueryResult obyekti.

Nə hayə t, Hit obyektlə rində istifadə edə bilə cə yiniz filtr və xə ritə üsulları da var. QueryResult-dan fə rqli olaraq obyekt, Hit obyektlə rində yalnız bir filtr variantı (Hit.filter) və xə ritə nin bir variantı (Hit.map) var. Hə r ikisi Hit.filter və Hit.map bir Hitin malik olduğu HSP obyektlə ri üzə rində işlə yır.

11.1.3 HSP

HSP (yükə k bal cütü) sorğu ardıcılığına ə hə miyyə tli uyğunlaşma(lar) ehtiva edə n vuruş ardıcılığında bölgə (lə ri) tə msil edir. O, sorğu ardıcılığınız və verilə nlə r bazası girişə arasında faktiki uyğunluğu ehtiva edir. Bu uyğunluq ardıcılıqla axtarış alə tinin alqoritmlə ri ilə müə yyə n edildiyi üçün HSP obyekti axtarış alə ti tə rə fində n hesablanmış statistik mə lumatların ə sas hissə sini ehtiva edir. Bu, hə məcən QueryResult və ya Hit obyektlə rində gördüğünüz fə rqlə rlə müqayisə də müxtə lif axtarış alə tlə rində n HSP obyektlə ri arasındaki fə rqı daha aydın edir.

BLAST və BLAT axtarışlarından bə zi nümunə lə rə baxaq. Ə vvə lcə BLAST HSP-ə baxacağıq:

```
>>> Bio import SearchIO- dan >>>
blast_qresult = SearchIO.read("my_blast.xml", "blast-xml") >>> blast_hsp = blast_qresult[0]
[0] # ilk vuruş, ilk hsp >>> çap (blast_hsp)
```

Sorğu: 42291 mystery_seq
 Hit: gi|262205317|ref|NR_030195.1 | Homo sapiens mikroRNA 520b (MIR520...
 Sorğu diapazonu: [0:61] (1)
 Hit diapazonu: [0:61] (1)
 Sürə tli statistika: qiymə tlə ndirin 4.9e-23; bitscore 111.29
 Fragmentlə r: 1 (61 sütun)
 Sorğu - CCCTCTACAGGGAAAGCGCTTCTGTTCTGAAAGAAAGAAAGTGCTTCCTTTAGAGGG
 ||||||| ||||| ||||| ||||| ||||| ||||| ||||| ||||| ||||| |||||
 Hit - CCCTCTACAGGGAAAGCGCTTCTGTTCTGAAAGAAAGAAAGTGCTTCCTTTAGAGGG

QueryResult və Hit kimi, HSP-də çapı çağırmaq onun ümumi tə fə rrüatlarını göstə rir:

- Sorğu və hit identifikatorları və tə svirlə ri var. HSP-mizi müə yyə n etmə k üçün bunlara ehtiyacımız var.
- Biz hə məcən sorğu və hit ardıcılığının uyğun diapazonunu ə ldə etdik. Burada istifadə etdiyimiz dilim notasiyası diapazonun Python-un indekslə şdirmə stilində n (sıfır ə saslı, yarı açıq) istifadə edilə rə k göstə rildiyinin göstə ricisidir. Mötə rizə nin içində ki nömrə ipi bildirir. Bu halda, hə r iki ardıcılığın artı ipi var.
- Bə zi sürə tli statistika mövcuddur: e-də yə r və bitscore.
- HSP fragmetlə ri haqqında mə lumat var. Hə lə lik buna mə hə l qoymayıñ; daha sonra izah edilə cə k.
- Və nə hayə t, sorğu və hit ardıcılığının uyğunlaşdırılmasının özü var.

QueryResult və Hit-də olduğu kimi, bu tə fə rrüatlara nöqtə notasiyasından istifadə etmə klə müstə qil olaraq daxil olmaq olar:

```
>>> blast_hsp.query_range (0, 61)
```

```
>>> blast_hsp.eval 4.91307e-23
```

Baxmayaraq ki, onlar mövcud olan yeganə atribut deyillə r. HSP obyektlə ri standart xüsusiyyə tlə r də sti ilə gə lir onların müxtə lif detallarını araşdırmağı asanlaşdırır. Budur bə zi nümunə lə r:

```
>>> blast_hsp.hit_start # vuruş ardıcılığının başlanğıc koordinatı 0

>>> blast_hsp.query_span # sorğu ardıcılığında neçə qalıq 61

>>> blast_hsp.aln_span # hizalanma 61 nə qə də rdir
```

HSP **sə nə dlə rini** yoxlayın bu ə vvə lə də n tə yin edilmiş xüsusiyyə tlə rin tam siyahısı üçün.

Bundan ə lava , hə r bir ardıcılıqla axtarış alə ti adə tə n HSP obyektlə ri üçün öz statistikasını / tə fə rrüatlarını hesablayır.

Mə sə lə n, XML BLAST axtarışı hə m də boşluqların və eyni qalıqların sayını verir. Bu atributlara aşağıdakı kimi daxil olmaq olar:

```
>>> blast_hsp.gap_num # boşluqların sayı 0
```

```
>>> blast_hsp.ident_num # eyni qalıqların sayı 61
```

Bu tə fə rrüatlar formata xasdır; onlar başqa formatlarda olmaya bilə r. Verilmiş ardıcılıqla axtarış alə ti üçün hansı detalların mövcud olduğunu görmə k üçün Bio.SearchIO-da formatın **sə nə dlə rini** yoxlamalısınız.

Alternativ olaraq, mövcud olanların qısa siyahısı üçün **._dict_.keys()** funksiyasından da istifadə edə bilə rsiniz:

```
>>> blast_hsp._dict_.keys() ['bitscore',
'qiymə tlə ndirmə ', 'ident_num', 'boşluq_num', 'bitscore_raw', 'pos_num', '_items']
```

Nə hayə t, siz HSP-nin sorğu və hit atributlarının sadə cə adı sə tirlə r olmadığını fə rq etmiş ola bilə rsiniz:

```
>>> blast_hsp.query
```

```
SeqRecord(seq=Seq('CCCTCTACAGGGAAAGCGTTCTGTTCTGAAAGAAAAAGAAAGTGCCTCCTT...GGG'), id='42291', ad='al >>> blast_hsp.hit
```

```
SeqRecord(seq=Seq('CCCTCTACAGGGAAAGCGTTCTGTTCTGAAAGAAAAAAAGTGCCTCCTT...GGG'), id='gi|262205317|ref
```

Onlar 4-cü Bölüm də ə vvə llə r gördüğünüz SeqRecord obyektlə ridir ! Bu o demə kdir ki, siz hə r cür edə bilə rsiniz HSP.query və /və ya HSP.hit-də SeqRecord obyektlə ri ilə edə bilə cə yiniz maraqlı şeylə r.

HSP obyektinin MultipleSeqAlignment obyekti olan hizalama xassə sına malik olması sizi tə ə ccüblə ndirmə mə lidir:

```
>>> çap (blast_hsp.aln)
```

2 sıra və 61 sütunla düzülmə

```
CCCTCTACAGGGAAAGCGTTCTGTTCTGAAAGAAAAAG...GGG 42291
```

```
CCCTCTACAGGGAAAGCGTTCTGTTCTGAAAGAAAAAAAG...GGG gi|262205317|ref|NR_030195.1 |
```

BLAST HSP-ni yoxladıqdan sonra, indi fə rqli bir HSP növü üçün BLAT nə ticə lə rimizdə n HSP-lə rə nə zə r salaq. Hə mişə olduğu kimi, biz onun üzə rində çapı çağrımaqla başlayacaqıq:

```
>>> blat_qresult = SearchIO.read("my_blat.psl", "blat-psl") >>> blat_hsp = blat_qresult[0]
[0] # ilk vuruş, ilk hsp >>> çap (blat_hsp)
```

Sorğu: mystery_seq <nəmə lum tə svir>

Hit: chr19 <nəmə lum tə svir>

Sorğu diapazonu: [0:61] (1)

Hit diapazonu: [54204480:54204541] (1)

Sürə tli statistika: qiymə tlə ndirmə k ?; bitscore?

Fraqmentlə r: 1 (? sütun)

Çıxışlardan bə zilə rini artıq tə xmin etmiş ola bilə rsiniz. Bizdə sorğu və hit identifikatorları, tə svirlə r və ardıcılıqlı koordinatları var. Qiymə tlə ndirmə və bitscore üçün də yə rlə r '?' çünkü BLAT HSP-lə rdə bu atributlar yoxdur.

Ancaq burada ə n böyük fə rq ondadır ki, siz heç bir ardıcıl düzülmə ni görmürsünüz. Ə gə r yaxından baxsanız, PSL formatlarının özlə rində heç bir hit və ya sorğu ardıcılığı yoxdur, ona görə də Bio.SearchIO heç bir ardıcılıq və ya uyğunlaşdırma obyekti yaratmayacaq. HSP.query, HSP.hit və ya HSP.aln-ə daxil olmağa çalışsanız nə baş verir? Bu atributlar üçün defolt də yə rlə ri ə ldə edə cə ksiniz, bu heç biri:

```
>>> blat_hsp.hit heç bir doğru deyil
```

```
>>> blat_hsp.query heç biri doğru deyil
```

```
>>> blat_hsp.aln heç bir doğru deyil
```

Bu, digər attributlara təsir etmir. Məsələn, siz hələ də sorğunun uzunluğuna və ya hit hizasına daxil ola bilərsiniz. Heç bir attributun göstərilmə məsini baxmayaraq, PSL formatında hələ də bu məlumat var, ona görədə Bio.SearchIO onları çıxara bilər:

```
>>> blat_hsp.query_span # sorğu uyğunluğunun uzunluğu 61 >>> blat_hsp.hit_span #
hit
uyğunluğunun uzunluğu 61
```

Digər formata xas attributlar hələ də mövcuddur:

```
>>> blat_hsp.score # PSL bali 61
```

```
>>> blat_hsp.mismatch_num # uyğunsuzluq sütunu 0
```

İndiyə qədər bu qədər yaxşıdır? BLAT nəticələrimizə HSP-nin başqa bir "variantına" baxdıqda işlər daha maraqlı olur. BLAT arxalarında bəzən nəticələrimizi "bloklara" ayırdığımızı xatırlaya bilərsiniz.

Bu bloklar, mahiyətəcə, aralarında bəzi müdaxilə ardıcılığına malik ola bilən hizalama fragmentləridir.

Bio.SearchIO-nun bununla necə məşğul olduğunu görmək üçün çoxlu bloklardan ibarət BLAT HSP-ə nəzər salaq:

```
>>> blat_hsp2 = blat_qresult[0][1] # birinci vuruş, ikinci hsp >>> çap (blat_hsp2)
```

Sorğu: mystery_seq <naməlum təsvir>

Hit: chr19 <naməlum təsvir>

Sorğu diapazonu: [0:61] (1)

Hit diapazonu: [54233104:54264463] (1)

Sürətli statistika: qiymətləndirin?; bitscore?

Fraqmentləri: -----

#	Aralıq	Sorğu diapazonu	Hit diapazonu
0	?	[0:18]	[54233104:54233122]
1	?	[18:61]	[54264420:54264463]

Burada nə baş verir? Biz hələ də bəzi vacib təfərrüatları əhatə edirik: ID-lər və təsvirlər, koordinatlar və sürətli statistika əvvəllər gördükərinizlə eynidir. Ancaq fragmentlərin təfərrüatları hamısı fərqlidir.

"Fragmentləri: 1'i göstərməkə və zinəndi iki məlumat sətirindən ibarət cədvəlimiz var.

Bio.SearchIO çoxlu fragmentlərə malik HSP-lərlə belə məşğul olur. Daha əvvəl qeyd edildiyi kimi, HSP düzülüşü ardıcıl ardıcılıqla fragmentlərə ayrıla bilər. Aradan keçən ardıcılıqlar sorğunun vurulduğu uyğunluğun bir hissəsi deyil, ona görədə onlar sorğunun və ya hit ardıcılığının bir hissəsi hesab edilməməlidir. Bununla belə, onlar ardıcılıq koordinatları ilə necə davrandığımıza təsir edir, ona görədə biz onlara məhəl qoymamırıq.

Yuxarıdakı HSP-nin hit koordinatına nəzər salın. Hit range: sahə sində koordinatın [54233104:54264463] olduğunu görürük. Lakin cədvəl sətirlərinə nəzər salıqda görünür ki, bu koordinatın əhatə etdiyi bütün region sorğumuza uyğun gəlmir. Xüsusiylə, müdaxilə edən bölgə 54233122 ilə 54264420 arasındadır.

Bəs niyə sorğu koordinatları bitişik görünür, soruştursunuz? Bu mükəmməl idir. Bu halda bu o deməkdir ki, sorğu uyğunluğu bitişikdir (arada olan bölgələr yoxdur), hit uyğunluğu isə deyil.

Yeri gəlmışkən, bütün bu attributlar birbaşa HSP-dənəldə edilə bilər:

```
>>> blat_hsp2.hit_range # bütün HSP-nin başlangıç və son koordinatlarını vur (54233104, 54264463) >>> blat_hsp2.hit_range_all #
hə r bir fragmentin başlangıç və
son koordinatlarını vur [(54233104, 5423542,) 54264463]) >>> blat_hsp2.hit_span # bütün HSP 31359- un hit aralığı

>>> blat_hsp2.hit_span_all # hə r bir fragmentin vuruş aralığı [18, 43] >>>

blat_hsp2.hit_inter_ranges # vuruş ardıcılığında müdaxilə edə n regionların başlangıç və son koordinatları [(54233122, 54264420)] s_span #splathi
blat_hsp2.hit_inter_ranges vuruş
ardıcılığında müdaxilə edə n bölgə lə r [31298]
```

Bu atributların əksəriyyəti əlimizdə olan PSL faylında mövcud deyil, lakin siz PSL fayınlını təhlil edərkən Bio.SearchIO onları sizin üçün tez hesablayır. Ona lazım olan hər bir fragmentin başlangıç və son koordinatlarıdır.

Sorğu, hit və aln atributları haqqında nə demək olar? HSP-də bir neçə fragment varsa, siz bu atributlardan istifadə edə bilməyə cəksiniz, çünki onlar yalnız bir SeqRecord və ya MultipleSeqAlignment obyektlərinin gətirirləri. Bununla belə, siz onların *_all həmkarlarından istifadə edə bilərsiniz: query_all, hit_all və aln_all. Bu xüsusiyətlə r HSP fragmentinin hər birindən SeqRecord və ya MultipleSeqAlignment obyektlərinin ehtiva edən siyahı qaytaracaq. Eyni şəkildə davranan başqa attributlar da var, yəni onlar yalnız bir fragmentli HSP-lər üçün işləyirlər. HSP [sənədlərinin](#) yoxlayın tam siyahı üçün.

Nəhayət, bir neçə fragmentinizin olub-olmadığını yoxlamaq üçün `is_fragmented` xassə sindən aşağıdakı kimi istifadə edə bilərsiniz:

```
>>> blat_hsp2.is_fragmented # BLAT HSP 2 fragmentla Doğrudur

>>> blat_hsp.is_fragmented # BLAT HSP əvvəlindən, bir fragmentla False
```

Davam etməzdənə və lə biləlisiniz ki, biz `QueryResult` və ya `Hit` obyektləri kimi HSP obyektlərinə diliq qeydindən istifadə edə bilərik. Bu notasiyadan istifadə etdiyiniz zaman sizə və zində obyekt modelinin sonuncu komponenti olan `HSPFragment` obyekti alacaqsınız.

11.1.4 HSPFragment

`HSPFragment` sorğu və hit ardıcılığı arasında tek, bitişik uygunluğu təmsil edir. Siz onu obyekt modelinin və axtarış nəticəsininə səsi hesab edə bilərsiniz, çünki axtarışınızın nəticələrinin olub-olmadığını müəyyən edən bu fragmentlərin olmasıdır.

Əksər hallarda, `HSPFragment` obyektləri ilə birbaşa məşğul olmaq məcburiyyətində deyilsiniz, çünki çoxlu ardıcılıq axtarış alətləri `HSP`-lərinin parçalarıdır. Onlarla məşğul olmaq məcburiyyətində olduğunuz zaman, yadda saxlamağınızın lazımlığı, `HSPFragment` obyektlərinin mümkün qədər yığacam olması üçün yazılmışdır. Əksər hallarda, onlar yalnız birbaşa ardıcılıqla əlaqəli attributları ehtivədir: zəncirlər, oxuçırcıvələr, molekul növləri, koordinatlar, ardıcılıqların özləri və onların identifikatorları və təsvirləri.

`HSPFragment`-də çapçı çağrıdığınız zaman bu attributlar asanlıqla göstərilir. Budur, götürülmüş bir nümunə BLAST axtarışımızdan:

```
>>> Bio import SearchIO- dan >>
blast_qresult = SearchIO.read("my_blast.xml", "blast-xml") >>> blast_frag = blast_qresult[0][0][0] #
ilk vuruş, ilk hsp, ilk fragment >>> çap (blast_frag)
```

Sorğu: 42291 mystery_seq
Hit: gi|262205317|ref|NR_030195.1 | Homo sapiens mikroRNA 520b (MIR520...)

Sorğu diapazonu: [0:61] (1)

Hit diapazonu: [0:61] (1)

Fraqmentlə r: 1 (61 sütun)

Sorğu - CCCTCTACAGGAAAGCGTTCTGTTCTGAAAGAAAAGTGCTTCCTTTAGAGGG

|||||||

Hit - CCCTCTACAGGAAAGCGTTCTGTTCTGAAAGAAAAGTGCTTCCTTTAGAGGG

Bu sə viyyə də , BLAT fragmenti BLAST fragmentinə olduqca bə nəzə yir, sorğu üçün qə naə t edin və vurun mövcud olmayan ardıcılıqlar:

```
>>> blat_qresult = SearchIO.read("my_blat.psl", "blat-psl") >>> blat_frag = blat_qresult[0][0][0]
# ilk vuruş, ilk hsp, ilk fragment >>> print(blat_frag)
```

Sorğu: mystery_seq <namə lum tə svir> Hit: chr19 <namə lum
tə svir> Sorğu diapazonu: [0:61] (1)

Hit diapazonu: [54204480:54204541] (1)

Fraqmentlə r: 1 (? sütun)

Bütün hallarda, bu atributlar sevimli nöqtə qeydimizdə n istifadə etmə klə ə ldə edilə bilə r. Bə zi nümunə lə r:

```
>>> blast_frag.query_start # sorğunun başlanğıc koordinatı 0
```

```
>>> blast_frag.hit_strand # vuruş ardıcılığı 1
```

```
>>> blast_frag.hit # vuruş ardıcılığı, SeqRecord obyekti kimi
```

```
SeqRecord(seq='CCCTCTACAGGAAAGCGTTCTGTTCTGAAAGAAAAGAAAGTGCTTCCTTT...GGG'), id='gi|262205317|ref
```

11.2 Standartlar və konvensiyalar haqqında qeyd

Əsas funksiyalara keçmə zdə nə və və l Bio.SearchIO-nun istifadə etdiyi standartlar haqqında bilmə li olduğunuz bir şey var. Bir neçə ardıcıl axtarış alə tlə ri işləmiş olsanız, hə r bir programın ardıcılıq koordinatları kimi şeylə rla mə şəkil olduğu bir çox fə rqli yollarla mə şəkil olmalı ola bilə rsiniz. Bu axtarış vasitə lə rının adə tə n öz standartları olduğundan xoş tə crübə olmaya bilə rdi. Mə sə lə n, bir alə t bir ə səsli koordinatlardan, digə ri isə sıfır ə səsli koordinatlardan istifadə edə bilə r. Yaxud, bir program strand mə nfi olarsa, başlanğıc və son koordinatlarını də yişdirə bilə r, digə rlə ri isə yox. Qısacısı, bunlar tez-tez meydana gə tirə n lazımsız qarşıqlıqlarla mübarizə aparılmalıdır.

Biz bu problemi özümüz də rk edirik və onu Bio.SearchIO-da hə ll etmə k niyyə tində yik. Bütün bunlardan sonra, Bio.SearchIO-nun mə qsə dlə rində n biri müxtə lif axtarış çıxış faylları ilə mə şəkil olmaq üçün ümumi, istifadə si asan interfeys yaratmaqdır. Bu, indicə gördüğünüz obyekt modelində n kə nara çıxan standartların yaradılması demə kdir.

İndi siz şikayə t edə bilə rsiniz: "Başqa standart deyil!". Yaxşı, nə hayə t, bu və ya digə r konvensiyani seçmə liyik, buna görə də bu lazımdır. Üstəlik, biz burada tamamilə yeni bir şey yaratmırıq; Python programçısı üçün ə n yaxşı hesab etdiyimiz standartı qə bul etmə k (hə r seyda n sonra bu, Biopython'dur).

Bio.SearchIO ilə işlə yə rkə n gözlə yə bilə cə yiniz üç gizli standart var:

- Birincisi ardıcılıq koordinatlarına aiddir. Bio.SearchIO-da bütün ardıcılıq koordinatları Python-un koordinat üslubuna uyğundur: sıfır ə səsli və yarı açıq. Mə sə lə n, BLAST XML çıkış faylında HSP-nin başlanğıc və son koordinatları 10 və 28-dirsə , Bio.SearchIO-da onlar 9 və 28-ə çevrilə cə k. Başlanğıc koordinatı 9 olur, çünkü Python indekslə ri sıfırdan başlayır, son koordinat isə 28 olaraq qalır, çünkü Python dilimlə ri intervalda sonuncu elementi buraxır.

- İkincisi ardıcılıq koordinatları üzrə dir. Bio.SearchIO-da başlanğıc koordinatları hə mişə son koordinatlardan kiçik və ya onlara bə rabə rdir. Bu, bütün ardıcılıq axtarış alə tlə rində hə mişə belə olmur, çünkü onların bə zilə ri ardıcılıq zolağı mə nfi olduqda daha böyük başlanğıc koordinatlarına malikdir.

- Sonuncu tel və oxu çə rçivə də yə rlə rində dir. Tellə r üçün yalnız dörd etibarlı seçim var: 1 (ə lavə zə ncir), -1 (mə nfi zə ncir), 0 (zülal ardıcılılığı) və Heç biri (telsiz). Çə rçivə lə ri oxumaq üçün etibarlı seçimlə r -3-də n 3-ə qə də r tam ə də dlə r və Yoxdur.

Qeyd edə k ki, bu standartlar yalnız Bio.SearchIO obyektlə rində mövcuddur. Ə gə r siz Bio.SearchIO obyektlə rini çıkış formatına yazsanız, Bio.SearchIO çıkış üçün formatın standartından istifadə edə cə k. O, öz standartını çıkış fayliniza mə cbur etmir.

11.3 Axtarış çıkış fayllarının oxunması

Bio.SearchIO obyektlə rində axtarış çıkış fayllarını oxumaq üçün istifadə edə bilə cə yiniz iki funksiya var: oxumaq və tə hlil etmə k. Onlar Bio.SeqIO və ya Bio.AlignIO kimi digə r alt modullarda oxumaq və tə hlil etmə k funksiyalarına mahiyyə tcə bə nza yırlə r. Hə r iki halda, axtarış çıkış faylinin adını və fayl formatının adını Python sə tirlə ri kimi tə qdim etmə lisiniz. Bio.SearchIO-nun tanıldığı format adlarının siyahısı üçün sə nə dlə ri yoxlaya bilə rsiniz.

Bio.SearchIO.read yalnız bir sorğu ilə axtarış çıkış fayllarını oxumaq üçün istifadə olunur və QueryResult obyektiini qaytarır. Ə vvə lki nümunə lə rimizdə istifadə olunan oxunduğuunu gördünüz. Görmə diyiniz şey budur ki, oxumaq fayl formatından asılı olaraq ə lava açar söz arqumentlə rini də qə bul edə bilə r.

Burada bə zi nümunə lə r var. Birincidə , biz BLAST cə dvə lli çıkış faylini oxumaq üçün ə və lki kimi oxudan istifadə edirik. İkincisində biz də yişdirmə k üçün açar söz arqumentində n istifadə edirik ki, o, BLAST cə dvə l variantını şə rhlə rlə tə hlil etsin:

```
>>> Bio import SearchIO - dan >>> qresult =
SearchIO.read("tab_2226_tblastn_003.txt", "blast-tab") >>> qresult QueryResult(id='gi|16080617|ref|NP_391444.1|',
3 baxış) >>> qresult SearchIO.read("tab_2226_tblastn_007.txt", "blast-tab",
şə rhlə r=True) >>> qresult2 QueryResult(id='gi|16080617|ref|NP_391444.1|', 3 baxış)
```

Bu açar söz arqumentlə ri fayl formatları arasında fə rqıla nr. Parserin davranışını də yişdirə n açar söz arqumentlə rinin olub olmadığını görmə k üçün format sə nə dlə rini yoxlayın.

Bio.SearchIO.parse-ə gə ldikdə , o, istə nilə n sayıda sorğu ilə axtarış çıkış fayllarını oxumaq üçün istifadə olunur. Funksiya hə r iterasiyada QueryResult obyekti verə n generator obyektiini qaytarır. Bio.SearchIO.read kimi, o da formata aid açar söz arqumentlə rini qə bul edir:

```
>>> Bio import SearchIO -dan >>> qresults
= SearchIO.parse("tab_2226_tblastn_001.txt", "blast-tab") >>> qnə tica də qresult üçün : print(qresult.id)

...
...
gi|16080617|ref|NP_391444.1| gi|
11464971:4-101 >>>
qresults2 = SearchIO.parse("tab_2226_tblastn_005.txt", "blast-tab", comments=True) >>> qresults2 -də qresult üçün : print(qresult.id)

...
...
random_s00 gi|
16080617|ref|NP_391444.1| gi|11464971:4-101
```

11.4 İndekslə mə ilə böyük axtarış çıxışı faylları ilə mə şəkil olmaq

Bə zən sizə təhlil etməli olduğunuz yüzlərlə və ya minlərlə sorğudan ibarət axtarış çıxışı faylı verilir. Sizə libəttə ki, bu faylı üçün Bio.SearchIO.parse-dən istifadə edə bilərsiniz, lakin sorğuların yalnız bir neçəsinə daxil olmaq lazımdırsa, bu, tamamilə səmərə siz olacaq. Bunun səbəbi, parse maraq sorğunuzu götürməzdənəvvəl görüyüb bütün sorğuları təhlil etdəkdir.

Bu halda, ideal seçim Bio.SearchIO.index və ya Bio.SearchIO.index_db istifadə edərək faylı indekslə şəhərməklərdir. Əgər adlar tanışsa slənir, bunun səbəbi siz onları evvəllər Bölmə [5.4.2-də görmüsünüz](#). Bu funksiyalar həmçinin formata xüsusi açar söz arqumentləri əlavə etməklə öz Bio.SeqIO analoqları ilə eyni şəkildə davranırlar.

Burada bəzi nümunələr var. İndeksdən yalnız faylı adı və format adı ilə istifadə edə bilərsiniz:

```
>>> Bio import SearchIO -dan
>>> idx = SearchIO.index("tab_2226_tblastn_001.txt", "blast-tab") >>> sorted(idx.keys()) ['gi|11464971:4-101', 'gi|16080617|ref|NP_391444.1|']
16080617|ref|NP_391444.1|"]
```

QueryResult(id='gi|16080617|ref|NP_391444.1|', 3 baxış) >>> idx.close()

Və ya həmçinin formata aid açar söz arqumenti ilə:

```
>>> idx = SearchIO.index("tab_2226_tblastn_005.txt", "blast-tab", comments=True) >>> çeşidləndi(idx.keys()) ['gi|11464971:4-101', 'gi|16080617|ref|NP_391444.1|"]
idx['gi|16080617|ref|NP_391444.1|']
```

QueryResult(id='gi|16080617|ref|NP_391444.1|', 3 baxış) >>> idx.close()

Və ya Bio.SeqIO-da olduğu kimi key_function arqumenti ilə:

```
>>> key_function = lambda id: id.upper() # düymələri böyük hərf edir >>> idx = SearchIO.index("tab_2226_tblastn_001.txt", "blast-tab", key_function=key_function) >>> sorted(idx.keys())
['GI|11464971:4-101', 'GI|16080617|REF|NP_391444.1|'] >>> idx["GI|16080617|REF|NP_391444.1|"]
```

QueryResult(id='gi|16080617|ref|NP_391444.1|', 3 baxış) >>> idx.close()

Bio.SearchIO.index_db indeks kimi işləyir, yalnız sorğu ofsetlərinin SQLite verilənləri bazası faylına yazır.

11.5 Axtarış çıxış fayllarının yazılması və konvertasiyası

Çıxış faylından axtarış nəticələrini manipulyasiya etmək və onu yenidən yeni fayla yaza bilmək bəzən faydalıdır. Bio.SearchIO sizə məhz bunu etməyi imkan verən yazma funksiyası təqdim edir. O, öz arqumentləri kimi təkrarlanan qaytarılan QueryResult obyektlərinin, yazmaq üçün çıxış faylı adını, yazmaq üçün format adını və isteğe bağlı olaraq bəzidən formata xas açar söz arqumentlərinin götürür. O, yazılın nömrəni və ya QueryResult, Hit, HSP və HSPFragment obyektlərinin dörd elementdən ibarət tuple qaytarır.

```
>>> Bio import SearchIO -dan
>>> qresults = SearchIO.parse("mirna.xml", "blast-xml") # XML faylini oxuyun >>> SearchIO.write(qresults, "results.tab", "blast-tab") # cədvəl faylına yazın (3, 239, 277, 277)
```

Qeyd etmə lisiniz ki, müxtə lif fayl formatları QueryResult, Hit, HSP və HSPFragment obyektlə rinin müxtə lif atributlarını tə lə b edir. Bu atributlar mövcud deyilsə , yazı işlə mə ya cə k. Başqa sözlə , siz hə mişə istə diyiniz çıxış formatına yaza bilmə zsiniz. Mə sə lə n, BLAST XML faylini oxusunuz, nə tıca lə ri PSL faylına yaza bilmə yə cə ksınız, çünki PSL faylları BLAST tə rə fində n hesablanmayan atributları tə lə b edir (mə sə lə n, tə krar uyğunluqların sayı). Hə qıqə tə n PSL-ə yazmaq istə yırsınızsa , bu atributları hə mişə ə l ilə tə yin edə bilə rsiniz.

Oxu, tə hlil, indeks və index_db kimi, yazma da formata aid açar söz arqumentlə rini qə bul edir. Yoxlayın Bio.SearchIO-nun yaza bilə cə yi formatlarının tam siyahısı və onların arqumentlə ri üçün sə nə dlə ri tə qdim edin. Nə hayə t, Bio.SearchIO hə m də sadə cə Bio.SearchIO.parse və Bio.SearchIO.write üçün qısa yol olan çevirmə funksiyasını tə min edir. Çevirmə funksiyasından istifadə edə rə k yuxarıdakı nümunə miz belə olardı:

```
>>> Bio import SearchIO- dan >>>
SearchIO.convert("mirna.xml", "blast-xml", "results.tab", "blast-tab") (3, 239, 277, 277)
```

Konvertasiya yazmaqdan istifadə etdiyinə görə , o, yalnız bütün tə lə b olunan atributlara malik format çevirme lə ri ilə mə hədudlaşır. Burada BLAST XML faylı BLAST cə dva l faylinin tə lə b etdiyi bütün standart də yə rlə ri tə min edir, ona görə də o, yaxşı işlə yir. Bununla belə , ilk növbə də tə lə b olunan atributları ə l ilə tə yin etmə lisiniz, çünki digə r format çevrilmə lə rının işlə mə ehtimalı azdır.

Fə sil 12

NCBI-nin Entrez verilə nlə r bazasına daxil olmaq

Entrez (<https://www.ncbi.nlm.nih.gov/Web/Search/entrezfs.html>) istifadə çılə rə NCBI-nin PubMed, GenBank, GEO və bir çox başqaları kimi verilə nlə r bazalarına çıxısını tə min edə n mə lumat axtarış sistemidir. Siz sorğuları ə l ilə daxil etmə k üçün veb-brauzerdə n Entrez-ə daxil ola bilə rsiniz və ya Entrez-ə programlı daxil olmaq üçün Biopython-un Bio.Entrez modulundan istifadə edə bilə rsiniz. Sonuncu, mə sə lə n, PubMed-də axtarış aparmağa və ya Python skripti daxilində GenBank qeydla rini endirma ya imkan verir.

Bio.Entrez modulu NCBI-nin <https://www.ncbi.nlm.nih.gov/books/NBK25501/> sə hifə sində ə trafli tə svir olunan sə kkiz alə tdə n ibarə t Entrez Proqramlaşdırma Utilitlə rində n (hə məcniñ EUtils kimi tanınır) istifadə edir. Bu alə tlə rin hə r biri aşağıdakı bölmə lə rdə tə svir olunduğu kimi Bio.Entrez modulunda bir Python funksiyasına uyğundur. Bu modul sorğular üçün düzgün URL-nin istifadə olunduguna və NCBI-nin mə lumatlara cavabdeh giriş üçün tə limatlarına ə mə l edildiyinə ə min olur.

Entrez Proqramlaşdırma Utilities tə rə fində n qaytarılan çıxış adə tə n XML formatında olur. Belə tə hlil etmə k çıxış üçün bir neçə seçiminiz var:

1. XML çıxışını Python obyektinə tə hlil etmə k üçün Bio.Entrez-in analizatorundan istifadə edin;
2. Python-un standart kitabxanasında mövcud olan XML analizatorlarından birini istifadə edin;
3. XML çıxışını xam mə tn kimi oxuyun və onu sə tir axtarışı və manipulyasiya yolu ilə tə hlil edin.

Python-un standart kitabxanasındaki XML analizatorlarının tə sviri üçün Python sə nə dlə rinə baxın. Burada biz Biopython-un Bio.Entrez modulunda tə hlilçini müzakirə edirik. Bu analizator Bio.Entrez-in programmatik çıkış funksiyaları vasitə silə Entrez-ə tə qdim edilə n mə lumatları tə hlil etmə k üçün istifadə oluna bilə r, hə m də faylda saxlanılan NCBI Entrez-də n XML mə lumatlarını tə hlil etmə k üçün istifadə edilə bilə r. Sonuncu haldə, Bio.Entrez-də XML analizatorunun düzgün işlə mə si üçün XML faylı ikili rejimda (mə sə lə n, open("myfile.xml", "rb")) açılmalıdır. Alternativ olaraq, siz faylin adını və ya yolunu XML faylına ötürə və Bio.Entrez-ə faylin açılması və bağlanması ilə mə şgul ola bilə rsiniz.

NCBI XML fayllarında olan mə lumatların strukturunu tə svir etmə k üçün DTD (Sə nə d Tipinin Tə rifi) fayllarından istifadə edir. NCBI tə rə fində n istifadə edilə n DTD fayllarının ə ksə riyyə ti Biopython paylanmasına daxildir. Bio.Entrez analizatoru NCBI Entrez tə rə fində n qaytarılmış XML faylini tə hlil edə rkə n DTD fayllarından istifadə edir.

Bə zə n müə yyə n bir XML faylı ilə ə laqə li DTD faylinin Biopython paylanmasında olmadığını görə bilə rsiniz. Xüsusilə , bu, NCBI DTD fayllarını yenilə dikdə baş verə bilə r. Bu baş verə rsa , Entrez.read çatışmayan DTD faylinin adı və URL ilə xə bə rdarlıq mesajı göstə rə cə k. Tə hlilçi internet vasitə silə çatışmayan DTD faylına daxil olmağa davam edə cə k və XML faylinin tə hlilini davam etdirmə yə imkan verə cə k. Bununla belə , DTD faylı yerli olaraq mövcuddursa, analizator daha sürə tli olur. Bu mə qsə dlə , lütfə n, DTD faylini xə bə rdarlıq mesajındakı URL-də n yükə yin və onu digə r DTD fayllarını ehtiva edə n ...site-packages/Bio/Entrez/DTDs qovluğuna yerlə şdirin. Ə gə r bu qovluğa yazma imkanınız yoxdursa, siz DTD faylini ~/.biopython/Bio/Entrez/DTD-ıə rə də yerlə şdirə bilə rsiniz, burada ~ ev kataloqunuza tə msil edir. Bu qovluq ...site-packages/Bio/Entrez/DTDs qovluğundan ə vvə l oxunduguna görə , ə gə r ...site-packages/Bio/Entrez/DTD-ıə rdə olanlar olarsa, DTD fayllarının daha yeni versiyalarını da yerlə şdirə bilə rsiniz.

köhnə lmişdir. Alternativ olaraq, Biopython-u mə nbə də n quraşdırınızsa, DTD faylini mə nbə kodunun Bio/Entrez/DTDs kataloquna ə lavə edə və Biopython-u yenidə n quraşdırı bilə rsiniz. Bu, yeni DTD faylini digə r DTD faylları ilə birlikdə düzgün yerə quraşdıracaq.

Entrez Proqramlaşdırma Utilitlə ri, hə mçinin Bölüm 12.13- də müzakirə olunan ardıcıl verilə nlə r bazası üçün Fasta və ya GenBank fayl formatları və ya ə də biyyat verilə nlə r bazası üçün MedLine formatı kimi digə r formatlarda çıxış yarada bilə r.

Bio.Entrez-də Entrez-ə proqramlı giriş üçün funksiyalar tə lə b olunan mə lumatların növündə n asılı olaraq mə lumatları ikili formatda və ya mə tn formatında qaytarır. Ə ksə r hallarda, bu funksiyalar kodlaşdırmanın UTF-8 olduğu fə rziyə si ilə NCBI Entrez-də n Python sə tirlə rinə alınan mə lumatların şifrə sini açaraq mə tn formatında mə lumatları qaytarır. Bununla belə , XML mə lumatları ikili formatda qaytarılır. Bunun sə bə bi kodlaşdırmanın XML sə nə dinin özündə göstə rilmə sidir, yə ni biz faylı tə hlil etmə ya başlayana qə də r düzgün kodlaşdırımı bilmə yə cə yik. Bio.Entrez-in tə hlilçisi buna görə də mə lumatları binar formatda qə bul edir, kodlaşdırımı XML-də n çıxarı və ondan XML sə nə dində ki bütün mə tri Python sə tirlə rinə deşifrə etmə k üçün istifadə edir və bütün mə tnin (xüsusilə inglis dilində n başqa dilla rdə) düzgün şə rh olunmasını tə min edir. Faylı tə hlil etmə k üçün Bio.Entrez-in analizatorundan istifadə etmə k istə diyiniz zaman XML faylini ikili rejimdə açmağınızın sə bə bi də budur.

12.1 Entrez Tə limatları

NCBI-nin onlayn resurslarına (Bio.Entrez və ya bə zi digə r modullar vasitə silə) daxil olmaq üçün Biopython-dan istifadə etmə zdə nə və I NCBI-nin Entrez İstifadə çi Tə lə blə rini oxuyun. NCBI sizin sistemlə rində n sui-istifadə etdiyinizi aşkar edə rsə , onlar sizin girişinizi qadağan edə bilə r və qadağa qoya bilə r!

İfadə etmə k üçün:

- 100-də n çox sorğu seriyası üçün bunu hə ftə sonları və ya ABŞ-dan kə narda edin. Bu bitdi sizə itəə t etmə k.
- <https://eutils.ncbi.nlm.nih.gov> ünvanından istifadə edin standart NCBI Veb ünvanı deyil. Biopiton bu internet ünvanından istifadə edir.
- Ə gə r siz API açarından istifadə edirsizsə , saniyə də e n çoxu 10 sorğu, ə ks halda saniyə də e n çoxu 3 sorğu edə bilə rsiniz. Bu avtomatik olaraq Biopython tə rə fində n hə yata keçirilir. Arqument siyahısına api_key="MyAPIkey" daxil edin və ya onu modul sə viyyə li də yişə n kimi tə yin edin:

```
>>> Bio import Entrez >>> Entrez.api_key
= "MyAPIkey"
```

- Problem olduqda NCBI-nin sizinlə e laqə saxlaması üçün ə lavə e-poçt parametrində n istifadə edin. Siz bunu açıq şə kildə Entrez-ə hə r zə ngdə parametr kimi tə yin edə bilə rsiniz(mə sə lə n, arqumentlə r siyahısına email="ANOther@example.com" daxil edin) və ya qlobal e-poçt ünvanı tə yin edə bilə rsiniz:

```
>>> Bio import Entrez >>> Entrez.email =
"ANOther@example.com"
```

Bio.Entrez daha sonra Entrez-ə hə r zə ngdə bu e-poçt ünvanından istifadə edə cə k. example.com ünvanı xüsusi olaraq sə nə dlə r üçün qorunan domen adıdır (RFC 2606). Lütfə n, tə sadüfi e-poçtdan istifadə etmə yin - ümumiyyə tlə e-poçt göndə rmə mə k daha yaxşıdır. E-poçt parametri 1 iyun 2010-cu il tarixində n mə cburidir. Hə ddində n artıq istifadə halında, NCBI E-utilitlə rə giriş bloklanmasından ə və I göstə rilə n e-poçt ünvanı ilə istifadə çi ilə e laqə saxlamağa çalışacaq.

- Ə gə r daha böyük program də stində Biopython istifadə edirsizsə , bunu müə yyə n etmə k üçün alə t parametrində n istifadə edin. Siz ya açıq şə kildə Entrez-ə hə r zə nglə alə t adını parametr kimi tə yin edə bilə rsiniz(mə sə lə n, arqumentlə r siyahısına tool="MyLocalScript" daxil edin) və ya qlobal alə t adını tə yin edə bilə rsiniz:

```
>>> Bio import Entrez >>> Entrez.tool = "
MyLocalScript"
```

Alə t parametri defolt olaraq Biopython olacaq.

- Böyük sorğular üçün NCBI hə məçin öz sessiya tarixçə si funksiyasından istifadə etməyi tövsiyə edir (WebEnv sessiyası kuki sə tri, Bölmə 12.16-a baxın). Bu, yalnız bir az daha müəkkəbdir.

Nəticə olaraq, istifadəsə viyyələrinizlə həssas olun. Çoxlu məlumat yükəlməyi planlaşdırırsınızsa, digər variantları nəzərdən keçirin. Məsələn, bütün insan genlərinə asanlıqla daxil olmaq istəyirsinizsə, hər bir xromosomu FTP vasitəsilə GenBank faylı kimi əldə etməyi və onları öz BioSQL verilənlər bazanızda iddal etməyi düşünün (bax: Bölmə 22.3).

12.2 EInfo: Entrez verilənlər bazası haqqında məlumat əldə etmək

EInfo NCBI verilənlər bazalarının hər biri üçün sahə indeksi müddətərinin sayılarını, son yeniləməni və mövcud bağlantıları təmin edir. Bundan əlavə, siz Entrez utilitəri vasitəsilə əldə edildə bilən bütün verilənlər bazası adlarının siyahısını əldə etmək üçün EInfo-dan istifadə edə bilərsiniz:

```
>>> from Bio import Entrez >>>
Entrez.email = "ANOOther@example.com" # Həmişə NCBI-ya kim olduğunuzu bildirin >>> axın = Entrez.einfo() >>>
nəticə = stream.read() >>> stream.close()
```

Dəyişən nəticə indi XML formatında verilənlər bazalarının siyahısını ehtiva edir:

```
>>> çap(nəticə) <?xml
version="1.0"?>
<!DOCTYPE eInfoResult PUBLIC "-//NLM//DTD eInfoResult, 11 may 2002//EN" "https://www.ncbi.nlm.nih.gov/
  entrez/query/DTD/eInfo_020511.dtd">
<eInfoResult>
<DbList>
  <DbName>nəşr olundu</DbName>
  <DbName>protein</DbName>
  <DbName>nukleotid</DbName>
  <DbName>nuccore</DbName>
  <DbName>nucgss</DbName>
  <DbName>nuceest</DbName>
  <DbName>struktur</DbName>
  <DbName>genom</DbName>
  <DbName>kitablar</DbName>
  <DbName>xərçəng xromosomları</DbName>
  <DbName>cdd</DbName>
  <DbName>böşləq</DbName>
  <DbName>domenlər</DbName>
  <DbName>gen</DbName>
  <DbName>genomeprj</DbName>
  <DbName>gensat</DbName>
  <DbName>coğrafi</DbName>
  <DbName>gds</DbName>
  <DbName>homologen</DbName>
  <DbName>jurnallar</DbName>
  <DbName>mesh</DbName>
  <DbName>ncbisearch</DbName>
```

```

<DbName>nlmcatalog</DbName>
<DbName>omia</DbName>
<DbName>omim</DbName>
<DbName>pmc</DbName>
<DbName>popset</DbName>
<DbName>probe</DbName>
<DbName>zülal qrupları</DbName>
<DbName>pcassay</DbName>
<DbName>pccompound</DbName>
<DbName>pcsubstance</DbName>
<DbName>snp</DbName>
<DbName>taksonomiya</DbName>
<DbName>ala t də sti</DbName>
<DbName>unigen</DbName>
<DbName>unists</DbName>
</DbList> </
eInfoResult>
```

Bu, kifayə t qə də r sadə XML faylı olduğundan, onun ehtiva etdiyi mə lumatları sadə cə sə tirlə çıxara bilə rik axtarış. Bunun ə və zinə Bio.Entrez-in tə hlilçisində n istifadə edə rə k, biz bu XML faylini birbaşa Python obyektiñə tə hlil edə bilə rik:

```
>>> Bio import Entrez >>> axın =
Entrez.einfo() >>> rekord = Entrez.read
(axın)
```

İndi qeyd tam bir açarı olan lüğə tdir:

```
>>> record.keys()
dict_keys(['DbList'])
```

Bu açıarda saxlanılan də yə rlə r yuxarıda XML-də göstə rilə n verilə nlə r bazası adlarının siyahısıdır:

```
>>> rekord["DbList"]
['pubmed', 'protein', 'nukleotid', 'nuccore', 'nucgss', 'nucest',
 'struktur', 'genom', 'kitablar', 'xə rçə ng xromosomları', 'cdd', 'boşluq', 'domenlə r', 'gen',
 'genomeprj', 'gensat', 'geo', 'gds', 'homologene', 'jurnallar', "'nlmcatalog', 'omia', 'omim', 'pmc',
 'popset', 'probe', 'proteinclusters', 'pcassay', 'pccompound', 'pcsubstance', 'snp', 'takonomy',
 "toolkigene"]
```

Bu verilə nlə r bazalarının hə r biri üçün biz daha çox mə lumat ə ldə etmə k üçün yenidə n EInfo-dan istifadə edə bilə rik:

```
>>> from Bio import Entrez >>>
Entrez.email = "ANOther@example.com" # Hə mişə NCBI-ya kim olduğunuzu bildirin >>> stream =
Entrez.einfo(db="pubmed") >>> rekord =
Entrez.read(stream) >>> rekord["DbInfo"]
["Tə svir"]
'PubMed bibliografiq qeyd'

>>> rekord["DbInfo"]["Count"]
'17989604'
>>> rekord["DbInfo"]["Son yenilə mə "]
'24/05/2008 06:45'
```

Bu qeyddə saxlanılan digə r mə lumatlar üçün record["DbInfo"].keys() cə hd edin. Ə n faydalı olanlardan biri ESearch ilə istifadə üçün mümkün axtarış sahə lə rinin siyahısıdır:

```
>>> qeyddə ki sahə üçün ["DbInfo"]["FieldList"]:
...     print("%(Name)s, %(FullName)s, %(Tə_svir)s" % sahə_si)
...
BÜTÜN, Bütün Sahə lər, Axtarla bilən bütün sahə lərdən bütün terminlər
UID, UID, nə şərətə yin edilmiş unikal nömrə
FILT, Filter, Qeydləri məhdudlaşdırır
TITL, Başlıq, Nəşrin başlığında sözlər
WORD, Mətn Word, Nəşrlərə laqəli sərbəst mətn
MESH, MeSH Şərtləri, Nəşr üçün təyin olunan Tibbi Mövzu Başlıqları
MAJR, MeSH Əsas Mövzu, MeSH şərtləri nəşr üçün böyükəhəmiyyət kəsb edir
AUTH, Müəllif, Nəşrin Müəllif(ləri).
JOUR, Journal, Jurnal nəşrin abbreviaturası
AFFL, Mənsubiyəti, Müəllifin institusional mənsubiyəti və ünvanı
...

```

Bu, uzun bir siyahıdır, lakin dolayısıyla bu, PubMed verilənlərinə rəsədi üçün müəllif sahəsində axtarış etmək üçün Jones[AUTH] və ya Sanger Mərkəzində müəllifləri məhdudlaşdırmaq üçün Sanger[AFFL] kimi işlər görə biləcəyiniz söyləyir. Bu çox lazımlı ola bilər - xüsusən də müəyyən bir verilənlərinə rəsədi ilə o qədər də tanış deyilsinizsə.

12.3 ESSearch: Entrez verilənlərinə rəsədi bazalarının axtarışı

Bu verilənlərinə rəsədi bazalarından hansı birini axtarmaq üçün biz Bio.Entrez.esearch() istifadə edirik. Mənşələn, başlığında Biopython olan nəşrləri PubMed-də axtaraq:

```
>>> from Bio import Entrez >>>
Entrez.email = "ANOther@example.com" # Həmişə NCBI-ya kim olduğunuzu bildirin >>> stream =
Entrez.esearch(db="pubmed", term="biopython[title]", retmax="40") >>> rekord = Entrez.read(stream) >>> "183"
rekord [4] "19"ist"
```

Doğrudur

```
>>> çap edin(rekord["IdList"]) ['22909249',
'19304878']
```

Bu çıkışda siz EFetch tərəfindən nə İdə edilə bilən PubMed ID-lərini (o cümlədən Biopython program qeydi üçün PMID olan 19304878) görürsünüz (12.6-ci bölümə yəxəbaxın).

Siz həmçinin GenBankı axtarmaq üçün ESearch-dən istifadə edə bilərsiniz. Burada Cypripedioideae səhəblərində matK geni üçün sürətlidə axtarış aparacaq (hər bir Entrez verilənlərinə rəsədi hansı sahələri axtara biləcəyinizi öyrənmək üçün EInfo haqqında Bölümə 12.2-yə baxın):

```
>>> axın = Entrez.esearch(
...     db="nukleotid", term="Cypripedioideae[Orgn] AND matK[Gene]", idtype="acc"
... )
>>> rekord = Entrez.read(axın) >>>
rekord["Saymaq"]
'348'
>>> qeyd["IdList"]
['JQ660909.1', 'JQ660908.1', 'JQ660907.1', 'JQ660906.1', ..., 'JQ660890.1']
```

ID-lərin hər biri (JQ660909.1, JQ660908.1, JQ660907.1, ...) GenBank identifikatorudur (Qoşulma nömrəsi). Bu GenBank qeydlərinə necə yükləmək barədə məlumat üçün Bölümə 12.6-a baxın.

Nəzərə alın ki, Cypripedioideae[Orgn] kimi növ adı və zinəsiz NCBI takson identifikatorundan istifadə edərək axtarışı məhdudlaşdırıla bilərsiniz, burada bu txid158330[Orgn] olacaq. Bu hal hazırda sənədləşdirilmə yib

ESearch yardım sə hifə si - NCBI bunu e-poçt sorğusuna cavab olaraq izah etdi. Siz tez-tez Entrez veb interfeysi ilə oynayaraq axtarış termininin formatını çıxara bilərsiniz. Mə sə lən, genom axtarışına tam [prop] daxil olmaqla, yalnız tamamlanmış genomları məhdudlaşdırır.

Son nümunə olaraq, hesablama jurnallarının adlarının siyahısını alaq:

```
>>> axin = Entrez.esearch(db="nlmcatalog", term="hesablamalı[jurnal]", retmax="20") >>> qeyd = Entrez.read(axin) >>> print("• hesablamalı jurnalı tapıldı".format(record["Count"])) 117
hesablamalı jurnalı çap olundu ( ilk çap >>2 " \n{} ".format(rekord["IdList"])) ['101660833', '101664671',
'101661657', '101659814', '101657941', '101657941',
'10165376', '10165378' '101649614', '101647835', '101639023', '101627224', '101647801',
'101589678', '101585369', '101585369', '10126' '101582229', '101574747', '101564639', '101671907']
```

Yenə də bu jurnal ID-lərinin hər biri üçün daha çox məlumat əldə etmək üçün EFetch-dən istifadə edə bilərik.

ESearch çox faydalı seçimlərə malikdir — [ESSearch yardım səhifəsinə](#) baxın əlavə məlumat üçün.

12.4 EPost: İdentifikatorların siyahısının yüklenməsi

EPost sonrakı axtarış strategiyalarında istifadə üçün UI siyahısını yükler yir; [EPost yardım səhifəsinə](#) baxın əlavə məlumat üçün.

`Bio.Entrez.epost()` funksiyası vasitəsilə Biopython-dan əldə edilə bilər.

Bunun nə vaxt faydalı olduğuna dair bir misal göstərmək üçün, EFetch istifadə edərək yüklenmək istədiyiniz identifikatorların uzun bir siyahısının olduğunu düşünək (bəlkə ardıcılıqlar, bəlkə sitatlar - hər hansı). EFetch ilə sorğu göndərdiyiniz zaman identifikatorların siyahısı, verilənlər bazası və s. hamısı serverə göndərilən uzun URL-ə çevrilir. Əgər identifikator siyahınız uzundursa, bu URL uzanır və uzun URL-lər pozulub bilər (məsələn, bəzi proksilər yaxşı öhdə sindən gəlmir).

Bunun əvəzinə, siz bunu iki mərhələ yəbənmək olar, əvvəlcə EPost-dan istifadə edərək ID-lərin siyahısını yüklenəyə bilərsiniz (bu, uzun URL problemini həll etmək üçün "HTML-əldə etmək" və "zincir", daxildə "HTML yazısı" istifadə edir). Tarixdə stəyi ilə siz daha sonra ID-lərin bu uzun siyahısına müraciət edərək EFetch ilə əlaqəli məlumatları endirə bilərsiniz.

EPost-un necə işlədiyini görmək üçün sadə bir nümunə yəbənzər - bəzi PubMed identifikatorlarını yüklenmək:

```
>>> from Bio import Entrez >>> Entrez.email =
"ANOther@example.com" # Həmişə NCBI-yə kim olduğunuzu bildirin >>> id_list = ["19304878", "18606172", "16403221",
"16377612", "14871861", "14871866"]3 " >>6 ", çap edin(Entrez.epost("pubmed", id=".join(id_list).read()")<?xml version="1.0"?>
```

```
<!DOCTYPE ePostResult PUBLIC "-//NLM//DTD ePostResult, 11 may 2002//EN" "https://www.ncbi.nlm.nih.gov/
entrez/query/DTD/ePost_020511.dtd">
<ePostResult>
    <QueryKey>1</QueryKey>
    <WebEnv>NCID_01_206841095_130.14.22.101_9001_1242061629</WebEnv>
</ePostResult>
```

Qaytarılan XML-ə iki mühüm mətədir, QueryKey və WebEnv daxildir və bunlar birlikdə tarix sessiyanızı müəyyən edir. EFetch kimi başqa bir Entrez çağrıları ilə istifadə etmək üçün bu dəyərləri çıxardınız:

```
>>> from Bio import Entrez >>>
Entrez.email = "ANOther@example.com" # Həmişə NCBI-yə kim olduğunuzu bildirin >>> id_list = ["19304878", "
18606172", "16403221", "16377612", "14871861", "14871861", "14871861", "14871861", "14871861", "14871861",
Entrez.read(Entrez.epost("pubmed", id=".join(id_list))) >>> webenv = search_results["WebEnv"] >>> query_key =
search_results["QueryKey"]
```

Bölmə [12.16](#) tarixçə funksiyasından necə istifadə olunacağını göstərir.

12.5 EXÜLASƏ : Ə sas identifikatorlardan xülasə lərin alınması

Esummary ə sas identifikatorlar siyahısından sənəd xülasə sinə alır (Esummary yardım səhifəsinə baxın əlavə məlumat üçün). Biopython-da ESummary Bio.Entrez.esummary() kimi mövcuddur. Xuxarıdakı axtarış nəticə sindən istifadə edərək, məsələn, ID 30367 olan jurnal haqqında daha çox məlumat əldə edə bilərək:

```
>>> from Bio import Entrez >>> Entrez.email
= "ANOther@example.com" # Həmişə NCBI-ya kim olduğunuzu bildirin >>> stream = Entrez.esummary(db="nlmcatalog",
id="101660833") >>> rekord = Entrez.read(stream) >>> info [0]["T"] = rekord["Mail cap ("Jurnal
məlumat\\nid: \nBaşlıq: ".format(seydi[0]["Id"]),
məlumat["Başlıq"]))
```

Jurnal məlumat

ID: 101660833

Başlıq: Hesablaşdırma təsviri üzrə IEEE əməliyyatları.

12.6 EFetch: Entrez-dən tam qeydlərin endirilir

EFetch, Entrez-dən tam qeydi əldə etmək istədiyiniz zaman istifadə etdiyiniz şəydir. Bu, ə sas EFetch Yardım səhifəsindən təsvir olunduğu kimi bir neçə mümkün verilənlərin bazasını əhatə edir.

Verilənlərin bazalarının əksəriyyəti üçün NCBI bir neçə fərqli fayl formatını daşınır. Bio.Entrez.efetch() istifadə edərək Entrez-dən xüsusi fayl formatının tələb edilməsi istəgə bağlı argumentlərin rettype və/və ya retmode təyin edilməsinə tələb edir. Fərqli birləşmələr rəsəd NCBI efetech vəbə hifə sindən əlaqələndirilmiş səhifələrdə hər bir verilənlərin bazası növü üçün təsvir edilmişdir.

Ümumi istifadə lərə rdən biri FASTA və ya GenBank/GenPept düz mətn formatlarında ardıcıllıqların endirilməsidir (sonra Bio.SeqIO ilə təhlil edilə bilər, Bölmə 5.3.1 və 12.6-a baxın). Xuxarıdakı Cypripedioideae nümunə sindən biz Bio.Entrez.efetch istifadə edərək GenBank qeydini EU490707 yükləyə bilərik:

```
>>> from Bio import Entrez >>> Entrez.email
= "ANOther@example.com" # Həmişə NCBI-ya kim olduğunuzu bildirin >>> stream = Entrez.efetch(db="nucleotide",
id="EU490707", rettype="gb", retmode="text") >>> print(stream).read
```

Locus EU490707 1302 bp Təsviri	Selenipedium aequinoctiale maturase K (matK) geni,	DNT	xətti PLN 26-İYUL-2016
qismən CD; xloroplast.			

QOŞULMA EU490707

VERSİYA	EU490707.1
---------	------------

AÇAR SÖZLƏR	.
-------------	---

MƏNBƏ	xloroplast Selenipedium aequinoctiale
-------	---------------------------------------

ORQANİZM	Selenipedium aequinoctiale Eukaryota; viridiplante; streptofitlər; embriyofitlər; traxeofitlər; Spermatofitlər; Magnoliopsida; Liliopsida; asparagales; Orchidaceae; Cypripedioideae; Selenipedium.
----------	---

ARAYIŞ 1 (ə sas 1-dən 1302-yə qədər)

Müəlliflər Neubig, KM, Whitten, WM, Carlsward, BS, Blanco, MA, Endara, L.,

Williams, NH və Moore, M.

TITLE Orkidələr rəsəd ycf1-in filogenetik faydası: matK-dan daha yüksək plastid gen

JOURNAL Bitki Sistemi. Təkmül. 277 (1-2), 75-84 (2009)

ARAYIŞ 2 (ə sas 1-dən 1302-yə qədər)

Müəlliflər Neubig, KM, Whitten, WM, Carlsward, BS, Blanco, MA,

Endara, CL, Williams, NH və Moore, MJ

TITLE Birbaşa Tə qdimat
 JURNAL Tə qdim Edildi (14 Fevral 2008) Botanika Departamenti, Florida Universiteti, 220 Bartram Hall, Gainesville, FL
 32611-8526, ABŞ Yer/Tə sdiqedicilə r 1..1302

XÜSUSİYYƏ TLƏ Rİ

mə nbe

```
/organism="Selenipedium aequinoctiale" /organelle="plastid:xloroplast" /
mol_type="genomic DNT" /specimen_voucher="FLAS:Blanco
2475" /db_xref="taxon:256374"
```

gen	<1..>1302 <gene="matk"></gene="matk">
CDS	<1..>1302 <gene="matk" <br=""></gene="matk"> codon_start=1 / transl_table=11 / product="maturase K" / protein_id="ACC99456.1" / translation="IFYEPVEIFGYDNKSSLVLVKRLITRMYQQNFISSVNDNSNQKG FWGHKHFFSSHFSQMVSEGFGVILEIPFSSQLVSSLEEKKIPKYQNLRSIHSIFPFL EDKFLHLNVSDLLLIPHPIHLEILVQILQCRIKDVPSSLHLLRLFHEYHNLSLTSK KFIYAFSKRKFLWLLYNSYVYCEYLFQFLRKQSSYLRTSSGVFLERTHLYVKIE HLLVVCCNSFQRILCFLKDPFMHYVRYQGKAILASKGTLLMKKWFHVNFWQSYPFH FWSQPYRIHIKQLSNYSFSFLGYFSSVLENHLVVRNQMLENSFIINLLTKKFDTIAPV ISLIGSLSKAQFCVTLGHPIISKPIWTDFSDILDRCRICRNLCRYHSGSSKKQVLY RIKYILRLSCARTLARKHKSTREETFMRLLGSSLI"

MƏ ŞKİL

1 atttttacg aacctgtgga aatttttgt tatgacaata aatctagttt agtacttg 61 aaacgttaa ttactcaat gtatcaacag aatttttgaa
tttcctcggt taatgattct 121 aacaaaaaa gattcattcggt ctcatcttc ttctcaaatg 181 gtatcagaag gttttggagt cattctggaa attccattct
cgtcgcaatt agtatcttc 241 cttagaaaaaa aaaaatacc aaaatcatg aatttacatg ctattcattttaat0aat1 acattttgaat tatgtgtcgat atctactaat
accccatccc 361 atccatctgg aatcttgg tcaaatctt caatgcggg tcaaggatgt tcattcttt 421 cattttatgc gattgcttt ccacaaat
cataatttgaat421 acgcctttc aaaaagaaag aaaagatcc ttgggtact atataattct 541 tatgtatag aatgcgaata tctattccag ttcttcgtat
aacagtcctt ttatcacg 601 tcaacatctt ctggagttt actaaat agaacatctt 661 ctatgtgtt gttgtatcc ttttcagagg atcttatgtt ttctcaagg
tccttcatg 721 cattatgtc gatatcaagg aaaagcaatt ctggctcaa agggactct tattctgtatgt 781 tggcaatctt attttactt ttggctcaa 841
ccgtatagga ttcatataaa gcaattatcc aacttccctt tcattttctt ggggtatctt 901 tcaagtgtac tagaaaatca ttggtagta agaaatcaaatcaa 841
ccgtatagga ttcatataaa tgactagaatcattcgtatcc atagccccatcc ttatcttctt tattggatca 1021 ttgtcgaaag ctcattttg tactgtatggatctt
tttagtaacc gatctggacc 1081 gatttctcggtt attctgatccgaaattt ctctttctt ggggtatctt 1141 tatcagcgcg gatcctcaa aaaacaggtt ttgtatcgta
taaaatataatcttccgactt 1201 tcgtgtgcta gaaccttggc acggaaacatc aaaagtgatc acgtacttgattt12 agaattctt atggaagaag aa

//
<BLANKLINE>

<BLANKLINE>

Nə zə rə alın ki, 2016-ci ilin oktyabr ayından etibarən GI identifikatorları qoşulma nömrə lə rının xeyrinə dayandırılıb. Siz hə lə də onların GI-yə ə saslanaraq ardıcılıqları ə ldə edə bilə rsiniz, lakin yeni ardıcılıqlara artıq bu identifikator verilmir. Bunun ə və zinə misalda göstə rildiyi kimi onlara "Qoşulma nömrə si" ilə müraciət etmə lisiniz.

rettype="gb" və retmode="text" arqumentləri bu qeydi GenBank formatında endirmə yə imkan verir.

Qeyd edək ki, 2009-cu il Pasxa bayramına qədər Entrez EFetch API sizə qayıdış növü kimi "genbank"dan istifadə etmə yə imkan verir, lakin NCBI indi internetdə təsvir olunduğu kimi "gb" və ya "gbwithparts" (və ya zülallar üçün "gp") rəsmi qaytarma növlə rindən istifadə etmə kdə israrlıdır. Onu da qeyd edək ki, 2012-ci ilin fevralına qədər Entrez EFetch API standart olaraq düz mətn fayllarını qaytarırı, lakin indi defolt olaraq XML-dir.

Alternativ olaraq, məsələn, Fasta formatını əldə etmək üçün rettype="fasta" istifadə edə bilərsiniz; [EFetch Sequences Yardım səhifəsinə](#) baxın digər variantlar üçün. Unutmayın - mövcud formatlar hansı verilənlər bazasından endirdiyinizdən asıldır - əsas EFetch Yardım səhifəsinə baxın.

Əgər siz qeydi Bio.SeqIO tərəfindən qəbul edilən formatlardan birində əldə etsem niz (Fəsil 5-ə baxın), siz birbaşa onu SeqRecord-a təhlil edin:

```
>>> Bio import SeqIO >>> from Bio
import Entrez >>> Entrez.email =
"ANOther@example.com" # Həmişə NCBI-ye kim olduğunuzu bildirin >>> axın = Entrez.efetch(db="nucleotide",
id="EU490707", rettype="gb", retmode="textstrebanks") >>> ad "rekord "(rekord = >>>re). stream.close() >>> print(record.id)
```

EU490707.1 >>>

```
çap (record.name)
EU490707 >>>
```

```
çap (rekord.təsvir)
Selenipedium aequinoctiale maturase K (matK) geni, qismən CD; xloroplast >>> print(len(record.features)) 3 >>> record.seq
```

Seq('ATTTTTACGAACCTGTGAAATTTGGTTATGACAATAATCTAGTTAGTA..GAA')

Qeyd edək ki, daha tipik istifadə ardıcılıq məlumatlarını yerli faylda saxlamaq və sonra onu Bio.SeqIO ilə təhlil etmək olardı. Bu, skriptiniz üzərində işləyərkən eyni faylı təkrar-təkrar yükləmək məkmətburjuyətindən xilas edə bilər və NCBI serverlərinə daha az yük verir. Məsələn:

Bio- dan os
idxalı Bio idxal Entrez - dən
SeqIO idxal

```
Entrez.email = "ANOther@example.com" # Həmişə NCBI-ye kim olduğunuzu bildirin = "EU490707.gbk"
əgər os.path.isfile(filename) deyilsə :
# Yüklə nəzər... axın = Entrez.efetch(db="nucleotide",
id="EU490707", "b" retty
= "tt" output ", b open(fayl adı, "w") output.write(stream.read()) output.close() stream.close() print("Saved")
```

çap ("Ayışma...")

```
rekord = SeqIO.read (fayl adı, "genbank") çap (qeyd)
```

Bio.Entrez.read() funksiyasından istifadə edə rə k tə hlil edə bilə cə yiniz XML formatında çıxış a ldə etmə k üçün retmode="xml" istifadə edin:

```
>>> from Bio import Entrez
Entrez.email = "ANOther@example.com" # Həmişə NCBI-ya kim olduğunuzu bildirin >>> axın =
Entrez.efetch(db="nucleotide", id="EU490707", retmode="xml") >>> rekord = Entrez.read(close) (close) >>> > axın .
rekord[0]["GBSeq_definition"]
```

'Selenipedium aequinoctiale maturase K (matK) geni, qismə n CD; xloroplast' >>> rekord[0]["GBSeq_source"] 'xloroplast
Selenipedium aequinoctiale'

Bələ liklə , ardıcılıqla mə şgul olurdu. Digə r verilə nlə r bazalarına xas olan fayl formatlarının tə hlili nümunə lə ri üçün (mə sə lə n, PubMed-də istifadə olunan MEDLINE formatı) Bölmə 12.13-a baxın.

Bio.Entrez.esearch() ilə axtarış aparmaq və sonra qeydlə ri yüklə mə k istə yirsinizsə Bio.Entrez.efetch(), siz WebEnv tarixçə si funksiyasından istifadə etmə lisiniz – Bölmə 12.16-a baxın.

12.7 ELink: NCBI Entrez-də ə laqə li elementlə rin axtarışı

Biopython-dan Bio.Entrez.elink() kimi a ldə olunan ELink NCBI Entrez verilə nlə r bazasında ə laqə li elementlə ri tapmaq üçün istifadə edilə bilə r. Mə sə lə n, gen verilə nlə r bazasında bir giriş üçün nukleotid girişlə rini və digə r gözə l şeylə ri tapmaq üçün bunu bizə verə bilə rsiniz.

Bioinformatikada də rc olunan Biopython tə tbiqi qeydinə aid mə qalə lə ri tapmaq üçün ELink-də n istifadə edə k. 2009. Bu mə qalə nin PubMed ID-si 19304878-dir:

```
>>> from Bio import Entrez
Entrez.email = "ANOther@example.com" # Həmişə NCBI-yə kim olduğunuzu bildirin >>> pmid = "19304878" >>>
rekord =
Entrez.read(Entrez.elink(dbfrom="pubmed", id=pmid))
```

Qeyd də yişə ni axtarış etdiyimiz hə r bir verilə nlə r bazası üçün bir Python siyahısından ibarə tdir. Axtarmaq üçün yalnız bir PubMed ID-si tə yin etdiyimiz üçün qeyddə yalnız bir element var. Bu element bizim axtarış terminimiz, elə cə də tapılmış bütün ə laqə li maddə lə r haqqında mə lumatları özündə cə mlə şdirə n lüğə tdir:

```
>>> rekord[0]["DbFrom"] 'pubmed'
>>> rekord[0]
["IdList"] ['19304878']
```

"LinkSetDb" düymə si hə r bir hə də f verilə nlə r bazası üçün bir elementdə n ibarə t siyahı kimi saxlanılan axtarış nəticə lə rini ehtiva edir. Axtarış nəticə lə rimizdə biz yalnız PubMed verilə nlə r bazasında hitlə ri tapırıq (kateqoriyalara bölünsə k də):

```
>>> len(record[0]["LinkSetDb"])
8
```

Zamanla də qıq rə qə mlə r artmalıdır:

```
>>> qeyddə linksetdb üçün [0]["LinkSetDb"]: çap(linksetdb["DbTo"],
...     linksetdb["LinkName"], len(linksetdb["Link"]))
...
pubmed pubmed_pubmed 284
pubmed pubmed_pubmed_alsoviewed 7 pubmed
pubmed_pubmed_citedin 926 pubmed
pubmed_pubmed_combined 6 pubmed
pubmed_pubmed_five 6 pubmed
pubmed_pubmed_refs 12view pubmed
pubmed_pubmed_reviews_five 6
```

Faktiki axtarış nə tıçə lə ri "Link" düymə si altında saxlanılır.
İndi ilk axtarış nə tıçə sinə baxaq:

```
>>> rekord[0]["LinkSetDb"][0]["Link"][0]
{'Id': '19304878'}
```

Bu, axtardığımız mə qalə dir, bizə çox kömə k etmir, ona görə də ikinci axtarış nə tıçə sinə baxaq:

```
>>> rekord[0]["LinkSetDb"][0]["Link"][1]
{'Id': '14630660'}
```

PubMed ID 14630660 olan bu mə qalə Biopython PDB analizatoru haqqındadır.

Bütün PubMed ID-lə rini çap etmə k üçün bir döngə də n istifadə edə bilə rik:

```
>>> qeyddə ki keçid üçün [0]["LinkSetDb"][0]["Link"]: çap(link["Id"])
...
...
19304878
14630660
18689808
17121776
16377612
12368254
....
```

İndi bu gözə lidi, amma şə xsə n mə n bir mə qalə yə istinad edilib-edilmə diyini öyrə nmə k üçün daha çox maraqlanıram. Yaxşı, ELink bunu da edə bilə r - e n azı Pubmed Central-dakı jurnallar üçün (bax: Bölüm 12.16.3).

ELink-də kömə k üçün [ELink yardım sə hifə sinə baxın](#). Yalnız link adları üçün bütöv bir alt sə hifə var, tə svir edə n müxtə lif verilə nlə r bazalarına necə çarpez istinad edilə bilə r.

12.8 EGQuery: Qlobal Sorğu - axtarış şə rtlə ri üçün hesablanır

EGQuery, Entrez verilə nlə r bazalarının hə r birində (yə ni qlobal sorğu) axtarış termini üçün hesablamaları tə min edir. Bu, ESSearch ilə faktiki olaraq çoxlu ayrıca axtarışlar etmə də n axtarış şə rtlə rinizin hə r bir verilə nlə r bazasında neçə element tapacağını öyrə nmə k üçün xüsusilə faydalıdır (aşağıda 12.15.2- də ki nümunə yə baxın).

Bu misalda biz Bio.Entrez.egquery()-də n "Biopython" üçün sayıları a ldə etmə k üçün istifadə edirik:

```
>>> from Bio import Entrez >>>
Entrez.email = "ANOther@example.com" # Həmişə NCBI-ya kim olduğunuzu bildirin >>> stream =
Entrez.egquery(term="biopython") >>> rekord = Entrez.read(stream)
```

```
>>> rekorddakı sıra üçün ["eGQueryResult"]:
...     çap(sə tir["DbName"], sira["Saymaq"])
...
nə şr 6 pmc
62 jurnal
0
...
```

EGQuery yardım sə hifə sinə baxın ə lavə mə lumat üçün.

12.9 ESpell: Orfoqrafiya tə kliflə rinin alınması

ESpell orfoqrafiya tə kliflə rini alır. Bu misalda biz Bio.Entrez.espell()-də n istifadə edə rə k Biopython-un düzgün yazılışını ə ldə edirik:

```
>>> from Bio import Entrez >>>
Entrez.email = "ANOther@example.com" # Həmişə NCBI-ya kim olduğunuzu bildirin >>> stream =
Entrez.espell(term="biopythoон") >>> rekord = Entrez.read(stream)
>>> rekord["Query"] 'biopythoон' >>>
recorded["Cory"]
```

ESpell yardım sə hifə sinə baxın ə lavə mə lumat üçün. Bunun ə sas istifadə si axtarış şərtlə ri üçün avtomatik tə kliflə r vermə k üçün GUI alə tlə ri üçündür.

12.10 Böyük Entrez XML fayllarının tə hlili

Entrez.read funksiyası Entrez tə rə fində n qaytarılmış bütün XML faylı yaddaşa saxlanılan vahid Python obyektinə oxuyur. Yaddaşa sığmayacaq qədər böyük olan Entrez XML fayllarını tə hlil etmə k üçün siz Entrez.parse funksiyasından istifadə edə bilə rsiniz. Bu, XML faylındakı qeydlə ri bir-bir oxuyan generator funksiyasıdır. Bu funksiya yalnız XML faylı Python siyahısı obyektiñi e ks etdiirdikdə faydalıdır (başqa sözlə , sonsuz yaddaş resurslarına malik kompüterdə Entrez.read Python siyahısını qaytararsa).

Mə sə lə n, verilmiş orqanizm üçün bütün Entrez Gene verilə nlə r bazasını NCBI-nin ftp saytından fayl kimi yükə yə bilə rsiniz. Bu fayllar çox böyük ola bilə r. Nümunə olaraq, 4 sentyabr 2009-cu ildə insan üçün Entrez Gene verilə nlə r bazasını ehtiva edə n Homo_sapiens.ags.gz faylinin ölçüsü 116576 kB idi. ASN formatında olan bu fayl NCBI-nin gene2xml programından istifadə edə rə k XML faylına çevrilə bilə r (ə traflı mə lumat üçün NCBI-nin ftp saytına baxın):

```
$ gene2xml -b T-i Homo_sapiens.ags -o Homo_sapiens.xml
```

Nə ticə XML faylinin ölçüsü 6,1 GB-dır. Bu faylda Entrez.read cə hdi a ilə nə ticə lə nə cə k Bir çox kompüterdə MemoryError.

Homo_sapiens.xml XML faylı hə r biri insanda bir Entrez geninə uyğun gə lə n Entrez gen qeydlə rinin siyahısından ibarə tdir. Entrez.parse bu gen qeydlə rini bir-bir ə ldə edir. Daha sonra qeydlə r üzə rində tə karrlamaqla hə r bir qeyddə müvafiq mə lumati çap edə və ya saxlaya bilə rsiniz. Mə sə lə n, bu skript Entrez geni üzə rində tə karlanır və bütün cari genlə r üçün gen nömrə lə rini və adlarını çap edir:

```
>>> from Bio import Entrez >>>
Entrez.email = "ANOther@example.com" # Həmişə NCBI-ye kim olduğunuzu bildirin >>> stream =
open("Homo_sapiens.xml", "rb") >>> records =
Entrez.parse(stream)
```

Alternativ olaraq istifadə edə bilərsiniz

```
>>> qeydlə r = Entrez.parse("Homo_sapiens.xml")
```

və Bio.Entrez faylin açılması və bağlanması ilə məşğul olsun. Bu daha təhlükəsizdir, çünkü fayl təhlil edildikdən sonra və ya xəta baş verərsə avtomatik olaraq bağlanacaq.

>>> qeydlə r üçün :

```
...     status = rekord["Entrezgene_track-info"]["Gene-track"]["Gene-track_status"] if status.attributes["value"] ==
...     "dayandırılıb":
...         davam et
...
...     geneid = rekord["Entrezgene_track-info"]["Gene-track"]["Gene-track_geneid"] genadı = qeyd["Entrezgene_gene"]
...     ["Gene-ref"]["Gene-ref_locus"] çap(geneid, genadı)
...
...
1 A1BG
2 A2M
3 A2MP
8 AA
9 NAT1
10 NAT2
11 AACP
12 SERPINA3
13 AADAC
14 AAMP
15 AANAT
16 AARS
17 AAVS1
...
```

12.11 HTML qaçış simvolları

Nəşr edilmiş qeydlər, məsələn, alt yazıları, yuxarı yazıları və ya kursiv mətnləri, habelə MathML vasitəsilə riyazi simvolları göstərmək üçün HTML teqlərindən ibarət ola bilər. Varsayılan olaraq, Bio.Entrez təhlilçisi bütün mətni işaretləmədən düz mətn kimi qəbul edir; məsələn, Pubmed qeydinin abstraktindəki " $P < 0.05$ " fragmenti kimi kodlanmışdır.

`<i>P</i> < 0.05`

Entrez tərəfindən qaytarılan XML-də Python sətirinə çevrilir

`'<i>P</i> < 0,05'`

Bio.Entrez təhlilçisi tərəfindən. Bu, daha çox insan tərəfindən oxuna bilməolsa da, kiçik işaret yəgənə etibarlı HTML deyil və mətnin, məsələn, HTML analizatoru tərəfindən sonrakı işlənməsinə qeyri-mümkün edir. Təhlilçi tərəfindən qaytarılan bütün sətirlərin etibarlı HTML olduğundanəmin olmaq üçün qaçış argumentini True olaraq təyin etməklə Entrez.read və ya Entrez.parse-a zəng edin:

```
>>> rekord = Entrez.read (axın, escape=Doğru)
```

Sonra təhlilçi HTML-də icazə verilməyən bütün simvolları HTML-dən qaçan ekvivalenti ilə əvəz edəcək; yuxarıdakı misalda parser yaradacaq

`'<i>P</i> < 0,05'`

etibarlı HTML fragmentidir. Varsayılan olaraq escape False-dir.

12.12 Sə hvlə rin idarə edilmə si

Fayl XML faylı deyil

Mə sə lə n, Fasta faylini XML faylı kimi tə hlil etmə ya çalışdığınız zaman bu xə ta baş verir:

```
>>> Bio import Entrez >>> stream =
open("NC_005816.fna", "rb") # a Fasta faylı >>> rekord = Entrez.read(stream)
```

Traceback (ə n son zə ng):

...

Bio.Entrez.Parser.NotXMLError: XML mə lumatını tə hlil etmə k alınmadı (sintaksis xə tası: sə tir 1, sütun 0). Xahiş edirə m ana

Burada tə hlilçi XML faylinin başlamalı olduğu <?xml ... teqini tapmadı və buna görə də faylin XML faylı olmadığına qərar verdi (düzgün).

Fayl vaxtından ə vvə l bitir və ya başqa cür zə də lə nib Fayliniz XML

formatında olduqda, lakin zə də lə ndikdə (mə sə lə n, vaxtından ə vvə l bitirmə klə), tə hlilçi CorruptedXMLError-u qaldıracaq.

Budur vaxtından ə vvə l bitə n bir XML faylı nümunə si:

```
<?xml version="1.0"?>
<!DOCTYPE eInfoResult PUBLIC "-//NLM//DTD eInfoResult, 11 may 2002//EN" "https://www.ncbi.nlm.nih.gov/e
<eInfoResult>
<DbList>
    <DbName>nə şr olundu</DbName>
    <DbName>protein</DbName>
    <DbName>nukleotid</DbName>
    <DbName>nuccore</DbName>
    <DbName>nucgss</DbName>
    <DbName>nuceest</DbName>
    <DbName>struktur</DbName>
    <DbName>genom</DbName>
    <DbName>kitablar</DbName>
    <DbName>xə rçə ng xromosomları</DbName>
    <DbName>cdd</DbName>
```

aşağıdakı geri izlə mə yaradacaq:

>>> Entrez.read(axın)

Traceback (ə n son zə ng):

...

Bio.Entrez.Parser.CorruptedXMLError: XML mə lumatını tə hlil etmə k alınmadı (heç bir element tapılmadı: sə tir 16, sütun 0)

Qeyd edə k ki, xə ta mesajı XML faylinin hansı nöqtə sində xə tanın aşkar edildiyini bildirir.

Faylda ə laqə li DTD-də çatışmayan elementlə r var

Bu, müvafiq DTD faylinda tə sviri olmayan etiketlə ri ehtiva edə n XML faylinin nümunə sidir:

```
<?xml version="1.0"?> <!DOCTYPE
eInfoResult PUBLIC "-//NLM//DTD eInfoResult, 11 may 2002//EN" "https://www.ncbi.nlm.nih.gov/e
```

```

<eInfoResult>
    <DbInfo>
        <DbName>nə şr olundu</DbName>
        <MenuName>PubMed</MenuName>
        <Tə svir>PubMed bibliografik qeydi</ Tə svir>
        <Count>20161961</Count>
        <Son Yenilə mə >10/09/2010 04:52</Son Yenilə mə >
        <Sahə siyahısı>
            <Sahə >
        ...
        </Sahə >
    </FieldList>
    <Sə nə d Siyahısı>
        <Sə nə d>
            <DsName>Yayım tarixi</DsName>
            <DsType>4</DsType>
            <DsTypeName>string</DsTypeName>
        </Sə nə d>
        <Sə nə d>
            <DsName>EPubDate</DsName>
        ...
    </DbInfo> <
eInfoResult>

```

Bu faylda nə də nsə <DocsumList> teqि (və bir neçə başqaları) ikinci sə tirdə bu XML faylı üçün DTD kimi göstə rilə n eInfo_020511.dtd DTD faylında qeyd edilmə yib. Defolt olaraq, DTD-də bə zi teq tapa bilmə dikdə tə hlilçi dayandıracaq və ValidationException-u qaldıracaq:

```

>>> Bio import Entrez >>> stream =
open("einfo3.xml", "rb")>>> rekord = Entrez.read
(stream)
Traceback (ə n son zə ng):
...
Bio.Entrez.Parser.ValidationError: DTD-də 'DocsumList' teqini tapmaq alınmadı. Bütün teqlə ri keçmə k üçün

```

İsteğe bağlı olaraq, tə hlilçiyə ValidationError yaratmaq ə və zinə belə teqlə ri atlamağı tapşırı bilə rsiniz. Bu, Entrez.read və ya Entrez.parse arqumentini doğrulamaqla False-ə bə rabə r zə ng etmə klə edilir:

```

>>> Bio import Entrez >>> stream =
open("einfo3.xml", "rb")>>> rekord =
Entrez.read(stream, validate=False)>>> stream.close()

```

Ə lbə ttə ki, DTD-də olmayan XML teqlə rində olan mə lumatlar Entrez.read tə rə fində n qaytarılan qeyddə yoxdur.

Faylda xə ta mesajı var

Bu, mə sə lə n, mövcud olmayan PubMed ID-si üçün PubMed qeydinə daxil olmağa cə hd etdiyiniz zaman baş verə bilə r. Varsayılan olaraq, bu RuntimeError-u qaldıracaq:

```

>>> from Bio import Entrez >>>
Entrez.email = "ANOther@example.com" # Hə mişə NCBI-yə kim olduğunuzu bildirin

```

```
>>> axin = Entrez.esummary(db="pubmed", id="99999999") >>> rekord = Entrez.read(axin)
```

Traceback (a n son zə ng):

...

RuntimeError: UID=99999999: sə nə d xülasə sini ə ldə etmə k mümkün deyil

Ə gə r siz çoxlu PubMed qeydlə rinə daxil olursunuzsa, PubMed ID-lə rində n biri sə hv olarsa, RuntimeError PubMed qeydlə rində n hə r hansı biri üçün nə ticə lə rə ldə etmə yinizə mane olacaq. Bunun qarşısını almaq üçün ignore_errors arqumentini True olaraq tə yin edə bilə rsiniz. Bu, etibarlı PubMed ID-lə ri üçün tə lə b olunan nə ticə lə ri və sə hv ID üçün ErrorElementi qaytaracaq:

```
>>> from Bio import Entrez >>> Entrez.email
= "ANOther@example.com" # Hə mişə NCBI-ya kim olduğunuzu söylə yin >>> stream = Entrez.esummary(db="pubmed",
id ="19304878,99999999,31278684") >>> record = Entrez. len (rekord)
```

3

```
>>> qeyd[0]. 'Sə nə dYə mimi'
etiketi
```

```
>>> rekord[0]["Başlıq"]
```

'Biopython: hesablama molekulyar biologiya və bioinformatika üçün sə rbə st mövcud Python alə tlə ri.' >>> qeyd[1]. 'SƏ HV' etiketi

```
>>> rekord[1]
```

ErrorElement('UID=99999999: sə nə din xülasə sini ə ldə etmə k mümkün deyil') >>> rekord[2].tag
'DocSum'

```
>>> qeyd[2]["Başlıq"]
```

'Programlaşdırma Resurslarının Bio* Layihə lə ri Arasında Paylaşılması.'

12.13 İxtisaslaşdırılmış tə hlilçilə r

Bio.Entrez.read() funksiyası Entrez tə rə fində n qaytarılan XML çıkışlarının ə ksə riyyə tini (hamısı olmasa da) tə hlil edə bilə r. Entrez adə tə n başqa formatlarda qeydlə ri ə ldə etmə yə imkan verir ki, bu da oxunaqlılıq (və ya yükə mə ölçüsü) baxımından XML formatı ilə müqayisə də bə zi üstünlükə rə malik ola bilə r.

Bio.Entrez.efetch()-də n istifadə edə rə k Entrez-də n xüsusi fayl formatını tə lə b etmə k üçün tə krar tip və /və ya retmode isteğə bağlı arqumentlə rin tə yin edilmə si tə lə b olunur. Fə rqli birlə şmə lə r NCBI efetech vəb sə hifə sində hə r bir verilə nlə r bazası növü üçün tə svir edilmişdir.

Aşkar hallardan biri odur ki, siz ardıcılığı FASTA və ya GenBank/GenPept düz mə tn formatlarında yükə mə yə üstünlük verə bilə rsiniz (bunlar sonra Bio.SeqIO ilə tə hlil edilə bilə r, Bölmə 5.3.1 və 12.6-a baxın). Ə də biyyat verilə nlə r bazası üçün Biopython PubMed-də istifadə olunan MEDLINE formatı üçün tə hlilçi ehtiva edir.

12.13.1 Medline qeydlə rinin tə hlili

Medline analizatorunu Bio.Medline-da tapa bilə rsiniz. Tutaq ki, biz bir Medline qeydində n ibarə t pubmed_result1.txt faylini tə hlil etmə k istə yirik. Bu faylı Biopython-un Tests\Medline kataloqunda tapa bilə rsiniz. Faylı belə görünür:

PMID- 12230038

ÖZ - NLM

STAT-MEDLINE

DA - 20020916

DCOM- 20030606

LR - 20041117

PUBM - Çap et

IS - 1467-5463 (Çap)

VI - 3

IP - 3

DP - 2002 Sentyabr TI -

Bio* alə t də stlə ri - qısa icmal.

PG - 296-302

AB - Bioinformatika tə dqiqatını kommersiya program tə minati ilə etmə k çox vaxt çə tindir. The

Açıq Mə nbə BioPerl, BioPython və Biojava layihə lə ri alə t də stlə ri ilə tə min edir

...

Əvvəlcə faylı açırıq və sonra təhlil edirik:

```
>>> Bio import Medline -dan >>> yayım  
olaraq açıq("pubmed_result1.txt") ilə : rekord = Medline.read(axın)  
...  
...
```

Rekord indi Python lügəti kimi Medline rekordunu ehtiva edir:

```
>>> qeyd["PMID"] '12230038'
```

```
>>> qeyd["AB"]
```

"Bioinformatika tə dqiqatını kommersiya program tə minati ilə etmə k çox vaxt çə tindir.

Açıq Mənbəli BioPerl, BioPython və Biojava layihə ləri fərdiləşdirilmiş boru kəmərləri və ya təhlillər yaratmağı asanlaşdırınan çoxsaylı funksionallığı olan alətdə stləri təqdim edir. Bu icmalda əsas dillərin qəribəlikləri və bio analoqlarının funksionallığı, sənədləri, faydalılılığı və nisbi üstünlükleri, xüsusən başlanğıc bioloq programçının nöqtəyi-nəzərində n qısa şəkildə müqayisə edilir.'

Medline qeydində istifadə olunan əsas adlar olduqca qaranlıq ola bilər; istifadə edin

```
>>> kömək(qeyd)
```

qısa xülasə üçün.

Coxlu Medline qeydləri olan faylı təhlil etmək üçün bunun əvəzinə təhlil funksiyasından istifadə edə bilərsiniz:

```
>>> Bio import Medline -dan >>> ilə  
open("pubmed_result2.txt") ilə axın olaraq : Medline.parse(stream)  
... -da qeyd üçün : print(record["TI"])  
...  
...
```

SCOP və ASTRAL üçün yüksək səviyyəli interfeys python-da həyata keçirilir.

GenomeDiagram: geniş məqyaslı genomik məlumatların vizuallaşdırılması üçün python paketi.

Açıq mənbəli klaster programı.

PDB faylı analizatoru və struktur sinifi Python-da həyata keçirilir.

Fayllarda saxlanan Medline qeydlərinə təhlil etmək əvəzinə siz həmçinin tərəfindən endirilmiş Medline qeydlərinə təhlil edə bilərsiniz Bio.Entrez.efetch. Məsələn, PubMed-də Biopython ilə əlaqəli bütün Medline qeydlərinə baxaqsız:

```
>>> from Bio import Entrez
Entrez.email = "ANOther@example.com" # Həmişə NCBI-ya kim olduğunuzu bildirin >>> stream =
Entrez.esearch(db="pubmed", term="biopython") >>> rekord = Entrez.read(axin) >>>
rekord["IdList"] [786,196,81930,16403221,
'16377612', '14871861',
'14630660', '12230038']
```

Biz indi bu Medline qeydlə rini yüklə mə k üçün Bio.Entrez.efetch istifadə edirik:

```
>>> idlist = qeyd["IdList"] >>> axin =
Entrez.efetch(db="pubmed", id=idlist, rettype="medline", retmode="text")
```

Burada Medline qeydlə rini düz mə tnlı Medline formatında ə ldə etmə k üçün rettype="medline", retmode="text" tə yin edirik. İndi biz bu qeydlə ri tə hlil etmə k üçün Bio.Medline istifadə edirik:

```
>>> Bio import Medline -dan >>> qeydlə r =
Medline.parse(stream) >>> qeydlə rde qeyd üçün :
...
... çap (rekord["AU"])
...
['Cock PJ', 'Antao T', 'Chang JT', 'Chapman BA', 'Cox CJ', 'Dalke A', ..., ['Munteanu CR', 'Gonzalez-Diaz H', 'Magalhaes AL'] 'de Hoon MJ']
...
['Casbon JA', 'Crooks GE', 'Saqi MA']
['Pritchard L', 'White JA', 'Birch PR', 'Toth IK'] ['de Hoon MJ', 'Imoto S', 'Nolan J',
'Miyano S']
['Hamelryck T', 'Manderick B']
['Mangalam H']
```

Müqayisə üçün burada XML formatından istifadə edə n bir nümunə göstəririk:

```
>>> axin = Entrez.efetch(db="pubmed", id=idlist, rettype="medline", retmode="xml") >>> qeydlə r = Entrez.read(stream) >>> qeydlə rde qeyd üçün ["PubmedArticle"]:
```

```
...
... print(record["MedlineCitation"]["Mə qalə "]["Mə qalə nin Başlığı"])
...
```

Biopython: hesablama molekulyar biologiyası üçün sə rbə st mövcud Python alə tlə ri və bioinformatika.

Tə rkibinə , ardıcılığına, 3D və topoloji göstə ricilə rə ə saslanan fermentlə rin/ferment olmayanların tə snifatı modelinin mürə kkə bliyi.

SCOP və ASTRAL üçün yüksə k sə viyyə li interfeys python-da hə yata keçirilir.

GenomeDiagram: geniş miqyaslı genomik mə lumatların vizuallaşdırılması üçün python paketi.

Açıq mə nbə li klaster programı.

PDB fayl analizatoru və struktur sinfi Python-da hə yata keçirilir.

Bio* alə t də stlə ri - qısa icmal.

Qeyd edə k ki, bu nümunə lə rin hə r ikisində sadə lik üçün biz sadə lövhəcə sinə ESearch və EFetch-i birlə şdirmişik. In bu və ziyyə tdə , NCBI sizdə n Bölmə 12.16-da göstə rildiyi kimi öz tarix funksiyasından istifadə etmə yinizi gözlə yir .

12.13.2 GEO qeydlə rinin tə hlili

GEO ([Gen İfadə si Omnibus](#)) yüksə k mə hsuldarlıqlı gen ifadə si və hibridlə şmə massivi mə lumatlarının mə lumat anbarıdır. Bio.Geo modulu GEO formatlı mə lumatları tə hlil etmə k üçün istifadə edilə bilə r.

Aşağıdakı kod fragmnti GSE16.txt nümunə GEO faylini qeydə necə tə hlil etmə k və yazının çapını göstərir:

```
>>> Bio import Geo- dan
>>> axın = açıq("GSE16.txt")
>>> qeydlə r = Geo.parse(axın)
>>> qeydlə r üçün :
...
    çap (qeyd)
...
```

Siz "gds" verilə nlə r bazasında (GEO verilə nlə r bazası) ESearch ilə axtarış edə bilə rsiniz:

```
>>> Bio import Entrez- də n
>>> Entrez.email = "ANOther@example.com" # Həmişə NCBI-yə kim olduğunuzu bildirin
>>> axın = Entrez.esearch(db="gds", termin="GSE16")
>>> rekord = Entrez.read (axın)
>>> stream.close()
>>> qeyd["Saymaq"]
'27'

>>> qeyd["IdList"]
['200000016', '100000028', ...]
```

Entrez veb saytından, UID "200000016" GDS16, digər "100000028" isə əlaqəlidir platforma, GPL28. Təəssüf ki, yazı zamanı NCBI GEO-nun yüklenmə sini dəstəkləmir faylları Entrez istifadə edərək (XML kimi deyil, nədə Simple Omnibus Format in Text (SOFT) formatında deyil). Bununla belə, <ftp://ftp.ncbi.nih.gov/pub/geo/> əziz zinə. Bu halda siz <ftp://ftp.ncbi.nih.gov/pub/geo/DATA/> istəyə bilərsiniz. SOFT/by_series/GSE16/GSE16_family.soft.gz (sixılmış fayl, Python modulunun gzip-ə baxın).

12.13.3 UniGene qeydlərinin təhlili

UniGene transkriptomun NCBI verilənlər bazasıdır və hər bir UniGene qeydi transkript dəstini göstərir. Müəyyən bir organizmdə müəyyən bir gen ilə əlaqəli olanlar. Tipik bir UniGene qeydi belə görünür:

ID	Hs.2
TITLE	N-asetiltransferaza 2 (arilamin N-asetiltransferaza)
GEN	NAT2
SİTOBAND	8p22
GENE_ID	10
LOCUSLINK	10
HOMOL	Bələdi
EXPRESS	sümük birləşdirici toxuma bağırsaq qaraciye r qaraciye r şisi normal yumşaq toxuma/əzələ toxuması
RESTR_EXPR	böyüklər
XROMOSOM	8
STS	ACC=PMC310725P3 UNISTS=272646
STS	ACC=WIAF-2120 UNISTS=44576
STS	ACC=G59899 UNISTS=137181
...	
STS	ACC=GDB:187676 UNISTS=155563
PROTSİM	ORG=10090; PROTGI=6754794; PROTID=NP_035004.1; PCT=76,55; ALN=288
PROTSİM	ORG=9796; PROTGI=149742490; PROTID=XP_001487907.1; PCT=79,66; ALN=288
PROTSİM	ORG=9986; PROTGI=126722851; PROTID=NP_001075655.1; PCT=76,90; ALN=288
...	
PROTSİM	ORG=9598; PROTGI=114619004; PROTID=XP_519631.2; PCT=98,28; ALN=288

```

SCOUNT      38
ARDILIQ     ACC=BC067218.1; NID=g45501306; PID=g45501307; SEQTYPE=mRNA ACC=NM_000015.2; NID=g116295259;
ARDILIQ     PID=g116295260; SEQTYPE=mRNA ACC=D90042.1; NID=g219415; PID=g219416; SEQTYPE=mRNA ACC=D90040.1; NID=g219411;
ARDILIQ     PID=g219412; SEQTYPE=mRNA ACC=BC015878.1; NID=g16198419; PID=g16198420; SEQTYPE=mRNA
ARDILIQ     ACC=CR407631.1; NID=g47115198; PID=g47115199; SEQTYPE=mRNA ACC=BG569293.1; NID=g13576946;
ARDILIQ     KLON=ŞƏ KİL:4722596; END=5'; LID=6989; SEQTYPE=EST; TRACE=44
ARDILIQ
ARDILIQ
...
ARDILIQ //   ACC=AU099534.1; NID=g13550663; KLON=HSI08034; END=5'; LID=8800; SEQTYPE=EST

```

Bu xüsusi qeyd N-asetiltransferazanı kodlayan insan NAT2 genində n yaranan transkriptlə r də stini (SEQUENCE sə tirlə rində göstə rilir) göstə rir. PROTSIM xə tlə ri NAT2 ilə ə hə miyyə tli oxşarlığı olan zülalları göstə rir, STS xə tlə ri isə genomda müvafiq ardıcılıqla işarə lə nmiş saytları göstə rir.

UniGene fayllarını tə hlil etmə k üçün Bio.UniGene modulundan istifadə edin:

```

>>> Bio import UniGene -də n >>> giriş
= open("myunigenefile.data") >>> rekord =
UniGene.read(giriş)

```

UniGene.read tə rə fində n qaytarılan qeyd, sahə lə rə uyğun atributları olan Python obyektidir. UniGene rekordu. Mə sə lə n,

```

>>> qeyd.ID
"Hs.2"
>>> qeyd.başlıq
"N-asetiltransferaza 2 (arilamin N-asetiltransferaza)"

```

EXPRESS və RESTR_EXPR sə tirlə ri Python sə tirlə rinin siyahısı kimi saxlanılır:

```

[
    "sümük",
    "birlə şdirici toxuma",
    "bağırsaq",
    "qaraciyə r",
    "qaraciyə r şışı",
    "normal",
    "yumşaq toxuma/ə zə lə toxuması şışı",
    "böyüklə r",
]

```

İxtisaslaşdırılmış obyektlə r STS, PROTSIM və SEQUENCE sə tirlə ri üçün qaytarılır, burada göstə rilə n düymə lə ri saxlayır. hə r bir sə tir atribut kimi:

```

>>> rekord.sts[0].acc
'PMC310725P3'
>>> rekord.sts[0].units
'272646'

```

və oxşar şə kildə PROTSIM və SEQUENCE xə tlə ri üçün.

Birdə n çox UniGene qeydi olan faylı tə hlil etmə k üçün Bio.UniGene-də tə hlil funksiyasından istifadə edin:

```

>>> Bio import UniGene -də n >>> input
= open("unigenerecords.data")

```

```
>>> qeydlə r = UniGene.parse(giriş) >>> qeydlə rda qeyd
Üçün :
...
    çap (rekord.ID)
...
```

12.14 Proksidə n istifadə

Normalda proksidə n istifadə etmə kdə n narahat olmayacaqsınız, lakin bu, şə bə kə nizdə problemdirə , bununla necə mə şəkil olmaq olar. Daxili olaraq, Bio.Entrez NCBI serverlə rinə daxil olmaq üçün standart Python kitabxana urllibibində n istifadə edir. Bu, istə nilə n sadə proxy-ni avtomatik konfiqurasiya etmə k üçün http_proxy adlı mühit də yişə nini yoxlayacaq. Tə ə ssüf ki, bu modul autentifikasiya tə lə b edə n proksilə rin istifadə sini də stə klə mir.

Siz http_proxy mühit də yişə nini bir də fə tə yin etməyi seçə bilərsiniz (bunu necə edə cə yiniz programdan asılı olacaq ə mə liyyat sistemi). Alternativ olaraq, bunu skriptinizin başlanğıcında Python-da tə yin edə bilərsiniz, mə sə lə n:

`idxal os`

```
os.environ["http_proxy"] = "http://proxyhost.example.com:8080"
```

Urllib sə nə dərinə baxın ətraflı məlumat üçün.

12.15 Nümunə lər

12.15.1 PubMed və Medline

Ə gə r tibb sahə sində sinizsə və ya insan problemləri ilə maraqlanırsınızsa (və bir çox hallarda belə olmasanız belə !), PubMed (<https://www.ncbi.nlm.nih.gov/PubMed/>) bütün növ nemətlərinə la mənbə yidir. Beləliklə , digər şeylər kimi, biz də ondan məlumat əldə etmək və Python skriptlərində istifadə etmək istərdik.

Bu nümunə də biz PubMed-ə səhləblərlə əlaqəli bütün məqalələr üçün sorğu göndərəcəyik (bizim məlumatlarımıza üçün bölmə 2.3-a baxın. motivasiya). Əvvəlcə belə məqalələrin neçə olduğunu yoxlayırıq:

```
>>> from Bio import Entrez >>>
Entrez.email = "ANOther@example.com" # Həmişə NCBI-ya kim olduğunuzu bildirin >>> stream =
Entrez.egquery(term="orchid") >>> rekord = Entrez.read(stream)
>>> rekorddakı sətir üçün ["eGQueryResult"]:
    əgər sətir varsa [""] p" çap (sətir["Saymaq"])
...
...
...
463
```

İndi bu 463 məqalənin PubMed ID-lərini yükərəmək üçün Bio.Entrez.efetch funksiyasından istifadə edirik:

```
>>> from Bio import Entrez >>>
Entrez.email = "ANOther@example.com" # Həmişə NCBI-ya kim olduğunuzu bildirin >>> axın =
Entrez.esearch(db="pubmed", term="orchid", retmax=463) >>> rekord = Entrez.read(stream) >>>
stream.close() ist >>> L qeyd = [""]
```

Bu, səhləblərlə aid məqalələrin bütün PubMed ID-lərini ehtiva edən Python siyahısını qaytarır:

```
>>> çap (idlist) ['18680603',
'18665331', '18661158', '18627489', '18627452', '18612381', '18594007', '18594007', '1859', '1859', '18579475',
'18575811', '18575690',
...]
```

İndi onları e ldə etdikdə n sonra biz açıq şə kildə müvafiq Medline qeydlə rini e ldə etmə k və onlardan mə lumatları çıxarmaq istə yirik. Burada biz Medline qeydlə rini Medline düz fayl formatında endirə cə yik və onları tə hlil etmə k üçün Bio.Medline modulundan istifadə edə cə yik:

```
>>> Bio import Medline -dan >>> axın =
Entrez.efetch(db="pubmed", id=idlist, rettype="medline", retmode="text") >>> qeydlə r = Medline.parse(stream)
```

QEYD - Biz indicə ayrıca axtarış etdik və buradan e ldə etdik, NCBI bu və ziyyə tdə onların tarix də stə yində n istifadə etmə yi sizə daha çox üstünlük verir. 12.16-ci bölmə yə baxın.

Nə zə rə alın ki, qeydlə r iteratordur, ona görə də qeydlə ri yalnız bir də fə tə karrlaya bilə rsiniz. İstə sə n qeydlə ri saxlamaq üçün onları siyahıya çevirə bilə rsiniz:

```
>>> qeydlə r = siyahı (qeydlə r)
```

İndi hə r bir qeyd haqqında bə zi mə lumatları çap etmə k üçün qeydlə ri tə karrlayaq:

```
>>> qeydlə r üçün :
...
    print("title:", record.get("TI", "?")) print("müə lliflə r:",
...
    record.get("AU", "?")) print("mə nbə :", record.get("SO", "?"))
...
print("")
```

Bunun üçün çıkış belə görünür:

başlıq: Erkə n hörümçə k sə hlə bində cinsi feromon tə qlidi (ophrys sphegodes): cinsi aldatma yolu ilə tozlanma üçün e sas mexanizm kimi karbohidrogen nümunə lə ri [In Process Citation] müə lliflə r: ['Schiestl FP', 'Ayasse M', 'Paulus HF', 'Lofstedans Cson', BS', BS

'Ibarra F', 'Francke W'] mə nbə :J
Comp Physiol [A] 2000 İyun;186(6):567-74

Standart Python siyahısı kimi qaytarılan müə lliflə rin siyahısı xüsusiilə maraqlıdır. Bu, standart Python alə tlə rində n istifadə edə rə k manipulyasiya və axtarışı asanlaşdırır. Mə sə lə n, aşağıdakı kimi kodu olan müə yyə n bir müə llifi axtaran bütün yazılar toplusunu dolaşa bilə rik:

```
>>> search_author = "T gözlə yir"
>>> qeydlə r üçün :
...
    qeyddə " AU " deyilsə :
...
        a gə r
...
    axtarış_müə llifi qeyddə ["AU"] davam edirsə :
...
        print("Müə llif %s tapıldı: %s" % (axtar_yazar, qeyd["Yə ni"]))
```

Ümid edirik ki, bu bölmə sizə Entrez və Medline interfeyslə rinin gücү və çevikliyi haqqında fikir verdi və birlikdə necə istifadə edilə bilə r.

12.15.2 Entrez Nucleotide qeydlərinin axtarışı, yüklənməsi və təhlili

Burada uzaq Entrez sorğusunun yerinə yetirilməsinin sadə bir nümunəini göstərəcəyik. Təhlil nümunəsi rəsminin 2.3-cü hissəsində biz NCBI-nin Entrez vəb saytından istifadə edərək, Cypripedioideae, dostlarımız olan qadın başlıqları səhəblərin haqqında məlumat üçün NCBI nukleotid verilənlərin bazasında axtarış etmək haqqında danışdıq. İndi biz bu prosesi Python skripti ilə necə avtomatlaşdıracağımıza baxacaqıq. Bu nümunədə biz yalnız Entrez modulu ilə bütün işləri görərək necə qoşulması, nəticələri rəsmini göstərəcəyik.

Birincisi, biz EGQuery-dən istifadə edərək, onları yükləməzdən sonra və ləğdən sonra edəcəyimiz nəticələrin sayı öyrənirik. EGQuery bizə verilənlərin bazalarının hər birində necə axtarış nəticəsinin tapıldığı söyləyəcək, lakin bu nümunə üçün bizi yalnız nukleotidlər maraqlandırır:

```
>>> from Bio import Entrez
>>> Entrez.email
= "ANOther@example.com" # Həmişə NCBI-yə kim olduğunuzu bildirin >>> axın = Entrez.egquery(term="Cypripedioideae")
>>> rekord = Entrez.read(axın) >>> rekorddakı sıra üçün ["eGQueryResult"] = :
uc:b çap (sə tir["Saymaq"])

...
...
...
4457
```

Bələdiyliklə, biz 4457 Entrez Nucleotide qeydini tapmağı gözləyirik (bu, 2008-ci ildə ki 814 qeyddən artdı; yəqin ki, gələcəkdə də artmağa davam edəcək). Əgər gülünc dərəcə yüksək sayıda hit tapsanız, onların hamısını həqiqətən yükləmək istəyirsinizsə, yenidən nəzərdən keçirmək istəyə bilərsiniz, bu bizim növbəti addımımızdır. 2008-ci ildə əldə edilən qeydlərin maksimum sayını məhdudlaşdırmaq üçün retmax argumentində istifadə edək:

```
>>> from Bio import Entrez
>>> Entrez.email
= "ANOther@example.com" # Həmişə NCBI-yə kim olduğunuzu bildirin >>> stream = Entrez.esearch(
...     db="nukleotid", termin="Cypripedioideae", retmax=814, idtype="acc"
... ) >>>
rekord = Entrez.read(stream) >>> stream.close()
```

Burada qeyd axtarış nəticələrini saxlamaq üçün Python lügətidir. Sadəcə məlumat üçün bu lügətdən nəticəyi saxlamağına nəzər salaq:

```
>>> çap (record.keys())
['Count', 'RetMax', 'IdList', 'TranslationSet', 'RetStart', 'QueryTranslation']
```

Əvvəlcə neçə nəticənin tapıldığı yoxlayaq:

```
>>> çap (record["Count"])
4457
```

Bunun 814, əldə etmək istədiyimiz qeydlərin maksimum sayını gözləyirdiniz. Bununla belə, Count həmin axtarış üçün mövcud olan qeydlərin ümumi sayını göstərir, nəqədə rəsmin əldə edildiyini deyil. Alınan qeydlər qeyd["IdList"-də saxlanılır, burada sorğuladığımız ümumi sayı olmalıdır:

```
>>> len(record["IdList"])
814
```

İlk beş nəticə yənəzər salaq:

```
>>> qeyd["IdList"][:5]
['KX265015.1', 'KX265014.1', 'KX265013.1', 'KX265012.1', 'KX265011.1']
```

Bu qeydlə rı efetech istifadə edə rə k yükə yə bilə rik. NCBI-nin serverlə rində ki yükü azaltmaq üçün bu qeydlə rı bir-bir yükə yə bilsə niz də , aşağıda göstərilə n bir də stə qeydi eyni vaxtda götürmə k daha yaxşıdır.
Bununla belə , bu və ziyyə təsiz Bölmə 12.16-da daha sonra təsvir edilə n tarixçə funksiyasından istifadə etmə lisiniz .

```
>>> idlist = " ".join(record["IdList"][:5]) >>> print(idlist)
```

```
KX265015.1, KX265014.1, KX265013.1, KX265012.1, KX265011.1] >>> axın =
Entrez.efetch(db="nukleotid", id=idlist, retmode="xml") >>> reads = () kord >>> read >> ()lent
```

5

Bu qeydlə rin hər biri bir GenBank qeydində uyğundur.

```
>>> çap (records[0].keys())
['GBSeq_moltype', 'GBSeq_source', 'GBSeq_sequence', 'GBSeq_primary-
accession', 'GBSeq_definition', 'GBSeq_accession-version', 'GBSeq_topology', 'GBSeq_length', '-
GBSeq_length', '-GBSeq_length', 'GB-Seq_te'cet 'GBSeq_other-seqids',
'GBSeq_division', 'GBSeq_taxonomy', 'GBSeq_references', 'GBSeq_update-date',
'GBSeq_organism', 'GBSeq_locus', 'GBSeq_strandedness']
```

>>> çap (qeydlə r[0]["GBSeq_primary-accession"])

DQ110336

>>> print(records[0]["GBSeq_other-seqids"]) ['gb|

DQ110336.1|', 'gi|187237168']

>>> çap (qeydlə r[0]["GBSeq_tərif"])

Cypripedium calceolus vaucher Davis 03-03 A maturaza (matR) geni, qismən CD; mitochondrial

>>> çap (qeydlə r[0]["GBSeq_orqanizm"])

Cypripedium calceolus

Siz bundan axtarışları cəld qurmaq üçün istifadə edə bilərsiniz – lakin ağır istifadə üçün Bölmə 12.16-a baxın.

12.15.3 GenBank qeydlərinin axtarışı, yükə nməsi və təhlili

GenBank qeyd formatı ardıcılıqlar, ardıcılıqlı xüsusiyyətləri və digərə laqəli ardıcılıqları məlumatları haqqında məlumat saxlamaq üçün çox məşhur üsuldur. Format <https://www.ncbi.nlm.nih.gov/> ünvanında NCBI verilənlər bazalarından məlumat əldə etmək üçün yaxşı bir yoldur.

Bu nümunədə biz NCBI verilənlər bazalarını necə sorğulamağı, sorğudan qeydləri əldə etməyi və sonra Bio.SeqIO-dan istifadə edərək onları təhlil etməyi göstərəcəyik - Bölmə 5.3.1-də toxunulan bir şey. Sadəlik üçün bu nümunə WebEnv tarixçəsi funksiyasından istifadə etmir - bunun üçün Bölmə 12.16-a baxın.

Birincisi, biz sorğu etmək və əldə ediləcək qeydlərin id-lərini öyrənmək istəyirik. Burada sevimli orqanizmlərimizdən biri olan Opuntia (tikanlı armud kaktusları) üçün sürətli axtarış edəcəyik. Biz sürətli axtarış apara və bütün müvafiq qeydlər üçün GI-ləri (GenBank identifikatorları) geri ala bilərik. Əvvəlcə orada neçə qeyd olduğunu yoxlayırıq bunlardır:

```
>>> from Bio import Entrez >>>
```

```
Entrez.email = "ANOther@example.com" # Həmişə NCBI-ya kim olduğunuzu bildirin >>> stream =
Entrez.esearch(term="Opuntia AND rpl16")
```

```
>>> rekord = Entrez.read(stream) >>> rekorddaki
sıra üçün ["eGQueryResult"]: if row["DbName"] == "nuccore":
...     print(row["Count"])
...
...
9
```

İndi GenBank identifikatorlarının siyahısını yükə yirik:

```
>>> axın = Entrez.esearch(db="nuccore", term="Opuntia AND rpl16") >>> rekord = Entrez.read(axın) >>> gi_list
= rekord["IdList"] >>> gi_list ['57240072', '57240071',
'57240071', '29', '29', '2'7' '6273290', '6273289',
'6273286',
'6273285', '6273284']
```

İndi biz GenBank qeydlə rini endirmə k üçün bu GI-lə rdə n istifadə edirik - nə zə rə alın ki, Biopython-un köhnə versiyaları ilə siz Entrez-ə vergüllə ayrılmış GI nömrə lə rinin siyahısını tə qdim etmə li idiniz, Biopython 1.59-dan etibarə n siyahı keçə bilə rsiniz və bu sizin üçün çevrilir:

```
>>> gi_str = ",".join(gi_list) >>> axın =
Entrez.efetch(db="nuccore", id=gi_str, rettype="gb", retmode="text")
```

Xam GenBank fayllarına baxmaq istə yirsinizsə , bu axından oxuya və nə ticə ni çap edə bilə rsiniz:

```
>>> mə tn = axın.read() >>> çap (mə tn)
```

LOCUS	AY851612	892 bp	DNT	xə tti PLN 10-APR-2007
TƏ YİF	Opuntia subulata rpl16 geni, intron; xloroplast.			
QOŞULMA	AY851612			
VERSİYA	AY851612.1 GI:57240072			
AÇAR SÖZLƏ R	.			
MƏ NBƏ	xloroplast Austrocylindropuntia subulata ORGANISM			
	Austrocylindropuntia subulata Eukaryota; viridiplantae;			
	streptofitlə r; embriyofitlə r; traxeofitlə r; Spermatofitlə r; Magnoliophyta; eudikotiledonlar; ə sas evdikotiledonlar; karyofilallar; kaktuslar; Opuntioideae; Austrocylindropuntia.			

ARAYIŞ 1 (ə sas 1-də n 892-yə qə də r)

Müə lliflə r Butterworth, CA və Wallace, RS

...

Bu və ziyyə tdə , biz yalnız xam qeydlə ri alırıq. Qeydlə ri daha Python-a uyğun formada ə ldə etmə k üçün Bio.SeqIO-dan istifadə edə rə k GenBank mə lumatlарını SeqRecord obyektlə rinə , o cümlə də n SeqFeature obyektlə rinə tə hlil edə bilə rik (bax. Fə sil 5):

```
>>> Bio import SeqIO -dan >>> axın =
Entrez.efetch(db="nuccore", id=gi_str, rettype="gb", retmode="text") >>> qeydlə r = SeqIO.parse(axın, "gb")
```

İndi qeydlə ri nə zə rdə n keçirə və bizi maraqlandıran mə lumatlara baxa bilə rik:

```
>>> qeydlə r üçün :
...     print(f"{record.name}, uzunluq {len(record)}, {len(record.features)} xüsusiyyə tlə ri ilə ")
```

AY851612, uzunluq 892, 3 xüsusiyyə tə
AY851611, uzunluq 881, 3 xüsusiyyə tə
AF191661, uzunluq 895, 3 xüsusiyyə tə
AF191665, uzunluq 902, 3 xüsusiyyə tə
AF191664, uzunluq 899, 3 xüsusiyyə tə
AF191663, uzunluq 899, 3 xüsusiyyə tə
AF191660, uzunluq 893, 3 funksiya ilə
AF191659, uzunluq 894, 3 funksiya ilə
AF191658, uzunluq 896, 3 funksiya ilə

Bu avtomatlaşdırılmış sorğu axtarışı funksiyasından istifadə işləri əlli yerinə yetirmək üçün böyük bir üstünlükdür. Modul NCBI-nin sənəd də maksimum üç sorğu qaydasına tabe olmasına baxmayaq, NCBI-nin pik saatlardan qaçmaq kimi başqa tövsiyələri də var. Bölmə **12.1-a baxın**. Xüsusiylə, sadəlik üçün bu nümunə də WebEnv tarixçəsi funksiyasından istifadə olunmadığını nəzərə alın. Siz bundan hər hansı qeyri-ciddi axtarış və yükləmə işi üçün istifadə etməlisiniz, Bölmə **12.16-a baxın**.

Nəhayət, əgər NCBI-dən faylları yüklemək və dərhədilə təhlil etməkə və zinətə həllinizi təkrarlaması planlaşdırırsınızsa (bu nümunə də göstərildiyi kimi), siz sadəcə olaraq qeydləri bir dəfə endirib sabit diskinizdə saxlamalı və sonra yerli faylı təhlil etməlisiniz.

12.15.4 Orqanizmin nəslinin tapılması

Bitki nümunəsi ilə qalaraq, indi Cypripedioideae səhəb ailəsinin nəslini tapaq. Əvvəlcə Taksonomiya verilənlər bazasında Cypripedioideae üçün axtarış aparırıq ki, bu da tam olaraq bir NCBI taksonomiya identifikasiatorunu verir:

```
>>> from Bio import Entrez >>> Entrez.email = "ANOther@example.com" # Həmişə NCBI-yə kim olduğunuzu bildirin >>> stream = Entrez.esearch(db="Taxonomy", term="Cypripedioideae") >>> rekord = Entrez.read(stream) >>> rekord["IdList"] > 15>>3 [] 0 qeyd["IdList"][] 158330'
```

İndi biz bu girişin Taksonomiya verilənlər bazasına yükləmək üçün efetch-dən istifadə edirik və sonra onu təhlil edirik:

```
>>> axın = Entrez.efetch(db="Taksonomiya", id="158330", retmode="xml") >>> qeydlər = Entrez.read(axın)
```

Yenə də bu qeyd çoxlu məlumat saxlayır:

```
>>> qeydlər[0].keys()
['Lineage', 'Division', 'ParentTaxId', 'PubDate', 'LineageEx',
 'Yaradılma tarixi', 'Vergi ID'si', 'Rank', 'GeneticCode', 'ScientificName',
 'MitoGeneticCode', 'Update Date']
```

Nəsəbi birbaşa bu qeyddən nəldə edə bilərik:

```
>>> qeydlər[0]["Soy"] 'hüceyrə
orqanizmləri; eukariota; viriplantae; streptofitlər; streptofitina;
embriyofitlər; traxeofitlər; Euphylophyta; Spermatofitlər; Magnoliopsida;
Liliopsida; asparagales; Orchidaceae'
```

Qeyd məlumatı burada göstərilən məlumatdan daha çoxunu ehtiva edir - məsələn, aşağıya baxın "Lineage" və zinə "LineageEx" və siz nəsil girişlərinin NCBI takson identifikasiatorlarını da nəldə edəcəksiniz.

12.16 Tarixçə də n və WebEnv-də n istifadə

Tez-tez bir sıra ə laqə li sorğular etmə k istə yə cə ksiniz. Ə n adə tə n, axtarış aparmaq, bə lkə də axtarışı də qıqla şdirmə k və sonra ə traflı axtarış nə ticə lə rini ə ldə etmə k. Bunu Entreze bir sıra ayrıca zə nglə r etmə klə edə bilə rsiniz. Bununla belə , NCBI sizə öz tarix də stə yində n yararlanmağı üstün tutur - mə sə lə n, ESearch və EFetch-i birlə şdirin.

Tarix də stə yinin başqa bir tipik istifadə si EPost və EFetch-i birlə şdirmə k olardı. Yeni tarix seansına başlayan identifikatorların siyahısını yükə mə k üçün EPost-dan istifadə edirsiz. Daha sonra sessiyaya istinad edə rə k (identifikatorların ə və zinə) EFetch ilə qeydlə ri yükə yırsınız.

12.16.1 Tarixçə də n istifadə edə rə k ardıcılıqların axtarılması və yükə nmə si

Tutaq ki, biz bütün Opuntia rpl16 nukleotid ardıcılıqlarını axtarmaq və yükə mə k və onları FASTA faylında saxlamaq istə yirik. Bölmə 12.15.3-də göstə rildiyi kimi , biz sadə lövhəcə sinə Bio.Entrez.esearch() funksiyasını birlə şdirə rə k Qoşulma nömrə lə rinin siyahısını ə ldə edə və sonra hamısını yükə mə k üçün Bio.Entrez.efetch()-ə zə ng edə bilə rik.

Bununla belə , tə sdiq edilmiş yanaşma axtarışı tarixçə xüsusiyyə ti ilə aparmaqdır. Sonra, biz gə tirə bilə rsiniz NCBI-nin tə xmin edə və keşələ yə bilə cə yi axtarış nə ticə lə rinə istinad edə rə k nə ticə lə r.

Bunu etmə k üçün normal olaraq Bio.Entrez.esearch()-a zə ng edin, lakin ə lavə usehistory="y" arqumenti ilə ,

```
>>> Bio import Entrez >>> Entrez.email
= "history.user@example.com" # Həmişə NCBI-ya kim olduğunuzu bildirin >>> axın = Entrez.esearch( db="nukleotid",
term="Opuntia[orgn] və rpl16",
...
usehistory="y", idtype= "acc"
...
)
>>> search_results = Entrez.read(stream) >>> stream.close()
```

Əvvə liki kimi (Bölmə 12.15.2-yə baxın), XML çıxışı ilk retmax axtarış nə ticə lə rini ehtiva edir, retmax defolt olaraq 20:

```
>>> acc_list = search_results["IdList"] >>> count =
int(search_results["Count"]) >>> len(acc_list)
```

20

```
>>> saymaq
28
```

Siz hə məcənin iki ə lavə mə lumat ə ldə edə cə ksiniz, WebEnv sessiya kukisi və QueryKey:

```
>>> webenv = search_results["WebEnv"] >>> query_key =
search_results["QueryKey"]
```

Bu də yə rlə ri də yişə nlə rin sessiya kukisində və sorğu açarında saxladıqdan sonra biz GI nömrə lə rini identifikator kimi vermə k ə və zinə Bio.Entrez.efetch()-də parametr kimi istifadə edə bilə rik.

Kiçik axtarıclar üçün hə r şeyi bir anda endirmə k yaxşı olar, ancaq toplu şə kildə yükə mə k daha yaxşıdır. Qaytarmaq istə diyiniz axtarış nə ticə lə rinin hansı diapazonunu (sıfır ə səslə hesablamadan istifadə edə rə k giriş və qaytarılacaq nə ticə lə rin maksimum sayı) tə yin etmə k üçün yenida n başlama və retmax parametrlə rində n istifadə edirsiz. Qeyd edə k ki, Biopython NCBI ilə ə laqə qurarkə n HTTP 500 cavabı kimi keçici uğursuzluqla qarşılaşarsa, o, avtomatik olaraq bir neçə də fə yenida n cə hd edə cə k. Mə sə lə n,

```
# Bu güman edir ki, siz artıq yuxarıda göstə rildiyi kimi axtarış aparmışınız, # və count,
webenv, query_key də yişə nlə rini tə yin edin
```

```

batch_size = 3 çıkış =
open("orchid_rpl16.fasta", "w") diapazonda başlamaq üçün (0,
say, batch_size): end = min(count, start + batch_size) print(" %i
rekordu %i -yə endirilir " % (start + 1, end)) axın =
Entrez.efetch( d == "b", retmode = "mə tn",

```

```

yenidə n başladın=başla,
retmax=batch_size,
webenv=webenv,
query_key=query_key,
idtype="acc",

) data = stream.read()
stream.close()
output.write(data)
output.close()

```

İllüstrasiya məqsə dlə ri üçün bu nümunə FASTA qeydlə rini üç qrup halında endirmişdir. Ə gə r siz genomları və ya xromosomları endirmə sə niz, adə tə n daha böyük partiya ölçüsünü seçə rdiniz.

12.16.2 Tarixçə də n istifadə edə rə k tezislə rin axtarışı və yüklə nmə si

Budur başqa bir tarix nümunə si, Opuntia haqqında keçə n ildə də rc edilmiş sə nə dlə ri axtararaq və sonra onları MedLine formatında fayla endirmə k:

Bio import Entrez- də n

```

Entrez.email = "history.user@example.com" search_results =
Entrez.read()

Entrez.esearch( db="pubmed", termin="Opuntia[ORGN]", reldate=365, datetype="pdat", usehistory="y"
)

) count = int(axtarış_nə ticə lə ri["Sayım"]) çap (" %i
nə ticə tapıldı" % sayı)

batch_size = 10 çıkış =
open("recent_orchid_papers.txt", "w") diapazonda başlamaq üçün (0,
count, batch_size): end = min(count, start + batch_size) print(" %i
rekordunu % i -yə endirmə yə gedir " % (start + 1, end))
axın = Entrez.efet(
    db="pubmed",
    rettype="medline",
    retmode="text",
    restart=start,
    retmax=batch_size,
    webenv=search_results["WebEnv"],
    query_key=search_results["QueryKey"],

```

```

) data = stream.read()
stream.close()
output.write(data)
output.close()

```

Yazı zamanı bu, 28 uyğunluq verdi - lakin bu, tarixdə n asılı axtarış olduğundan, bu, ə lbə ttə ki, də yişə cə k. Yuxarıdakı Bölmə 12.13.1- də tə svir olunduğu kimi , siz daha sonra saxlanılan qeydlə ri tə hlil etmə k üçün Bio.Medline istifadə edə bilə rsiniz.

12.16.3 Sitatların axtarışı

12.7-ci Bölmə də qeyd etdi ki, ELink verilmiş mə qalə yə istinadları axtarmaq üçün istifadə edilə bilə r. Tə ə ssüf ki, bu, yalnız PubMed Central üçün indekslə şdirilmiş jurnalları ə hatə edir (bunu PubMed-də ki bütün jurnallar üçün etmə k NIH üçün daha çox iş deməkdir). Gə lin bunu Biopython PDB analiz kağızı, PubMed ID 14630660 üçün sınayaq:

```

>>> from Bio import Entrez >>>
Entrez.email = "ANOOther@example.com" # Həmişə NCBI-yə kim olduğunuzu bildirin >>> pmid = "14630660"
>>> nəticə = Entrez.read()

...     Entrez.elink(dbfrom="pubmed", db="pmc", LinkName="pubmed_pmc_refs", id=pmid)
...
>>> pmc_ids = [link["Id"] nəticə lə rdə ki keçid üçün [0]["LinkSetDb"][0]["Link"]] >>> pmc_ids ['2744707',
'2705363',
'2682512', ...,
'1190160']


```

Ə la - on bir mə qalə . Bə s niyə Biopython program qeydi tapılmadı (PubMed ID 19304878)? Bə li, də yişə n adlarından təxmin etdiyiniz kimi, ə slində PubMed ID-lə ri deyil, PubMed Central ID-lə ri var. Bizim ə rizə qeydimiz hə min siyahidakı üçüncü istinad sə nə didir, PMCID 2682512.

Bə s, ə gə r (mə nim kimi) PubMed ID-lə rinin siyahısını geri qaytarmaq istə sə niz nə olacaq? Onları tə rcümə etmə k üçün yenidə n ELink-ə zəng edə bilə rik. Bu, iki addımlı prosesə çevrilir, ona görə də indi siz onu yerinə yetirmə k üçün tarix funksiyasından istifadə etməyi gözlə mə lisiniz (Bölmə 12.16).

Ancaq ə və lə ELink-ə ikinci (ayrıca) zəng etmə k üçün daha sadə yanaşmadan istifadə edin:

```

>>> nəticə = Entrez.read()
...     Entrez.elink(dbfrom="pmc", db="pubmed", LinkName="pmc_pubmed", id=".join(pmc_ids)")
...
>>> pubmed_ids = [link["Id"] results2 [0]["LinkSetDb"][0]["Link"]] >>> pubmed_ids ['19698094', '19450287',
'19304878', ...,
'15985178']


```

Bu də fə siz də rhal Biopython program qeydini üçüncü hit kimi görə bilə rsiniz (PubMed ID 19304878).

İndi gə lin bunu tarixlə birlikdə tə krar edə k... İŞLƏ MƏ K.

Və nə hayə t, Entrez zənglə rinə öz e-poçt ünvanınızı daxil etməyi unutmayın.

Fəsil 13

Swiss-Prot və ExpPASy

13.1 Swiss-Prot fayllarının təhlili

Swiss-Prot (https://web.expasy.org/docs/swiss-prot_guideline.html) zülal ardıcılığının əlliə hazırlanmış verilənlər bazasıdır. Biopython, hələ də Swiss-Prot, TrEMBL və PIR-PSD-ni birləşdirən UniProt Biliq bazası üçün istifadə olunan "düz mətn" Swiss-Prot fayl formatını təhlil edə bilər.

Baxmayaraq ki, aşağıda biz daha yaşlı insanlar tərəfindən oxuna bilən düz mətn formatına diqqət yetirsək də, Bio.SeqIO oxuya bilər həm bu, həm də annotasiya edilmiş protein ardıcılığı üçün daha yeni UniProt XML fayl formatı.

13.1.1 Swiss-Prot qeydlərinin təhlili

Bölmə 5.3.2-də biz SeqRecord obyekti kimi Swiss-Prot qeydinin ardıcılığının necə çıxarılaçığını təsvir etdik. Alternativ olaraq, siz Swiss-Prot qeydini Bio.SwissProt.Record obyektiində saxlaya bilərsiniz ki, bu da əslində Swiss-Prot qeydində olan tam məlumatı saxlayır. Bu bölmədə biz Bio.SwissProt.Record obyektlərinin Swiss-Prot faylından necə çıxarılaçığını təsvir edirik.

Swiss-Prot rekordunu təhlil etmək üçün əvvəlcə Swiss-Prot rekordunu əldə edirik. Bunun bir nəcəf yolu var. Beləliklə, Swiss-Prot rekordunun harada və necə saxlanmasından asılı olaraq:

- Swiss-Prot faylini yerli olaraq açın:

```
>>> tutacaq = açıq ("SwissProt/F2CXE6.txt")
```

- Gziplənmiş Swiss-Prot faylini açın:

```
>>> idxal gzip >>> sapi  
= gzip.open("myswissprotfile.dat.gz", "rt")
```

- Internet üzərindən Swiss-Prot faylini açın:

```
>>> urllib.request import urlopen >>> url = "https://  
raw.githubusercontent.com/biopython/biopython/master/Tests/SwissProt/F2CXE6.txt" >>> sapi = urlopen(url)
```

oxumaq üçün zəng etməzdən əvvəl Internetdə saxlanan faylı açmaq üçün.

- ExPASy verilənlər bazasından internet üzərindən Swiss-Prot faylini açın (bax. bölmə 13.5.1):

```
>>> Bio idxaldan ExpPASy >>> sapi =  
ExpPASy.get_sprot_raw ( "F2CXE6" )
```

Ə sas mə qam ondan ibarə tdir ki, tə hlilçi üçün, Swiss-Prot formatında mə lumatlara işaret etdiyi müddə tə sapın necə yaradıldığı önə mli deyil. Də stə k ikili rejimdə açılırsa, analizator mə lumatı avtomatik olaraq ASCII (Swiss-Prot tə rə fində n istifadə edilə n kodlaşdırma) kimi deşifrə edə cə k.

Fayl formatı aqnostik SeqRecord obyektlə rini ə ldə etmə k üçün Bölmə 5.3.2-də tə svir olunduğu kimi Bio.SeqIO-dan istifadə edə bilə rik. Alternativ olaraq, ə sas fayl formatına daha yaxın olan Bio.SwissProt.get Bio.SwissProt.Record obyektlə rində n istifadə edə bilə rik.

Də stə kdə n bir İsveçrə -Prot qeydini oxumaq üçün biz read() funksiyasından istifadə edirik:

```
>>> Bio import SwissProt- dan >>> rekord  
= SwissProt.read (tutacaq)
```

Də stə yin tam olaraq bir Swiss-Prot qeydinə işaret etdiyi halda bu funksiyadan istifadə edilməlidir. Heç bir Swiss-Prot qeydi tapılmadıqda və hə mçinin birdə n çox qeyd aşkar edildikdə ValueError yaradır.

İndi bu qeyd haqqında bə zi mə lumatları çap edə bilə rik:

```
>>> çap (rekord.tə svir)
```

AltAd: Tam=Plazma membranının daxili züllə {ECO:0000313|EMBL:BAN04711.1}; AltAd: Tam= Record.references -də istinad üçün proqnozlaşdırılan >>> :

```
...     print("müə lliflə r:", ref.authors)  
...     print("başlıq:", ref.title)  
...
```

müə lliflə r: Matsumoto T., Tanaka T., Sakai H., Amano N., Kanamori H., Kurita K., Kikuta A., Kamiya K., Yam başlığı: 12 klon kitabxanasından ə ldə edilə n 24,783 arpa tam uzunluqlu cDNA-nın hə rtə rə fli ardıcılıq tə hlili, müə lliflə r: U, Katsuhara, U., Katsui, S. M. adı: Arpada yeni plazma membranının daxili züllənin2 funksional xarakteristikası. müə lliflə r: Shibasaki M., Katsuhara M., Sasano S. başlıq:

```
>>> çap (record.organism_classification)
```

['Eukaryota', 'Viriplantae', 'Streptophyta', 'Embryophyta', 'Tracheophyta', 'Spermatophyta', 'Magnoli'

Birdə n çox Swiss-Prot qeydini ehtiva edə n faylı tə hlil etmə k üçün biz bunun ə və zinə parse funksiyasından istifadə edirik. Bu funksiya fayldakı qeydlə ri tə krarlamağa imkan verir.

Mə sə lə n, tam Swiss-Prot verilə nlə r bazasını tə hlil edə k və bütün tə svirlə ri toplayaq. Siz bunu ExpASy FTP saytından yükə yə bilə rsiniz tə k gzip faylı uniprot_sprot.dat.gz (tə xminə n 300MB) kimi. Bu, uniprot_sprot.dat (1,5 GB-dan çox) olan tə k fayldan ibarə t sixilmiş fayldır.

Bu bölmənin ə vvə linda tə svir edildiyi kimi, siz .gz faylini açmaq və sixışdırmaq üçün Python gzip kitabxanasından istifadə edə bilə rsiniz, mə sə lə n:

```
>>> idxl gzip >>>  
sapi = gzip.open("uniprot_sprot.dat.gz", "rt")
```

Bununla belə , böyük bir faylin sixılmasını açmaq vaxt tə lə b edir və siz bu şə kildə oxumaq üçün faylı hə r də fə açdığınız zaman o, tez bir zamanda sixılmalı olmalıdır. Belə liklə , ə gə r siz diskdə yer saxlaya bilsə niz, uniprot_sprot.dat faylini içə riyə daxil etmə k üçün ə vvə lə faylı diskə açsanız, uzun müddə tə vaxta qə naə t etmiş olarsınız. Sonra faylı hə mişə ki kimi oxumaq üçün aça bilə rsiniz:

```
>>> tutacaq = açıq ("uniprot_sprot.dat")
```

2009-cu ilin iyun ayına olan mə lumatə görə , ExPASy-də n endirilmiş tam Swiss-Prot verilə nlə r bazası 468851 Swiss-Prot ehtiva edir. qeydlə r. Rekord tə svirlə rinin siyahısını tə rtib etmə yin qısa yollarından biri siyahının başa düşülmə sidir:

```
>>> Bio import SwissProt >>> sapi =  
open("uniprot_sprot.dat") >>> tə svirlə r =  
[ SwissProt.parse(handle) -də qeyd üçün qeyd.description ]
```

```
>>> len(tə svirlə r) 468851 >>>
```

```
tə svirlə r[:5]
['RecName: Tam=Protein MGF 100-1R;',
 'RecName: Tam=Protein MGF 100-1R;',
 'RecName: Tam=Protein MGF 100-1R;',
 'RecName: Tam=Protein MGF 100-1R;',
 'RecName: Tam=Protein MGF 100-2L;']
```

Və ya qeyd iteratoru üzə rində for loopundan istifadə edə rə k:

```
>>> Bio import SwissProt -dan >>> tə svirlə ri
= [] >>> sapi =
open("uniprot_sprot.dat") >>> SwissProt.parse -də qeyd
etmə k üçün (handle):
...      tə svirlə r.ə lavə (qeyd.tə svir)
...
>>> len(tə svirlə r) 468851
```

Bu, belə böyük bir giriş faylı olduğundan, hə r iki halda mə nim yeni masaüstü kompüterimdə tə xminə n on bir də qıqə çə kir (giriş kimi sixilmamış uniprot_sprot.dat faylından istifadə etmə klə).

Swiss-Prot qeydlə rində n istə diyiniz hə r cür mə lumatı çıxarmaq eyni də rə cə də asandır. Görmə k üçün Swiss-Prot rekordunun üzvlə ri, istifadə

```
>>> dir(record) ['__doc__', '__init__', '__module__', 'accessions', 'annotation_update', 'comments', 'created', 'cross_references', 'data_class', 'description', 'entryfe', 'gen_adl', 'ev sahibi_organizm', 'açar sözlə r', 'molekul_növü', 'orqanel', 'organizm', 'organizm_tə snifikasi', 'istinadlar', 'seqinfo', 'ardicilliq', 'ardicilliq_uzunluğu', 'ardicilliq_uzunluğu', 'daxililik']
```

13.1.2 Swiss-Prot açar sözünün və kateqoriya siyahısının tə hlili

Swiss-Prot hə mçinin Swiss-Prot-da istifadə olunan açar sözlə r və kateqoriyaları sadalayan keywlist.txt faylini paylayır. Faylda aşağıdakı formada qeydlə r var:

ID 2Fe-2S.
AC KW-0001
DE Ə n azı bir 2Fe-2S də mir-kükürd klasterini ehtiva edə n protein: 2 də mir
DE atomları zülaldan 2 qeyri-üzvi sulfid və 4 kükürd atomu DE sisteinlə rinə komplekslə şir.
SY Fe2S2; [2Fe-2S] çoxluğu; [Fe2S2] çoxluğu; Fe2/S2 (qeyri-üzvi) klaster; SY Di-mu-sulfido-diiron; 2 də mir, 2 kükürd klaster bağlayıcı.
GO GO: 0051537; 2 də mir, 2 kükürd klasteri bağlayıcı HI Ligand: Də mir; Də mir-kükürd; 2Fe-2S.
HI Ligand: Metal bağlayıcı; 2Fe-2S.
CA Liqası. //

ID 3D strukturu.
AC KW-0002

DE Protein və ya üçölçülü quruluşu olan zülalın bir hissə si
 DE eksperimental olaraq həll edilmişdir (məsələn, rentgen kristalloqrafiyası və ya
 DE NMR spektroskopiyası) və koordinatları PDB-də mövcud olan
 DE verilənlər bazası. Nəzəri modellər üçün dəstə istifadə edilə bilər.

H1 Texniki termin: 3D-struktur.

CA Texniki termini.

//

ID 3Fe-4S.

...

Bu fayldə qeydlər Bio.SwissProt.KeyWList modulunda təhlil funksiyası ilə təhlil edilə bilər. Hər bir giriş daha sonra Python lüğəti olan Bio.SwissProt.KeyWList.Record kimi saxlanılır.

```
>>> Bio.SwissProt -dan KeyWList import >>> sapi = open("keywlist.txt")
>>> qeydlər = KeyWList.parse(handle) >>> qeydlər.read()
qeyd etmək üçün:

...
    çap(rekord["ID"])
...
    çap(qeyd["DE"])
...
```

Bu çap edir

2Fe-2S.

Ən azı bir 2Fe-2S dəmir-kükürd klasteri olan zülal: 2 dəmir atomu zülaldan 2 qeyri-üzvi sulfid və sisteinlərin 4 kükürd atomuna kompleksləşir.

...

13.2 Prosite qeydlərinin təhlili

Prosite protein domenlərinin, zülal ailələrinin, funksional saytları, həmçinin onları tanımaq üçün nümunələri rəsmiprofiləri ehtiva edən verilənlər bazasıdır. Prosite Swiss-Prot ilə paralel olaraq hazırlanmışdır. Biopython-da Prosite rekordu Bio.ExPASy.Prosite.Record sınıfı ilə təmsil olunur, onun üzvləri Prosite rekordunun müxtəlif sahələrinə uyğundur.

Ümumiyyətlə, Prosite faylında birdən çox Prosite qeydi ola bilər. Məsələn, [ExPASy FTP saytından](#) tələfayl (prosite.dat) kimi endiriləbilən Prosite qeydlərinin tam dəsteti, 2073 qeyddən ibarətdir (20.24 versiyası 4 dekabr 2007-ci ildə buraxılmışdır). Belə bir faylı təhlil etmək üçün yenidən iteratordan istifadə edirik:

```
>>> Bio.ExPASy import Prosite >>> handle =
open("myprositefile.dat") >>> records = Prosite.parse
(handle)
```

İndi biz qeydləri bir-bir götürüb bəzi məlumatları çap edə bilərik. Məsələn, fayldan istifadə etməklə tam Prosite verilənlər bazası olan, biz tapa bilərsiniz

```
>>> Bio.ExPASy import Prosite >>> sapi =
open("prosite.dat") >>> qeydlər =
Prosite.parse(handle) >>> rekord = sonrakı(qeydlər)

>>> qeyd.qoşulma
'PS00001'
>>> qeyd.adi
```

```
'ASN,GLYCOSYLATION' >>>
```

```
record.pdoc 'PDOC00001'
```

```
>>> rekord = növbəti(qeydlər)
```

```
>>> qeyd.qoşulma
```

```
'PS00004'
```

```
>>> qeyd.adı
```

```
'CAMP_PHOSPHO_SITE' >>>
```

```
rekord.pdoc 'PDOC00004'
```

```
>>> rekord = növbəti(qeydlər)
```

```
>>> qeyd.qoşulma
```

```
'PS00005'
```

```
>>> qeyd.adı
```

```
'PKC_PHOSPHO_SITE' >>>
```

```
rekord.pdoc 'PDOC00005'
```

və s. Nə qədər Prosite qeydinin olması ilə maraqlanırsınızsa, istifadə edə bilərsiniz

```
>>> from Bio.ExPASy import Prosite >>> handle =
```

```
open("prosite.dat") >>> records = Prosite.parse(handle)
```

```
>>> n = 0
```

```
>>> qeydlər üçün :
```

```
... n += 1
```

```
...
```

```
>>> n
```

```
2073
```

Dəstəkdən tam olaraq bir Prosite oxumaq üçün oxu funksiyasından istifadə edə bilərsiniz:

```
>>> Bio.ExPASy import Prosite >>> sapi =
```

```
open("mysingleprositerecord.dat") >>> rekord = Prosite.read(handle)
```

Bu funksiya heç bir Prosite qeydi tapılmadıqda və həmçinin birdən çox Prosite qeydi aşkar edildikdə ValueError yaradır.

13.3 Prosite sənədləri qeydlərinin təhlili

Yuxarıdakı Prosite nümunəsində rekord.pdoc qoşulma nömrələri 'PDOC00001', 'PDOC00004', 'PDOC00005' və s. Prosite sənədlərinə istinad edir. Prosite sənədləri qeydləri ExPASy-dən fərdi fayllar kimi və bütün Prosite sənədlərinini ehtiva edən bir fayl (prosite.doc) kimi mövcuddur.

Prosite sənədlərinin qeydlərini təhlil etmək üçün Bio.ExPASy.Prodoc-da təhlilcən istifadə edirik. Məsələn, Prosite sənədləşmə qeydinin bütün qoşulma nömrələrinin siyahısını yaratmaq üçün istifadə edə bilərsiniz

```
>>> Bio.ExPASy import Prodoc >>> handle =
```

```
open("prosite.doc") >>> records = Prodoc.parse(handle)
```

```
>>> accessions = [ record.accession for records ]
```

Yenə də oxu() funksiyası tutacaqdən tam olaraq bir Prosite sənədləri qeydini oxumaq üçün təqdim olunur.

13.4 Enzim qeydlərinin təhlili

ExPASy-nin Enzyme verilənlər bazası ferment nomenklaturasına dair məlumat anbarıdır. Tipik bir ferment qeydi aşağıdakı kimi görünür:

ID 3.1.1.34

DE Lipoprotein lipazi.

AN Təmizləyici amil lipaz.

AN Diasilqliserol lipazi.

AN Digliserid lipaz.

CA Triaçılqliserol + H(2)O = diasilqliserin + karboksilat.

CC -!- Xilomikronlarda və çox aşağı sıxlıqlı lipoproteinlərdə (VLDL) triaçılqliserolları hidroliz edir.

CC

CC -!- Həmçinin diasilqliserini hidroliz edir.

PR PROSITE; PDOC00110; DR

P11151, LIPL_BOVIN ; P11153, LIPL_CAVPO ; P11602, LIPL_CHICK ; DR P55031, LIPL_FELCA ; P06858,

LIPL_HUMAN ; P11152, LIPL_MOUSE ; DR O46647, LIPL_MUSVI ; P49060, LIPL_PAPAN ; P49923,

LIPL_PIG ; DR Q06000, LIPL_RAT ; Q29524, LIPL_SHEEP ; //

Bu nümunədə birinci sətir lipoprotein lipazının EC (Ferment Komissiyası) sayını göstərir (ikinci sətir). Lipoprotein lipazanın alternativ adları "Təmizləyici amil lipaz", "diasilqliserol lipaz" və "digliserid lipaz"dır (3-dən 5-ə qədər). "CA" ilə başlayan xətti bu fermentin katalitik fəaliyyətinə göstərir. Şərh sətirləri "CC" ilə başlayır. "PR" sətri Prosite Documentation qeydlərinə istinadları, "DR" sətirləri isə Swiss-Prot qeydlərinə istinadları göstərir. Bu qeydlərdən heç bir Ferment qeydində mütləq mövcud deyil.

Biopython-da Ferment qeydi Bio.ExPASy.Enzyme.Record sınıfı ilə təmsil olunur. Bu qeyd Python lügətindən götürülmüşdür və Enzim fayllarında istifadə olunan iki hərfli kodlara uyğun düymələr malikdir.

Tərkibində bir enzim qeydi olan Enzim faylini oxumaq üçün Bio.ExPASy.Enzyme-də oxu funksiyasından istifadə edin:

```
>>> Bio.ExPASy->dən idxl fermenti >>> açıq
("lipoprotein.txt") ilə tutacaq olaraq : qeyd =
...
    Enzyme.read(tutacaq)
...
>>> qeyd["ID"]
'3.1.1.34'
>>> qeyd["DE"]
'Lipoprotein lipaz.' >>>
qeyd["AN"]
['Təmizləyici amil lipaz.', 'Diasilqliserol lipaz.', 'Digliserid lipaz.']
>>> qeyd["CA"]
'Triasilqliserol + H(2)O = diasilqliserol + karboksilat.' >>> rekord["PR"]
['PDOC00110']
>>> qeyd["CC"]
['Xilomikronlarda və çox aşağı sıxlıqlı lipoproteinlərdə (VLDL) triaçılqliserolları hidroliz edir.', 'Həmçinin diasilqliserini hidroliz edir.']
>>> qeyd["DR"]
[['P11151', 'LIPL_BOVIN'], ['P11153', 'LIPL_CAVPO'], ['P11602', 'LIPL_CHICK'], ['P55031', 'LIPL_FELCA'], ['P06858', 'LIPL_FELCA'], ['P06858', 'LIPL_H111' 'LIPL_MOUSE'], ['O46647', 'LIPL_MUSVI'], ['P49060', 'LIPL_PAPAN'], ['P49923', 'LIPL_PIG'], ['Q06000', 'LIPL_RAT'], ['2']5, ['']29
```

Oxuma funksiyası heç bir ferment qeydi tapılmadiqda və hə mçinin birdə n çox ferment qeydi aşkar edildikdə ValueError yaradır.

Ferment qeydlərinin tam də sti [ExPASy FTP saytından tək fayl \(enzyme.dat\)](#) kimi endirilə bilər, 4877 qeyddən ibarətdir (3 mart 2009-cu il buraxılışı). Çoxlu ferment qeydləri olan belə faylı təhlil etmək üçün təkrarlayıcı əldə etmək üçün Bio.ExPASy.Enzyme-də təhlil funksiyasından istifadə edin:

```
>>> Bio.ExPASy -dən idxal fermenti >>> sapi =
open("enzyme.dat") >>> qeydləri =
Enzyme.parse(tutacaq)
```

İndi qeydləri bir-bir təkrarlaya bilərik. Məsələn, Ferment qeydinin mövcud olduğu bütün EC nömrələrinin siyahısını tərtib edə bilərik:

```
>>> ecnumbers = [ qeydlərdə qeyd üçün qeyd["ID"] ]
```

13.5 ExpASy serverinə daxil olmaq

Swiss-Prot, Prosite və Prosite sənədlərinin qeydləri <https://www.expasy.org> ünvanında olan ExPASy veb serverindən endiriləbilərlər. ExPASy-dən dörd növ sorğu mövcuddur:

prodəc girişini əldə edin HTML formatında Prosite sənədləri qeydini yükləmək üçün
prosite girişini əldə edin HTML formatında Prosite rekordunu yükləmək üçün
Prosite raw əldə edin Prosite və ya Prosite sənədlərini xam formatda yükləmək üçün
get sprot raw Swiss-Prot rekordunu xam formatda yükləmək üçün

Bu veb serverə Python skriptindən daxil olmaq üçün biz Bio.ExPASy modulundan istifadə edirik.

13.5.1 İsveçrə -Prot rekordunun bərpası

Tutaq ki, biz orkide üçün xalkon sintazalarına baxırıq (səhəblər haqqında maraqlı şeylər axtarmaq üçün bəzi əsaslandırmalar üçün [2.3-cü bölmə](#) yə baxın). Xalkon sintaza bitkilərdə flavanoidlərin biosintezində iştirak edir və flavanoidlər pigment rəngləri və UV qoruyucuları kimi çoxlu sərin şeylər yaradır.

Swiss-Prot-da axtarış etsəniz, Chalcone Synthase üçün üç orkide zülal tapa bilərsiniz, id nömrələri O23729, O23730, O23731. İndi bunları tutan və bəzi maraqlı məlumatları təhlil edən bir skript yazaq.

Əvvəlcə Bio.ExPASy-nin `get_sprot_raw()` funksiyasından istifadə edərək qeydləri götürürük. Bu funksiya çox gözəldir, çünkü siz onu bir id ilə təxmin edə və xam mətn qeydinə dəstək ala bilərsiniz (hansısa HTML yoxdur!). İsveçrə -Prot rekordunu çıxarmaq üçün Bio.SwissProt.read və ya SeqRecord əldə etmək üçün Bio.SeqIO.read istifadə edə bilərik. Aşağıdakı kod indicə yazıqlarımı yerinə yetirir:

```
>>> Bio import ExpASy >>> Bio import
SwissProt -dan

>>> qosulmalar = ["O23729", "O23730", "O23731"] >>> qeydlər = []
```

>>> qosulmalarla qosulmaq üçün :

```
...     sapi = ExpASy.get_sprot_raw(qosulma) qeydi =
...     SwissProt.read(sapi) qeydləri.append(qeyd)
...
...
```

Ə gə r ExPASy.get_sprot_raw-a tə qdim etdiyiniz qoşulma nömrə si mövcud deyilsə , SwissProt.read(handle) bir ValueError qaldıracaq. Yanlış qoşulma nömrə lə rini aşkar etmə k üçün ValueException istisnalarını tuta bilə rsiniz:

```
>>> qoşulmalarда qoşulmaq üçün :
...
...     sapi = ExpPASy.get_sprot_raw (qoşulma) cə hd edin:
...
...
...     rekord = SwissProt.read (sapi)
...
...     ValueException istisna olmaqla :
...
...         print("XƏ BƏ RDARLIQ: %s qoşulma tapılmadı" % qoşulma)
...
...     qeydlə r.append(qeyd)
...
```

13.5.2 Axtarış Swiss-Prot

İndi qeyd edə bilə rsiniz ki, mə n qeydlə rin qoşulma nömrə lə rini ə vvə lcə də n bilirdim. Hə qıqə tə n, get_sprot_raw() ya giriş adına, ya da qoşulma nömrə sinə ehtiyac duyur. Ə linizdə olmadıqda, hazırda <https://www.uniprot.org/> istifadə edə bilə rsiniz. lakin bunu skriptdə n axtarmaq üçün bizim Python paketimiz yoxdur.

Bə lkə burası töhfə verə rdiniz?

13.5.3 Prosite və Prosite sə nə dlə rinin ə ldə edilmə si

Prosite və Prosite sə nə dlə ri ya HTML formatında, ya da xam formatda ə ldə edilə bilə r. Biopython ilə Prosite və Prosite sə nə dlə rinin qeydlə rini tə hlil etmə k üçün siz qeydlə ri xam formatda ə ldə etmə lisiniz. Diga r mə qədər üçün HTML formatında bu qeydlə rlə maraqlana bilə rsiniz.

Prosite və ya Prosite sə nə dlə rini xam formatda ə ldə etmə k üçün get_prosite_raw() istifadə edin. üçün mə sə lə n, Prosite rekordunu yüklə mə k və onu xam mə tn formatında çap etmə k üçün istifadə edin

```
>>> Bio import ExPASy >>> sapi =
ExpPASy.get_prosite_raw("PS00001") >>> mə tn = handle.read ()
>>> çap (mə tn)
```

Prosite qeydini ə ldə etmə k və onu Bio.Prosite.Record obyektinə tə hlil etmə k üçün istifadə edin

```
>>> Bio import ExPASy >>> from Bio
idxal Prosite >>> sapi =
ExpPASy.get_prosite_raw("PS00001") >>> rekord =
Prosite.read(handle )
```

Eyni funksiyadan Prosite sə nə dlə şmə qeydini ə ldə etmə k və onu Bio.ExPASy.Prodoc.Record obyektinə tə hlil etmə k üçün istifadə edilə bilə r:

```
>>> Bio- dan ExPASy >>> Bio.ExPASy -
də n idxal Prodoc >>> sapi =
ExpPASy.get_prosite_raw("PDOC00001") >>> rekord =
Prodoc.read(handle)
```

Qeyri-mövcud qoşulma nömrə lə ri üçün, ExPASy.get_prosite_raw boş sə tirə qolu qaytarır. Boş sə tirlə qarşılaşdıqda, Prosite.read və Prodoc.read ValueError-u qaldıracaq. Etibarsız qoşulma nömrə lə rini aşkar etmə k üçün bu istisnaları tuta bilə rsiniz.

Get_prosite_entry() və get_prodoc_entry() funksiyaları Prosite və Prosite yükə mə k üçün istifadə olunur. HTML formatında sə nə d qeydlə ri. Bir Prosite qeydini göstə rə n bir veb sə hifə yaratmaq üçün istifadə edə bilə rsiniz

```
>>> Bio import ExpPASy >>> sapi =
ExpASy.get_prosite_entry("PS00001") >>> html = handle.read() >>> ile
open("myprositerecord.html", "
w") ile out_handle: out_handle.write(html)
...
...
```

və eynilə Prosite sə nə dlə ri qeydi üçün:

```
>>> Bio import ExpPASy >>> sapi =
ExpASy.get_prodoc_entry("PDOC00001") >>> html = handle.read() >>>
ile open("myprodorecord.html",
" w") ile out_handle: out_handle.write(html)
...
...
```

Bu funksiyalar üçün etibarsız qoşulma nömrə si HTML formatında sə hv mesajı qaytarır.

13.6 Prosite mə lumat bazasının skan edilmə si

[ScanProsite](#) UniProt və ya PDB ardıcılıq identifikatorunu və ya ardıcılığın özünü tə min etmə klə Prosite verilə nlə r bazasına qarşı onlayn zülal ardıcılığını skan etmə yə imkan verir. ScanProsite haqqında ə traflı mə lumat üçün [ScanProsite](#) sə nə dlə rinə baxın hə mçinin [ScanProsite](#) proqram tə minatına daxil olmaq üçün sə nə dlə r.

Python-dan Prosite verilə nlə r bazasını skan etmə k üçün Biopython-un Bio.ExPASy.ScanProsite modulundan istifadə edə bilə rsiniz. Bu modul hə m ScanProsite-ə proqramlı şə kildə daxil olmağa, hə m də ScanProsite tə rə fində n qaytarılan nəticə lə ri tə hlil etmə yə kömə k edir. Aşağıdakı zülal ardıcılığında Prosite nümunə lə rini skan etmə k üçün:

```
MEHKEVLLLLFLKSGQGEPLDDYVNTQGASLFSVTKKQLGAGSIEECAAKCEEDEEFT
CRAFQYHSKEQQCVIMAENRKSSIIIRMRDVVLFEKKVYLSECKTNGKNYRGTMSTKN
```

aşağıdakı kodu istifadə edə bilə rsiniz:

```
>>> ardıcılığı = (
...     "MEHKEVLLLLFLKSGQGEPLDDYVNTQGASLFSVTKKQLGAGSIEECAAKCEEDEEFT"
...     "CRAFQYHSKEQQCVIMAENRKSSIIIRMRDVVLFEKKVYLSECKTNGKNYRGTMSTKN"
... )
>>> Bio.ExPASy -də n idxl ScanProsite >>> sapi =
ScanProsite.scan(seq=ardıcılıq)
```

handle.read() funksiyasını yerinə yetirmə klə siz axtarış nəticə lə rini xam XML formatında ə ldə edə bilə rsiniz. Bunun ə və zinə istifadə edə k Bio.ExPASy.ScanProsite.read xam XML-i Python obyekti tə hlil etmə k üçün:

```
>>> nəticə = ScanProsite.read(sapi) >>> növü(nəticə )
<class
'Bio.ExPASy.ScanProsite.Record'>
```

Bio.ExPASy.ScanProsite.Record obyekti siyahıdan ə ldə edilir və siyahıdakı hə r bir element bir ScanProsite hitini saxlayır. Bu obyekt, hə mçinin ScanProsite tə rə fində n qaytarılan axtarış ardıcılığının sayını, hə mçinin hitlə rin sayını saxlayır. Bu ScanProsite axtarışı altı hitlə nəticə lə ndi:

```
>>> nəticə .n_seq 1
>>> nəticə .n_match
```

```
1  
>>> len(nəticə)  
1  
>>> nəticə[0]  
{'sequence_ac': 'USERSEQ1', 'start': 16, 'stop': 98, 'imza_ac': 'PS50948', 'bal': '8.873', 'leve
```

Digər ScanProsite parametrləri açar söz argumentləri kimi ötürürlər bilər; [ScanProsite program təminatına daxil olmaq üçün sənədlərə](#) baxın əlavə məlumat üçün. Nümunə olaraq, aşağıda viyyəti xalları olan matçları daxil etmək üçün aşağıda bal=1 keçidi əlavə bir hit tapmağa imkan verir:

```
>>> sapi = ScanProsite.scan(ardicilliq=ardicilliq, aşağı xal=1) >>> nəticə =  
ScanProsite.read(handle) >>> nəticə.n_match
```

2

Fəsil 14

3D-ə getmək: PDB modulu

Bio.PDB bioloji makromoleküllerin kristal strukturları ilə işləmə yə yönəlmüş Biopython moduludur.

Digər şeylər arasında, Bio.PDB fayldakı atom məlumatlarına rahat şəkildə daxil olmaq üçün istifadə edilə bilən Struktur obyektini yaradan PDBParser sınıfının ehtiva edir. PDB başlığında olan məlumatların təhlili üçün məhdud da stək var. PDB fayl formatı artıq yeni məzmunu da stəkləmək üçün dəyişdirilmir və ya genişləndirilmir və PDBx/mmCIF 2014-cü ildə standart PDB arxiv formatına çevrildi. Bütün Ümumdünya Protein Məlumat Bankı (wwPDB) saytları PDB daxiletmələrinin məlumat məzmununu təsvir etmək üçün makromolekulyar Kristalloqrafiya Məlumat Faylı (mmCIF) məlumat lügətlərindən istifadə edir. mmCIF makromolekulyar struktur məlumatlarını təmsil etmək üçün çevik və genişləndirilə bilən açar-dəyişmək rütubətindən istifadə edir və bir PDB girişində təmsil oluna bilən atomların, qəliqların və ya zəncirlərin sayına heç bir məhdudiyyət qoymur (parçalanmış girişlər yoxdur!).

14.1 Kristal struktur fayllarının oxunması və yazılması

14.1.1 mmCIF faylinin oxunması

Əvvəlcə MMCIFParser obyekti yaradın:

```
>>> Bio.PDB.MMCIFParser -dən idxal MMCIFParser >>> parser =
MMCIFParser()
```

Sonra mmCIF faylından struktur obyekti yaratmaq üçün bu analizatordan istifadə edin:

```
>>> strukturu = parser.get_structure("1fat", "1fat.cif")
```

Bir mmCIF faylına daha aşağıda verilmiş girişə ildə etmək üçün mmCIF faylındaki bütün mmCIF təqərlərinin onlarından dəyişmək rüətinə uyğunlaşdırılan Python lügəti yaratmaq üçün MMCIF2Dict sınıfında nistifadə edə bilərsiniz. Çoxsayılı dəyişmə rin (bütün atomların yə koordinatlarını saxlayan _atom_site.Cartn_y təqində olduğu kimi) və ya tək dəyişmə rin (ilkin yerləşdirilmə tarixi kimi) olmasından asılı olmayaraq, təqərləri sıyahısına uyğunlaşdırılır. Lügət mmCIF faylından aşağıdakı kimi yaradılmışdır:

```
>>> Bio.PDB.MMCIF2Dict -dən idxal MMCIF2Dict >>> mmcif_dict =
MMCIF2Dict("1FAT.cif")
```

Nümunə : solvent tərkibini mmCIF faylından əldə edin:

```
>>> sc = mmcif_dict["_exptl_crystal.density_percent_sol"]
```

Misal: bütün atomların yə koordinatlarının sıyahısını əldə edin

```
>>> y_list = mmcif_dict["_atom_site.Cartn_y"]
```

14.1.2 MMTF formatında fayolların oxunması

Fayldan strukturu oxumaq üçün birbaşa MMTFParser-də n istifadə edə bilərsiniz:

```
>>> Bio.PDB.mmtf -də n idxal MMTFParser >>> struktur =  
MMTFParser.get_structure("PDB/4CUP.mmtf")
```

Və ya PDB ID-si ilə struktur a ldə etmə k üçün eyni sinifdə n istifadə edə bilərsiniz:

```
>>> struktur = MMTFParser.get_structure_from_url("4CUP")
```

Bu sizə PDB və ya mmCIF faylından oxunmuş kimi Struktur obyekti verir.

Siz hə mçinin Biopython-un daxili istifadə etdiyi xarici MMTF kitabxanasından istifadə edə rəkəsə sas mə lumatlara çıxış a ldə edə bilərsiniz:

```
>>> mmtf idxalından a ldə etmə k >>>  
decoded_data = gə tirmə k ("4CUP")
```

Mə sə lə n, yalnız X koordinatına daxil ola bilərsiniz.

```
>>> çap (decoded_data.x_coord_list)
```

14.1.3 PDB faylinin oxunması

Əvvə lca PDBParser obyekti yaradırıq:

```
>>> Bio.PDB.PDBParser -də n idxal PDBParser >>> parser =  
PDBParser(PERMISSIVE=1)
```

İCAZƏ VERİLƏN bayraq göstərir ki, PDB faylları ilə a laqə li bir sıra ümumi problemlərin (bax 14.7.1) nə zərə alınmayacaq (lakin bəzi atomların və /yaxud qalıqların a skik olacağını nə zərə alıñ). Əgər bayraq mövcud deyilsə, tə hlil a mə liyyati zamanı hər hansı problem aşkar edilərsə, PDBConstructionException yaradılacaq.

Struktur obyekti daha sonra PDBParser obyektiñə PDB faylini tə hlil etmə yə icazə vermə klə istehsal olunur (bu halda PDB faylı pdb1fat.ent adlanır, 1fat struktur üçün istifadəçi tərəfindən müəyyən edilmiş addır):

```
>>> structure_id = "1fat" >>> faylı adı =  
"pdb1fat.ent" >>> struktur =  
parser.get_structure (structure_id, faylı adı)
```

Siz get başlığı ilə PDBParser obyektindən PDB faylinin başlığını və treylerini (sə tirlərin sadə siyahıları) çıxara və treyler metodlarını a ldə edə bilərsiniz. Qeyd edək ki, bir çox PDB fayllarında natamam və ya sə hv mə lumatlar olan başlıqlar var. Sə hylərin çoxu ekvivalent mmCIF fayllarında düzə ldildi.

Bələliklə, a gər başlıq mə lumatları ilə maraqlanırsınızsa, PDB başlığını tə hlil etmək a və zinə, yuxarıda tə svir olunan MMCIF2Dict alətiindən istifadə edərək mmCIF fayllarından mə lumat çıxarmaq yaxşı bir fikirdir.

İndi bu aydınlaşdı, gəlin PDB başlığını tə hlil etmə yə qayıdaq. Struktur obyekti adlı atribut var başlıq qeydləri də yərlərinə uyğunlaşdırılan Python lüğətidir.

Misal:

```
>>> resolution = structure.header["resolution"] >>> açar sözlər =  
structure.header["açar sözlər"]
```

Mövcud açarlar ad, baş, yerlə şdirmə _tarixi, buraxılış_tarixi, struktur_metod, qətnamə, struktur_istinad (istinadlar siyahısına uyğunlaşdırılır), jurnal_istinad, müəllif, birləşmə (kristallaşmış birləşmə haqqında müxtəlif mə lumatlar olan lüğətə xəritə verir), has_missing_residues, missing_residues və astral domeni haqqında a lavə mə lumat tə qdim edir.

`has_missing_residues` ə n azı bir boş olmayan REMARK 465 başlıq xə tti tapılarsa, Doğru olan bool ilə xə ritə lə r. Bu halda, tə crübə də istifadə olunan molekulda ATOM koordinatlarının müə yyə n edilə bilmə yə n bə zi qalıqları olduğunu düşünmə lisiniz. `missing_residues` itkin qalıqlar haqqında mə lumat olan lügə tlə rin siyahısına xə ritə lə r verir. PDB başlığı PDB spesifikasiyasındaki şablona uyğun gə lmirsə , çatışmayan qalıqların siyahısı boş və ya natamam olacaq.

Lügə t Structure obyekti yaratmadan da yaradıla bilə r, yə ni. birbaşa PDB faylından:

```
>>> Bio.PDB -də n parse_pdb_header >>> açıq (fayl adı , "r") ilə
idxal edin :
...         header_dict = parse_pdb_header(sapi)
...
```

14.1.4 PQR faylinin oxunması

PQR faylini tə hlil etmə k üçün PDB fayllarında olduğu kimi eyni şə kildə davam edin:

is_pqr bayrağından istifadə edə rə k PDBParser obyekti yaradın:

```
>>> Bio.PDB.PDBParser -də n idxal PDBParser >>> pqr_parser =
PDBParser(PERMISSIVE=1, is_pqr=Doğru)
```

Doğru olaraq tə yin edilmiş is_pqr bayrağı tə hlil edilə cə k faylin PQR faylı olduğunu və tə hlilçinin hə r bir atom girişi üçün atom yükünü və radius sahə lə rini oxumalı olduğunu göstə rir. PQR faylları ilə eyni prosedurdan sonra Struktur obyekti hazırlanır və PQR faylı tə hlil edilir.

```
>>> structure_id = "1fat" >>> fayl adı =
"pdb1fat.ent" >>> struktur =
parser.get_structure (structure_id, fayl adı, is_pqr=Doğru)
```

14.1.5 PDB XML formatında faylların oxunması

Bu hə lə də stə klə nmir, lakin biz mütlə q gə lə cə kdə bunu də stə klə mə yi planlaşdırırıq (bu, çox iş deyil).
Ə gə r ehtiyacınız varsa, poçt siyahısı vasitə silə Biopython tə rtibatçıları ilə ə laqə saxlayın.

14.1.6 mmCIF fayllarının yazılması

MMCIFIO sinfi strukturları mmCIF fayl formatına yazmaq üçün istifadə edilə bilə r:

```
>>> io = MMCIFIO() >>>
io.set_structure(s) >>> io.save("out.cif")
```

Select sinfi aşağıdakı PDBIO-ya oxşar şə kildə istifadə edilə bilə r. MMCIF2Dict istifadə edə rə k oxunan mmCIF lügə tlə ri də yazılıa bilə r:

```
>>> io = MMCIFIO() >>>
io.set_dict(d) >>>
io.save("out.cif")
```

14.1.7 PDB fayllarının yazılması

Bunun üçün PDBIO sinifində n istifadə edin. Ə lbə ttə ki, strukturun xüsusi hissə lə rini yazmaq da asandır.

Misal: strukturun saxlanması

```
>>> io = PDBIO()
>>> io.set_structure(s)
>>> io.save("out.pdb")
```

Əgər strukturun bir hissəini yazmaq istəyirsinizsə, Select sınıfında n istifadə edin (həmçinin PDBIO-da). Select dörd üsula malikdir:

- qəbul_model(model)
- qəbul_zəncir(zəncir)
- qəbul_qalıq(qalıq)
- qəbul_atom(atom)

Varsayılan olaraq, hər bir metod 1 qaytarır (bu o deməkdir ki, model/zəncir/qalıq/atom çıkışa daxil edilir). Select subclassing və uyğun olduqda 0 qaytarmaqla siz modelləri, zəncirləri və s-ni çıxışdan xaric edə bilərsiniz. Çətin olabilər, amma çox güclü. Aşağıdakı kod yalnız qalıqlarını yazır:

```
>>> sinif GlySelect(Seç):
...     def accept_residue(self,
...         residue):
...         if residue.get_name() == "GLY":
...             qaytarmaq Doğrudur
...         başqa:
...             ...
...             False qaytarın
...
...>>> io = PDBIO()
...>>> io.set_structure(s)
...>>> io.save("gly_only.pdb", GlySelect())
```

Əgər bu sizin üçün çox müəkkəbdirsə, Zar modulu başlangıç və son qalıqlar arasındaki zəncirdəki bütün qalıqları yazan lazımlı çıxarış funksiyasını ehtiva edir.

14.1.8 PQR fayllarının yazılması

PDBIO sınıfında n PDB faylı üçün istifadə etdiyiniz kimi istifadə edin, bayraq pqr=Doğrudur. PDBIO metodları PQR faylları və ziyyətində də istifadə edilə bilər.

Misal: PQR faylinin yazılması

```
>>> io = PDBIO(is_pqr=True)
>>> io.set_structure(s)
>>> io.save("out.pdb")
```

14.1.9 MMTF fayllarının yazılması

MMTF fayl formatına strukturları yazmaq üçün:

```
>>> Bio.PDB.mmtf -dən idxal MMTFIO >>> io =
MMTFIO()
>>> io.set_structure(s)
>>> io.save("out.mmtf")
```

Select sınıfı yuxarıdakı kimi istifadə edilə bilər. Qeyd edək ki, standart MMTF fayllarında olan birləşmə məlumatları, ikinci dərəcəli struktur təyinatı və bəzi digər məlumatlar struktur obyektiindən müəyyən etmək asan olmadığı üçün yazılmır. Bundan əlavə, standart MMTF fayllarında eyni obyektdə qruplaşdırılan molekullar MMTFIO tərəfindən ayrıca obyektlər kimi qəbul edilir.

14.2 Struktur tə msili

Struktur obyektinin ümumi tə rtibatı SMCRA (Struktur/Model/Zə ncirvari/Qalıq/Atom) arxitekturasına uyğundur:

- Struktur modellə rdə n ibarə tdir
- Model zə ncirlə rdə n ibarə tdir
- Zə ncir qalıqlardan ibarə tdir
- Qalıq atomlardan ibarə tdir

Bu, bir çox struktur bioloqların/bioinformatikaçlarının struktur haqqında düşünmə üsuludur və strukturla mə şəkil olmaq üçün sadə, lakin sə mə rə li yol tə qdim edir. Lazım olduqda ə lavə maddə lə rə səsənə lava olunur. Structure obyektinin UML diaqramı (Hazırda Disordered siniflə rini unut) Şəkil 14.1-də göstə rılmışdır. Belə bir mə lumat strukturu strukturun makromolekulyar mə zmununun tə sviri üçün mütləq qən uyğun deyil, lakin strukturu tə svir edən faylda (adətən PDB və ya MMCF faylı) mövcud olan mə lumatların yaxşı şərhi üçün mütləq lazımdır. Bu iyerarxiya strukturu faylinin mə zmununu təmsil edə bilərsə, faylda xəta olduğu və ya nə azı strukturu birmə nali tə svir etmə diyi kifayət qədər dər qısqıdır. SMCRA mə lumat strukturu yaradıla bilərsə, problemdən şübhə lə nmə k üçün əsas var. Beləliklə, mümkün problemləri aşkar etmək üçün bir PDB faylinin təhlili istifadə edilə bilər. Biz 14.7.1-ci bölmədə bunun bir neçə nümunəsini verəcəyik.

Struktur, Model, Zə ncir və Qalıq Müəssisə əsas sinifinin bütün alt sinifləridir. Yalnız Atom sinifi (qismən) Müəssisə alt sinfi üçün siz açar kimi həmin uşaq üçün unikal id-dən istifadə etməklə uşağı çıxara bilərsiniz (məsələn, siz atom adı sətirindən açar kimi istifadə etməklə Qalıq obyektindən Atom obyekti çıxara bilərsiniz, onun zəncir identifikasiatorundan açar kimi istifadə edərək Model obyektindən Zə ncir obyekti çıxara bilərsiniz).

Bozuk atomlar və qalıqlar DisorderedEntityWrapper əsas sinifinin alt sinifləri olan DisorderedAtom və DisorderedResidue sinifləri ilə təmsil olunur. Onlar nizamsızlıqla bağlı mürəkkəbliyi gizlədirər və tam olaraq Atom və Qalıq obyektləri kimi davranırlar.

Ümumiyyətlə, uşaq Müəssisə obyekti (yəni Atom, Qalıq, Zə ncir, Model) onun əsas elementindən çıxarıla bilər. (yəni, Qalıq, Zə ncir, Model, Struktur) açar kimi id istifadə edərək.

```
>>> uşaq_müəssisə = valideyn_müəssisə[uşaq_id]
```

Siz həmçinin əsas Müəssisə obyektinin bütün uşaq Müəssisələrinin siyahısını əldə edə bilərsiniz. Qeyd edək ki, bu siyahı ilə sıralanır xüsusi üsul (məsələn, Model obyektiñdeki Zə ncir obyektləri üçün zəncir identifikasiatoruna görə).

```
>>> uşaq_siyahısı = parent_entity.get_list()
```

Siz həmçinin valideyni uşaqdan ala bilərsiniz:

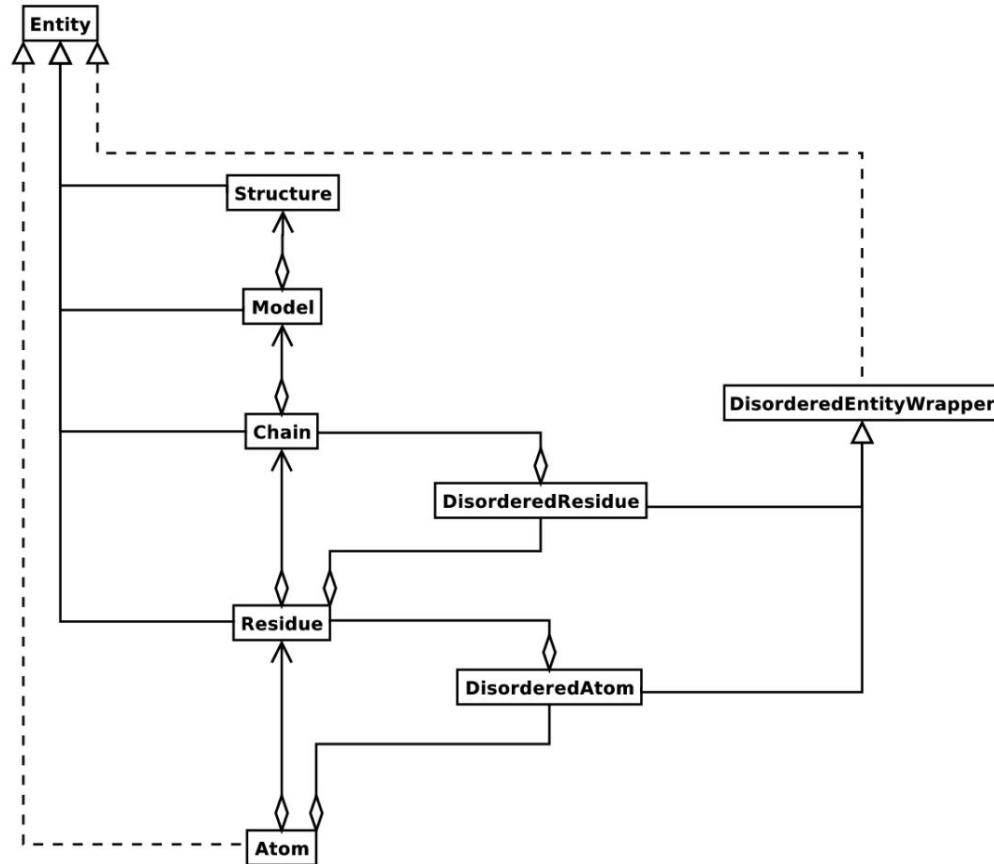
```
>>> valideyn_müəssisə = uşaq_entity.get_parent()
```

SMCRA iyerarxiyasının bütün səviyyələrində siz həmçinin tam id çıxara bilərsiniz. Tam id yuxarı obyektdən (Struktur) başlayaraq cari obyektdə qədər bütün id-ləri ehtiva edən dəstdir. Qalıq obyekti üçün tam id, məsələn, belə bir şeydir:

```
>>> full_id = residue.get_full_id() >>> print(full_id)
("1abc", 0, "A", ("", 10, "A"))
```

Bu uyğun gəlir:

- "1abc" id ilə Struktur



Şəkil 14.1: Makromolekulyar strukturu təmsil etmək üçün istifadə edilən Structure sinfinin SMCRA arxitekturasının UML diaqramı. Brilyantlı tam xətlər aqreqasiyani, oxları olan tam xətlər istinadi, tam üçbucaqlı xətlər irsiyyəti, üçbucaqlı kəsikli xətlər isə interfeysin reallaşmasını bildirir.

- ID 0 olan Model
- "A" id ilə Zə ncir
- İd ilə Qalıq ("", 10, "A")

Qalıq identifikasiatoru qalığın hetero-qalıq (nə də su) olmadığını göstərir, çünkü onun boş hetero sahəsi var, onun ardıcılıq identifikasiatoru 10 və onun daxiletmə kodu "A"dir.

Müəssisənin id-sini əldə etmək üçün get_id metodundan istifadə edin:

```
>>> entity.get_id()
```

Siz has_id metodundan istifadə edərək müəssisədə verilmiş id-yə malik uşaq olub-olmadığını yoxlaya bilərsiniz:

```
>>> entity.has_id(entity_id)
```

Müəssisənin uzunluğu onun uşaqlarının sayına bərabərdir:

```
>>> nr_children = len(müəssisə)
```

Əsas müəssisədə nə uşaq obyektləri silmək, adını dəyişmək, əlavə etmək və s. mümkünündür, lakin bu, hər hansı sağlam düşüncə yoxlamalarını əhatə etmir (məsələn, bir zəncirə eyni id-yə malik iki qalıq əlavə etmək mümkünündür). Bu, həqiqətən, bütövluğun yoxlanılmasını ehtiva edən gözəl bir Dekorator sınıfı vasitəsilə edilməlidir, lakin xam interfeysdən istifadə etmək istəyirsinizsə, koda (Entity.py)నəzər salıbilsiniz.

14.2.1 Struktur

Struktur obyekti iyerarxiyanın yuxarı hissəsindədir. Onun id-si istifadəçi tərəfindən verilmişədir. Struktur bir sıra Model uşaqlarını ehtivadır. Əksər kristal strukturlar (lakin hamısı deyil) bir modeldən ibarətdir, NMR strukturları isə adətən bir neçə modeldən ibarətdir.

Molekulların böyük hissələrinin kristal strukturlarının pozulması da bir neçə modellənəticələnəbilər.

14.2.2 Model

Model obyektinin identifikasiatoru modelin təhlil edilmiş fayldakı mövqeyindən əldə edilən tamədəddir (onlar avtomatik olaraq 0-dan başlayaraq nömrələndir). Kristal strukturların ümumiyyətlə yalnız bir modeli var (id 0 ilə), NMR fayllarında isə adətən bir neçə model var. Bir çox PDB təhlililəri yalnız bir modelin olduğunu güman etdiyi halda, Bio.PDB-də Structure sınıfı elə qurulub ki, o, birdən çox modellə PDB fayllarını asanlıqla idarəedə biləsin.

Nümunə olaraq Structure obyektindən ilk modeli əldə etmək üçün istifadə edin

```
>>> birinci_model = quruluş[0]
```

Model obyekti Zəncir uşaqlarının siyahısını saxlayır.

14.2.3 Zəncir

Zəncir obyekti id-si PDB/mmCIF faylındaki zəncir identifikasiatorundan əldə edilir və tək simvoldur (adətən ha rf). Model obyektindəki hər bir Zəncir unikal identifikasiatora malikdir. Nümunə olaraq, Model obyektindən "A" identifikasiatoru olan Zəncir obyekti əldə etmək üçün istifadə edin

```
>>> zəncir_A = model["A"]
```

Zəncir obyekti Qalıq uşaqlarının siyahısını saxlayır.

14.2.4 Qalıq

Qalıq identifikatoru üç elementdə n ibarə t bir də stdır:

- Hetero-sahə (hetfield): bu
 - Su molekulu üçün 'W'; - 'H_' və digə r hetero qalıqlar üçün qalıq adı (mə sə lə n, qlükoza və ziyyə tində 'H_GLC' molekul);
 - standart amin və nuklein turşuları üçün blank.

Bu sxem 14.4.1-ci bölmə də tə svir olunan sə bə blə rə görə qə bul edilir .

- Ardıcılıq identifikatoru (resseq), zə ncirdə ki qalığın və ziyyə tini tə svir edə n tam ə də d (mə sə lə n, 100);
- Daxiletmə kodu (icode); sə tir, mə sə lə n, 'A'. Daxiletmə kodu bə zə n müə yyə n bir arzu olunan qalıq nömrə lə mə sxemini qorumaq üçün istifadə olunur. Ser 80 daxiletmə mutanti (mə sə lə n, Thr 80 və Asn 81 qalığı arasına daxil edilir) mə sə lə n, aşağıdakı kimi ardıcılıq identifikatorlarına və daxiletmə kodlarına malik ola bilə r: Thr 80 A, Ser 80 B, Asn 81. Bu yolla qalıq nömrə lə mə sxemi və hşı tip strukturunun quruluşuna uyğun olaraq qalır.

Belə liklə , yuxarıdakı qlükoza qalığının id-si ('H GLC', 100, 'A') olacaqdır. Hetero-bayraq və daxiletmə kodu boşdursa, yalnız ardıcılıq identifikatorundan istifadə edilə bilə r:

```
# Tam id
>>> qalıq = zə ncir[(" ", 100, "")]
# Qisayol identifikatoru
>>> qalıq = zə ncir[100]
```

Hetero-bayrağın sə bə bi odur ki, bir çox, çoxlu PDB faylları amin turşusu və hetero-qalıq və ya su üçün eyni ardıcılıq identifikatorundan istifadə edir ki, bu da hetero-bayraqdan istifadə olunmasa, aşkar problemlə r yarada bilə r.

Tə ə cüblü deyil ki, Qalıq obyekti bir sıra Atom uşaqlarını saxlayır. O, hə məcən qalığın adını (mə sə lə n, "ASN") və qalığın seqment identifikatorunu (X-PLOR istifadə çilə rinə yaxşı mə lummur, lakin SMCRA mə lumat strukturunun qurulmasına istifadə edilmir) müə yyə n edə n sə tirdə n ibarə tdir.

Gə lin bə zi nümunə lə rə baxaq. Boş daxiletmə kodu olan Asn 10-da qalıq identifikatoru (' ', 10, ' ') olacaq. Su 10-da qalıq identifikatoru olacaq ('W', 10, " "). Ardıcılıq identifikatoru 10 olan qlükoza molekulunun (qalıq adı GLC olan hetero qalıq) qalıq identifikatoru ('H GLC', 10, ' ') olacaq. Bu şə kildə , üç qalıq (eyni daxiletmə kodu və ardıcılıq identifikatoru ilə) eyni zə ncirin bir hissə si ola bilə r, çünkü onların qalıq id-lə ri fə rqlidir.

Əksə r hallarda hetflag və daxiletmə kodu sahə lə ri boş olacaq, mə sə lə n (' ', 10, ' '). Bu hallarda, ardıcılıq identifikatoru tam id üçün qısa yol kimi istifadə edilə bilə r:

```
# tam id istifadə edin
>>> res10 = zə ncir[(" ", 10, "")]
# qisayoldan istifadə edin
>>> res10 = zə ncir[10]
```

Zə ncir obyektində ki hə r Qalıq obyektinin unikal identifikatoru olmalıdır. Bununla belə , nizamsız qalıqlar hə ll edilir 14.3.3-cü bölmə də tə svir olunduğu kimi xüsusi üsulla .

Qalıq obyektinin bir sıra e lavə üsulları var:

```
>>> residue.get_resname() # qalıq adını qaytarır, mə sə lə n, "ASN" >>> residue.is_disordered() #
ə gə r qalığın nizamsız atomları varsa 1 qaytarır >>> residue.get_segid() # SEGID-i qaytarır, mə sə lə n, "CHN1" >>>
müə yyə n qalıq atomadı varsa.has.has .
```

Qalıq obyektinin amin turşusu olub olmadığını yoxlamaq üçün is aa(qalıq) istifadə edə bilə rsiniz.

14.2.5 Atom

Atom obyekti atomla ə laqə li mə lumatları saxlayır və usaqları yoxdur. Atomun id-si onun atom adıdır (mə sə lə n, Ser qalığının yan zə ncirinin oksigeni üçün "OG"). Atom id-si Qalıqda unikal olmalıdır. Yenə də 14.3.2-ci bölüm də tə svir olunduğu kimi nizamsız atomlar üçün istisna edilir.

Atom id sadə cə atom adıdır (mə sə lə n, 'CA'). Praktikada atom adı hamisinin soyulması ilə yaradılır PDB faylındakı atom adından boşluqlar.

Bununla belə, PDB fayllarında boşluq atom adının bir hissə si ola bilər. Çox vaxt kalsium atomlarını Ca atomlarından ("CA." adlanır) fə riqə ndirmək üçün "CA." adlandırırlar. Boşluqların soyulması problem yarada bilər və hallarda (yə ni eyni qalıqda "CA" adlanan iki atom) boşluqlar saxlanılır.

PDB faylında atom adı adətən aparıcı və arxa boşluqlarla 4 simvoldan ibarətdir. Tez-tez bu boşluqlar istifadə rahatlığı üçün silinə bilər (mə sə lə n, amin turşusu Ca atomu PDB faylında ".CA." kimi etiketlənir, burada nöqtələr boşluqları təmsil edir). Atom adını (və beləliklə, atom identifikatorunu) yaratmaq üçün boşluqlar silinir, əgər bu Qalıqda adların toqquşması ilə nəticələr nəmə yəcəksə (yə ni eyni atom adı və id-si olan iki Atom obyekti).

Sonuncu halda, boşluqlar daxil olmaqla atom adı sınanır. Bu və ziyyət, məsələn, bir qalıqda ".CA" adları olan atomlar olduqda baş verə bilər. və "CA..", baxmayaraq ki, bu çox ehtimal deyil.

Saxlanılan atom məlumatlarına atom adı, atom koordinatları (əgər varsa standart kənarlaşma daxil olmaqla), B faktoru (anizotrop B faktorları və varsa standart yayınma daxil olmaqla), altlok spesifikatoru və boşluqlar daxil olmaqla tam atom adı daxildir. Atom element nömrəsi və ya bəzən PDB faylında göstərilən atom yükü kimi daha az istifadə olunan elementlər saxlanılır.

Atom koordinatlarını manipulyasiya etmək üçün Atom obyektinin çevirmə metodundan istifadə edin. Quraşdırılmış korddan istifadə edin birbaşa atom koordinatlarını təyin etmək üsulu.

Atom obyekti aşağıdakı əlavə üsullara malikdir:

```
>>> a.get_name() # atom adı (boşluqlar silindi, məsələn, "CA") >>> a.get_id() # id (atom
adına bərabə rdir) >>> a.get_coord() # atom koordinatları >>>
a.get_vector() # Vektor obyekti kimi atom koordinatları >>>
a.get_bfactor() # isotropic __ faktoru >>> cupancy >>> a.get_altloc() # alternativ yer
təyinedicisi >>> a.get_sigatm() # atom parametrlərinin standart
kənarlaşması >>> a.get_siguij() # anizotrop B faktorunun standart kənarlaşması >>> a.get_anisou() # anizotrop B faktoru >>>
a.get_fullname() # atomların adı (with.CA, boşluq).
```

Atom koordinatlarını təmsil etmək üçün siguij, anizotrop B faktoru və sigatm Numpy massivlərindən istifadə olunur.

Get vektor metodu atom koordinatları üzərində vektor əməliyyatları etməyi imkan verən Atom obyektinin koordinatlarının Vektor obyekti təsvirini qaytarır. Vector 3D vektor əməliyyatlarının tam dəstini, matris vurma (sol və sağ) və bəzi qabaqcıl fırlanması ilə bağlı əməliyyatları da həyata keçirir.

Bio.PDB-nin Vektor modulunun imkanlarına misal olaraq, fərqli edək ki, siz Gly qalığının Cβ atomunun mövqeyini, əgər varsa, tapmaq istərdiniz. Gly qalığının N atomunu Ca-C bağlı boyunca -120 dərəcədən yuxarı fırlatmaq onu təxminən virtual Cβ atomu və ziyyətinə gətirir. Vektor modulunun rotaxis metodundan (müəyyən ox ətrafında fırlanması yaratmaq üçün istifadə edilə bilər) istifadə edərək bunu necə etmək olar:

```
# atom koordinatlarını vektor kimi əldə edin >>> n =
qaliq["N"].get_vector() >>> c = qaliq["C"].get_vector() >>> ca =
qaliq["CA"].get_vector() # başlanğıcda mərkəz >>> n = n - ca
```

```
>>> c = c - təxminən
```

```
# ca-c vektoru boyunca n # -120 də rə cə firlanan firlanma matrisini
tapın >>> rot = rotaxis(-pi * 120.0 / 180.0, c) # firlanmani ca-n
vektoruna tə tbiq edin >>> cb_at_origin = n.left_multiply(rot) # ca
atomunun üstüne qoyun >>> cb = cborigin
```

Bu nümunə göstərir ki, atom verilənla rü üzə rində olduqca faydalı ola biləcək bəzi qeyri-trivial vektor əməliyyatları etmək mümkündür. Bütün adı vektor əməliyyatları (çarpaz istifadə **) və nöqtə (istifadə *) hasil, bucaq, norma və s.) və yuxarıda qeyd olunan rotaxis funksiyasına əlavə olaraq, Vektor modulunda bir vektoru digərinin üzərinə çevirmək (rotmat) və ya əks etdirmək (refmat) üsulları da var.

14.2.6 Quruluşdan xüsusi atom/qalıq/zəncir/modelin çıkarılması

Bunlar bəzi nümunələrdir:

```
>>> model = struktur[0] >>>
zəncir = model["A"] >>> qalıq
= zəncir[100] >>> atom =
qalıq["CA"]
```

Qeyd edək ki, qisayoldan istifadə edə bilərsiniz:

```
>>> atom = quruluş[0]["A"][100]["CA"]
```

14.3 Narahatlıq

Bio.PDB həm nizamsız atomları, həm də nöqtə mutasiyalarını (yəni, eyni mövqedə olan Gly və Ala qalığını) idarə edə bilər.

14.3.1 Ümumi yanaşma

Bozukluğa iki nöqtəyi-nəzərdən yanaşmaq lazımdır: atom və qalıq nöqtəyi-nəzərdən. Ümumiyyətlə, biz nizamsızlıqdan yaranan bütün mürəkkəbliyi əhatə etmək yə çalışdıq. Əgər siz sadəcə olaraq bütün Ca atomları üzərində dövrə vurmaq istəyirsinizsə, bəzi qalıqların yanına ncirinin nizamsız olmasına əhəmiyyət vermirsınız. Digərtərəfdən, məlumat strukturunda pozğunluğu tamamilə təmsil etmək də mümkün olmalıdır. Buna görə nizamsız atomlar və ya qalıqlar heç bir nizamsızlıq yoxdur kimi davranış xüsusi obyektlərdə saxlanılır. Bu, yalnız nizamsız atomların və ya qalıqların bir hissəsinə təmsil etməklə əhəmiyyət keçirilir. Hansı alt çoxluğun seçildiyi (məsələn, Ser qalığının iki nizamsız OG yanına ncirinin atom mövqelərindən hansı istifadə olunur) istifadəçı tərəfindən müəyyən edilə bilər.

14.3.2 Nizamsız atomlar

Nizamsız atomlar adı Atom obyektləri ilə təmsil olunur, lakin eyni fiziki atomu təmsil edən bütün Atom obyektləri DisorderedAtom obyektiində saxlanılır (bax Şəkil 14.1). Disordered-Atom obyektiində ki hər bir Atom obyekti onun altloc təyin edicisindən istifadə edərək unikal şəkildə indeksləşdiriləbilər. DisorderedAtom obyekti bütün tutulmamış metod çağrılarını seçilmiş Atom obyektiinə yönəldirir, defolt olaraq onun yüksək doluluq olan atomu təmsil edən obyekt. İstifadəçi əlibəttə ki, onun altloc təyin edicisindən istifadə edərək seçilmiş Atom obyektiini dəyişəbilər. Bu şəkildə atom pozğunluğu çox əlavə mürəkkəblik olmadan düzgün şəkildə təmsil olunur. Başqa sözlə, atom pozğunluğu sizi maraqlandırmırsa, bundan narahat olmayacaqsınız.

Hər nizamsız atomun xarakterik altlok identifikatoru var. Siz DisorderedAtom obyektiində yin edə bilərsiniz xüsusi altloc identifikatoru ilə əlaqəli Atom obyekti kimi davranışmalıdır:

```
>>> atom.disordered_select("A") # altloc seçin A atom >>> çap (atom.get_altloc())
"A"
>>> atom.disordered_select("B") # altloc B atomunu seçin >>> çap (atom.get_altloc())
"B"
```

14.3.3 Qaydasız qalıqlar

Ümumi hal

Ə n çox rast gə linə n hal bir və ya bir neçə nizamsız atomu ehtiva edə n qalıqdır. Bu, açıq şə kildə nizamsız atomları tə msil etmə k üçün DisorderedAtom obyektlə rində n istifadə etmə klə və adı Atom obyektlə ri kimi DisorderedAtom obyektini Qalıq obyektində saxlamaqla ha ll olunur. DisorderedAtom, tutulmamış bütün metod çağırışlarını onun tə rkibində ki Atom obyektlə rində n birinə (seçilmiş Atom obyekti) yönəl ndirmə klə , özünü tam olaraq adı atom (ə slində ə n yüksə k doluluğla malik atom) kimi aparacaq.

Nöqtə mutasiyaları

Xüsusi bir və ziyyə t, pozğunluq bir nöqtə mutasiyası ilə ə laqə dar olduqda, yə ni kristalda polipeptidin iki və ya daha çox nöqtə mutantları olduqda yaranır. Bunun nümunə sini PDB strukturunda 1EN2 tapmaq olar.

Bu qalıqlar fə rqli qalıq tipinə aid olduqları üçün (mə sə lə n, deyə k ki, Ser 60 və Cys 60) ümumi halda olduğu kimi tə k Qalıq obyektində saxlanılmamalıdır. Bu halda, hə r bir qalıq bir Qalıq obyekti ilə tə msil olunur və hə r iki Qalıq obyekti bir DisorderedResidue obyektində saxlanılır (bax Şəkil 14.1).

DisorderedResidue obyekti bütün tutulmamış metodları seçilmiş Qalıq obyektinə yönəl ndirir (defolt olaraq sonuncu Qalıq obyekti ə lavə olunur) və belə liklə , adı qalıq kimi davranır. DisorderedResidue obyektində ki hə r bir Qalıq obyekti qalıq adı ilə unikal şə kildə müə yyə edilə bilə r. Yuxarıdakı misalda Ser 60 qalığının DisorderedResidue obyektində "SER" identifikatoru, Cys 60 qalığının isə "CYS" id-si olacaq.

İstifadə çı bu id vasitə silə DisorderedResidue obyektində aktiv Qalıq obyektini seçə bilə r.

Misal: fə rz edə k ki, zə ncirin Ser və Cys qalığından ibarə t 10-cu mövqedə nöqtə mutasiyası var.

Bu zə ncirin 10 qalığının Cys qalığı kimi davrandığından ə min olun.

```
>>> qalıq = zə ncir[10] >>>
residue.disordered_select("CYS")
```

Bundan ə lavə , siz (Disordered)Residue obyektiinin paketdə n çxarılan siyahı ə ldə etmə metodundan istifadə edə rə k, bütün Atom obyektlə rinin siyahısını ə ldə edə bilə rsiniz (yə ni, bütün DisorderedAtom obyektlə ri fə rdi Atom obyektlə rinə "açılır").

14.4 Hetero qalıqlar

14.4.1 Ə laqə li problemlə r

Hetero qalıqlarla bağlı ümumi problem eyni zə ncirdə mövcud olan bir neçə hetero və qeyri-hetero qalıqların eyni ardıcılıq identifikatorunu (və daxiletmə kodunu) paylaşmasıdır. Buna görə də , hə r bir hetero qalıq üçün unikal id yaratmaq üçün sular və digə r hetero qalıqlar fə rqli üsullarla işlə nir.

Yadda saxlayın ki, Qalıq obyektində id kimi də ftə r (hetfield, resseq, icode) var. Hetfield amin və nuklein turşuları üçün boş (" "), sular və digə r hetero qalıqlar üçün isə sə tirdir. Hetfieldin mə zmunu aşağıda izah edilir.

14.4.2 Su qalıqları

Su qalığının hetfield sə tri "W" hə rfında n ibarə tdir. Belə liklə , bir su üçün tipik qalıq identifikatoru ("W", " ") olur. 1,

14.4.3 Digər hetero qalıqlar

Digər hetero qalıqlar üçün hetfield sətri "H" ilə başlayır, sonra qalıq adı gəlir. Qalıq adı "GLC" olan qlükoza molekulunun hetfield "H GLC" olacaqdır. Onun qalıq identifikasiatoru məsələn ola bilər ("H GLC", 1,-) .

14.5 Struktur obyekti üzrə naviqasiya

PDB faylini təhlil edin və bəzi Model, Zəncir, Qalıq və Atom obyektlərinini çıxarıın

```
>>> Bio.PDB.PDBParser -dən idxl PDBParser >>> parser =
PDBParser() >>> structure =
parser.get_structure("test", "1fat.pdb") >>> model = struktur[0] >>> zəncir =
model["A"] >>> qalıq = zəncir[1]
>>> atom = qalıq["CA"]
```

Bir strukturun bütün atomları arasında təkrarlanır

```
>>> p = PDBParser() >>> strukturu
= p.get_structure("X", "pdb1fat.ent") >>> strukturdakı model üçün :
...
...     modelda zəncir üçün :
...         zəncirdə ki qalıq üçün :
...             qalıqdakı atom üçün :
...                 çap (atom)
...             ...
```

Bir strukturdakı bütün atomları təkrarlamak istəyirsizsə, qısa yol var:

```
>>> atomlar = structure.get_atoms()
>>> atomlardakı atom üçün :
...
...     çap (atom)
...
Eynilə, bir zəncirdə ki bütün atomları təkrarlamak üçün istifadə edin
>>> atomlar = chain.get_atoms()
>>> atomlardakı atom üçün :
...
...     çap (atom)
...     ...
```

Modelin bütün qalıqları üzərində iterasiya

və ya modeldəki bütün qalıqları təkrarlamak istəyirsizsə :

```
>>> qalıqlar = model.get_residues() >>> qalıqlardakı qalıq üçün :
...
...     çap (qalıq)
...
...
```

Siz həmçinin bir strukturdan bütün qalıqları elde etmək üçün Selection.unfold_entities funksiyasından istifadə edə bilərsiniz:

```
>>> res_list = Selection.unfold_entities(struktur, "R")
```

və ya bir zə ncirdən bütün atomları almaq üçün:

```
>>> atom_list = Selection.unfold_entities(zə_ncir, "A")
```

Aydındır ki, A=atom, R=qalıq, C=zə ncir, M=model, S=struktur. Siz yuxarı qalxmaq üçün bundan istifadə edə bilərsiniz iyerarxiya, məsələn, Atomlar siyahısından (unikal) Qalıq və ya Zə ncir valideynlərin siyahısını əldə etmək üçün:

```
>>> residue_list = Selection.unfold_entities(atom_list, "R")
>>> chain_list = Selection.unfold_entities(atom_list, "C")
```

Əlavə məlumat üçün API sənədlərinə baxın.

Zə ncirdən hetero qalıqları çıxarın (məsələn, qlükoza (GLC) hissəsi resseq 10 ilə)

```
>>> residue_id = ("H_GLC", 10, " ")
>>> qalıq = zə_ncir[residue_id]
```

Bütün hetero qalıqları zə ncirdə çap edin

```
>>> chain.get_list()-də ki qalıq üçün:
...
...     residue_id = residue.get_id()
...     hetfield = residue_id[0]
...     ə gər hetfield[0] == "H":
...         çap (qalıq_id)
...
```

B faktoru 50-dən yuxarı olan strukturdakı bütün CA atomlarının koordinatlarını çap edin

```
>>> model üçün structure.get_list():
...     model.get_list() -də zə ncir üçün:
...         chain.get_list()-də ki qalıq üçün:
...             ə gər residue.has_id("CA"):
...                 ca = qalıq["CA"]
...                 ə gər ca.get_bfactor() > 50.0:
...                     çap (ca.get_coord())
...
```

Tərkibində nizamsız atomlar olan bütün qalıqları çap edin

```
>>> model üçün structure.get_list():
...     model.get_list() -də zə ncir üçün:
...         chain.get_list()-də ki qalıq üçün:
...             ə gər residue.is_sordered():
...                 resseq = residue.get_id()[1]
...                 resname = residue.get_resname()
...                 model_id = model.get_id()
...                 chain_id = chain.get_id()
...                 çap (model_id, zə_ncir_id, adının dəyişdirilməsi, resseq)
...
```

Bütün nizamsız atomlar üzə rində dövr edin və altlok A olan bütün atomları seçin (ə gə r varsa)

Bu, SMCRA mə lumat strukturunun yalnız A altloklu atomların mövcud olduğu kimi davranışına ə min olacaq.

```
>>> quruluş.get_list()-də model üçün:  
...     model.get_list () - də zə ncir üçün :  
...         chain.get_list()- də ki qalıq üçün :ə gə r  
...             residue.is_disordered(): residue.get_list  
...                 (): atom.is_disordered ():ə gə r  
...                     atom.disordered_id("A");  
...                         atom.disordered_select ("A")  
...  
...
```

Struktur obyektində n polipeptidlə rin çıxarılması

Strukturdan polipeptidlə ri çıxarmaq üçün PolypeptideBuilder-də n istifadə edə rə k Struktur obyektində n Polipeptid obyektlə rinin siyahısını aşağıdakı kimi qurun:

```
>>> model_nr = 1 >>>  
polypeptide_list = build_peptides(struktur, model_nr) >>> polipeptid_siyahısındaki polipeptid  
üçün :  
...     çap (polipeptid)  
...
```

Polipeptid obyekti sadə cə olaraq Qalıq obyektlə rinin istifadə ci Siyahısıdır və həmişə bir Modeldə n yaradılır (bu halda model 1). Ardıcılığı Seq obyekti kimi ə ldə etmə k və ya Cα atomlarının siyahısını ə ldə etmə k üçün ortaya çıxan Polipeptid obyektində n istifadə edə bilə rsiniz. Polipeptidlə r CN və ya Ca-Cα mə safə meyarından istifadə etmə klə tikilə bilə r.

Misal:

```
#  
ppb.build_peptides(struktur)  
-da pp üçün CN >>> ppb = PPBuilder() >>> istifadə  
...     edə rə k: print(pp.get_sequence())  
...  
# CA-CA-dan  
istifadə >>> ppb =  
CaPPBuilder() >>> ppb.build_peptides ( struktur): print  
...     (pp.get_sequence())  
...
```

Qeyd edə k ki, yuxarıdakı halda strukturun yalnız 0 modeli PolypeptideBuilder tə rə fində n nə zə rdə n keçirilir. Bununla belə , Model və Zə ncir obyektlə rində n Polipeptid obyektlə ri qurmaq üçün PolypeptideBuilder-də n istifadə etmə k mümkündür.

Bir strukturun ardıcılığının ə ldə edilmə si

Edilə cə k ilk bütün polipeptidlə ri strukturdan çıxarmaqdır (yuxarıda olduğu kimi). Bundan sonra hə r bir polipeptidin ardıcılığı Polipeptid obyektlə rində n asanlıqla ə ldə edilə bilə r. Ardıcılıq Biopython Seq obyekti kimi tə qdim olunur.

Misal:

```
>>> seq = polypeptide.get_sequence() >>> seq
```

```
Seq('SNDIYFNFQRFNETNLILQRDASVSSSGQLRLTNLN')
```

14.6 Strukturların təhlili

14.6.1 Məsafələrin ölçüməsi

İki atom arasındaki məsafəni qaytarmaq üçün atomlar üçün mənfi operator həddindən artıq yükənmisdir.

```
# Bəzi atomlar alın >>> ca1
```

```
= qalıq1["CA"] >>> ca2 = qalıq2["CA"]
```

```
# Məsafəni almaq üçün atomları çıxmaq kifayatdır >>> məsafə = ca1 - ca2
```

14.6.2 Bucaqların ölçüməsi

Atom koordinatlarının vektor təsvirindən və Vektor modulundan hesablama bucağı funksiyasından istifadə edin:

```
>>> vektor1 = atom1.get_vektor() >>> vektor2 =
atom2.get_vektor() >>> vektor3 = atom3.get_vektor()
>>> bucaq = hesab_bucaq(vektor1, vektor2, vektor3)
```

14.6.3 Burulma bucaqlarının ölçüməsi

Atom koordinatlarının vektor təsvirindən və Vektor modulundan calc_dihedral funksiyasından istifadə edin:

```
>>> vektor1 = atom1.get_vector() >>> vektor2 =
atom2.get_vektor() >>> vektor3 = atom3.get_vector()
>>> vektor4 = atom4.get_vector() >>> bucaq =
hesablama_dihedral(vektor1, vektor2, vektor3,
vektor4)
```

14.6.4 Daxili koordinatlar - məsafələr, bucaqlar, burulma bucaqları, məsafə qrafikləri, və s

Protein strukturları normal olaraq PDB və ya mmCIF faylında olduğu kimi sabit mənşəyə nisbətən 3D XYZ koordinatlarında verilir. Daxili koordinatlar modulu bu sistemi bağuzunluqlarına, bucaqlara və dihedral bucaqlara çevirməyi asanlaşdırır. Zülal strukturları üzrə standart psi, phi, chi və s. hesablamaları dəstəkləməklə yanaşı, bu təqdimat tərcümə və fırıldanma üçün invariantdır və təbiq struktur analizi üçün çoxlu üstünlükəri ortaya qoyur.

Sonrakı nümunələr üçünəvvəlca burada bəzi modulları yükəyin:

```
>>> Bio.PDB.PDBParser -dən PDBParser idxalı >>> Bio.PDB.Chain
idxal Zəncirindən >>> Bio.PDB.internal_coords -dan
idxal * >>> Bio.PDB.PICIO -dan import write_PIC, read_PIC,
read_PIC_seq >>> Bio.PDB.ic_rebuild strukturundan , write_rebuild strukturundan >>> _plicate
Bio.PDB.SCADIO import write_SCAD >>> Bio.Seq -dən idxal Seq >>> Bio.SeqRecord -dan SeqRecord >>> Bio.PDB.PDBIO- dan
idxal PDBIO >>> idxal numpy kimi np
```

14.6.4.1 Dihedrallara, bucaqlara və bağ uzunluqlarına daxil olmaq

Bir struktur üçün daxili koordinatları hesablamaq üçün sadə və ziyyətlə başlayırıq:

```
>>> # strukturu normal şəkilde yükləyin, ilk zənciri əldə edin >>> parser
= PDBParser() >>> myProtein =
parser.get_structure("1a8o", "1A8O.pdb") >>> myChain = myProtein[0]["A"]
```

```
>>> # bağ uzunluqlarını, bucaqları, dihedral bucaqları hesablayın >>>
myChain.atom_to_internal_coordinates(verbose=True) MaxPeptideBond (1.4
angstroms) səbəbindən THR 186-da zəncirvari qırılma THR 216-da zəncir kəsilməsin keçdi (MaxPeptide-i
aşdı)
```

1A8O üçün zəncirvari qırılma xəbərdarlığı yuxarıdakı verbose=True seçimini silməklə aradan qaldırılır. Fasilə yaratmamaq və bunun
əvəzinə qeyri-real uzun NC istiqrazlarına icazə vermək üçün MaxPeptideBond sınıfının lağvı edin, məsələn:

```
>>> IC_Chain.MaxPeptideBond = 4.0 >>>
myChain.internal_coord = Yox # yeni komşularla ilə struktur datasını yenidən yükleməyi məcbur edin >>>
myChain.atom_to_internal_coordinates(verbose=Doğru)
```

Bu nöqtədə yərlər həm zəncir, həmdə qalıq səviyyəsində mövcuddur. 1A8O-nun ilk qalığı HETATM MSE-dir
(selenomethionine), ona görə də biz kanonik adları və ya atom təyin edicilərinde n-istifadə edərək aşağıda 2-ci qalığı araşdırırıq. Burada
ad və atom ardıcılığına görə chi1 dihedral və tau bucaqlarını və atom cütünü təyin etməklə Ca-Cβ məsafəsinin əldə edirik:

```
>>> r2 = myChain.child_list[1] >>> r2
```

```
<Qalıq ASP het=resseq=152 icode=> >>> r2ic =
r2.internal_coord >>> print(r2ic, ":", 
r2ic.pretty_str(), ":", r2ic.rbase, ":", r2ic.ic) ('1a8o', ' ', ' ', '0') : ASP 152 : (152, Yoxdur, 'D') : D
```

```
>>> r2chi1 = r2ic.get_angle("chi1") >>> çap
(də yirmi(r2chi1, 2))
-144.86

>>> r2ic.get_angle("chi1") == r2ic.get_angle("N:CA:CB:CG")
Doğrudur

>>> çap(round(r2ic.get_angle("tau"), 2)) 113.45

>>> r2ic.get_angle("tau") == r2ic.get_angle("N:CA:C")
Doğrudur

>>> çap(round(r2ic.get_length("CA:CB"), 2)) 1.53
```

Chain.internal coord obyekti-hedra (3 bağlı atom) və dihedra (4 bağlı atom) obyektlərinin massivlərinin və lügətlərinin saxlayır.
Lügətlər AtomKey obyektlərinin dəstələri ilə indekslənir; AtomKey obyektləri qalıq mövqeyini, daxiletmə kodunu, 1 və ya 3 simvoller
qalıq adını, atom adını, altlok və işçəli əlavə keçirir.

Aşağıda Zəncir massivlərinin birbirə indeksləşdirilməklə yuxarıdakı kimi eyni chi1 və tau bucaqlarını əldə edirik.
Zəncir massivlərinin indeksləşdirilməsi üçün AtomKeys:

```
>>> myCic = myChain.internal_coord
```

```
>>> r2chi1_object = r2ic.pick_angle("chi1") >>> # və ya eyni şey (yuxarıdakı
get_angle() üçün olduğu kimi): >>> r2chi1_object == r2ic.pick_angle("N:CA:CB:CG")
```

Doğrudur

```
>>> r2chi1_key = r2chi1_object.atomkeys >>> r2chi1_key # r2chi1_key
AtomKeys də stidir (152_D_N, 152_D_CA, 152_D_CB, 152_D(CG)
```

```
>>> r2chi1_index = myCic.dihedraNdx[r2chi1_key] >>> # və ya eyni şey: >>
r2chi1_index == r2chi1_object.ndx
```

Doğrudur

```
>>> çap(round(myCic.dihedraAngle[r2chi1_index], 2)) -144.86
```

>>> # hə məçinim:

```
>>> r2chi1_object == myCic.dihedra[r2chi1_key]
```

Doğrudur

```
>>> # hedra bucaqları oxşardır: >>> r2tau =
r2ic.pick_angle("tau") >>> print(round(myCic.hedraAngle[r2tau.ndx],
2)) 113.45
```

Zə ncir sə viyyə sində bağ uzunluğu mə lumatlarını ə ldə etmə k daha mürə kkə bdır (və tövsiyə edilmir). Göstə rildiyi kimi burada birdə n çok hedra fə rqli mövqelə rdə bir istiqraz paylaşacaq:

```
>>> r2CaCb = r2ic.pick_length("CA:CB") # istiqrazi ehtiva edə n hedra siyahısını qaytarır >>> r2CaCb[0][0].atom düymə lə ri (152_D_CB, 152_D_CA,
152_D_C) >>> çap(round(myCic.hedraL12[r2CaCb[0]
[0].ndx], 2)) # mövqe 1-2 1.53
```

```
>>> r2CaCb[0][1].atom düymə lə ri (152_D_N,
152_D_CA, 152_D_CB) >>> çap
(round(myCic.hedraL23[r2CaCb[0][1].ndx], 2)) # mövqe 2-3 1.53
```

```
>>> r2CaCb[0][2].atom düymə lə ri (152_D_CA,
152_D_CB, 152_D(CG) >>> çap
(round(myCic.hedraL12[r2CaCb[0][2].ndx], 2)) # mövqe 1-2 1.53
```

Ə və zində Qalıq sə viyyə si tə yin edilmiş uzunluq funksiyasından istifadə edin.

14.6.4.2 Konstruksiyaların tamliğinin yoxlanılması

İtkin atomlar və digə r problemlə r strukturun yenidə n qurulması zamanı problemlə r yarada bilə r. Quruluşun tə miz yenidə n qurulması üçün kifayə t qə də r mə lumatın olub olmadığını tez müə yyə n etmə k üçün strukturun yenidə n qurulması testində n istifadə edin. Ə lava et verbose=True və /yaxud daha ə traflı mə lumat üçün nə tica lügə tini yoxlayın:

```
>>> # myChain-in mə nasını yoxlayın (bucaqlar ə ldə edə və eyni quruluşu yenidə n qura bila r) >>> resultDict =
structure_rebuild_test(myChain) >>> resultDict["keç"]
```

Doğrudur

14.6.4.3 Konstruksiyaların də yişdirilməsi və yenidən qurulması

Zəncir strukturlarına birbaşa daxil olmaqdansa, daxili koordinatları də yişdirmək üçün qalıq səviyyəsinin təyin edilmiş bucağı və təyin olunmuş uzunluq qurğularından istifadə etmək daha məqsəd uyğundur. Hedra bucaqlarının birbaşa də yişdirilməsi təhlükəsiz olsa da, yuxarıda qeyd edildiyi kimi bağ uzunluqları çoxlu üst-üstə düşən hedrada görünür və bu, müəyyən edilmiş uzunluqla idarə olunur. Dihedral bucağı tətbiq edildikdən, təyin edilmiş bucaq nəticəni +/-180-ə bükəcək və bitişik dihedranı da döndərəcək (məsələn, izolösin chi1 bucağı üçün hər iki bağ -yə qızın ki, istədiyiniz budur).

```
>>> # qalıq 2 chi1 bucağını -120 dərəcədə çevirin >>> r2ic.set_angle("chi1", r2chi1 - 120.0) >>> print(round(r2ic.get_angle("chi1"), 2)) 95.14
```

```
>>> r2ic.set_length("CA:CB", 1.49) >>>
print(round(myCic.hedraL12[r2CaCb[0][0].ndx], 2)) # Cb-Ca-C mövqeyi 1-2 1.49
```

Daxili koordinatlardan strukturun yenidən qurulması daxili_to_atom_coordinates():

```
>>> myChain.internal_to_atom_coordinates()
```

```
>>> # yalnız sübut üçün: >>>
myChain.internal_coord = Yoxdur # bütün daxili_koord məlumatları silindi, yalnız atomlar qaldı >>> myChain.atom_to_internal_coordinates() # daxili koordinatlari yenidən yaradın >>> r2ic = myChain.child_list[1].internal_coord (r2ic).set_value(95.14) # ölçülümiş dərəcənin 95.14-ndən yuxarıda təyin olunanlara uyğun olduğunu göstərin
```

```
>>> çap(round(myCic.hedraL23[r2CaCb[0][1].ndx], 2)) # N-Ca-Cb mövqeyi 2-3 1.49
```

Yaradılan struktur normal olaraq PDBIO ilə yazılıa bilər:

```
write_PDB(myProtein, "myChain.pdb") # və ya
sadəcə başlıqsız ATOM qeydləri: io = PDBIO()
io.set_structure(myProtein)
io.save("myChain2.pdb")
```

14.6.4.4 Protein Daxili Koordinatı (.pic) faylları və standart dərəcələr

Protein zəncirlərinin ilkin koordinatlara nisbətən hedra və dihedra kimi təsvir etmək üçün PICIO modulunda fayl formatı müəyyən edilmişdir. Faylin qalıq ardıcılılığı məlumatından başqa bütün hissələri (məsələ s. ('1A8O', 0, 'A', ' ', 153, ' ') ILE) istəgə bağlıdır və göstərilmədikdə defolt dərəcələr doldurulacaq və oxunmuş PIC defaults=True seçimi ilə çağırılır. Defolt dərəcələr sentyabr 2019-cu il Dunbrack cullpdb_pc20 res2.2 R1.0 tarixindən hesablanır.

Burada biz 'myChain'i daxili koordinat spesifikasiyalarının .pic faylı kimi yazılıq və sonra onu 'myProtein2' olaraq yenidən oxuyuruq.

```
# zəncir 'zülal daxili koordinatları' (.pic) faylı olaraq yazın write_PIC(myProtein,
"myChain.pic") # oxumaq .pic faylı myProtein2 =
read_PIC("myChain.pic")
```

Bütün daxili koordinat dərəcələri defoltlarla əvəz edilə bildiyinə görə, PICIO.read_PIC seq olaraq verilir. Bir giriş ardıcılığından etibarlı (əsasən spiral) standart struktur yaratmaq üçün faydalı funksiya:

```
# .pic faylı olaraq oxumaqla tə sadüfi ardıcılıq üçün standart struktur yaradın myProtein3 =
```

```
read_PIC_seq( SeqRecord( Seq("GAVLIMFPSTCNQYWDEHKR"),
    id="1RND",
    description="mə nim tə sadüfi ardıcılığım",
)

) myProtein3.internal_to_atom_coordinates() write_PDB(myProtein3,
"myRandom.pdb")
```

Daxili koordinatlardan strukturlar yaradan zaman omeqa bucaqlar (180.0), hedra bucaqları və /yaxud bağ uzunuqlarında tə lə b olunan də qıqliyi araşdırmaq maraqlı ola bilə r. PIC yazmaq üçün picFlags seçimi buna imkan verir və defolt də yə rlə ri ə ldə etmə k üçün .pic faylinə mə lumatların seçilmə sinə və qeyd olunmamış qalmasına imkan verir.

Müxtə lif kombinasiyalar mümkündür və bə zi ə vvə lcə də n tə yinlə r verilir, mə sə lə n klassik yalnız yazacaq psi, phi, tau, prolin omeqa və yan zə ncir chi bucaqlarını .pic faylinə:

```
write_PIC(myProtein, "myChain.pic", picFlags=IC_Residue.pic_flags.classic) myProtein2 = read_PIC("myChain.pic",
defaults=Doğru)
```

14.6.4.5 Bütün atomlu AtomArray-a daxil olmaq

Biopython Atom obyektlə rində ki bütün 3D XYZ koordinatları Zə ncir sinfində tə k böyük massivə köçürürlür və atomun daxili koordinatlara ilk addimında bu massivdə Numpy "görünüşlə r" ilə ə və z olunur. Biopython Atom koordinatlarına daxil olan program tə minatı tə sirlə nmir, lakin yeni massiv gə lə cə k iş üçün sə mə rə lilik tə klif edə bilə r.

Atom XYZ koordinatlarından fə rqli olaraq, AtomArray koordinatları homogendir, yə ni dördüncü element kimi 1,0 olan [xyz 1.0] kimi massivlə rdir. Bu, daxili koordinatlar modulu boyunca birlə şdirilmiş tə rcümə və firlanma matrislə rində n istifadə edə rə k sə mə rə li transformasiyanı asanlaşdırır. AtomKeylə ri koordinatlarına uyğunlaşdırılan müvafiq AtomArrayIndex lügə ti var.

Burada massivdə n müə yyə n bir Cβ atomu üçün oxu koordinatlarını nümayiş etdiririk, sonra massiv də yə rinin də yişdirilmə sinin Atom obyektini eyni zamanda də yişirdiyini göstə ririk:

```
>>> # zə ncir üçün bütün atomların massivinə daxil olun, mə sə lə n, yuxarıdakı r2 qalıqdır 152 C-beta >>> r2_cBeta_index =
myChain.internal_coord.atomArrayIndex [AtomKey("152_D_CB")]
>>> r2_cBeta_coords
myChain.internal_coord.atomArray[r2_cBeta_index] >>> çap (np.round(r2_cBeta_coords, 2)) [-0,75 -1,18 -0,51 1. ]
```

```
>>> # Biopython Atom koord massivi indi atomArray-a bir görünüşdür, buna görə >>> assert r2_cBeta_coords[1]
== r2["CB"].coord[1] >>> r2_cBeta_coords[1] += 1.0 # Y koordu 1 ang = 1
angstrom >> [1coord] də yişdirin [1coord] >>> r2["CB"].coord[1] >>> # onlar həmişə eynidir (eyni
yaddaşı paylaşırlar) >>> r2_cBeta_coords[1] -= 1.0 # bə rpa
```

Qeyd edə k ki, Atom koord massivlə ri ilə atomArray zə nciri arasında görünüş ə laqə sini "qırmaq" asandır. Atom koordinatlarını birbaşa də yişdirə rkə n, bunun qarşısını almaq üçün element-element surə ti üçün sintaksisdə n istifadə edin:

```
# bunlardan istifadə edin:
myAtom1.coord[:] = myAtom2.coord
myAtom1.coord [...] = myAtom2.coord
```

```
myAtom1.coord[:] = [1, 2, 3]
diapazondaki i üçün (3):
    myAtom1.coord[i] = myAtom2.coord[i]
```

istifadə etmə yin:

```
myAtom1.coord = myAtom2.coord
myAtom1.coord = [1, 2, 3]
```

AtomArrayIndex-də n və AtomKey sınıfı biliklə rində n istifadə biza Numpy 'selektorları' yaratmağa imkan verir, yalnız Ca atomunun koordinatlarından ibarət massiv çıxarmaq üçün aşağıda göstərildiyi kimi:

```
>>> # bütün atom massivində n yalnız C-alfa atomlarını süzgə cdə n keçirmək üçün selektor yaradin >>
atmNameNdx = AtomKey.fields.atm >>> aaI =
myChain.internal_coord.atomArrayIndex >>> CaSelect =
[aaI.get(k) aaI.keys ()d ]km- də k üçün [aaI.get(k) ə gə r "[ N] x "m. >>> # indi C-alfa atomu
koordinatlarının sıralanmış massivi belədir: >>> CA_coords =
myChain.internal_coord.atomArray[CaSelect] >>> # qeyd edin ki, bu, Numpy
fantastik indekslə şdirmədə n istifadə edir, ona görə də CA_coords yeni nüsxədir >>> #
(ə gə r onu də yişdirse niz, orijinal atomarray) tə sirsizdir.
```

14.6.4.6 Mə safə sahə ləri

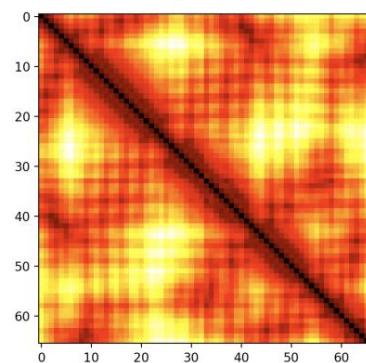
AtomArray-in üstünlüyü ondan ibarətdir ki, ondan mə safə qrafası yaratmaq Numpy kodunun bir sətridir:

```
np.linalg.norm(atomArray[:, Yoxdur, :] - atomArray[Yox, :, :], ox=-1)
```

Qısa olmasına baxmayaraq, idiom camını yadda saxlamaq çətindir və yuxarıdakı formada arzu olunduğu kimi klassik Ca sujetindən daha çox bütün atom mə safə ləri yaradır. Mə safə xətti metodu yuxarıdakı xətti ə hatədir və əvvəlki bölmədə müəyyən edilmiş CaSelect kimi isteğe bağlı seçici qəbul edir. Şəkil 14.2-ə baxın.

```
# C-alfa mə safə si qrafiki yaradin
caMə safə lə r = myChain.internal_coord.distance_plot(CaSelect) # məsələn,
Matplotlib ilə göstərin: matplotlib.pyplot
plt olaraq idxal edin

plt.imshow(caMə safə lə r, cmap="isti", interpolation="ən yaxın") plt.show()
```



Şəkil 14.2: PDB faylı 1A8O (HIV capsid C-terminal domeni) üçün Ca mə safə qrafiki

14.6.4.7 Mə safə li sahə də n strukturun qurulması

Bütün atom mə safə si qrafiki zülal strukturunun başqa bir tə sviridir, eyni zamanda tə rcümə və firlanma üçün invariantdır, lakin xiralliq mə lumati yoxdur (güzgü tə sviri strukturu eyni mə safə sahə sini yaradacaq). Mə safə matrisini hə r dihedral bucağın işaretə lə ri ilə birlə şdirə rə k daxili koordinatları bə rpa etmə k mümkündür.

Bu işdə <https://math.stackexchange.com/a/49340/409> saytında müzakirə edilə n Hedronometer Blue tə rə fində n hazırlanmış tə nliklə rdə n istifadə olunur. daha sonra <http://daylateanddollarshort.com/mathdocs/Heron-like-Results-for-Tetrahedra.pdf>-də .

Başlamaq üçün 'myChain'-də n mə safə lə r və xiralliq də yə rlə rini çıxarıraq:

```
>>> # mə safə də n struktur qurmaq:
```

```
>>> ## bütün atom mə safə si qrafikini yaradin >>> mə safə lə r
= myCic.distance_plot() >>> ## dihedral bucaqların
işara lə rini a ldə edin >>> xiralliq = myCic.dihedral_signs()
```

Onu düzgün şə kildə yenidə n qurmaq üçün "myChain"-ə uyğun gə lə n etibarlı mə lumat strukturuna ehtiyacımız var; yuxarıda oxunan PIC ardıcılığından istifadə ümumi halda işlə yə cə k, lakin burada istifadə olunan 1A8O nümunə si yalnız ardıcılığın yaratmayıacağı bə zi ALTLOC mürə kkə bliyinə malikdir. Nümayiş üçün ə n asan yanaşma sadə cə 'myChain' strukturunu tə krarlaşıqdır, lakin biz bütün atom və daxili koordinat zə nciri massivlə rini 0-a tə yin etdik (yalnız nümayiş üçün) orijinal strukturdan heç bir mə lumatin olmadığına ə min olmaq üçün:

```
>>> ## yeni, boş mə lumat strukturu a ldə edin : data strukturunu myChain-də n köçürün >>> myChain2 =
IC_duplicate(myChain)[0]["A"] >>> cic2 = myChain2.internal_coord
```

```
>>> ## yeni atomArray və di/hedra də yə r massivlə rini tə mizlə yin, sada cə sübut üçün >>> cic2.atomArray =
np.zeros((cic2.AAsiz, 4), dtype=np.float64) >>> cic2.dihedraAngle[:] = 0.0 >>> cic2.hedra = []Ang .
cic2.hedraL12[:] = 0.0 >>> cic2.hedraL23[:] = 0.0
```

Yanaşma daxili koordinatları mə safə qrafiki mə lumatlarından bə rpa etmə k, sonra yuxarıda göstə rildiyi kimi daxili koordinatlardan atom koordinatlarını yaratlaşdırır. Son yaradılan strukturu başlangıç strukturla eyni koordinat mə kanına yerlə şdirmə k üçün biz yalnız ilk üç N-Ca-C atomunun koordinatlarını 'myChain' zə ncirinin başlangıcından 'myChain2' strukturuna köçürüük (bu, yalnız sonunda ekvivalentliyi nümayiş etdirmə k üçün lazımdır):

```
>>> ## strukturların üst-üstə düşmə si üçün yalnız ilk N-Ca-C koordinatlarını köçürün: >>>
cic2.copy_initNCaCs(myChain.internal_coord)
```

Daxili koordinatlara olan mə safə , hə-də f struktur üçün hə r dihedron üçün altı atomlararası mə safə də n ibarə t massivlə rə ehtiyac duyur. Rahatlıq rutininin distplot to dh massivlə ri bu də yə rlə ri lazım olduqda ə vvə llə r yaradılmış mə safə matrisində n çıxarıır və bu mə lumatları Chain.internal coords obyektində ki massivlə rə yazmaq üçün istifadə ci metodu ilə ə və z edilə bilə r.

-

```
>>> ## mə safə lə ri zə ncirvari massivlə rə köçürün: >>>
cic2.distplot_to_dh_arrays(mə safə lə r, xiralliq) >>> ## mə safə lə rdə n bucaqları
və dihedral bucaqları hesablayın: >>> cic2.distance_to_internal_coordinates()
```

Aşağıdakı addımlar yeni yaradılan 'myChain2' daxili koordinatlarından atom koordinatlarını yaradır, sonra bütün də yərlərin PDB faylından daha yaxşı uyğunlaşdırılmasını təsdiqləmək üçün Numpy allclose rutinindən istifadə edin:

```
>>> ## daxili koordinatlardan XYZ koordinatlarını yaradin: >>>
myChain2.internal_to_atom_coordinates() >>> ## atomArray
nəticəsinin orijinal struktura uyğun olduğunu təsdiqləyin: >>> np.allclose(cic2.atomArray,
myCic.atomArray)
```

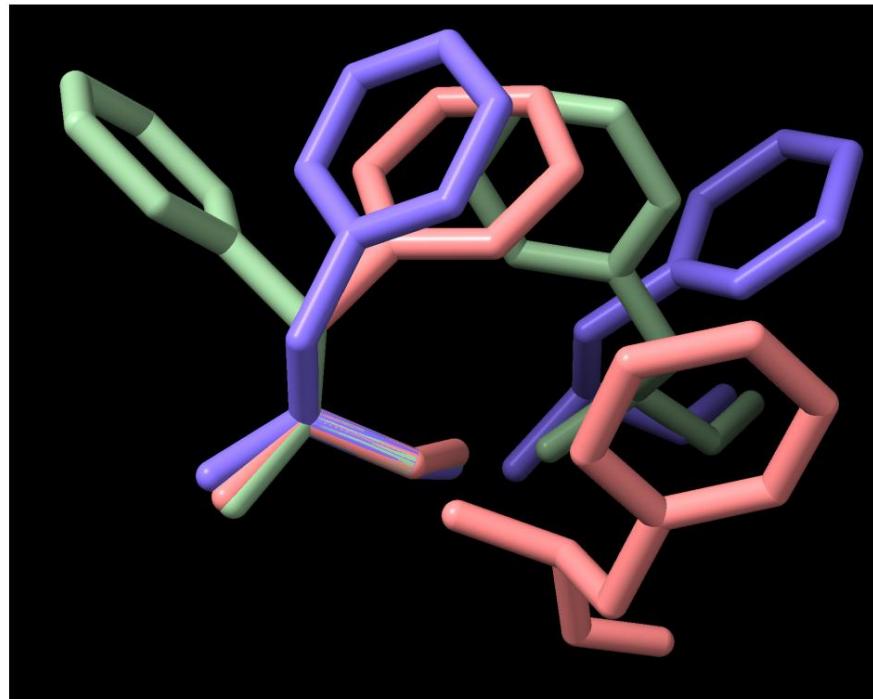
Doğrudur

Nəzərə alın ki, bu prosedur bütün məsafə matrisini deyil, hər dihedral bucağın dörd atomu arasında yalnız altı yerli məsafə dən istifadə edir.

14.6.4.8 Qalıqların və onların məsafələrinin üstü-üstə yığıılması

Daxili koordinatlar modulu həm burulma bucaqlarının hesablanması, həmdə strukturların yenidən qurulması üçün müxtəlif koordinat boşluqları arasında atom koordinatlarının çevrilmesi sinəsaslanır. Hər bir dihedronun birinci atomunu XZ məstəvisində, ikinci atomu başlanğıcda və üçüncü atomu +Z oxunda yerləşdirən koordinat fəza çevrilmişsi, həmçinin onu orijinal strukturdakı koordinatlara qaytaracaq müvafiq tərs çevrilmiş var.

Bu çevirmə matrisləri aşağıda göstərilidiyi kimi istifadə edilə bilər. İstinad dihedronunun düzgün seçilməsi ilə ikili və daha yüksək dərəcəli qarışılıqlı təsirləri bir çox zülal strukturları arasında araşdırıla və vizuallaşdırıla bilər, məsələn, Şəkil 14.3.



Şəkil 14.3: PDB faylında qonşu fenilalanin yanəncirləri 3PBL (insan dopamin D3 reseptoru)

Bu nümunə hər bir PHE qalığını N-C α -C β atomlarına zəncirdə yerləşdirir və zəncirdəki bütün PHE-ləri müvafiq koordinat məkanında sadə nümayiş kimi təqdim edir. Cütlü yanəncirliliqə laqənin daha real tədqiqi müvafiqədə biyyatda müzakirə edildiyi kimi, strukturların məlumat dəstini və qarışılıqlı laqə sınıfları üçün filtri araşdıracaq.

```
# bütün phe-phe cütlərini üstü-üstə qoyun - yalnız konsepsiyanı nümayiş etdirmək üçün sürətli hack edin # cüt
qarışılıqlı təsirlərini təhlil etmək üçün. PDB ATOM qeydlərini yaradır
```

```
# hə r bir PHE-i mə nşə yə yerlə şdirmə k və ə trafdakı bütün digə r PHE-lə ri göstə rmə k

## ə sas də yişə nlə r üçün stenoqram: cic =
myChain.internal_coord resNameNdx =
AtomKey.fields.resname aaNdx = cic.atomArrayIndex

## yalnız PHE atomlarını seçin:
pheAtomSelect = [aaNdx.get(k) in aaNdx.keys() üçün k.əkl [resNameNdx] == " F"] aaF = cic.atomArray[pheAtomSelect] # numpy
zə rif indekslə şdirmə KOPYA-nın görünmə mə sinə sə bə b olur

cic.ordered_aa_ic_list- də ric üçün :get_residues() funksiyasının # daxili_koords versiyası
if ric.lc == "F": # ə gə r PHE, chi1 dihedral üçün çevirmə matrislə ri alın
    chi1 = ric.pick_angle("chi1") # N:CA:CB:CG mə kanının mə nşə yində C-alfa var cst = np.transpose(chi1.cst) #
    transform TO chi1 space # rcst = np.transpose(chi1.rcst) # transform FROM chi1 space
    (burada lazımlı deyil) cic.atomSelect . myChain.get_residues ()-də yalnız res üçün PHE-lə ri çevirir : # yeni koordinat mə kanında
    PHE-lə ri çap edin

    ə gə r ["PHE"] -də res.resname :
        print(res.internal_coord.pdb_residue_string()) cic.atomArray[pheAtomSelect]
= aaF # surə tdə n koordinat boşluğununu bə rpa edin
```

14.6.4.9 3D çap zülal strukturları

OpenSCAD (<https://openscad.org>) möhkə m 3D CAD obyektlə ri yaratmaq üçün bir dildir. Daxili koordinatlardan zülal strukturunun qurulması alqoritmi OpenSCAD-də strukturu tə svir edə n verilə nlə rlə tə min edilir ki, 3D çap üçün uyğun model yaradılsın. Digə r program tə minati 3D çap üçün göstə rmə seçimi kimi STL mə lumatlarını yarada bilsə də (mə sə lə n, Chimera, <https://www.cgl.ucsf.edu/chimera/>), bu yanaşma çıxış kimi kürə lə ri və silindrə ri yaradır və bunu görə də 3D çap zülal strukturlarına uyğun də yişikliklə rə daha münasibdir. Fə rdi qalıqlar və bağlar OpenSCAD kodunda xüsusi istifadə üçün seçilə bilə r, mə sə lə n, ölçüyə görə vurğulamaq və ya xüsusi mövqelə rdə firlanan istiqrazlar ə lavə etmə k (bax: <https://www.thingiverse.com/thing:3957471> misal üçün).

```
# 3d çap myChain magistralına kürə lə rin və silindrə rin OpenSCAD programını yazın ## yalnız onurğa sütununu qə bul
etmə k üçün atom yükü filтрini tə yin edin: IC_Residue.accept_atoms =
IC_Residue.accept_backbone ## itkin qalıqları uzun bağlarla bağlamaq üçün çox
yükə k zə ncir kə silmə sini tə yin edin . Yuxarıda tə yin edilmiş atributlarla bütün atomları yenidə n oxumağa mə cbur
etmə k üçün mövcud mə lumatları silin:
myChain.internal_coord = Heç biri write_SCAD(myChain, "myChain.scad", miqyas=10.0)
```

14.6.4.10 daxili koordinatlara nə zarə t atributları

Daxili koordinatlar hesablanarkə n mə lumatları də yişdirmə k və ya süzmə k üçün daxili koordinatlar siniflə rində bir neçə nə zarə t atributları mövcuddur. Bunlar Cə dvə l 14.1-də verilmişdir :

14.6.5 Atom-atom kontaktlarının müə yyə n edilmə si

Qonşu axtarışını yerinə yetirmə k üçün NeighborSearch istifadə edin. Qonşu axtarışı C dilində yazılmış KD ağac modulundan istifadə etmə klə hə yata keçirilir (Bio.PDB.kdtrees modulunda KDTree sinfinə baxın və bu, onu çox sürə tli edir. O, hə məcən bir-birində n müə yyə n mə safə də olan bütün nöqtə cütlə rini tapmaq üçün sürə tli metodu ehtiva edir.

Sinif	Atribut	Defolt	Effekt
AtomKey d2h	False	Doğrudursa, D atomlarını H-yə çevirir	
yaratmaq üçün böyük etmək MaxPeptideBond			Zə ncir qırılmadan maksimum CN uzunluğu; IC Chain üzərində əlaqə
1.4 IC Qalıq atomları əsasında ncirlər qəbul edir, hidrogen atomları bəzi və ya bütün yan zəncirləri, H, D-ləri çıxarmaq üçün üstün tutur			
CYG, YCM, UNK adlarının da yışdırılmış sini qəbul edin HETATM-lərin emal edilməsi üçün 3 hərfli adlar, verilənlər bazası avəsasında Gly			
Cβ atomlarını yaratmaq üçün yalnız uglybeta False olmalıdır			

Cədvəl 14.1: Bio.PDB.daxili koordinatlarda idarəetmə atributları.

14.6.6 İki strukturun üst-üstə qoyulması

İki koordinat dəstini üst-üstə qoymaq üçün Superimposer obyektindən istifadə edin. Bu obyekt fırlanması hesablayır və iki atom siyahısını bir-birinin üstündə elə çevirən tərcümə matrisi, onların RMSD minimuma endirilir. Təbii ki, iki siyahıda eyni sayıda atom olmalıdır. Superimposer obyekti fırlanması/tərcüməni atomların siyahısına da tətbiq edə bilər. Fırlanması və tərcümənin kimi saxlanılır. Superimposer obyektiinin rotran atributunda tuple (qeyd edək ki, fırlanması düzgün çoxalır!). The RMSD rmsd atributunda saxlanılır.

Superimposer tərəfindən istifadə edilən alqoritm [15, Golub & Van Loan]-dan gəlir və təkəndən istifadə edildə yərə parçalanması (bu, ümumi Bio.SVDSuperimposer modulundan həyata keçirilir).

Misal:

```
>>> sup = Superimposer()
# Atom siyahılarını göstərin
# 'sabit' və 'hərəkətli' Atom obyektlərinin siyahısıdır
# Hərəkət edən atomlar sabit atomların üzərinə qoyulacaq
>>> sup.set_atoms(sabit, hərəkətli)
# Çap fırlanması/tərcümə /rmsd
>>> çap(sup.rotran)
>>> çap(sup.rms)
# Hərəkət edən atomlara fırlanması/tərcümə tətbiq edin
>>> sup.apply(hərəkətli)
```

Aktiv sahələrə əsaslanan iki strukturun üst-üstə düşməsi üçün aktiv sahə atomlarını hesablamak üçün istifadə edin fırlanması/tərcümə matrisləri (yuxarıdakı kimi) və bunları bütün molekula tətbiq edin.

14.6.7 Bir-biri ilə əlaqəli iki strukturun qalıqlarının xəritələşdirilməsi

Əvvəlcə FASTA formatında uyğunlaşdırma faylı yaradın, sonra StructureAlignment sınıfından istifadə edin. Bu sınıf bilər iki dən çox strukturu olan hizalamalar üçündən istifadə edilə bilər.

14.6.8 Yarım Sfera Ekspozisiyasının Hesablanması

Half Sphere Exposure (HSE) həllədici təsirə məruz qalmayan yeni, 2D ölçüsüdür [18]. Prinsipcə, rəqəmi hesablayır Ca atomlarının qalıq ətrafında onun yan zəncirinin istiqamətində və əks istiqamətdə (radius daxilində 13.9 Å). Sadəliyinə baxmayaraq, həllədinin təsirinə dair bir çox digər ölçüləri üstələyir.

HSE iki çeşiddə gəlir: HSE α və HSE β . Birincisi yalnız Ca atomu mövqelərini istifadə edir sonuncu Ca və C β atom mövqelərini istifadə edir. SƏTƏM ölçüsü HSEposure sınıfı ilə hesablanır əlaqə nömrəsinidən hesablaya bilər. Sonuncu sınıfda xəritəsinin verənlügətləri qaytaran üsullar var. Onun müvafiq HSE α , HSE β və əlaqə nömrəsi dəyişə bilər. qalıq obyekti.

Misal:

```
>>> model = struktur[0]
>>> hse = HSEposure()
```

Kod	İkinci də rə cə li quruluş
H	α-sarmal
B	Tə crid olunmuş β-körpü qalığı
E	Strand
G 3-10	sarmal Π-sarmal
I	
T	Dön
S	a yılmə k
-	Digə r

Cə dvə l 14.2: Bio.PDB-də DSSP kodları.

```
# HSEalpha hesablayın >>>
exp_ca = hse.calc_hs_exposure(model, seçim="CA3")
# HSEbeta-nı hesablayın >>>
exp_cb = hse.calc_hs_exposure(model, seçim="CB")
# Klassik koordinasiya nömrə sini hesablayın >>> exp_fs =
hse.calc_fs_exposure(model)
# Qalıq üçün HSEalpha çap edin >>> çap
(exp_ca[bə zi_qalıq])
```

14.6.9 İkinci də rə cə li strukturun müə yyə n edilmə si

Bu funksionallıq üçün siz DSSP quraşdırılmalısınız (və bunun üçün lisensiya ə ldə etmə lisiniz — akademik istifadə üçün pulsuzdur, bax <https://swift.cmbi.umcn.nl/gv/dssp/>). Sonra Qalıq obyektlə rini onların ikincil strukturuna (və ə Içatan sə th sahə sinə) uyğunlaşdırın DSSP sinifində istifadə edin. DSSP kodları Cə dvə l 14.2-də verilmişdir. Qeyd edə k ki, DSSP (program və nə ticə də sinif) birdə n çox modeli idarə edə bilmə z!

DSSP sinifi qalığın ə Içatan sə th sahə sini hesablamaq üçün də istifadə edilə bilə r. Lakin 14.6.10-cu bölmə yə də baxın.

14.6.10 Qalıq də rinliyinin hesablanması

Qalıq də rinliyi, qalıq atomlarının hə llədicinin ə Içatan sə thində n orta mə safə sidir. Bu, hə llədicinin ə Içatanlığının kifayə t qə də r yeni və çox güclü parametrlə şdirilmə sidir. Bu funksionallıq üçün siz Mişel Sannerin MSMS programını quraşdırılmalısınız (https://www.scripps.edu/sanner/html/msms_home.html). Sonra ResidueDepth sinifində n istifadə edin. Bu sinif, Qalıq obyektlə rini müvafiq (qalıq də rinliyi, Ca də rinliyi) çubuqlara uyğunlaşdırın lüğə t kimi davranışır. Ca də rinliyi qalığın Ca atomunun hə llədicinin ə Içatan sə thinə olan mə safə sidir.

Misal:

```
>>> model = struktur[0] >>> rd =
QalıqDə rinliyi(model, pdb_fayl) >>> qalıq_də rinliyi,
ca_də rinliyi = rd[bə zi_qalıq]
```

Siz hə mçinin sə th nöqtə lə ri ilə Rə qə msal Python massivi şə klində molekulyar sə thin özünə (get sə thi funksiyası vasitə silə) çıkış ə ldə edə bilə rsiniz.

14.7 PDB fayllarında ümumi problemlə r

Mə lümdür ki, bir çox PDB fayllarında semantik xə talar var (strukturların özlə ri deyil, onların PDB fayllarında tə msil olunması). Bio.PDB bunu iki yolla idarə etmə yə çalır. PDBParser obyekti iki şə kildə davrana bilə r: mə hədudlaşdırıcı üsul və defolt olan icaza verə n üsul.

Misal:

```
# İcazə verən təhlidici
>>> parser = PDBParser(PERMISSIVE=1) >>> parser =
PDBParser() # Eyni (defolt)
# Strict parser >>>
strict_parser = PDBParser(PERMISSIVE=0)
```

İcazə verilən və ziyyətdə (DEFAULT), açıq-aydın xətalari ehtiva edən PDB faylları "düzəlir" (yəni bəzi qalıqlar və ya atomlar buraxılır). Bu səhvlərə aşağıdakılardan daxildir:

- Eyni identifikatorla çoxsaylı qalıqlar
- Eyni identifikatora malik çoxsaylı atomlar (altloc identifikatoru nəzərə alınmaqla)

Bu səhvlər PDB faylındaki real problemləri göstərir (ətraflı məlumat üçün bax [16, Hamelryck və Manderick, 2003]).

Məhdudiyyətləri və ziyyətdə səhvləri olan PDB faylları istisnanın baş verməsinə səbəb olur. Bu, PDB fayllarında səhvləri tapmaq üçün faydalıdır.

Ancaq bəzi səhvlər avtomatik olaraq düzəldilir. Normalda hər bir nizamsız atomun boş olmayan altlok identifikatoru olmalıdır. Bununla belə, bu konvensiyaya əmək etməyi yənən və eyni atomun iki nizamsız mövqeyi üçün boş veya boş olmayan identifikatoru olan bir çox struktur var. Bu avtomatik olaraq düzgün şəkildə şəhərlənən olunur.

Bəzən struktur A zəncirinə aid qalıqların siyahısını ehtiva edir, ardınca B zəncirinə aid qalıqlar və yenidən A zəncirinə aid qalıqlar gəlir, yəni zəncirlər "qırılırlar". Bu da düzgün şəhərlənən olunur.

14.7.1 Nümunələr

PDBParser/Structure sınıfı təxminən 800 strukturda sınıqlanır keçirilib (hər biri unikal SCOP super ailəsinə aiddir). Bu, təxminən 20 dəqiqə və ya hər bir struktur üçün orta hesabla 1,5 saniyə çəkir. Təxminən 64000 atom ehtiva edən böyük ribosomal alt bölmənin (1FKK) strukturunun təhlili 1000 MHz kompüterdə 10 saniyə çəkir.

Birmənən məlumat strukturunun qurulmasının mümkün olmadığı hallarda üç istisna yaradıldı. Hər üç halda, ehtimal olunan səbəb PDB faylında düzəldilməli olan xətadir. Bu hallarda bir istisna yaratmaq, məlumat strukturunda strukturu səhvə svir etmək şansını işə salmaqdan daha yaxşıdır.

14.7.1.1 Dublikat qalıqlar

Bir struktur eyni ardıcılıqla identifikatoru (resseq 3) və ikoda malik bir zəncirdə iki amin turşusu qalığını ehtiva edir. Yoxlama zamanı məlumat oldu ki, bu zəncirdə Thr A3, ..., Gly A202, Leu A3, Glu A204. Aydındır ki, Leu A3 Leu A203 olmalıdır. 1FFK strukturu üçün bir neçə oxşar və ziyyət mövcuddur (məsələn, rəsibində Gly B64, Met B65, Glu B65, Thr B67, yəni qalıq Glu B65 Glu B66 olmalıdır).

14.7.1.2 Dublikat atomlar

1EJG strukturunda 22-ci mövqedə A zəncirində Ser/Pro nöqtəsi mutasiyası var. Öz növbəsində, Ser 22 bəzi nizamsız atomları ehtiva edir. Gözlənildiyi kimi, Ser 22-yə aid olan bütün atomlar boş olmayan altlok spesifikatoruna (B və ya C) malikdir. Pro 22-nin bütün atomlarında boş altlok olan N atomu istisna olmalıdır. A altlokları var. Bu, bir istisna yaradır, çünkü bir nöqtə mutasiyasında iki qalığa aid olan bütün atomlar boş olmayan altloklarına malik olmalıdır. Məlumat olub ki, bu atom, yəqin ki, Ser və Pro 22-tərəfindən paylaşılır, çünkü Ser 22 N atomunu əldən verir. Yenə də bu, fayldaki probleme işaret edir: N atomu həm Ser, həm də Pro qalıqlarında mövcud olmalıdır, hər iki halda uyğun altloc identifikatoru ilə əlaqələndirilir.

14.7.2 Avtomatik düzə lis

Bə zi sə hvlə r olduqca yaygındır və sə hv şə rh etmə k riski olmadan asanlıqla düzə ldilə bilə r. Bu hallar aşağıda verilmişdir.

14.7.2.1 Nizamsız atom üçün boş altlok

Normalda hə r bir nizamsız atomun boş olmayan altlok identifikatoru olmalıdır. Bununla belə , bu konvensiyaya ə mə l etmə yə n və eyni atomun iki nizamsız mövqeyi üçün boş və boş olmayan identifikatoru olan bir çox struktur var. Bu avtomatik olaraq düzgün şə kildə şə rh olunur.

14.7.2.2 Qırılmış zə ncirlə r

Bə zə n strukturda A zə ncirinə aid qalıqların siyahısı, ardınca B zə ncirinə aid qalıqlar və yenidə n A zə ncirinə aid qalıqlar gə lir, yə ni zə ncirlə r "qırılır". Bu düzgün şə rh olunur.

14.7.3 Fatal sə hvlə r

Bə zə n PDB faylı birmə nalı şə kildə şə rh edilə bilmə z. Sə hvi tə xmin etmə k və riskə atmaq ə və zinə , bir istisna yaradılır və istifadə çinin PDB faylini düzə ltmə si gözlə nilir. Bu hallar aşağıda verilmişdir.

14.7.3.1 Dublikat qalıqlar

Zə ncirdə ki bütün qalıqların unikal identifikatoru olmalıdır. Bu id aşağıdakılara ə sasə n yaradılıb:

- Ardıcılıq identifikatoru (resseq).
- Daxiletmə kodu (ikod).
- Hetfield sə tri ("W" sular üçün və "H" sonra digə r hetero qalıqlar üçün qalıq adı)
- Nöqtə mutasiyaları zamanı qalıqların qalıq adları (Qalıq obyektlə rini bir yerdə saxlamaq üçün DisorderedResidue obyekti).

Bu, unikal id-yə gə tirib çıxarmazsa, çox güman ki, nə sə sə hvdır və istisna yaranır.

14.7.3.2 Dublikat atomlar

Qalıqdakı bütün atomların unikal identifikatoru olmalıdır. Bu id aşağıdakılara ə sasə n yaradılıb:

- Atom adı (boşluqlar olmadan və ya problem yaranarsa boşluqlarla).
- Altloc tə yinedicisi.

Bu, unikal id-yə gə tirib çıxarmazsa, çox güman ki, nə sə sə hvdır və istisna yaranır.

14.8 Protein Mə lumat Bankına daxil olmaq

14.8.1 Protein Mə lumat Bankından strukturların yüklə nmə si

Strukturları PDB-də n (Protein Mə lumat Bankı) PDBList obyektində pdb faylini geri götür metodundan istifadə etmə klə endirmə k olar. Bu metodun arqumenti strukturun PDB identifikatorudur.

```
>>> pdbl = PDBList()
>>>
pdbl.retrieve_pdb_file("1FAT")
```

PDBList sinfi komanda xə tti alə ti kimi də istifadə edilə bilə r:

```
python PDBList.py 1fat
```

Yüklə nmiş fayl pdb1fat.ent adlanacaq və cari iş kataloqunda saxlanılacaq. Qeyd edə k ki, pdb faylini ə ldə etmə k metodunda yüklə nmiş PDB fayllarının saxlanacağı xüsusi qovluğu tə yin edə n pdır opsiya arqumenti də var.

Pdb faylini geri götürmə metodunda yüklə mə üçün istifadə edilə n sixılma formatını və yerli dekompressiya üçün istifadə olunan programı (defolt .Z formatı və gunzip) tə yin etmə k üçün bə zi seçimlə r də var. Bundan ə lavə , PDB ftp saytı PDBList obyektinin yaradılması ilə müə yyə n edilə bilə r. Varsayılan olaraq, Ümumdünya Protein Mə lumat Bankının serveri (<ftp://ftp.wwpdb.org/pub/pdb/data/structures/divided/pdb/>) istifadə olunur. Ətraflı mə lumat üçün API sə nə dlə rinə baxın. Bu modulu bağışladığı üçün Kristian Roterə bir daha tə şə kkür edirik.

14.8.2 Bütün PDB-nin yüklə nmə si

Aşağıdakı ə mrlə r bütün PDB fayllarını /data/pdb qovluğunda saxlayacaq:

```
python PDBList.py bütün /data/pdb
```

```
python PDBList.py bütün /data/pdb -d
```

Bunun üçün API metodu bütün pdb-ni yüklə mə k adlanır. -d seçiminin ə lavə edilmə si bütün faylları eyni qovluqda saxlayacaq. Əks halda, onlar PDB ID-lə rinə uyğun olaraq PDB tipli alt kataloqlara çeşidlə nirlə r. Trafikdə n asılı olaraq, tam yüklə mə 2-4 gün çə kə cə k.

14.8.3 PDB-nin yerli nüsxə sinini yeni saxlamaq

Bu, PDBList obyektində n istifadə etmə klə də edilə bilə r. Biri sadə cə PDBList obyekti yaradır (PDB-nin yerli nüsxə sinin mövcud olduğu qovluğu göstə rə rə k) və yenilə mə pdb metodunu çağırır:

```
>>> pl = PDBList(pdb="/data/pdb") >>>
pl.update_pdb()
```

Yerli nüsxə ni avtomatik olaraq yeni saxlamaq üçün ə lbə ttə ki, bundan hə ftə lik cronjob edə bilə rsiniz. PDB ftp saytı da müə yyə n edilə bilə r (API sə nə dlə rinə baxın).

PDBList-də istifadə oluna bilə cə k bə zi ə lavə üsullar var. Get all köhnə lmiş metodu bütün köhnə lmiş PDB girişlə rinin siyahısını ə ldə etmə k üçün istifadə edilə bilə r. Bu hə ftə də yişdirilmiş metod-cari hə ftə ə rzində ə lavə edilmiş, də yişdirilmiş və ya köhnə lmiş qeydlə ri ə ldə etmə k üçün istifadə edilə bilə r. PDBList imkanları haqqında ə traflı mə lumat üçün API sə nə dlə rinə baxın.

14.9 Ümumi suallar

14.9.1 Bio.PDB nə də rə cə də yaxşı sinaqdan keçirilib?

Çox yaxşı, ə slində .Bio.PDB PDB-də n 5500-ə yaxın strukturda geniş şə kildə sinaqdan keçirilmişdir - bütün strukturların düzgün tə hlil edildiyi görünür. Ətraflı mə lumatı Bio.PDB Bioinformatika mə qalə sində tapa bilə rsiniz.

Bio.PDB etibarlı vasitə kimi bir çox tə dqiqat layihə lə rində istifadə edilmişdir/istifadə olunur. Ə slində , mə n Bio.PDB-də n demə k olar ki, hə r gün tə dqiqat mə qsə dlə ri üçün istifadə edirə m və onu tə kmillə şdirmə k və yeni funksiyalar ə lavə etmə k üzə rində işlə mə yə davam edirə m.

14.9.2 Nə qə də r sürə tlidir?

PDBParser performansı tə xminə n 800 strukturda sinaqdan keçirilmişdir (hə r biri unikal SCOP super ailə sinə aiddir). Bu, tə xminə n 20 də qıqə və ya hə r bir struktur üçün orta hesabla 1,5 saniyə çə kir. Tə xminə n 64000 atom ehtiva edə n böyük ribosomal alt bölmə nin (1FKK) strukturunun tə hlili 1000 MHz kompüterdə 10 saniyə çə kir.

Qisacısı: bir çox program üçün kifayə t qə də r sürə tlidir.

14.9.3 Molekulyar qrafika də stə yi varmı?

Birbaşa deyil, ə sasə n artıq bir neçə Python ə sası/Python mə lumatlı hə llə r olduğundan Bio.PDB ilə potensial olaraq istifadə edilə bilə r. Mə nim seçimim Pymol, BTW (mə n bunu Bio.PDB ilə uğurla istifadə etmişə m və yə qin ki, tezliklə / bir gün Bio.PDB-də xüsusi PyMol modulları olacaq). Python ə sası/xə bə rrar molekulyar qrafik hə llə ri daxildir:

- PyMol: <https://pymol.org/>
- Kimera: <https://www.cgl.ucsf.edu/chimera/>
- PMV: <http://www.scripps.edu/~sanner/python/>
- Coot: <https://www2.mrc-lmb.cam.ac.uk/personal/pemsley/coot/>
- CCP4mg: <http://www ccp4.ac.uk/MG/>
- mmLib: <http://pymmlib.sourceforge.net/>
- VMD: <https://www.ks.uiuc.edu/Research/vmd/>
- MMTK: <http://dirac.cnrs-orleans.fr/MMTK/>

14.9.4 Bio.PDB-də n kim istifadə edir?

Bio.PDB proteinlə rdə nizamsız bölgə lə ri proqnozlaşdırın DISEMBL veb serverinin tıkıntısında istifadə edilmişdir (<http://dis.embl.de/>). Bio.PDB hə mçinin PDB-də zülal strukturları arasında aktiv sahə lə r oxşarlıqlarının geniş miqyaslı axtarışını hə yata keçirmə k üçün [17, Hamelryck, 2003] və xə tti ikincil struktur elementlə rini müə yyə n edə n yeni alqoritm hazırlamaq üçün istifadə edilmişdir [31, Majumdar et al., 2005].

Xüsusiyyə tlə r və mə lumat sorğularına ə sasə n, Bio.PDB bir neçə LPC tə rə fində n də istifadə olunur (Böyük Pharmaceutical şirkə tlə ri :-).

Fəsil 15

Bio.PopGen: Populyasiya genetikası

Bio.PopGen populyasiya genetikasını dəstəkləyən Biopython moduludur, Biopython 1.44-dən sonra mövcuddur. Modulun məqsədi geniş istifadə olunan məlumat formatlarını, programları və verilənlər bazalarını dəstəkləməkdir.

15.1 GenePop

GenePop (<http://genepop.curtin.edu.au/>) Hardy-Weinberg testlərinin, əlaqənin tarazlığını, əhalinin diferensiasiyasını, əsas statistikani, Fst və miqrasiya təxminlərinin və başqalarını dəstəkləyən məşhur populyasiya genetik program paketidir. GenePop ardıcılıqlı məlumatlarını idarə etmədiyi üçün ardıcılıqla əsaslanan statistika təqdim etmir. GenePop fayl formatı geniş spektrli digər populyasiya genetik program programları tərəfindən dəstəklənir və beləliklə onu populyasiya genetikası sahəsində müvafiq formata çevirir.

Bio.PopGen GenePop fayl formatının təhlilçisi və generatoru təqdim edir. Qeydin məzmununu manipulyasiya etmək üçün kommunal programlar da təmin edilir. GenePop faylinin oxunmasına dair bir nümunə (məsələn, GenePop məlumat fayllarını Biopython-un Test/PopGen kataloqunda tapa bilərsiniz):

Bio.PopGen -dən GenePop idxali

```
Dəstəkimi open("example.gen") ilə : rec =
    GenePop.read(handle)
```

Bu, example.gen adlı faylı oxuyacaq və onu təhlil edəcək. Rec çap etsəniz, yazı yenidən GenePop formatında çıxacaq.

Rec-də ən vacib məlumat yer adları və əhali məlumatı olacaq (lakin daha çox şey var – API sənədlərini yoxlamaq üçün yardımından (GenePop.Record) istifadəedin). Loci adları rec.loci siyahısında tapılı bilər. Əhali haqqında məlumatı rec.populations saytından əldə etmək olar. Əhalilər hər bir əhali üçün bir elementdən ibarət siyahıdır. Hər bir elementin özü fərdlərin siyahısıdır, hər bir fərd fərdi ad və allellərin siyahısından ibarət cütdür (hər markerə 2), burada rec.populations üçün bir nümunə var:

```
[
    [
        ("Ind1", [(1, 2), (3, 3), (200, 201)]),
        ("Ind2", [(2, Yoxdur), (3, 3), (Heç biri, Yoxdur)]),
    ],
    [
        ("Digər1", [(1, 1), (4, 3), (200, 200)]),
    ],
]
```

Bələ liklə , bizim iki populyasiyamız var, birincidə iki fə rd, ikincidə yalnız bir. İlk populyasiyanın ilk fə rdinə Ind1 deyilir, 3 lokusun hə r bürün allelik mə lumatlar aşağıdakılardır. Nə zə rə alın ki, hə r hansı bir yer üçün mə lumat çatışa bilə r (nümunə olaraq yuxarıda Ind2-yə baxın).

GenePop qeydlə rini manipulyasiya etmə k üçün bir neçə communal funksiya mövcuddur, burada bir nümunə :

[Bio.PopGen -də n GenePop idxalı](#)

Tə sə vvür edin ki, yuxarıdakı kod parçasına uyğun olaraq rec yükə misiniz...

```
rec.remove_population(pos)
```

Populyasiyanı qeyddə n silir, pos # rec.populations-də ə hali mövqeyidir, onun 0 mövqeyində n başlığıını unutmayın. # rec da yişdirilir.

```
rec.remove_locus_by_position(pos)
```

Lokusu mövqeyinə görə silir, pos # rec.loci_list-də ki yer mövqeyidir, onun 0 mövqeyində n başlığıını unutmayın. # rec də yişdirilir.

```
rec.remove_locus_by_name(ad)
```

Lokus adı ilə silir, ad # rec.loci_list-də olduğu kimi yer adıdır. Ə gə r ad mövcud deyilsə , funksiya # sə ssizə ugursuz olur. # qeyd də yişdirildi.

```
rec_loci = rec.split_in_loci()
```

Lokuslarda rekordu ayırrı, yə ni hə r bir lokus üçün tə k lokus və bütün populyasiyalarla yeni # rekord yaradır.

Nə ticə hə r bir ə sas yerin adı olmaqla lügə tdə qaytarılır.

Də ya r GenePop rekordudur. # qeyd də yişdirilmə yib.

```
rec_pops = rec.split_in_pops (pop_adları)
```

Populyasiyalar üzrə rekordu ayırrı, yə ni hə r populyasiya üçün tə k populyasiya və bütün lokuslarla # yeni rekord yaradır.

Nə ticə hə r bir açar # ə halinin adı olmaqla lügə tdə qaytarılır. Populyasiya adları GenePop-da mövcud olmadığı üçün # onlar massivdə (pop_adları) ötürülür.

Hə r lügə t girişinin də ya ri GenePop qeydiidir. # qeyd də yişdirilmə yib.

GenePop populyasiya adlarını də stə klə mir, bu mə hdudiyyə t bə zə n çə tin ola bilə r. Hal-hazırda Biopython üçün populyasiya adlarını aktivlə şdirmə k üçün funksionallıq planlaşdırılır. Bu uzantılar standart formatla heç bir şə kildə uyğunluğu pozmayacaq. Ortamüddə tli perspektivdə biz GenePop vəb xidmə tini də də stə klə mə k istə rdik.

Fə sil 16

Bio.Phylo ilə Filogenetika

Bio.Phylo modulu Biopython 1.54-də tə qdim edilmişdir. SeqIO və AlignIO-nun rəhbərliyinin ardınca o, mənbə məlumat formatından asılı olmayaraq filogenetik ağaclarla işləmək üçün ümumi üsul, həmçinin I/O əməliyyatları üçün ardıcıl API təmin etmək məqsədi daşıyır.

Bio.Phylo açıq girişli jurnal məqalə sindən təsvir edilmişdir [45, Talevich et al., 2012], siz də faydalı tapa bilərsiniz.

16.1 Demo: Ağacda nə var?

Modulla tanış olmaq üçün artıq qurduğumuz ağacdan başlayaq və onu bir neçə fərqli yolla yoxlayaqla. Sonra xüsusi bir phyloXML xüsusiyyətindən istifadə etmək üçün budaqları rəngləndirəcəyi və nəhayət onu saxlayacağıq.

Sevimli mətn redaktorunuzdan istifadə edərək simple.dnd adlı sadə Newick faylı yaradın və ya [simple.dnd](#) istifadə edin Biopython mənbə kodu ilə təmin edilmişdir:

```
((A,B),(C,D)),(E,F,G));
```

Bu ağacın budaq uzunluğu yoxdur, yalnız topologiya və etiketlənmüş terminallar var. (Əgər real ağac fayliniz varsa, bununla və zinət bu demonu izləyə bilərsiniz.)

Seçdiyiniz Python tərcümə çisinini işə salın:

```
$ ipython -pylab
```

İnteraktiv iş üçün IPython tərcümə çisinin -pylab bayrağı ilə işə salınması matplotlib integrasiyasını təmin edir, beləliklə, qrafiklər avtomatik olaraq açılır. Bu demo zamanı bundan istifadə edəcəyi.

İndi Python daxilində, faylin adını və formatının adını verərək ağac faylini oxuyun.

```
>>> Bio import Phylo-dan
>>> ağaç
= Phylo.read ("simple.dnd", "newick")
```

Ağac obyektini sətir kimi çap etmək bizə bütün obyekti yierarxiyasına nəzər salmağa imkan verir.

```
>>> çap (ağaç)
Ağac(köklü=Yanlış, çəki=1.0)
Clade()
Clade()
Clade()
Clade(ad='A')
Clade(ad='B')
Clade()
```

```

    Clade(ad='C')
    Clade(ad='D')
Clade()
    Clade(ad='E')
    Clade(ad='F')
    Clade(ad='G')

```

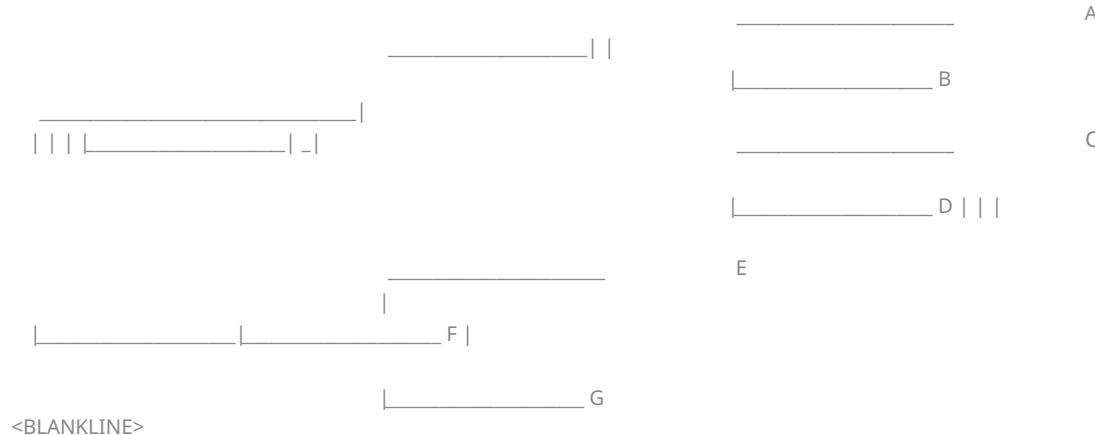
Ağac obyekti ağac haqqında qlobal mə lumatı ehtiva edir, mə sə lə n, onun köklü və ya köksüz olması. Bu bir kök tə bə qə si var və bunun altında ipuçlarına qə də r yuva siyahıları var.

`Draw_ascii` funksiyası sadə ASCII-art (düz mə tn) dendrogramı yaradır. Bu ə lverişlidir daha yaxşı qrafik alə tlə r olmadıqdə interaktiv kə şfiyyat üçün vizuallaşdırma.

```

>>> Bio import Phylo-dan >>> ağac
= Phylo.read ("simple.dnd", "newick")
>>> Phylo.draw_ascii(ağac)

```



Ə gə r sizdə matplotlib və ya pylab quraşdırılırsa, siz draw funksiyasından istifadə edə rə k qrafik ağac yarada bilə rsiniz.

```
>>> tree.rooted = Doğrudur
```

```
>>> Phylo.draw(ağac)
```

Şəkil 16.1-ə baxın.

16.1.1 Ağacın içə risində budaqların rə nglə nmə si

Çə kmə funksiyası ağacda müxtə lif rə nglə rin və budaq enlə rinin göstə rilmə sinini də stə klə yir. Biopython 1.59-dan etibarə n rə ng və genişlik atributları ə sas Clade obyektində mövcuddur və onlardan istifadə etmə k üçün ə lavə heç nə tə lə b olunmur. Hə r iki atribut verilmiş tə bə qə yə rə hbə rlik edə n filiala istinad edir və rekursiv şə kildə tə tbiq olunur, belə liklə , bütün nə sil budaqlar da ekran zamanı tə yin olmuş en və rə ng də yə rlə rini miras alacaq.

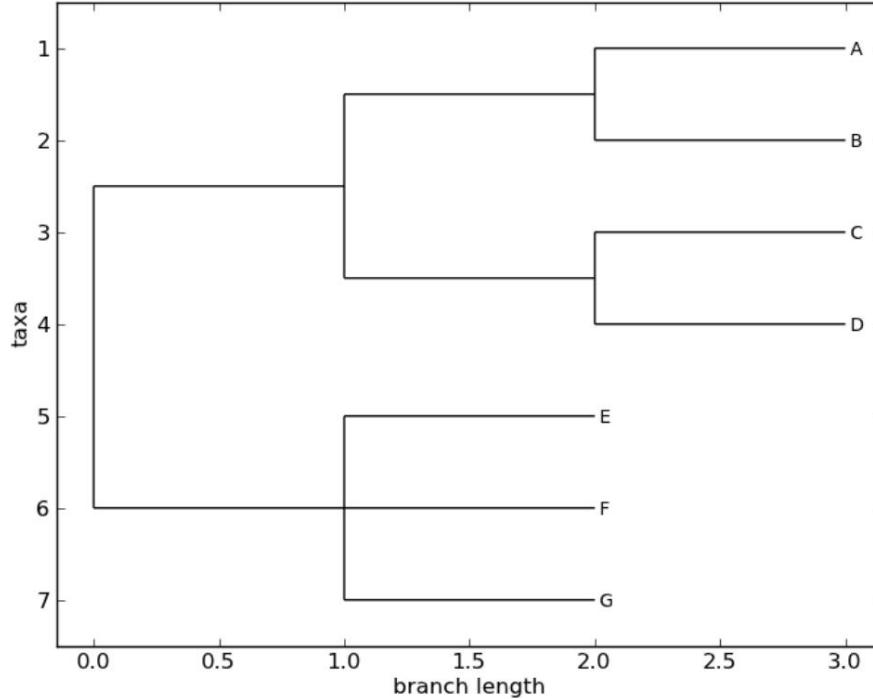
Biopython-un ə vvə lki versiyalarında bunlar PhyloXML ağacılarının xüsusi xüsusiyyə tlə ri idи və attributlardan istifadə etmə k üçün ə vvə lcə ağacı Bio.Phylo.PhyloXML modulundan Phylogeny adlı ə sas ağac obyektiinin alt sinfinə çevirmə k lazımdır.

Biopython 1.55 və sonrakı versiyalarında bu, ə lverişli ağac üsuludur:

```
>>> ağac = tree.as_phyloxml()
```

Biopython 1.54-də bir ə lavə idxalla eyni şeyi edə bilə rsiniz:

```
>>> Bio.Phylo.PhyloXML -də n idxal Filogeniya >>> ağac =
Phylogeny.from_tree(ağac)
```



Şəkil 16.1: Phylo.draw ilə çəkilmiş köklü ağac.

Qeyd edək ki, Newick və Nexus fayl formatları budaq rənglərinin və ya genişliklərinin dəstəkləmir, ona görə də bu atributları Bio.Phylo da istifadə etsəniz, siz də yərləri yalnız PhyloXML formatında saxlaya biləcəksiniz. (Siz hələ də ağacı Newick və ya Nexus kimi saxlaya bilərsiniz, lakin rəng və genişlik də yərləri çıkış faylında atlanacaq.)

İndi rəngləri təyin etməyi başlaya bilərik. Əvvəlcə kök təbəqəsini boz rəngə boyayacaqıq. Biz bunu 24 bitlik rəng dəyişirini RGB üçlü, HTML tipli hex sətir və ya avvalı təyin edilmiş rənglərdən birinin adı kimi təyin etməklə edə bilərik.

```
>>> tree.root.color = (128, 128, 128)
```

Və ya:

```
>>> tree.root.color = "#808080"
```

Və ya:

```
>>> tree.root.color = "boz"
```

Bir təbəqə üçün rənglər bütün təbəqə boyunca aşağıya doğru sıralanır, buna görə də burada kök rəngləndirdikdə, bütün ağacı boz rəngə çevirir. Ağacın aşağısına fərqli bir rəng təyin etməklə bunu aradan qaldırıbilərik.

"E" və "F" adlı qovşaqlarınə nəson ortaqəcdadını (MRCA) hədəfləyək. Ortaqəcdad metodunu orijinal ağacda həmin təbəqə yə istinadlı qaytarır, buna görə də biz həmin "somon" baliqini rəngləndirdikdə rəng orijinal ağacda görünəcək.

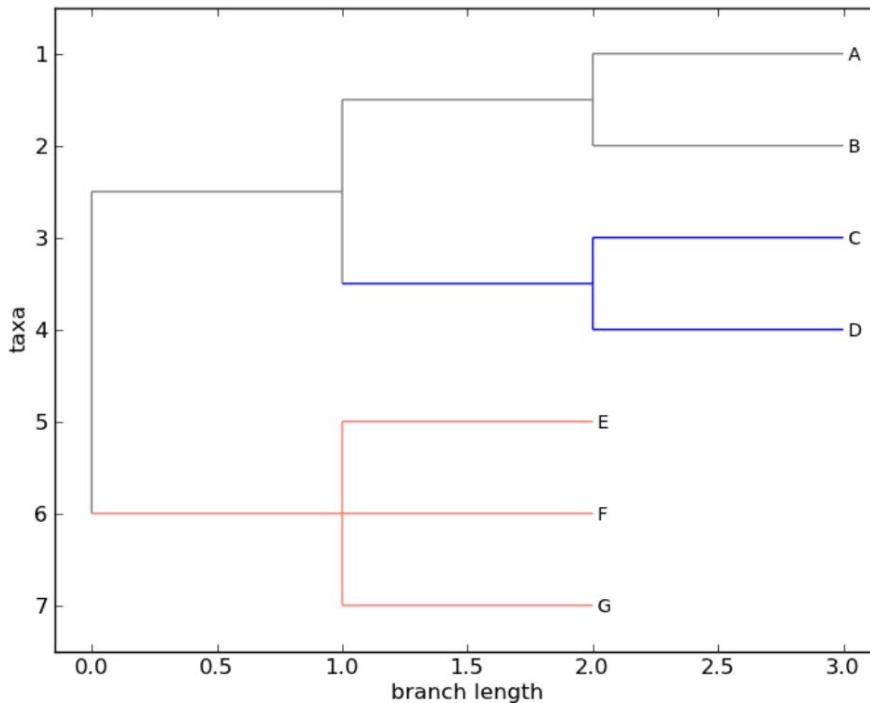
```
>>> mrca = tree.common_ancestor({"name": "E"}, {"name": "F"}) >>> mrca.color = "somon"
```

Müəyyən bir təbəqənin ağaçının harada olduğunu dəqiqlişək, iç içəsi siyahı girişləri baxımından, onu indeksləşdirməklə birbaşa ağacdakı həmin mövqeyə keçəbilərik. Burada [0,1] indeksi kökün birinci uşağıının ikinci uşağına aididir.

```
>>> tree.clade[0, 1].color = "mavi"
```

Nə həyə t, işimizi göstərin:

```
>>> Phylo.draw(ağac)
```



Şəkil 16.2: Phylo.draw ilə çəkilmiş rəngli ağac.

Şəkil 16.2-ə baxın.

Nə zərə alın ki, bir təbəqənin rənginə həmin təbəqə yə aparan budaq, eləcə də onun tövəmələri daxildir. E və F-nin ortaqlıcdadı kökün yalnız altında olduğu ortaya çıxır və bu rənglər mənilə Ağacın kökünün harada olduğunu dəqiqliyə bilərik.

Canım, biz çox şeyə nail olduq! Gəlin burada dincə ləkəyi işimizi xilas edək. Faylı adı və ya sapı ilə yazma funksiyasına zəng edin - burada nə yazılacağını görmək üçün standart çıkışdan istifadə etdirik - və phyloxml formatı. PhyloXML təyin etdiyimiz rəngləri saxlayır, beləliklə siz bu phyloXML faylini Archaeopteryx kimi başqa ağac görüntülər yicisində aça bilərsiniz və rənglər orada da görünəcək.

```
>>> import sys >>> n
= Phylo.write(ağac, sys.stdout, "phyloxml") # doctest:+ELLIPSIS <phyloxml ...> <filogeniya köklü="true">
```

```
<clade>
<rəng>
<red>128</red>
<green>128</green>
<blue>128</blue> </
color>
<clade>
<clade>
```

<clade>

<name>A</name> </

clade>

<clade>

<name>B</name> </

clade> </

clade>

<clade>

<r̄ ng>

<red>0</red>

<green>0</green>

<blue>255</blue> </

color>

<clade>

<name>C</name> </

clade>

...

...

</clade> </

filogenez> </phyloxml>

>>> n

1

Bu fə slin qalan hissə si Bio.Phylo-nun ə sas funksionallığını daha ə tərflı ə hatə edir. Bio.Phylo istifadə sinə dair daha çox nümunə üçün Biopython.org saytındaki yemək kitabına baxın: <http://biopython.org/>

[wiki/Phylo_cookbook](#)

16.2 I/O funksiyaları

SeqIO və AlignIO kimi, Phylo da dörd funksiya vasitə silə fayl girişini və çıxışını idarə edir: tə hlil etmə k, oxumaq, yazmaq və çevirmə k, bunların hamısı Newick, NEXUS, phyloXML və NeXML ağac fayl formatlarını, elə cə də Müqayisə li Mə lumat Tə hlili Ontologiyasını (CDAO) də stə klə yir.

Oxu funksiyası verilmiş faylda bir ağacı tə hlil edir və onu qaytarır. Diqqə tli; olarsa, xə ta qaldıracaq faylda birdən çox ağac var və ya ağac yoxdur.

```
>>> Bio import Phylo >>> tree =
Phylo.read("Tests/Nexus/int_node_labels.nwk", "newick") >>> print(ağac) # doctest:+ELLIPSIS
Tree(rooted=False, weight=1.0)
```

```
Clade(budaq_uzunluğu=75.0, adı='gymnosperm')
    Clade(budaq_uzunluğu=25.0, adı='İyna yarpaqlar')
        Clade(budaq_uzunluğu=25.0)
            Clade(branch_length=10.0, name='Vergi+nonSci')
                Clade(budaq_uzunluğu=90.0, adı='Taxaceae')
                    Clade(branch_length=125.0, name='Cephalotaxus')
                    ...

```

(Nümunə faylları Biopython paylanması Test/Nexus/ və Tests/PhyloXML/ kataloqlarında mövcuddur.)

Çoxsaylı (yaxud namə ləm sayda) ağacları idarə etmə k üçün hə r birini tə krarlayan tə hlil funksiyasından istifadə edin verilmiş fayldakı ağaclar:

```
>>> ağaclar = Phylo.parse("Testlə r/PhyloXML/phyloxml_examples.xml", "phyloxml") >>> ağaclardakı ağac üçün :
...
    çap(ağac) # doctest:+ELLIPSIS
...
Phylogeny(tə svir='phyloXML ya "branch_length" atributundan istifadə etmə yə imkan verir...', name='nümunə
Clade())
    Clade(budaq_uzunluğu=0,06)
        Clade(budaq_uzunluğu=0,102, ad='A')
...

```

Yazma funksiyası ilə bir ağac və ya tə krarlanan ağacları fayla yazın:

```
>>> ağaclar = Phylo.parse("Testlə r/PhyloXML/phyloxml_examples.xml", "phyloxml") >>> ağac1 = növbə ti(ağaclar)

>>> Phylo.write(tree1, "tree1.nwk", "newick") 1

>>> Phylo.write(ağaclar, "digə r_ağaclar.xml", "phyloxml") # qalan ağacları yazın 12
```

Dönüştürmə funksiyası ilə faylları də stə klə nə n formatlardan hə r hansı biri arasında çevirin:

```
>>> Phylo.convert("tree1.nwk", "newick", "tree1.xml", "nexml")
1
>>> Phylo.convert("digə r_ağaclar.xml", "filoxml", "digə r_ağaclar.nex", "nexus")
12
```

Sə tirlə ri faktiki fayllar a və zinə giriş və ya çıkış kimi istifadə etmə k üçün, SeqIO və ilə işlə diyiniz kimi StringIO-dan istifadə edin. AlignIO:

```
>>> Bio import Phylo >>> from io
import StringIO >>> handle =
StringIO("(((A,B),(C,D)),(E,F,G);") >>> ağac = Phylo.read(handle, "newick")
```

16.3 Ağaclarla baxmaq və ixrac etmə k

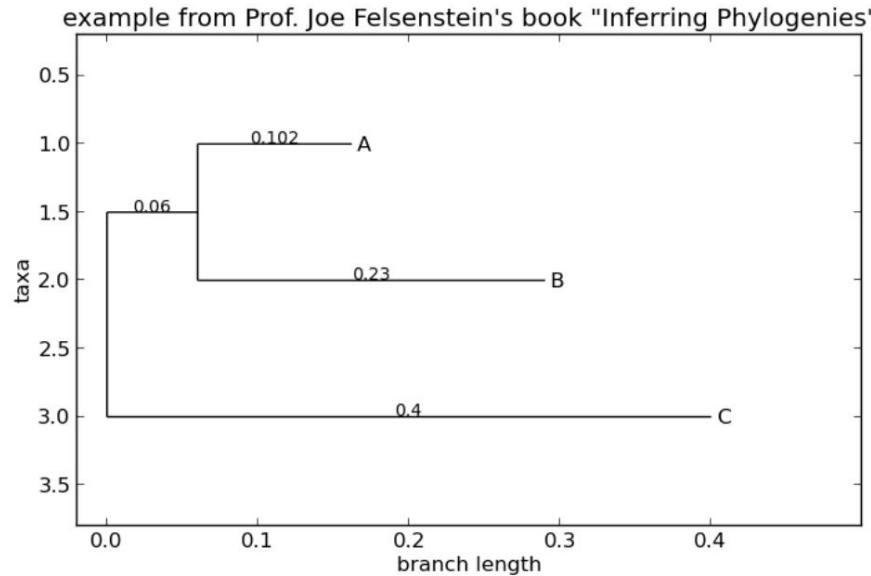
Ağac obyektinin ümumi görünüşünü ə ldə etmə yin ə n sadə yolu onu çap etmə kdir:

```
>>> Bio import Phylo >>> ağac =
Phylo.read("PhyloXML/example.xml", "phyloxml") >>> çap(ağac)
```

```
Phylogeny(tə svir='phyloXML ya "branch_length" atributundan istifadə etmə yə imkan verir...', name='nümunə
Clade())
    Clade(budaq_uzunluğu=0,06)
        Clade(budaq_uzunluğu=0,102, ad='A')
            Clade(budaq_uzunluğu=0,23, ad='B')
                Clade(budaq_uzunluğu=0,4, ad='C')
```

Bu, bipython-un ağacı tə msil etmə k üçün istifadə etdiyi obyekt iyerarxiyasının konturudur. Amma daha çox ehtimal ağacın rə smini görmə k istə rdiniz. Bunu etmə k üçün üç funksiya var.

Nümayişdə gördükümüz kimi, draw_ascii ağacın ascii-art rə smini (köklü filogram) standart çıxışa və ya ə gə r verilmişdir, , açıq fayl sapını çap edir. Ağac haqqında mövcud olan mə lumatların hamısı göstə rilmir, lakin o, heç bir xarici asılılığa etibar etmə də n ağaca tez baxmaq üçün bir yol tə qdim edir.



Şəkil 16.3: Çəkmə funksiyası ilə çəkilmiş sadə köklü ağac.

```
>>> ağaç = Phylo.read("PhyloXML/example.xml", "phyloxml")
>>> Phylo.draw_ascii(ağaç)
```

```
_____
| |
|_____ B |
|
|_____ C
<BLANKLINE>
```

Draw funksiyası matplotlib kitabxanasından istifadə edərək daha çəlbedici şəkil çəkir. API sənədlərinə baxın çıxışı fərdiləşdirmək üçün qəbul etdiyi arqumentlər haqqında təfərruatlar üçün.

```
>>> Phylo.draw(ağaç, budaq_etiketləri=lambda c: c.budaq_uzunluğu)
```

Məsələn, Şəkil 16.3-ə baxın.

Biopython wiki-də Phylo sənədini baxın (<http://biopython.org/wiki/Phylo>) təsvirlər üçün və draw_ascii, draw_graphviz və to_networkx-də daha təkmil funksionallığı nümunələri.

16.4 Tree və Clade obyektlərinə istifadə

Təhlil və oxunma ilə istehsal olunan Ağac obyektləri kök atributunda Ağac obyektinə əlavə edilmiş rekursiv alt ağaclar üçün konteynerlərdir (filogenetik ağacın əsliндə köklü hesab edilib-edilmə məsindən asılı olmayıaraq). Ağacda köklülük və tək Clade istinadı kimi filogeniya üçün qlobal olaraq tətbiq olunan məlumat var; bir Clade budaq uzunluğu kimi qovşaq və klade-xüsusi məlumatlara və clades atributuna əlavə edilmiş öz nəsil Clade nümunələrinin siyahısına malikdir.

Bələliklə, ağac və ağac arasında fərqli var. kök. Təcrlübədə, buna baxmayaraq, nadir hallarda bu barədə narahat olmaq lazımdır. Fərqi aradan qaldırmaq üçün həm Tree, həm də Clade, ağacı və ya onun hər hansı bir təbəqəsinə axtarmaq, yoxlamaq və ya dəyişdirmək üçün ümumi istifadə olunan metodların tətbiqlərinin ehtiva edən TreeMixin-dən miras alır. Bu o deməkdir ki, ağacın dəstəklədiyi metodların demək olar ki, hamısı tree.root-da və onun altındakı istenilən növdədə mövcuddur. (Clade həmçinin kök xassəsinə malikdir və bu, clade obyektinin özünü qaytarır.)

16.4.1 Axtarış və keçid üsulları

Rahatlıq üçün bütün xarici və ya daxili qovşaqları birbaşa siyahı kimi qaytaran bir neçə sadə lə şdirilmiş üsul tə qdim edirik:

get terminals bu ağacın bütün terminal (yarpaq) qovşaqlarının siyahısını tə rtib edir.

get nonterminals bu ağacın bütün qeyri-terminal (daxili) qovşaqlarının siyahısını tə rtib edir.

Bunların hə r ikisi ağacın kecidinə tam nə zarə t edə n metodu ə hata edir, find_clades. Daha iki kecid metodu, find_elements və find_any, eyni ə sas funksionallığı güvə nir və eyni arqumentlə ri qə bul edir, daha yaxşı tə svirin olmaması üçün "hə də f spesifikasiyası" adlandıracıq. Bunlar ağacdakı hansı obyektlə rin uyğunlaşdırılacağını və iterasiya zamanı qaytarılacağını müə yyə nlə şdirir. Birinci arqument aşağıdakı növlə rdə n hə r hansı biri ola bilə r:

- TreeElement nümunə si, hansı ağac elementlə ri şə xsiyyə tə uyğun gə lir – belə liklə , Clade ilə axtarış hə də f kimi mə sə lə n, ağacda hə min tə bə qə ni tapacaq;
- Ağac elementlə rinin sə tir tə qdimatına uyğun gə lə n sə tir — xüsusə n də sinfin adı (ə lavə olunur) Biopython 1.56-da);
- Eyni tipli (və ya alt tipli) hə r bir ağac elementinin uyğunlaşdırılacağı sinif və ya tip;
- Açırların ağac elementinin atributları olduğu və də yə rlə rin hə r bir ağac elementinin müvafiq atributuna uyğunlaşdırıldığı lügə t. Bu daha da tə fə rrüatlı olur:
 - Ə gə r int verilirsə , o, ə də di olaraq bə rabə r atributlara uyğun gə lir, mə sə lə n, 1 1 və ya 1.0-a uyğun gə lir – Ə gə r mantiq verilirsə (Doğru və ya Yanlış), müvafiq atribut də yə ri boolean kimi qiymə tlə ndirilir. və eyni şeyi yoxladı
 - Heç biri uyğun gə lmir
 - Ə gə r sə tir verilirsə , də yə r normal ifadə kimi qə bul edilir (bu, sadə cə prefiks deyil, müvafiq element atributunda bütün sə tirlə uyğun olmalıdır). Xüsusi regex simvolları olmayan verilmiş sə tir sə tir atributlarına tam uyğun olacaq, ona görə də regexlə rdə n istifadə etmirsinizsə , bu barə də narahat olmayın.
 - Mə sə lə n, Foo1, Foo2 və Foo3 adları olan ağacdakı tree.find_clades({"name": "Foo1"}) Foo1-ə uyğun gə lir, {"name": "Foo.*"} hə r üç tə bə qə yə uyğun gə lir və {"name": "Foo"} heç nə ilə uyğun gə lmir.

Üzə n nöqtə li arifmetika bə zi qə ribə davranışlar yarada bildiyinə görə , biz birbaşa uyğun gə lə n üzmə lə ri də stə klə mirik. Bunun ə və zinə , göstərilə n atributda sıfırdan fə rqli də yə rə malik hə r bir elementi uyğunlaşdırmaq üçün boolean True istifadə edin, sonra hə min atributda bə rabə rsizliklə (və ya tə hlükə li yaşamağı sevirsinizsə , də qıq rə qə mla) ə l ilə filtrlə yin.

Lügə tda bir neçə giriş varsa, uyğun element verilmiş atribut də yə rlə rinin hə r birinə uyğun olmalıdır — "və ya" deyil, "və " hesab edin.

- True və ya False qaytaran tə k arqument götürə n funksiya (ağacdakı hə r bir elementə tə tbiq olunacaq). Rahatlıq üçün, LookupError, AttributeError və ValueError susdurulur, buna görə də bu, ağacda üzə n nöqtə də yə rlə ri və ya daha mürə kkə b xarakteristikani axtarmaq üçün başqa tə hlükə siz yol tə qdim edir.

Hə də fdə n sonra iki istege bağlı açar söz arqumenti var:

terminal — Terminal tə bə qə lə ri (aka yarpaq qovşaqları) üçün və ya ona qarşı seçmə k üçün mantiqi də yə r: Yalnız terminal tə bə qə lə ri üçün True axtarışları, qeyri-terminal (daxili) tə bə qə lə r üçün False və standart, None, hə m terminal, hə m də qeyri-terminal tə bə qə lə ri, hə məcəninin is_terminal metodu olmayan hə r hansı ağac elementlə rini axtarır.

order — Ağacın kecmə qaydası: "öncə də n sıfırış" (defolt) də rənlik-ilk axtarışdır, "postorder" alt ilə DFS-dir valideynlə rdə n ə vvə İki qovşaqlar və "sə viyyə " genişlik-ilk axtarışdır.

Nə hayə t, üsullar ixtiyari açar söz arqumentlə rini qə bul edir ki, bunlara dictionary hə də f spesifikasiyası kimi yanaşılır: düymə lə r axtarılacaq element atributunun adını gösta rir və arqument də ya ri (sə tir, tam, Yox və ya mantiq) tapılan hə r bir atributun da yə ri ilə müqayisə edilir. Heç bir açar söz arqumenti verilmirsə , istə nilə n TreeElement növlə ri uyğunlaşdırılır. Bunun üçün kod ümumiyyə tlə lügə ti hə də f spesifikasiyası kimi ötürmə kdə n daha qısadır: tree.find_clades({"name": "Foo1"}) tree.find_clades(name="Foo1") kimi qısaltıldı bilə r.

(Biopython 1.56 və ya daha sonrakı versiyalarda bu daha da qısa ola bilə r: tree.find_clades("Foo1"))

İndi hə də f spesifikasiyalarını mə nimsə diyimizə görə , ağaçdan keçmə k üçün istifadə olunan üsullar bunlardır:

`find_clades` Uyğun elementi ehtiva edə n hə r kladi tapın. Yə ni hə r elementi `find_elements` ilə olduğu kimi tapın, lakin müvafiq clade obyektiini qaytarın. (Adə tə n istə diyiniz budur.)

Nə ticə , defolt olaraq ilk olaraq də rınlıyi axtararaq bütün uyğun gə lə n obyektlə r arasında tə krarlanan bilə ndir. Bu, Newick, Nexus və ya XML mə nbə faylinda görünə n elementlə rlə mütlə q eyni sıra deyil!

`find_elements` Verilmiş atributlara uyğun gə lə n bütün ağac elementlə rini tapın və uyğun elementlə ri özlə rinə qaytarın. Sadə Newick ağaclarının mürə kkə b alt elementlə ri yoxdur, ona görə də bu, onlarda `find_clades` ilə eyni davranışdır. PhyloXML ağaclarında çox vaxt tə bə qə lə rə ə lavə edilmiş mürə kkə b obyektlə r olur, ona görə də bu üsul onları çıxarmaq üçün faydalıdır.

hə r hansı tapmaq `find_elements()` və ya Yox tə rə fində n tapılan ilk elementi qaytarın. Bu, ağacda hə r hansı uyğun elementin olub-olmadığını yoxlamaq üçün də faydalıdır və şə rtli olaraq istifadə edilə bilə r.

Daha iki üsul ağaçdakı qovşaqlar arasında naviqasiya etmə yə kömə k edir:

`get_path` Birbaşa ağac kökü (və ya cari tə bə qə) və verilmiş hə də f arasında tə bə qə lə ri sadalayıñ. Bu yol boyunca verilmiş hə də flə bitə n, lakin kök tə bə qə si istisna olmaqla, bütün tə bə qə obyektlə rinin siyahısını qaytarır.

`trace` Bu ağaçdakı iki hə də f arasındaki bütün klad obyektlə rinin siyahısı. Başlanğıc, o cümlə də n bitmə istisna olmaqla.

16.4.2 İnfomasiya üsulları

Bu üsullar bütün ağac (və ya hə r hansı bir tə bə qə) haqqında mə lumat verir.

`ortaq_a_cdad` Bütün verilmiş hə də flə rinə n son ortaq a cdadını tapın. (Bu Clade obyekti olacaq). Heç bir hə də f verilmə dikdə , cari tə bə qə nin kökünü qaytarır (bu metodun çağırıldığı); 1 hə də f verilirsə , bu hə də fin özünü qaytarır. Bununla belə , göstə rilə n hə də flə rdə n hə r hansı biri cari ağacda (və ya kladada) tapılmazsa, istisna qaldırılır.

`count_terminals` Ağac daxilində terminal (yarpaq) qovşaqlarının sayını hesablayır.

də rınlıklə r Ağac tə bə qə lə rinin də rınlıklə rə xə ritə sini yaradın. Nə ticə , açarların ağaçdakı bütün Clade nümunə lə ri olduğu və də yə rlə rın kökdə n hə r bir tə bə qə yə (terminallar daxil olmaqla) olan mə safə olduğu lügə tdir. Defolt olaraq, mə safə qə fə sə aparan mə cmu budaq uzunluğudur, lakin `unit_branch_lengths=True` seçimi ilə yalnız budaqların sayı (ağaçdakı sə viyyə lə r) sayılır.

mə safə iki hə də f arasındaki budaq uzunluqlarının cə mini hesablayın. Yalnız bir hə də f göstə rilibsə , digə ri bu ağacın köküdür.

ümumi budaq uzunluğu Bu ağaçdakı bütün budaq uzunluqlarının cə mini hesablayın. Bu adə tə n filogenetikada ağacın "uzunluğu" adlanır, lakin biz Python terminologiyası ilə qarışıqlığın qarşısını almaq üçün daha açıq addan istifadə edirik.

Bu üsulların qalan hissə si boolean yoxlamalarıdır:

bifurcating Doğrudur, θ gə r ağac ciddi şə kildə ikilə şırsə ; yə ni bütün qovşaqlarda ya 2, ya da 0 uşaq var (daxili və ya xarici). Kökün 3 nə sli ola bilə r və hə lə də bifurkasiyanın bir hissə si hesab olunur ağaç.

θ gə r verilmiş hə də flə rin hamısı tam yarımkaddan ibarə tdırsə , monofiletik testdir - yə ni, terminalları verilmiş hə də flə rlə eyni də stdə olan tə bə qə mövcuddur. Hə də flə r ağacın terminalları olmalıdır. Rahatlıq üçün bu üsul hə də flə rin monofiletik (Doğru də yə rının e və zinə), θ ks halda isə False olduqda onların ümumi θ cdadını (MCRA) qaytarır.

Hə...də f bu ağaçın nə slində ndırsə , True'nin valideynidir - birbaşa nə slin olması tə lə b olunmur. Bir sinfin birbaşa nə slini yoxlamaq üçün sadə cə olaraq siyahı üzvlük testində n istifadə edin: θ gə r alt sınıf sınıfı : ...

bütün birbaşa nə sillə r terminaldırısa, preterminal Doğrudur; Hə r hansı birbaşa eniş terminal deyilsə , yanlışdır.

16.4.3 Də yişiklik üsulları

Bu üsullar ağaçı yerində də yişdirir. Orijinal ağaçı toxunulmaz saxlamaq istə yirsinizsə , θ və icə Python-un surə t modulundan istifadə edə rə k ağaçın tam surə tini çıxarıın:

```
ağac = Phylo.read("example.xml", "phyloxml") idxal surə ti
```

```
yeni ağac = copy.deepcopy(ağac)
```

yixılma Hə də fi ağacdan silir, uşaqları valideynlə ri ilə yenidə nə laqə lə ndırır.

hamısını çökdürün Bu ağaçın bütün nə slini yixin, yalnız terminalları buraxın. Filial uzunuqları saxlanılır, yə ni hə r terminala olan mə safə eyni qalır. Hə də f spesifikasiyası ilə (yuxariya bax) yalnız spesifikasiyaya uyğun gə lə n daxili qovşaqları yiğir.

ladderize Terminal qovşaqlarının sayına görə kladeslə ri yerində sıralayıın. θ n də rin tə bə qə lə r sonuncu yerlə şdirilir default olaraq. Qrupları θ n də rində n dayaza qə də r çeşidlə mə k üçün tə rs=True istifadə edin.

prune Ağacdan bir terminal tə bə qə ni budanır. Takson bifurkasiyadandırısa, birlə şdirə n qovşaqla dağılacaq və onun filial uzunuğu qalan terminal qovşağına θ lavə olunacaq. Bu artıq mə nali də yə r olmaya bilə r.

kök kə nar qrupla Bu ağaçı, verilmiş hə də flə ri, yə ni kə nar qrupun ümumi θ cdadını ehtiva edə n xarici qrup tə bə qə si ilə yenidə n köklə yin. Bu üsul yalnız Ağac obyektlə rində mövcuddur, Clades deyil.

θ gə r kə nar qrup self.root ilə eynidirsə , heç bir də yişiklik baş vermir. θ gə r kə nar qruplar terminaldırısa (mə sə lə n, tə k terminal qovşağı xarici qrup kimi verilir), verilmiş kə nar qrupa 0 uzunuqlu budaqla yeni ikilə şə n kök tə bə qə si yaradılır. θ ks halda, kə nar qrupun altındaki daxili dügün bütün ağac üçün trifurkasiya kökü olur. Orijinal kök bifurkasiya edirdisə , o, ağacdan atılır.

Bütün hallarda ağaçın ümumi budaqla uzunluğu eyni qalır.

root at midpoint Bu ağaçı ağaçın θ n uzaq iki ucu arasında hesablanmış orta nöqtə də kökündə n çıxarıın.

(Bu, başlıq altında root_with_outgroup istifadə edir.)

split n (standart 2) yeni nə sil yaradın. Növ ağaçında bu növlə şmə hadisə sidir. Yeni sınıflə r verilmiş budaqla uzunluğununa və bu tə bə qə ni kökü ilə eyni ada və üstə gə l tam θ də d şə kilçisinə (0-dan sayımaqla) malikdir – mə sə lə n, "A" adlı tə bə qə ni bölmə k "A0" və "A1" alt sınıflə rini yaradır.

Biopython wiki-də Phylo sə hifə sınıfı baxın (<http://biopython.org/wiki/Phylo>) mövcud üsullardan istifadə yə dair daha çox nümunə üçün.

16.4.4 PhyloXML ağaclarının xüsusiyyə tlə ri

PhyloXML fayl formatına ə lavə mə lumat növlə ri və vizual işaretə lə r ilə ağacları qeyd etmə k üçün sahə lə r daxildir.

Biopython wiki-də PhyloXML sə hifə sinə baxın (<http://biopython.org/wiki/PhyloXML>) tə svir etmə k üçün -

PhyloXML tə rə fində n tə min edilə n ə lavə annotasiya xüsusiyyə tlə rində n istifadə nümunə lə ri və nümunə lə ri.

16.5 Xarici proqramların işə salınması

Bio.Phylo ağacları düzə şdirmə nin özündə nə ticə çıxarmasa da, bunu edə n üçüncü tə rə f proqramları var.

Bunlara alt proses modulundan istifadə etmə klə python daxilində n daxil olmaq olar.

Aşağıda PhyML (<http://www.atgc-montpellier.fr/phym/>).

Proqram phylip-relaxed formatda giriş düzülmə sini qə bul edir (bu Phylip formatıdır, lakin takson adlarında 10 simvol mə hdudiyyə ti olmadan) və müxtə lif variantlar.

```
>>> idxal alt prosesi
>>> cmd = "phym -i Testlə r/Phylib/random.phy"
>>> nə ticə lə r = subprocess.run(cmd, shell=Doğru, stdout=subprocess.PIPE, mə tn=Doğru)
```

'stdout = subprocess.PIPE' arqumenti proqramın çıxışını sazlama mə qsə dlə ri üçün 're-sults.stdout' vasita silə a lçatan edir (eyni şey 'stderr' üçün də edilə bilə r) və 'text=True' geri qaytarılanı edir.

mə lumat 'bayt' obyekti ə və zinə python sa tri olmalıdır.

Bu, [giriş faylinin adı]_phyml_tree.txt və [giriş faylinin adı]_phyml_stats.txt adları ilə ağac faylı və stats faylı yaradır. Ağac faylı Newick formatındadır:

```
>>> Bio idxal Phylo- dan
>>> ağac = Phylo.read("Testlə r/Phylib/random.phy_phym_tree.txt", "newick")
>>> Phylo.draw_ascii(ağac)
```



Subproses modulu hə mçinin ə mr verə n hə r hansı digə r proqramlarla qarşılıqlı ə laqə üçün istifadə edilə bilə r RAxML kimi xə tt interfeysi (<https://sco.h-its.org/exelixis/software.html>), FastTree (<http://www.microbesonline.org/fasttree/>), dnaml və protml.

16.6 PAML integrasiyası

Biopython 1.58 PAML üçün də stə k gə tirdi (<http://abacus.gene.ucl.ac.uk/software/paml.html>), maksimum ehtimalla filogenetik analiz üçün programlar də sti. Hal-hazırda codeml, baseml və yn00 programları hə yata keçirilir. PAML-in icra vaxtı seçimlərinə nə zarət etmək üçün əmr xətti argumentlərinə nə daha çox nə zarət fayllarından istifadə etdiyinə görə, bu sərgidən istifadə Biopython-da digər program sərgilərinin formatından kənara çıxır.

Tipik bir iş axını, uyğunlaşdırma faylini, ağac faylini, çıkış faylini və işçi qovluğununu göstərərək PAML obyektini işə salmaq olardı. Sonra, iş vaxtı seçimləri set options() metoduna ilə və ya mövcud nəzarət faylini oxumaqla təyin edilir. Nəhayət, program run() metoduna ilə işə salınır və çıkış faylı avtomatik olaraq nəticələr ləğətinə təhlil edilir.

Codeml-in tipik istifadəsinə bir nümunə :

```
>>> Bio.Phylo.PAML import codeml -dən >>> cml =  
codeml.Codeml() >>> cml.alignment =  
"Testlər/PAML/Alignments/alignment.phylip" >>> cml.tree = "Testlər/PAML/Trees/species.tree"  
>>> cml.out_file = "nəticələri cml.dir", >>> cml.set_options(seqtype=1,  
ətraflı=0, səskünlü=0, RateAncestor=0, model=0,  
NSsites=[0, 1, 2], CodonFreq=2, cleandata=1,  
fix_alpha=1, kappa=4.54006,  
...  
...  
...  
...  
...  
...  
...  
...  
...  
...  
...  
...  
...  
...  
...  
...  
...  
... )  
>>> nəticələri r = cml.run()  
ns_sites = results.get("NSsites") >>> m0 = ns_sites.get(0)  
>>> m0_params =  
m0.get("parametrləri") >>>  
print(m0_params.get("omega"))
```

Mövcud çıkış faylları modulun read() funksiyasından istifadə etməklə də təhlil edilə bilər:

```
>>> nəticələri r = codeml.read("Testlər/PAML/Results/codeml/codeml_NSSites_all.out") >>> çap(results.get("InL  
max"))
```

Bu yeni modul üçün ətraflı sənədlər hazırda Biopython wiki-də mövcuddur: <http://biopython.org/wiki/PAML>

16.7 Gələcək planlar

Bio.Phylo aktiv inkişaf mərhələ sindədir. Gələcək buraxılışlarda əlavə edə biləcə yimiz bəzi xüsusiyyətlər bunlardır:

Yeni üsullar Ağac və ya Clade obyektləri ilə işləmək üçün ümumiyyətlə faydalı funksiyalar əvvəlcə Biopython vikisində görünür ki, təsadüfi istifadəçilər biz onları Bio.Phylo-ya əlavə etməzdən əvvəl onları sınaqdan keçirə və faydalı olub-olmadığını

qərar verə bilənlər: http://biopython.org/wiki/Phylo_cookbook

Bio.Nexus portu Bu modulun büyük hissə si phyloXML mə lumat formatı üçün Python də stə yinin hə yata keçirilmə si layihə si kimi NESCent-in himayə si altında 2009-cu ilin Google Yay Kod dövründə yazılmışdır (bax [16.4.4](#)).

Newick və Nexus formatlarına də stə k mövcud Bio.Nexus modulunun bir hissə sini Bio.Phylo tə rə fində n istifadə edilə n yeni siniflə rə köçürmə klə ə lavə edildi.

Hal-hazırda, Bio.Nexus hə lə Bio.Phylo siniflə rinə köçürülməmiş bə zi faydalı funksiyaları ehtiva edir - xüsusilə konsensus ağacının hesablanması. Ə gə r siz Bio.Phylo-da bə zi funksionallığın çatışmadığını görürsünüzsə , bunun ə və zinə Bio.Nexus-da olub-olmadığını yoxlayın.

Biz bu modulun funksionallığını və istifadə sini yaxşılaşdırmaq üçün istə nilə n tə klifə açğıq; yalnız bizə icazə verin poçt siyahısında və ya sə hv mə lumat bazamızda bilirik.

Nə hayə t, Phylo moduluna hə lə daxil edilməmiş ə lavə funksionallığa ehtiyacınız varsa, onun DendroPy (<https://dendropy.org/>) kimi filogenetika üçün yüksək keyfiyyə tli Python kitabxanalarından başqa birində mövcud olub-olmadığını yoxlayın. və ya PyCogent (<http://pycogent.org/>). Bu kitabxanalar filogenetik ağaclar üçün standart fayl formatlarını da də stə klə diyi üçün siz müvəqqəti fayla və ya StringIO obyektiñə yazmaqla mə lumatları kitabxanalar arasında asanlıqla ötürə bilə rsiniz.

Fəsil 17

Bio.motiflə rədən istifadə edərək motiv təhlili ardıcılılığı

Bu fəsildə Biopythona daxil olan Bio.motifs paketinin funksionallığı haqqında ümumi məlumat verilir. O, ardıcıl motivlərin təhlili ilə məşğul olan insanlar üçün nəzərdə tutulub, ona görədə güman edirəm ki, siz motiv təhlilininə səs anlayışları ilə tanışsınız. Əgər bir şey aydın deyilsə, lütfən, bəzi müvafiq keçidlər üçün Bölmə [17.10](#)-a baxın.

Bu fəsildən əksəriyyəti Biopython 1.61-dən sonra daxil edilmiş yeni Bio.motifs paketini təsvir edir ki, bu da öz növbəsində iki köhnə keçmiş Biopython moduluna, Bio.AlignAce və Bio.MEME-ə əsaslanan Biopython 1.50 ilə təqdim edilmiş köhnə Bio.Motif paketini əvəz edir. Bu, onların funksionallığınınə əksəriyyətini vahid motiv obyektiininə təbiqilərə min edir.

Digər kitabxanalardan danişarkən, bunu oxuyursunuzsa, [TAMO ilə maraqlana bilərsiniz](#), ardıcıl motivlərlə məşğul olmaq üçün nəzərdə tutulmuş başqa bir python kitabxanası. O, daha çox de-novo motiv tapıcılarıdır stəkləyir, lakin o, Biopython-un bir hissəsi deyil və kommersiya istifadəsinə dair bəzi məhdudiyyətlərə malikdir.

17.1 Motif obyektləri

Motif təhlili ilə maraqlandığımız üçün ilk növbədə Motif obyektlərinə nəzər salmalıyıq. Bunun üçün Bio.motifs kitabxanasını idxlə etməliyik:

>>> Bio.idxlə motivlərindən

və biz ilk motiv obyektlərimizi yaratmağa başlaya bilərik. Biz ya motiv nümunələri siyahısından Motif obyekti yarada bilərik, ya da motivlərbazasından və ya motiv tapmaq programından faylı təhlil edərək Motif obyektiini əldə edə bilərik.

17.1.1 Nümunələrindən motiv yaratmaq

Tutaq ki, bizdə DNT motivinin bu nümunələri var:

>>> Bio.Seq -dən idxlə Seq >>>

nümunələri =

```
...      [ Seq("TACAA"),
...      Seq("TACGC"),
...      Seq("TACAC"),
...      Seq("TACCC"),
...      Seq("AACCC"),
...      Seq("AATGC"),
```

```
...           Seq("AATGC"),
... ]
```

sonra aşağıdakı kimi Motif obyekti yarada bilə rik:

```
>>> m = motifs.create(nümunə lə r)
```

Bu motivin yaradıldığı nümunə lə r.alignment xassə sində saxlanılır:

```
>>> çap (m.alignment.sequences)
[Seq('TACAA'), Seq('TACGC'), Seq('TACAC'), Seq('TACCC'), Seq('AACCC'), Seq('AATGC'), Seq('AATGC')]
```

Motif obyektinin çap edilmə si onun qurulduğu nümunə lə ri göstə rir:

```
>>> çap (m)
TACAA
TACGC
TACAC
TACCC
AACCC
AATGC
AATGC
```

Motifin uzunluğu ardıcılıq uzunluğu kimi müə yyə n edilir və bütün nümunə lə r üçün eyni olmalıdır:

```
>>> len(m)
5
```

Motif obyekti hə r mövqedə hə r nukleotidin sayını ehtiva edə n .counts atributuna malikdir. Bu sayma matrisini çap etmə k onu asanlıqla oxuna bilə n formatda göstə rir:

```
>>> çap (m.sayı)
      1       2       3       4
A: 3.00 7.00 0.00 2.00 1.00
C: 0,00 0,00 5,00 2,00 6,00
G: 0,00 0,00 0,00 3,00 0,00
T: 4.00 0.00 2.00 0.00 0.00
<BLANKLINE>
```

Bu saylara lügə t kimi daxil ola bilə rsiniz:

```
>>> m.sayı["A"] [3.0, 7.0,
0.0, 2.0, 1.0]
```

ancaq siz onu hə m də birinci ölçü kimi nukleotid və ikinci ölçü kimi mövqeyi olan 2D massiv kimi də düşünə bilə rsiniz:

```
>>> m.counts["T", 0] 4.0 >>>
```

```
m.counts["T", 2] 2.0
```

```
>>> m.counts["T", 3] 0.0
```

Siz hə mçinin sayma matrisinin sütunlarına birbaşa daxil ola bilə rsiniz

```
>>> m.sayilar[:, 3]
{'A': 2.0, 'C': 2.0, 'T': 0.0, 'G': 3.0}
```

Nukleotidin özününə və zinə motivinə lifbasındaki nukleotidin indeksindəndə istifadə edə bilərsiniz:

```
>>> m.ə lifbasi 'ACGT'
```

```
>>> m.sayir["A", :] (3.0, 7.0, 0.0,
2.0, 1.0) >>> m.sayir[0, :] (3.0, 7.0, 0.0,
2.0, 1.0)
```

17.1.2 Konsensus ardıcılığınınə ldə edilməsi

Motifin konsensus ardıcılığı .counts matrisinin müvafiq sütunlarındaən böyük dəyərin alındığı motivin mövqeləri boyunca hərflərin ardıcılığı kimi müəyyən edilir:

```
>>> m.konsensus
Seq('TACGC')
```

Əksinə, antikonsensus ardıcılığı .counts matrisinin sütunlarındakıən kiçik qiymətlərə uyğundur:

```
>>> m.antikonsensus
Seq('CCATG')
```

Nəzərə alın ki, bəzi sütunlarda çoxsaylı nukleotidlərin maksimum və ya minimum sayı varsa, konsensus və antikonsensus ardıcılığının tərifində müəyyən qeyri-müəyyənlilik var.

DNT ardıcılığı üçün siz həmçinin qeyri-müəyyən nukleotidlərin olduğu degenerativ konsensus ardıcılığını istəyə bilərsiniz. Çox sayda nukleotidin olduğu mövqelər üçün istifadə olunur:

```
>>> m.degenerate_consensus
Seq('WACVC')
```

Burada W və R IUPAC nukleotid qeyri-müəyyənlilik kodlarınaəməledirlər: W ya A, ya T, V isə A, C və ya G [7].

Degenerasiya konsensus ardıcılığı Cavener [3] tərəfindən müəyyən edilmiş qaydalaraəsasən qurulur.

`motif.counts.calculate_consensus` metodu konsensus ardıcılığının necə hesablanacağınıətraflı şəkildə müəyyən etməyə imkan verir. Bu üsuləsasən EMBOSS programınınəmənficəhərlərinin konvensiyalarınaəməledir və aşağıdakı arqumentləri götürür:

Əvəzətmə matrisi Ardıcılıqları müqayisədə rəkən istifadə olunan bal matrisi. Varsayılan olaraq, Yoxdur, bu halda biz sadəcə olaraq hərfin tezliyini hesablayırıq. Varsayılan da yərəvəzinəsiz Bio.Align.substitution_matrislərində mövcud olanəvəzedici matrislərdən istifadədə bilərsiniz. Ümumi seçimlər rüzlər üçün BLOSUM62 (həmçinin EBLOSUM62 kimi tanınır) və nukleotidlər üçün NUC.4.4 (həmçinin EDNAFULL kimi tanınır) olur.

QEYD: Hal-hazırda, bu üsul standart dəyər Yoxdan başqa dəyərlər üçün hələtəbliğ edilməyib.

Çoxluq Konsensusa nail olmaq üçün tələb olunan müsbət uyğunluqların sayı üçün həddədəyəri, sütundakı ümumi sayı bölünür.

Əvəzətmə _matrisası Yoxdur, bu arqument dəyəri olmalıdır və nəzərə alınmır; ValueError başqa cür qaldırılır.

Əvəzətmə _matrisası Yox deyilsə, çoxluğun standart dəyəri 0,5-dir.

identity Konsensus dəyəri müəyyən etmək üçün tələb olunan identifikasiyaların sayı, sütundakı ümumi sayı bölünür.

Əgər eyniliklərin sayı sütundakı ümumi sayı vurulan eynilikdən azdırsa, konsensus ardıcılığında qeyri-müəyyən xarakter (nukleotidlər üçün N və amin turşusu ardıcılığı üçün X) istifadə olunur.

Şəxsiyyət 1.0 olarsa, konsensusa yalnız eyni hərflərdən ibarət sütunlar kömək edir. Defolt dəyəri sıfır.

müsəbət uyğunluqlar üçün setcase həddi, yuxarıda konsensusun böyük hərf olduğu və konsensusun kiçik hərf olduğu sütundakı ümumi saya bölünür. Varsayılan olaraq, bu 0,5-ə bərabərdir.

Bu bir nümunədir:

```
>>> m.counts.calculate_consensus(identity=0,5, setcase=0,7) 'tACNC'
```

17.1.3 Motifin tərs tamamlanması

Motifinəks tamamlamasını onun üzərində tərs_tamamlayıcı metodunu çağırmaqla eldə edə bilərik:

```
>>> r = m.reverse_complement() >>> r.consensus
```

```
Seq('GCGTA') >>>
r.degenerate_consensus Seq('GBGTW')
>>> çap(r)
```

```
TTGTA
GCGTA
GTGTA
GGGTA
GGGTT
GCATT
GCATT
```

Əks tamamlayıcı yalnız DNT motivləri üçün müəyyən edilir.

17.1.4 Motifin dilimlənməsi

Seçilmiş mövqelər üçün yeni Motif obyekti eldə etmək üçün motivi dilimləyə bilərsiniz:

```
>>> m_sub = m[2:-1] >>> çap
(m_sub)
CA
CG
CA
CC
CC
TG
TG
>>> m_sub.consensus Seq('CG')
>>>
m_sub.degenerate_consensus Seq('CV')
```

17.1.5 Nisbi entropiya Motifin sütununun

nisbi entropiyası (və ya Kullback-Leibler məsafəsi) Hj kimi müəyyən edilir [39, 11]

$$H_j = \sum_{i=1}^M p_{ij} \log \frac{p_{ij}}{b_i}$$

harada:

- M – Ə libbadakı hə rflə rin sayı (`len(m.alphabet)`) ilə verilir;
- p_{ij} – j-ci sütunda normallaşdırılmış i hə rfinin müşahidə edilə n tezliyi (aşağıya bax);
- b_i – i hə rfinin fon ehtimalı (`m.background[i]`) tə rə fində n verilmişdir).

Müşahidə olunan tezlik p_{ij} aşağıdakı kimi hesablanır:

$$= \frac{c_{ij} + k}{C_j + k}$$

harada:

- c_{ij} – düzülmə nin j sütununda i hə rfinin neçə də fə görünmə si (`m.counts[i, j]` ilə verilir);
- C_j – j sütununda hə rflə rin ümumi sayı: $C_j = \sum_{i=1}^M c_{ij}$ (ə mi ilə verilir (`m.saymalar[:, j]`)).
- k_i – i hə rfinin pseudocountu (`m.pseudocounts[i]`) tə rə fində n verilmişdir).
- k – ümumi pseudohesab: $k = \sum_{i=1}^M k_i$ (`sum(m.pseudocounts.values())`) ilə verilir.

Bu tə riflə rlə hə m p_{ij} , hə m də b_i 1-ə normallaşdırılır:

$$\begin{aligned} p_{ij} &= 1 \\ &\sum_{i=1}^M p_{ij} = 1 \\ b_i &= 1 \\ &\sum_{i=1}^M b_i = 1 \end{aligned}$$

Fon paylanması vahiddirsə , nisbi entropiya informasiya mə zmunu ilə eynidir.

Motifin hə r bir sütunu üçün nisbi entropiya nisbi_entropiya xassə sində n istifadə etmə klə ə ldə edilə bilə r:

```
>>> m.nisbi_entropiya
massivi([1.0147186, 2.1.13687943, 0.44334329, 1.40832722])
```

Bu də yə rlə r baza-2 loqarifmi ilə hesablanır və buna görə də bit vahidlə rində dir. İkinci sütun (yalnız A nukleotidlə rində n ibarə tdir) ə n yüksə k nisbi entropiyaya malikdir; dördüncü sütun (A, C və ya G nukleotidlə rində n ibarə tdir) ə n aşağı nisbi entropiyaya malikdir). Motifin nisbi entropiyasını sütunlar üzə rində cə mlə mə klə hesablamaq olar:

```
>>> cə mi(m.nisbi_entropiya) # doctest:+ELLIPSIS 6.003321...
```

17.1.6 Ardıcılıq loqosunun yaradılması

İnternetə çıxışımız varsa, [vebloqu](#) yarada bilə rik:

```
>>> m.weblogo("mymotif.png")
```

Logomuzu müə yyə n edilmiş faylda PNG olaraq saxlamalıyıq.

17.2 Oxu motivlə ri

Nümunə lə rdə n motivlə rin ə l ilə yaradılması bir az darixdircidir, ona görə də motivlə ri oxumaq və yazmaq üçün bə zi I/O funksiyalarının olması faydalıdır. Motiflə rin saxlanması üçün hə qıqə tə n yaxşı müə yyə n edilmiş standartlar yoxdur, lakin digə rlə rində n daha çox istifadə olunan bir neçə format var.

17.2.1 JASPAR

Ə n mə şur motiv verilə nlə r bazalarından biri [JASPAR-dir](#). Motif ardıcılılığı mə lumatlarına ə lavə olaraq, JASPAR verilə nlə r bazası hə r motiv üçün çoxlu meta-mə lumat saxlayır. Bio.motifs modulu bu meta-mə lumatın atributlar kimi tə qdim olunduğu ixtisaslaşdırılmış jaspar.Motif sinfində n ibarə tdir:

- matrix_id - unikal JASPAR motiv ID-si, mə sə lə n, 'MA0004.1'
- ad - TF-nin adı, mə sə lə n, 'Arnt'
- kolleksiya - motivin aid olduğu JASPAR kolleksiyası, mə sə lə n, 'CORE'
- tf_class - bu TF-nin struktur sinfi, mə sə lə n, "Zipper-Type"
- tf_family - bu TF-nin aid olduğu ailə , mə sə lə n, 'Helix-Loop-Helix'
- növlə r - bu TF-nin aid olduğu növlə r, bir neçə qiymə tə malik ola bilə r, bunlar kimi göstə rilir
taksonomiya identifikasiyaları, mə sə lə n, 10090
- vergi_qrupu - bu motivin aid olduğu taksonomik superqrup, mə sə lə n, 'onurğalılar'
- acc - TF züllalının qoşulma nömrə si, mə sə lə n, 'P53762'
- data_type - bu motivi yaratmaq üçün istifadə olunan mə lumat növü, mə sə lə n, 'SELEX'
- medline - bu motivi də stə klə yə n ə də biyyatın Pubmed ID-si, bir neçə də yə r ola bilə r, mə sə lə n, 7592839
- pazar_id - PAZAR verilə nlə r bazasında TF-yə xarici istinad, mə sə lə n, 'TF0000003'
- şə rh - motivin qurulması haqqında qeydlə ri ehtiva edə n sə rbə st formada mə tn

jaspar.Motif sinfi ümumi Motif sinfində n miras qalır və buna görə də bütün imkanları tə min edir motiv formatlarından hə r hansı biri - motivlə rin oxunması, motivlə rin yazılmazı, motiv nümunə lə ri üçün ardıcılığın skan edilmə si və s. JASPAR motivlə ri üç müxtə lif düz fayl formatı və SQL verilə nlə r bazası kimi bir neçə fə rqli yolla saxlayır. Bütün bu formatlar sayma matrisinin qurulmasını asanlaşdırır. Bununla belə , yuxarıda tə svir edilə n meta mə lumatların miqdarı formata görə də yişir.

JASPAR saytları formatı

Üç düz fayl formatından birincisi nümunə lə rin siyahısını ehtiva edir. Nümunə olaraq bunlar JASPAR Arnt.sites faylinin başlanğıc və son sə tirlə ridir və Arnt siyan spiral-döşə mə -spiral transkripsiya faktorunun mə lum bağlanması yerlə rini göstə rir.

```
>MA0004 ARNT 1
CACGTGatgtccctc
>MA0004 ARNT 2
CACGTGgggaggtagc
>MA0004 ARNT 3
CACGTGccgcgcgc
...
>MA0004 ARNT 18
AACGTGacagccctcc
>MA0004 ARNT 19
AACGTGcacatcgccc
>MA0004 ARNT 20
agaatCGCGTGC
```

Ardıcılığın büyük hə rflərlə yazılmış hissələri bir-birinə uyğun gələn motiv nümunələridir.

Bu nümunələr dən aşağıdakı kimi Motif obyekti yarada bilərilər:

```
>>> Bio idxal motivlərindən >>>
qulp kimi open("Arnt.sites") ilə :
...     arnt = motifs.read(qulp, "saytlar")
...
```

Bu motivin yaradıldığı nümunələr .alignment xassəsində saxlanılır:

```
>>> çap (arnt.alignment.sequences[3])
[Arnt.alignment.sequences -də ardıcılıqları üçün [Seq('CACGTG'),
Seq('CACGTG'), Seq('CACGTG')] >>> print(sequence)
...
...
CACGTG
CACGTG
CACGTG
CACGTG
CACGTG
CACGTG
CACGTG
CACGTG
CACGTG
CACGTG
CACGTG
CACGTG
CACGTG
CACGTG
CACGTG
AACGTG
AACGTG
AACGTG
AACGTG
CGCGTG
```

Bu motivin say matrisi avtomatik olaraq misallardan hesablanır:

```
>>> çap edin (arnt.sayilar) 1 2
          0           3       4       5
A: 4.00 19.00 0.00 0.00 0.00 0.00
C: 16.00 0.00 20.00 0.00 0.00 0.00 G: 0.00 1.00 0.00 20.00
0.00 20.00 T: 0.00 0.00 0.00 0.00 20.00 LINE.
```

Bu format heç bir meta məlumat saxlamır.

JASPAR pfm formatı

JASPAR həmçinin motivləri yaradıldığı nümunələr olmadan birbaşa say matrisi kimi təqdim edir. Bu pfm formatı yalnız bir motiv üçün sayma matrisini saxlayır. Məsələn, bu, insan SRF transkripsiya faktoru üçün say matrisini ehtiva edən SRF.pfm JASPAR faylıdır:

```
2 9 0 1 32 3 46 1 43 15 2 2
1 33 45 45 1 1 0 0 0 1 0 1
39 2 1 0 0 0 0 0 0 44 43
4 2 0 0 13 42 0 45 3 30 0 0
```

Bu sayma matrisi üçün aşağıdakı kimi motiv yarada bilə rik:

```
>>> tutacaq olaraq open("SRF.pfm") ilə :
...
...     srf = motivlə r.oxu (tutacaq, "pfm")
...
>>> çap (srf.counts)
      0       1       2       3       4       5       6       7       8       9       10      11
A: 2,00 9,00 0,00 1,00 32,00 3,00 46,00 1,00 43,00 15,00 2,00 2,00
C: 1,00 33,00 45,00 45,00 1,00 1,00 0,00 0,00 0,00 1,00 0,00 1,00
G: 39,00 2,00 1,00 0,00 0,00 0,00 0,00 0,00 0,00 44,00 43,00
T: 4,00 2,00 0,00 0,00 13,00 42,00 0,00 45,00 3,00 30,00 0,00 0,00
<BLANKLINE>
```

Bu motiv bilavasitə saylar matrisində n yaradıldığı üçün onunla ə laqə li heç bir nümunə yoxdur:

```
>>> çap (srf.alignment)
Heç biri
```

İndi bu iki motivin konsensus ardıcılığını istə yə bilə rik:

```
>>> çap (arnt.counts.consensus)
CACGTG
>>> çap (srf.counts.consensus)
GCCCATATGG
```

Nümunə lə r faylında olduğu kimi, bu formatda heç bir meta mə lumat saxlanılmır.

JASPAR formatı jaspar

Jaspar fayl formatı bir faylda çoxlu motivlə rin göstə rilmə sinə imkan verir. Bu formatda hə r bir motiv qeydlə r başlıq sə tirində n sonra saylar matrisini tə yin edə n dörd sə tirdə n ibarə tdir. Başlıq xə tti ilə başlayır a > simvolu (Fasta fayl formatına bə nzə yir) və onun ardınca unikal JASPAR matris identifikatoru və TF adı. Aşağıdakı misal üç motivli Arnt, RUNX1 olan jaspar formatlı faylı göstə rir və MEF2A:

```
>MA0004.1 Arnt
A [ 4 1 9 0 0 0 0 ]
C [ 1 6 0 2 0 0 0 0 ]
G [ 0 1 0 2 0 0 2 0 ]
T [ 0 0 0 0 2 0 0 ]
>MA0002.1 RUNX1
A [ 1 0 1 2 4 1 2 2 0 0 0 8 1 3 ]
C [ 2 2 7 1 0 8 0 0 1 2 2 ]
G [ 3 1 1 0 2 3 0 2 6 2 6 0 0 4 ]
T [ 1 1 1 1 1 4 2 4 1 1 6 0 0 2 5 1 6 7 ]
>MA0052.1 MEF2A
A [ 1 0 5 7 2 9 6 3 7 2 5 6 6 ]
C [ 5 0 0 1 1 0 0 0 0 0 0 ]
G [ 0 0 0 0 0 0 0 0 2 5 0 ]
T [ 7 5 8 0 5 5 4 9 5 2 2 1 5 6 0 2 ]
```

Motiflə r aşağıdakı kimi oxunur:

```
>>> fh = open("jaspar_motifs.txt")
>>> m üçün motifs.parse (fh, "jaspar"):
...     çap (m)
...
TF adı          Arnt
Matris ID       MA0004.1
Matris:
    0      1      2      3      4      5
A: 4,00 19,00 0,00 0,00 0,00 0,00
C: 16,00 0,00 20,00 0,00 0,00 0,00
G: 0,00 1,00 0,00 20,00 0,00 20,00
T: 0,00 0,00 0,00 0,00 20,00 0,00
```

```
TF adı          RUNX1
Matris ID       MA0002.1
Matris:
    0      1      2      3      4      5      6      7      8      9      10
A: 10,00 12,00 4,00 1,00 2,00 2,00 0,00 0,00 8,00 13,00
C: 2,00 2,00 7,00 1,00 0,00 8,00 0,00 0,00 1,00 2,00 2,00
G: 3,00 1,00 1,00 0,00 23,00 0,00 26,00 26,00 0,00 0,00 4,00
T: 11,00 11,00 14,00 24,00 1,00 16,00 0,00 0,00 25,00 16,00 7,00
```

```
TF adı          MEF2A
Matris ID       MA0052.1
Matris:
    0      1      2      3      4      5      6      7      8      9
A: 1,00 0,00 57,00 2,00 9,00 6,00 37,00 2,00 56,00 6,00
C: 50,00 0,00 1,00 1,00 0,00 0,00 0,00 0,00 0,00 0,00
G: 0,00 0,00 0,00 0,00 0,00 0,00 0,00 2,00 50,00
T: 7,00 58,00 0,00 55,00 49,00 52,00 21,00 56,00 0,00 2,00
```

Qeyd edə k ki, JASPAR motivinin çapı hə m sayma mə lumatlarını, hə m də mövcud meta-mə lumatı verir.

JASPAR verilə nlə r bazasına daxil olmaq

Bu düz fayl formatlarını tə hlil etmə klə yanaşı, biz JASPAR SQL verilə nlə r bazasından motivlə ri də e ldə edə bilə rik. Düz fayl formatlarından fə rqli olaraq, JASPAR verilə nlə r bazası müə yyə n edilmiş bütün mümkün meta mə lumatların saxlanmasına imkan verir JASPAR Motif sinfi. JASPAR-in necə qurulacağını tə svir etmə k bu sə nə din e hatə dairə sində n kə nardadır verilə nlə r bazası (ə sas [JASPAR](#) vebsayt). Motiflə r JASPAR verilə nlə r bazasından istifadə edə rə k oxunur Bio.motifs.jaspar.db modulu. Ə və lə icə JASPAR5 sinifində n istifadə edə rə k JASPAR verilə nlə r bazasına qoşulun e n son JASPAR sxemini modellə şdirir:

```
>>> Bio.motifs.jaspar.db -də n idxal JASPAR5
>>>
>>> JASPAR_DB_HOST = "yourhostname" # bu də yə rlə ri doldurun
>>> JASPAR_DB_NAME = "verilə nlə r bazanız"
>>> JASPAR_DB_USER = "user adınız"
```

```
>>> JASPAR_DB_PASS = "parolunuz"
```

```
>>>
```

```
>>> jdb = JASPAR5(
...     host=JASPAR_DB_HOST,
...     ad=JASPAR_DB_NAME,
...     istifadə_çı=JASPAR_DB_USER,
...     parol = JASPAR_DB_PASS,
... )
```

İndi `fetch_motif_by_id` metodunu ilə unikal JASPAR ID-si ilə tək motivi əldə edə bilərik. Qeyd JASPAR ID-nin əsas identifikatorдан və onluq nöqtə ilə ayrılmış versiya nömrə sindən ibarət olduğunu, məsələn, 'MA0004.1'. `fetch_motif_by_id` metodunu tam məsəyyən edilmiş ID-dən, ya da sadəcə əsas ID-dən istifadə etməyə imkan verir. Kaşki əsas identifikator verilir, motivin ən son versiyası qaytarılır.

```
>>> arnt = jdb.fetch_motif_by_id("MA0004")
```

Motifin çapı JASPAR SQL verilənlər bazasında mənzildən daha çox meta-məlumat saxladığı göstərir. fayllar:

```
>>> çap (arnt)
```

TF adı Arnt

Matris ID MA0004.1

CORE kolleksiyası

TF sinfi	Fermuar Tipi
TF ailəsi	Helix-Loop-Helix
Növlər	10090
Taksonomik qrup	onurğalılar
Qoşulma	[P53762]
İstifadə olunan məlumat növü	SELEX
Medline	7592839
PAZAR ID	TF0000003
Şəhərlər	-
Matris:	
	0 1 2 3 4 5
A:	4.00 19.00 0.00 0.00 0.00
C:	16.00 0.00 20.00 0.00 0.00
G:	0.00 1.00 0.00 20.00 0.00 20.00
T:	0.00 0.00 0.00 20.00 0.00

Biz motivləri adla da gətirə bilərik. Adəqiq uyğunluq olmalıdır (qismən uyğunluqlar və ya verilənlər bazası joker işarəsi lər hazırladı dəstəklənmir). Qeyd edək ki, adın unikal olacağına zəmanət verilmədiyi üçün `fetch_motifs_by_name` metodunu əsliндə siyahı qaytarır.

```
>>> motivlər = jdb.fetch_motifs_by_name("Arnt")
```

```
>>> çap (motivlər[0])
```

TF adı Arnt

Matris ID	MA0004.1
Kolleksiya	CORE
TF sinfi	Fermuar Tipi
TF ailəsi	Helix-Loop-Helix
Növlər	10090
Taksonomik qrup	onurğalılar

Qoşulma	[P53762]
İstifadə olunan mə lumat növü	SELEX
Medline	7592839
PAZAR ID	TF0000003
Şə rhlə r	-
Matris:	
	0 1 2 3 4 5
A: 4,00 19,00 0,00 0,00 0,00 0,00	
C: 16,00 0,00 20,00 0,00 0,00 0,00	
G: 0,00 1,00 0,00 20,00 0,00 20,00	
T: 0,00 0,00 0,00 20,00 0,00	

fetch_motifs metodu müə yyə n meyarlar də stinə uyğun gə lə n motivlə ri ə ldə etmə yə imkan verir. Bunlar meyarlara yuxarıda tə svir edilə n meta mə lumatlardan hə r hansı biri, hə mçinin müə yyə n matris xassə lə ri daxildir minimum mə lumat mə zmunu (aşağıdakı misalda min_ic), matrisin minimum uzunluğu və ya matrisin qurulması üçün istifadə edilə n saytların minimum sayı. Yalnız bütün göstə rilə n meyarlari keçə n motivlə r qaytarılır. Nə zə rə alın ki, çoxluğa imkan verə n meta mə lumatlara uyğun gə lə n seçim meyarlari də yə rlə r tə k də yə r və ya də yə rlə r siyahısı kimi göstə rilə bilə r, mə sə lə n, vergi_qrupu və tf_family aşağıda misal.

```
>>> motiflə r = jdb.fetch_motifs(
...     kolleksiya = "CORE",
...     tax_group=["onurğalılar", "böcə klə r"],
...     tf_class="Qanadlı Spiral-Dönüş-Heliks",
...     tf_family=["Forkhead", "Ets"],
...     min_ic=12,
... )
>>> motiflə rdə motiv üçün :
...     pass # motivi ilə bir şey edin
...
```

Perl TFBS modulları ilə uygunluq

Qeyd etmə k lazımdır ki, JASPAR Motif sınıfı ilə uyğun olmaq üçün nə zə rdə tutulmuşdur mə şhur [Perl TFBS modulları](#). Buna görə də fon üçün standartların seçilmə si ilə bağlı bə zi xüsusiyyə tlə r və yalançı hesablar, elə cə də mə lumat mə zmununun necə hesablandığı və nümunə lə r üçün axtarılan ardıcılığın ə sasıldığı bu uygunluq meyarlari üzrə . Bu seçimlə r aşağıdakı xüsusi alt bölmə lə rdə qeyd olunur.

- Fon seçimi:
Perl TFBS modulları xüsusi fon ehtimallarının seçiminiə imkan verir (baxmayaraq ki sə nə dlə r vahid fonun qə bul edildiyini bildirir). Ancaq standart formadan istifadə etmə kdir fon. Buna görə hesablama üçün vahid fondan istifadə etmə yiniz tövsiyə olunur mövqeyə xüsusi xal matrisi (PSSM). Bu, Biopython motiflə ri modulundan istifadə edə rkə n defoltdur.
 - Pseudocounts seçimi:
Varsayılan olaraq, Perl TFBS modulları N bg[nukleotid]-ə bə rabə r psevdohesabdan istifadə edir, burada N tə msil edir matrisin qurulması üçün istifadə olunan ardıcılıqların ümumi sayı. Eyni psevdocount formulunu tə tbiq etmə k üçün, jaspar.calculate_pseudocounts() funksiyasından istifadə edə rə k motiv pseudocounts atributunu tə yin edin:
- ```
>>> motif.pseudocounts = motifs.jaspar.calculate_pseudocounts(motif)
```

Nə zə rə alın ki, sayılar matrisinin sütunları tə şkil edə n qeyri-bə rabə r sayda ardıcılılığı ola bilə r. Pseudocount hesablama matrisi tə şkil edə n ardıcılıqların orta sayından istifadə edir.

Bununla belə , hesablamalar matrisində normallaşdırma çağrırlıqdır, sütundakı hə r bir hesablama də yə ri ardıcılıqların orta sayına deyil, hə min xüsusi sütunu tə şkil edə n ardıcılıqların ümumi sayına bölünür.

Bu, Perl TFBS modullarından fə rqlə nır, çünkü normallaşdırma ayrıca bir addım kimi aparılmış və buna görə də pssm-in hesablanması zamanı orta ardıcılıq sayından istifadə olunur. Buna görə də , qeyri-bə rabə r sütun sayıları olan matrislə r üçün motivlə r modulu tə rə fində n hesablanan PSSM Perl TFBS modulları tə rə fində n hesablanan pssm-də n bir qə də r fə rqlə nə cə kdir.

- Matris mə lumat mə zmununun hesablanması: Matrisin informasiya mə zmunu (IC) və ya spesifikliyi PositionSpecificScoringMatrix sinifinin orta metodundan istifadə etmə klə hesablanır. Bununla belə , qeyd edə k ki, Perl TFBS modullarında defolt davranış ilkin olaraq PSSM-lə r yuxarıda tə svir olunduğu kimi psevdothesablardan istifadə edilmə klə hesablanısa da, ilk növbə də psevdothesabları tə tbiq etmə də n IC-ni hesablamacıdır.
- Nümunə lə r axtarılır:  
Perl TFBS motivlə ri ilə nümunə lə rin axtarışı adə tə n nisbi xal hə ddində n, yə ni 0-dan 1-ə qə də r olan xalla hə yata keçirilirdi. Nisbi bala uyğun gə lə n mütlə q PSSM xalını hesablamaq üçün tə nlilikdə n istifadə etmə k olar:

```
>>> abs_score = (pssm.max - pssm.min) * rel_score + pssm.min
```

Nümunə nin mütlə q xalını nisbi bala çevirmə k üçün tə nlilikdə n istifadə etmə k olar:

```
>>> rel_score = (abs_score - pssm.min) / (pssm.max - pssm.min)
```

Mə sə lə n, a vvə l Arnt motivində n istifadə edə rə k, nisbi xal hə ddi 0,8 olan ardıcılığı axtaraq.

```
>>> test_seq = Seq("TAAGCGTGCACGCGCAACGTGCATTA")>>>
arnt.pseudocounts = motifs.jaspar.calculate_pseudocounts(arnt)>>> pssm = arnt.pssm >>>
max_score = pssm.max =>>
min >>> ms . abs_score_threshold
= (max_score - min_score) * 0.8 +
min_score >>> pos üçün , xal pssm.search (test_seq, threshold=abs_score_threshold): rel_score =
(bal - min_score) / (max_score - min_score) / (maksimum_score - min_score) = 5 xal : posf . rel . xal =
... {rel_score:5.3f}"
```

Mövqe 2: xal = 5.362, rel. xal = 0,801 Mövqe 8: xal = 6,112, rel. xal = 0,831 Mövqe -20: xal = 7,103, rel. xal = 0,870 Mövqe 17: xal = 10,351, rel. xal = 1.000 Mövqe -11: xal = 10.351, rel. xal = 1.000

## 17.2.2 MEME

MEME [2] ə laqə li DNT və ya zülal ardıcılığı qrupunda motivlə ri aşkar etmə k üçün bir vasitə dir. Giriş kimi bir qrup DNT və ya zülal ardıcılığı götürür və tə lə b olunduğu qə də r çox motivi çıxarır. Buna görə də , JASPAR fayllarından fə rqlı olaraq, MEME çıxış faylları adə tə n çoxlu motivlə rdə n ibarə tdir. Bu bir nümunə dir.

MEME tə rə fində n yaradılan çıxış faylinin yuxarı hissə sində MEME haqqında bə zi fon mə lumatları göstərilir və istifadə olunan MEME versiyası:

```

MEME - Motif kə şf alə ti

```

MEME versiyası 3.0 (buraxılış tarixi: 2004/08/18 09:07:01)

...

Daha sonra, mə şq ardıcılığının giriş də sti tə krarlanır:

```

```

TƏ LİM DƏ STİ

```

```

DATAFILE= INO\_up800.s

Ə LİFBA = ACGT

| Ardiciliğin adı | Çə ki Uzunluq Sırasının adı | Çə ki Uzunluğu |
|-----------------|-----------------------------|----------------|
| CHO1            | 1.0000 800 CHO2             | 1.0000 800     |
| FAS1            | 1.0000 800 FAS2             | 1.0000 800     |
| ACC1            | 1.0000 800 INO1             | 1.0000 800     |
| OPI3            | 1.0000 800                  |                |

və istifadə olunan də qıq ə mr xə tti:

```

```

Ə MƏ R XƏ TTİNİN XÜLASƏ Sİ

```

```

Bu mə lumat hə mçinin bildirmə k istə diyiniz halda faydalı ola bilə r a  
MEME proqram tə minatı ilə bağlı problem.

ə mr: meme -mod oops -dna -revcomp -nmotifs 2 -bfile yeast.nc.6.freq INO\_up800.s

...

Sonra tapılan hə r bir motiv haqqında ə traflı mə lumat verilir:

```

```

MOTIF 1 eni = 12 sayt = 7 llr = 95 E-də yə r = 2.0e-001

```

```

Motif 1 Tə sviri

|                                                |                                                                   |
|------------------------------------------------|-------------------------------------------------------------------|
| Sadə lə şdirilmiş mövqə-xüsusi ehtimal matrisi | A ::9:a:::3:<br>C ::a:9:11691a<br>G ::::1:94:4:<br>T aa:1::9::11: |
|------------------------------------------------|-------------------------------------------------------------------|

Bu faylı (meme.dnaoops.txt kimi saxlanılır) tə hlil etmə k üçün istifadə edin

```
>>> qulp kimi open("meme.INO_up800.classicoops.xml") ilə :
... rekord = motifs.parse (tutacaq, "meme")
...
```

motifs.parse ə mri tam faylı birbaşa oxuyur, belə liklə siz motifs.parse çağırıldıqdan sonra faylı bağlaya bilə rsiniz. Başlıq mə lumatları atributlarda saxlanılır:

```
>>> rekord.versiya
'5.0.1'
>>> qeyd.mə lumat faylı
'ümumi/INO_up800.s'
```

```
>>> record.command 'meme
common/INO_up800.s -oc results/meme10 -mod oops -dna -revcomp -bfile common/yeast.nc.6.freq -nmot >>> record.alphabet 'ACGT'
```

```
>>> qeyd.sequences
['ardicilliq_0', 'ardicilliq_1', 'ardicilliq_2', 'ardicilliq_3', 'ardicilliq_4', 'ardicilliq_5', 'ardicilliq_6']
```

Qeyd Bio.motifs.meme.Record sinifinin obyektidir. Sınıf siyahıdan miras qalır və siz qeydi Motif obyektlərinin siyahısı kimi düşünə bilərsiniz:

```
>>> len (rekord)
2
>>> motiv = qeyd[0] >>> çap
(motif.consensus)
GCGGCATGTGAAA
>>> çap (motif.degenerate_consensus)
GSKGCATGTGAAA
```

Bu ümumi motiv attributlarına əlavə olaraq, hər bir motiv MEME tərəfindən hesablanmış xüsusi məlumatı da saxlayır. Məsələn,

```
>>> motiv.sayı_başvermə 1 >>>
```

motiv.uzunluq 13

```
>>> qiymə tləndirmə = motiv.qiymə tləndirmə
>>> çap ("%3.1g" % qiymə tləndirmə)
0.2
>>> motif.name
'GSKGCATGTGAAA'
>>> motif.id
'motif_1'
```

Yuxarıda etdiyimiz kimi qeyddə indeksdən istifadə etməklə yanaşı, onu adı ilə də tapa bilərsiniz:

```
>>> motiv = qeyd["GSKGCATGTGAAA"]
```

Hər bir motivin atributuna malikdir. motivin tapıldığı ardıcılıqla düzülərək, ardıcılığın hər biri haqqında bəzi məlumat verir:

```
>>> len(motif.alignment) 7 >>>

motif.alignment.sequences[0]
Nümunə ('GCGGCATGTGAAA') >>>
motif.alignment.sequences[0].motif_name 'GSKGCATGTGAAA' >>>

motif.alignment.sequences[0].sequence_name 'INO1' >>>

motif.alignment.sequences[0].sequences >> _ 5 motiv.alignment.sequences[0].start
```

620

```
>>> motif.alignment.sequences[0].strand
```

```
'+'

>>> motif.alignment.sequences[0].uzunluq

13

>>> pvalue = motif.alignment.sequences[0].pvalue

>>> çap ("%.3g" % pvalue)

1.21e-08
```

MAST

### 17.2.3 TRANSFAC

TRANSFAC transkripsiya faktorlarının genomik bağlanması ile birlikdə ə l ilə seçilmiş verilə nlə r bazasıdır saytlar və DNT bağlama profillə ri [32]. TRANSFAC verilə nlə r bazasında istifadə olunan fayl formatı indiki vaxtda olduğu halda başqaları tə rə fində n də istifadə olunarsa, biz ona TRANSFAC fayl formatı kimi istinad edə cə yik.

TRANSFAC formatında minimal fayl aşağıdakı kimi görünür:

ID motivi 1

| P0 | A | C | G | T |   |
|----|---|---|---|---|---|
| 01 | 1 | 2 | 2 | 0 | S |
| 02 | 2 | 1 | 2 | 0 | R |
| 03 | 3 | 0 | 1 | 1 | A |
| 04 | 0 | 5 | 0 | 0 | C |
| 05 | 5 | 0 | 0 | 0 | A |
| 06 | 0 | 0 | 4 | 1 | G |
| 07 | 0 | 1 | 4 | 0 | G |
| 08 | 0 | 0 | 0 | 5 | T |
| 09 | 0 | 0 | 5 | 0 | G |
| 10 | 0 | 1 | 2 | 2 | K |
| 11 | 0 | 2 | 0 | 3 | Y |
| 12 | 1 | 0 | 3 | 1 | G |
| // |   |   |   |   |   |

Bu fayl 12 nukleotiddə n ibarə t motiv motivinin1 tezlik matrisini göstə rir. Ümumiyyə tlə , TRANSFAC-da bir fayl formatda çoxlu motivlə r ola bilə r. Mə sə lə n, bu, nümunə TRANSFAC faylinin mə zmunudur transfac.dat:

VV NÜMUNƏ Sİ 15 yanvar 2013-cü il  
XX

//

ID motivi 1

| P0  | A | C | G | T |   |
|-----|---|---|---|---|---|
| 01  | 1 | 2 | 2 | 0 | S |
| 02  | 2 | 1 | 2 | 0 | R |
| 03  | 3 | 0 | 1 | 1 | A |
| ... |   |   |   |   |   |
| 11  | 0 | 2 | 0 | 3 | Y |
| 12  | 1 | 0 | 3 | 1 | G |
| //  |   |   |   |   |   |

ID motivi 2

| P0  | A | C | G | T |   |
|-----|---|---|---|---|---|
| 01  | 2 | 1 | 2 | 0 | R |
| 02  | 1 | 2 | 2 | 0 | S |
| ... |   |   |   |   |   |

|    |   |   |   |   |   |
|----|---|---|---|---|---|
| 09 | 0 | 0 | 0 | 5 | T |
|    | 0 | 2 | 0 | 3 | Y |

10 //

TRANSFAC faylinı tə hlil etmə k üçün istifadə edin

```
>>> tutacaq olaraq open("transfac.dat") ilə :
...
... qeyd = motivlə r.parse(tutacaq, "TRANSFAC")
...
```

Faylin mə zmunu ilə TRANSFAC fayl formatı arasında hə r hansı uyğunsuzluq aşkar edilə rsə , ValueError qaldırıcı. Nə zə rə alın ki, siz TRANSFAC formatına ciddi ə mə l etmə yə n fayllarla qarşılaşa bilə rsiniz. Mə sə lə n, sütunlar arasındaki boşluqların sayı fə rqli ola bilə r və ya boşluq yerinə nişan istifadə edilə bilə r. istifadə edin strict=ValueError-u yüksə ltmə də n belə faylların tə hlilini aktivlə şdirmə k üçün False:

```
>>> qeyd = motifs.parse(tutacaq, "TRANSFAC", strict=False)
```

Uyğun olmayan faylı tə hlil edə rkə n, ə min olmaq üçün motif.parse tə rə fində n qaytarılan qeydi yoxlamağı tövsiyə edirik. faylin mə zmununa uyğundur.

Ümumi versiya nömrə si, ə gə r varsa, record.version kimi saxlanılır:

```
>>> rekord.versiya
'NÜMUNƏ 15 yanvar 2013'
```

Qeyddə olan hə r bir motiv Bio.motifs.transfac.Motif sinifinin nümunə sidir və hə r ikisini miras alır. Bio.motifs.Motif sınıfı və Python lüğə tində n. Lüğə t saxlamaq üçün iki hə rfli düymə lə rdə n istifadə edir motiv haqqında hə r hansı ə lavə mə lumat:

```
>>> motiv = qeyd[0]
>>> motif.degenerate_consensus # Bio.motifs.Motif xüsusiyyə tində n istifadə
Seq('SRACAGGTGKYG')
>>> motiv["ID"] # Motifdə n lüğə t kimi istifadə
'motif1'
```

TRANSFAC faylları adə tə n bu misaldan daha müə kkə bdir və çoxlu ə lavə lə ri ehtiva edir motiv haqqında mə lumat. Cə dvə l 17.2.3-də ümumiyyə tlə rast gə linə n iki hə rfli sahə kodları verilmişdir TRANSFAC faylları:

Hə r motivin də bir atributu var. İstifadə motivi ilə ə laqə li istinadları ehtiva edə n istinadlar bu iki hə rfli açarlar:

Motivlə rin çapı onları öz doğma TRANSFAC formatında yazır:

```
>>> çap (qeyd)
VV NÜMUNƏ Sİ 15 yanvar 2013-cü il
XX
//
ID motivi 1
XX
P0 A C G T
01 1 2 2 0 S
02 2 1 2 0 R
03 3 0 1 1 A
04 0 5 0 0 C
05 5 0 0 0 A
06 0 0 4 1 G
```

|                                                                |
|----------------------------------------------------------------|
| Cədvəl 17.1: TRANSFAC fayllarında tez-tez rast gəlinən sahələr |
| AC qoşulma nömrəsi                                             |
| AS Qoşulma nömrələri, ikinci dərəcəli                          |
| BA Statistikəsas                                               |
| BF Bağlayıcı amillər                                           |
| BS Matrisin altında yatan faktor bağlama yerləri               |
| CC şərhləri                                                    |
| CO Müəllif hüquqları haqqında bildiriş                         |
| DE Qısa faktor təsviri                                         |
| DR Xarici verilənlər bazası                                    |
| DT Tarixi yaradıldı/yeniləndi                                  |
| HC alt ailələri                                                |
| HP Super ailələri                                              |
| ID identifikatoru                                              |
| NA Bağlayıcı amilin adı                                        |
| OC taksonomik təsnifikasi                                      |
| OS növləri/Takson                                              |
| OV Kōhnə versiya                                               |
| PV-yə üstünlük verilən versiya                                 |
| TY Növü                                                        |
| XX boş xətti; bunlar Qeyddə saxlanılır.                        |

Cədvəl 17.2: TRANSFAC fayllarında istinadları saxlamaq üçün istifadə olunan sahələr

|                         |
|-------------------------|
| RN İstinad nömrəsi      |
| RA Referans müəllifləri |
| RL İstinad məlumatları  |
| RT İstinad başlığı      |
| RX PubMed ID            |

|    |   |   |   |   |   |
|----|---|---|---|---|---|
| 07 | 0 | 1 | 4 | 0 | G |
| 08 | 0 | 0 | 0 | 5 | T |
| 09 | 0 | 0 | 5 | 0 | G |
| 10 | 0 | 1 | 2 | 2 | K |
| 11 | 0 | 2 | 0 | 3 | Y |
| 12 | 1 | 0 | 3 | 1 | G |

XX

//

ID motivi 2

XX

|    |   |   |   |   |   |
|----|---|---|---|---|---|
| P0 | A | C | G | T |   |
| 01 | 2 | 1 | 2 | 0 | R |
| 02 | 1 | 2 | 2 | 0 | S |
| 03 | 0 | 5 | 0 | 0 | C |
| 04 | 3 | 0 | 1 | 1 | A |
| 05 | 0 | 0 | 4 | 1 | G |
| 06 | 5 | 0 | 0 | 0 | A |
| 07 | 0 | 1 | 4 | 0 | G |
| 08 | 0 | 0 | 5 | 0 | G |

```

09 0 0 0 5 T
10 0 2 0 3 Y
XX
//
<BLANKLINE>

```

Siz motivlə ri TRANSFAC formatında ixrac edə bilə rsiniz.  
bir fayl:

```

>>> mə tn = str (qeyd)
>>> out_handle kimi open("mytransfacfile.dat", "w") ilə :
... out_handle.write(mə tn)
...

```

## 17.3 Yazı motivlə ri

İxracdan söz düşmüşkə n, ümumi olaraq ixrac funksiyalarına nə zə r salaq. Biz formatın daxili funksiyasından istifadə edə bilə rik motivi sadə JASPAR pfm formatında yazın:

```

>>> çap (format(arnt, "pfm"))
4.00 19.00 0.00 0.00 0.00 0.00
16.00 0.00 20.00 0.00 0.00 0.00
0,00 1,00 0,00 20,00 0,00 20,00
0,00 0,00 0,00 0,00 20,00 0,00

```

Eynilə , JASPAR jaspar formatında motivi yazmaq üçün formatdan istifadə edə bilə rik:

```

>>> çap (format(arnt, "jaspar"))
>MA0004.1 Arnt
A [4.00 19.00 0.00 0.00 0.00]
C [16.00 0.00 20.00 0.00 0.00]
G [0,00 1,00 0,00 20,00 0,00 20,00]
T [0,00 0,00 0,00 0,00 20,00 0,00]

```

Motivi TRANSFAC kimi matris formatında yazmaq üçün istifadə edin

```

>>> çap (format(m, "transfac"))
P0 A C G T
01 3 0 0 4 w
02 7 0 0 0 A
03 0 5 0 2 C
04 2 2 3 0 V
05 1 6 0 0 C
XX
//
<BLANKLINE>

```

Çoxlu motivlə ri yazmaq üçün motifs.write istifadə edə bilə rsiniz. Bu funksiyadan asılı olmayaraq istifadə edilə bilə r motivlə r TRANSFAC faylından yaranmışdır. Mə sa lə n,

```

>>> iki_motif = [arnt, srf]
>>> çap (motifs.write(iki_motif, "transfac"))
P0 G 01 0 A T
 4 C 16 0 C

```

```

02 19 1 0 A
03 0 0 20 0 0 C
04 0 0 20 0 G
05 0 0 20 T
06 0 0 20 0 G
XX
//

P0 A C G T
01 2 1 39 4 G
02 9 33 2 2 C
03 0 45 1 0 C
04 1 45 0 0 C
05 32 1 0 13 A
06 3 1 0 42 T
07 46 0 0 0 A
08 1 0 0 45 T
09 43 0 0 3 A
10 15 1 0 30 w
11 2 0 44 0 G
12 2 1 43 0 G
XX
//

<BLANKLINE>

```

Və ya jaspar formatında çoxlu motivlə r yazmaq üçün:

```

>>> iki_motif = [arnt, mef2a]
>>> çap(motifs.write(iki_motif, "jaspar"))
>MA0004.1 Arnt
A [4.00 19.00 0.00 0.00 0.00]
C [16.00 0.00 20.00 0.00 0.00]
G [0,00 1,00 0,00 20,00 0,00 20,00]
T [0,00 0,00 0,00 0,00 20,00 0,00]
>MA0052.1 MEF2A
A [1,00 0,00 57,00 2,00 9,00 6,00 37,00 2,00 56,00 6,00]
C [50,00 0,00 1,00 1,00 0,00 0,00 0,00 0,00 0,00 0,00 0,00]
G [0,00 0,00 0,00 0,00 0,00 0,00 0,00 0,00 2,00 50,00]
T [7,00 58,00 0,00 55,00 49,00 52,00 21,00 56,00 0,00 2,00]

```

## 17.4 Mövqe-Çə ki Matrislə ri

Motif obyektinin .counts atributu hə r nukleotidin hə r mövqedə nə qə də r tez-tez göründüğünü göstə rir hizalanma. Bu matrisi düzülmə də ki misalların sayına bölmə klə normallaşdırı bilə rik, nə ticə də hizalanma boyunca hə r mövqedə hə r bir nukleotidin ehtimalında. Biz bu ehtimallara istinad edirik mövqe-çə ki matrisi. Ancaq diqqə tli olun ki, ə də biyyatda bu terminə istinad etmə k üçün də istifadə oluna bilə r aşağıda müzakirə etdiyimiz mövqeyə xas qol matrisi.

Adə tə n, normallaşmadan ə vvə l hə r mövqeyə psevdocountlar ə lavə edilir. Bu, hə ddində n artıq yükə nmə nin qarşısını alır Mövqe-agırılıq matrisinin hizalanmasında mə hdud sayda motiv nümunə lə rinin qarşısını da ala bilə r ehtimalların sıfıra çevrilmə si. Bütün mövqelə rdə ki bütün nukleotidlə rə sabit psevdocount ə lavə etmə k üçün a Pseudocounts arqumenti üçün nömrə :

```

>>> pwm = m.counts.normalize(psevdocounts=0,5)
>>> çap(pwm)

```

|   |   |   |   |
|---|---|---|---|
| 1 | 2 | 3 | 4 |
|---|---|---|---|

A: 0,39 0,83 0,06 0,28 0,17  
 C: 0,06 0,06 0,61 0,28 0,72  
 G: 0,06 0,06 0,06 0,39 0,06  
 T: 0,50 0,06 0,28 0,06 0,06  
 <BLANKLINE>

Alternativ olaraq, pseudocounts hər bir nukleotid üçün psevdocountları göstərirənlügət ola bilər. Məsələn, insan genomunun GC tərkibi təqribən 40% olduğu üçün psevdocountları seçmək istəyəbilərsiniz.  
 müvafiq olaraq:

```
>>> pwm = m.counts.normalize(pseudocounts={"A": 0,6, "C": 0,4, "G": 0,4, "T": 0,6})
```

çap (pwm)

|   |   |   |   |   |
|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|

A: 0,40 0,84 0,07 0,29 0,18  
 C: 0,04 0,04 0,60 0,27 0,71  
 G: 0,04 0,04 0,04 0,38 0,04  
 T: 0,51 0,07 0,29 0,07 0,07  
 <BLANKLINE>

Mövqe çəki matrisinin konsensus, antikonsensus və degenerasiya hesablanması üçün öz üsulları var.  
 konsensus ardıcılığı:

```
>>> pwm.consensus
Seq('TACGC')
>>> pwm.anticonsensus
Seq('CCGTG')
>>> pwm.degenerate_consensus
Seq('WACNC')
```

Qeyd edək ki, psevdosaymala görə pozulmuş konsensus ardıcılığı mövqe çəkisindən hesablanır.  
 matris motivdəki nümunələrdən hesablanmış degenerativ konsensus ardıcılığından bir qədərfərqlidir:

```
>>> m.degenerate_consensus
Seq('WACVC')
```

Mövqe-çəki matrisininəks tamamlayıcısı birbaşa pwm-dən hesablanır:

```
>>> rpwm = pwm.reverse_complement()
>>> çap (rpwm)
0 1 2 3 4
A: 0,07 0,07 0,29 0,07 0,51
C: 0,04 0,38 0,04 0,04 0,04
G: 0,71 0,27 0,60 0,04 0,04
T: 0,18 0,29 0,07 0,84 0,40
<BLANKLINE>
```

## 17.5 Mövqe Xüsusi Qiymətləndirmə Matrisləri

Arxa plan paylanması və PWM-dən istifadə edərək psevdosaymalarəlavə olunmaqla, log-əməlləri hesablamaq asandır. Nisbətləri, bizə qarşı bir motivdən gələn müəyyən bir simvolun log ehtimallarının nə olduğunu izah edir. Fon. Mövqe-çəki matrisində .log\_odds() metodundan istifadə edə bilərik:

```
>>> pssm = pwm.log_odds() >>> çap (pssm) 1

0 2 3 4

A: 0,68 1,76 -1,91 0,21 -0,49
C: -2,49 -2,49 1,26 0,09 1,51
G: -2,49 -2,49 -2,49 0,60 -2,49
T: 1,03 -1,91 0,21 -1,91 -1,91
<BLANKLINE>
```

Burada biz motivdə fonda olduğundan daha tez-tez simvollar üçün müsbət də ya rələri və fonda daha tez-tez simvollar üçün mənfi də ya rələri görə bilərik. 0.0 o deməkdir ki, onun fonda və motivdə simvolu görmə ehtimalı eynidir.

Bu, A, C, G və T-nin arxa planda eyni dərəcədə ehtimal olunduğunu nəzərdə tutur. A, C, G, T üçün qeyri-bərabər ehtimallara malik fonda mövqeyə aid xal matrisini hesablamaq üçün fon arqumentindən istifadə edin.

Məsələn, 40% GC məzmunu olan bir fonda istifadə edin

```
>>> fon = {"A": 0,3, "C": 0,2, "G": 0,2, "T": 0,3} >>> pssm = pwm.log_odds(fon) >>> print(pssm) 0 1 4 A:
0,42 1,49 -2,17 -0,17 -0,12:0,05 1,58 0,42 1,83 G: -2,17 -2,17 -2,17
0,92 -2,17 T: 0,77 -2,17
-0,05 -2,17 -2,17 2 3
```

<BLANKLINE>

PSSM-dənəldə edilən maksimum və minimum xal .max və .min xassələrinde saxlanılır:

```
>>> çap ("%.4.2f" % pssm.max) 6.59
>>> çap ("%.4.2f" % pssm.dəq) -10.85
```

Müəyyən bir fonla bağlı PSSM ballarının orta və standart sapması hesablanır  
.mean və .std üsulları ilə.

```
>>> orta = pssm.mean(fon) >>> std =
pssm.std(fon) >>> print("orta = %.2f, standart
kənarlaşma = %.2f" % (orta, std)) orta = 3.21, standart kənarlaşma = 2.59
```

Fon müəyyən edilmə dikdən vahid fon istifadə olunur. Orta dəyər Bölmə 17.1.5-dən təsvir edilən Kullback-Leibler divergensiyasına və ya nisbi entropiyaya bərabərdir.

.reverse\_complement, .consensus, .anticonsensus və .degenerate\_consensus üsulları birbaşa PSSM obyektlərinə tətbiq oluna bilər.

## 17.6 Nümunələrin axtarışı

Motif üçünən çox istifadə edilən nümunələri müəyyən ardıcılıqla tapmaqdır. Bu bölmənin xatirinə biz belə bir süni ardıcılıqlıdan istifadə edəcəyik:

```
>>> test_seq = Seq("TACACTGCATTACAACCCAAGCATTA") >>> len(test_seq) 26
```

### 17.6.1 Də qıq uyğunluqların axtarışı

Nümunə lə ri tapmağın ə n sadə yolu, motivin ə sl nümunə lə rinin də qıq uyğunluqlarını axtarmaqdır:

```
>>> pos üçün , test_seq.search(m.alignment) bölmə sində ardıcılıq:
... çap ("%i %s" % (pos, seq))
...
0 TACAC
10 TACAA
13 AACCC
```

Eyni şeyi ə ks tamamlayıcı ilə də edə bilə rik (tamamlayıcı zə ncirdə nümunə lə ri tapmaq üçün):

```
>>> pos üçün , test_seq.search(r.alignment) bölmə sində ardıcılıq:
... çap ("%i %s" % (pos, seq))
...
6 GCATT
20 GCATT
```

### 17.6.2 PSSM xalından istifadə edə rə k uyğunluqların axtarışı

Mövqelə ri axtarmaq eyni də rə cə də asandır, motivimizə qarşı yüksə k log-ə mə lli xallara sə bə b olur:

```
>>> mövqe üçün , pssm.search -də xal (test_seq, eşik=3.0): print("Mövqe %d: xal =
... %5.3f" % (mövqe, xal))
...
Mövqe 0: xal = 5,622
Mövqe -20: xal = 4,601
Mövqe 10: xal = 3,037
Mövqe 13: xal = 5,738
Mövqe -6: xal = 4,601
```

Mə nfi mövqelə r test ardıcılığının ə ks zolağında təpilən motiv nümunə lə rinə istinad edir və mə nfi indekslə r üzrə Python konvensiyasına ə mə l edir. Buna görə də , pos-dakı motiv nümunə si pos-un hə m müsbə t, hə m də mə nfi qiymə tlə ri üçün test\_seq[pos:pos+len(m)]-də yerlə şir.

Burada ixtiyari olaraq 3.0-a tə yin edilmiş eşik parametрini görə bilə rsiniz. Bu, log2-də dir, ona görə də biz indi yalnız motiv modelində arxa fonda olduğundan sə kkiz də fə çox olan sözlə ri axtarıraq.

Defolt hə ddi 0.0-dır və bu, fondan daha çox motivə bə nzə yə n hə r şeyi seçir.

Siz hə mçinin ardıcılıqla bütün mövqelə rdə xalları hesablaya bilə rsiniz:

```
>>> pssm.calculate(test_seq) massivi([5.62230396,
-5.6796999 , -3.43177247, 0.93827754, -6.84962511, -2.04066086, -69, -106. -3.65614533, -0.03370807,
-3,91102552, 3,03734159, -2,14918518, -0,6016975 5,7381525 , -0,50977498, -0,50977494, -32,3,
-283,385 -0,09919716, -0,6016975 , -2,39429784, -10,84962463, -3,65614533], dtype=float32)
```

Ümumiyyə tlə , bu, PSSM ballarını hesablamaq üçün ə n sürə tli yoldur. Pssm.calculate tə rə fində n qaytarılan ballar yalnız irə li strand üçündür. Ə ks strand üzrə xalları ə ldə etmə k üçün PSSM-nin ə ks tamamlayıcısını götürə bilə rsiniz:

```
>>> rpssm = pssm.reverse_complement() >>
rpssm.calculate(test_seq)
```

```
massiv([-9.43458748, -3.06172252, -7.18665981, -7.76216221, -2.04066086, -4.26466274,
 4.60124254, -4.248034, -2.26503372, -6.49598789, -5.64668512, -8.73414803, ,
 -10.84962463, -4.82356262, -4.82356262, -5.64668570, -5.64668513, -
 -4.15613794, -5.6796999, 4.60124254, -4.2480607], dtype=float32)
```

### 17.6.3 Hesab hə ddinin seçilmə si

Ə gə r hə dlə ri seçmə k üçün daha az ixtiyari üsuldan istifadə etmə k istə yırsınızsa , PSSM ballarının paylanmasıni araşdırı bilə rsiniz. Hesabin paylanması üçün yer motiv uzunluğu ilə eksponent olaraq artdığından, hesablama xə rclə rini idarə olunan saxlamaq üçün verilmiş də qıqlıklə tə xmini istifadə edirik:

```
>>> paylama = pssm.distribution(fon=fon, də qıqlıq=10**4)
```

Paylanma obyekti bir sıra müxtə lif hə dlə ri müə yyə n etmə k üçün istifadə edilə bilə r. Biz tə lə b olunan yalan-müsəbə t də rə cə sini tə yin edə bilə rik (arxa fonda yaradılan ardıcılılıqda motiv nümunə sinin "tapılması" ehtimalı):

```
>>> hə ddi = paylama.threshold_fpr(0.01) >>> print("%5.3f" % hə ddi) 4.009
```

və ya yalan-mə nfi nisbə t (motivdə n yaranan nümunə nin "tapılmaması" ehtimalı):

```
>>> hə ddi = paylama.threshold_fnr(0.1) >>> print("%5.3f" % hə ddi) -0.510
```

və ya yalan-müsəbə t də rə cə ilə yanlış-mə nfi nisbə t (fnr t) arasında müə yyə n ə laqə ni tə min edə n hə dd (tə xminə n):

```
>>> hə ddi = paylama.hə ddi_balanced(1000) >>> print("%5.3f" % hə ddi) 6.241
```

və ya yalan-müsəbə t nisbə tin loqu ilə informasiya mə zmunu arasındaki bə rabə rliyi (tə xminə n) tə min edə n hə dd (Hertz və Stormo tə rə fində n patser proqramında istifadə edildiyi kimi):

```
>>> hə ddi = paylama.threshold_patser() >>> print("%5.3f" % hə ddi) 0.346
```

Mə sə lə n, bizim motivimizdə sızə eyni nə tica lə ri verə n hə ddi ə ldə edə bilə rsiniz (bunun üçün ardıcılılığı) 1000 də rə cə si ilə balanslaşdırılmış hə ddi olan nümunə lə rin axtarışı kimi.

```
>>> hə ddi = paylama.threshold_fpr(0.01) >>> print("%5.3f" % hə ddi) 4.009
```

```
>>> mövqe üçün , pssm.search -də xal (test_seq, ə rə fə =threshold): print("Mövqe %d: xal = %5.3f" %
... (mövqe, xal))
...
Mövqe 0: xal = 5,622
Mövqe -20: xal = 4,601 Mövqe 13: xal = 5,738
Mövqe -6: xal = 4,601
```

## 17.7 Hə r motiv obyektinin ə laqə li Mövqe Xüsusi Qiymə tlə ndirmə si var Matris

PSSM-lə rdə n istifadə edə rə k potensial TFBS-lə rin axtarışını asanlaşdırmaq üçün hə m mövqe çə kisi matrisi, hə m də mövqeyə xas xal matrisi hə r bir motivlə ə laqə lə ndirilir. Arnt motivində n nümunə kimi istifadə :

```
>>> Bio idxal motivlə rinda n
>>> qulp kimi open("Arnt.sites") ilə :
...
... motiv = motifs.read(tutacaq, "saytlar")
...
...
>>> çap (motif.sayilar)
 0 1 2 3 4 5
A: 4.00 19.00 0.00 0.00 0.00 0.00
C: 16.00 0.00 20.00 0.00 0.00 0.00
G: 0,00 1,00 0,00 20,00 0,00 20,00
T: 0,00 0,00 0,00 0,00 20,00 0,00
<BLANKLINE>
>>> çap (motif.pwm)
 0 1 2 5 3 4
A: 0,20 0,95 0,00 0,00 0,00 0,00
C: 0,80 0,00 1,00 0,00 0,00 0,00
G: 0,00 0,05 0,00 1,00 0,00 1,00
T: 0,00 0,00 0,00 0,00 1,00 0,00
<BLANKLINE>
>>> çap (motif.pssm)
 0 1 2 3 4 5
A: -0,32 1,93 -inf -inf -inf -inf
C: 1,68 -inf 2,00 -inf -inf -inf
G: -inf -2,32 -inf 2,00 -inf 2,00
T: -inf -inf -inf -inf 2,00 -inf
<BLANKLINE>
```

Mə nfi sonsuzluqlar burada görünür, çünkü tezlik matrisində müvafiq giriş 0-dır, biz isə defolt olaraq sıfır psevdocountlardan istifadə :

```
>>> "ACGT" -də mə ktub üçün :
...
... çap("%s: %4.2f" % (mə ktub, motiv.pseudocounts[letter]))
...
A: 0.00
C: 0.00
G: 0.00
T: 0.00
```

Ə gə r siz .pseudocounts atributunu, mövqe tezliyi matrisini və mövqeyə xas xalları də yişsə niz matris avtomatik olaraq yenidə n hesablanır:

```
>>> motif.pseudocounts = 3.0
>>> "ACGT" -də mə ktub üçün :
...
... çap("%s: %4.2f" % (mə ktub, motiv.pseudocounts[letter]))
...
A: 3.00
C: 3.00
```

G: 3.00  
T: 3.00  
**>>> çap (motif.pwm)**

| 0       | 1    | 2    | 3    | 4    | 5    |
|---------|------|------|------|------|------|
| A: 0,22 | 0,69 | 0,09 | 0,09 | 0,09 | 0,09 |
| C: 0,59 | 0,09 | 0,72 | 0,09 | 0,09 | 0,09 |
| G: 0,09 | 0,12 | 0,09 | 0,72 | 0,09 | 0,72 |
| T: 0,09 | 0,09 | 0,09 | 0,09 | 0,72 | 0,09 |

<BLANKLINE>

**>>> çap (motif.pssm)**

| 0        | 1     | 2     | 3     | 4     | 5     |
|----------|-------|-------|-------|-------|-------|
| A: -0,19 | 1,46  | -1,42 | -1,42 | -1,42 | -1,42 |
| C: 1,25  | -1,42 | 1,52  | -1,42 | -1,42 | -1,42 |
| G: -1,42 | -1,00 | -1,42 | 1,52  | -1,42 | 1,52  |
| T: -1,42 | -1,42 | -1,42 | -1,42 | 1,52  | -1,42 |

<BLANKLINE>

Fə rqlı istifadə etmə k istə yirsinizsə , siz hə mçinin dörd nukleotid üzə rində lügə tə .pseudocounts tə yin edə bilə rsiniz. onlar üçün psevdocountlar. motif.pseudocounts-un Yox-a tə yin edilmə si onu standart sıfır də yə rinə qaytarır.

Mövqe üçün xüsusi bal matrisi defolt olaraq vahid olan fon paylanmasından asılıdır:

```
>>> "ACGT" -də mə ktub üçün :
...
... çap("%s: %4.2f" % (mə ktub, motiv.fon[mə ktub]))
...
A: 0,25
C: 0,25
G: 0,25
T: 0,25
```

Yenə də , fon paylanmasından yışdırırsə niz, mövqeyə xas xal matrisi yenidə n hesablanır:

```
>>> motif.background = {"A": 0,2, "C": 0,3, "G": 0,3, "T": 0,2}
>>> çap (motif.pssm)
```

| 0        | 1     | 2     | 3     | 4     | 5     |
|----------|-------|-------|-------|-------|-------|
| A: 0,13  | 1,78  | -1,09 | -1,09 | -1,09 | -1,09 |
| C: 0,98  | -1,68 | 1,26  | -1,68 | -1,68 | -1,68 |
| G: -1,68 | -1,26 | -1,68 | 1,26  | -1,68 | 1,26  |
| T: -1,09 | -1,09 | -1,09 | -1,09 | 1,85  | -1,09 |

<BLANKLINE>

motif.background-un Heç biri kimi tə yin edilmə si onu vahid paylanması sıfırlayır:

```
>>> motif.background = Yoxdur
>>> "ACGT" -də mə ktub üçün :
...
... çap("%s: %4.2f" % (mə ktub, motiv.fon[mə ktub]))
...
A: 0,25
C: 0,25
G: 0,25
T: 0,25
```

Ə gə r motif.background-u tə k də yə rə bə rabə r tə yin etsə niz, o, GC mə zmunu kimi şə rh olunacaq:

```
>>> motif.background = 0,8 >>> "ACGT" hə rf
üçün :
...
çap("%s: %4.2f" % (mə ktub, motiv.fon[mə ktub]))
...
A: 0.10
C: 0.40
G: 0.40
T: 0.10
```

Qeyd edə k ki, indi PSSM ballarının hesablandığı fonda ortasını hesablaya bilə rsiniz:

```
>>> çap("%f" % motif.pssm.mean(motif.fon)) 4.703928
```

elə cə də onun standart sapması:

```
>>> çap ("%f" % motif.pssm.std(motif.fon)) 3.290900
```

və onun paylanması:

```
>>> paylama = motif.pssm.distribution(background=motif.background) >>> hə ddi = paylama.threshold_fpr(0.01) >>> print("%f"
% hə ddi) 3.854375
```

Nə zə rə alın ki, hə r də fə motif.pwm və ya motif.pssm-ə zə ng etdikdə mövqeq çə kisi matrisi və mövqeyə xas xal matrisi yenidə n hesablanır. Sürət problemdirə və siz PWM və ya PSSM-də n də fə lə rlə istifadə etmə k istə yırsınızsa , bunları aşağıda olduğu kimi də yişə n kimi saxlaya bilə rsiniz.

```
>>> pssm = motiv.pssm
```

## 17.8 Motivlə rin müqayisə si

Birdə n çox motivimiz olduqda, onları müqayisə etmə k istə yə bilə rik.

Motiflə ri müqayisə etmə yə başlamazdan ə vvə l qeyd etmə liyə m ki, motiv sə rhə dlə ri adə tə n olduqca ixtiyari olur. Bu o demə kdir ki, biz tez-tez müxtə lif uzunluqdakı motivlə ri müqayisə etmə li oluruq, buna görə də müqayisə bir növ uyğunlaşdırmanı ə hatə etmə lidir. Bu o demə kdir ki, biz iki şeyi nə zə rə almaliyiq:

- motivlə rin uyğunlaşdırılması
- düzülmüş motivlə ri müqayisə etmə k üçün bə zi funksiyalar

Motiflə ri uyğunlaşdırmaq üçün biz PSSM-lə rin boşluqsuz düzülüşündə n istifadə edirik və matrislə rin ə vvə lində və sonunda çatışmayan sütunlar üçün sıfırları ə və z edirik. Bu o demə kdir ki, biz PSSM-də çatışmayan sütunlar üçün fon paylanmasından effektiv şə kildə istifadə edirik. Mə safə funksiyası sonra motivlə r arasındaki minimal mə safə ni, elə cə də onların düzülüşündə müvafiq ofseti qaytarır.

Nümunə vermə k üçün ə vvə lə test motivimiz m-ə bə nza yə n başqa bir motivi yükə yə k:

```
>>> qulp kimi open("REB1.pfm") ilə : m_reb1 =
...
motifs.read(tutacaq, "pfm")
...
>>> m_reb1.consensus
```

```

Seq('GTTACCCGG')
>>> çap(m_reb1.counts)
0 1 2 3 4 5 6 7 8
A: 30,00 0,00 0,00 100,00 0,00 0,00 0,00 15,00
C: 10,00 0,00 0,00 0,00 100,00 100,00 0,00 15,00
G: 50,00 0,00 0,00 0,00 0,00 60,00 55,00
T: 10,00 100,00 100,00 0,00 0,00 0,00 40,00 15,00
<BLANKLINE>

```

Motiflə ri müqayisə etmək üçün biz psevdocountlar və fon üçün eyni dəyişərləri seçirik  
bizim motivimiz kimi paylamaq:

```

>>> m_reb1.pseudocounts = {"A": 0,6, "C": 0,4, "G": 0,4, "T": 0,6}
>>> m_reb1.background = {"A": 0,3, "C": 0,2, "G": 0,2, "T": 0,3}
>>> pssm_reb1 = m_reb1.pssm
>>> çap(pssm_reb1)
0 1 2 3 4 5 6 7 8
A: 0,00 -5,67 -5,67 1,72 -5,67 -5,67 -5,67 -0,97
C: -0,97 -5,67 -5,67 -5,67 2,30 2,30 2,30 -5,67 -0,41
G: 1,30 -5,67 -5,67 -5,67 -5,67 -5,67 1,57 1,44
T: -1,53 1,72 1,72 -5,67 -5,67 -5,67 0,41 -0,97
<BLANKLINE>

```

Pearson korrelyasiyasından istifadə edərək bu motivləri müqayisə edəcəyik. Onun məsafə ölçüsünə bənzəməsinə istədiyimiz üçün,  
bizə slində 1 rəsmini, burada Pearson korrelyasiya məsaldır (PCC):

```

>>> məsafə, offset = pssm.dist_pearson(pssm_reb1)
>>> print("məsafə = %5.3g" % məsafə)
məsafə = 0,239
>>> çap(offset)
-2

```

Bu o deməkdir ki, motiv məsafə m\_reb1 arasında nümayiş PCC aşağıdakı düzülmə ilə əldə edilir:

```

m: bbTACGCbb
m_reb1: GTTACCCGG

```

burada b fon paylaşılması deməkdir. PCC-nin özü təxminən  $1 - 0,239 = 0,761$ -dir.

## 17.9 Yeni motivin tapılması

Hal-hazırda, Biopython de novo motiv tapmaq üçün yalnız məhdud dəstəyə malikdir. Daha doğrusu, qəçməğidən sonra kləyirik  
xxmotif və həmçinin MEME-nin təhlili. Motif tapmaq vasitəsi rəsminin sayı sürətlə artlığından töhfələr  
yeni təhlilcılər xoş gəlmisiniz.

### 17.9.1 MEME

Tutaq ki, siz MEME-ni sevdiyiniz parametrlərlə seçdiyiniz ardıcılıqla işlətmisiniz və yadda saxlamışınız.  
meme.out faylında çıkış. MEME tərəfindən bildirilən motivləri aşağıdakı parçanı işlətməklə əldə edə bilərsiniz  
kodu:

```

>>> Bio.idxal motivlərinde
>>> dəstəyi kimi open("meme.psp_test.classic.zoops.xml") ilə :
... motivlərM = motivlər.parse(tutacaq, "meme")
...

```

```
>>> motifsM
```

```
[<Bio.motifs.meme.Motif obyekti 0xc356b0>]
```

Ən çox axtarılan motivlər siyahısından başqa, nəticə obyekti əlçatan olan daha faydalı məlumatları ehtiva edir. Özünü izah edən adlarla xassələr vasitəsilə :

- .ə\_lifba
- .datafile
- .ardicilliqlar
- .versiya
- .command

MEME Parser tərəfindən qaytarılan motivlərin adı Motif obyektləri kimi işlənə bilər (instansiyalarla), onlar həmçinin nümunələrlə haqqında əlavə məlumat əlavə etməklə bəzi əlavə funksionallığı təmin edir.

```
>>> motifsM[0].consensus Seq('GCTTATGTAA')
```

```
>>>
```

```
motifsM[0].alignment.sequences[0].sequence_name 'iYFL005W' >>>
```

```
motifsM[0].alignment.sequences[0].sequence_id 'sequence_15' >>>
```

```
motifsM[0].alignment.sequences[0].start 480 >>>
```

```
motifsM[0].alignment.sequences[0].strand
'+'
```

```
>>> motifsM[0].alignment.sequences[0].pvalue
1.97e-06
```

## 17.10 Faydalı bağlantılar

- [Ardıcılıq motivi](#) vikipediyyada
- [PWM](#) vikipediyyada
- [Konsensus ardıcılığı](#) vikipediyyada
- [Müxtəlif motiv tapmaq programlarının müqayisəsi](#)

## Fəsil 18

# Klaster təhlili

Klaster təhlili, maddə lərin bir-birinə oxşarlığına əsaslanaraq, klasterlərə qruplaşdırılmasıdır.

Bioinformatikada klasterləşmə oxşar gen ifadə profillərinə malik gen qruplarını tapmaq üçün gen ifadəsi məlumatlarının təhlilində geniş istifadə olunur. Bu, funksional olaraq əlaqəli genləri müəyyən edə bilər, həmçinin hazırda naməlum genlərin funksiyasını təklif edə bilər.

Biopython modulu Bio.Cluster tez-tez istifadə olunan klasterləşdirmə alqoritmlərini təmin edir və gen ifadə məlumatlarına tətbiqi nəzərə alınmaqla hazırlanmışdır. Bununla belə, bu modul digər məlumat növlərinin klaster analizi üçündə istifadə edilə bilər. Bio.Cluster və əsas C Clustering Library De Hoon et al. [10].

Bio.Cluster-də aşağıdakı dörd klasterləşdirmə yanaşması həyata keçirilir:

- İerarxik qruplaşma (cüt-cüt mərkəzli, tək-, tam- və orta əlaqə);
- k-ortalar, k-medianlar və k-medoidlər qruplaşması;
- Özünü təşkil edən xəritələr;
- Əsas Komponent Təhlili.

## Data təmsili

Klasterləşdiriləcək məlumatlar  $n \times m$  Rəqəmsal Python massivi verilənləri ilə təmsil olunur. Gen ifadəsi məlumatlarının klasterləşdirilməsi kontekstində adətən sətirlər müxtəlif genlərə, sütunlar isə müxtəlif eksperimental şərtlərə uyğun gəlir. Bio.Cluster-də klasterləşdirmə alqoritmləri həm sətirlərə (genlərə), həm də sütunlara (təcrübələrə) tətbiq oluna bilər.

Çatışmayan dəyişikliklər

$n \times m$  Rəqəmsal Python tam massivi maskası verilənlərdəki dəyişikliklərə rəsəd olunur. Hər hansı birininə skik olub olmadığını göstərir. Əgər maska[i, j] == 0 olarsa, o zaman verilənlərdə r[i, j] yoxdur və təhlidənəzərə alınır.

Təsadüfi ədəd generatoru k-means/

median/medoid qruplaşma alqoritmləri və Özünü təşkil edən Xəritələr (SOMs) təsadüfi ədəd generatorunun istifadəsinə hatə edir. Bio.Cluster-də vahid təsadüfi ədədlər generatoru L'Ecuyer [27] alqoritminə əsaslanır, binomial paylanmadan sonra təsadüfi ədədlər isə Kachitvichyanukul və Schmeiser [23] tərəfindən BTPE alqoritmi ilə yaradılır. Təsadüfi nömrə generatoru ilk zamanı avtomatik olaraq işə salınır. Bu təsadüfi ədəd generatoru iki multiplikativ xətti konqrueşsial generatorun birləşməsindən istifadə etdiyinə görə işə salmaq üçün iki (tam ədəd) toxum lazımdır, bunun üçün biz sistem tərəfindən təmin edilmiş təsadüfi ədədlər generatorundan istifadə edirik (C standart kitabxanasında). Zəng edərək bu generatoru işə salırıq

saniyə lə rələ dövr vaxtı ilə sərand və rand tərəfinə findən yaradılan ilk iki təsadüfi adədi Bio.Cluster-də vahid təsadüfi adədlə r generatoru üçün toxum kimi istifadə edin.

## 18.1 Məsafə funksiyaları

Maddələri oxşarlığına görə qruplara bölmək üçünəvvəl oxşar dedikdədə qıqınəyi nəzərdə tutduğumuzu müəyyən etməliyik. Bio.Cluster oxşarlığı və yaəksinə məsafəni ölçmək üçün bir simvolla göstərilənəkkiz məsafə funksiyasını təmin edir:

- 'e': Euklid məsafəsi;
- 'b': Şəhər-blok məsafəsi;
- 'c': Pearson korrelyasiyaəmsalı;
- 'a': Pearson korrelyasiyaəmsalının mütləq qiyməti;
- 'u': Mərkəzsiz Pearson korrelyasiyası (iki məlumat vektoru arasındaki bucağın kosinusuna ekvivalent);
- 'x': Mütləq mərkəzsiz Pearson korrelyasiyası;
- 's': Spearmanın dərəcə korrelyasiyası;
- 'k': Kendall's τ.

İlk ikisi üçbucaq bərabərsizliyini təmin edən həqiqi məsafə funksiyalarıdır:

$$d(\underline{u}, \underline{v}) = d(\underline{u}, \underline{w}) + d(\underline{w}, \underline{v}) \text{ hamısı üçün } \underline{u}, \underline{v}, \underline{w} \text{ } \dots$$

və buna görədə ölçülər addanır. Gündəlik dildə bu o deməkdir ki, iki nöqtə arasındakıən qısa məsafə düz xəttidir.

Qalan altı məsafə ölçüsü korrelyasiyaəmsalı ilə bağlıdır, burada d məsafəsi r korrelyasiya baxımından  $d = 1 - r$  ilə müəyyən edilir. Qeyd edək ki, bu məsafə funksiyaları üçbucaq bərabərsizliyini təmin etməyən yarımmetrlərdir. Məsələn, üçün

$$\underline{u} = (1, 0, -1);$$

$$\underline{v} = (1, 1, 0); \underline{w}$$

$$= (0, 1, 1);$$

$$d(\underline{u}, \underline{v}) + d(\underline{v}, \underline{w}) = 1,6340 \text{ isə biz Pirsonun məsafəsini } d(\underline{u}, \underline{w}) = 1,8660 \text{ tapırıq.}$$

### Euklid məsafəsi

Bio.Cluster-də biz Euklid məsafəsin olaraq təyin edirik

$$d = \sqrt{\frac{1}{n} \sum_{i=1}^n (x_i - y_i)^2}.$$

Yalnız həm  $x_i$ , həm də  $y_i$  mövcud olan cəmlənməyə oşərtlər daxil edilir və məhrəc nəməvafiq olaraq seçilir. İfadə verilənləri  $x_i$  və  $y_i$  birbaşa bir-birindən çıxıldığından, Euklid məsafə sindən istifadə edərkən ifadə məlumatlarının düzgün normallaşdırıldığınaəmin olmalıdır.

## Şə hə r-blok mə safə si

Alternativ olaraq Manhetten mə safə si kimi tanınan şə hə r bloku mə safə si Evklid mə safə si ilə ə laqə dardır. Evklid mə safə si iki nöqtə arasındakı ə n qısa yolun uzunluğuna uyğun gə lidiyi halda, şə hə r-blok mə safə si hə r ölçü boyu mə safə lə rin cə midir. Gen ifadə si mə lumatlarında ə skik də yə rlə r var, buna görə Bio.Cluster-də biz şə hə r bloku mə safə sini ölçülə rin sayına bölünmüş mə safə lə rin cə mi kimi müə yyə n edirik:

$$d = \frac{1}{n} \sum_{i=1}^n |x_i - y_i|.$$

Bu, şə hə rin blokları ilə getmə li olduğunuz şə hə rin iki nöqtə si arasında getmə li olduğunuz mə safə yə bə rabə rdir. Evklid mə safə sinə gə lincə , ifadə mə lumatları birbaşa bir-birində n çıxarılır və buna görə də onların düzgün normallaşdırıldığına ə min olmalıdır.

## Pearson korrelyasiya ə msalı

Pearson korrelyasiya ə msalı kimi müə yyə n edilir

$$r = \frac{1}{n} \sum_{i=1}^n \frac{x_i - \bar{x}}{\sigma_x} \frac{y_i - \bar{y}}{\sigma_y},$$

burada  $\bar{x}$ ,  $\bar{y}$  müvafiq olaraq  $x$  və  $y$ -nin nümunə ortalamasıdır və  $\sigma_x$ ,  $\sigma_y$   $x$  və  $y$ -nin nümunə standart kə narlaşmasıdır. Pearson korrelyasiya ə msalı düz xə ttin  $x$  və  $y$  sə pə lə nmə qrafikinə nə qə də r uyğun ola bilə cə yini göstə rə n ölçündür. Ə gə r sə pə lə nmə qrafikində ki bütün nöqtə lə r düz xə tt üzə rində yerlə şırsə , Pearson korrelyasiya ə msalı xə ttin yamacının müsbə t və ya mə nfi olmasından asılı olaraq ya +1, ya da -1 olur. Pearson korrelyasiya ə msalı sıfıra bə rabə rdırsə ,  $x$  və  $y$  arasında korrelyasiya yoxdur.

Pearson mə safə si daha sonra müə yyə n edilir

$$dP = 1 - r.$$

Pearson korrelyasiya ə msalı -1 ilə 1 arasında olduğundan, Pearson mə safə si 0 ilə 2 arasındadır.

## Mütlə q Pearson korrelyasiyası

Pearson korrelyasiyasının mütlə q qiymə tini götürmə klə biz 0 ilə 1 arasında bir ə də d tapırıq. Mütlə q qiymə t 1 olarsa, sə pilmə qrafasındaki bütün nöqtə lə r ya müsbə t, ya da mə nfi yamaclı düz xə tt üzə rində yerlə şir. Mütlə q də yə r sıfıra bə rabə rdırsə ,  $x$  və  $y$  arasında korrelyasiya yoxdur.

Müvafiq mə safə kimi müə yyə n edilir

$$dA = 1 - |r|,$$

burada  $r$  Pearson korrelyasiya ə msalıdır. Pearson korrelyasiya ə msalının mütlə q qiymə ti 0 ilə 1 arasında olduğu üçün müvafiq mə safə də 0 ilə 1 arasındadır.

Gen ifadə si tə crübə lə ri kontekstində , iki genin gen ifadə profillə ri ya tam olaraq eyni və ya tam ə ks olarsa, mütlə q korrelyasiya 1-ə bə rabə rdir. Buna görə mütlə q korrelyasiya ə msalı ehtiyatla istifadə edilmə lidir.

## Mə rkə zsiz korrelyasiya (bucağın kosinusu)

Bə zi hallarda adı Pearson korrelyasiya ə msalı ə və zinə mə rkə zsiz korrelyasiyadan istifadə etmə k daha mə qsə də uyğun ola bilə r. Mə rkə zsiz korrelyasiya kimi müə yyə n edilir

$$rU = \frac{1}{n} \sum_{i=1}^n \frac{x_i}{\sigma_x^{(0)}} \frac{y_i}{\sigma_y^{(0)}},$$

harada

$$\sigma_x^{(0)} = \sqrt{\frac{1}{n} \sum_{i=1}^n x_i^2}$$

$$\sigma_y^{(0)} = \sqrt{\frac{1}{n} \sum_{i=1}^n y_i^2}$$

Bu, adı Pearson korrelyasiya  $\rho$  msalı ilə eyni ifadədir, istisna olmaqla, nümunə  $\bar{y}$  sıfır bərabərdir. Sıfır istinad və ziyyəti olduqda  $\rho$  rəksiz korrelyasiya uyğun ola bilər. Məsələn,  $x$  üçün, log-nisbət tərəf baxımından verilmiş gen ifadə məlumatı və ziyyətində, sıfır bərabər olan log-nisbət yaşıl və qırmızı siqnalın bərabər olmasına uyğundur, bu da eksperimental manipulyasiyanın gen ifadəsinə təsir etmə məsələsində kdir.

Mərəksiz korrelyasiya  $\rho$  msalına uyğun olan məsafə kimi müəyyən edilir

$$dU = 1 - rU,$$

burada  $rU$  mərəksiz korrelyasiyadır. Mərəksiz korrelyasiya  $\rho$  msalı  $-1$  ilə  $1$  arasında olduğu üçün müvafiq məsafə  $0$  ilə  $2$  arasındadır.

Mərəksiz korrelyasiya  $n$  ölçülü fəzada iki məlumat vektorunun bucağının kosinusuna bərabərdir və çox vaxt belə adlandırılır.

### Mütləq mərəksiz korrelyasiya

Müntəzəm Pearson korrelyasiyasına gəldikdə, mərəksiz korrelyasiyanın mütləq qiyməti 0 ilə 1 arasında olduğu üçün müvafiq məsafə ölçüsünü təyin edə bilərik:

$$dAU = 1 - rU, \text{ burada}$$

$rU$  mərəksiz korrelyasiya  $\rho$  msalıdır. Mərəksiz korrelyasiya  $\rho$  msalının mütləq qiyməti 0 ilə 1 arasında olduğu üçün müvafiq məsafə  $0$  ilə  $1$  arasındadır.

Həndəsi olaraq mərəksiz korrelyasiyanın mütləq qiyməti iki məlumat vektorunun dayaq xətti arasındakı kosinusa bərabərdir (yəni vektorların istiqamətinə nəzərə almadan bucaq).

### Spearman dərəcə korrelyasiyası

Spearman rütbə korrelyasiyası qeyri-parametrik oxşarlıq ölçüsünə nümunədir və Pearson korrelyasiyasından fərqli olaraq kənar göstəricilərə qarşı daha möhkəm olmağa meyllidir.

Spearman rütbəsi korrelyasiyasını hesablamak üçün hər bir vektordakı məlumatları dəyişrinə görə sıralasaq, hər bir məlumat dəyişrinin dərəcə ilə əvəz edirik. Sonra məlumat vektorları əvəzinə iki dərəcə vektor arasında Pearson korrelyasiyasını hesablayırıq.

Pearson korrelyasiya və ziyyətində olduğu kimi, biz Spearmana uyğun məsafə ölçüsünü təyin edə bilərik kimi rütbə korrelyasiyası

$$dS = 1 - rS,$$

burada  $rS$  Spearman dərəcə korrelyasiyasıdır.

### Kendall's $\tau$

Kendall  $\tau$  qeyri-parametrik oxşarlıq ölçüsünün başqa bir nümunəsidir. Bu, Spearman dərəcə korrelyasiyasına bənzəyir, lakin  $\tau$ -ni hesablamak üçün rütbələrin özlərinin əvəzinə yalnız nisbi dərəcələrdən istifadə olunur (bax: Snedecor & Cochran [43]).

Biz Kendall  $\tau$ -ə uyğun məsafə ölçüsünü təyin edə bilərik

$$dK = 1 - \tau.$$

Kendall  $\tau$  həmişə  $-1$  ilə  $1$  arasında olduğundan, müvafiq məsafə  $0$  ilə  $2$  arasında olacaq.

## Çə ki

Bio.Cluster-də mövcud olan mə safə funksiyalarının əksəriyyəti üçün çəki vektoru təbiq oluna bilər. Çəki vektorunda məlumat vektorunda olan maddələr üçün çəkilər var. Əgər i elementinin çəkisi wi-dir, o zaman həmin element datada wi dəfə baş vermiş kimi qəbul edilir. Çəkinin tam adəd olması lazımdır.

## Məsafə matrisinin hesablanması

Məsafə matrisi verilənlərdəki elementlər arasında bütün cüt məsafələrə malik kvadrat matrisdir və Bio.Cluster modulunda məsafə matrisi funksiyası ilə hesablanır:

```
>>> Bio.Cluster.idxal(məsafəsi) = matrisi =
məsafə matrisi(məlumat)
```

burada aşağıdakı arqumentlər müəyyən edilir:

- **data** (tələb olunur)  
Elementlər üçün məlumatları ehtiva edən massiv.
- **maska** (defolt: Yoxdur)  
Hansı datanın çatışmadığını göstərən tam adədlər massivi. Əgər maska[i, j] == 0 olarsa, [i, j] məlumatı yoxdur. Maska Yoxdursa, bütün məlumatlar mövcuddur.
- **çəki** (defolt: Yoxdur)  
Məsafələri hesablayarkəndə istifadə ediləcək çəkilər. Əgər çəki Yoxdursa, onda bərabər çəkilər qəbul edilir.
- **köçürmə** (defolt: 0)  
Verilənlərin sətirləri arasındaki məsafələrin hesablanacağını (köçürmə yanlışdır) və ya verilənlər sütunları arasında (köçürmə doğrudur) müəyyən edir.
- **dist** (defolt: 'e', Euklid məsafəsi)  
İstifadə olunacaq məsafə funksiyasını müəyyən edir (bax 18.1).

Yaddaşı saxlamaq üçün məsafə matrisi 1D massivlərinin siyahısı kimi qaytarılır. Hər birində sütunların sayı sıra sıra nömrəsinə bərabərdir. Beləliklə, birinci cərgədə sıfır element var. Məsələn,

```
>>> numpy.idxal(massivindən) >>>
Bio.Cluster.idxal(məsafəsi) matrisindən >>> data = massiv
([[0, 1, 2, 3], [4, 5, 6, 7], [8, 9, 10, 11], [1, 2, 3,
... 4]]) # fmt: atla
...
...
...
...
>>> məsafələr = məsafəmatrisi(məlumat, dist="e")
```

məsafə matrisi verir

```
>>> məsafələr
[massiv([], dtype=float64), massiv([16.]), massiv([64., 16.]), massiv([1.,
... 9., 49.])]
kimi yenidən yazmaq olar
[massiv([], dtype=float64), massiv([16.0]), massiv([64.0, 16.0]), massiv([1.0, 9.0, 49.0])]
```

Bu məsafə matrisinə uyğundur:

|    |    |    |   |    |   |    |
|----|----|----|---|----|---|----|
| 0  | 16 | 64 | 1 | 16 | 0 | 16 |
| 9  | 64 | 16 | 0 | 49 | 1 | 9  |
| 49 | 0  |    |   |    |   |    |

## 18.2 Klaster xassə lə rinin hesablanması

### Klaster mə rkə zlə rinin hesablanması

Klasterin mə rkə zini ya orta, ya da bütün klaster elementlə ri üzə rində hə r ölçüsün medianı kimi tə yin etmə k olar. Bio.Cluster-də ki clustercentroids funksiyası aşağıdakılardan birini hesablamaq üçün istifadə edilə bilə r:

```
>>> Bio.Cluster -də n idxal clustercentroids >>> cdata, cmask =
clustercentroids(data)
```

burada aşağıdakı arqumentlə r müə yyə n edilir:

- data (tə lə b olunur)
 

Elementlə r üçün mə lumatları ehtiva edə n massiv.
- maska (defolt: Yoxdur)
 

Hansı datanın çatışmadığını göstə rə n tam ə də dlə r massivi. Ə gə r maska[i, j] == 0 olarsa, [i, j] mə lumatı yoxdur. Maska Yoxdursa, bütün mə lumatlar mövcuddur.
- klasterid (defolt: Yoxdur)
 

Hə r bir elementin hansı klasterə aid olduğunu göstə rə n tam ə də dlə rin vektoru. Ə gə r klasterid Yoxdursa, bütün elementlə rin eyni klasterə aid olduğu güman edilir.
- metod (defolt: 'a')
 

Klaster mə rkə zini hesablamaq üçün arifmetik ortanın (metod=='a') yoxsa medianın (metod=='m') istifadə olunduğunu müə yyə n edir.

- köçürmə (defolt: 0)
 

Mə lumat sə tirlə rinin mə rkə zlə rinin hesablanacağını (köçürmə yanlışdır) və ya verilə nlə r sütunlarının mə rkə z hissə lə rinin (köçürmə Doğrudur) hesablanacağını müə yyə n edir.

Bu funksiya də zgahı qaytarır (cdata, cmask). Centroid mə lumatları 2D Rə qə msal Python massivi cdatasında saxlanılır, çatışmayan mə lumatlar 2D Rə qə msal Python tam massivi cmask ilə göstə rilir. Bu massivlə rin ölçülə ri 0-dırsa (klasterlə rin sayı, sütunların sayı) və ya (sə tirlə rin sayı, köçürmə 1-dırsə , klasterin sayı) belə dir. Hə r bir sə tir (köçürmə 0-dırsa) və ya sütun (ə gə r köçürmə 1-dırsə ) hə r bir klasterin mə rkə zinə uyğun gə lə n orta mə lumatı ehtiva edir.

### Qruplar arasındaki mə safə nin hesablanması

Maddə lə r arasındaki mə safə funksiyasını nə zə rə alaraq, iki klaster arasındaki mə safə ni bir neçə yolla müə yyə n edə bilə rik. İki klasterin arifmetik vasitə lə ri arasındaki mə safə qoşa mə rkə zli ə laqə klasterlə şmə sində və k-vasitə silə çoxluqda istifadə olunur. k-medoid klasterlə şmə sində bunun ə və zinə iki klasterin medianları arasındaki mə safə da n istifadə edilir. İki klasterin elementlə ri arasında ə n qısa cüt mə safə ikili tə k bağlantılı klasterlə şmə da , ə n uzun cütlük mə safə si isə ikili maksimum ə laqə li klasterlə şmə da istifadə olunur. Cütlü orta bağlantılı klasterlə şmə da iki klaster arasındaki mə safə cüt mə safə lə r üzə rində orta hesabla müə yyə n edilir.

İki klaster arasındaki mə safə ni hesablamaq üçün istifadə edin

```
>>> Bio.Cluster import clusterdistance >>> mə safə =
clusterdistance (data)
```

burada aşağıdakı arqumentlə r müə yyə n edilir:

- data (tə lə b olunur)
 

Elementlə r üçün mə lumatları ehtiva edə n massiv.

- maska (defolt: Yoxdur)

Hansı datanın çatışmadığını göstərən tam adədlər massivi. Eger maska[i, j] == 0 olarsa, [i, j] məlumatı yoxdur.

Maska Yoxdursa, bütün məlumatlar mövcuddur.

- çəki (defolt: Yoxdur)

Məsafələri hesablayarkəndə istifadəediləcəkçəkilər. Eger çəki Yoxdursa, onda bərabər çəkilər qəbul edilir.

- indeks1 (defolt: 0)

Birinci klasterə aid olan maddələrin indekslərini ehtiva edən siyahı. Yalnız bir elementindən ibarət klaster ya [i] siyahı, ya da tam adədi kimi təqdim ediləbilər.

- indeks2 (defolt: 0)

İkinci klasterə aid olan maddələrin indekslərini ehtiva edən siyahı. Yalnız bir elementindən ibarət klaster ya siyahı [i], ya da tam adədi kimi təqdim ediləbilər.

- metod (defolt: 'a')

Klasterlər arasındaki məsafənin necətəyin olunduğunu müəyyən edir:

- 'a': İki klaster mərkəzi arasındaki məsafə (orta arifmetik); - 'm': İki klaster mərkəzi arasındaki məsafə (median); - 's': İki çoxluqdakı elementlər arasında ən qısa cüt məsafə; - 'x': İki çoxluqdakı elementlər arasında ən uzun cüt məsafə; - 'v': İki klasterdəki elementlər arasındaki cüt məsafələrin üzərində orta.

- dist (defolt: 'e', Evklid məsafəsi)

İstifadə olunacaq məsafə funksiyasını müəyyən edir (baş 18.1).

- köçürmə (defolt: 0)

Köçürmə Yanlışdırsa, verilənlərləri arasındaki məsafəni hesablayın. Eger köçürmə doğrudursa, verilənlərin sütunları arasındaki məsafəni hesablayın.

## 18.3 Bölmə alqoritmləri

Bölmə alqoritmləri elementləri kələstərə bölmək, elementlərin üzərindən onların klaster mərkəzlərinə olan məsafələrinin cəmi minimal olsun. Klasterlərin sayı istifadəçi tərəfindən müəyyən edilir. Bio.Cluster-də üç bölmə alqoritmi mövcuddur:

- k-klasterləşmə deməkdir

- k-medianlarının klasterləşməsi

- k-medoidlərin klasterləşməsi

Bu alqoritmlər klaster mərkəzinin necətəyin olunduguna görə fərqlənir. K-vasitəsilə klasterləşmədə klaster mərkəzi klasterdəki bütün elementlər üzrə orta hesablanmış orta məlumat vektoru kimi müəyyən edilir. Orta vəzini, k-medianda klasterləşmədə median məlumat vektorunda hər bir ölçü üçün hesablanır. Nəhayət, çoxluq təşkil edən k-medoidlərdən klaster mərkəzi klasterdəki digər elementlərə ən kiçik məsafələrə minə malik olan element kimi müəyyən edilir. Bu klasterləşdirmə alqoritmi məsafə matrisinin məlumat olduğunu, lakin orijinal məlumat matrisinin mövcud olmadığı hallar üçün uyğundur, məsələn, zülalların struktur oxşarlığına əsasən qruplaşdırılması zamanı.

Bu bölməni k qrupa tapmaq üçün gözləmə -maksimasiya (EM) alqoritmini istifadə olunur. EM alqoritminin işə salınmasında biz elementləri klasterlərə təsadüfi olaraq təyin edirik. Baş klasterlərin yaranmasını təmin etmək üçün hər klasterdəki elementlərin sayını bir və ya daha çox olmaq üçün təsadüfi seçimək üçün binomial paylamadan istifadə edirik. Sonra biz təsadüfi olaraq klasterlərin yinatlarını elə elementlərə dəyişdiririk ki, hər bir elementin istənilən klasterdə olma ehtimalı bərabər olsun. Beləliklə, hər bir klasterin ən azı bir elementdən ibarət olacağına əmanət verilir.

Sonra təkrar edirik:

- Hə r bir klasterin orta, medianı və ya medoidi kimi tə yin olunan mə rkə zini hesablayın.
- klaster;
- Hə r bir elementin klaster mə rkə zlə rinə olan mə safə lə rini hesablamaq;
- Hə r bir element üçün hansı klaster mə rkə zinin e n yaxın olduğunu müə yyə nla şdirin;
- Hə r bir elementi e n yaxın klasterə yenidə n tə yin edin və ya başqa bir elementin yenidə n tə yinatı aparılmasa, iterasiyanı dayandırın yer.

İterasiya zamanı klasterlə rin boş olmasının qarşısını almaq üçün k-orta və k-medianlarda klasterlə şdirmə algoritmi hə r klasterdə ki elementlə rin sayını izlə yir və klasterdə qalan sonuncu elementin başqa klasterə tə yin edilmə sini qadağan edir. K-medoidlə rin klasterlə şmə si üçün belə bir yoxlamaya ehtiyac yoxdur, çünkü klaster mə rkə zi funksiyasını yerinə yetirə n element özü ilə sıfır mə safə ya malikdir və buna görə də heç vaxt başqa klasterə yaxın olmayacağıq.

Elementlə rin klasterlə rə ilkin tə yin edilmə si tə sadüfi olaraq hə yata keçirildiyi üçün, adə tə n, EM algoritmi hə r də fə icra edildikdə fə rqli klasterlə şdirmə hə lli tapılır. Optimal klaster hə llini tapmaq üçün k-orta algoritmi hə r də fə fə rqli ilkin tə sadüfi klasterlə şmə də n başlayaraq də fə lə rlə tə karlanır. Elementlə rin klaster mə rkə zinə olan mə safə lə rinin cə mi hə r bir qaçış üçün saxlanılır və bu mə blə ğin e n kiçik də yə ri olan hə ll ümumi klaster hə lli kimi qaytarılacaqdır.

EM algoritminin nə qə də r tez-tez işlə dilmə si qruplaşdırılan elementlə rin sayından asılıdır. Bir qayda olaraq, optimal hə llin nə qə də r tez-tez tapıldığını nə zə rdə n keçirə bilə rik; bu nömrə bu kitabxanada tə tbiq olunan bölmə algoritmlə rilə qaytarılır. Ə gə r optimal hə ll də fə lə rlə tapılıbsa, tapılandan daha yaxşı hə llə rin olması ehtimalı azdır. Bununla belə, optimal hə ll yalnız bir də fə tapılsa, klasterdaxili mə safə lə rin daha kiçik cə minə malik başqa hə llə r də ola bilə r. Maddə lə rin sayı böyükdürsə (bir neçə yüzdə n çox), qlobal miqyasda optimal hə lli tapmaq çə tin ola bilə r.

EM algoritmi heç bir yenidə n tə yinat olmadıqda dayandırılır. Bə zi ilkin klaster tə yinatları üçün EM algoritminin az sayda iterasiya addımından sonra vaxtaşırı yenidə n görünə n eyni klaster hə lli sə bə bində n birlə şə bilmə diyini gördük. Buna görə də biz tə karrlama zamanı belə dövri hə llə rin baş vermə sini yoxlaysınq. Verilmiş sayıda iterasiya addımlarından sonra cari klasterlə şdirmə nə ticə si istinad kimi saxlanılır. Hə r bir sonrakı iterasiya addımından sonra klasterlə şdirmə nə ticə sini istinad və ziyyə ti ilə müqayisə edə rə k, biz e vva llə r rast gə linə n klasterlə şdirmə nə ticə sinin tapılıb-tapılmadığını müə yyə n edə bilə rik. Bu və ziyyə tdə tə karrlama dayandırılır. Verilmiş sayıda iterasiyadan sonra istinad və ziyyə tina hə lə rast gə linmə yibə , cari klasterlə şdirmə hə lli yeni istinad və ziyyə ti kimi istifadə edilmə k üçün saxlanılır. İlkin olaraq, istinad və ziyyə tini yenidə n saxlamazdan e vvə l on iterasiya addımı yerinə yetirilir. Bu iterasiya addımlarının sayı hə r də fə iki də fə artırılır ki, daha uzun dövrlə rlə dövri davranışlar da aşkarlana bilsin.

## k-orta və k-medianlar

K-orta və k-median algoritmlə ri Bio.Cluster-də kcluster funksiyası kimi hə yata keçirilir:

```
>>> Bio.Cluster import kcluster >>> clusterid, xə ta,
nfound = kcluster (data)
```

burada aşağıdakı arqumentlə r müə yyə n edilir:

- data (tə lə b olunur)
 

Elementlə r üçün mə lumatları ehtiva edə n massiv.
- klasterlə r (defolt: 2)
 

Klasterlə rin sayı k.
- maska (defolt: Yoxdur)
 

Hansı datanın çatışmadığını göstə rə n tam e də dlə r massivi. Ə gə r maska[i, j] == 0 olarsa, [i, j] mə lumatı yoxdur. Maska Yoxdursa, bütün mə lumatlar mövcuddur.

- çə ki (defolt: Yoxdur)

Mə safə lə ri hesablayarkə n istifadə edilə cə k çə kilə r. Ə gə r çə ki Yoxdursa, onda bə rabə r çə kilə r qə bul edilir.

- köçürmə (defolt: 0)

Sə tirlə rin (köçürmə 0-dir) və ya sütunların (köçürülmə 1-dir) qruplaşdırılacağını müə yyə nlə şdirir.

- npass (defolt: 1)

Hə r də fə fə rqli (tə sadüfi) ilkin şə rtə k-means-/median klasterlə şdirmə alqoritminin yerinə yetirilmə sayı. Başlanğıc işarə si verilirsə , npass də yə ri nə zə rə alınmir və klasterlə şdirmə alqoritmi yalnız bir də fə işə salınır, çünkü o, bu halda determinist şə kildə davranışır.

- üsul (defolt: a)

Klasterin mə rkə zinin necə tapıldığına tə svir edir:

- metod=='a': arifmetik orta (k-klasterlə şmə demə kdir); -

metod=='m': median (k-median qruplaşması).

Metodun digə r qiymə tlə ri üçün arifmetik orta istifadə olunur.

- dist (defolt: 'e', Evklid mə safə si)

Istifadə olunacaq mə safə funksiyasını müə yyə n edir (bax 18.1). Bütün sə kkiz mə safə ölçülə ri kcluster tə rə fində n qə bul edildiyi halda, nə zə ri baxımdan k-orta alqoritmi üçün Evklid mə safə sində n, k-medianlar üçün isə şə hə r-blok mə safə sində n istifadə etmə k daha yaxşı olar.

- başlanğıcid (defolt: Yoxdur)

EM alqoritmi üçün istifadə edilə cə k ilkin klasterlə şmə ni müə yyə n edir. Başlanğıc identifikatoru Yoxdursa, EM alqoritminin npass qəçışlarının hə r biri üçün fə rqli tə sadüfi ilkin qruplaşma istifadə olunur. Başlanğıc işarə si Heç biri deyilsə , o, hə r bir element üçün klaster nömrə sini (0 və klasterlə r-1 arasında) ehtiva edə n 1D massivinə bə rabə r olmalıdır. Hə r klasterdə ə n azi bir element olmalıdır. İlkin klasterlə şmə ilə EM alqoritmi deterministikdir.

Bu funksiya klasteri (klasterid, xə ta, nfound) qaytarır, burada clusterid hə r bir sə tir və ya klasterin tə yin olunduğu klasterin sayını ehtiva edə n tam ə də d massividir, xə ta optimal klaster hə lli üçün klasterdaxili mə safə lə rin cə midir və nfound bu optimal hə llin tapılma sayıdır.

### k-medoidlə rin klasterlə şmə si

Kmedoidlə r programı mə safə matrisində n və istifadə ci tə rə fində n ötürürlə n klasterlə rin sayından istifadə edə rə k, verilmiş elementlə r də sti üzrə k-medoidlə rin klasterlə şdirilmə sini hə yata keçirir:

>>> Bio.Cluster -də n idxal kmedoidlə ri >>>

clusterid, xə ta, nfound = kmedoids(mə safə )

burada aşağıdakı arqumentlə r müə yyə n edilir: , nclusters=2, npass=1, initialid=None)—

- mə safə (tə lə b olunur)

Maddə lə r arasındaki mə safə lə ri ehtiva edə n matris; Bu matris üç yolla müə yyə n edilə bilə r:

- 2D Rə qə msal Python massivi kimi (burada massivin yalnız sol-aşağı hissə sinə daxil olmaq olar):

mə safə = massiv([[0.0, 1.1, 2.3], [1.1, 0.0, 4.5], [2.3, 4.5, 0.0]])

- sol-aşağı hissə sində ardıcıl olaraq mə safə lə ri ehtiva edə n 1D Rə qə msal Python massivi kimi mə safə matrisi:

mə safə = massiv([1.1, 2.3, 4.5])

- mə safə matrisinin sol-aşağı hissə sinin sə tirlə rini ehtiva edə n siyahı kimi: mə safə = [massiv[], massiv([1.1]), massiv([2.3, 4.5])]

Bu üç ifadə eyni mə safə matrisinə uyğundur.

- klasterlə r (defolt: 2)  
Klasterlə rin sayı k.
- npass (defolt: 1)  
k-medoid klasterlə şdirmə alqoritminin hə r də fə fə rqli (tə sadüfi) ilkin şə rtlə yerinə yetirilmə sayı. Başlanğıc işaretə si verilirsə , npass də yə ri nə zə rə alınmır, çünki klasterlə şdirmə alqoritmi bu halda deterministik davranışır.
- başlanğıcid (defolt: Yoxdur)  
EM alqoritmi üçün istifadə edilə cə k ilkin klasterlə şmə ni müə yyə n edir. Başlanğıc identifikasiatoru Yoxdursa, EM alqoritminin npass qəçişlarının hə r biri üçün fə rqli tə sadüfi ilkin qruplaşma istifadə olunur. Başlanğıc işaretə si Heç biri deyilsə , o, hə r bir element üçün klaster nömrə sini (0 və klasterlə r-1 arasında) ehtiva edə n 1D massivinə bə rabə r olmalıdır. Hə r klasterdə ə n azı bir element olmalıdır. İlkin klasterlə şmə ilə EM alqoritmi deterministikdir.

Bu funksiya də zgahı qaytarır (klasterid, sə hv, nfound), burada klasterid hə r bir elementin tə yin olunduğu klasterin sayını ehtiva edə n massivdir, xə ta optimal k-medoidlə rin klasterlə şdirmə hə lli üçün klasterdaxili mə safə lə rin cə midir və nfound optimal hə llin tapılma sayıdır. Qeyd edə k ki, klasterdə ki klaster nömrə si klaster mə rkə zini tə msil edə n elementin element nömrə si kimi müə yyə n edilir.

## 18.4 İerarxik qruplaşma

İerarxik klasterlə şdirmə metodları mahiyyə t etibarılı k-vasitə silə klasterlə şdirmə metodundan fə rqlidir. İerarxik klasterlə şmə də genlə r və ya eksperimental şə rtlə r arasında ifadə profilində ki oxşarlıq ağac quruluşu şə klində tə msil olunur. Bu ağac strukturu Treeview və Java Treeview kimi proqramlar vasitə silə qrafik şə kildə göstə rilə bilə r ki, bu da gen ifadə si mə lumatlarının tə hlilində iyerarxik klasterlə şmə nin populyarlaşmasına töhfə verdi.

İerarxik klasterlə şmə də ilk addım klasterlə şdirilə cə k elementlə r arasındaki bütün mə safə lə ri də qıqlə şdirə rə k mə safə matrisini hesablamaqdır. Sonra, ə n yaxın iki elementi birlə şdirə rə k bir node yaradır. Sonrakı qovşaqlar, bütün elementlə r eyni node aid olana qə də r, elementlə rin və ya qovşaqların aralarındaki mə safə də n asılı olaraq ikili birlə şmə si ilə yaradılır. Daha sonra hansı elementlə rin və qovşaqların birlə şdirildiyini tə krar izlə mə klə ağac strukturu yaradıla bilə r. EM alqoritmində n fə rqli olaraq k-vasitə silə klasterlə şmə də istifadə olunan iyerarxik klasterlə şmə nin tam prosesi deterministik karakter daşıyır.

İyerarxik klasterlə şmə nin bir neçə lə zzə ti mövcuddur ki, bu da alt qovşaqlar arasındaki mə safə nin üzvlə ri baxımından necə tə yin olunduğuuna görə fə rqla nır. Bio.Cluster-də cüt-cüt tə k, maksimum, orta və centroid ə laqə mövcuddur.

- Cütlü tə k bağlantılı klasterlə şmə də iki qovşaq arasındaki mə safə ə n qısa mə safə kimi müə yyə n edilir.  
iki qovşağın üzvlə ri arasındaki cüt mə safə lə r arasında.
- Alternativ olaraq cütlü tam ə laqə li klasterlə şdirmə kimi tanınan cütlü maksimum ə laqə li klasterlə şmə də iki qovşaq arasındaki mə safə iki qovşağın üzvlə ri arasındaki cüt mə safə lə r arasında ə n uzun mə safə kimi müə yyə n edilir.
- Cütlü orta ə laqə qruplaşmasında iki qovşaq arasındaki mə safə orta hesabla müə yyə n edilir.  
iki qovşağın elementlə ri arasındaki bütün cüt mə safə lə r.

• Cüt mə rkə zli ə laqə klasterində iki qovşaq arasındaki mə safə onların mə rkə zlə ri arasındaki mə safə kimi müə yyə n edilir. Mə rkə zlə r klasterdə ki bütün elementlə r üzə rında n orta alınmaqla hesablanır. Hə r bir yeni yaranan qovşaqdan mövcud qovşaqlara və elementlə rə olan mə safə nin hə r bir addımda hesablanması lazım olduğuna görə , ikili mə rkə zə laqə si klasterlə şdirmə nin hesablama vaxtı digə r iyerarxik klasterlə şdirmə metodlarından ə hə miyyə tli də rə cə də uzun ola bilə r. Başqa bir özə llik ondan ibarə tdir ki, (Pearson korrelyasiyasına ə sasanan mə safə ölçüsü üçün) klaster ağacında yuxarı qalxarkə n mə safə lə r mütlə q artmir və hə tta azala bilə r. Bu, Pearson korrelyasiyasından istifadə edə rkə n mə rkə zin hesablanması ilə mə safə nin hesablanması arasında uyğunsuzluqdan qaynaqlanır: Pearson korrelyasiyası mə safə nin hesablanması üçün mə lumatları effektiv şə kildə normallaşdırıldığı halda, mə rkə zin hesablanması üçün belə normallaşma baş vermir.

Cüt şə kildə tə k, tam və orta ə laqə qruplaşması üçün iki qovşaq arasındaki mə safə ni birbaşa ayrı-ayrı elementlə r arasındaki mə safə lə rdə n tapmaq olar. Buna görə də , mə safə matrisi mə lum olduqdan sonra, klasterlə şdirmə algoritminin orijinal gen ifadə mə lumatlarına girişə ehtiyacı yoxdur. Cüt mə rkə zli ə laqə klasterlə şmə si üçün isə yeni yaranmış alt qovşaqların mə rkə zlə ri mə safə matrisində n deyil, yalnız ilkin mə lumatlar ə sasında hesablanır.

Cütlü tə k bağınlılı iyerarxik klasterlə şmə nin hə yata keçirilmə si SLINK algoritmine [41] ə saslanır ki , bu da cütlü tə k bağınlılı klasterlə şdirmə nin sadə hə yata keçirilmə sində n daha sürə tli və daha sə mə rə li yaddaşa malikdir. Bu algoritm tə rə fində n ə ldə edilə n klasterlə şdirmə nə ticə si adi tə k bağınlılı algoritm tə rə fində n tapılan klasterlə şdirmə hə lli ilə eynidir. Bu kitabxanada hə yata keçirilə n tə k bağınlılı iyerarxik klasterlə şdirmə algoritmi, hə ddində n artıq yaddaş tə lə blə ri və işlə mə müddə ti sə bə bində n adi iyerarxik klasterlə şdirmə algoritmlə rinin uğursuz olduğu böyük gen ifadə si mə lumat də stlə rini qruplaşdırmaq üçün istifadə edilə bilə r.

## İerarxik klaster hə llini tə msil edir

İerarxik klasterlə şmə nin nə ticə si hə r bir qovşağıın iki elementi və ya alt qovşağı birlə şdiridiyi qovşaqlar ağacından ibarə tdir. Adə tə n, bizi yalnız hə r bir qovşaqda hansı elementlə rin və ya alt qovşaqların birlə şdirildiyi deyil, hə m də onların birlə şdirildiyi kimi oxşarlığı (və ya mə safə si) maraqlandırır. Bir qovşağı iyerarxik çoxluq ağacında saxlamaq üçün Bio.Cluster-də müə yyə n edilmiş Node sınıfında n istifadə edirik. Node nümunə sinin üç atributu var:

- sol
- sağ
- mə safə

Burada sol və sağ bu qovşaqda birlə şdirilə n iki elementə və ya alt qovşaqlara istinad edə n tam ə də dlə rdir və mə safə onların arasındaki mə safə dir. Qruplaşdırılan elementlə r 0-dan (maddə lə rin sayı - 1), klasterlə r isə -1-də n (bə ndlə rin sayı - 1) arasında nömrə lə nir. Qeyd edə k ki, qovşaqların sayı elementlə rin sayından bir azdır.

Yeni Node obyekti yaratmaq üçün sol və sağı müə yyə n etmə liyik; mə safə isteğe bağlıdır.

```
>>> Bio.Cluster import Node >>> Node(2, 3)(2, 3):
0 >>> Node(2, 3, 0.91)
(2, 3): 0.91
```

Mövcud Node obyektinin sol, sağ və mə safə atributları birbaşa də yişdirilə bilə r:

```
>>> node = Node(4, 5)
>>> node.left = 6
>>> node.right = 2 >>>
node.distance = 0,73
>>> qovşağı
(6, 2): 0,73
```

Sol və sağ tam ə də dlə r deyilsə və ya mə safə üzə n nöqtə də yərinə çevrilə bilmirsə , xə ta artır.

Python sinfi Ağacı tam iyerarxik klasterlə şdirmə hə llini tə msil edir. Ağac obyekti yaradıla bilə r Node obyektlə rinin siyahısından:

```
>>> Bio.Cluster import Node, Tree >>> qovşaqları = [Node(1, 2, 0.2), Node(0, 3, 0.5), Node(-2, 4, 0.6), Node(-1, -3, 0.9)] >>> tree = Tree(qovşaqlar) >>> print(ağac) (1, 2, 0): 0,6 (-1, -3): 0,9
```

Ağac başlıcısı qovşaqların siyahısının etibarlı iyerarxik klasterlə şdirmə nə ticə si olub olmadığını yoxlaysı:

```
>>> qovşaqlar = [Node(1, 2, 0.2), Node(0, 2, 0.5)]
>>> Ağac (qovşaqlar)
Traceback (ə n son zə ng):
Fayl "<stdin>", sə tir 1, ?
ValueError: Uyğun olmayan ağac
```

Ağac obyektində ki fə rdi qovşaqlara kvadrat mötə rizə lə r vasitə silə daxil olmaq olar:

```
>>> qovşaqları = [Node(1, 2, 0.2), Node(0, -1, 0.5)] >>> tree = Tree(qovşaqlar)
>>> tree[0] (1, 2): 0.2 >>> tree[1] (0, -1): 0.5 >>>
tree[-1] (0, -1): 0.5
```

Tree obyekti də yişmə z olduğundan, biz Tree obyektində ki fə rdi qovşaqları də yişə bilmə rik. Bununla belə , edə bilə rik ağacı qovşaqların siyahısına çevirin, bu siyahını də yişdirin və bu siyahıdan yeni ağac yaradın:

```
>>> ağac = Ağac([Node(1, 2, 0.1), Node(0, -1, 0.5), Node(-2, 3, 0.9)]) >>> print(ağac) (1, 2): 0.1 (0, -1): 0.5 (-2, 3): 0.9
>>> qovşaqlar = ağac
yoxdur (1, 0)
0.2) >>>
qovşaqlar[1].sol =
2 >>> ağac = Ağac(qovşaqlar)
>>> çap(ağac) (0, 1): 0.2 (2, -1): 0.5 (-2, 3): 0.9
```

Bu, istə nilə n Ağac obyektiinin hə mişə yaxşı formalaşmasına zə manə t verir.

Java Treeview kimi vizuallaşdırma proqramları ilə iyerarxik klaster hə llini göstə rmə k üçün bütün qovşaqlar mə safə lə rini sıfırdan birə qə də r ölçmə k daha yaxşıdır. Bu, mövcud Ağac obyektində miqyas metodunu çağırmaqla hə yata keçirilə bilə r:

```
>>> tree.scale()
```

Bu üsul heç bir arqument götürmür və Heç biri qaytarır.

Ağacı çə kmə zdə nə vvə l, siz də ağaç qovşaqlarını yenidə n sıralamaq istə yə bilə rsiniz. n elementin iyerarxik klaster hə lli, hə r bir qovşaqda sol və sağ alt qovşaqın keçidi ilə 2n - 1 fə rqli, lakin ekvivalent dendrogram kimi tə rtib edilə bilə r.

Tree.sort(order) metodu iyerarxik klasterlə şdirmə ağaçındakı hə r bir qovşağı ziyanı t edir və sol alt qovşaqın orta sıra də yə rinin sağ alt qovşaqın orta sıfariş qiymə tində n az və ya ona bə rabə r olub olmadığını yoxlayır. Ə ks tə qdirdə , sol və sağ alt düyünlə r də yişdirilir. Burada maddə lə rin sıfariş qiymə tlə ri istifadə ci tə rə fində n verilir. Yaranan dendrogramda soldan sağa sıralanan maddə lə r artan sıfariş də yə rə rinə malik olacaq. Metod çeşidlə ndikdə n sonra elementlə rin indekslə rini soldan sağa ardıcılıqla qaytaracaq:

```
>>> indekslə r = tree.sort (sifariş)
```

bələ ki, maddə indekslə ri[i] dendrogramda i mövqeyində baş verə cə kdir.

İyerarxik klasterlə şmə də n sonra, ağacı kə smə klə , Ağac obyektində saxlanılan ağaç strukturuna ə sasə n elementlə ri k qrupa qruplaşdırmaq olar:

```
>>> clusterid = tree.cut(nclusters=1)
```

burada nclusters (defolt olaraq 1) klasterlə rin istə nilə n sayıdır k. Bu üsul ağaç strukturunda yuxarı k - 1 ə laqə lə ndirici hadisə lə rə mə hə l qoymur, nə tice də k ayrı-ayrı element qrupları yaranır. Klasterlə rin sayı k müsbə t, elementlə rin sayından az və ya bə rabə r olmalıdır. Bu üsul hə r bir elementin tə yin olunduğu klasterin nömrə sini ehtiva edə n massiv klasteridi qaytarır. Klasterlə r dendrogramda soldan sağa sıra ilə 0-dan k - 1-ə qə də r nömrə lə nir.

### İyerarxik klasterlə şdirmə nin hə yata keçirilmə si

İyerarxik klasterlə şmə ni hə yata keçirmə k üçün Bio.Cluster-də ağaç klaster funksiyasından istifadə edin.

```
>>> Bio.Cluster -də n ağaç klasterini idxal edin >>> ağaç =
treecluster(data)
```

burada aşağıdakı arqumentlə r müə yyə n edilir:

- elementlə r üçün verilə nlə ri ehtiva edə n data Massivi.

- maska (defolt: Yoxdur)

Hansı datanın çatışmadığını göstə rə n tam ə də dlə r massivi. Ə gə r maska[i, j] == 0 olarsa, [i, j] mə lumatı yoxdur. Maska Yoxdursa, bütün mə lumatlar mövcuddur.

- çə ki (defolt: Yoxdur)

Mə safə lə ri hesablayarkə n istifadə edilə cə k çə kilə r. Ə gə r çə ki Yoxdursa, onda bə rabə r çə kilə r qə bul edilir.

- köçürmə (defolt: 0)

Sə tirlə rin (köçürmə nin Yanlışdır) və ya sütunların (köçürmə nin Doğrudur) qruplaşdırılacağını müə yyə nlə şdirir.

- metod (defolt: 'm') istifadə

olunacaq ə laqə metodunu müə yyə n edir:

- metod=='s': ikili tə k bağlantılı klasterlə şdirmə - metod=='m':

- cütlük maksimum- (və ya tam-) ə laqə klasterlə şdirmə - üsul=='c': cüt mə rkə zli

- ə laqə klasterlə şdirmə - metod=='a': cütlük orta ə laqə

- qruplaşması

- dist (defolt: 'e', Evklid mə safə si)
 

İstifadə olunacaq mə safə funksiyasını müə yyə n edir (bax 18.1).

Əvvə lə də n hesablanmış mə safə matrisində iyerarxik klasterlə şdirmə ni tə tbiq etmə k üçün mə safə matrisi arqumentini göstərin data arqumenti ə və zinə treecluster funksiyasını çağırarkə n:

```
>>> Bio.Cluster -də n ağaç klasterini idxl edin >>> ağaç
= treecluster(distancematrix=distance)
```

Bu və ziyyə tdə aşağıdakı arqumentlə r müə yyə n edilir:

- mə safə matrisi
 

Üç yolla tə yin edilə bilə n mə safə matrisi:

- 2D Rə qə msal Python massivi kimi (burada massivin yalnız sol-aşağı hissə sinə daxil olmaq olar):

```
mə safə = massiv([[0.0, 1.1, 2.3], [1.1, 0.0, 4.5], [2.3, 4.5, 0.0]])
```

- sol-aşağı hissə sində ardıcıl olaraq mə safə lə ri ehtiva edə n 1D Rə qə msal Python massivi kimi mə safə matrisi:

```
mə safə = massiv([1.1, 2.3, 4.5])
```

- mə safə matrisinin sol-aşağı hissə sinin sə tirlə rini ehtiva edə n siyahı kimi: mə safə =

```
[massiv([]), massiv([1.1]), massiv([2.3, 4.5])]
```

Bu üç ifadə eyni mə safə matrisinə uyğundur. Ağaç klasteri klasterlə şdirmə algoritminin bir hissə si kimi mə safə matrisində ki də yə rlə ri qarışdırıa bildiyinə görə , daha sonra ehtiyac duyarsanız, ağaç klasterinə zə ng etmə zdə n ə vvə l bu massivi başqa də yişə ndə yadda saxlamağınızdan ə min olun.

- üsul

İstifadə edilə cə kə laqə üsulu:

- metod=='s': ikili tə k bağlılıkları klasterlə şdirmə -

metod=='m': ikili maksimum- (və ya tam-) ə laqə klasterlə şdirmə - metod=='a':

ikili orta ə laqə qruplaşması

Cütlü tə k, maksimum və orta bağlılıkları klasterlə şmə yalnız mə safə matrisində n hesablanıa bilsə də , cüt mə rkə zli ə laqə mümkün deyil.

Trecluster çağırarkə n, ya data, ya da mə safə matrisi None olmalıdır.

Bu funksiya Tree obyektini qaytarır. Bu obyekt (maddə lə rin sayı - 1) qovşaqları ehtiva edir, burada elementlə rin sayı satırlar çoxluq tə şkil edilmişsə sə tirlə rin sayı və ya sütunlar qruplaşdırılıbsa sütunların sayıdır.

Hə r bir qovşaq cüt-cüt ə laqə lə ndirə n hadisə ni tə svir edir, burada qovşaq atributlarının hə r biri bir elementin və ya alt qovşaqının sayını ehtiva edir və onlar arasındaki mə safə ni ayırr. Maddə lə r 0-dan (ba ndla rin sayı - 1) nömrə lə nır, çoxluqlar isə -1-də n - (ba ndla rin sayı - 1) nömrə lə nır.

## 18.5 Özünü tə şkil edə n xə ritə lə r

Özünü tə şkil edə n xə ritə lə r (SOM) neyron şə bə kə lə ri tə svir etmə k üçün Kohonen tə rə fində n icad edilmişdir (mə sə lə n, bax: Kohonen, 1997 [25]). Tamayo (1999) ilk də fə özünü tə şkil edə n xə ritə lə ri gen ifadə mə lumatlarına tə tbiq etdi [46].

SOM-lar elementlə ri bə zi topologiyada yerlə şə n klasterlə rə tə şkil edir. Adə tə n düzbucaklı topologiya seçilir. SOM-ların yaratdığı klasterlə r elə dir ki, topologiyadakı qonşu klasterlə r topologiyada bir-birində n uzaq olan klasterlə rə nisbə tə n bir-birinə daha çox bə nzə yir.

SOM-u hesablamak üçün ilk addım topologiyada hə r bir çoxluq üçün mə lumat vektorunu tə sadüfi tə yin etmə kdir. Ə gə r sə tirlə r qruplaşdırılırsa, onda hə r bir mə lumat vektorunda elementlə rin sayı sütunların sayına bə rabə rdir.

Daha sonra sıraları bir-bir götürə rə k və topologiyada hansı klasterin a n yaxın mə lumat vektoruna malik olduğunu tapmaqla SOM yaradılır. Hə min klasterin, elə cə də qonşu klasterlə rin mə lumat vektoru nə zə rdə n keçirilə n sıranın mə lumat vektorundan istifadə etmə klə tə nzimlə nır. Tə nzimlə mə tə rə fində n verilir

$$x_{cell} = \tau \cdot x_{row} - x_{cell} . .$$

$\tau$  parametri hə r iterasiya addımında azalan parametrdir. İterasiya addımının sadə xə ttii funksiyasından istifadə etdik:

$$i \tau = \tau_{init} \cdot 1 - \frac{i}{n},$$

$\tau_{init}$  istifadə ci tə rə fində n müə yyə n edilmiş  $\tau$ -nin ilkin qiymə tidir, i cari iterasiya addımının sayı, n isə yerinə yetirilmə li olan iterasiya addımlarının ümumi sayıdır. İterasiyanın a vva linda də yişikliklə r sürə tlə hə yata keçirildiyi halda, iterasiyanın sonunda yalnız kiçik də yişikliklə r edilir.

R radiusunda olan bütün klasterlə r nə zə rdə n keçirilə n genə uygunlaşdırılır. Bu radius azaldıqca azalır hesablama kimi irə liliə yir

$$R = R_{max} \cdot 1 - \frac{i}{n},$$

maksimum radius kimi müə yyə n edilir

$$R_{max} = \sqrt{N_x^2 + N_y^2},$$

burada ( $N_x$ ,  $N_y$ ) topologiyani tə yin edə n düzbucaqlının ölçülə ridir.

Somcluster funksiyası düzbucaqlı bir şə bə kə də Özünü Tə şkil edə n Xə ritə ni hesablamaq üçün tam alqoritmi hə yata keçirir. Ə vva icə tə sadüfi a də d generatorunu işə salır. Sonra qovşaq mə lumatları tə sadüfi a də dlə r generatorundan istifadə edə rə k işə salınır. SOM-u də yişdirmə k üçün genlə rin və ya nümunə lə rin istifadə olunma sırası da tə sadüfilə şdirilmişdir. SOM alqoritmində tə kraların ümumi sayı istifadə ci tə rə fində n müə yyə n edilir.

Somcluster-i işə salmaq üçün istifadə edin

```
>>> Bio.Cluster -də n idxal somcluster >>> clusterid,
celldata = somcluster(data)
```

burada aşağıdakı arqumentlə r müə yyə n edilir:

- data (tə lə b olunur)
 

Elementlə r üçün mə lumatları ehtiva edə n massiv.
- maska (defolt: Yoxdur)
 

Hansı datanın çatışmadığını göstə rə n tam ə də dlə r massivi. Ə gə r maska[i, j] == 0 olarsa, [i, j] mə lumatı yoxdur. Maska Yoxdursa, bütün mə lumatlar mövcuddur.
- çə ki (defolt: Yoxdur)
 

mə safə lə rin hesablanması zamanı istifadə edilə cə k çə kilə ri ehtiva edir. Ə gə r çə ki Yoxdursa, onda bə rabə r çə kilə r qə bul edilir.
- köçürmə (defolt: 0)
 

Sə tirlə rin (köçürmə 0-dir) və ya sütunların (köçürülmə 1-dir) qruplaşdırılacağını müə yyə nlə şdirir.
- nxgrid, nygrid (defolt: 2, 1)
 

Özünü tə şkil edə n Xə ritə nin hesablandığı düzbucaqlı şə bə kə də üfüqü və şaquli xanaların sayı.
- başlanğıc (defolt: 0.02)
 

SOM alqoritmində istifadə olunan  $\tau$  parametrinin ilkin də yə ri. İnittau üçün standart də yə r Michael Eisenin Cluster/TreeView programında istifadə edilmiş 0.02-dir.

- niter (defolt: 1)  
İcra edilə cə k tə krarların sayı.

- dist (defolt: 'e', Evklid mə safə si)  
İstifadə olunacaq mə safə funksiyasını müə yyə n edir (bax 18.1).

Bu funksiya də zgahı qaytarır (clusterid, celldata):

- klasterid:

İki sütunlu massiv, burada sə tirlə rin sayı qruplaşdırılmış elementlə rin sayına bə rabə rdir. Hə r bir sıra elementin tə yin olunduğu düzbucaqlı SOM şə bə kə sində ki xananın xə y koordinatlarını ehtiva edir.

- mobil mə lumat:

Ə gə r sə tirlə r klasterlə şırsə ölçülə ri (nxgrid, nygrid, sütunların sayı) və ya (nxgrid, nygrid, sütunlar klasterlə şırsə sayı) olan massiv. Bu massivin hə r bir elementi [ix][iy] koordinatdakı klasterin mə rkə zi üçün gen ifadə si datasını ehtiva edə n 1D vektordur [iy] koordinatda [iy].

## 18.6 Ə sas Komponent Tə hlili

Principal Component Analysis (PCA) çoxdə yişə nli mə lumatların tə hlili üçün geniş istifadə olunan bir texnikadır. Gen ifadə mə lumatlarına Principal Component Analysis tə tbiqinin praktiki nümunə si Yeung və Ruzzo (2001) [52] tə rə fində n tə qdim edilmişdir.

Ə slində , PCA, mə lumat matrisində ki hə r bir sə tirin ə sas komponentlə r adlanan ə sas vektorlar üzə rində xə ttı cə mi kimi yazıldığı, hə r biri mə lumat vektorlarında qalan fə rqı maksimum şə kildə izah edə cə k şə kildə sıralanan və seçilə n bir koordinat çevrilmə sidir. Mə sə lə n, n × 3 mə lumat matrisi üç ölçülü fə zada n nöqtə də n ibarə t ellipsoidal bulud kimi tə qdim edilə bilə r. Birinci ə sas komponent ellipsoidin ə n uzun oxu, ikinci ə sas komponent ellipsoidin ikinci ə n uzun oxu, üçüncü ə sas komponent isə ə n qisa oxdur. Mə lumat matrisində ki hə r bir sıra ə sas komponentlə rin uyğun xə ttı kombinasiyası kimi yenidə n qurula bilə r. Bununla belə , mə lumatların ölçüsünü azaltmaq üçün adə tə n yalnız ə n vacib ə sas komponentlə r saxlanılır. Mə lumatda mövcud olan qalan dispersiya daha sonra izah edilmə miş dispersiya kimi qə bul edilir.

Ə sas komponentlə ri verilə nlə rin kovariasiya matrisinin öz vektorlarını hesablamaqla tapmaq olar. Müvafiq xüsusi də yə rlə r verilə nlə rdə mövcud olan dispersiyaların hə r bir ə sas komponent tə rə fində n nə qə də r izah edildiyini müə yyə n edir.

Ə sas komponent tə hlilini tə tbiq etmə zdə nə vvə l, adə tə n, mə lumat matrisinin hə r sütunundan orta qiymə t çıxarılır. Yuxarıdakı misalda bu, ellipsoidal buludu 3D mə kanında mə rkə zinin ə trafında effektiv şə kildə mə rkə zlə şdirir, ə sas komponentlə r ellipsoidal buluddakı nöqtə lə rin onların mə rkə zlə rinə nisbə tdə də yişmə sini tə svir edir.

Aşağıdakı pca funksiyası verilə nlə r matrisinin xüsusi qiymə tlə rini və öz vektorlarını hesablamaq üçün ə vvə lə tə k də yə r parçalanmasından istifadə edir. Sinqulyar də yə rin parçalanması Ev sahibinin bidiaqonallaşdırılmasından və QR alqoritminin variantından istifadə edə n Algol prosedurunun svd [14] C dilində tə rcümə si kimi hə yata keçirilir. Ə sas komponentlə r, ə sas komponentlə r boyunca hə r bir mə lumat vektorunun koordinatları və ə sas komponentlə rə uyğun gə lə n xüsusi qiymə tlə r daha sonra qiymə tlə ndirilir və öz də yə rin böyüklüğünün azalan ardıcılığı ilə qaytarılır. Mə lumatların mə rkə zlə şdirilmə si istə nirsə , pca rutininə zə ng etmə zdə nə vvə l mə lumat matrisinin hə r bir sütunundan orta çıxılmalıdır.

Ə sas Komponent Analizini düzbucaqlı matris verilə nlə rinə tə tbiq etmə k üçün istifadə edin

>>> Bio.Cluster -də n idxlal pca >>> sütun  
ortası, koordinatlar, komponentlə r, öz də yə rlə ri = pca(data)

Bu funksiya sütun sütununu, koordinatları, komponentlə ri, öz də yə rlə rini qaytarır:

- sütun ortası

Mə lumatda hə r bir sütunun ortasını ehtiva edə n massiv.

- koordinatlar

Əsas komponentlə rə münasibə tdə verilə nlə rdə ki hə r bir sıranın koordinatları.

- komponentlə r

Əsas komponentlə r.

- xüsusi də yə rlə r

Əsas komponentlə rin hə r birinə uyğun olan öz qiymə tlə ri.

Orijinal matris mə lumatları sütun orta + nöqtə (koordinatlar, komponentlə r) hesablamalı yenidə n yaradıla bilə r.

## 18.7 Cluster/TreeView tipli faylların işlə nmə si

Cluster/TreeView gen ifadə mə lumatlarının qruplaşdırılması üçün GUI ə saslı kodlardır. Onlar ə və lə Michael Eisen tə rə fində n yazılmışdır Stenford Universitetində olarkə n [12]. Bio.Cluster Cluster/TreeView üçün müə yyə n edilmiş formata uyğun olan mə lumat fayllarını oxumaq və yazmaq üçün funksiyaları ehtiva edir. Xüsusilə , klasterlə şdirmə nə ticə sinə hə min formatda saxlamaqla, klasterlə şdirmə nə ticə lə rini vizuallaşdırmaq üçün TreeView istifadə edilə bilə r. Alok Saldanhanının <http://jtreeview.sourceforge.net/Java-dan> istifadə etmə yi tövsiyə edirik TreeView proqramı [38], iyerarxik, elə cə də k-vasitə lə ri klasterlə şdirmə nə ticə lə rini göstə rə bilə r.

Record sinifinin obyekti Cluster/TreeView tipli mə lumat faylında saxlanılan bütün mə lumatları ehtiva edir. Kimə Mə lumat faylında olan mə lumatları Record obyektində saxlamaq üçün ə və lə lə faylı açıb sonra oxuyuruq:

```
>>> Bio Import Cluster -də n >>>
də stə yi kimi open("mydatafile.txt") ilə :
...
qeyd = Cluster.read (sapı)
...
```

Bu iki addımlı proses sizə mə lumat mə nbə yində müə yyə n rahatlıq verir. Mə sə lə n, istifadə edə bilə rsiniz

```
>>> import gzip # Python standart kitabxanası >>> handle =
gzip.open("mydatafile.txt.gz", "rt")

gziplə nmış faylı açmaq üçün və ya

>>> urllib.request import urlopen >>> from io import
TextIOWrapper >>> url = "https://
raw.githubusercontent.com/biopython/biopython/master/Tests/Cluster/cyano.txt" >>> sapı = TextIOWrapper (urlopen(url))
```

oxunmuş zə ng etmə zdə n ə və l Internetdə saxlanan faylı açmaq üçün.

Oxu ə mri Michael Eisenin Cluster/TreeView proqramı üçün müə yyə n edilmiş formatda gen ifadə si mə lumatlarını ehtiva edə n nişanla ayrılmış mydatafile.txt mə tn faylini oxuyur. Bu fayl formatında sıralar genlə ri, sütunlar isə nümunə lə ri və ya müşahidə lə ri tə msil edir. Sadə vaxt kursu üçün minimal giriş faylı belə görünə cə k: Hə r bir sıra (gen) hə mişə birinci sütündə gedə n identifikasiatora malikdir. Bu nümunə də

biz maya açıq oxu çə rçivə kodlarından istifadə edirik. Hə r bir sütunun (nümunə nin) birinci sə tirdə etiketi var. Bu nümunə də etiketlə r nümunə nin götürüldüyü vaxtı tə svir edir. Birinci sə trin birinci sütununda proqrama hə r cə rgə də hansı növ obyektlə rin olduğunu bildirə n xüsusi sahə var. Bu halda, YORF maya açıq oxu çə rçivə si demə kdir. Bu sahə istə nilə n alfasayısal də yə r ola bilə r. Cə dvə ldə ki qalan hüceyrə lə r müvafiq gen və nümunə üçün mə lumatları ehtiva edir. 2-ci sə tirdə ki 5.8 sütun 4 YAL001C geni üçün müşahidə edilə n də yə r demə kdir

|         | 0 də qıqə | 30 də qıqə | 1 saat | 2 saat | 4 saat |      |  |
|---------|-----------|------------|--------|--------|--------|------|--|
| YORF    |           |            |        |        |        |      |  |
| YAL001C | 1         |            | 1.3    | 2,4    | 5.8    | 2.4  |  |
| YAL002W | 0,9       |            | 0,8    | 0,7    | 0,5    | 0,2  |  |
| YAL003W | 0,8       |            | 2,1    | 4,2    | 10,1   | 10,1 |  |
| YAL005C | 1,1       |            | 1,3    | 0,8    |        | 0,4  |  |
| YAL010C | 1,2       |            | 1      | 1,1    | 4,5    | 8,3  |  |

| YORF    | NAME      | çə ki             | QORDERİ | 0 | 30 | 1   | 2   | 4 |   |     |     |     |
|---------|-----------|-------------------|---------|---|----|-----|-----|---|---|-----|-----|-----|
| ÇƏ Kİ   |           |                   |         |   |    |     |     |   | 1 | 1   | 1   | 0   |
| SİFARIŞ |           |                   |         |   |    |     |     |   | 5 | 3   | 2   | 1   |
| YAL001C | TFIIIC    | 138 KD alt vahidi |         |   |    |     |     |   | 1 | 1   | 1,3 | 2,4 |
| YAL002W | name      | IUM               |         |   |    | 1   |     |   | 3 | 0,9 | 0,8 | 0,7 |
| YAL003W | UZUNMA    | FAKTOR EF1-BETA   |         |   |    | 0,4 |     |   | 2 | 0,8 | 2,1 | 4,2 |
| YAL005C | CYTOSOLIC | HSP70             |         |   |    | 0,4 | 0,4 |   | 5 | 1   | 1,3 | 0,8 |

2 saatda 5.8 idi. Çatışmayan də yərlər məqbuldur və boş xanalarla təyin olunur (məsələn, YAL004C 2 saat).

Giriş faylında əlavə məlumat ola bilər. Maksimum giriş faylı belə görünür:

Əlavə edilmiş NAME, GWEIGHT və GORDER sütunları və EWEIGHT və EORDER sətirləri isteğə bağlıdır. NAME sütunu 1-ci sütundakı ID-dən fərqli olan hər bir gen üçün etiket təyin etməyi imkan verir.

Record obyekti aşağıdakı atributlara malikdir:

- verilənlər

Gen ifadə məlumatlarını ehtiva edən məlumat massivi. Genlər sırası ilə saxlanılır, nümunələrin isə saxlanılır sütun üzrə saxlanılır.

- maska

Bu massiv verilənlər massivində hansı elementlərin, əgər varsa, çatışmadığını göstərir. Əgər maska[i, j] == 0 olarsa, onda data[i, j] yoxdur. Heç bir məlumatın çatışmadığı aşkar edilərsə, maska Yox olaraq təyin edilir.

- geneid

Bu, hər bir gen üçün unikal təsviri ehtiva edən siyahıdır (yəni, ORF nömrələri).

- gen adı

Bu, hər bir gen (yəni, gen adı) üçün təsviri ehtiva edən siyahıdır. Məlumat faylında yoxdursa, genadi Yoxdur.

- çəki

Genlər arasındaki ifadə profilində ki məsafəni hesablamaq üçün istifadə ediləcək çəkilər. Əgər yoxsa məlumat faylında mövcud olduqda, gweight Yoxdur.

- qarder

Genlərin çıkış faylında saxlanması üçün üstünlük verilən sıra. Məlumat faylında yoxdursa, Gorder Yox olaraq təyin edilib.

- eksid

Bu, hər bir nümunənin təsvirini ehtiva edən siyahıdır, məsələn, eksperimental və ziyyət.

- çəki

Nümunələrin arasındaki ifadə profilində ki məsafəni hesablamak üçün istifadə ediləcək çəkilər. Əgər yoxsa məlumat faylında mövcud olduqda çəki Yoxdur.

- eorder

Nümunə lərin çıkış faylında saxlanması üçün üstünlük verilən sıra. Məlumat faylında yoxdursa, eorder Yox olaraq təyin edilir.

- uniqid

Məlumat faylında UNIQID əvəzinə istifadə edilmiş sətir.

Record obyektini yükə dikdən sonra bu atributların hər birinə birbaşa daxil olmaq və onları dəyişdirmək olar. üçün məsələn, qeyd.data loqarifmini götürərək verilənlər log-çevrilə bilər.

## Məsafə matrisinin hesablanması

Qeyddə saxlanılan elementlər arasındaki məsafə matrisini hesablamak üçün istifadə edin

```
>>> matris = qeyd.distancematrix()
```

burada aşağıdakı arqumentlər müəyyən edilir:

- köçürmə (defolt: 0)

Verilənlərin sətirləri arasındaki məsafələrin hesablanacağını (köçürmə yanlışdır) və ya verilənlər sütunları arasında (köçürmə Doğrudur) müəyyən edir.

- dist (defolt: 'e', Evklid məsafəsi)

İstifadə olunacaq məsafə funksiyasını müəyyən edir (bax [18.1](#)).

Bu funksiya məsafə matrisini sətirlərin siyahısı kimi qaytarır, burada hər bir sətirin sütunlarının sayı sıra nömrəsinə bərabərdir ([18.1-ci bölmə yə baxın](#)).

## Klaster mərkəzlərinin hesablanması

Qeyddə saxlanılan elementlərin klasterlərinin mərkəzlərinin hesablaması üçün istifadə edin

```
>>> cdata, cmask = rekord.clustercentroids()
```

- klasterid (defolt: Yoxdur)

Hər bir elementin hansı klasterə aid olduğunu göstərən tamədələrin vektoru. Əgər klasterid verilmirsə, onda bütün elementlərin eyni klasterə aid olduğu güman edilir.

- metod (defolt: 'a')

Klaster mərkəzinin hesablaması üçün arifmetik ortanın (metod=='a') yoxsa medianının (metod=='m') istifadə olunduğunu müəyyən edir.

- köçürmə (defolt: 0)

Məlumat sətirlərinin mərkəzlərinin hesablanmasını (köçürmə yanlışdır) və ya verilənlər sütunlarının mərkəz hissələrinin (köçürmə Doğrudur) hesablanmasını müəyyən edir.

Bu funksiya cdata, cmask silsiləsini qaytarır; təsvir üçün bölmə [18.2-ə](#) baxın.

## Qruplar arasındaki məsafənin hesablanması

Qeyddə saxlanılan elementlərin qrupları arasındaki məsafəni hesablamak üçün istifadə edin

```
>>> məsafə = rekord.clusterdistance()
```

burada aşağıdakı arqumentlər müəyyən edilir:

- indeks1 (defolt: 0)
 

Birinci klasterə aid olan maddə lərin indekslərini ehtiva edən siyahı. Yalnız bir i elementindən ibarət klaster ya [i] siyahı, ya da tam adədi kimi təqdim edilə bilər.

- indeks2 (defolt: 0)
 

İkinci klasterə aid olan maddə lərin indekslərini ehtiva edən siyahı. Yalnız bir i elementindən ibarət klaster ya [i] siyahı, ya da tam adədi kimi təqdim edilə bilər.

- metod (defolt: 'a')

Klasterlər arasındaki məsafənin necə təyin olunduğunu müəyyən edir:

- 'a': İki klaster mərkəzi arasındaki məsafə (orta arifmetik); - 'm': İki klaster mərkəzi arasındaki məsafə (median); - 's': İki çoxluqdakı elementlər arasında ən qısa cüt məsafə; - 'x': İki çoxluqdakı elementlər arasında ən uzun cüt məsafə; - 'v': İki klasterdəki elementlər arasındaki cüt məsafələr üzərində orta.

- dist (defolt: 'e', Evklid məsafəsi)

İstifadə olunacaq məsafə funksiyasını müəyyən edir (bax 18.1).

- köçürmə (defolt: 0)

Köçürmə Yanlışdırsa, verilənlərlə rəsətləri arasındaki məsafəni hesablayın. Eger köçürmə doğrudursa, verilənlərin sütunları arasındaki məsafəni hesablayın.

### İyerarxik klasterlər şdirmənin həyata keçirilməsi

Qeyddə saxlanılan elementlərdə iyerarxik klasterlər şdirməni həyata keçirmək üçün istifadə edin

>>> ağaç = rekord.treecluster()

burada aşağıdakı arqumentlər müəyyən edilir:

- köçürmə (defolt: 0)

Sətirlərin (köçürmənin Yanlışdır) və ya sütunların (köçürmənin Doğrudur) qruplaşdırılacağını müəyyənləşdirir.

- metod (defolt: 'm') istifadə

olunacaq əlaqə metodunu müəyyən edir:

- metod=='s': ikili tək bağıntılı klasterlər şdirmə - metod=='m': cütlük maksimum- (və ya tam-) əlaqə klasterlər şdirmə - üsul=='c': cüt mərkəzli əlaqə klasterlər şdirmə - metod=='a': cütlük orta əlaqə qruplaşması

- dist (defolt: 'e', Evklid məsafəsi)

İstifadə olunacaq məsafə funksiyasını müəyyən edir (bax 18.1).

- transpose

Genlərin və ya nümunələrin qruplaşdırılıp-toplanmadığını müəyyən edir. Transpozisiya Yanlışdırsa, genlər (sətirlər) qruplaşdırılır. Köçürmə doğrudursa, nümunələr (sütunlar) qruplaşdırılır.

Bu funksiya Tree obyektini qaytarır. Bu obyekt (maddələrin sayı - 1) qovşaqları ehtiva edir, burada elementlərin sayı satırlar çoxluq təşkil edilmişsə sətirlərin sayı və ya sütunlar qruplaşdırılırsa sütunlarının sayıdır.

Hər bir qovşaq cüt-cüt əlaqələndirən hadisəni təsvir edir, burada qovşaq attributlarının hər biri bir elementin və ya alt qovşağıın sayını ehtiva edir və onlar arasındaki məsafəni ayırrı. Maddələrin 0-dan (bəndlərin sayı - 1) nömrələri nömrələri, çoxluqlar isə -1-dən - (bəndlərin sayı - 1) nömrələri nömrələridir.

## k-orta və ya k-median qruplaşmasını yerinə yetirmə k

Qeyddə saxlanılan elementlə r üzə rində k-orta və ya k-median qruplaşmasını yerinə yetirmə k üçün istifadə edin

```
>>> clusterid, xə ta, nfound = record.kcluster()
```

burada aşağıdakı arqumentlə r müə yyə n edilir:

- klasterlə r (defolt: 2)  
Klasterlə rin sayı k.
- köçürmə (defolt: 0)  
Sə tirlə rin (köçürmə 0-dir) və ya sütunların (köçürülmə 1-dir) qruplaşdırılacağını müə yyə nlə şdirir.
- npass (defolt: 1)  
Hə r də fə fə rqli (tə sadüfi) ilkin şə rtə k-means/-median klasterlə şdirmə alqoritminin yerinə yetirilmə sayı. Başlanğıc işarə si verilirsə , npass də yə ri nə zə rə alınmır və klasterlə şdirmə alqoritmi yalnız bir də fə işə salınır, çünki o, bu halda determinist şə kildə davranışır.
- üsul (defolt: a)  
klasterin mə rkə zinin necə tapıldığıni tə svir edir:
  - metod=='a': arifmetik orta (k-klasterlə şmə demə kdir); -
  - metod=='m': median (k-median qruplaşması).

Metodun digə r qiymə tlə ri üçün arifmetik orta istifadə olunur.

- dist (defolt: 'e', Evklid mə safə si)  
İstifadə olunacaq mə safə funksiyasını müə yyə n edir (bax [18.1](#)).

Bu funksiya klasteri (clusterid, xə ta, nfound) qaytarır, burada clusterid hə r bir sə tir və ya klasterin tə yin olunduğu klasterin sayını ehtiva edə n tam a də d massividir, xə ta optimal klaster hə lli üçün klasterdaxili mə safə lə rin cə midir və nfound bu optimal hə llin tapılma sayıdır.

## Özünü tə şkil edə n xə ritə nin hesablanması

Qeyddə saxlanılan elementlə rin Özünü Tə şkil edə n Xə ritə sinə hesablamaq üçün istifadə edin

```
>>> clusterid, celldata = record.somcluster()
```

burada aşağıdakı arqumentlə r müə yyə n edilir:

- köçürmə (defolt: 0)  
Sə tirlə rin (köçürmə 0-dir) və ya sütunların (köçürülmə 1-dir) qruplaşdırılacağını müə yyə nlə şdirir.
- nxgrid, nygrid (defolt: 2, 1)  
Özünü tə şkil edə n Xə ritə nin hesablandığı düzbucaqlı şə bə kə də üfüqi və şaquli xanaların sayı.
- başlanğıc (defolt: 0.02)  
SOM alqoritmində istifadə olunan t parametrinin ilkin də yə ri. İnititau üçün standart də yə r Michael Eisenin Cluster/TreeView programında istifadə edilmiş 0.02-dir.
- niter (defolt: 1)  
İcra edilə cə k tə krarların sayı.
- dist (defolt: 'e', Evklid mə safə si)  
İstifadə olunacaq mə safə funksiyasını müə yyə n edir (bax [18.1](#)).

Bu funksiya də zgahı qaytarır (clusterid, celldata):

- klasterid:

İki sütunlu massiv, burada sə tirlə rin sayı qruplaşdırılmış elementlə rin sayına bə rabə rdir. Hə r bir sıra elementin tə yin olunduğu düzbücaqlı SOM şə bə kə sində ki xananın x və y koordinatlarını ehtiva edir.

- mobil mə lumat:

Ə gə r sa tirlə r klasterlə şırsə ölçülə ri (nxgrid, nygrid, sütunların sayı) və ya (nxgrid, nygrid, sütunlar klasterlə şırsə sayı) olan massiv. Bu massivin hə r bir elementi [ix][iy] koordinatdakı klasterin mə rkə zi üçün gen ifadə si datasını ehtiva edə n 1D vektordur [iy] koordinatda [iy].

### Klasterlə şdirmə nə ticə sinin saxlanması

Klasterlə şdirmə nə ticə sinin saxlamaq üçün istifadə edin

**>>> rekord.save (iş adı, gen qrupları, xaricetmə lə r)**

burada aşağıdakı arqumentlə r müə yyə n edilir:

- iş adı Sə tir

iş adı saxlanacaq faylların adları üçün ə sas ad kimi istifadə olunur.

- genklasterlə r

Bu arqument gen (sə tir-müdrik) qruplaşma nə ticə sinin tə svir edir. k-vasitə silə klasterlə şmə və ziyyə tində , bu, hə r bir genin aid olduğu klasterin sayını ehtiva edə n 1D massivdir. Bunu kcluster istifadə edə rə k hesablamaq olar. İyerarxik klasterlə şmə və ziyyə tində genklasterlə r Ağac obyektidir.

- eksklüzivlə r Bu

arqument eksperimental şə rtə r üçün (sütun baxımından) klasterlə şdirmə nə ticə sinin tə svir edir. k-orta klasterlə şmə və ziyyə tində , bu, hə r bir eksperimental şə rtin aid olduğu klasterin sayını ehtiva edə n 1D massivdir. Bunu kcluster istifadə edə rə k hesablamaq olar. İerarxik klasterlə şmə və ziyyə tində , xaricedicilə r Ağac obyektidir.

Bu üsul Java TreeView programı tə rə fində n sonradan oxunmaq üçün jobname.cdt, jobname.gtr, jobname.atr, jobname\*.kgg və /və ya jobname\*.kag mə tn faylini yazar. Ə gə r genklasterlə r və xaric edə nlə r heç biri deyilsə , bu üsul yalnız jobname.cdt mə tn faylini yazar; bu fayl sonradan yeni Record obyektiñde oxuna bilə r.

## 18.8 Hesablama nümunə si

Bu, genlə r üçün tə kə laqə klasterində n və eksperimental şə rtə r üçün maksimum ə laqə klasterində n istifadə edə n iyerarxik klaster hesablamasının nümunə sidir. Euklid mə safə si genlə rin klasterlə şdirılma si üçün istifadə edildiyi üçün genetree qovşaqlarının mə safə lə rini elə miqyaslaşdırmaq lazımdır ki, onların hamısı sıfır və bir arasında olsun.

Bu ağac diaqramını düzgün göstə rmə k üçün Java TreeView kodu üçün lazımdır. Eksperimental şə rtə ri qruplaşdırmaq üçün mə rkə zsiz korrelyasiya istifadə olunur. Bu halda heç bir miqyaslama tə lə b olunmur, cünki ekstrida olan mə safə lə r artıq sıfır və iki arasındadır.

Nümunə verilə nlə r cyano.txt Biopython's Tests/Cluster alt kataloqunda tapıla bilə r və kağızdandır [20, Hihara et al., 2001].

```
>>> Bio import Cluster -də n >>> açıq
("cyano.txt") ilə tutacaq olaraq : qeyd = Cluster.read(handle)
...
...
```

```
>>> genetree = record.treecluster(method="s") >>> genetree.scale()
>>> exptree =
record.treecluster(dist="u", transpose=1) >>> record.save("cyano_result",
genetree, exptree)
```

Bu, cyano\_result.cdt, cyano\_result.gtr və cyano\_result.atr fayllarını yaradacaq.  
Eynilə , biz k-vasitə silə klasterlə şdirmə hə llini saxlaya bilə rik:

```
>>> Bio import Cluster -də n >>> açıq
("cyano.txt") ilə tutacaq olaraq : qeyd =
... Cluster.read(handle)
...
>>> (geneclusters, xə ta, ifound) = record.kcluster(nclusters=5, npass=1000) >>> (exclusters, error, ifound) =
record.kcluster(nclusters=2, npass=100, transpose=1) >>> record.save("cyano_result", genecluster", expass)
```

Bu, cyano\_result\_K\_G2\_A2.cdt, cyano\_result\_K\_G2.kgg və cyano\_result\_K\_A2.kag fayllarını yaradacaq.

# Fə sil 19

## GenomeDiagram daxil olmaqla qrafika

Bio.Graphics modulu üçüncü tərəf Python kitabxanası [ReportLab-dan asılıdır](#). Diqqətinin PDF fayllarının istehsalına yönəltməsinə baxmayaraq, ReportLab həmçinin kapsullaşdırılmış postscript (EPS) və (SVG) faylları yarada bilər. Bu vektor əsaslı şəkillərə əlavə olaraq, [Python Imaging Library \(PIL\)](#) kimi müəyyənətərəfli əlavə asılılıqlar təmin edilmişdir. quraşdırıldıqda, ReportLab həmçinin bitmap şəkillərinin (JPEG, PNG, GIF, BMP və PICT formatları daxil olmaqla) çıxarılmasına bilər.

### 19.1 Genom Diaqramı

#### 19.1.1 Giriş

Bio.Graphics.GenomeDiagram modulu əvvəllər Biopython-dan asılı olaraq ayrıca Python modulu kimi mövcud olan Biopython 1.50-ə əlavə edildi. GenomeDiagram, Pritchard et al tərəfindən Bioinformatika jurnalının nəşrində təsvir edilmişdir. (2006) [35], bəzi nümunələrlə rəsədi kifayətən daxildir. Burada köhnə təlimatın PDF nüsxəsi var, <http://biopython.org/DIST/docs/GenomeDiagram/userguide.pdf> daha bir neçə nümunə var.

Adından göründüyü kimi, GenomeDiagram bütün genomları, xüsusən də prokaryotik genomları ya xətti diaqramları (istəyən görə daha yaxşı uyğunlaşmaq üçün fragmentlər bölündür) və ya dairəvi təkcə kərədiaqramları kimi çəkmək üçün nəzərdə tutulmuşdur. Toth və başqalarında şəkil 2-ə nəzər salın. (2006) [47] yaxşı nümunə üçün. O, həmçinin fag, plazmidlər və ya mitokondriya kimi kiçik genomlar üçün kifayət qədər təkcə rəsədi rəsədi məlumatçıdır. Van der Auwera və digərlərin rindən şəkil 1 və 2-ə baxın. (2009) [49] (əlavə ilə redaktə ilə göstərilir).

Bu moduldan istifadə etmək ən asandır, əgər genomunuz çoxlu sayıda elementi ehtiva edən SeqRecord obyekti kimi yüklenmişdirsə. SeqFeature obyektləri - məsələn, GenBank faylından yüklenəndiyi kimi (Fəsil 4 və 5-ə baxın).

#### 19.1.2 Diaqramlar, treklər, xüsusiyyətlər və xüsusiyyətlər

GenomeDiagram daxili obyektlər dəstində istifadə edir. Üstəviyyədə, üfüqi ox (və ya dairə) boyunca ardıcılılığı (və ya ardıcılılıq bölgəsinə) təmsil edən bir diaqram obyektiniz var. Diaqramda şəhərlər (və ya dairəvi diaqramlarda radial) yığılmış şəkildə göstərilən bir və ya daha çox yol ola bilər. Bunlar adətən eyni uzunluğa malik olacaq və eyni ardıcılılıq bölgəsinə təmsil edəcək. Gen yerlərinin göstərmək üçün bir trekdən, tənzimləyi bölgələri göstərmək üçün digər rindən və GC faizi göstərmək üçün üçüncü trekdən istifadə edə bilərsiniz.

Ən çox istifadə edilən trek növü funksiya dəstək rində birləşdirilmiş funksiyaları ehtiva edəcəkdir. Siz bütün CDS funksiyalarınız üçün bir funksiya dəstini, tRNA xüsusiyyətləri üçün isə digərini istifadə etməyi seçəbilərsiniz. Bu tələb olunmur - onların hamısı eyni funksiya dəstini daxil ola bilər, lakin bu, sadəcə seçilmiş funksiyaların xassələrini yeniləməyi asanlaşdırır (məsələn, bütün tRNAT xüsusiyyətlərini qırmızı edin).

Tam diaqram qurmağın ikiəsas yolu var. Birincisi, yuxarıdan aşağıya, diaqram obyekti yaratdığınız və sonra onun metodlarından istifadə edərək trek(lər)ə əlavə edin və xüsusiyyətlərə əlavə etmək üçün track metodlarından istifadə edin.

set(lə r) və funksiyaları əlavə etmək üçün onların metodlarından istifadə edin. İkincisi, ayrı-ayrı obyektləri (kodunuza uyğun gələn ardıcılıqla) yarada və sonra birləşdirə bilərsiniz.

### 19.1.3 Yuxarıdan aşağıya nümunə

Biz GenBank faylından oxunan SeqRecord obyektindən bütöv bir genom çəkəcə yik (bax. Fəsil 5). Bu nümunə *Yersinia pestis* biovar *Microtus* şirkətinin pPCP1 plazmidindən istifadə edir, fayl GenBank qovluğu altında Biopython vahidi testlərinə daxil edilir və ya onlayın [NC\\_005816.gb](#) saytımızdan.

```
reportlab.lib- dən rəngləri reportlab.lib.unit
-dən idxal edin Bio.Graphics -dən sm-ni idxal edin
Bio -dan GenomeDiagram-ı idxal edin SeqIO
```

```
qeyd = SeqIO.read("NC_005816.gb", "genbank")
```

Biz yuxarıdan aşağı yanaşmadan istifadə edirik, ona görə də ardıcılığımızı yükə dikdən sonra boş diaqram yaradırıq, sonra (boş) trekəlavə edin və buna (boş) xüsusiyyətlər dəstəsi əlavə edin:

```
gd_diagram = GenomeDiagram.Diagram("Yersinia pestis biovar Microtus plasmid pPCP1") gd_track_for_features =
gd_diagram.new_track(1, name="Annotasiya edilmiş Xüsusiyyətlər") gd_feature_set = gd_track_for_features.new_
```

İndi əyləncəli hissə - biz SeqRecord-da hər bir gen SeqFeature obyektini götürürük və ondan diaqramda bir xüsusiyyət yaratmaq üçün istifadə edirik. Biz onları tünd mavi və açıq mavi arasında dəyişən mavi rəngə boyayacaqıq.

record.features -dəki xüsusiyyət üçün :

```
if feature.type != "gen": # Bu xüsusiyyəti istisna
 edin davam edin

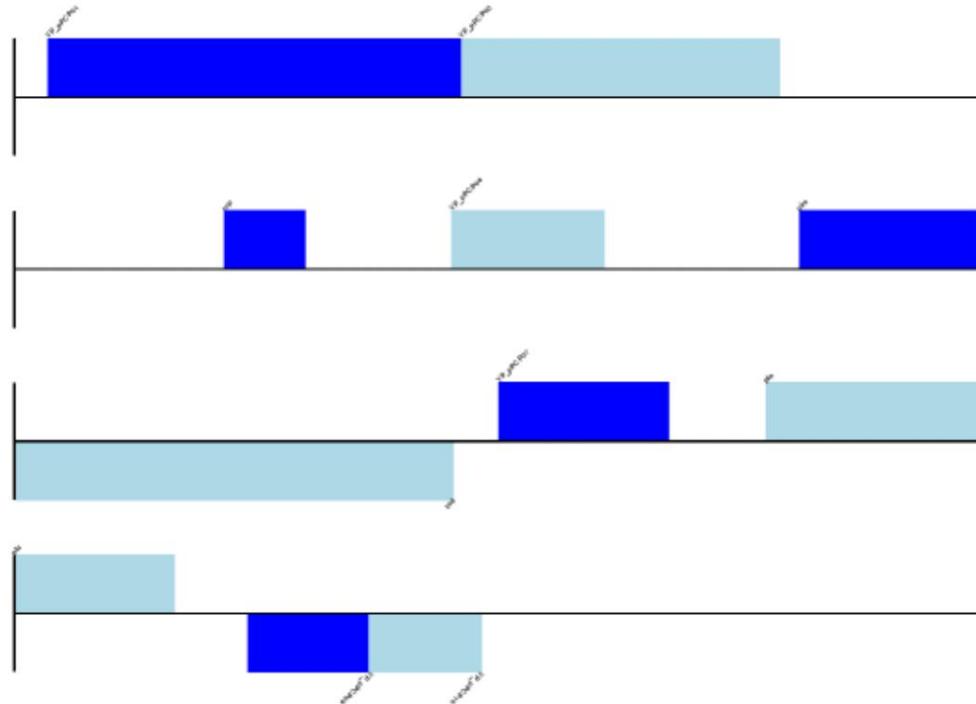
 a ga r len(gd_feature_set) % 2 == 0: rəng = rənglər.mavi
 başqa:
 color = colors.lightblue
 gd_feature_set.add_feature(xüsusiyyət, rəng=rəng, etiket=Doğru)
```

İndi faktiki olaraq çıkış faylı hazırlamağa gəlirik. Bu, iki mərhələdə baş verir, əvvəlcə ReportLab obyektlərinə istifadə edərək bütün formaları yaradan draw metodunu çağırırıq. Sonra bunları tələb olunan fayl formatına çevirən yazma metodunu çağırırıq. Bir neçə fayl formatında çıkış edə biləcəyinizi unutmayın:

```
gd_diagram.draw(format="xətti",
 orientation="landşaft", sehi hifə
 ölçüsü="A4",
 fragmentlər=4,
 başlanğıc=0,
 son=len(qeyd),

) gd_diagram.write("plazmid_xətti.pdf", "PDF")
gd_diagram.write("plazmid_xətti.eps", "EPS")
gd_diagram.write("plazmid_xətti.svg", "SVG")
```

Həmçinin, sizdə asılılıqlar quraşdırılırsa, bitmaplar da edə bilərsiniz, məsələn:



Şəkil 19.1: Yersinia pestis biovar Microtus plazmidi pPCP1 üçün sadə xətti diaqram.

```
gd_diagram.write("plazmid_xətti.png", "PNG")
```

Gözlənilən məhsul Şəkil 19.1-də göstərilmişdir. Diqqət yetirin ki, fragmentlər müəyyən etdiyimiz argumentdir dörd genomun neçə hissəyə parçalandığını idarə edir.

Dairəvi bir fiqur etmək istəyirsizsinizsə, bunu cəhd edin:

```
gd_diagram.draw(format="dairəvi",
 dairəvi=True,
 se_hifə_ölçüsü=(20 * sm, 20 * sm),
 başlanğıc=0,
 son=len(qeyd),
 dairə_nüvəsi=0.7,
```

```
) gd_diagram.write("plazmid_circular.pdf", "PDF")
```

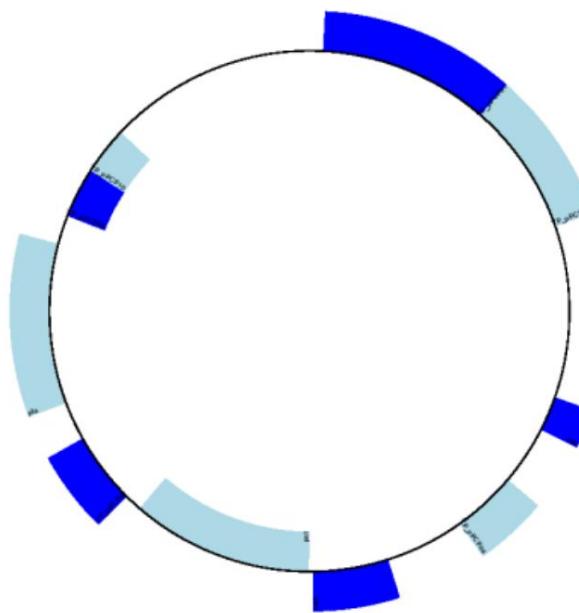
Gözlənilən məhsul Şəkil 19.2-də göstərilmişdir. Burada qəmlər çox maraqlı deyil, lakin biz yeni başlamışıq.

#### 19.1.4 Aşağıdan yuxarıya nümunə

İndi tam eyni rəqəmləri istehsal edək, lakin aşağıdan yuxarıya doğru yanaşmadan istifadə edərək. Bu o deməkdir ki, biz müxtəlif obyektləri birbaşa yaradırıq (və bu, demək olar ki, istənilən qaydada ediləbilər) və sonra onları birləşdiririk.

`reportlab.lib -dən rəngləri idxal.`

`reportlab.lib.unit -dən idxal sm`



Şəkil 19.2: Yersinia pestis biovar Microtus plazmidi pPCP1 üçün sadə dairəvi diaqram.

[Bio.Graphics](#) idxlalı GenomeDiagram - dan [Bio](#) idxlal SeqIO

```
qeyd = SeqIO.read("NC_005816.gb", "genbank")

Record.features -də xüsusiyyət üçün gd_feature_set = GenomeDiagram.FeatureSet()
funksiya də stini və onun xüsusiyyət obyektlərini yaradin :

if feature.type != "gen": # Əgər
 len(gd_feature_set) davam edərsə , bu
 funksiyani

istisna edin % 2 == 0: rəng = rəng + mavi

başqa:
 color = colors.lightblue
 gd_feature_set.add_feature(xüsusiyyət, rəng=rəng, etiket=Doğru)
(bu for loop əvvəlki nümunədə ki kimidir)

Track və diaqram yaradin gd_track_for_features
= GenomeDiagram.Track(name="Annotated Features") gd_diagram = GenomeDiagram.Diagram("Yersinia
pestis biovar Microtus plasmid pPCP1")

İndi bitəri bir-birinə yapışdırmaq lazımdır...
gd_track_for_features.add_set(gd_feature_set)
gd_diagram.add_track(gd_track_for_features, 1)
```

İndi istifadə edərək xətt və ya dairəvi diaqram yaratmaq üçün çəkilişi çağırıb əvvəlki üsulları yaza bilərsiniz  
yuxarıdan aşağıya nümunənin sonundakı kod. Rəqəmlər eyni olmalıdır.

### 19.1.5 SeqFeature olmayan funksiyalar

Yuxarıdakı misalda biz diaqramımızı qurmaq üçün SeqRecord-un SeqFeature obyektlə rində n istifadə etdik (hə məciniin Bölmə 4.3-ə baxın). Bə zə n sizdə SeqFeature obyektlə ri olmayacaq, sadə cə çə kmə k istə diyiniz funksiyanın koordinatları olacaq. Siz minimal SeqFeature obyekti yaratmalısınız, lakin bu asandır:

**Bio.SeqFeature idxal SeqFeature , SimpleLocation**

```
my_seq_feature = SeqFeature(SimpleLocation(50, 100, strand=+1))
```

Tel üçün, irə li ip üçün +1, tə rs ip üçün -1 və hə ikisi üçün heç biri istifadə edin. Budur qısa öz-özünə nümunə :

**Bio.SeqFeature- də n SeqFeature , SimpleLocation-dan Bio.Graphics idxal  
GenomeDiagram - dan reportlab.lib.units idxal sm**

```
gdd = GenomeDiagram.Diagram("Test Diaqramı")
gdt_features = gdd.new_track(1, greytrack=Yanlış gds_features =
gdt_features.new_set()

Strand seçimlə rini göstə rma k üçün üç xüsusiyyə tə lava edin,
xüsusiyyə t = SeqFeature(SimpleLocation(25, 125, strand=+1))
gds_features.add_feature(xüsusiyyə t, name="Irə li", label=True) xüsusiyyə t =
SeqFeature(SimpleLocation(150, 250, strand_feature) (strand_feature) name="Strandless",
label=True) xüsusiyyə t = SeqFeature(SimpleLocation(275, 375, strand=-1))
gds_features.add_feature(xüsusiyyə t, ad="Tə rs", etiket=Doğru)
```

```
gdd.draw(format="xə tti", sə hifə ölçüsü=(15 * sm, 4 * sm), fragmentlə r=1, başlanğıc=0, son=400)
gdd.write("GD_labels_default.pdf", "pdf")
```

Çıxış Şə kil 19.3- ün yuxarı hissə sində göstə rilmişdir (standart xüsusiyyə t rə ngində , solğun yaşıł). Diqqə t yetirin ki, biz bu xüsusiyyə tlə r üçün başlıq mə tnini müə yyə n etmə k üçün burada ad arqumentində n istifadə etmişik. Bu, daha sonra daha ə traflı müzakirə olunur.

### 19.1.6 Xüsusiyyə t başlıqları

Yada salaq ki, diaqrama xüsusiyyə tə lava etmə k üçün aşağıdakılardan istifadə etdik (burada xüsusiyyə t SeqFeature obyekti idi):

```
gd_feature_set.add_feature(xüsusiyyə t, rə ng=rə ng, etiket=Doğru)
```

Yuxarıdakı nümunə də SeqFeature annotasiyası xüsusiyyə tlə r üçün hə ssas başlıq seçmə k üçün istifadə edilmişdir. Varsayılan olaraq, SeqFeature obyektiinin kvalifikatorlar lügə ti altında aşağıdakı mümkün qeydlə r istifadə olunur: gen, etiket, ad, lokus etiketi.və mə hsul. Daha sadə desə k, adı birbaşa tə yin edə bilə rsiniz:

```
gd_feature_set.add_feature(xüsusiyyə t, rə ng=rə ng, etiket=Doğrudur, ad="Mə nim Genim")
```

Hə r bir funksiyanın etiketi üçün başlıq mə tninə ə lava olaraq, siz hə məciniin şrift, mövqe (bu, işarə nin başlanğıcı üçün standartdır, siz orta və ya sonunda da seçə bilə rsiniz) və oriyentasiyanı (yalnız xə tti diaqramlar üçün, burada defolt olaraq 45 də rə cə fırlanan) seçə bilə rsiniz:

```

Büyük şrift, trekə ilə parallel
gd_feature_set.add_feature(xüsusiyyə t,
 label=True, color="yaşıl", label_size=25, label_angle=0
)

Çox küçük şrift, trekə perpendikulyar (ona doğru) gd_feature_set.add_feature(
 xüsusiyyə t,
 etiket=Doğru,
 rəng="bə növşəyi",
 label_position="son",
 label_size=4,
 label_angle=90,
)

Kiçik şrift, trekə perpendikulyar (ondan uzaqda) gd_feature_set.add_feature(
 xüsusiyyə t,
 etiket=Doğru,
 rəng="mavi",
 label_position="orta", label_size=6,
 label_angle=-90,
)

```

Bu üç fragmntin hər birini əvvəlki bölmədə ki tam nümunə ilə birləşdirmək, Şəkil [19.3-də ki treklərə bənzər bir şey verməlidir.](#)

Biz onu burada göstərməmişik, lakin siz etiketin rəngini idarəetmək üçün etiket rəngini dətə yin edə bilərsiniz (Bölmə [19.1.9-da istifadə olunur](#)).

Defolt şriftin olduqca kiçik olduğunu görəcəksiniz - bu mənqliqidir, çünkü siz adətən çoxlu şəkil çəkəcəksiniz. Burada göstərildiyi kimi bir neçə böyük deyil, səhifədəki (kiçik) xüsusiyyətlər.

### 19.1.7 Xüsusiyyəti işaretələri

Yuxarıdakı misalların hamısı GenomeDiagram-in son açıqburaxılmış müstəqil versiyasında mövcud olan sadə bir qutu olan xüsusiyyəti üçün standart işaretədən istifadə etmişdir. GenomeDiagram Biopython 1.50-ə əlavə edildikdə ox işaretələri daxil edildi:

```

Defolt BOX sigil
gd_feature_set.add_feature(xüsusiyyə t) istifadə edir

Bunu açıq şəkildə göstərə bilərsiniz:
gd_feature_set.add_feature(xüsusiyyə t, sigil="BOX")

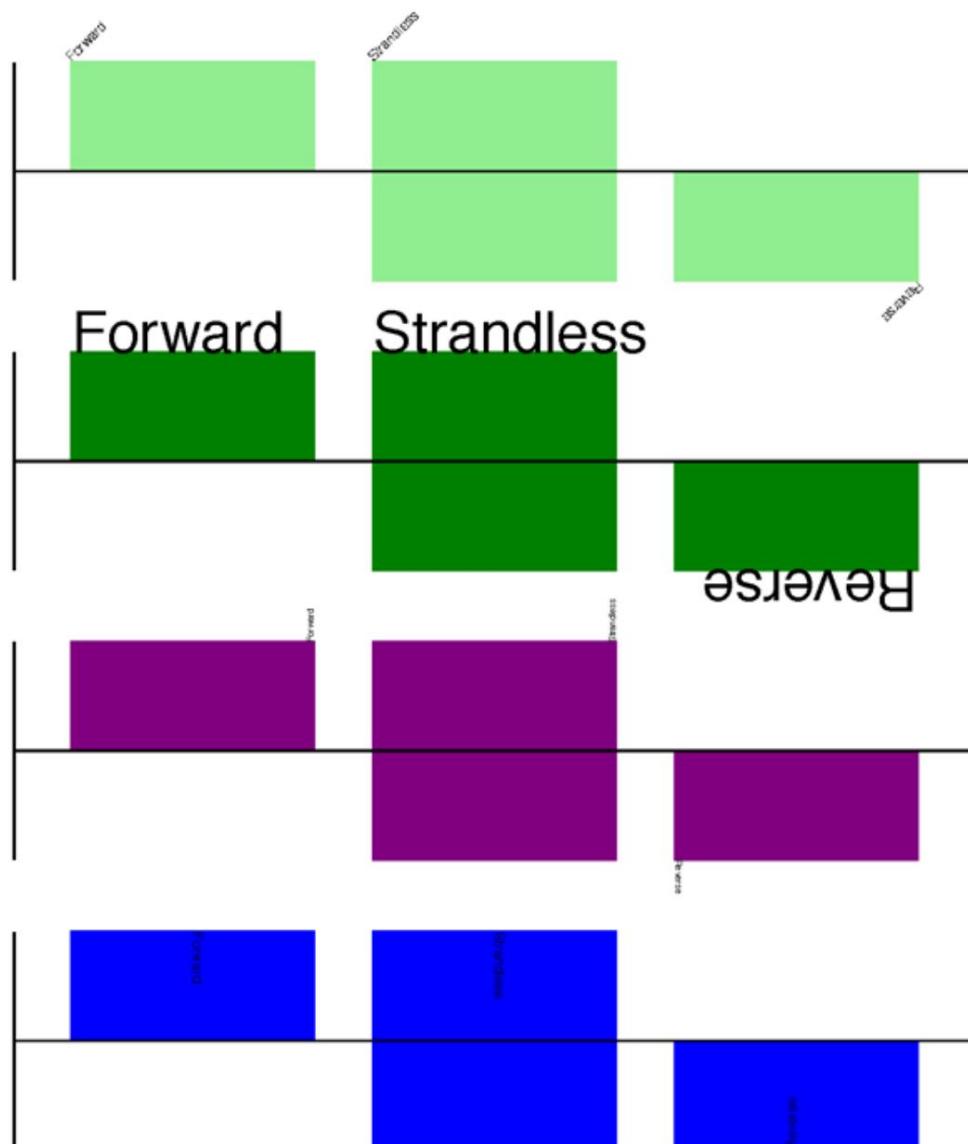
Və ya oxu seçin:
gd_feature_set.add_feature(xüsusiyyə t, sigil="ARROW")

Biopython 1.61 daha üç işaretə əlavə etdi,

Küncləri kəsilmiş qutu (onu səkkizbucaqlı edir)
gd_feature_set.add_feature(xüsusiyyə t, sigil="OCTO")

Kəsik kənarları olan qutu (içərisində olan fasılıtəri göstərmək üçün faydalıdır)

```



Şəkil 19.3: Etiket seçimlərinini göstərən sadə Genom Diaqramı. Solğun yaşıl rəngdə olan yuxarı qrafik standart etiket parametrlərinini (bax: Bölmə 19.1.5), qalanları isə etiket ölçüsündə, mövqeyində və oriyentasiyasında da yışıqlılığı ri göstərir (bax: Bölmə 19.1.6).

```
gd_feature_set.add_feature(xüsusiyyə t, sigil="JAGGY")

Gd_feature_set.add_feature (xüsusiyyə t, sigil="BIGARROW") istiqamə ti üçün istifadə edilə n tel ilə
oxu ə hatə edə n ox
```

Bunlar Şəkil 19.4-də göstərilmişdir. Əksər işarələr bir məhdudlaşdırıcı qutuya (standart BOX işarəsi ilə verilmişdir) uyğun gəlir, ya ləri və ya geri tel üçün oxun yuxarısında və ya altında, ya da telsiz xüsusiyyətlər üçün onu sıxışdırır (ikiqat hündürlük). BIGARROW işarəsi fərqlidir, həmçinin funksiyasının stendindən götürülmüş istiqamətlər oxun üstündə gəzir.

### 19.1.8 Ox işarələri

Əvvəlki bölmədə ox işarələrinin qdim etdik. Okların formalarını tənzimləmək üçün iki əlavə seçim var, ilk növbədə ox şaftının qalınlığı, məhdudlaşdırıcı qutunun hündürlüyünə nisbət olaraq verilir:

```
Tam hündürlükdə vallar, uclu qutular verir:
gd_feature_set.add_feature(xüsusiyyə t, sigil="ARROW", color="qəhvəyi", arrowshaft_height=1.0)
Və ya nazik millər:
gd_feature_set.add_feature(xüsusiyyə t, sigil="ARROW", color="çay ağacı", arrowshaft_height=0.2)
Və ya çox nazik millər:
gd_feature_set.add_feature(xüsusiyyə t,
 sigil="ARROW", color="tünd yaşıl", arrowshaft_height=0.1
)
```

Nəticələr Şəkil 19.5-də göstərilmişdir.

İkincisi, ox başının uzunluğu - məhdudlaşdırıcı qutunun hündürlüyünə nisbət olaraq verilir (standart olaraq 0,5 və ya 50%):

```
Qısa ox başlıqları:
gd_feature_set.add_feature(xüsusiyyə t, sigil="ARROW", color="mavi", ox_başı_uzunluğu=0.25)
Və ya daha uzun ox başlıqları:
gd_feature_set.add_feature(xüsusiyyə t, sigil="ARROW", color="narıncı", arrowhead_length=1)
Və ya, çox uzun ox başlıqları (yəni bütün baş, mil yoxdur, yəni üçbucaqlar): gd_feature_set.add_feature(xüsusiyyə t,
 sigil="ARROW", color="red", arrowhead_length=10000)
```

Nəticələr Şəkil 19.6-da göstərilmişdir.

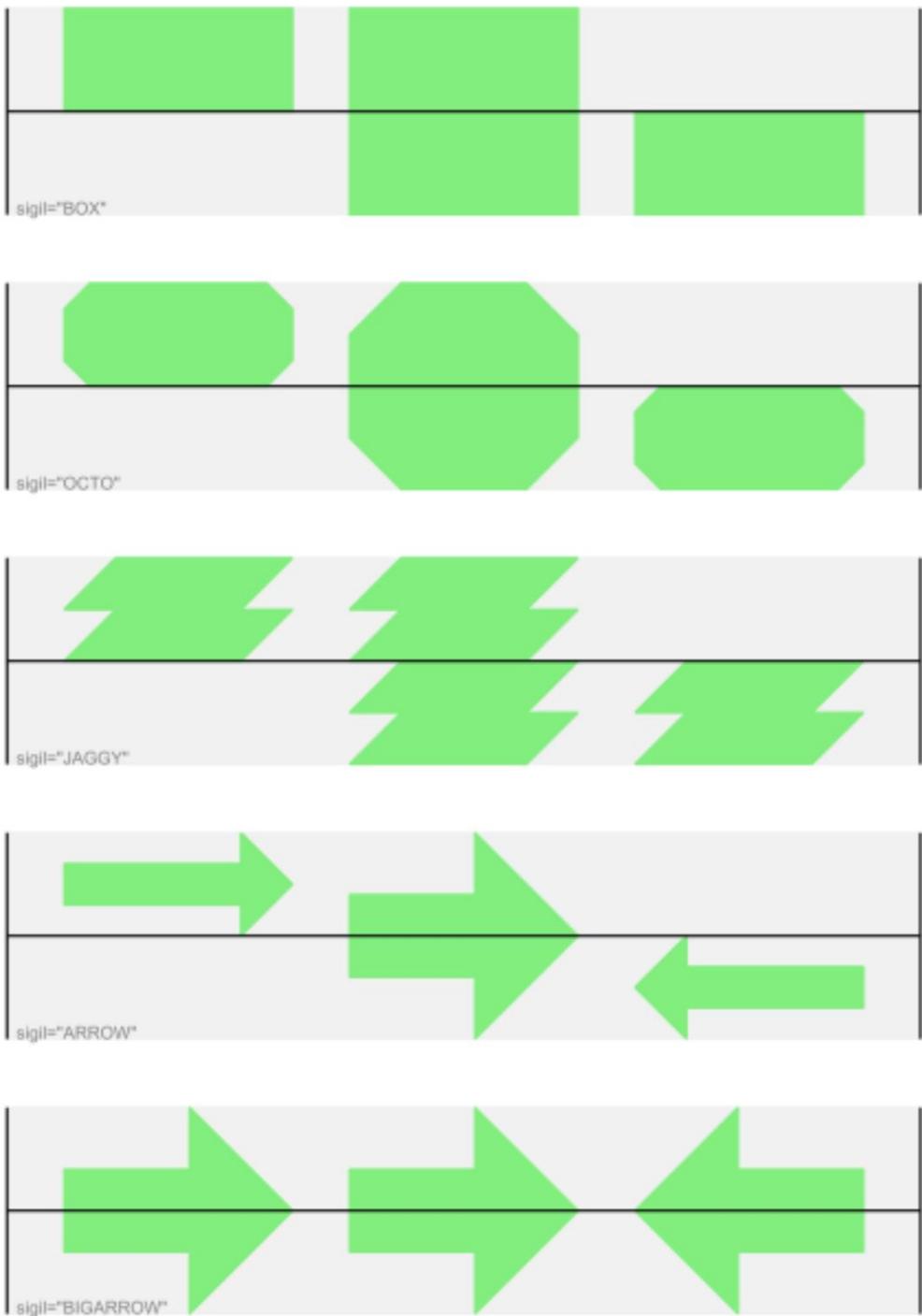
Biopython 1.61, hər zaman oxun üstündə gəzən, əksinə sola işarə edən yeni BIGARROW işarəsi əlavə edir. strand və ya başqa cür:

```
Ox boyunca uzanan böyük ox:
gd_feature_set.add_feature(xüsusiyyə t, sigil="BIGARROW")
```

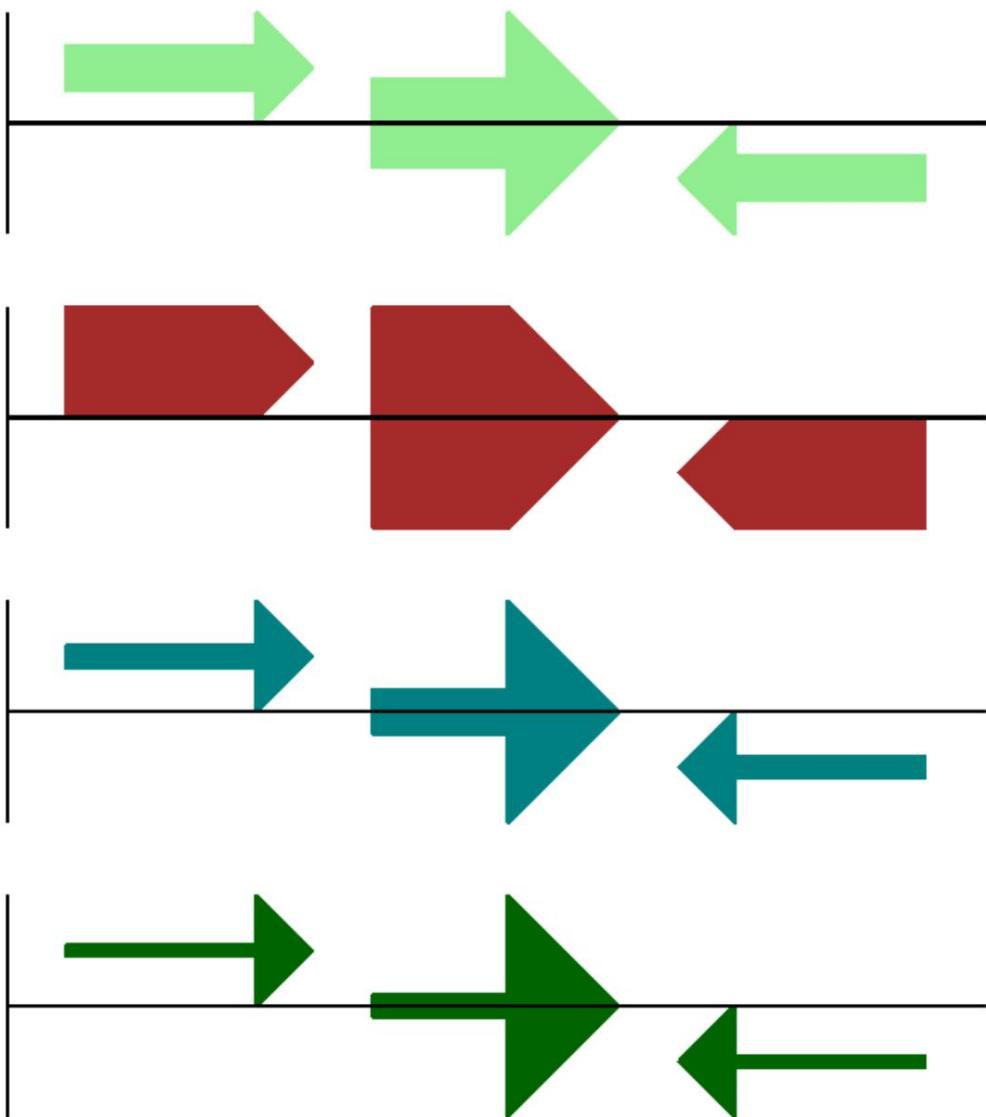
ARROW işarəsi üçün yuxarıda göstərilən bütün mil və ox başı seçimləri BIGARROW işarəsi üçün də istifadə edilə bilər.

### 19.1.9 Gözəlnümənə

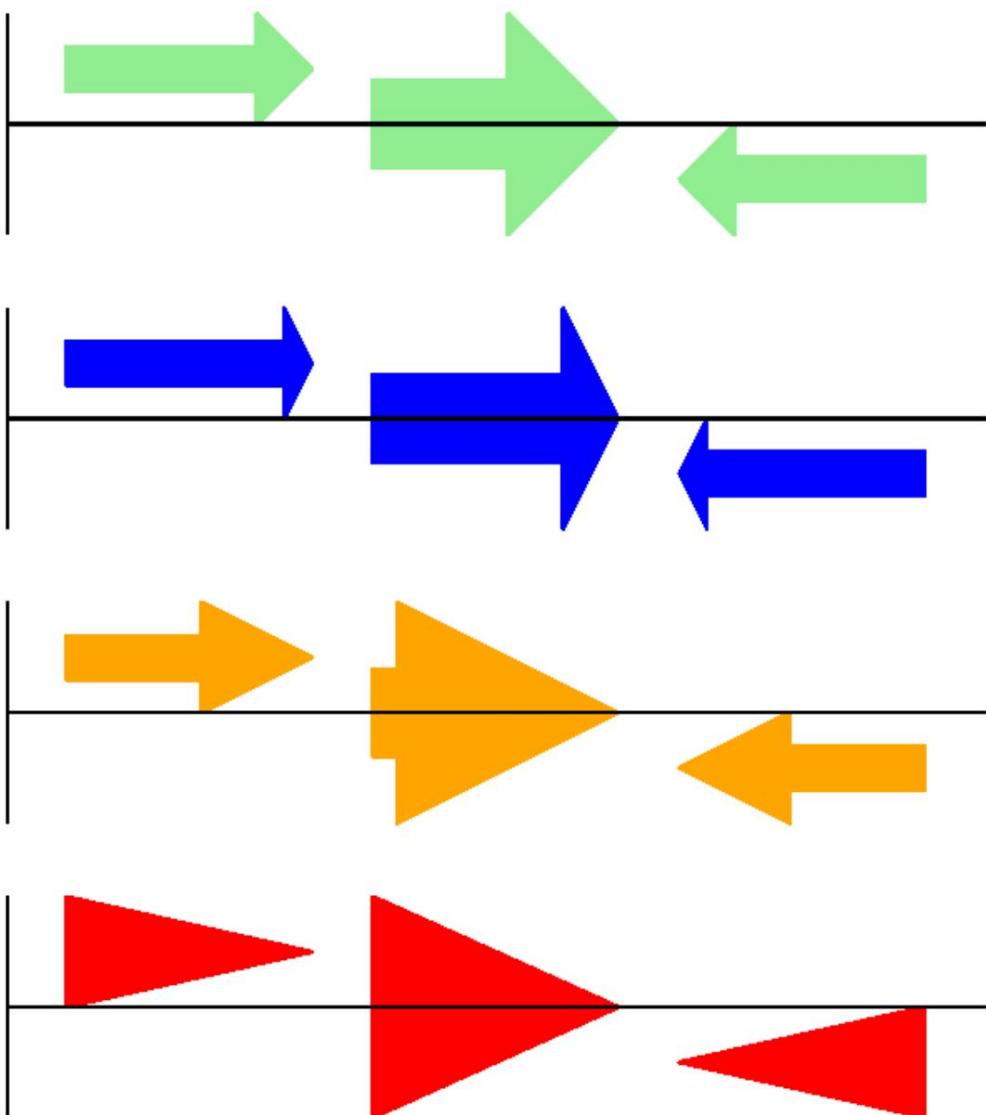
İndi Yersinia pestis biovar Microtus-dan pPCP1 plazmidinə və Bölüm 19.1.3-də istifadə olunan yuxarıdan aşağı yanaşmaya qayıdaq, lakin indi müzakirə etdiyimiz sigil seçimlərindən yararlanın. Bu dəfə biz genellər üçün oxlardan istifadə edəcəyi və onları bəzi məhdudlaşdırıcı həzm saytlarının mövqeyini göstərən ipsiz xüsusiyyətlərlə (sadə qutular kimi) örtəcəyi.



Şəkil 19.4: Müxtəlif işarələri göstərən sadə Genom Diaqramı (bax: Bölmə 19.1.7)



Şəkil 19.5: Ox mili variantlarını göstərən sadə Genom Diaqramı (bax. Bölmə 19.1.8)



Şəkil 19.6: Ox başlı variantları göstərən sadə Genom Diaqramı (bax: Bölmə 19.1.8)

```

reportlab.lib -də n rə nglə ri reportlab.lib.unit
-də n idxal et Bio.Graphics -də n sm idxal
GenomeDiagram Bio- dan idxal SeqIO- dan
Bio.SeqFeature idxal
SeqFeature, SimpleLocation

qeyd = SeqIO.read("NC_005816.gb", "genbank")

gd_diagram = GenomeDiagram.Diagram(record.id)
gd_track_for_features = gd_diagram.new_track(1, name="Annotasiya edilmiş Xüsusiyyə tlə r") gd_feature_set
= gd_track_for_features.new_set()

record.features -də ki xüsusiyyə t üçün :
 if feature.type != "gen": # Bu xüsusiyyə ti istisna
 edin davam edin

 ə gə r len(gd_feature_set) % 2 == 0:
 rə ng = rə nglə r.mavi
 başqa:
 rə ng = rə nglə r.açıq mavi
 gd_feature_set.add_feature(
 xüsusiyyə t, sigil="ARROW", rə ng=rə ng, etiket=Doğru, label_size=14, label_angle=0
)

Bə zi simsiz funksiyaları daxil etmə k istə yirə m, buna görə də mə sə lə n # EcoRI
tanınma saytlarından və s. istifadə edə cə k. sayt, ad,
rə ng [("GAATTC", "EcoRI",
 rə nglə r.yaşıl), ("CCCGGG", "SmaI",
 rə nglə r.narancı), ("AAGCTT", "HindIII", color.red),
 ("GGATCC", Bpur ", .

]:
indeks = 0
doğru olsa da:
 index = record.seq.find(sayt, start=index) ə gə r indeks == -1:
 fasılə
 xüsusiyyə t = SeqFeature(SimpleLocation(indeks, indeks + len(sayt)))
 gd_feature_set.add_feature(
 xüsusiyyə t,
 rə ng = rə ng,
 ad=ad, etiket=Doğru,
 label_size=10,
 label_color=rə ng,
)

) indeks += len(sayt)

gd_diagram.draw(format="xə tti", sə hifə ölçüsü="A4", fragmetlə r=4, başlanğıc=0, son=len(qeyd))
gd_diagram.write("plazmid_linear_nice.pdf", "PDF")
gd_diagram.write("plazmid_xə tti_nice.eps", "EPS")

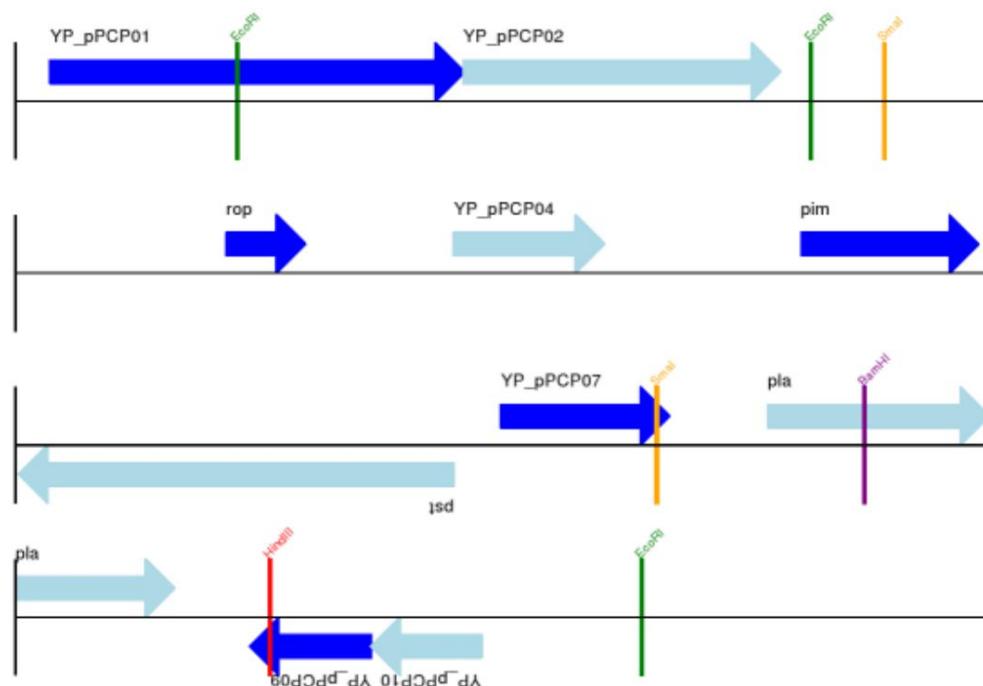
```

```
gd_diagram.write("plazmid_linear_nice.svg", "SVG")
```

```
gd_diagram.draw(format="dairə vi",
 dairə vi=Doğru,
 sə hifə ölçüsü=(20 * sm, 20 * sm),
 başlanğıc=0,
 son=len(qeyd),
 dairə _nüvə si=0,5,
)

) gd_diagram.write("plazmid_dairə vi_gözə l.pdf", "PDF")
gd_diagram.write("plazmid_dairə vi_nice.eps", "EPS")
gd_diagram.write("plazmid_dairə vi_gözə l.svg", "SVG")
```

Gözlə nilə n mə hsul Şəkil 19.7 və 19.8-də göstə rılmışdır .

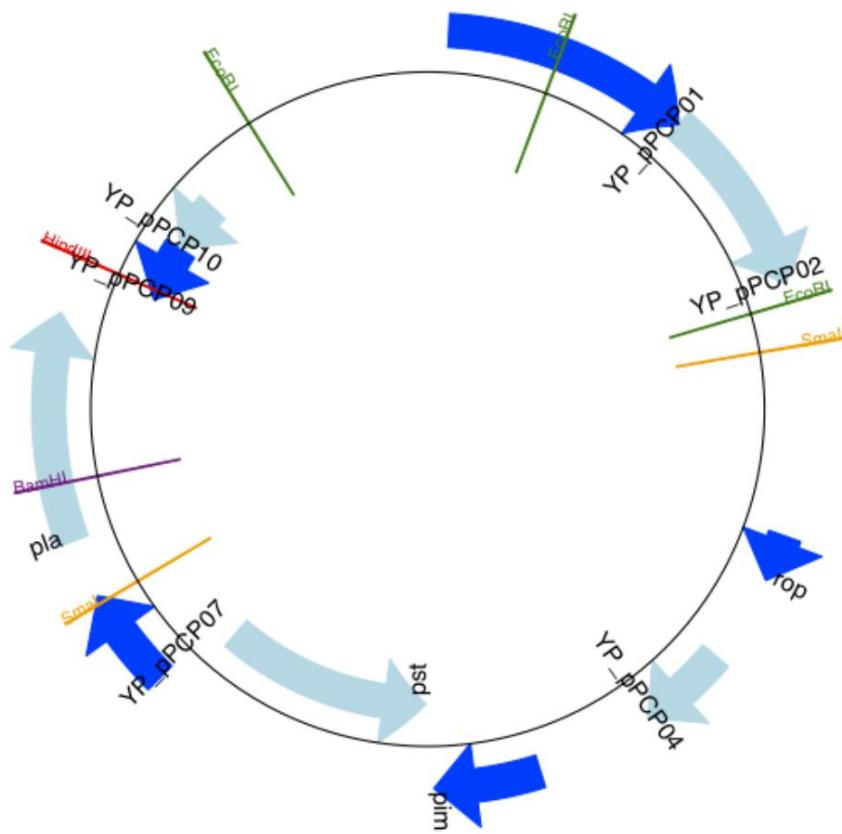


Şəkil 19.7: Yersinia pestis biovar Microtus plazmidi pPCP1 üçün seçilmiş mə hədudlaşdırıcı hə zm yerlə rini göstə rə n xə tti diaqram (bax: Bölmə 19.1.9).

### 19.1.10 Çoxsaylı treklə r

İndiyə qə də rki bütün nümunə lə r bir trekdə n istifadə edib, lakin birdə n çox trek ola bilə r - mə sə lə n, birində genlə ri göstə rin, digə rində isə bölgə lə ri tə krarlayın. Bu nümunə də biz Proux və digə rlə rində ki Şəkil 6-dan ilhamlanaraq, miqyasda yan-yana üç faj genomunu göstə rə cə yik. (2002) [36]. Aşağıdakı üç faj üçün GenBank fayllarına ehtiyacımız olacaq:

- NC\_002703 – Lactococcus fage Tuc2009, tam genom (38347 bp)



Şəkil 19.8: Yersinia pestis biovar Microtus plazmidi pPCP1 üçün seçilmiş məhdudlaşdırıcı həzərini göstərən dairəvi diaqram (bax: Bölmə 19.1.9).

- AF323668 – Bakteriofaq bIL285, tam genom (35538 bp)
- NC\_003212 – Listeria innocua Clip11262, tam genomu, biz onun yalnız integrasyasına dikkət yetiririk. prophage 5 (oxşar uzunluq).

İstə yirsizsə, bunları Entrez istifadə edə rək yüklə yə bilərsiniz, ətraflı məlumat üçün Bölmə [12.6-a](#) baxın. Üçüncü qeyd üçün biz faqın genomun harada integrasiya olunduğunu işləyib hazırladıq və onu çıxarmaq üçün qeydi kəsdi (xüsusiyyətləri qorunub saxlanılmaqla, Bölmə [4.7-yə](#) baxın) və həmçinin ilk iki faqın oriyentasiyasına uyğunlaşmaq üçün komplemanı tərsinə çevirməliyik (yenidən xüsusiyyətləri qorumaqla, Bölmə [4.9-a](#) baxın):

[Bio idxaldan SeqIO](#)

```
A_rec = SeqIO.read("NC_002703.gbk", "gb")
B_rec = SeqIO.read("AF323668.gbk", "gb")
C_rec = SeqIO.read("NC_003212.gbk", "gb")[2587879:2625807].reverse_complement(ad=Doğru)
```

Təqlid etdiyimiz figur müxtəlif gen funksiyaları üçün müxtəlif rənglərdən istifadə edirdi. Bunu etmənin bir yolu, hər bir xüsusiyyət üçün rəng seçimlərinin qeyd etmək üçün GenBank faylini redakta etməkdir - [Sanger's Artemis redaktori](#). edir və hansı GenomeDiagram başa düşməlidir. Bununla belə, biz yalnız üç rəng siyahısını kodlaşdıracağız.

Nəzərə alın ki, GenBank fayllarında olan annotasiya Proux və digərlərinde göstərilənlərə tam uyğun gəlmir, onlar bəzi qeyd olunmamış genlər çəkmmişlər.

[reportlab.lib.colors idxalından](#) (qırmızı,

```
boz,
narinci,
yaşıl,
qəhvəyi,
mavi,
açıq mavi,
bənövşəyi,
)
```

```
A_colors =
([qırmızı] *
5 + [boz] * 7 +
[narinci] * 2 + [boz]
* 2 + [narinci] +
[boz] * 11 +
[yarııl] * 4 + [boz] +
[yarııl] * 2 + [boz,
yaşıl] +
[qəhvəyi] * 5 +
[mavi] * 4 + [qəhvəyi]
* 5 + [mavi] * 4 +
[açıq yarııl] * 2 +
[boz]
```

```
)
B_rənglər =
([qırmızı] * 6
```

```

+ [boz] * 8 +
[narinci] * 2 + [boz] +
[narinci] +
[boz] * 21 +
[yasili] * 5 + [boz] +
[qə hvə yi] * 4 +
[mavi] * 3 +
[açıq mavi] * 3 +
[boz] * 5 +
[bə növşə yi] * 2

)
C_colors = ([boz]
 * 30 + [yasili] * 5
 + [qə hvə yi] * 4 +
 [mavi] * 2 + [boz,
 mavi] + [açıq mavi]
 * 2 + [boz] * 5

)

```

İndi onları çə kmə k üçün - bu də fə diaqrama üç trek ə lavə edirik və onların fə rqli verildiğini görürük onların müxtə lif uzunluqlarını ə ks etdirmə k üçün başlanğıc/bitmə də yə rlə ri (bunun üçün Biopython 1.59 və ya daha sonraki versiya tə ə b olunur).

#### Bio.Graphics idxal GenomeDiagram

```

ad = "Proux Fig 6" gd_diagram
= GenomeDiagram.Diagram(name) max_len = 0 qeyd üçün ,
gen_rə nglə ri
zipda ([A_rec, B_rec, C_rec], [A_colors, B_colors, C_colors]): max_len = max(max_lent, g_track) gd_diagram.new_track(1, ad=record.name,
greytrack=Doğru, start=0, end=len(qeyd)

) gd_feature_set = gd_track_for_features.new_set()

i = 0
record.features -də ki xüsusiyyə t üçün :
 if feature.type != "gen": # Bu xüsusiyyə ti istisna
 edin davam edin

 gd_feature_set.add_feature(xüsusiyyə t,
 sigil="ARROW",
 rə ng=gen_rə nglə r[i],
 etiket=Doğru,
 ad=str(i + 1),
 label_position="start", label_size=6,
 label_angle=0,

```

```

)
i += 1

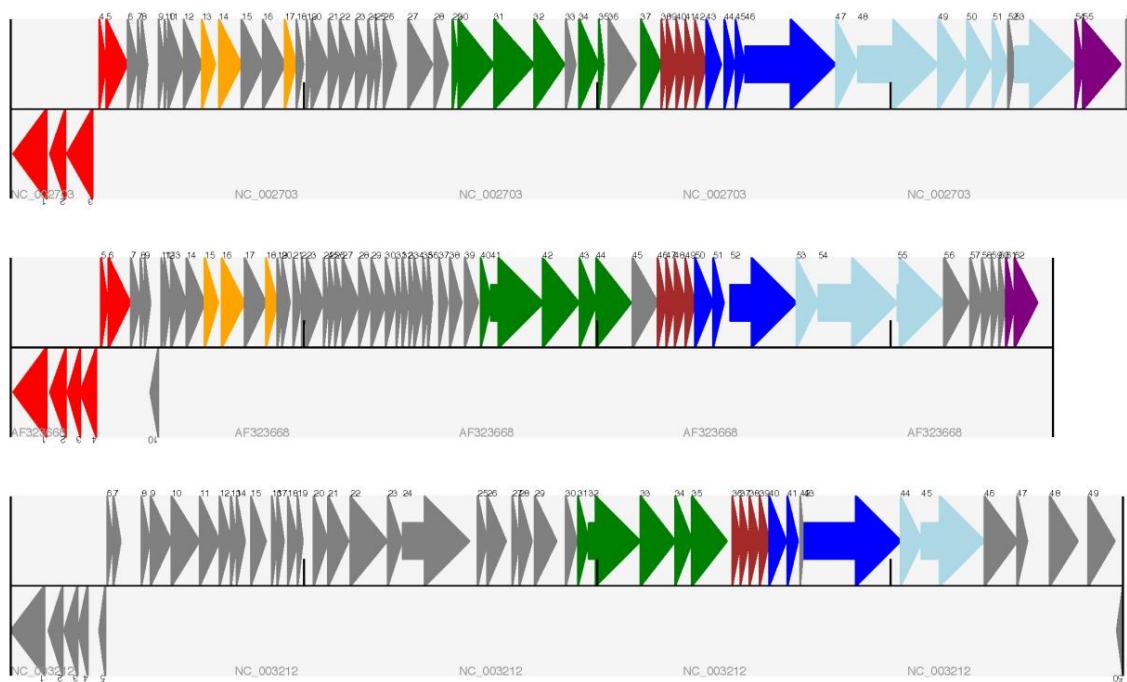
```

```

gd_diagram.draw(format="xə tti", sə hifə ölçüsü="A4", fragmentlə r=1, start=0, end=max_len)
gd_diagram.write(ad + ".pdf", "PDF")
gd_diagram.write(ad + ".eps", "EPS")
gd_diagram.write(ad , "SV"s

```

Gözlə nilə n mə hsul Şə kil 19.9-da göstə rılmışdır . Mə n tə ə ccüb etdim ki, orijinal ə lyazmada niyə yoxdur?



Şə kil 19.9: Lactococcus fage Tuc2009 (NC 002703), bakteriofaq bIL285 (AF323668) və Listeria innocua Clip11262 (NC 003212) profaq 5 üçün üç yollu xə tti diaqram (bax. 1110).

alt fajda qeyd olunan qırmızı və ya narıncı genlə r. Başqa bir vacib mə qam, burada fajların müxtə lif uzunluqlarla göstə rildiyidir - bunun sə bə bi, onların hamisinin eyni miqyasda çə kilmə lə ridir (onlar müxtə lif uzunluqlarıdır).

Də rc edilmiş rə qə mdə n ə sas fə rq, onların oxşar zülallar arasında rə ng kodlu ə laqə lə ri olmasıdır - növbə ti hissə də nə edə cə yik.

#### 19.1.11 Treklə r arasında keçidlə r

Biopython 1.59 treklə r arasında çarraz bağantwortalar çə kmə k qabiliyyə tini ə lavə etdi - burada göstə rə cə yimiz kimi hə m sadə xə tti diaqramlar, hə m də fragmentlə rə və dairə vi diaqramlara bölünmüş xə tti diaqramlar.

Proux və digərlərindən Şəkil 6-dan ilhamlanan əvvəlki bölmədən nümunəni davam etdirərək. 2002 [36], istifadə etmək üçün xal və ya rənglə birlikdə gen cütləri arasında çarbazlaşdırılmış siyahısına ehtiyacımız olacaq. Həqiqətən, siz bunu BLAST faylından hesablama yolu ilə çıxara bilərsiniz, amma mən onları əllilə daxil etmişəm. Mənim adlandırma konvensiyam üç faja A, B və C kimi istinad etməyə davam edir. İstədiyimiz bağıntıları bunlardır. A və B arasında göstərmək, çubuqların siyahısı kimi verilmişdir (faizlə oxşarlıq balı, A-da gen, B-də gen).

```
Tuc2009 (NC_002703) vs bIL285 (AF323668)
```

```
A_vs_B = [
 (99, "Tuc2009_01", "int"),
 (33, "Tuc2009_03", "orf4"),
 (94, "Tuc2009_05", "orf6"),
 (100, "Tuc2009_06", "orf7"),
 (97, "Tuc2009_07", "orf8"),
 (98, "Tuc2009_08", "orf9"),
 (98, "Tuc2009_09", "orf10"),
 (100, "Tuc2009_10", "orf12"),
 (100, "Tuc2009_11", "orf13"),
 (94, "Tuc2009_12", "orf14"),
 (87, "Tuc2009_13", "orf15"),
 (94, "Tuc2009_14", "orf16"),
 (94, "Tuc2009_15", "orf17"),
 (88, "Tuc2009_17", "rusA"),
 (91, "Tuc2009_18", "orf20"),
 (93, "Tuc2009_19", "orf22"),
 (71, "Tuc2009_20", "orf23"),
 (51, "Tuc2009_22", "orf27"),
 (97, "Tuc2009_23", "orf28"),
 (88, "Tuc2009_24", "orf29"),
 (26, "Tuc2009_26", "orf38"),
 (19, "Tuc2009_46", "orf52"),
 (77, "Tuc2009_48", "orf54"),
 (91, "Tuc2009_49", "orf55"),
 (95, "Tuc2009_52", "orf60"),
]
```

Eynilə B və C üçün:

```
bIL285 (AF323668) vs Listeria innocua prophage 5 (NC_003212-də)
```

```
B_vs_C = [(42,
 "orf39", "lin2581"), (31, "orf40",
 "lin2580"), (49, "orf41", "lin2579"), #
 terL (54, "orf42", "lin2578"), # portal (55, "orf47", """
 lin2578"), # portal (55, "orf47", "lin258" "lin2576"), #
 mhp (51, "orf46", "lin2575"), (33, "orf47", "lin2574"), (40,
 "orf48", "lin2573"), (25, "orf49", "lin2572"), (50,
 "orf571", "lin2570"), (24, "orf52",
 "lin2568"),
```

```
(30, "orf53", "lin2567"), (28, "orf54",
"lin2566"),
]
```

Birinci və sonuncu fəq üçün bu identifikatorlar lokus teqləridir, orta fəq üçün isə heç bir lokus teqləri yoxdur, ona görə də mən bunun ə və zinə gen adlarından istifadə etmişəm. Aşağıdakı kiçik köməkçi funksiya bizə lokus etiketi və ya gen adından istifadə edərək bir xüsusiyyət axtarmağa imkan verir:

```
def get_feature(features, id, tags=["locus_tag", "gene"]): """Verilmiş teqlər altında identifikator üçün
SeqFeature obyektlərinin axtarış siyahısı. Xüsusiyyətlərdə rəsəd üçün:
```

```
teqlər rəsəd ki açar üçün :
f.qualifiers.get(açar, []): x üçün bu funksiyada # teq olmaya
bilər :
ə gər x == id:
qayitmaq f
```

Açar **Xə tası (id) qaldırın**

İndi biz həmin identifikator cütlərinin siyahısını SeqFeature cütlərinə çevirə və bununla da onların yerləşdiyi yerin koordinasiyasını tapa bilərik. İndi biz bütün kodu və aşağıdakı fragmənti vəvvəl iki nümunə yəlavə edə bilərik (gd\_diagram.draw(...)) sətirindən nə bir qədərə vəvvəl -bitmiş nümunə skriptinə baxın [Proux et al 2002 Şəkil 6.py](#) şəklə carpaz bağıntıları yəlavə etmək üçün Biopython mənbə kodunun Doc/examples qovluğuna daxil edin:

[Bio.Graphics.GenomeDiagram-](#) dan [CrossLink-](#) i [reportlab.lib-](#) dən idxal rəsəngərinin idxal edin

```
Qeyd edək ki, [(A_rec, 3, B_rec, 2, A_vs_B), (B_rec, 2, C_rec, 1, B_vs_C), rec_X, tn_X, rec_Y, tn_Y, X_vs_Y üçün trek
nömrələrinin açıq şəkildə təyin etmək daha aydın ola
bilərdi .
```

]:

```
track_X = gd_diagram.tracks[tn_X] track_Y =
gd_diagram.tracks[tn_Y], X_vs_Y-də id_X, id_Y:
feature_X = get_feature (rec_X.features, id_X)
feature_Y = get_feature (rec_Y.features.Colorbycolorly),
```

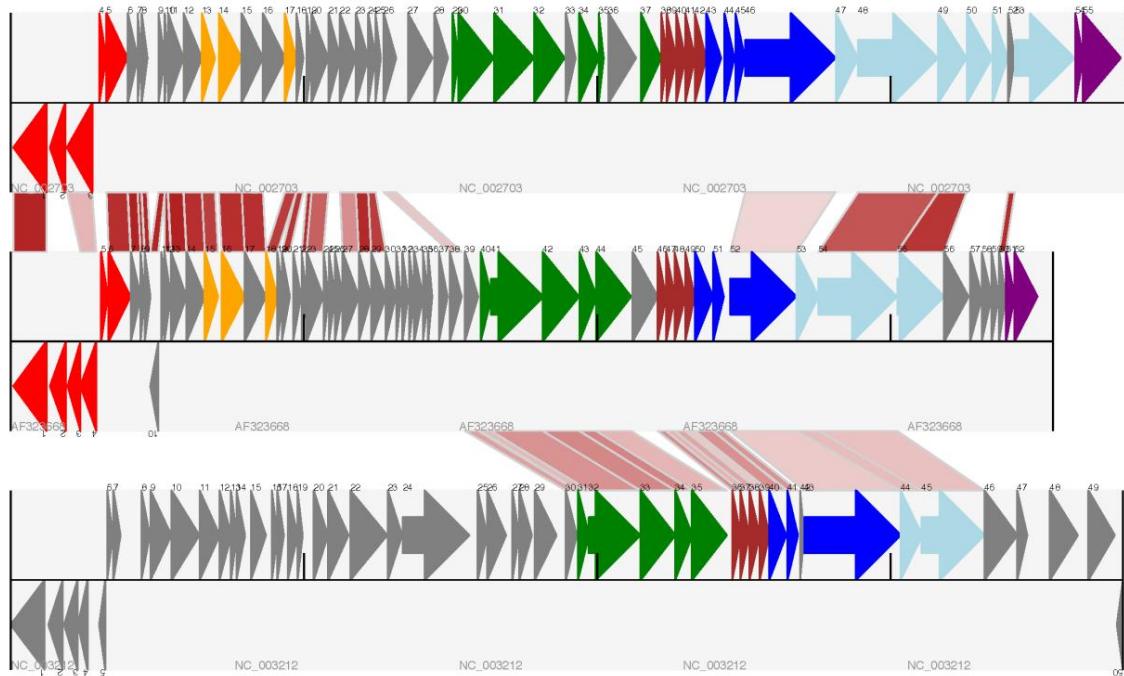
rəsəngər.ağı, rəsəngər.ka\_rpic, 0, 100, xal

```
) link_xy = CrossLink((track_X,
feature_X.location.start, feature_X.location.end) .
```

```
) gd_diagram.cross_track_links.append(link_xy)
```

Bu kodun bir neçə vacib hissəsi var. Əvvəlcə GenomeDiagram obyektində cross\_track\_links atributu var, bu atribut sadəcə CrossLink obyektlərinin siyahısıdır. Hər bir CrossLink obyekti trek üçün xüsusi koordinatların iki dəstini götürür (burada dəstlər kimi verilmişdir, bunun əvəzinə alternativ olaraq GenomeDiagram.Feature obyektindən istifadə edə bilərsiniz). Siz isteğe bağlı olaraq rəsəng, haşıyərəngi təqdim edə və bu linkin çevrilmiş şəkildə çəkilib çəkilməməsi lazımdır. Siz isteğe bağlı olaraq göstərmək üçün faydalıdır).

Siz hə mçinin aq (0%) və tünd qırmızı (100%) arasında interpolyasiya edə rə k BLAST faizinin eynilik xalını rə ngə necə çevirdiyimizi görə bilə rsiniz. Bu nümunə də çarpaz keçidlə rin üst-üstə düşmə si ilə bağlı heç bir problemimiz yoxdur. Bunun öhdə sində n gə lmə yin bir yolu, alfa kanal də sti ilə rə nglə rdə n istifadə edə rə k ReportLab-da şə ffaflıqdan istifadə etmə kdir. Bununla belə , üst-üstə düşə n şə ffaflıqla birlə şə n bu cür kölgə li rə ng sxemini şə rh etmə k çə tin olardı. Gözlə nilə n mə hsul Şə kil 19.10-da göstə rilmişdir .

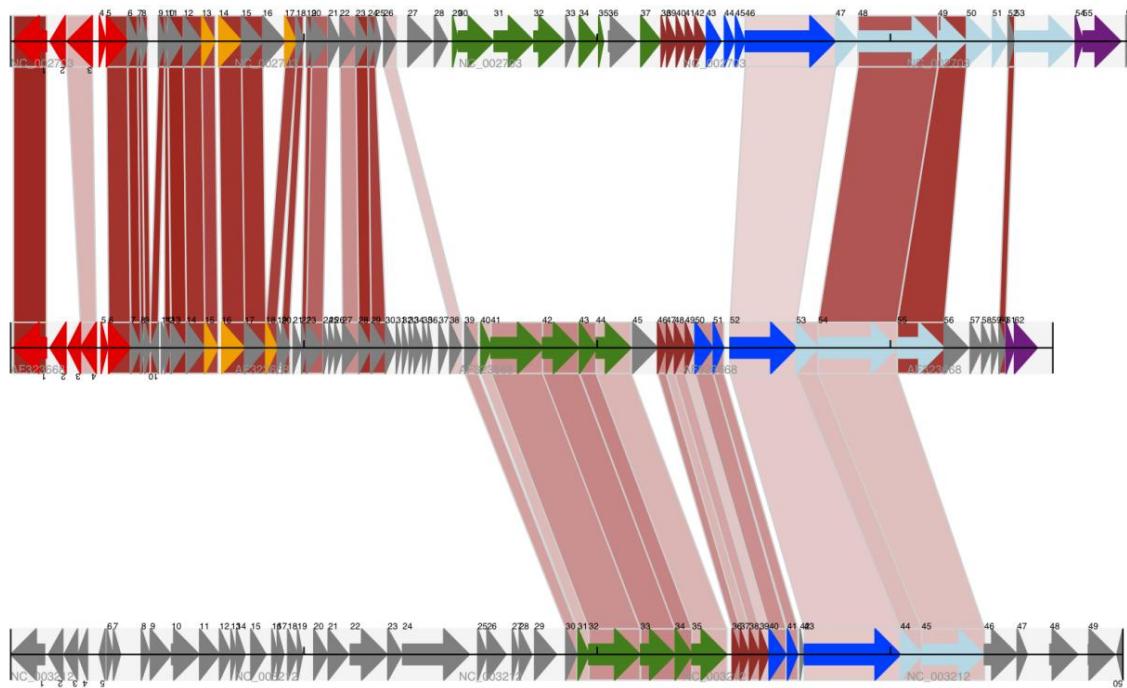


Şə kil 19.10: Lactococcus fage Tuc2009 (NC 002703), bakteriofaq bIL285 (AF323668) və Listeria innocua Clip11262 (NC 003212) 5-ci profaq üçün üç yollu xə tti diaqram, üstə gə lə sas kə sışmə faizi **19.1.11**.

Bu rə qə mi yaxşılaşdırmaq üçün Biopython daxilində hə lə çox şey edilə bilə r. Hə r şeydə nə vvə l, bu və ziyyə tdə çarpaz bağıntılar xüsusi bir zə ncirdə çə kilmış zülallar arasındadır. O, çarpaz keçidi genişlə ndırma k üçün funksiya trekinə arxa plan bölgə sini ('BOX' işarə sində n istifadə edə n xüsusiyyə tə lavə etmə ya kömə k edə bilə r. Hə mçinin, biz funksiya treklə rının şaquli hündürlüğünü azaltmaq və zinə linklə rə daha çox yer ayıra bilə rik – bunun bir yolu boş treklə r üçün yer ayırmadır. Bundan ə lavə , böyük gen üst-üstə düşmə sinin olmadığı bu kimi hallarda, trek üçün lazım olan şaquli mə kanı daha da azaltmağa imkan verə n ox-straddling BIGARROW sigilində n istifadə edə bilə rik. Bu tə kmillə şdirmə lə r Proux et al 2002 Şə kil 6.py skript nümunə sində nümayiş etdirilir Biopython mə nbə kodunun Doc/examples qovluğuna daxil edilmişdir. Gözlə nilə n mə hsul Şə kil 19.11-də göstə rilmişdir .

Bundan ə lavə , vektor şə kil redaktörunda ə l ilə etmə k istə yə bilə cə yiniz son toxunuşlara incə tə nzimlə mə daxildir gen etiketlə rinin yerlə şdirilmə si və xüsusi bölgə lə rin vurğulanması kimi digə r xüsusi annotasiyaların ə lavə edilmə si.

Çarpaz keçidlə rin heç biri üst-üstə düşmə diyi üçün bu nümunə də hə qıqə tə n zə ruri olmasa da, ReportLab-da şə ffaf rə ngdə n istifadə çoxsaylı keçidlə rin üst-üstə qoyulması üçün çox faydalı bir texnikadir. Lakin bu halda a



Şəkil 19.11: Lactococcus fage Tuc2009 (NC 002703), bakteriofaq bIL285 (AF323668) və Listeria innocua Clip11262 (NC 003212) 5-ci profaq üçün üç yollu xətti diaqramı **19.1.11**.

kölgə li rə ng sxemində n qaćınmaq lazımdır.

### 19.1.12 Əlavə variantlar

Siz miqyası göstərmək üçün işarələrə nəzarət edə bilərsiniz - axırda hər bir qrafik öz vahidlərinin və boz yol etiketlərinin sayını göstərməlidir.

Həmçinin, biz indiyə qədər yalnız FeatureSet-dən istifadə etmişik. GenomeDiagram-da həmçinin xətt qrafikləri, bar diaqramları və istilik qrafikləri üçün istifadə edilə bilər GraphSet də var (məsələn, funksiyalara paralel trekdə GC% sahələrin göstərmək üçün).

Bu seçimlərlə hələ burada əhatə olunmayıb, ona görə də hələlik sizi [İstifadəçi Təlimatına \(PDF\)](#) istinad edirik. İlaç daxildir GenomeDiagram-in müstəqil versiyası (lakin əvvəlcə növbəti bölməni oxuyun) və docstrings.

### 19.1.13 Köhnə kodun çevrilməsi

GenomeDiagram-in müstəqil versiyasından istifadə edərək yazılmış köhnə kodunuz varsa və onu Biopython-a daxil olan yeni versiyadan istifadə etmək istəyirsinizsə, onda siz bir neçə dəyişiklik etməli olacaqsınız - ənənəsini idxal bəyanatlarına.

Həmçinin, GenomeDiagram-in köhnə versiyasında yalnız İngiltərədəki rəng və mərkəzin (rəng və mərkəz) hər rüflə rindən istifadə olunurdu. Bir neçə il GenomeDiagram-in Biopython versiyası hər ikiini dəstəkləsə də, Amerika yazımlarına keçməli olacaqsınız.

Məsələn, əgər sizdə əvvəllər varsa:

[GenomeDiagram -dan GDFeatureSet, GDDiagram idxalı](#)

```
gdd = GDDiagram("Nümunə")
...
```

idxal ifadələrini belə dəyişə bilərsiniz:

[Bio.Graphics.GenomeDiagram- dan FeatureSet-i GDFeatureSet kimi, Diaqramı GDDiagram kimi idxal edin](#)

```
gdd = GDDiagram("Nümunə")
...
```

və ümidi edirəm ki, bu kifayətdir. Uzunmüddəli perspektivdə siz yeni adlara keçmək istəyə bilərsiniz, lakin kodunuzun daha çoxunu dəyişməli olacaqsınız:

[Bio.Graphics.GenomeDiagram - dan idxal FeatureSet, Diaqram](#)

```
gdd = Diaqram ("Nümunə")
...
```

və ya:

[Bio.Graphics idxal GenomeDiagram](#)

```
gdd = GenomeDiagram.Diagram("Nümunə")
...
```

Çətinliklə karşılaşsanız, məsləhət üçün Biopython poçt siyahısından soruşun. Bir məqam ondan ibarətdir ki, biz hələ köhnə GenomeDiagram.GDUtilities modulunu daxil etməmişik. Buraya, ehtimal ki, daha sonra Bio.SeqUtils altında birləşdiriləcək bir sıra GC% ilə əlaqəli funksiyalar daxildir.

## 19.2 Xromosomlar

Bio.Graphics.BasicChromosome modulu xromosomların çə kilmə sinə imkan verir. Jupe və başqalarında bir nümunə var. (2012) [22] (açıq giriş) müxtəlif gen ailələrinin vurğulamaq üçün rənglərdən istifadə edir.

### 19.2.1 Sadə Xromosomlar

Budur çox sadə bir nümunə - bunun üçün Arabidopsis thaliana istifadə edəcəyik.

Siz bu biti ötürə bilərsiniz, amma və ləmən NCBI-nin FTP saytından [ftp://ftp.ncbi.nlm.nih.gov/genomes/archive/old\\_refseq/Arabidopsis\\_thaliana/](ftp://ftp.ncbi.nlm.nih.gov/genomes/archive/old_refseq/Arabidopsis_thaliana/)-dan beş fərdi FASTA faylı kimi beş ardıcıl xromosomu endirdim. Və sonra uzunluqlarını öyrənmək üçün onları Bio.SeqIO ilə təhlil etdi. Bunun üçün GenBank fayllarından istifadə edə bilərsiniz (və növbəti misalda xromosomların planlaşdırılması üçün istifadə olunur), lakin istədiyiniz uzunluqdursa, bütün xromosomlar üçün FASTA fayllarından istifadə etmək daha sürətlidir:

**Bio idxaldan SeqIO**

```
qeydlər r = [
 ("Chr I", "CHR_I/NC_003070.fna"), ("Chr II", "CHR_II/
 NC_003071.fna"), ("Chr III", "CHR_III/NC_003074.fna"),
 ("Chr IV", "CHR_IV/NC_003070"), ("Chr"), ("Chr. "CHR_V/
 NC_003076.fna"),
```

```
] ad, girişlər rda fayl adı üçün : qeyd =
SeqIO.read(fayl adı, "fasta") çap (ad, len(record))
```

Bu, indi BasicChromosom modulunun aşağıdakı qısa nümayişində istifadə edəcəyiz biz beş xromosomun uzunluqlarını verdi:

**reportlab.lib.unit -dən sm-ni Bio.Graphics -dən idxal**  
BasicChromosom

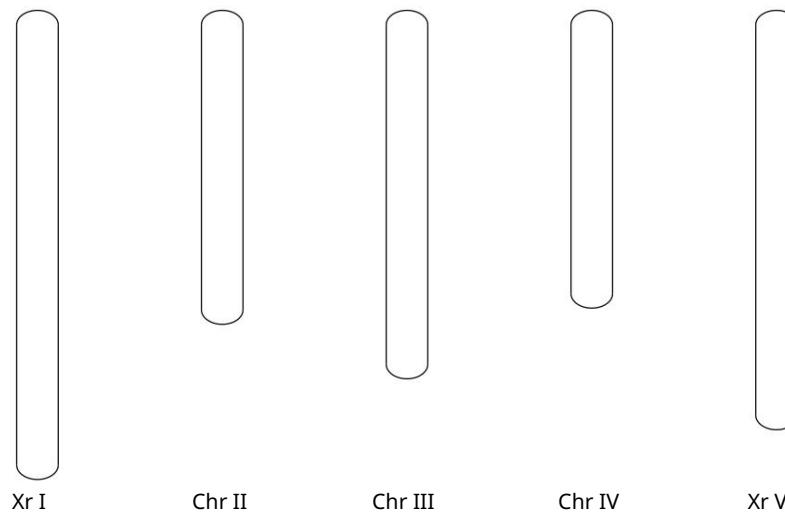
```
qeydlər r = [
 ("Chr I", 30432563),
 ("Chr II", 19705359),
 ("Chr III", 23470805),
 ("Chr IV", 18585042),
 ("Chr V", 26992728),
]
```

```
max_len = 30432563 # Bunu qeydlərda hesablamaq olar dict telomere_length = 1000000 # Təsvir
für
```

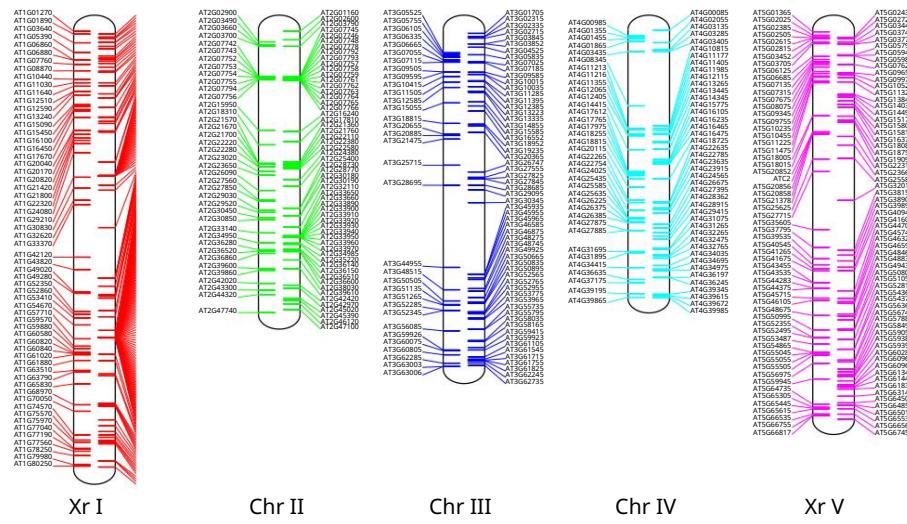
```
chr_diagram = BasicChromosome.Organism()
chr_diagram.page_size = (29.7 * sm, 21 * sm) # A4 landscape
```

```
ad, girişlər rda uzunluq üçün :
cur_chromosome = BasicChromosome.Chromosome(name)
Şkalanı MAKSIMUM uzunluğa üstə gələndə iki telomera təyin edin, # bütün beş xromosomda aynı
miqyasdan istifadə etmək istəyir ki, onlar # bir-biri ilə müqayisə olunsun cur_chromosome.scale_num
= max_len + 2 * telomere_length
```

## Arabidopsis thaliana

Şəkil 19.12: *Arabidopsis thaliana* üçün sadə xromosom diaqramı.

## Arabidopsis thaliana

Şəkil 19.13: tRNA genlə rini göstərən *Arabidopsis thaliana* üçün xromosom diaqramı.

```

Açılış telomer başlanlığını ə lava
edin = BasicChromosome.TelomereSegment() start.scale =
telomere_length cur_chromosome.add(start)

Bə də n ə lava edin - burada miqyas uzunluğu kimi bp istifadə
edin.bə də n = BasicChromosome.ChromosomeSegment()
body.scale = length
cur_chromosome.add(body)

Bağlanan telomer ucu ə lava
edin = Ə sas Xromosom. Telomer Seqmenti (ters çevrilmiş=Doğru)
end.miqyas = telomer_uzunluğu
cur_chromosome.add(son)

Bu xromosom tamamlandı
chr_diagram.add(cur_xromosom)

chr_diagram.draw("simple_chrom.pdf", "Arabidopsis thaliana")

```

Bu, Şəkil 19.12-də göstərilən çox sadə PDF faylı yaratmalıdır. Bu misal qəsdən qıсадır və şirin. Növbəti nümunə maraqlı xüsusiyyətlərin yerini göstərir.

### 19.2.2 Annotasiyalı Xromosomlar

Əvvəlki nümunədən davam edərək tRNNT genlərinin göstərək. Beş Arabidopsis thaliana xromosomu üçün GenBank fayllarını təhlil edərək onların yerlərinin əldə edəcəyi yik. Siz bu faylları NCBI FTP saytından yükleyə bilisiniz [ftp://ftp.ncbi.nlm.nih.gov/genomes/archive/old\\_refseq/Arabidopsis\\_thaliana/](ftp://ftp.ncbi.nlm.nih.gov/genomes/archive/old_refseq/Arabidopsis_thaliana/), və alt kataloq adlarını qoruyun və ya aşağıdakı yolları redaktə edin:

```

reportlab.lib.unit -dən idxal sm Bio- dan idxal
SeqIO- dan Bio.Graphics- dən
idxal BasicChromosome

qeydlə r = [
 ("Chr I", "CHR_I/NC_003070.gbk"), ("Chr II",
 "CHR_II/NC_003071.gbk"), ("Chr III", "CHR_III/
 NC_003074.gbk"), ("Chr IV", "CHR_IV/NC_0030",
 "CHR_IV/ NC_0030"), ("CHR_V/NC_003076.gbk"),
]

max_len = 30432563 # Bunu qeydlərdən hesablamaq olar dict telomere_length = 1000000
Təsvir üçün

chr_diagram = BasicChromosome.Organism()
chr_diagram.page_size = (29.7 * sm, 21 * sm) # A4 landşaft

indeks üçün, (ad, fayl adı) sıralamada (girişlərdən):
rekord = SeqIO.read(fayl adı, "genbank") uzunluq = len
(qeyd)

```

xüsusiyyə tlə r = [ qeyddə f üçün f.f. tipli xüsusiyyə tlə r == "tRNA"]  
 # Xüsusiyyə tin seçicilə rində Artemis üslubunda tam ə də d rə ngini qeyd edin, # 1 = Qara, 2 = Qırmızı, 3 = Yaşıl, 4 = mavi,  
 5 = mavi, 6 = xüsusiyyə tlə rdə f üçün bə növşə yi :

```
f.qualifiers["rə ng"] = [indeks + 2]
```

```
cur_chromosome = Ə sas Xromosoma.Xromosom(ad)

Şkalanı MAKŞİMUM uzunluğa üstə gə l bp-də iki telomera tə yin edin, # bütün beş xromosomda

eyni miqyasdan istifadə etmə k istə yirik ki, onlar # bir-biri ilə müqayisə olunsun

cur_chromosome.scale_num =

max_len + 2 * telomere_length
```

```
Açılış telomer başlanğıcını ə lavə

edin = BasicChromosome.TelomereSegment() start.scale =

telomere_length cur_chromosome.add(start)
```

```
Gövdə ə lavə edin - yenə burada miqyas uzunluğu kimi bp istifadə edin. gövdə

= BasicChromosome.AnnotatedChromosomeSegment(uzunluq, xüsusiyyə tlə r) body.scale = length

cur_chromosome.add(body)
```

```
Bağlanan telomer ucu ə lava edin

= Ə sas Xromosom. Telomer Seqmenti (ters çevrilmiş=Doğru) end.miqyas =

telomer_uzunluğu cur_chromosome.add(son)
```

```
Bu xromosom tamamlandı

chr_diagram.add(cur_xromosome)
```

```
chr_diagram.draw("tRNA_chrom.pdf", "Arabidopsis thaliana")
```

Bu, etiketlə rin bir-birinə çox yaxın olması barə də sizi xə bə rdar edə bilə r - irə li ipə nə zə r salın (sağda Chr I-in ə l tə rə fi), lakin o, Şəkil 19.12-də göstərilə nə rə ngli PDF faylı yaratmalıdır .

# 20-ci fə sil

## KEGG

KEGG (<https://www.kegg.jp/>) molekulyar sə viyyə li mə lumatlardan, xüsusə n genom ardıcılığı və digə r yüksə k mə hsuldarlıqli eksperimental texnologiyalarla yaradılan geniş miqyaslı molekulyar mə lumat də stlə rində n hüceyrə , orqanizm və ekosistem kimi bioloji sistemin yüksə k sə viyyə li funksiyalarını və kommunallarını anlamaq üçün verilə nlə r bazası resursudur.

Nə zə rə alın ki, Biopython-da KEGG parser tə tbiqi natamamdır. KEGG veb-sayıtı bir çox düz fayl formatlarını göstə rsə də , hazırda yalnız birlə şmə , ferment və xə ritə üçün tə hlilçilə r və yazılıclar tə tbiq olunur. Bununla belə , digə r formatları idarə etmə k üçün ümumi analizator tə tbiq olunur.

### 20.1 KEGG qeydlə rinin tə hlili

KEGG qeydini tə hlil etmə k Biopython-da hə r hansı digə r fayl formatı analizatorundan istifadə etmə k qə də r sadə dir. (Aşağıdakı kodları işə salmazdan ə vvə l <http://rest.kegg.jp/get/ec:5.4.2.2> ünvanını açın. veb brauzerinizlə və onu ec\_5.4.2.2.txt olaraq qeyd edin.)

```
>>> Bio.KEGG -də n idxal fermenti >>> qeydla ri =
Enzyme.parse(open("ec_5.4.2.2.txt")) >>> rekord = siyahı(qeydlə r)[0] >>> rekord.classname
['Isomerases;', 'Intramolecular transferases;',
 'Fosphotransferases;',
 'Fosphotransferases;', 'Fosphotransferases;]phosphotransferases (rekord) phosphases . '5.4.2.2'
```

Alternativ olaraq, giriş KEGG faylında tam olaraq bir giriş varsa, oxumaqdən istifadə edə bilə rsiniz:

```
>>> Bio.KEGG -də n idxal fermenti >>> rekord =
Enzyme.read(open("ec_5.4.2.2.txt")) >>> record.classname
['İzomerazlar;', 'Molekuladxılı transferazlar;', 'Fosfotransferazlar (fosfomutazlar)'] >>> qeyd.giriş '5.4.2.2'
```

Aşağıdakı bölmə KEGG api-də n istifadə edə rə k yuxarıdakı fermentin necə endirilmə sini və necə yükə nə cə yini göstə rə cə kdir hə yata keçirilə n xüsusi analizatoru olmayan verilə nlə rlə ümumi analizatordan istifadə etmə k.

### 20.2 KEGG API sorğusu

Biopython KEGG api-nin sorğulanması üçün tam də stə yə malikdir. Bütün KEGG son nöqtə lə rinin sorğulanması də stə klə nir; KEGG tə rə fində n sə nə dlə şdirilmiş bütün üsullar (<https://www.kegg.jp/kegg/rest/keggapi.html>) də stə klə nir. The

interfeys KEGG saytında müə yyə n edilmiş qaydalara ə mə l edə n sorğuların bə zi tə sdiqinə malikdir. Bununla belə , 400 və ya 404 qaytaran etibarsız sorğular istifadə ci tə rə fində n idarə olunmalıdır.

Birincisi, yuxarıdakı nümunə ni müvafiq fermenti yüklə mə k və onu ötürmə klə necə genişlə ndirmə k olar ferment analizatoru vasitə silə .

```
>>> Bio.KEGG -də n idxal REST >>> Bio.KEGG
-də n idxal Ferment >>> sorğu =
REST.kegg_get("ec:5.4.2.2") >>> open("ec_5.4.2.2.txt",
"w").write(request.read()) >>> qeydlə r = Enzyme.parse(açıq.2)(rekord > _ .) " 2
siyahı(qeydlə r)[0]

>>> rekord.classname
['İzomerazlar', 'Molekulda xili transferazlar', 'Fosfotransferazlar (fosfomutazlar)'] >>> qeyd.giriş '5.4.2.2'
```

İndi burada KEGG API sorğusunun birlə şmə sini göstə rə n daha real bir nümunə var. Bu, DNT tə miri ilə ə laqə li bütün insan yolu gen simvollarının unikal də stinin necə çıxarılacağını nümayiş etdirə cə kdir.

Bunun üçün atılmalı olan addımlar aşağıdakılardır. Birincisi, bütün insan yollarının siyahısını ə ldə etmə liyik. İkincisi, "tə mir"ə aid olanları süzgə cdə n keçirmə liyik. Nə həyə t, bütün tə mir yollarında bütün gen simvollarının siyahısını ə ldə etmə liyik.

#### **Bio.KEGG -də n idxal REST**

```
human_pathways = REST.kegg_list("yol", "hsa").read()

Tə mir yolları üçün bütün insan yollarını düzgün repair_pathways = []
human_pathways.rstrip().split("\n")
daxilində ki xə tt üçün): giriş, tə svir = line.split("\t") ə gə r "tə mir"
tə svirdə :

tə mir_yolları.append(giriş)

Yollar üçün genlə ri ə ldə edin və onları tə mir_yollarında yol üçün
tə mir_genlə ri = []
siyahısına ə lavə edin : pathway_file =
 REST.kegg_get(pathway).read() # sorğu və hə r yolu oxuyun

 # faylin hansı bölmə sində olduğumuzu # izlə yə rə k hə r bir KEGG yol faylini tə krarlayın, yalnız hə r bir yoldakı
 geni oxuyun current_section = pathway_file.rstrip().split("\n") xə ttində heç biri yoxdur :

bölmə = line[:12].strip() # bölmə adları bölmə deyilsə , 12 sütun daxilində dir == "": cari_bölmə = bölmə

ə gə r cari_bölmə == "GEN":
 gen_identifikatorları, gen_tə sviri = xə tt[12:].split("; ") gen_id, gen_simvolu =
 gen_identifikatorları.split()

 tə mir_genlə rində gen_simvolu deyilsə :
 tə mir_genlə ri.append(gen_simvolu)
```

```
print("
 %d tə mir yolu və %d tə mir geni var . Genlə r bunlardır:" % (len(repair_pathways),
 len(repair_genes))

) print(", ".qoşulmaq(repair_genes))
```

KEGG API paketi bütün son nöqtə lə rə uyğundur. İstifadə mahiyyə tcə url-də ki bütün kə sik işaretə lə rini vergüllə rlə  
ə və z etmə k və hə min siyahıdan KEGG modulunda müvafiq metoda arqument kimi istifadə etmə kdir.

Budur api sə nə dlə rində n bir neçə nümunə (<https://www.kegg.jp/kegg/docs/keggapi.html>).

|                                     |                                                         |
|-------------------------------------|---------------------------------------------------------|
| /list/hsa:10458+ece:Z5100 /tap/     | -> REST.kegg_list(["hsa:10458", "ece:Z5100"])           |
| mürə kkə b/300-310/mol_weight /get/ | -> REST.kegg_find("mürə kkə b", "300-310", "mol_çə ki") |
| hsa:10458+ece:Z5100/aaseq           | -> REST.kegg_get(["hsa:10458", "ece:Z5100"], "aaseq")   |

# 21-ci fə sil

## Bio.fenotip: fenotipik mə lumatları tə hlil edin

Bu fə sildə Biopythona daxil olan Bio.phenotype paketinin funksiyaları haqqında ümumi mə lumat verilir.

Bu paketin ə hata dairə si fenotipik mə lumatların tə hlilində n ibarə tdir ki, bu da hüceyrə mə də niyyə tlə rinin böyümə ölçmə lə rinin tə hlili və tə hlili demə kdir. Hazırkı və ziyyə tdə paket [Phenotype Microarray texnologiyası](#) ilə istehsal olunan yüksə k mə hsuldarlıqlı fenotipik tə crübə lə rin tə hlilinə yönə ldilmişdir. Lakin gə lə cə k inkişafalar başqa platformalar və formatlar da daxil ola bilə r.

### 21.1 Fenotipli mikroarraylar

Fenotip [Mikroarray](#) bakterial və eukaryotik hüceyrə lə rin metabolizmasını tə xminə n 2000 kimyə vi maddə üzə rində ölçü n texnologiyadır, iyirmi 96 quyu boşqablara bölünür. Texnologiya hüceyrə tə rə fində n istehsali hüceyrə metabolizması üçün proksi kimi istifadə edilə n NADH tə rə fində n tetrazolium boyasının azaldılmasını ölçür; bu boyanın azalması sə bə bində n rə ng inkişafi adə tə n hə r 15 də qıqə də bir də fə ölçülür. Hüceyrə lə r hüceyrə mübadilə sinə tə min edə n mühitdə böyüdükdə , qeydə alınmış fenotipik mə lumatlar bir sıra böyümə parametrlə rini ə ldə etmə k mümkün olan sigmoid böyümə ə yrisinə bə nzə yir.

#### 21.1.1 Phenotype Microarray verilə nlə rinin tə hlili

Bio.phenotype paketi Phenotype Microarray mə lumatlarının iki müxtə lif formatını tə hlil edə bilə r: [CSV](#) (vergüllə ayrılmış də yə rlə r) maşının xüsusi program tə minati və [JSON](#) tə rə fində n hazırlanmış fayllar [opm](#) kimi analiz programı tə rə fində n hazırlanmış fayllar və ya [DuctApe](#). Parser oxu və ya tə hlil metodunun istifadə edilib-edilmə mə sində n asılı olaraq PlateRecord obyektlə rinin birini və ya generatorunu qaytaracaq. [Plates.csv](#)- də n istifadə edə rə k tə hlil funksiyasını sınaya bilə rsiniz fayl Biopython mə nbə kodu ilə tə min edilir.

```
>>> Bio idxl fenotipində n >>>
phenotype.parse ("Plates.csv", " pm - csv"); print("%s %i" % (record.id,
... len(record)))
...
PM01 96
PM01 96
PM09 96
PM09 96
```

Parser, hə r biri 8 cə rgə də və 12 sütunda düzülmüş bir sıra WellRecord obyektlə rini (hə r quyunun eksperimental mə lumatlarını saxlayan) ehtiva edə n bir sıra PlateRecord obyektlə rini qaytarır; hə r bir sıra a ilə göstə rilir

A-dan H-ə qə də r böyük hə rf, sütunlar isə 01-də n 12-yə qə də r iki rə qə mli rə qə mla göstə rildiyi halda. PlateRecord obyektlə rində n WellRecord obyektlə rinə daxil olmağın bir neçə yolu var:

Quyu identifikatoru Ə gə r quyu identifikatorunu (sə tir + sütun identifikatorları) bilirsınızsə , istə diyiniz quyuya daxil ola bilə rsiniz. birbaşa.

```
>>> qeyd["A02"]
```

```
WellRecord([(0.0, 12.0), (0.25, 18.0), (0.5, 27.0), (0.75, 35.0), (1.0, 37.0), ..., (71.75, 143.0)
```

Quyu boşqab koordinatları Eyni quyu cə rgə və sütun nömrə lə rində n (0 ə saslı) istifadə etmə klə götürülə bilə r. indeks).

```
>>> Bio idxal fenotipində n >>> rekord =
```

```
siyahı(phenotype.parse("Plates.csv", "pm-csv"))[-1] >>> çap (record[0, 1].id)
```

```
A02
```

Satır və ya sütun koordinatları PlateRecord obyektlə rində python siyahısı dilimlə mə sintaksisində n istifadə etmə klə lövhə də bir-birinə bitişik olan bir sıra WellRecord obyektlə ri toplu olaraq ə ldə edilə bilə r; sə tir və sütunlar 0 ə saslı indekslə nömrə lə nir.

```
>>> çap (qeyd [0])
```

```
Plitə ID: PM09 Quyu: 12
```

```
Sira: 1 Sütun:
```

```
12
```

```
PlateRecord('WellRecord['A01'], WellRecord['A02'], WellRecord['A03'], ..., >>> çap (qeyd[:, 0])
```

```
WellRecord['A12'])
```

```
Nömrə ID: PM09
```

```
Yaxşı: 8
```

```
Sıralar: 8
```

```
Sütunlar: 1
```

```
PlateRecord('WellRecord['A01'], WellRecord['B01'], WellRecord['C01'], ..., >>> çap (qeyd[:, :3]))
```

```
WellRecord['H01'])
```

```
Nömrə ID: PM09
```

```
Yaxşı: 9
```

```
Sıralar: 3
```

```
Sütunlar: 3
```

```
PlateRecord('WellRecord['A01'], WellRecord['A02'], WellRecord['A03'], ...,
```

```
WellRecord['C03'])
```

## 21.1.2 Phenotype Microarray verilə nlə rının manipulyasiyası

### 21.1.2.1 Xam mə lumatlara giriş

PM fayllarından çıxarılan xam mə lumatlар vaxt (saatlarla) və kolorimetrik ölçüdə n (ixtiyari vahidlə rlə ) ibarə t hə r bir quyu üçün bir sıra tuplelə rdə n ibarə tdir. Adə tə n alə t hə r on beş də qıqə də n bir mə lumat toplayır, lakin bu, tə crübə lə r arasında də yişə bilə r. Xam mə lumatlara WellRecord obyektiində iterasiya yolu ilə daxil olmaq olar; aşağıdakı nümunə də yalnız ilk on zaman nöqtə si göstə rilir.

```
>>> Bio idxal fenotipində n >>> qeyd =
```

```
siyahı(phenotype.parse("Plates.csv", "pm-csv"))[-1] >>> quyu = qeyd["A02"]
```

```
>>> vaxt üçün, quyuda siqnal : çap (vaxt,
... siqnal)
...
(0,0, 12,0) (0,25,
18,0) (0,5, 27,0)
(0,75, 35,0) (1,0,
37,0) (1,25, 41,0)
(1,5, 44,0) (1,75,
44,0) (2,4,0) (2,4)
[...]
```

Bu üsul, xam mə lumatlara daxil olmaq üçün bir yol tə qdim edə rkə n, birbaşa müqayisə yə imkan vermir müxtə lif vaxt nöqtə lə rində ölçmə lə ri ola bilə n müxtə lif WellRecord obyektlə ri.

#### 21.1.2.2 İnterpolyasiya edilmiş verilə nlə rə daxil olmaq

Müxtə lif eksperimentlə ri müqayisə etmə yi asanlaşdırmaq və ümmumilikdə fenotipik mə lumatların daha intuitiv idarə edilmə sinə imkan vermə k üçün modul WellRecord obyektiñdə mövcud olan vaxt nöqtə lə rinin fə rdi dilimlə nmə sini tə yin etmə yə imkan verir. Birbaşa ölçüləməmiş vaxt nöqtə lə ri üçün kolorimetrik mə lumatlar mövcud mə lumatların xə tti interpolyasiyası vasitə silə ə ldə edilir, ə ks halda NaN qaytarılır. Bu üsul yalnız faktiki mə lumatların mövcud olduğu vaxt intervalında işlə yir. Vaxt intervalları siyahının indekslə şdirilmə si ilə eyni sintaksislə müə yyə n edilə bilə r; standart vaxt intervalı buna görə də bir saatdır.

```
>>> quyu[:10] [12.0,
37.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0]
```

Müxtə lif vaxt intervalları istifadə edilə bilə r, mə sə lə n, beş də qiqə :

```
>>> quyu[63:64:0.083] [12.0, 37.0,
44.0, 44.0, 44.0, 44.0, 44.0, 44.0] >>> quyu[9,55]
```

```
44.0
>>> quyu[63.33:73.33]
[113.3199999999999, 117.0,
120.3199999999999, 128.0,
129.6399999999999,
1392.999995
136.95999999999998, 140.0,
142.0,
nan]
```

#### 21.1.2.3 Quyunun çıxarılmasına nə zarə t

Bir çox Phenotype Microarray plitə lə rində nə zarə t quyusu (adə tə n A01) var, bu, medianın hə r hansı artımı də stə klə mə mə li olduğu quyudur; bu quyunun istehsal etdiyi aşağı siqnal digə r quyulardan çıxarıla bilə r. PlateRecord obyektlə rinin bunun üçün xüsusi funksiyası var ki, bu da düzə ldilmiş mə lumatlarla başqa PlateRecord obyektiñi qaytarır.

```
>>> düzə ldildi = qeyd.subtract_control(control="A01") >>> qeyd["A01"][[63]
```

336.0

```
>>> düzə ldildi["A01"][[63]
```

0.0

#### 21.1.2.4 Parametrlə rin çıxarılması

Metabolik aktivliyin müşahidə olunduğu quyular kolorimetrik mə lumatlar üçün sigmoid davranış nümayiş etdirir. Müxtə lif eksperimentlə ri daha asan müqayisə etmək üçün verilənlərə bir sigmoidə yeri quraşdırıla bilər ki, bir sıra xülasə parametrləri çıxarılsın və müqayisə lər üçün istifadə olunsun. Əyridən çıxarıla bilən parametrlər bunlardır:

- Minimum (min) və maksimum (maksimum) siqnal;
- Orta boy (orta boy);
- Ə yri altındakı sahə (sahə);
- Ə yri yayla nöqtəsi (yayla);
- Eksponensial metabolik aktivlik zamanı ə yri mailliyi (maili);
- Ə yri gecikmə vaxtı (lag).

Bütün parametrlər (min, max və orta hündürlükdən başqa) [scipy kitabxanasını](#) tələb edir quraşdırılmalıdır. Fit funksiyası üç sigmoid funksiyasından istifadə edir:

$$\text{Gompertz Ae-e} \quad \left( \frac{\mu_m}{A} - (\lambda - t) + 1 \right) + y_0$$

$$\text{Logistika} \quad \frac{A}{\left( \frac{\mu_m}{A} + 2 \right) 1 + e} + y_0$$

$$\text{Richards A} (1 + ve1 + v + e)^{\frac{\mu_m}{A} - (1+v)(1 + \frac{1}{v})} (\lambda - t)^{-\frac{1}{v}} + y_0$$

Harada:

A yayasına uyğun gəlir

$\mu_m$  yamacı uyğundur

$\lambda$  gecikmə yə uyğundur

Bu funksiyalar bu nə şrdən götürülüb. Varsayılan olaraq uyğunlaşdırma metodu əvvəlcə gompertz funksiyasını uyğunlaşdırmağa çalışır: uğursuz olarsa, logistika və sonra Richards funksiyasına uyğunlaşmağa çalışacaq. İstifadəçi həmçinin tətbiq olunacaq üç funksiyadan birini təyin edə bilər.

```
>>> Bio import fenotipindən >>> rekord
= siyahı(phenotype.parse("Plates.csv", "pm-csv"))[-1] >>> well = record["A02"] >>> well.fit()
>>> print("Funksiya quraşdırılıb:
%$%" % well.model)
```

Funksiya quraşdırılıb: gompertz >>>

```
["sahə", "orta_hündürlük", "lag", "maks", "min", "plato", "mail"] üçün param üçün : print("%s\t%.2f" % (param, getattr(yaxşı,
... param)))
...
sahə 4414.38
```

orta\_hündürlük 61.58 lag 48.60

maks 143.00

min 12.00

yayla 120.02 yamac

4.99

#### 21.1.3 Phenotype Microarray datasının yazılması PlateRecord

objektlə ri **JSON** formasında fayla yazılı bilə r. faylları, **opm** kimi digə r program paketlə ri ilə uyğun bir format və ya **DuctApe**.

```
>>> phenotype.write(qeyd, "out.json", "pm-json")
```

1

## 22-ci fə sil

### Yemə k də ftə ri - Onunla etmə k üçün gözə l şeylə r

Biopython-da indi iki "yemə k kitabı" nümunə lə ri kolleksiyası var – bu fə sil (uzun illə rdir ki, bu də rsliyə daxil edilib və getdikcə genişlə nır) və [http://biopython.org/wiki/Category:Kitab\\_kitabi](http://biopython.org/wiki/Category:Kitab_kitabi) Bu, bizim viki də istifadə çinin töhfə verdiyi kolleksiyadır.

Biz Biopython istifadə çilə rini vikiyə öz nümunə lə rini tə qdim etmə yə tə şviq etmə yə çalışırıq. İcmaya kömə k etmə klə yanaşı, belə bir nümunə ni paylaşmanın birbaşa faydalardan biri də odur ki, digə r Biopython istifadə çilə ri və tə rtibatçılarından kod haqqında bə zi rə ylə rə ldə edə bilə rsiniz - bu, bütün Python kodunuza tə kmillə şdirmə yə kömə k edə bilə r.

Uzunmüddə tli perspektivdə bu fə sildə ki bütün nümunə lə ri vikiyə və ya başqa yerə köçürə bilə rik də rslik çə rçivə sində .

#### 22.1 Ardıcılıq faylları ilə işlə mə k

Bu bölüm 5-ci fə sildə tə svir edilə n Bio.SeqIO modulundan istifadə edə rə k ardıcıl giriş/çıxış nümunə lə rini göstə rir .

##### 22.1.1 Ardıcılıq faylinin süzgə cdə n keçirilmə si Çox vaxt

sizdə çoxlu ardıcılıqlar olan böyük fayl (mə sə lə n, FASTA faylı və ya genlə r və ya oxunanların FASTQ və ya SFF faylı), maraq doğuran ardıcılıqların alt çoxluğu üçün ID-lə rin ayrıca daha qısa siyahısı olacaq və bu alt çoxluq üçün yeni ardıcılıq faylı yaratmaq istə yirsınız.

Deyə k ki, identifikatorların siyahısı hə r sə tirdə ilk söz kimi sadə mə tn faylındadır. Bu, birinci sütunun ID olduğu cə dvə l faylı ola bilə r. Bu kimi bir şey cə hd edin:

**Bio idxaldan SeqIO**

```
input_file = "big_file.sff" id_file =
"short_list.txt" output_file = "short_list.sff"

id_handle kimi open(id_file) ilə :
 İstə nilə n = set(line.rstrip("\n").split(Yox, 1)[0]) id_handle -də xə tt üçün)
 print(" % s-də %i unikal identifikator tapıldı" % (len(istə nilir), id_file))

qeydlə r = (SeqIO.parse -də r üçün r (input_file, "sff") a gə r r.id istə nirsə) count = SeqIO.write(records,
output_file, "sff") print("Saxlanılan %i qeydlə r %s- də n %s" % (count,
input_file, output_file))
```

```

if count < len(istedilir):
 print("Xə bə rdarlıq %i identifikatorları %s-də tapılmadı" % (len(istə nilir) - count, input_file))

Qeyd edə ki, biz siyahıdan çok Python də stində n istifadə edirik, bu, üzvlük testini daha sürə tli edir.
Bölmə 5.6-da müzakirə edildiyi kimi, sərət üçün böyük FASTA və ya FASTQ faylı üçün yüksək sə viyyə li SeqIO interfeysində n istifadə etmə sə niz, birbaşa sə tirlə rə işlə sə niz daha yaxşı olar. Bu növbəti nümunə bunu FASTQ faylları ilə necə edə cə yinizi göstərir – bu daha mürəkkəbdir:

Bio.SeqIO.QualityIO idxali FastqGeneralIterator- dan

input_file = "big_file.fastq" id_file =
"short_list.txt" output_file =
"short_list.fastq"

id_handle kimi open(id_file) ilə : # İdentifikator
 kimi hər sa tirdə ilk sözün alınması tələb olunur =
 set(line.rstrip("\n").split(None, 1)[0] üçün id_handle)
 print(" %s-də %i unikal identifikator tapıldı" % (len(istə nilir), id_file))

open(input_file) in_handle kimi : open(output_file, "w")
 with out_handle: başlıq, seq, qual üçün FastqGeneralIterator(in_handle) :

 # ID başlıq sə tirində ki ilk sözdür (@ işarə sindən sonra): e gər başlıq.split(Yox, 1)[0] istə nilirsə :
 # Bu standart 4 sə tırlik FASTQ girişini yaradır:
 out_handle.write("@%s\n%s\n+\n%s\n" % (başlıq, ardıcılıq, keyfiyyət))

 += 1 sayın
print("Saved %i qeydlə ri %s-də n %s-ə qədər" % (count, input_file, output_file)) if count < len(istedilən):
print("Xə bə rdarlıq %i
identifikatorları %s-də tapılmadı" % (len(istə nilir) - count, input_file))

```

## 22.1.2 Randomize genomların istehsalı

Tutaq ki, siz genom ardıcılığına baxırsınız, hansısa ardıcılıq xüsusiyyətini arxarırsınız – ola bilsin ki, həddindən artıq yerli GC% qərəziyi və ya mümkün məhdudiyyət həzər saytları. Python kodunuz real genom üzərində işlədikdən sonra statistik analiz üçün eyni genomun təsadüfi versiyalarında eyni arxası aparmağa cəhd etmək məqsədə uyğun ola bilər (axı, tapdığınız hər hansı "xüsusiyyətlər" təsadüfən orada ola bilər).

Bu müzakirə üçün biz Yersinia pestis biovar Microtus-dan pPCP1 plazmidi üçün GenBank faylından istifadə edəcəyik. Fayl GenBank qovluğu altında Biopython vahid testlərinə daxildir və ya siz onu [NC\\_005816.gb](#) vəbsaytımızdan indədə bilərsiniz. Bu fayl bir və yalnız bir qeyddən ibarətdir, ona görədə Bio.SeqIO.read() funksiyasından istifadə edərək onu SeqRecord kimi oxuya bilərik:

```
>>> Bio import SeqIO- dan >>>
original_rec = SeqIO.read("NC_005816.gb", "genbank")
```

Bələliklə, orijinal ardıcılığın qarışdırılmış versiyalarını necə yarada bilərik? Bunun üçün quraşdırılmış Python təsadüfi modulundan, xüsusən də random.shuffle funksiyasından istifadə edərdim – lakin bu, Python siyahısında işləyir. Ardıcılığımız Seq obyektidir, ona görədə onu qarışdırmaq üçün onu siyahıya çevirməliyik:

```
>>> idxal tə sadüfi >>>
nuc_list = list(original_rec.seq) >>> random.shuffle(nuc_list)
yerində işləyir!
```

İndi qarışdırılmış ardıcılığı çıxarmaq üçün Bio.SeqIO-dan istifadə etmək üçün bu qarışdırılmış siyahıdan istifadə edərək yeni Seq obyekti ilə yeni SeqRecord qurmalıyıq. Bunu etmək üçün nukleotidlərin siyahısını (tək hərfli sətirlər) uzun bir sətirə çevirməliyik – bunu etmək üçün standart Python yolu simobyektinin birləşmə üsuludur.

```
>>> Bio.Seq -dən idxal Seq >>>
Bio.SeqRecord -dan idxal SeqRecord >>> shuffled_rec =
SeqRecord(Seq("".join(nuc_list)),
... id="Shuffled", description="Based on %s" % original_rec.id
...)
```

Tək bir FASTA faylı yaradan tam Python skripti yaratmaq üçün bütün bu parçaları bir araya gətirək orijinal ardıcılığın 30 təsadüfi qarışdırılmış versiyasını ehtiva edir.

Bu ilk versiya sadəcə böyük for loopundan istifadə edir və qeydləri bir-bir yazır (SeqRecord-dan istifadə etməkla Bölmə 5.5.4-də təsvir olunan format metodu):

```
Bio.Seq -dən
təsadüfi idxal Seq- i Bio.SeqRecord
- dan idxal SeqRecord-u Bio-dan idxal SeqIO- dan
```

```
original_rec = SeqIO.read("NC_005816.gb", "genbank")

open("shuffled.fasta", "w") ilə output_handle kimi : üçün diapazon (30):
 nuc_list = list(original_rec.seq)
 random.shuffle(nuc_list) shuffled_rec = SeqRecord(
 Seq("".join(nuc_list)), id="Shuffled%i"
 % (i + 1), description="%s ə səsində"
 % original_rec.id ,
) output_handle.write(shuffled_rec.format("sürətlid"))
```

Şəxslən mən qeydi və generator ifadəsini qarışdırmaq funksiyasından istifadə edərək aşağıdakı versiyaya üstünlük verirəm for loop yerine :

```
Bio.Seq -dən
təsadüfi idxal Seq- i Bio.SeqRecord
- dan idxal SeqRecord-u Bio-dan idxal SeqIO- dan
```

```
def make_shuffle_record(record, new_id): nuc_list =
 list(record.seq) random.shuffle(nuc_list)
 return SeqRecord(Seq("".join(nuc_list)),
 id=new_id, description=
 "%s ə səsində" % original_rec.id ,
)
```

```

original_rec = SeqIO.read("NC_005816.gb", "genbank") shuffled_recs =
(make_shuffle_record(original_rec, "Shuffled%i" % (i + 1)) diapazondaki i üçün (30)
)

SeqIO.write(shuffled_recs, "qarışdırılmış.fasta", "fasta")

```

22.1.3 CDS qeydlə rinin FASTA faylinin tə rcümə si Tutaq ki, sizdə bə zi orqanizmlə r

üçün CDS qeydlə rinin giriş şaylı var və siz onların zülal ardıcılığını ehtiva edə n yeni FASTA şaylı yaratmaq istə yirsiniz. Yə ni orijinal fayldan hə r bir nukleotid ardıcılığını götürün və tə rcümə edin. Bölmə 3.8-də biz Seq obyektinin tə rcümə metodundan və alternativ başlanğıc kodonlarının düzgün tə rcümə sini tə min edə nə lavə cds arqumentində n necə istifadə edə ca yimizi gördük.

Biz bunu Bölmə 5.5.3-də ə ks tamamlayıcı nümunə də göstə rildiyi kimi Bio.SeqIO ilə birlə şdirə bilə rik. Ə sas mə qam ondan ibarə tdir ki, hə r bir nukleotid SeqRecord üçün biz SeqRecord zülalı yaratmalıyıq və onun adlandırılmasına diqqə t yetirmə liyik.

Ardıcılığınız üçün uyğun protein identifikatorlarını və uyğun genetik kodu seçə rə k bunu etmə k üçün öz funksiyanızı yaza bilə rsiniz. Bu nümunə də biz sadə cə standart cə dvə ldə n istifadə edirik və identifikatora prefiksə lavə edirik:

#### [Bio.SeqRecord- dan SeqRecord idxali](#)

```

def make_protein_record(nuc_record):
 """Tə rcümə edilmiş ardıcılıqla yeni SeqRecord qaytarır (defolt cə dvə l)."""
 SeqRecord(seq=nuc_record.seq.translate(cds=True), id="trans_" +
 nuc_record.id, description="CDS-in
 tə rcümə si, standart cə dvə ldə n istifadə etmə klə ",
)

```

Daha sonra giriş nukleotid qeydlə rini çıxış üçün hazır olan zülal qeydlə rinə çevirmə k üçün bu funksiyadan istifadə edə bilə rik. Bunu etmə k üçün zə rif bir yol və yaddaş sə mə rə li yolu generator ifadə sidir:

#### [Bio idxaldan SeqIO](#)

```

zülallar =
 (SeqIO.parse -də nuc_rec üçün
 make_protein_record(nuc_rec) ("coding_sequences.fasta", "fasta")
)
SeqIO.write(zülallar, "translations.fasta", "fasta")

```

Bu, tam kodlaşdırma ardıcılığının istə nilə n FASTA şaylı üzə rində işlə mə lidir. Ə gə r siz qismə n kodlaşdırma ardıcılığı üzə rində işlə yirsinzə , yuxarıdakı nümunə də nuc\_record.seq.translate(to\_stop=True) istifadə etmə yə üstünlük verə bilə rsiniz, çünki bu, etibarlı başlanğıc kodunu və s. olub olmadığını yoxlamaz.

#### 22.1.4 FASTA faylinin böyük hə rflə rində ardıcılığın yaradılması

Çox vaxt siz ə mə kdaşlardan mə lumatları FASTA şaylları kimi ə ldə edə cə ksınız və bə zə n ardıcılıqlar böyük və kiçik hə rflə rin qarışığında ola bilə r. Bə zi hallarda bu, qə sdə n edilir (mə sə lə n, keyfiyyə tsiz bölgə lə r üçün kiçik hə rf), lakin adə tə n bu vacib deyil. Siz hə r şeyi ardıcıl etmə k üçün şaylı redaktə etmə k istə yə bilə rsiniz (mə sə lə n, bütün böyük hə rflə r) və bunu SeqRecord obyektinin yuxarı () metodundan istifadə edə rə k asanlıqla edə bilə rsiniz (Biopython 1.55-də ə lavə olunub):

**Bio idxaldan SeqIO**

```
qeydlə r = (rec.upper() SeqIO.parse ("mixed.fas", "fasta")) count = SeqIO.write(records, "upper.fas",
"fasta") print(" %i qeydlə ri böyük hə rflə rə çəvrildi" % sayı)
```

Bu necə işlə yir? Birinci sə tir sadə cə Bio.SeqIO modulunu idxlə edir. İkinci sə tir maraqlı bitdir – bu giriş faylindan (mixed.fas) tə hlil edilmiş hə r bir qeydin böyük hə rf versiyasını verə n Python generator ifadə sidir. Üçüncü sə tirdə bu generator ifadə sini Bio.SeqIO.write() funksiyasına veririk və o, yeni böyük hə rflə r qeydlə rini çıxış faylımızda (upper.fas) saxlayır.

Generator ifadə sində n istifadə etmə yimizin sə bə bi (siyahı və ya siyahı başa düşmə kə və zinə ) bu, yaddaşa bir anda yalnız bir qeydin saxlanması demə kdir. Milyonlarla girişə olan böyük fayllarla mə şəkil olsanız, bu, hə qiqə tə n vacib ola bilə r.

**22.1.5 Ardıcılıq faylinin çeşidlə nmə si**

Tutaq ki, siz ardıcılıq faylini uzunluğa görə çeşidlə mə k istə yirsiniz (mə sə lə n, montajdan kontiglə r toplusu) və siz FASTA və ya FASTQ kimi Bio.SeqIO-nun oxuya, yaza (və indekslə yə bildiyi) fayl formatı ilə işlə yirsınız.

Ə gə r fayl kifayə t qə də r kiçikdirlə , siz hamısını SeqRecord obyektlə rinin siyahısı kimi bir anda yaddaşa yüklə yə , siyahını çeşidlə yə və saxlaya bilə rsiniz:

**Bio idxaldan SeqIO**

```
qeydlə r = siyahı(SeqIO.parse("ls_orchid.fasta", "fasta")) records.sort(key=lambda r:
len(r))
SeqIO.write(qeydlə r, "sorted_orchids.fasta", "fasta")
```

Yeganə ağıllı bit qeydlə ri necə çeşidlə mə k üçün müqayisə metodunu tə yin etmə kdir (burada onları uzunluğa görə çeşidlə yirik). Əvvə lə ə n uzun qeydlə ri istə sə niz, müqayisə ni çevirə və ya ə ks arqumentdə n istifadə edə bilə rsiniz:

**Bio idxaldan SeqIO**

```
qeydlə r = siyahı(SeqIO.parse("ls_orchid.fasta", "fasta")) records.sort(key=lambda r:
-len(r))
SeqIO.write(qeydlə r, "sorted_orchids.fasta", "fasta")
```

İndi bu, olduqca sadə dir - amma çox böyük bir fayliniz varsa və hamısını belə yaddaşa yükə yə bilmirsinizsə , nə baş verir? Mə sə lə n, uzunluğa görə çeşidlə mə k üçün bə zi yeni nə sil ardıcılıq oxunuşlarınız ola bilə r. Bunu Bio.SeqIO.index() funksiyasından istifadə etmə klə hə ll etmə k olar.

**Bio idxaldan SeqIO**

```
Uzunluqları və identifikatorları ə ldə edin və
SeqIO.parse("ls_orchid.fasta",
"fasta") proqramında rec üçün uzunluğu len_and_ids = sorted((len(rec), rec.id) üzrə çeşidlə yin.

) ids = tə rsinə çəvrildi([len_and_ids -də (uzunluq, id)]) del len_and_ids # bu
yaddaşı boşaldın rekord_index =
SeqIO.index("ls_orchid.fasta", "fasta") qeydlə r = (idlə rdə ki id üçün rekord_index[id])

SeqIO.write(qeydlə r, "sorted.fasta", "fasta")
```

Əvvəl icə Bio.SeqIO.parse() istifadə edərək faylı bir dəfə skan edirik, qeyd identifikatorlarını və onların uzunluqlarını dəstələr siyahısında qeyd edirik. Sonra onları uzunluq sırasına uyğunlaşdırmaq üçün bu siyahını çeşidləyirik və uzunluqları atırıq. Bio.SeqIO.index() identifikatorlarının bu çeşidlə nəmiş siyahısından istifadə bizi qeydləri bir-birə əldə etməyə imkan verir və biz onları çıxış üçün Bio.SeqIO.write()-ə ötürürük.

Bu nümunədə rəqəm hamisi Bio.SeqIO.write() istifadə edərək çıxarılan SeqRecord obyektlərinə qeydləri təhlil etmək üçün Bio.SeqIO-dan istifadə edir. Düz mətn SwissProt formatı kimi Bio.SeqIO.write() funksiyasının dəstəkləmədiyi fayl formatını çeşidləmək istəyirsinizsə nə etməli? Biopython 1.54-də Bio.SeqIO.index()-əlavə edilmiş get\_raw() metodundan istifadə etməklə alternativ həll variantıdır (bax: Bölüm 5.4.2.2).

#### [Bio idxaldan SeqIO](#)

```
Uzunluqları və identifikatorları əldə edin və
SeqIO.parse("ls_orchid.fasta",
 "fasta") programında rec üçün uzunluğu len_and_ids = sorted((len(rec), rec.id)) üzrə çeşidləyin.

) ids = reversed([len_and_ids -də (uzunluq, id)]) del len_and_ids # bu
yaddaşı boşaldın

rekord_index = SeqIO.index("ls_orchid.fasta", "fasta") açıq ("sorted.fasta", "wb")
ilə out_handle :
 id- də id üçün :
 out_handle.write(record_index.get_raw(id))
```

Qeyd edək ki, Python 3-dən sonra biz faylı binar rejimdə yazmaq üçün açmalıyıq, çünkü get\_raw() metodu bayt obyektlərinini qaytarır.

Bonus olaraq, məlumatları ikinci dəfə SeqRecord obyektlərinə təhlil etmədiyi üçün daha sürətli olmalıdır. Bunu yalnız FASTA formatı ilə istifadə etmək istəyirsinizsə, rekord identifikatorları və uzunluqları əldə etmək üçün aşağıda verilen FASTA analizatorundan istifadə etməklə bunu bir addım daha da sürətli edir:

#### [Bio.SeqIO.FastaIO- dan SimpleFastaParser -dən Bio idxal SeqIO -dan idxal](#)

```
Uzunluqları və idləri əldə edin və tutacaqdə olduğu kimi
open("ls_orchid.fasta") ilə uzunluğa görə çeşidləyin: başlıq
 üçün len_and_ids =
 sorted((len(seq), title.split(None, 1)[0]),
 SimpleFastaParser(in_handle) üçün seq
)
ids = reversed([len_and_ids -də (uzunluq, id)]) del len_and_ids # bu
yaddaşı boşaldın

rekord_index = SeqIO.index("ls_orchid.fasta", "fasta") açıq ("sorted.fasta", "wb")
ilə out_handle :
 id- də id üçün :
 out_handle.write(record_index.get_raw(id))
```

#### 22.1.6 FASTQ faylları üçün sadə keyfiyyətli filtrləmə

FASTQ formatı Sanger-də təqdim edildi və indi nukleotid ardıcılığının oxunuşlarını keyfiyyət balları ilə birlikdə saxlamaq üçün geniş istifadə olunur. FASTQ faylları (və müvafiq QUAL faylları) hər hərf annotasiyasının əla nümunəsidir, çünkü ardıcılıqlıqda hər bir nukleotid üçün əlaqəli keyfiyyət ballı var.

Hər hərf üçün hər hansı annotasiya SeqRecord-da letter\_annotations lüğətiində siyahı, dəftərvə ya sətir kimi saxlanılır (ardıcılığının uzunluğu ilə eyni sayıda elementlə).

Ümumi və zifə lə rdə n biri böyük ardıcılıqla oxunuşlar toplusunu götürmə k və keyfiyyə t ballarına ə sasə n onları süzgə cdə n keçirmə k (və ya kə smə kdir). Aşağıdakı nümunə çox sadə dir, lakin SeqRecord obyektində keyfiyyə tli verilə nlə rlə işlə mə yin ə saslarını tə svir etmə lidir. Burada edə cə yimiz hə r şey FASTQ mə lumat faylini oxumaq və yalnız PHRED keyfiyyə t balları bə zi hə ddə n yuxarı olan qeydlə ri seçmə k üçün onu süzgə cdə n keçirmə kdir (burada 20).

Bu nümunə üçün biz ENA ardıcılığı oxu arxivində n, <ftp://ftp-də n yüklə nmiş bə zi real mə lumatlardan istifadə edə cə yik.sra.ebi.ac.uk/vol1/fastq/SRR020/SRR020192/SRR020192.fastq.gz> (2MB) 19MB-lıq SRR020192.fastq faylinə açılır. Bu, virusa yoluxmuş Kaliforniya də niz şirlə rində n alınan bə zi Roche 454 GS FLX tə k son mə lumatlarıdır (bax: <https://www.ebi.ac.uk/ena/data/view/SRS004476> tə fə rrüatlar üçün).

Əvvə lə oxunanları hesablayaq:

#### Bio idxaldan SeqIO

```
count = 0
SeqIO.parse ("SRR020192.fastq", "fastq"):
 say += 1 çap("%i
oxuyur" % sayı)
```

İndi minimum PHRED keyfiyyə ti 20 üçün sadə filtrlə mə edə k:

#### Bio idxaldan SeqIO

```
yaxşı_oxumaq = (
 rec
 rec üçün SeqIO.parse ("SRR020192.fastq", "fastq") ə gə r
 min(rec.letter_annotations["phred_quality"]) >= 20

) count = SeqIO.write(yaxşı_oxumalar, "yaxşı_keyfiyyə tli.fastq", "fastq") çap ("Saved %
oxuyur" % sayı)
```

Bu, indiki 41892 oxunuşdan yalnız 14580-ni çıxardı. Daha mə qbul bir şey oxunuşları keyfiyyə tli şə kildə kə smə k olardı, lakin bu, yalnız bir nümunə kimi nə zə rdə tutulmuşdur.

FASTQ fayllarında milyonlarla qeyd ola bilə r, ona görə də onların hamısını bir anda yaddaşa yüklə mə k daha yaxşıdır. Bu nümunə generator ifadə sində n istifadə edir, bu o demə kdir ki, hə r hansı yaddaş mə hdudiyyə tində n qaçaraq eyni anda yalnız bir SeqRecord yaradılır.

Qeyd edə k ki, burada aşağı sə viyyə li FastqGeneralIterator analizatorundan istifadə etmə k daha sürə tli olardı (bax. Bölmə 5.6), lakin bu, keyfiyyə t sə tirini tam xallara çevirmir.

### 22.1.7 Astar ardıcılığının kə silmə si

Bu nümunə üçün biz GATGACGGTG-nin bə zi FASTQ formatlı oxu mə lumatlarında axtarmaq istə diyimiz 5' primer ardıcılığı olduğunu iddia edə cə yik. Yuxarıdakı nümunə də olduğu kimi, biz ENA-dan endirilmiş SRR020192.fastq faylından istifadə edə cə yik (<ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR020/SRR020192/SRR020192.fastq.gz>).

Əsas Bio.SeqIO interfeysində n istifadə etmə klə , eyni yanaşma mat üçün də stə klə nə n hə r hansı digə r fayl (mə sə lə n, FASTA faylları) ilə işlə yə cə k. Bununla belə , böyük FASTQ faylları üçün burada aşağı sə viyyə li FastqGeneralIterator analizatoru daha sürə tli olardı (əvvə lki nümunə yə və Bölmə 5.6-a baxın).

Bu kod generator ifadə si ilə Bio.SeqIO-dan istifadə edir (bütün ardıcılıqların yaddaşa yüklə nmə sinin qarşısını almaq üçün). də rhal) və oxunmanın primer ardıcılığı ilə başlayıb-başlamadığını görmə k üçün Seq obyektinin startswith metodu:

#### Bio idxaldan SeqIO

```
primer_reads = (
 rec
 Rec üçün SeqIO.parse ("SRR020192.fastq", "fastq")
```

```

if rec.seq.startswith("GATGACGGTGT")

) count = SeqIO.write(primer_reads, "with_primer.fastq", "fastq") print("Saved %i reads" %
count)

```

Bu, SRR014849.fastq-dan 13819 oxunuş tapmalı və onları primer.fastq ilə yeni FASTQ faylında saxlamalıdır.

İndi düşünək ki, bunun ə və zinəsiz bu oxunuşları ehtiva edən, lakin primer ardıcılılığı silinmiş FASTQ faylı yaratmaq istəyirdiniz? Bu, sadəcə kiçik bir dəyişiklidir, çünkü SeqRecord-u kəsəbilərik (bax: Bölüm 4.7) ilk on bir hərfi (primerimizin uzunluğu) silmək üçün:

[Bio idxaldan SeqIO](#)

```

trimmed_primer_reads = (rec[11:]
 rec for
 SeqIO.parse ("SRR020192.fastq", "fastq") if rec.seq.startswith("GATGACGGTGT")

) count = SeqIO.write(trimmed_primer_reads, "with_primer_trimmed.fastq", "fastq") print("Saved %i reads" % count)

```

Yenə də, SRR020192.fastq-dan 13819 oxunuş çıxarılmalıdır, lakin bu dəfə ilk on simvolu çıxarıb onları trimmed.fastq primeri ilə başqa yeni FASTQ faylında saxlayın.

İndi, tutaq ki, siz bu oxunuşların primerinin silindiyi, lakin bütün digər oxunuşların olduğu kimi saxlanıldığı yeni FASTQ faylı yaratmaq istəyirsiniz? Əgər rəhbərlik etmək istəyirksə, yəqin ki, öz trim funksiyamızı təyin etmək daha aydındır:

[Bio idxaldan SeqIO](#)

```

def trim_primer(qeyd, primer):
 if record.seq.startswith(primer): qaytar
 rekordu[len(primer) :]
 başqa:
 qayıdış rekordu

trimmed_reads =
 (trim_primer(rekord, "GATGACGGTGT")
 SeqIO.parse -də qeyd üçün ("SRR020192.fastq", "fastq")

) count = SeqIO.write(trimmed_reads, "trimmed.fastq", "fastq") print("Saved %i reads"
% count)

```

Bu, daha uzun çəkir, çünkü bu dəfə çıkış faylında bütün 41892 oxunuş var. Yenə də yaddaş probleminin qarşısını almaq üçün generator ifadəsindən istifadə edirik. Alternativ olaraq generator ifadəsi ə və zinə generator funksiyasından istifadə edə bilərsiniz.

[Bio idxaldan SeqIO](#)

```

def trim_primers(qeydlər, primer): """Oxumaların
başlanğıcında mükəmməl primer ardıcılıqlarını silir.

```

Bu generator funksiyasıdır, qeydlə r arqumenti SeqRecord obyektlə rini qaytaran siyahı və ya iterator olmalıdır.

```
len_primer = len(primer) # bunu daha sonra saxlamaq üçün yaddaşa saxlayın
qeydla rdə qeyd etmə k üçün :
```

```
if record.seq.startswith(primer): gə lir
 rekordu[len_primer:]
başqa:
 gə lir rekordu
```

```
original_reads = SeqIO.parse("SRR020192.fastq", "fastq") trimmed_reads =
trim_primers(original_reads, "GATGACGGTGT") count = SeqIO.write(trimmed_reads,
"trimmed.fastq") print ("%saat " oxundu "
```

Yalnız bə zi qeydlə rin olduğu daha mürə kkə b bir şey etmə k istə yirsinzə , bu forma daha çevikdir saxlanılır – növbə ti misalda göstə rildiyi kimi.

## 22.1.8 Adapter ardıcılığının kə silmə si

Bu, ə slində ə vvə iki nümunə nin sadə bir uzantısıdır. Biz GATGACGGTGT-nin bə zi FASTQ formatlı oxu mə lumatlarında adapter ardıcılığı olduğunu iddia edə cə yik, yenə NCBI-də n SRR020192.fastq faylı (<ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR020/SRR020192/SRR020/SRRg.02st>).

Bununla belə , bu də fə ardıcılığı yalnız başlangıçda deyil, oxunuşların istə nilə n yerində axtaracaqıq:

[Bio idxaldan SeqIO](#)

```
def trim_adaptors (qeydlə r, adaptor):
 """Mükə mmə l adapter ardıcılığını kə sir.
```

Bu generator funksiyasıdır, qeydlə r arqumenti SeqRecord obyektlə rini qaytaran siyahı və ya iterator olmalıdır.

```
len_adaptor = len(adaptor) # bunu daha sonra saxlamaq üçün yaddaşa saxlayın
```

```
qeydla rdə qeyd etmə k üçün :
```

```
index = record.seq.find(adaptor) ə gə r indeks == -1:

 # adapter tapılmadı, ona görə də gə lir rekordu
 kə silmə yə cə k
başqa:
 # adapter gə lir rekordunu
 kə sin [indeks + len_adaptor :]
```

```
original_reads = SeqIO.parse("SRR020192.fastq", "fastq") trimmed_reads =
trim_adaptors(original_reads, "GATGACGGTGT") count =
SeqIO.write(kesilmiş_oxumalar, "trimmed.fastq") printed % "idd "("fastq")
```

Bu misalda biz FASTQ daxiletmə faylından istifadə etdiyimiz üçün SeqRecord obyektlə rində keyfiyyə t balları üçün hə r hə rf annotasiyası var. SeqRecord obyektini dilimlə mə klə müvafiq ballar istifadə olunur

kə silmiş qeydlə r, biz də onları FASTQ faylı kimi çıxara bilə rik.

Hə r oxumanın ə vva linda yalnız primer/adaptor axtardığımız ə vva İki nümunənin çıxışı ilə müqayisə də , kə silmiş oxunuşların bə zilə rinin kə sildikdə n sonra olduqca qısa olduğunu görə bilə rsiniz (mə sə lə n, adapter başlanğıcın yaxınlığında deyil, ortada tapılıbsa). Belə liklə , minimum uzunluq tə lə bini də ə lavə edə k:

[Bio idxaldan SeqIO](#)

```
def trim_adaptors(qeydlə r, adaptor, min_len): """Mükə mmə |
 adaptor ardıcılığını kə sir, oxunma uzunluğunu yoxlayır.
```

Bu generator funksiyasıdır, qeydlə r arqumenti SeqRecord obyektlə rini qaytaran şiyahi və ya iterator olmalıdır.

len\_adaptor = len(adaptor) # bunu qeydlə rda qeyd etmə k üçün daha sonra keş edin :

```
len_record = len(record) # ə gə r len (record) < min_len:

 # Davam etmə k üçün çox qıсадır

 index = record.seq.find(adaptor) ə gə r index == -1: #
 adaptor tapılmadı, ona
 görə də mə hsuldarlıq qeydi kə silmə yə cə k elif len_record
 - index - len_adaptor
 >= min_len:
 # kə sdikdə n sonra bu hə lə kifayə t qə də r uzun mə hsul
 rekordu olacaq [index + len_adaptor :]
```

```
original_reads = SeqIO.parse("SRR020192.fastq", "fastq") trimmed_reads =
trim_adaptors(original_reads, "GATGACGGTGT", 100) count = SeqIO.write (kesilmiş_oxumalar ,
"kirpilmiş . saymaq)
```

Format adlarını də yişdirə rə k, bunun ə və zinə FASTA fayllarına tə tbiq edə bilə rsiniz. Bu kod eyni zamanda də qıq uyğunluq ə və zinə qeyri-sə lis uyğunluq etmə k üçün genişlə ndirilə bilə r(bə lkə də ikili uyğunlaşdırmadan istifadə etmə klə və ya oxunma keyfiyyə ti xallarını nə zə rə alaraq), lakin bu, daha yavaş olacaq.

## 22.1.9 FASTQ fayllarının konvertasiyası

Bölmə [5.5.2-](#) də biz iki fayl formatı arasında çevirmə k üçün Bio.SeqIO-dan necə istifadə edə cə yimizi göstə rdik. Burada ikinci nə sil DNT ardıcılığında istifadə olunan FASTQ faylları ilə bağlı bir az daha ə traflı mə lumat verə cə yik. Cock et al. (2009) [\[6\]](#) daha uzun tə svir üçün. FASTQ faylları hə m DNT ardıcılığını (sə tir kimi), hə m də ə laqə li oxu keyfiyyə tlə rini saxlayır.

PHRED balları (ə ksə r FASTQ fayllarında, hə mçinin QUAL fayllarında, ACE fayllarında və SFF fayllarında istifadə olunur) sadə on log çevrilmə sində n istifadə edə rə k verilmiş bazada ardıcılıq xə tası (burada Pe ilə işarə lə nir) ehtimalını tə msil etmə k üçün faktiki standarta çevrilmişdir :

$$\text{QPHRED} = 10 \times \log_{10}(\text{Pe}) \quad (22.1)$$

Bu o demə kdir ki, sə hə oxunuş ( $\text{Pe} = 1$ ) PHRED keyfiyyə ti 0,  $\text{Pe} = 0.00001$  kimi çox yaxşı oxunuş isə 50 PHRED keyfiyyə ti ə ldə edir. Xam ardıcılıq üçün bundan yüksə k mə lumat keyfiyyə tlə rina nadir hallarda rast gə linir.

oxunmuş xə ritə çə kmə və ya montaj kimi emal, tə xminə n 90-a qə də r keyfiyyə tlə r mümkündür (hə qıqə tə n, MAQ alə ti 0-dan 93-a qə də r aralığında PHRED balları üçün imkan verir).

FASTQ formatı tə k düz mə tn faylında oxunan ardıcılıq üçün hə rflə rin və keyfiyyə t xallarının saxlanması üçün faktiki standart olmaq potensialına malikdir. Mə lhə mdə ki yeganə milçə k odur ki, FASTQ formatının uyğun olmayan və ayırd etmə k çə tin olan ə n azi üç versiyası var...

1. Orijinal Sanger FASTQ formatı ASCII ofset 33 ilə kodlanmış PHRED keyfiyyə tlə rində n istifadə edir. NCBI bu formatı öz Qısa Oxuma Arxivində istifadə edir. Biz bunu Bio.SeqIO-da fastq (və ya fastq-sanger) formatı adlandırırıq.

2. Solexa (sonradan Illumina tə rə fində n satın alındı) bir kodla kodlanmış Solexa keyfiyyə tlə rində n istifadə edə rə k öz versiyasını tə qdim etdi. ASCII ofset 64. Biz bunu fastq-solexa formatı adlandırırıq.

3. Illumina boru kə mə ri 1.3-də n sonra PHRED keyfiyyə tlə rinə malik FASTQ faylları istehsal edir (bu, daha uyğundur), lakin ASCII ofset 64 ilə kodlanmışdır. Biz bunu fastq-illumina formatı adlandırırıq.

Solexa keyfiyyə t balları fə rqli log çevrilmə si ilə müə yyə n edilir:

$$Q_{\text{Solexa}} = \frac{Pe}{1 - Pe} \quad (22.2)$$

Nə zə rə alsaq ki, Solexa/Illumina artıq öz boru kə mə rinin 1.3 versiyasında PHRED ballarından istifadə etmə yə keçib, Solexa keyfiyyə t balları tə dricə n istifadə də n çıxacaq. Sə hə tə xminə rini (Pe) bə rabə rlə şdirlə niz, bu iki tə nlük iki qiymə tlə ndirmə sistemi arasında çevrilmə yə imkan verir - və Biopython köhnə Solexa/Illumina faylinı standart Sanger FASTQ faylinə çevirmə k üçün Bio.SeqIO istifadə etdiyiniz zaman çağırılan Bio.SeqIO.QualityIO modulunda bunu etmə k üçün funksiyaları ehtiva edir:

#### Bio idxaldan SeqIO

```
SeqIO.convert("solexa.fastq", "fastq-solexa", "standard.fastq", "fastq")
```

Yeni Illumina 1.3+ FASTQ faylinı çevirmə k istə yirsinzə , də yişə nərə rin hamısı ASCII ofsetidir, çünkü Fə rqli şə kildə kodlansa da, balların hamısı PHRED keyfiyyə tlə ridir:

#### Bio idxaldan SeqIO

```
SeqIO.convert("illumina.fastq", "fastq-illumina", "standard.fastq", "fastq")
```

Nə zə rə alın ki, belə Bio.SeqIO.convert() funksiyasından istifadə Bio.SeqIO.parse() və Bio.SeqIO.write() birlə şmə sində n daha sürə tlidir, çünkü optimallaşdırılmış kod FASTQ variantları arasında konvertasiya üçün (hə mçinin FASTQ-dan FASTA-ya çevrilənən) istifadə olunur.

Keyfiyyə tli oxumaq üçün PHRED və Solexa balları tə xminə n bə rabə rdir, yə ni hə m fasta-solexa, hə m də fastq-illumina formatları ASCII ofsetində n 64 istifadə etdiy üçün fayllar demə k olar ki, eynidir. Bu, Illumina tə rə fində n düşünülmüş dizayn seçimi idi, yə ni köhnə fasta-solexa üslublu faylları gözlə yə n proqramlar, yə qin ki, daha yeni fastq-illumina fayllarından (yaxşı mə lumatda) istifadə etmə klə yaxşı olacaq. Ə lbə ttə ki, hə r iki variant Sanger, NCBI və başqa yerlə rdə (format adı fastq və ya fastq-sanger) istifadə etdiy orijinal FASTQ standartından çox fə rqlidir.

Ə trafli mə lumat üçün daxili yardımə baxın (hə mçinin [onlayn](#)):

>>> Bio.SeqIO- dan QualityIO idxali >>> kömə k  
(QualityIO)

## 22.1.10 FASTA və QUAL fayllarının FASTQ fayllarına çevrilmə si

FASTQ faylları hə m ardıcılığı, hə m də onların keyfiyyə t sə tirlə rini saxlayır. FASTA faylları yalnız ardıcılığı, QUAL faylları isə yalnız keyfiyyə tlə rə malikdir. Beləliklə , tə k bir FASTQ faylı qoşlaşmış FASTA və QUAL fayllarına və ya onlardan çevrilə bilə r.

FASTQ-dan FASTA-ya keçmə k asandır:

**Bio idxaldan SeqIO**

```
SeqIO.convert("example.fastq", "fastq", "example.fasta", "fasta")
```

FASTQ-dan QUAL-a keçmə k də asandır:

**Bio idxaldan SeqIO**

```
SeqIO.convert("example.fastq", "fastq", "example.qual", "qual")
```

Ancaq bunun eksi bir az daha müəkkə bdir. Bir fayldakı qeydlə ri tə krarlamaq üçün Bio.SeqIO.parse() funksiyasından istifadə edə bilərsiniz, lakin bu halda bizdə iki giriş faylı var. Mümkün bir neçə strategiya var, lakin iki faylin hə qıraq tə n qoşlaşdığını fərz etsək, yaddaşda e sə mə rə li yol hə r ikisini bir-birinə çevirmə kdir. Kod bir az müəkkə bdir, ona görə də bunu etmə k üçün Bio.SeqIO.QualityIO modulunda PairedFastaQualIterator adlı funksiya tə qdim edirik. Bu, iki tutacaq alır (FASTA faylı və QUAL faylı) və SeqRecord iteratorunu qaytarır:

## Bio.SeqIO.QualityIO idxali PairedFastaQualIterator- dan

PairedFastaQualIterator -da qeyd etmə k üçün (open("example.fasta"), open("example.qual")):  
çap (qeyd)

Bu funksiya FASTA və QUAL fayllarının uygunluğunu yoxlayacaq (məsələn, qeydlər eyni sıradadır və eyni ardıcılıq uzunluğuna malikdir). Bir cüt FASTA və QUAL fayllarını tək FASTQ faylinə çevirmə k üçün bunu Bio.SeqIO.write() funksiyası ilə birləşdirə bilərsiniz:

**Bio idxalı SeqIO- dan**

**Bio.SeqIO.QualityIO- dan idxalı PairedFastaQualIterator**

```
open("example.fasta") ilə f_handle , open("example.qual") as q_handle: records =
 PairedFastaQualIterator(f_handle, q_handle) count = SeqIO.write(records,
 "temp.fastq", "fastq")
print("Çevrilmiş %i qeydlər" % sayıl)
```

## 22.1.11 FASTQ faylinin indekslə şdirilmə si

FASTQ faylları adətən çox böyükdir və milyonlarla oxunuşla. Məlumatın çoxluğu səbəbindən bütün qeydləri bir anda yaddaşa yükəmək mümkün deyil. Buna görə yuxarıdakı nümunə lər (süzgəcdən keçirmə və kəsmə) eyni anda yalnız bir SeqRecord-a baxaraq faylı üzərinə təkrarlanır.

Bununla belə, bəzən böyük bir döngə və ya iteratordan istifadə edə bilərsiniz - oxumaq üçün təsadüfi girişə ehtiyacınız ola bilər. Burada Bio.SeqIO.index() funksiyası çox faydalı ola bilər, çünkü o, sizə FASTQ faylında hər hansı oxunuşu öz adı ilə əldə etməyi imkan verir (Bölmə 5.4.2-yə baxın).

Yenə də ENA-dan SRR020192.fastq faylından istifadə edəcə yik (<ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR020/SRR020192/SRR020192.fastq.gz>), Baxmayaraq ki, bu, əslində 50.000-dən az oxunan olduqca kiçik bir FASTQ faylıdır:

```
>>> Bio idxaldan SeqIO >>> fq_dict =
SeqIO.index("SRR020192.fastq", "fastq") >>> len(fq_dict)

41892

>>> siyah(fq_dict.keys()[4]
['SRR020192.38240', 'SRR020192.23181', 'SRR020192.40568', 'SRR020192.23186'] >>> fq_dict["SRR020192.23186"] =
```

Seq('GTCCCAGTATTCGGATTGCTGCCAAACAATGAAATTGACACAGTTACAAC...CCG')

Bunu yeddi milyon oxunan FASTQ faylında sınaqdan keçirərkən indeksləşdirmətəxminən bir də qıqəçəkdi, lakin rekord giriş demək olar ki, dərhal idi.

Bio.SeqIO.index\_db() bacı funksiyası sizə indeksi dərhal təkrar istifadə üçün SQLite3 verilənlər bazası faylında saxlamağa imkan verir - dahaətraflı məlumat üçün Bölmə [5.4.2-ə](#) baxın.

Bölmə [22.1.5-](#)da ki nümunə Bio.SeqIO.index() funksiyasından necə istifadə edə biləcə yinizi göstərir. FASTA faylı - bu, FASTQ fayllarında da istifadə edilə bilər.

## 22.1.12 SFF fayllarının konvertasiyası

Əgərsiz 454 (Roche) ardıcılılığı datası ilə işləsəniz, ya qınıki, Standart Flowgram Format (SFF) faylı kimi xam məlumatlara girişəldə edəcəksiniz. Bu, keyfiyyət balları və orijinal axın məlumatı ilə ardıcıl oxunuşları (əsaslar adlanır) ehtiva edir.

Ümumi tapşırıq SFF-dən bir cüt FASTA və QUAL faylına və ya tək FASTQ faylına çevirməkdir.

Bio.SeqIO.convert() funksiyasından istifadə etməklə bu əməliyyatlarəhmətiyyətsizdir (bax: Bölmə [5.5.2](#)):

```
>>> Bio import SeqIO >>>
SeqIO.convert("E3MFGYR02_random_10_reads.sff", "sff", "reads.fasta", "fasta") 10

>>> SeqIO.convert("E3MFGYR02_random_10_reads.sff", "sff", "reads.qual", "qual") 10 >>> SeqIO.convert("E3MFGYR02_random_10_reads.sff",
"sff", "reads.fastq") 1,
```

Yadda saxlayın ki, çevirmə funksiyası qeydlərinin sayını qaytarır, bu misalda cəmi on. Bu, sizəkə silməmiş oxunuşlar verəcək, burada aparıcı və arxada olan keyfiyyətsiz ardıcılıq və ya adapter kiçik hərfə olacaq.

Əgər kəsilmiş oxunuşları istəyirsinizsə (SFF faylında qeydə alınmış kəsmə məlumatından istifadə etməklə) bundan istifadə edin:

```
>>> Bio import SeqIO >>>
SeqIO.convert("E3MFGYR02_random_10_reads.sff", "sff-trim", "trimmed.fasta", "fasta") 10

>>> SeqIO.convert("E3MFGYR02_random_10_reads.sff", "sff-trim", "trimmed.qual", "qual") 10

>>> SeqIO.convert("E3MFGYR02_random_10_reads.sff", "sff-trim", "trimmed.fastq", "fastq") 10
```

Əgərsiz Linux-u işlədirsinizsə, Roche-dan onların "alətdən kənar" alətlərinin (çox vaxt Newbler alətləri adlanır) surətinə istəyəbilərsiniz. Bu, komanda xəttində SFF-dən FASTA və ya QUAL-a çevrilənin alternativ yolunu təklif edir (lakin hazırda FASTQ çıxışı dəstəklənmir), məsələn

```
$ sffinfo -seq -notrim E3MFGYR02_random_10_reads.sff > reads.fasta $ sffinfo -qual -notrim
E3MFGYR02_random_10_reads.sff > reads.qual $ sffinfo -seq -trim E3MFGYR02_ad.trim E3MFGYR02_ad .
sffinfo -qual -trim E3MFGYR02_random_10_reads.sff > trimmed.qual
```

Biopython-un kə smə nöqtə lə rini tə msil etmə k üçün qarşıq hal ardıcılığı sə tirlə rində n istifadə etmə tə rzi Roche alə tlə rinin etdiklə rini qə sdə n tə qlid edir.

Biopython SFF də stə yi haqqında ə trafı mə lumat üçün daxili yardımına müraciə t edin:

```
>>> Bio.SeqIO -dan idxal SffIO >>> kömə k
(SffIO)
```

#### 22.1.13 Açıq oxu çə rçivə lə rinin müə yyə n edilmə si Mümkün

genlə ri müə yyə n etmə k üçün çox sadə lə şdirilmiş ilk addım açıq oxu çə rçivə lə rini (ORF) axtarmaqdır. Bununla biz dayanma kodonları olmayan uzun bölgə lə r üçün altı çə rçivə nin hamısına baxmağı nə zə rdə tuturuq - ORF sadə cə çə rçivə də dayanma kodonları olmayan nukleotidlə r bölgə sidir.

Ə lbə ttə ki, bir gen tapmaq üçün siz hə mçinin başlanğic kodonun, mümkün promotorların yerlə şdirilmə si barə də narahat olmalısınız - və Eukaryotlarda da narahat olmaq üçün intronlar var. Bununla belə , bu yanaşma viruslarda və prokaryotlarda hə lə də faydalıdır.

Biopython ilə buna necə yanaşa bilə cə yinizi gösta rmə k üçün biza axtarış ardıcılığına ehtiyacımız olacaq və nümunə olaraq yenidə n bakterial plazmidde n istifadə edə cə yik - baxmayaraq ki, bu də fə ə vvə lcə də n işarə lə nmiş genlə ri olmayan sadə FASTA faylı ilə başlayacaqıq: [NC\\_005816.fna](#). Bu bakteriya ardıcılığıdır, ona görə də biz NCBI kodon cə dvə li 11-də n istifadə etmə k istə rdik (tə rcümə haqqında Bölmə 3.8-ə baxın).

```
>>> Bio import SeqIO- dan >>> rekord =
SeqIO.read("NC_005816.fna", "fasta") >>> cə dvə l = 11
```

```
>>> min_pro_len = 100
```

Bütün mümkün ORF tə rcümə lə rinin siyahısını ə ldə etmə k üçün Seq obyektinin bölünmə metodundan istifadə edə n sə liqə li hiylə dir. altı oxu çə rçivə sində :

```
>>> ip üçün , [(+1, rekord.seq), (-1, rekord.seq.reverse_complement())]- də nuc:
... diapazondakı çə rçivə üçün (3):
... uzunluq = 3 * ((len(rekord) - kadr) // 3) # Nucda pro üçün üçdə n çox [frame : frame
... + length].translate(table).split("\n"): ə gə r len(pro) >= min_pro_len: print("%s...%s - uzunluq
... %i, strand %i, çə rçivə %i" %
...
... (pro[:-30], çə rçivə %i" (pro[:-30]) , pro[30]
...
...)
...
GCLMKSSIVATIITILSGSANAASSQLIP...YRF - uzunluq 315, strand 1, çə rçivə 0
KSGELRQTTPASSTLHLRLILQRSGVMMEL...NPE - uzunluq 285, strand 1, frame 1
GLNCSSFSICNWKFIDYINRLFQIYLYCKN, uzunluq167, çə rçivə 11...
VKKILYIKALFLCTVIKLRRFIFSVNNMFK...DLP - uzunluq 165, strand 1, çə rçivə 1
NQIQGVICSPDSGEFMVTFETVMEIKILHK...GVA - uzunluq 355, strand 1, frame 2
RRKEHVSKKRPQKRPRRRFFFHRLnd, strand1,strand1, frame1
TGKQNQCQMSAIWQLRQNTATKTRQRARI...AIK - uzunluq 100, strand 1, çə rçivə 2
QGSGYAFPHASILSGIAMSHFYFLVLHAVK...CSD - uzunluq 114, strand -1, frame 0
IYSTSEHTGEQVMRTLDEVIASRVSPESQnd -uzunluq...1 WGKLQVIGLSMWMVLFSQRFDWLNEQEDA...ESK
- uzunluq 125, strand -1, çə rçivə 1 RGIFMSDTMVVNGSGGVP AFLFGSTLSSY...LLK - uzunluq
361, strand -1, çə rçivə 1 WDVKTGTGVLHCPVATFQ1 uzunluq... tel -1, çə rçivə 1
LSHTVTDFTDQMAQVGLCQCVNVFLDEVTG...KAA - uzunluq 107, tel -1, çə rçivə 2
RALTGLSAPGIRSQTCDRLRELRYVPVSL...PLQ - uzunluq 119, tel -1, çə rçivə 2
```

Qeyd edə k ki, burada hə r bir ipin 5' ucundan (başlanğıcından) çə rçivə lə ri sayıraq. Həmişə irə li ipin 5' ucundan (başlanğıcından) saymaq bəzən daha asandır.

Namizəd zülalların siyahısını yaratmaq və ya çevirmək üçün yuxarıdakı loopə saslı kodu asanlıqla redaktə edə bilərsiniz bu siyahı anlamaq üçün. İndi bu kodun etmədiyi bir şey zülalların harada olduğunu izləməkdədir.

Bunu bir neçə yolla həll edə bilərsiniz. Məsələn, aşağıdakı kod zülalların hesablanması baxımından yerləri izləyir və üçəvuraraq, sonra çərçivə vəzəncir üçün tənzimləməklə ana ardıcılığı geri döndür:

#### [Bio idxaldan SeqIO](#)

```
rekord = SeqIO.read("NC_005816.gb", "genbank") cədvəl = 11
```

```
min_pro_len = 100
```

```
def find_orfs_with_trans(seq, trans_table, min_protein_length):
 cavab = []
 seq_len = len(seq)
 strand = seq[0]
 nuc_in = [(+1, seq), (-1, seq.reverse_complement())]
 for frame in nuc_in:
 if frame[0] == 1:
 seq = frame[1]
 else:
 seq = frame[1].reverse_complement()
 for start in range(0, seq_len - min_protein_length + 1, 3):
 aa_start = start
 aa_end = start + min_protein_length
 trans = seq[start:aa_end].translate(trans_table)
 trans_len = len(trans)
 if trans[-1] == '*':
 aa_end = start + trans_len
 if aa_end - aa_start >= min_protein_length:
 if aa_end % 3 == 1:
 start = frame[0] * start + frame[1].find('A') % 3
 end = min(seq_len, frame[0] * aa_end + 3)
 if start != end:
 başlanğıc = seq_len - (aa_end - start) % 3 - 3
 son = seq_len - (aa_end - start) % 3
 cavab.append((başlanğıc, son, strand, trans[start:aa_end]))
 aa_start = aa_end + 1
 cavab.sort()
 return cavab
```

```
orf_list = find_orfs_with_trans(record.seq, table, min_pro_len)
for start, end, strand, pro in orf_list:
 print("%s...%s - uzunluq %i, strand %i, %i:%i" % (pro[:30], pro[-3:], len(pro), strand, start, end))
```

```
)
```

Və çıxış:

```
NQIQGVICSPDSGEFMVTFETVMEIKILHK...GVA - uzunluq 355, strand 1, 41:1109 WDVKTVTGVLHHPFHLTFSLCPEGATQSGR...VKR
- uzunluq 111, strand -1, 491:827
```

KSGELRQTPPASSTLHLRLILQRSGVMMEL...NPE - uzunluq 285, strand 1, 1030:1888  
 RALTGLSAPGIRSQTSCDRLRELRYVPVSL...PLQ - uzunluq 119, strand -1, 2830:3190  
 RRKERFFHRLPRPRRPPRQSGVMMEL... 128, strand 1, 3470:3857 GLNCSSFCNWKFIDYINRLFQIIYLCKN...YYH  
 - uzunluq 176, strand 1, 4249:4780 RGIFMSDTMVNGSGVPAFLFSGSTLSSY...LLK - uzunluq:9401  
 VKKILYIKALFLCTVIKLRRFISVNNMKF...DLP - uzunluq 165, strand 1, 5923:6421  
 LSHTVTDFTDQMAQVGLCQCVCNFLDEVTG...KAA - uzunluq 107, strand -1, 5974:6298 GCLIFASSSQSSK...  
 315, strand 1, 6654:7602 IYSTSEHTGEQVMRTLDEVIASRSPESQTR...FHV - uzunluq 111, strand -1, 7788:8124  
 WGKLQVIGLSMWMVLFSQRFDDWLNEQEDA...ESK - uzunluq 84:12, strand18 -  
 TGKQNQCQMSAIWQLRQNTAKTRQNRARI...AIK - uzunluq 100, strand 1, 8741:9044  
 QGSGYAFPHASILSGIAMSIFYFLVLHAVK...CSD - uzunluq 114, strand -1, 9264:9609

Ə gə r siz çeşidlə mə ifadə sinə şə rh etsə niz, zülal ardıcılılığı ə vvə lki qaydada göstə rilə cə k, belə liklə , bunun eyni şeyi etdiyini yoxlaya bilə rsiniz. Burada GenBank faylındakı faktiki annotasiya ilə müqayisə etmə yi asanlaşdırmaq üçün onları yerə görə çeşidlə mişik (Bölmə [19.1.9-da göstə rildiyi kimi](#)).

Bununla belə , tapmaq istə diyiniz tə k şey açıq oxu cə rçivə lə rının yerlə ridirsə , o zaman bütün mümkün kodonları tə rcümə etmə k, o cümlə də nə ks ipi axtarmaq üçün tə rs tamamlama etmə k vaxt itkisidir.

Etmə li olduğunuz şey mümkün dayanma kodonlarını (və onların ə ks tamamlayıcılarını) axtarmaqdır. Normal ifadə lə rdə n istifadə burada açıq bir yanaşmadır (bax Python modulu re). Bunlar bir çox proqramlaşdırma dillə rində və hə mçinin grep kimi komanda xə tti alə tlə də stə klə nə n axtarış sə tirlə rini tə svir etmə k üçün son də rə cə güclü (lakin olduqca mürə kkə b) üsuldur). Bu mövzuda bütün kitabları tapa bilə rsiniz!

## 22.2 Ardıcılığın tə hlili və sadə süjetlə r

Bu bölmə 5-ci Fə sildə tə svir edilə n Bio.SeqIO modulundan , üstə gə l Python kitabxanasının matplotlib-in pylab plan tə rtib interfeysində n istifadə edə rə k ardıcılıqla tə hlilin daha bir neçə nümunə sinə göstə rir ( də rslik [Üçün matplotlib veb saytına baxın](#) ). Nə zə rə alın ki, bu nümunə lə ri izlə mə k üçün sizə matplotlib quraşdırılmalı olacaq - lakin onsuz siz hə lə də verila nlə rin tə hlili bitlə rini sınaya bilə rsiniz.

### 22.2.1 Ardıcılığın uzunluqlarının histoqramı

Verilə nlə r toplusunda ardıcılıq uzunluqlarının paylaşmasını vizuallaşdırmaq istə yə bilə cə yiniz çox vaxt var - mə sə lə n, genom montaj layihə sində kontig ölçülə rinin diapazonu. Bu nümunə də biz orxideya FASTA faylımızdan [ls orchid.fasta](#)- dan yenidə n istifadə edə cə yik yalnız 94 ardıcılığı var.

İlk növbə də , biz FASTA faylini tə hlil etmə k və bütün ardıcılıq uzunluqlarının siyahısını tə rtib etmə k üçün Bio.SeqIO-dan istifadə edə cə yik. Bunu for döngə si ilə edə bilə rsiniz, lakin siyahının başa düşülmə sinə daha xoş görürə m:

```
>>> Bio import SeqIO -dan >>> sizes
= [len(rec) in Rec in SeqIO.parse("ls_orchid.fasta", "fasta")] >>> len(ölçülə r), min(ölçülə r), maks(ölçülə r)
(94, 572, 789)
```

```
>>> ölçülə r
[740, 753, 748, 744, 733, 718, 730, 704, 740, 709, 700, 726, ..., 592]
```

İndi bütün genlə rin uzunluqlarına (tam ə də dlə rin siyahısı kimi) sahib olduğumuz üçün matplotlib histoqramından istifadə edə bilə rik. göstə rmə k üçün funksiya.

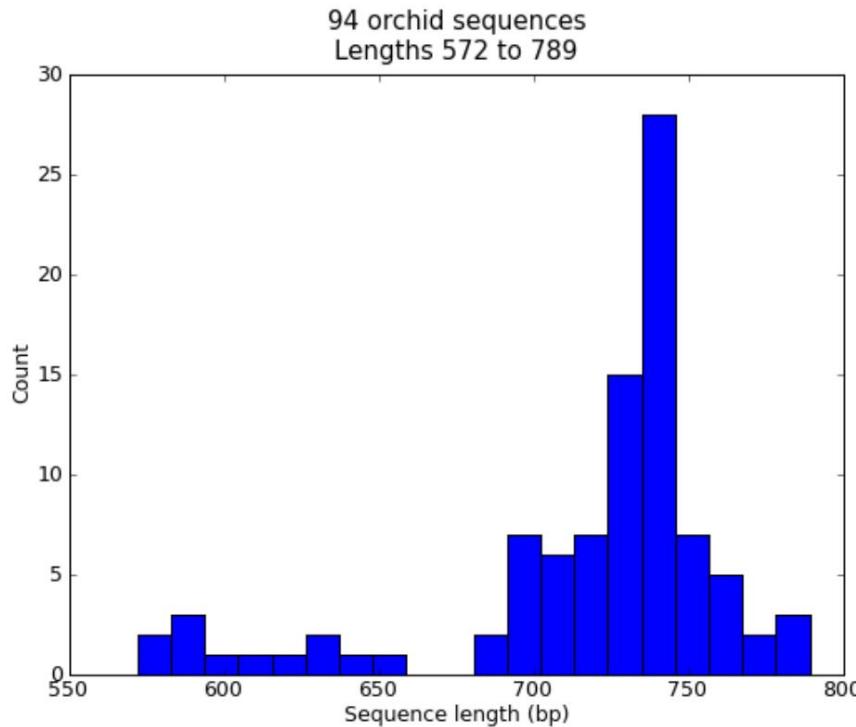
[Bio idxaldan SeqIO](#)

ölçülə r = [ SeqIO.parse ("ls\_orchid.fasta", "fasta")- da qeyd [Üçün len\(rec\) \]](#)

```
idxal pylab
```

```
pylab.hist(ölçülə r, qutular=20)
pylab.title("%i
 orkide sequences\nUzunluqlar %i - %i" % (len(ölçülə r), min(ölçülə r), maks(ölçülə r))

) pylab.xlabel("Ardıcılıq uzunluğu (bp)")
pylab.ylabel("Saymaq")
pylab.show()
```



Şəkil 22.1: Orxideya ardıcılığının uzunluqlarının histogramı.

Şəkil 22.1-də göstərilən qrafiki ehtiva edən yeni pəncərə açılmalıdır. Diqqət yetirin ki, bu orkide ardıcılıqlarının ekspresiyəti təxminən 740 bp uzunluğundadır və burada daha qısa ardıcılıqların altında stili iki fərqli ardıcılıq sinifi ola bilər.

İpucu: Sujeti pəncərə də göstərmək üçün pylab.show() istifadə etmək əvəzinə, siz pylab.savefig(...) istifadə edə bilərsiniz. Rəqəmi faylda saxlamaq üçün (məsələn, PNG və ya PDF kimi).

## 22.2.2 Ardıcılığın qrafiki GC% Nukleotid ardıcılığının

digər asanlıqla hesablanan kəmiyyəti GC%-dir. Məsələn, bir bakterial genomdakı bütün genlərin GC%-nə baxmaq və üfüqi gen transferi ilə bu yaxınlarda edilmiş hər hansı kənar göstəriciləri araşdırmaq istəyə bilərsiniz. Yenə də, bu nümunə üçün biz orxideya FASTA faylımızdan [ls orchid.fasta-dan yenidən istifadə edəcəyik](#).

- İlk növbədə, biz FASTA faylıını təhlil etmək və bütün GC faizlərinin siyahısını tərtib etmək üçün Bio.SeqIO-dan istifadə edəcəyik. Yenə də bunu for loopu ilə edə bilərsiniz, amma mən buna üstünlük verirəm:

```
Bio import SeqIO from
Bio.SeqUtils import gc_fraction

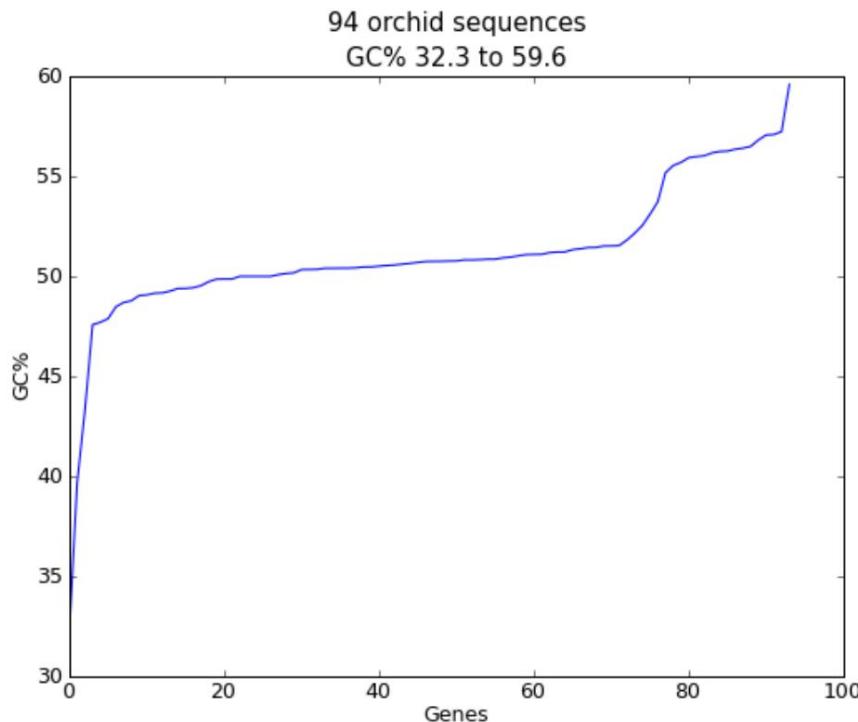
gc_values = sıralanmış (100
 * gc_fraction(rec.seq) SeqIO.parse -də qeyd üçün ("ls_orchid.fasta", "fasta")
)
```

Hər ardıcılıqla oxuyub GC%-ni hesablaşdırıqdan sonra biz onları artan qaydada sıraladıq. İndi biz üzən nöqtə də yərlərinin bu siyahısını götürəcəyik və onları matplotlib ilə tərtib edəcəyik:

```
idxal pylab

pylab.plot(gc_values) pylab.title(
 "%i orkide ardıcılığı\nGC% %0.1f - %0.1f" % (len(gc_values),
 min(gc_values), max(gc_values))

) pylab.xlabel("Genlər")
pylab.ylabel("GC%")
pylab.show()
```

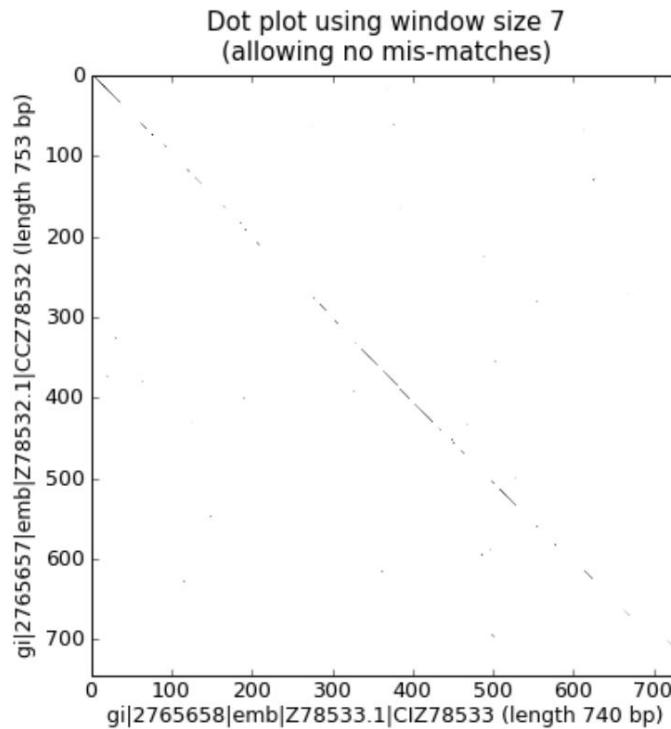


Şəkil 22.2: Orkide ardıcılığının uzunluqlarının histogramı.

Əvvəlki nümunədə olduğu kimi, Şəkil 22.2-də göstərilən qrafiklə yeni bir pəncərə açılmalıdır. Bunu bir orqanizmin genlərinin tam dəstində sırasanız, yəqin ki, bundan daha hamar bir süjetə ləğərdəniz.

## 22.2.3 Nukleotid nöqtə lə rının qrafiklə ri

Nöqtə xə tti iki nukleotid ardıcılığını bir-birinə oxşarlıq üçün vizual olaraq müqayisə etmə k üsuludur. Tez-tez uyğunsuzluq hə ddinə malik qısa alt ardıcılıqları bir-biri ilə müqayisə etmə k üçün sürüşmə pə ncə rə si istifadə olunur. Burada sadəlik üçün biz yalnız mükə mmə l uyğunluqlar axtaracaqıq (Şəkil 22.3-də qara rə ngdə göstə rilmışdır).



Şəkil 22.3: İki sə hlə b ardıcılığının uzunluğunun nukleotid nöqtə si qrafiki (pylab-in imshow funksiyasından istifadə etmə klə ).

Başlamaq üçün bize iki ardıcılıq lazımdır. Mübahisə xatırınə , sadə cə olaraq ilk ikisini özümüzən götürə cə yik orchid FASTA faylı [ls\\_orchid.fasta](#):

**Bio idxaldan SeqIO**

```
in_handle kimi open("ls_orchid.fasta") ilə : record_iterator =
 SeqIO.parse(in_handle, "fasta") rec_one = next(record_iterator) rec_two
 = next(record_iterator)
```

Biz iki yanaşma göstə rə cə yik. Birincisi, oxşarlıq matrisini tə rtib etmə k üçün bütün pə ncə rə ölçülü alt ardıcılıqları bir-biri ilə müqayisə edə n sadə sadə lövh bir tə tbiq. Siz bir matris və ya massiv obyekti qura bilə rsiniz, lakin burada biz sadə cə olaraq iç içə siyahı başa düşülmə si ilə yaradılmış booleanların siyahılarının siyahısını istifadə edirik:

```
pə ncə rə = 7
seq_one = rec_one.seq.upper() seq_two =
rec_two.seq.upper() data = [
 (seq Bir[i : i + pə ncə rə] != seq ikinci[j : j + pə ncə rə])
```

```
j üçün diapazonda (len(seq_one) - pə ncə rə)
```

```
] diapazondakı i üçün (len(seq_two) - pə ncə rə)
```

```
]
```

Qeyd edə k ki, biz burada ə ks tamamlayıcı uyğunluqları yoxlamamışq. İndi biz bu mə lumatları göstə rmə k üçün matplotlib-in pylab.imshow() funksiyasından istifadə edə cə yik, ə vvə lcə boz rə ng sxemini tə lə b edə cə yik ki, bu qara və aq rə ngdə olsun:

### İdxal pylab

```
pylab.gray()
pylab.imshow(data)
pylab.xlabel("%s (uzunluq %i bp)" % (rec_one.id, len(rec_one))) pylab.ylabel("%s (uzunluq %i bp)" %
(rec_two.id, len(rec_two) (" %i bp) istifadə edə rə k pylab. yanlış uyğunluqlar)" % pə ncə rə)
pylab.show()
```

Şəkil 22.3-də ki qrafiki göstə rə n yeni pə ncə rə açılmalıdır. Gözlə diyiniz kimi, bu iki ardıcılıqlı diaqonal boyunca pə ncə rə ölçülü uyğunluqların qismə n xə tti ilə çox oxşardır. İnversiyaların və ya digər maraqlı hadisələrin göstəricisi olan diaqonaldan kənar uyğunluqlar yoxdur.

Yuxarıdakı kod kiçik nümunələrdə yaxşı işləyir, lakin bunu daha böyük ardıcılılığı tətbiq etməkdə iki problem var ki, biz bunları aşağıda nəzərdən keçirəcə yik. Hər şeydən nə və l, bütün müqayisələrə qarşı bu kobud güc anlaşması çox yavaşdır. Ə və zində, pə ncə rə ölçülü alt ardıcılıqları yerləri ilə əlaqələndirənlərini tərtib edəcə yik və sonra hər iki ardıcılıqlıdan olan alt ardıcılıqları tapmaq üçün müəyyən edilmiş kəsişməni götürəcə yik. Bu, daha çox yaddaş istifadə edir, lakin daha sürətlidir. İkincisi, pylab.imshow() funksiyası göstərə biləcəyi matrisin ölçüsü ilə məhdudlaşır.

Alternativ olaraq, biz pylab.scatter() funksiyasından istifadə edəcə yik.

Pəncərə ölçülü alt ardıcılıqları yerlərin uyğunlaşdırılan lüğətlər yaratmaqla başlayırıq:

```
pə ncə rə = 7
dict_one = {} dict_two
= {} ardıcılıq üçün,
bölümə _dict [(rec_one.seq.upper(),
dict_one), (rec_two.seq.upper(), dict_two),
```

```
]:
```

```
diapazonda olan i üçün (len(seq) - pə ncə rə): bölümə
= seq[i : i + pə ncə rə] cəhd edin:
```

```
bölümə _dict[bölümə].append(i) istisna olmaqla, KeyError:
section_dict[bölümə] = [i]
```

```
İndi hər iki ardıcılıqlıda təqdim edilən hər hansı alt ardıcılığı təqdim etmək üçün uyğunluq =
set(dict_one).intersection(dict_two) print("%i unikal uyğunluq" % len(matlar))
```

pylab.scatter() funksiyasından istifadə etmək üçün bizə x və y koordinatları üçün ayrıca siyahılar lazımdır:

```
Şəhər şəhər adı xəttinin x və y koordinatlarının siyahılarını yaradın x = [] y = []
```

matçlardakı bölümə üçün:

```
i üçün dict_one [bölümə]:
```

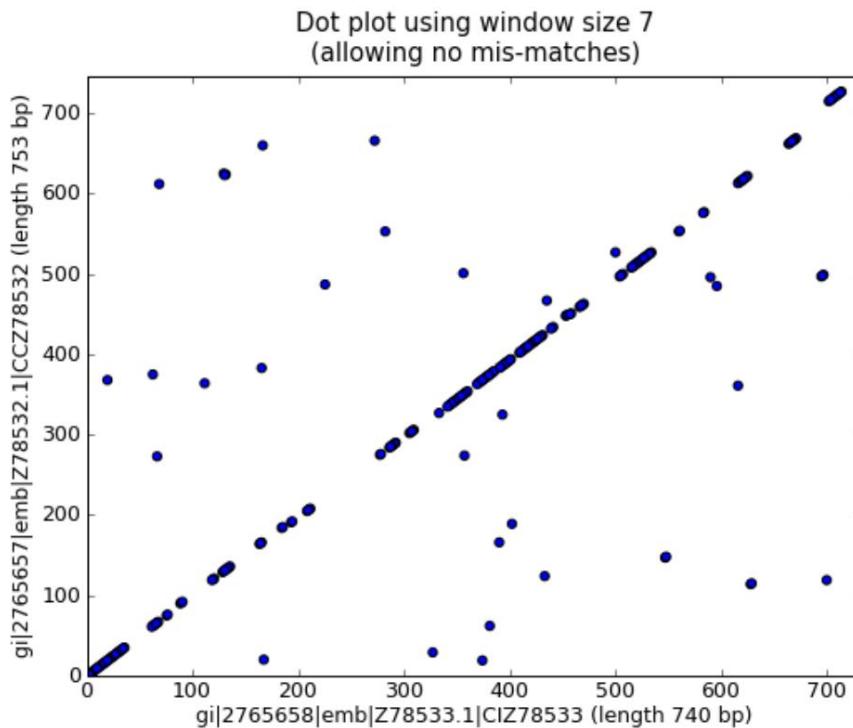
```
j üçün dict_ik [bölmə]: x.append(i)
y.append(j)
```

İndi yenidə n işlə nmış nöqtə süjetini sə pə lə nmə xə tti kimi çə kmə yə hazırlıq:

idxal pylab

```
pylab.cla() # hə r hansı ə vvə iki qrafiki tə mizlə yin
pylab.gray()
pylab.scatter(x, y) pylab.xlim(0,
len(rec_one) - pə ncə rə) pylab.ylim(0, len(rec_two) -
pə ncə rə) pylab.xlabel("%s (uzunluq %i bp)" %
(recylab.gray () pylabscatter .)) (uzunluq %i bp)" % (rec_two.id, len(rec_two))) pylab.title("Pə ncə rə
ölçüsündə n istifadə edə rə k nöqtə planı %i\n(sə hv uyğunluqlara imkan vermir)" % pə ncə rə)
pylab.show()
```

Şəkil 22.4-də ki qrafiki göstərə n yeni pə ncə rə açılmalıdır. Şəxə n mə n bu ikinci süjeti tapıram



Şəkil 22.4: İki orxideya ardıcılığının uzunluğunun nukleotid nöqtə sinin qrafiki (pylab-in sə pilmə funksiyasından istifadə etmə klə ).

oxumaq daha asan! Yenə də qeyd edə k ki, biz burada ə ks tamamlayıcı uyğunluqları yoxlamamışiq – bunu etmə k üçün bu nümunə ni genişlə ndirə və ola bilsin ki, irə li uyğunluqları bir rə ngdə və ə ks uyğunluqları başqa rə ngdə tə rtib edə bilə rsiniz.

#### 22.2.4 Oxunma mə lumatlarının ardıcılışdırılmasının keyfiyyə t ballarının planlaşdırılması

İkinci nə sil ardıcılıq mə lumatları ilə işlə yirsinzə , keyfiyyə tli mə lumatların planını qurmağa cə hd edə bilə rsiniz. Cütlə nmiş son oxunuşları ehtiva edə n iki FASTQ faylından istifadə edə n nümunə , irə li oxunmalar üçün SRR001666 1.fastq və ə ks oxunuşlar üçün SRR001666 2.fastq. Bunlar ENA ardıcılıqla oxunan arxiv FTP saytından endirilib ([ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR001/SRR001666\\_1.fastq.gz](ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR001/SRR001666/SRR001666_1.fastq.gz) və [ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR001/SRR001666\\_2.fastq.gz](ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR001/SRR001666/SRR001666_2.fastq.gz)), və E. coli-də ndir – bax <https://www.ebi.ac.uk/ena/data/view/SRR001666> ə traflı mə lumat üçün.

Aşağıdakı kodda pylab.subplot(...) funksiyası irə li və tə rsi göstə rmə k üçün istifadə olunur yan-yanı iki alt süjetdə keyfiyyə tlə r. Yalnız ilk ə lli oxunuşu tə rtib etmə k üçün bir az kod da var.

```
Bio import SeqIO
-dan pylab idxalı
```

[1, 2] -də alt rə qə m üçün :

```
fayl adı = "SRR001666_%i.fastq" % subfigure pylab.subplot(1, 2, subfigure) for
i, enumerate (SeqIO.parse(filename, "fastq")) ilə
qeyd edin): a gə r i >= 50:

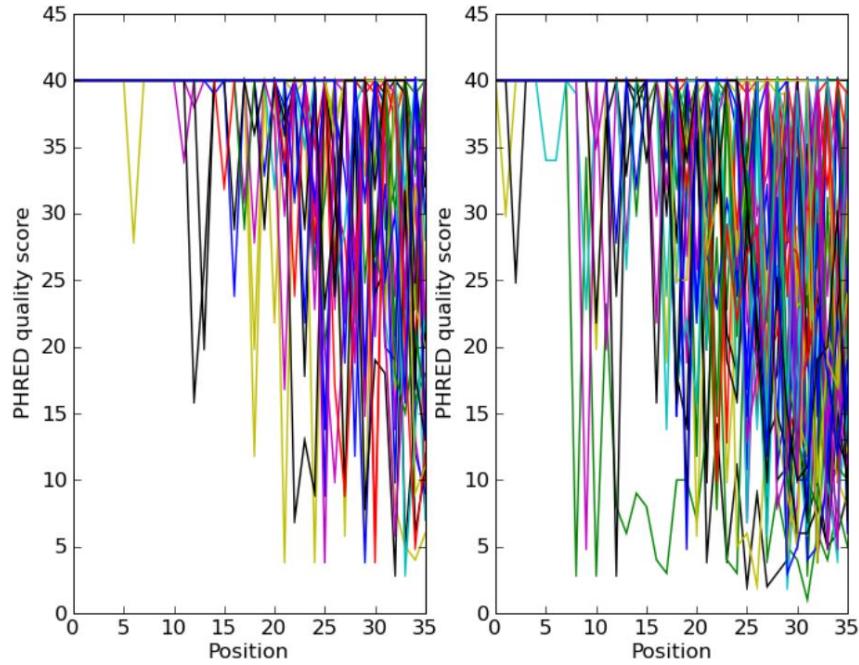
 qırın # hıylə !
 pylab.plot(record.letter_annotations["phred_quality"]) pylab.ylim(0, 45)
 pylab.ylabel("PHRED")
 keyfiyyə t balı") pylab.xlabel("Mövqe")
 pylab.savefig("SRR001666.png")
print("Bitti")
```

Qeyd edə k ki, biz burada Bio.SeqIO format adından fastq istifadə edirik, çünkü NCBI bu oxunuşları PHRED xalları ilə standart Sanger FASTQ formatından istifadə edə rə k saxlamışdır. Bununla belə , oxunan uzunluqlardan tə xmin edə bildiyniz kimi, bu mə lamat Illumina Genom Analizatorundan idi və çox güman ki, ə vvə lcə iki Solexa/Illumina FASTQ variant fayl formatından birində idi.

Bu nümunə pylab.show(...) ə və zinə pylab.savefig(...) funksiyasından istifadə edir, lakin ə vvə l qeyd edildiyi kimi hə r ikisi faydalıdır. Nə ticə Şəkil 22.5-də göstə rilmişdir .

#### 22.3 BioSQL – ə laqə li verilə nlə r bazasında ardıcılığın saxlanması

**BioSQL OBF** arasında birgə sə ydir layihə lə r (BioPerl, BioJava və s.) ardıcılıq mə lumatlarını saxlamaq üçün paylaşılan verilə nlə r bazası sxemini də stə klə mə k üçün. Teorik olaraq, siz BioPerl ilə GenBank faylinı verilə nlə r bazasına yüklə yə , sonra Biopython istifadə edə rə k bunu verilə nlə r bazasından xüsusiyyə tlə ri olan rekord obyekti kimi çıxara bilə rsiniz - və Bio.SeqIO istifadə edə rə k GenBank faylinı birbaşa SeqRecord kimi yükəlmiş kimi az-çox eyni şəyi ə ldə edə bilə rsiniz (Fə sil 5). Biopython-un BioSQL modulu hazırda <http://biopython.org/wiki/BioSQL> ünvanında sə nə dlə şdirilir. olan wiki sə hifə lə rimizin bir hissə sidir.



Şəkil 22.5: Bəzi qoşalaşmış son oxunuşlar üçün keyfiyyətli qrafik.

## 23-cü fə sil

### Biopython test çə rçivə si

Biopython, unittest-ə ə saslanan reqləsiyyə test çə rçivə sinə (`run_tests.py` faylı) malikdir, Python üçün standart vahid test çə rçivə si. Modullar üçün hə rtə rə fli testlə rin tə min edilmə si Biopython kodunun çıxmazdan ə vvə l mümkün qə də r sə hvsiz olmasına ə min olmağın ə n vacib aspektlə rində n biridir. Bu, hə m də töhfə vermə nin ə n də yə rsiz aspektlə rində n biri olmağa meyllidir. Bu fə sil Biopython testlə rini yerinə yetirmə yi və yaxşı test kodunu yazmağı mümkün qə də r asanlaşdırmaq üçün nə zə rdə tutulmuşdur. İdeal olaraq, Biopython-a daxil olan hə r bir modulun testi olmalıdır (və hə mçinin sə nə dlə ri olmalıdır!). Bütün tə rtibatçılarımız və Biopython-u mə nbə də n quraşdırın hə r kə s vahid testlə ri keçirmə yə tə şviq olunur.

#### 23.1 Testlə rin icrası

Biopython mə nbə kodunu yükla diyiniz zaman və ya mə nbə kodu anbarımızdan yoxladığınız zaman Testlə r alt kataloqunu tapmalısınız. Buraya `run_tests.py` açar skripti, `test_XXX.py` adlı çoxlu fə rdi skriptlə r və test paketi üçün daxiletmə faylları olan bir çox digə r alt kataloqlar daxildir.

Biopython-un qurulması və quraşdırılmasının bir hissə si olaraq siz adə tə n ə mrdə tam test paketini işlə də cə ksiniz Biopython mə nbə sinin yuxarı sə viyə li kataloqundan aşağıdakıları istifadə edə rə k sə tir:

```
$ python setup.py testi
```

Bu, ə slində Testlə r alt kataloquna getmə yə və işlə mə yə bə rabə rdir:

```
$ python run_tests.py
```

Siz tez-tez testlə rdə n bə zilə rini yerinə yetirmə k istə yə cə ksiniz və bu belə edilir:

```
$ python run_tests.py test_SeqIO.py test_AlignIO.py
```

Testlə rin siyahısını verə rkə n, .py uzantısı isteğe bağlıdır, ona görə də sadə cə olaraq yaza bilə rsiniz:

```
$ python run_tests.py test_SeqIO test_AlignIO
```

Docstring testlə rini işə salmaq üçün (23.3-cü bölmə yə baxın) istifadə edə bilə rsiniz

```
$ python run_tests.py doctest
```

Siz hə mçinin --offline, mə sə lə n, ə lava etmə klə açıq onlayn komponentlə quraşdırılmış hə r hansı testlə ri ötürə bilə rsiniz.

```
$ python run_tests.py --offline
```

Varsayılan olaraq, run\_tests.py docstring testlə ri də daxil olmaqla bütün testlə ri hə yata keçirir.

Fə rdi test uğursuz olarsa, siz onu birbaşa işə salmağa cəhd edə bilərsiniz ki, bu da sizə daha çox məlumat verə bilər.

Python-un standart vahid test çərçivəsinə əsaslanan testlər vahid testi idxal edəcək və sonra vahidtest.TestCase müəyyən edəcək kodun bəzi xüsusi aspektlərinin yoxlayan test\_ilə başlayan metodlar kimi hər biri bir və ya bir neçə alt testi olan siniflər.

### 23.1.1 Tox istifadə edərək testlərin aparılması

Əksər Python layihələri kimi sizdə [Tox-dan](#) istifadə edə bilərsiniz sisteminizdə artıq quraşdırılmış olması şərti ilə birdən çox Python versiyasında testləri hə yata keçirmək üçün.

İstifadəçi xas parametrləri (məsələn, Python versiyalarının icra edilə bilən adları) bağlamaqdakətinlikdən kəndliyimiz üçün biz kod bazamızda konfiqurasiya tox.ini faylini təmin etmirik. Siz həmçinin Biopython-u yalnız da stəklə diyimiz Python versiyalarının bir hissəsinə qarşı sınıqdan keçirməklə maraqlana bilərsiniz.

Əgər siz Tox-dan istifadə etməkdə maraqlısınızsa, aşağıda göstərilən tox.ini nümunəsi ilə başlaya bilərsiniz:

```
[tox]
envlist = pypy,py38,py39

[testenv]
changedir = Testəmrləri
= {envpython} run_tests.py --offline deps =

numpy
reportlab
```

Yuxarıdakı şablondan istifadə edərək, tox tətbiqi Biopython kodunuza PyPy, Python 3.8 və 3.9. Güman edilir ki, həmin Pythonların icra olunan faylları Python 3.8 üçün "python3.8" adlandırılıb və s.

## 23.2 Yazı testləri

Tutaq ki, siz Biospam adlı modul üçün bəzi testlər yazmaq istəyirsiniz. Bu, yazdığınız modul və ya hər hansı bir testi olmayan mövcud modul ola bilər. Aşağıdakı nümunələrdə biz güman edirik ki, Biospam sadə riyaziyyatı hə yata keçirən moduldur.

Hər bir Biopython testi testin özünü ehtiva edən skriptdən və istege bağlı olaraq girişi olan qovluqdan ibarətdir. testərək findən istifadə olunan fayllar:

1. test\_Biospam.py – Modulunuz üçün faktiki test kodu.

2. Biospam [istege bağlı] – İstənilən rəsüruri daxiletmə fayllarının yerləşəcəyi kataloq. Əlilənəzərdən keçirilməli olan hər hansı çıxış fayliniz varsa, əsas Testlər kataloqunun tixanmasını almaq üçün onları buradan çıxarın (lakin bu tövsiyə edilmir). Ümumiyyətlə, müvəqqəti fayl/qovluqdan istifadə edin.

Testlər kataloqunda test\_ prefiksi olan hər hansı skript run\_tests.py tərəfindən tapılacaq və işlədiləcək. Aşağıda test\_Biospam.py test skriptinin nümunəsinə göstəririk. Bu skripti Biopython Testləri kataloquna yerləşdirəniz, run\_tests.py onu tapacaq və içindeki testləri yerinə yetirəcək:

```
$ python run_tests.py test_Ace ...
tamam test_AlignIO ...
tamam test_BioSQL ... tamam
test_BioSQL_SeqIO ... tamam
test_Biospam ... tamam
```

```
test_CAPS ... tamam
test_Clustalw ... tamam
...

86,127 saniyə ərzində 107 test keçirdi
```

### 23.2.1 Unittestdə n istifadə edə rək testin yazılması

Unittest çərçivəsi 2.1 versiyasından bəri Python-a daxil edilmişdir və Python Kitabxanası Referansında sənədlə şdirilmişdir (təsviyyə olunduğu kimi yastığınızın altında saxladığınızı bilirəm). [Unittest üçün onlayn](#) sənədlərdə var. Əgər siz vahid test sistemi ilə (və ya burun testi çərçivəsi kimi oxşar bir şey) tanışsanız, heç bir problem yaşamanızızaq. Biopython-da mövcud nümunələrə baxmağı da faydalı tapa bilərsiniz.

Başlamaq üçün kopyalayıb yapışdırıb ilə cəyiniz Biospam üçün minimal vahid test tipli test skripti buradadır:

```
Bio import Biospam -
dan idxal vahid testi
```

```
sinif BiospamTestAddition(unittest.TestCase):
 def test_addition1(self):
 nəticə = Biospam.addition(2, 3)
 self.assertEqual(nəticə, 5)

 def test_addition2(self):
 nəticə =
 Biospam.addition(9, -1) self.assertEqual(nəticə,
 8)

sinif BiospamTestDivision(unittest.TestCase):
 def test_division1(self):
 nəticə = Biospam.division(3.0, 2.0)
 self.assertAlmostEqual(nəticə, 1.5)

 def test_division2(self):
 nəticə =
 Biospam.division(10.0, -2.0) self.assertAlmostEqual(nəticə,
 -5.0)

əgər __adi__ == "__əsas__":
 runner = unittest.TextTestRunner(verbosity=2)
 unittest.main(testRunner=runner)
```

Bölmə testlərinə, yuvarlaqlaşdırma xətalarına görə testlərin uğursuz olmasının qarşısını almaq üçün assertEquals və zinə assertAlmostEqual istifadə edirik; təfərruatlar və unittest-də mövcud olan digər funksiyalar üçün Python sənədlərində ki vahid test bölməsinə baxın ([onlayn arayış](#)).

Bunlar vahid testə əsaslanan testlərinə əsas məqsəmləridir:

- Test nümunələri iunittest.TestCase-dənəldə edilən və birəsas aspektiniəhatə edən siniflərdə saxlanılır kodunuz
- Hər bir sınaq metodundanəvvəl və sonra işə salınmalı olan hərhansı təkrarlanan kod üçün quraşdırma və yırtma üsullarından istifadə edəbilərsiniz. Məsələn, quraşdırma metodu obyektin nümunəsini yaratmaq üçün istifadə ediləbilər

sınayırsınız və ya fayl idarə sini açın. TearDown hər hansı bir "sə liqə ni" yerinə yetirməlidir, məsələn, fayl dəstəyini bağlamaq.

- Testlərə test\_ prefiks qoymalıdır və hər bir test case hədiyyəsiniz işin xüsusi bir hissəsinə hata etməlidir. Sınıfda istədiyiniz qədər test edə bilərsiniz.
- Test skriptinin sonunda istifadə edə bilərsiniz

```
a ga r __adi__ == "__ə sas__":
runner = unittest.TextTestRunner(verbosity=2) unittest.main(testRunner=runner)
```

Skript özü tərəfindən işlədildikdə testləri yerinə yetirmək üçün (run\_tests.py-dən iddal etməkə və zinə). Bu skript işlədirsinizsə, aşağıdakı kimi bir şey görebilərsiniz:

```
$ python test_BiosspamMyModule.py test_addition1
(_main_.TestAddition) ... ok test_addition2 (_main_.TestAddition) ... ok
test_division1 (_main_.TestDivision) ... ok test_division2 (_main_) ...
```

-----  
0,059 saniyədə 4 test keçirdi

OK

- Hər testin nə etdiyini daha aydın göstərmək üçün siz hər bir testə sənədsə tirəni əlavə edə bilərsiniz. Bunlar testləri yerinə yetirərkən göstərilir, test uğursuz olarsa faydalı məlumat ola bilər.

Bio import Biosspam -  
dan iddal vahid testi

```
sinif BiosspamTestAddition(unittest.TestCase): def test_addition1(self):
```

```
 """Əlavə test"""
 nəticə = Biosspam.addition(2, 3)
 self.assertEqual(nəticə, 5)
```

```
def test_addition2(self):
 """İkinci əlavə testi"""
 nəticə = Biosspam.addition(9, -1)
 self.assertEqual(nəticə, 8)
```

```
sinif BiosspamTestDivision(unittest.TestCase): def test_division1(self):
```

```
 """İndi bölməni yoxlayaq"""
 nəticə = Biosspam.division(3.0, 2.0)
 self.assertAlmostEqual(nəticə, 1.5)
```

```
def test_division2(self):
 """İkinci bölmə testi"""
 nəticə = Biosspam.division(10.0, -2.0)
```

```
self.assertAlmostEqual(nəticə, -5.0)

əgər __adi__ == "__əsas__":
 runner = unittest.TextTestRunner(verbosity=2)
 unittest.main(testRunner=runner)
```

Skripti işə salmaq indi sizə göstərəcək:

```
$ python test_BiospamMyModule.py
```

İkinci əlavə testi... tamam

İndi bölməni yoxlayaqla... ok

İkinci bölmə testi... tamam

---

0,001 saniyədə 4 test keçirdi

OK

Əgər modulunuz docstring testlərini ehtiva edirsə (bax bölmə [23.3](#)), siz onları həyata keçiriləcək testlərə daxil etmək istəyə bilərsiniz. Əgər \_\_name\_\_ == "\_\_main\_\_" altındaki kodu dəyişdirərək bunu aşağıdakı kimi edə bilərsiniz: bu kimi görünmək üçün:

```
əgər __adi__ == "__əsas__":
 unittest_suite = unittest.TestLoader().loadTestsFromName("test_Biospam")
 doctest_suite = doctest.DocTestSuite(Biospam)
 suite = unittest.TestSuite((unittest_suite, doctest_suite))
 runner = unittest.TextTestRunner(until=runner.run(suite))
```

Bu, yalnız `python test_Biospam.py` programını yerinə yetirərkən docstring testlərini işə salmaq istəyirsinizsə müvafiqdir. Bəzi mürəkkəb iş vaxtından asılılıq yoxlanışı varsa.

Ümumiyyətlə, aşağıda izah edildiyi kimi `run_tests.py`-a əlavə etməklə docstring testlərini daxil edin.

## 23.3 Təlimatın yazılıması

Python modulları, sınıfları və funksiyaları docstrings istifadə edərək qurulmuş sənədləri stəkləyir. Doctest [çərçivəsi](#) (Python ilə daxil edilir) tərtibatçıya iş nümunələrini docstrings daxil etməyə və bu nümunələri riavtomatik sınaqdandan keçirməyə imkan verir.

Hal-hazırda Biopython-un yalnız bir hissəsi doktestlərdən ibarətdir. `run_tests.py` skripti doktestlərin işə salınmasına diqqət yetirir. Bu məqsədlə, `run_tests.py` skriptinin yuxarı hissəsində keçmək üçün modulların əlliətərtib edilmiş siyahısı var, bu da quraşdırılabilir yənisey isteğə bağlı xarici asılılıqlar (məsələn, Reportlab və NumPy kitabxanaları) olduqda vacibdir. Beləliklə, əgər siz Biopython modulundan sənədlərinə tələb etmək istəyirsinizsə, onların Biopython test paketində xaric edilməsi üçün modulunuzu daxil etmək üçün `run_tests.py`ni yeniləməlisiniz.

Hazırda `run_tests.py`-nin müvafiq hissəsi aşağıdakı kimi görünür:

```
Aşağıdakı modulların tarixi uğursuzluqları var. Bunlardan birini # düzənləndir, buradan silin!
```

```
EXCLUDE_DOCTEST_MODULES = ["Bio.PDB",
```

```
"Bio.PDB.AbstractPropertyMap",
```

```

 "Bio.Phylo.Applications._Fasttree", "Bio.Phylo._io",

 "Bio.Phylo.TreeConstruction",
 "Bio.Phylo._utils",
]

Onlayn fə aliyə ti olan modulları xaric edin # Onlar
standart olaraq istisna edilmir, onları istisna etmə k üçün --offline istifadə edin
ONLINE_DOCTEST_MODULES = ["Bio.Entrez", "Bio.ExPASy", "Bio.TogoWS"]

Numpy tə lə b edə n modullar üçün hə r hansı doctestlə rə sə ssizcə mə hə l
qoymayın! numpy
Yoxdursa : EXCLUDE_DOCTEST_MODULES.extend (
[
 "Bio.Affy.CelFile",
 "Bio.Cluster",
 #...
]
)

```

Qeyd edə k ki, biz doktestlə ri ilk növbə də sə nə dlə şdirmə kimi qə bul edirik, ona görə də siz tipik istifadə yə sadiq qalmalısınız. Sə hə şə rtlə ri və buna bə nzə rlə ri ilə bağlı ümumiyyə tlə müvə kkə b nümunə lə r xüsusi bir vahid testinə buraxılmalıdır. Nə zə rə alın ki, a gə r siz fayl tə hlili ilə bağlı doktestlə r yazmaq istə yirsinzə , faylin yerini müə yyə n etmə k işlə ri çə tinlə şdirir. İdeal olaraq kodun Testlə r kataloqundan işlə dilə cə yini güman edə n nisbi yollardan istifadə edin, bunun nümunə si üçün Bio.SeqIO doktestlə rinə baxın.

Yalnız docstring testlə rini işə salmaq üçün istifadə edin

```
$ python run_tests.py doctest
```

Nə zə rə alın ki, doctest sistemi kövrə kdir və çıxışınızın bütün sistemlə rdə uyğun olmasını tə min etmə k üçün diqqə tli olmaq lazımdır Biopython-un də stə klə diyi müxtə lif Python versiyaları (mə sə lə n, üzə n nöqtə nömrə lə rində ki fə rqlə r).

## 23.4 Də rslikdə doktestlə rin yazılması

Oxuduğunuz bu Tə limdə çox vaxt doctest kimi formatlaşdırılan çoxlu kod parçaları var. Bizim test\_Tutorial.py faylında öz sistemimiz var ki, Tə lim mə nbə yində kod parçalarını etiketlə mə k üçün Python doktestlə ri kimi işə salınsın. Bu, hə r bir Python blokundan a vvə l xüsusi %doctest şə rh sə tirlə ri a lava etmə klə işlə yir, mə sə lə n

```
%doctest
\begin{minted}{pycon} >>> Bio.Seq
import Seq >>> s = Seq("ACGT") >>> len(s) 4
\end{minted}
```

Çox vaxt kod nümunə lə ri öz-özünə deyil, a vvə iki Python blokundan davam edir. Burada biz burada göstə rildiyi kimi sehri şə rh %cont-doctest istifadə edirik:

```
%cont-doctest
\begin{minted}{pycon}
```

```
>>> s == "ACGT"
Doğrudur
\end{zərb edilmiş}
```

Xüsusi %doctest şərh xətti hər hansı bir nümunə məlumat faylıınız varsa istifadə etmək üçün işlək qovluğu (Sənəd/qovluqla əlaqəli) götürə bilər, məsələn, %doctest nümunələri Doc/examples qovluğundan, %doctest ./Tests/GenBank isə Tests/GenBank qovluğundan istifadə edəcək.

Kataloq arqumentindən sonra siz idxlal XXX-nin işləməli olduğunu göstərmək üçün lib:XXXəlavə edərək testi işə salmaq üçün mövcud olan hər hansı Python asılılığını təyin edə bilərsiniz, məsələn, %doctest nümunələri lib:numpy

Tutorial doctests-i aşağıdakılardan vasitəsilə həyata keçirə bilərsiniz:

```
$ python test_Tutorial.py
```

və ya:

```
$ python run_tests.py test_Tutorial.py
```

## 24-cü fə sil

# Buradan hara getmə k olar – Biopythona töhfə vermə k

### 24.1 Baq Hesabatları + Xüsusiyyə t Sorğuları

Biopython modulları haqqında rə y almaq bizim üçün çox vacibdir. Bu kimi açıq mə nbə li layihə lə r, çoxlu sayıda ianə cılə rin rə ylə rində n, sə hə hesabatlarından (və yamaqlardan!) çox faydalınır.

Xüsusiyyə t sorğularını və potensial sə hvilə ri müzakirə etmə k üçün ə sas forumlar [Biopython poçt siyahısıdır](#) və GitHub-da sorğular verir və ya çə kin.

Ə lava olaraq, yeni bir sə hv tapdiğinizi düşünürsünüzsa , onu <https://github.com/biopython/biopython/issues> ünvanında problem izlə yicimiza tə qdim edə bilə rsiniz. (bu, RedMine izlə yicisi olan köhnə Open Bioinformatika Fondunu ə və z etdi). Belə liklə , o, heç kimin Gə lə nlə r qutusunda basdırılmayacaq və unudulmayaçaq.

### 24.2 Poçt siyahıları və yeni gə lə nlə rə kömə k etmə k

Biz bütün istifadə lə rimizi ə sas Biopython poçt siyahısına daxil olmağa tə şviq edirik. Biopython sahə si ilə tanış olduqdan sonra biz sizə yeni başlayanların suallarına cavab vermə yə kömə k edə rik. Axi siz başlanğıc idiniz bir da fə .

### 24.3 Tə qdim olunan sə nə dlə r

Biz ya sə hv hesabatı və ya E-poçt Siyahısı vasitə silə rə y və ya töhfə lə ri qə bul etmə kdə n mə mnunuq. Bu tə limati oxuyarkə n, ola bilsin ki, sizi maraqlandıran bə zi mövzuların çatışmadığını və ya aydın şə kildə izah olunmadığını görmüşünüz. Biopython-un daxili sə nə dlə ri də var (docstrings, bunlar da [onlayndır](#)), yenə də hə r hansı bir boşluğu doldurmağa kömə k edə bilə rsiniz.

### 24.4 Mə tbə x kitablarına töhfə verə n nümunə lə r

22-ci Fə sildə izah edildiyi kimi , Biopython hazırda istifadə ci tə rə fində n verilə n "yemə k kitabı" nümunə lə rində n ibarə t wiki kolleksiyasına malikdir, <http://biopython.org/wiki/Category:Kitab> kitabı - Bə lkə buna ə lavə edə bilə rsiniz?

### 24.5 Platforma üçün paylamanın saxlanması

Biz hal-hazırda mə nbə kodu arxivlə rini (hə r hansı bir ə mə liyyat sistemi üçün uyğundur, ə gə r düzgün quraşdırma alə tlə ri quraşdırılırsa) və ə vvə lcə də n tə rtib edilmiş tə kə rlə ri <https://github.com/biopython/biopython-wheels> vasitə silə tə min edirik. ə sas ə mə liyyati ə hatə etmə k

sistemlə ri.

Əksərə səs Linux paylamalarında bu mənbə kodu buraxılışlarını götürən və onları Linux istifadəçilərinin asanlıqla quraşdırması üçün paketlərə tərtib edən nəfərlər var (asılılıqların qayğısına qalmalı və s.). Bu, həqiqətəndə əladır və bizə ibəttəki, çox minnətdən. Bu işə töhfə vermək istəyirsinizsə, lütfən, Linux paylamalarının bunu necə idarəetdiyi barədə ətraflı məlumat əldə edin. <https://github.com/conda-forge/biopython-feedstock> konda-forge komandasına təşəkkür edirik.

Aşağıda müəyyən platformalar üçün insanlara kömək etməyə başlaya biləcək bəzi məsləhətlər verilmişdir:

**Windows** – Sizə və ya C kompüterinizdə C kompilyatorunuza olduğundan və hər şeyi tərtib edib quraşdırıldıysınızda nə min olmalıdır (bu çətin məsələdir - bunu necə etmək barədə məlumat üçün Biopython quraşdırma təlimatlarına baxın).

**RPM-lər** – RPM-lərbəzi Linux platformalarında oldukça populyar paket sistemləridir. <http://www.rpm.org> saytında RPM-lərlə bağlı çoxlu sənədlər mövcuddur onlarla başlamağa kömək etmək üçün. Platformanız üçün RPM yaratmaq həqiqətəndən asandır. Siz sadəcə mənbədən paketi qura biləlisiniz (işləyən C kompilyatorunun olması vacibdir) - bu barədə daha çox məlumat üçün Biopython quraşdırma təlimatlarına baxın.

RPM etmək üçün sadəcə etməlisiniz:

```
$ python setup.py bdist_rpm
```

Bu, xüsusi platformanız üçün RPM və kataloq distində mənbə RPM yaradacaq. Bu RPM yaxşı və getməyə hazır olmalıdır, ona görədə etməli olduğunuz hər şey budur! Gözəl və asan.

**Macintosh** – Apple Mac OS X-ə keçidkən sonra Mac-də işlər xeyli asanlaşdı. Biz ümumiyyətlə onu başqa bir Unix variantı kimi qəbul edirik və Biopython-u mənbədən quraşdırmaq Linux-da olduğu kimi asandır.

Bütün GCC kompilyatorlarını və s. quraşdırmanın en asan yolu Apple X-Kodunu quraşdırmaqdır. Biz Mac OS X üçün klik və çalıştır quraşdırıcılarını təmin edə bilərik, lakin bu günə qədər heç bir tələb olmamışdır.

Paketə idəetdikdən sonra, hər şeyi yaxşı şəkildə quraşdırığına və düzgün işlədiyinə emin olmaq üçün onu sisteminizdə sınadın keçirin. Bu barədə yaxşı hiss etdikdən sonra GitHub-da bir sorğu göndərin və [Biopython poçt siyahımıza yazın](#). Siz bunu etdiniz. Təşəkkürler!

## 24.6 Köməkçi Vahid Testləri

Biopython-a əlavə etmək üçün heç bir yeni funksiyanız olmasa belə, lakin bəzi kod yazmaq istəyirsinizsə, lütfən, vahid testə hədəf dairəmizi genişləndirməyi düşünün. Biz [23-cü Fəsilin](#) hamisini bu mövzuya həsr etdik.

## 24.7 Töhfə Kodu

Biopython kodunun inkişafına qoşulmaq üçün Python-da biologiya ilə əlaqəli kod yaratmaq marağından başqa heç bir maneə yoxdur. Maraqlı ifadə etmək üçün en yaxşı yer Biopython poçt siyahılarıdır - sadəcə kodlaşdırma ilə maraqlandığınızı və hansı növ üzrində işləmək istədiyinizi bizə bildirin. Normalda, modulları kodlaşdırmadan və və onları müzakirə etməyə çalışırıq, çünki bu, yaxşı ideyalar yaratmağa kömək edir - o zaman dərhədə daxil olub kodlaşdırmağa başlamaq üçün çekinmeyin!

Əsas Biopython buraxılışı istifadəçilər üçün asanlaşdırmaq üçün kifayət qədər və bir-biri ilə işləyə bilən olmağa çalışır. Biopython-da istifadə etməyə çalışdığımız bəzi (kifayət qədər rəqəri-rəsmi) kodlaşdırma üslubları qaydaları <http://biopython.org/wiki/Contributing> ünvanında iştirak edən sənədlərdə oxuya bilərsiniz. Biz həmçinin testlərlə (reqressiya testi) rəcipəsi haqqında dəhaçox məlumat üçün Fəsil [23-ə](#) baxın) və sənədlərlə birlikdə paylamaya kodəlavə etməyə çalışırıq ki, hər şey mümkün qədər işlək və yaxşı sənədləşdirilsin (sənədlər daxil olmaqla). Bu, əlbəttəki, ən ideal və ziyyətdir, bir çox hallarda siyahıda digər insanları tapa bilərsiniz.

Siz onu ə lçatan etdikdə n sonra kodunuz üçün sə nə dlə r və ya daha çox test ə lavə etmə yə kömə k etmə yə hazır olacaqlar. Belə liklə , bu paraqrafi sonuncu kimi bitirmə k üçün işə başlayın!

Nə zə rə alın ki, kod töhfə si vermə k üçün onu töhfə vermə k və Biopython lisenziyası altında lisensiya vermə k üçün qanuni hüququnuz olmalıdır. Ə gə r hamısını özünüz yazmışınızsa və o, başqa koda ə saslanmayıbsa, bu problem olmamalıdır. Bununla belə , siz törə mə işə töhfə vermə k istə yırısınızsa , problemlə r var - mə sə lə n, GPL və ya LPGL lisenziyalı koduna ə saslanan bir şey lisenziyamızla uyğun gə lmə yə cə k. Bununla bağlı hə r hansı sualınız varsa, poçt siyahısında və ya GitHub-da mə sə lə ni müzakirə edin.

Biopython-a hə r hansı ə lavə lə r üçün narahatlıq doğuran başqa bir mə qam hə r hansı bir tikinti vaxtı və ya işlə mə müddə tində n asılılıqlarla bağlıdır. Ümumiyyə tlə , müstə qil bir alə tlə (BLAST, EMBOSS və ya ClustalW kimi) qarşılıqlı ə laqə yaratmaq üçün kod yazmaq böyük problem yaratmir. Bununla belə , başqa bir kitabxanadan hə r hansı asılılıq - hə ttə Python kitabxanası (xüsusilə NumPy kimi Biopython-u tə rtib etmə k və quraşdırmaq üçün lazım olan) ə lavə müzakirə yə ehtiyac duyur.

Ə lavə olaraq, paylanmaya uyğun olmadığını düşündürüyünüz, lakin ə lçatan etmə k istə diyiniz kodunuz varsa, biz Skript Mə rkə zini qoruyuruq (<http://biopython.org/wiki/Scriptcentral>) bioinformatika üçün Python-da sə rbə st mövcud kodun göstə ricilə ri var.

Ümid edirə m ki, bu sə nə d sizi Biopython haqqında onu sınamaq üçün kifayə t qə də r hə yə canlandırıb (və ə n çoxu Ə sas odur ki, töhfə verin!). Bütün yolu oxuduğunuz üçün tə şə kkür edirik!

## 25-ci fəsil

### Əlavə : Python haqqında faydalı məlumatlar

Python-da programlaşdırmağa çox vaxt sərf etmə misinizsə, Biopython istifadə edərkən yaranan bir çox sual və problemlər çox vaxt Python-un özü ilə bağlıdır. Bu bölmə Biopython kitabxanalarından istifadə edərkən tez-tez ortaya çıxan bəzi ideyaları və kodları təqdim etməyə çalışır (ən azı bizim üçün!). Bura daxil ola biləcək faydalı göstəricilər üçün hər hansı təklifləriniz varsa, lütfən, töhfə verin!

#### 25.1 Tutacaq nədir?

Tutacaqlar bu sənədlərdə olduqca tez-tez xatırlanır və eyni zamanda kifayət qədər çəşdiricidir (ən azı mənim üçün!). Əsasən, siz tutacaq mətn məlumatı ətrafında "sarğı" kimi düşünə bilərsiniz.

Tutacaqlar düz mətn məlumatı ilə müqayisədə (ən azı) iki üstünlük təmin edir:

- Onlar müxtəlif yollarla saxlanılan məlumatlarla işləmək üçün standart üsul təqdim edirlər. Mətn məlumatı faylda və ya yaddaşda saxlanılan sənətlər və ya komanda xətti programından və ya bəzi uzaq veb-saytda ola bilər, lakin tutacaq bütün bu formatlarda məlumatla işləmək üçün ümumi üsul təqdim edir.
- Onlar mətn məlumatının hamısını birdən deyil, tədrīcən oxunmasına imkan verir. Hamısını yükəməli olursanız, bütün yaddaşınızı istifadə edəcək nəhəng mətn faylları ilə məşğul olduğunuz zaman bu, həqiqətən vacibdir.

Tutacaqlar oxunan (məsələn, fayldan oxumaq) və ya yazılıan (məsələn, fayla məlumat yazmaq) mətn məlumatı ilə məşğul ola bilər. "Oxu" tutacığı halda, ümumi istifadə edilən funksiyalar tutacaqdan bütün mətn məlumatını oxuyan `read()` və `informasiyanı bir sənət oxuyan readline()` funksiyalarıdır. "Yazmaq" tutacaqları üçün müntəzəm olaraq `write()` funksiyasından istifadə olunur.

Tutacaqlar üçünən ümumi istifadə daxili Python funksiyasından istifadə etməklə hər yata keçirilən fayldan məlumat oxumaqdır. Burada, biz [buradan](#) yükəmə biləcəyiniz `m_cold.fasta` faylini idarə edirik (və ya Biopython mənbə koduna `Doc/examples/m_cold.fasta` kimidən edilmiş tapın).

```
>>> tutacaq = open("m_cold.fasta", "r")
>>> handle.readline()
>>> BE037100 MP14H09 MP Mesembryanthemum ...
>>>
```

Dəstəklər biopython-da təhlilçilərə məlumat ötürmək üçün müntəzəm olaraq istifadə olunur. Məsələn, Biopython 1.54-dən bəri `Bio.SeqIO` və `Bio.AlignIO`-dakıəsas funksiyalar sizə tutacaq yerinə fayl adından istifadə etməyə icazə verdi:

[Bio idxaldan SeqIO](#)

```
qeyd üçün SeqIO.parse ("m_cold.fasta", "fasta"): print(record.id, len(record))
```

Biopython-un köhnə versiyalarında siz tutacaqdən istifadə etmə li idiniz, mə sə lə n

#### Bio idxaldan SeqIO

```
tutacaq = open("m_cold.fasta", "r") SeqIO.parse
-də qeyd üçün (handle, "fasta"): print(record.id, len(record))
 handle.close()
```

Bu nümunə hə lə də faydalıdır - mə sə lə n, istə diyiniz gzip sıxılmış FASTA fayliniz olduğunu düşünə k tə hlil:

#### Bio import

SeqIO - dan gzip idxal edin

```
tutacaq = gzip.open("m_cold.fasta.gz", "rt") SeqIO.parse
(handle, "fasta"): print (record.id, len(record)) handle.close()
```

Düz mə tn faylları üçün analizatorlarımızla mə tn rejimində gzipdə n istifadə etmə k vacibdir (defolt ikili rejimdir). bzip2 sıxılmış faylları oxumaq da daxil olmaqla, buna bə nzə r daha çox nümunə üçün Bölmə 5.2-ə baxın .

### 25.1.1 Sə tirdə n tutacaq yaratmaq

Faydalı bir şey, bir sə tirdə olan mə lumiati sapa çevirə bilmə kdir. Aşağıdakı nümunə Python standart kitabxanasından StringIO istifadə edə rə k bunun necə edilə cə yini göstə rir:

```
>>> my_info = "Çox sə tirlə sə tir\n ." >>> çap (mə nim_mə lumatım)
```

```
Çox
sə tirdə n ibarə t sə tir. >>>
io importundan StringIO >>>
my_info_handle = StringIO (my_info) >>> first_line =
my_info_handle.readline() >>> print(first_line)
```

```
Sə tir
<BLANKLINE>
>>> second_line = my_info_handle.readline() >>> çap
(ikinci_sə tir) çoxlu sə tirlə rlə .
```

## Biblioqrafiya

- [1] Stephen F. Altschul, Warren Gish, Webb Miller, Eugene W. Myers, David J. Lipman: "Basic Local Alignment Search Tool". Molekulyar Biologiya Jurnalı 215 (3): 403–410 (1990). <https://doi.org/10.1016/S0022-2836%2805%2980360-2>.
- [2] Timothy L. Bailey və Charles Elkan: "Biopolimerlə rdə motivlə ri kə şf etmə k üçün gözlə ntilə rin maksimumlaşdırılması ilə qarışıq modelinin uyğunlaşdırılması", Molekulyar Biologiya üçün İntellektual Sistemlə r üzrə İkinci Beynə Ixalq Konfransın materialları 28-36. AAAI Press, Menlo Park, Kaliforniya (1994).
- [3] Douglas R. Cavener: "Drosophila və onurgalılarda tə rcümə başlanğıc yerlə rini ə hatə edə n konsensus ardıcılığının müqayisə si." Nuklein turşularının tə dqiqatı 15 (4): 1353–1361 (1987). <https://doi.org/10.1093/nar/15.4.1353>
- [4] Brad Chapman və Jeff Chang: "Biopython: Hesablama biologiyası üçün Python alə tlə ri". ACM SIGBIO Xə bə r bülleteni 20 (2): 15–19 (Avqust 2000).
- [5] Peter JA Cock, Tiago Antao, Jeffrey T. Chang, Brad A. Chapman, Cymon J. Cox, Andrew Dalke, Iddo Friedberg, Thomas Hamelryck, Frank Kauff, Bartek Wilczynski, Michiel JL de Hoon: "Biopython: bioloji hesablamalar və bioloji tə dqiqatlar üçün sə rbə st mövcud alə tlə r" Bioinformatika 25 (11), 1422–1423 (2009).
- [6] Peter JA Cock, Christopher J. Fields, Naohisa Goto, Michael L. Heuer, Peter M. Race: "Keyfiyyə t xalları olan ardıcılıqlar üçün Sanger FASTQ fayl formatı və Solexa/Illumina FASTQ variantları". Nuklein turşularının tə dqiqatı 38 (6): 1767–1771 (2010). <https://doi.org/10.1093/nar/gkp1137> <https://doi.org/10.1093/bioinformatics/btp163>
- [7] Athel Cornish-Bowden: "Nuklein turşusu ardıcılığında natamam müə yyə n edilmiş ə saslar üçün nomenklatura: Tövsiyə lə r 1984." Nuklein turşularının tə dqiqatı 13 (9): 3021–3030 (1985). <https://doi.org/10.1093/nar/13.9.3021>
- [8] Aaron E. Darling, Bob Mau, Frederick R. Blattner, Nicole T. Perna: "Mauve: konservlə şdirilmiş genomik ardıcılığın yenidə n tə şkili ilə çoxlu uyğunlaşması." Genom Araşdırma 14 (7): 1394–1403 (2004). <https://doi.org/10.1101/gr.2289704>
- [9] MO Dayhoff, RM Schwartz və BC Orcutt: "Zülallarda Tə kamül Də yişikliyi Modeli." Zülalların ardıcılığı və strukturu atlası, cild 5, Ə lavə 3, 1978: 345-352. Milli Biotibbi Araşdırmlar Fondu, 1979.
- [10] Michiel JL de Hoon, Seiya Imoto, John Nolan, Satoru Miyano: "Açıq mə nbə li klaster programı". Bioinformatika 20 (9): 1453–1454 (2004). <https://doi.org/10.1093/bioinformatics/bth078>
- [11] Richard Durbin, Sean R. Eddy, Anders Krogh, Graeme Mitchison: "Bioloji ardıcılığın tə hlili: Zülalların və nuklein turşularının ehtimal-abilistik modellə ri". Cambridge University Press, Kembric, Böyük Britaniya (1998).

- [12] Michiel B. Eisen, Paul T. Spellman, Patrick O. Brown, David Botstein: "Klaster analizi və genom boyu ifadə nümunə lərinin nümayışı". ABŞ Milli Elmlər Akademiyasının Materialları 95 (25): 14863–14868 (1998). <https://doi.org/10.1073/pnas.96.19.10943-c>
- [13] Nick Goldman və Ziheng Yang: "Zülal kodlayan DNT ardıcılılığı üçün nukleotidlərin dəyişdirilmə sinin kodon əsaslı modeli." Molekulyar Biologiya və Təkmil 11 (5) 725–736 (1994). <https://doi.org/10.1093/oxfordjournals.molbev.a040153>.
- [14] Gene H. Golub, Christian Reinsch: "Tək dəyişir parçalanması və ən kiçik kvadratların həlli". Avtomatik Hesablama üçün Handbook, 2, (Xətti Cəbr) (JH Wilkinson və C. Reinsch, red.), 134–151. Nyu York: Springer-Verlag (1971).
- [15] Gene H. Golub, Charles F. Van Loan: Matrix computations, 2nd nəşr (1989).
- [16] Thomas Hamelryck və Bernard Manderick: "PDB parser və struktur sinifi Python-da həyata keçirilir". Bioinformatika 19 (17): 2308–2310 (2003) <https://doi.org/10.1093/bioinformatics/btg299>.
- [17] Thomas Hamelryck: "Çoxölçülü indeks ağacından istifadə edərək yanraz nümunə lərinin effektiv identifikasiyası". Zülallar 51 (1): 96–108 (2003). <https://doi.org/10.1002/prot.10338>
- [18] Thomas Hamelryck: "Amin turşusunun iki tərəfi var; Yeni 2D ölçü həllidicinin məruz qalmasının fərqli görünüşünü təmin edir". Zülallar 59 (1): 29–48 (2005). <https://doi.org/10.1002/prot.20379>.
- [19] Steven Henikoff, Jorja G. Henikoff: "Zülal bloklarından amin turşusu əvəzedici matrislər". ABŞ Milli Elmlər Akademiyasının Materialları 89 (2): 10915–10919 (1992). <https://doi.org/10.1073/pnas.89.22.10915>.
- [20] Yukako Hihara, Ayako Kamei, Minoru Kanehisa, Aaron Kaplan və Masahiko Ikeuchi: "Yüksək işığa alışma zamanı siyanobakteriya gen ifadəsinin DNT mikroar-ray analizi". Bitki Hüceyrəsi 13 (4): 793–806 (2001). <https://doi.org/10.1105/tpc.13.4.793>.
- [21] Richard Hughey, Anders Krogh: "Ardıcılıq təhlili üçün gizli Markov modelləri: əsas metodun genişləndirilməsi və təhlili". Bioelmlərdə Kompyuter Tətbiqləri: CABIOS 12 (2): 95–107 (1996). <https://doi.org/10.1093/bioinformatics/12.2.95>
- [22] Florian Jupe, Leighton Pritchard, Graham J. Etherington, Katrin MacKenzie, Peter JA Cock, Frank Wright, Sanjeev Kumar Sharma1, Dan Bolser, Glenn J Bryan, Jonathan DG Jones, Ingo Hein: "NB-LRR geni daxilində NB-LRR geninin identifikasiyası və lokalizasiyası". BMC Genomics 13: 75 (2012). <https://doi.org/10.1186/1471-2164-13-75>
- [23] Voratas Kachitvichyanukul, Bruce W. Schmeiser: Binomial Random Variate Generation. ACM-in kommunikasiyaları 31 (2): 216–222 (1988). <https://doi.org/10.1145/42372.42381>
- [24] W. James Kent: "BLAT — BLAST kimi Alignment Tool". Genom Araşdırma 12: 656–664 (2002). <https://doi.org/10.1101/gr.229202>
- [25] Teuvo Kohonen: "Özünü təşkil edən xəritələr", 2-ci nəşr. Berlin; New York: Springer-Verlag (1997).
- [26] Anders Krogh, Michael Brown, I. Saira Mian, Kimmen Sjölander, David Haussler: "Hesablama biologiyasında Gizli Markov Modelləri: Zülal modelləşdirməsinə tətbiqləri". Molekulyar Biologiya Jurnalı 235 (5): 1501–1531 (1994). <https://doi.org/10.1006/jmbi.1994.1104>
- [27] Pierre L'Ecuyer: "Effektiv və Portativ Kombinə Təsadüfi Nömrələr Generatorları". Rabitə ACM 31 (6): 742–749, 774 (1988). <https://doi.org/10.1145/62959.62969>

- [28] Wen-Hsiung Li, Chung-I Wu, Chi-Cheng Luo: "Nükleotid və kodon də yişikliklə rının nisbi ehtimalını nə zə rə alaraq, nükleotidlə rin də yişdirilmə sinin sinonim və qeyri-sinonim də rə cə lə rini qiymə tlə ndirmə k üçün yeni bir üsul." Molekulyar Biologiya və Tə kamül 2 (2): 150-174 (1985). <https://doi.org/10.1093/oxfordjournals.molbev.a040343>
- [29] Heng Li, Bob Handsaker, Alec Wysoker, Tim Fennell, Jue Ruan, Nils Homer, Gabor Marth, Goncalo Abecasis, Richard Durbin: "The Sequence Alignment/Map format and SAMtools." Bioinformatika 25 (16): 2078–2079 (2009). <https://doi.org/10.1093/bioinformatics/btp352>
- [30] David R. Maddison, David L. Swofford, Wayne P. Maddison: "Nexus: Sistematiq Mə lumat üçün Genişlə ndirilə bilə n Fayl Formatı." Sistematiq Biologiya 46 (4): 590–621 (1997). <https://doi.org/10.1093/sysbio/46.4.590>.
- [31] Indraneel Majumdar, S. Sri Krishna, Nick V. Grishin: "PALSSE: Protein strukturlarından xə tti ikincili struktur elementlə ri ayırmaq üçün proqram." BMC Bioinformatika 6: 202 (2005). <https://doi.org/10.1186/1471-2105-6-202>.
- [32] V. Matys, E. Fricke, R. Geffers, E. Gossling, M. Haubrock, R. Hehl, K. Hornischer, D. Karas, AE Kel, OV Kel-Margoulis, DU Kloos, S. Land, B. Lewicki-Potapov, H. Reun Michaelch, R. S. "Rotert, H. Saxel, M. Scheer, S. Thiele, E. Wingender E: "TRANSFAC: transkripsiya tə nzimlə mə si, nümunə lə rdə n profillə rə ." Nuklein turşularının tə dqiqatı 31 (1): 374–378 (2003). <https://doi.org/10.1093/nar/gkg108>
- [33] Masatoshi Nei və Takashi Gojobori: "Sinonim və qeyri-sinonim nükleotid ə və zetmə lə rinin sayını qiymə tlə ndirmə k üçün sadə üsullar." Molekulyar Biologiya və Tə kamül 3 (5): 418–426 (1986). <https://doi.org/10.1093/oxfordjournals.molbev.a040410>
- [34] William R. Pearson, David J. Lipman: "Bioloji ardıcılığın müqayisə si üçün tə kmillə şdirilmiş alə tlə r". ABŞ Milli Elmlə r Akademiyasının Materialları 85 (8): 2444–2448 (1988). <https://doi.org/10.1073/pnas.85.8.2444>
- [35] Leighton Pritchard, Jennifer A. White, Paul RJ Birch, Ian K. Toth: "GenomeDiagram: genişmiqyaslı genomik mə lumatların vizuallaşdırılması üçün piton paketi". Bioinformatika 22 (5): 616–617 (2006). <https://doi.org/10.1093/bioinformatics/btk021>
- [36] Caroline Proux, Douwe van Sinderen, Juan Suarez, Pilar Garcia, Victor Ladero, Gerald F. Fitzgerald, Frank Desiere, Harald Brüssow: "Sfi21-asid bakteriyasının müqayisə li genomikası ilə tə svir edilə n fag taksonomiyası dilemması". Bakteriologiya jurnalı 184 (21): 6026–6036 (2002). <https://doi.org/10.1128/JB.184.21.6026-6036.2002>
- [37] Peter Rays, Ian Longden, Alan Bleasby: "EMBOSS: The European Molecular Biology Open Software Suite." Genetikada Trendlə r 16 (6): 276-277 (2000). [https://doi.org/10.1016/S0168-9525\(00\)02024-2](https://doi.org/10.1016/S0168-9525(00)02024-2)
- [38] Alok Saldanha: "Java Treeview – mikroarray mə lumatlarının genişlə ndirilə bilə n vizuallaşdırılması". Bioinformatika 20 (17): 3246–3248 (2004). <https://doi.org/10.1093/bioinformatics/bth349>
- [39] Thomas D. Schneider, Gary D. Stormo, Larry Gold: "Nu-cleotide sequences haqqında bağlama saytlarının mə lumat mə zmunu". Molekulyar Biologiya Jurnalı 188 (3): 415–431 (1986). [https://doi.org/10.1016/0022-2836\(86\)90165-8](https://doi.org/10.1016/0022-2836(86)90165-8)
- [40] Adrian Schneider, Gina M. Cannarozzi və Gaston H. Gonnet: "Empirik kodon ə və zetmə matrisi". BMC Bioinformatika 6: 134 (2005). <https://doi.org/10.1186/1471-2105-6-134>
- [41] Robin Sibson: "SLINK: Tə k keçidli klaster metodu üçün optimal effektiv alqoritm". Kompüter Jurnalı 16 (1): 30-34 (1973). <https://doi.org/10.1093/comjnl/16.1.30>

- [42] Guy St C. Slater, Ewan Birney: "Bioloji ardıcılığın müqayisə si üçün evristikanın avtomatlaşdırılmış nə slı." BMC Bioinformatika 6: 31 (2005). <https://doi.org/10.1186/1471-2105-6-31>
- [43] George W. Snedecor, William G. Cochran: Statistik üsullar. Ames, Ayova: Ayova Dövlə t Universiteti Nə şriyyati (1989).
- [44] Martin Steinegger, Markus Meier, Milot Mirdita, Harald Vöhringer, Stephan J. Haunsberger, Johannes Stöding: "Sürə tli uzaqdan homologiya aşkarlanması və dərin protein annotasiyası üçün HH-suite3." BMC Bioinformatics 20: 473 (2019). <https://doi.org/10.1186/s12859-019-3019-7>
- [45] Eric Talevich, Brandon M. Invergo, Peter JA Cock, Brad A. Chapman: "BioPhylo: Biopython-da filogenetik ağacların işlənməsi, təhlili və vizuallaşdırılması üçün vahid alətdə sti". BMC Bioinformatics 13: 209 (2012). <https://doi.org/10.1186/1471-2105-13-209>
- [46] Pablo Tamayo, Donna Slonim, Jill Mesirov, Qing Zhu, Sutisak Kitareewan, Ethan Dmitrovsky, Eric S. Lander, Todd R. Golub: "Özünü təşkil edən nəxəritələrlə gen ifadə nümunələrinin şərh edilməsi: Hematopoietik differensiasiya üçün üsullar və tətbiq". ABŞ Milli Elmlər Akademiyasının Materialları 96 (6): 2907-2912 (1999). <https://doi.org/10.1073/pnas.96.6.2907>
- [47] Ian K. Toth, Leighton Pritchard, Paul RJ Birch: "Müqayisəli genomika enterobakterial bitki patogenini nəyin yaratdığını ortaya qoyur". Fitopatologyanın İllik İcmalı 44: 305-336 (2006). <https://doi.org/10.1146/annurev.phyto.44.070505.143444>
- [48] John W. Tukey: "Kəşfiyyat məlumatlarının təhlili". Reading, Massa.: Addison-Wesley Pub. Co (1977).
- [49] Géraldine A. van der Auwera, Jaroslaw E. Król, Haruo Suzuki, Brian Foster, Rob van Houdt, Celeste J. Brown, Max Mergeay, Eva M. Top: "C. metallidurans CH34-də tutulan plazmidlər: broad-plaad-host ailəsinin müəyyən edilməsi." Antonie van Leeuwenhoek 96 (2): 193-204 (2009). <https://doi.org/10.1007/s10482-009-9316-9>
- [50] Michael S. Waterman, Mark Eggert: "TRNT-rRNA müqayisələrinə tətbiqi ilə ən yaxşı ardıcıl uyğunlaşmalar üçün yeni alqoritm". Molekulyar Biologiya Jurnalı 197 (4): 723-728 (1987). [https://doi.org/10.1016/0022-2836\(87\)90478-5](https://doi.org/10.1016/0022-2836(87)90478-5)
- [51] Ziheng Yang və Rasmus Nielsen: "Realist təkamül modelləri altında sinonim və qeyri-sinonim əvəzetmədə rəsədinin tətbiqi". Molekulyar Biologiya və Təkamül 17 (1): 32-43 (2000). <https://doi.org/10.1093/oxfordjournals.molbev.a026236>
- [52] Ka Yee Yeung, Walter L. Ruzzo: "Gen ifadə məlumatlarının klasterləşdirilməsi üçün Əsas Komponent Analizi". Bioinformatics 17 (9): 763-774 (2001). <https://doi.org/10.1093/bioinformatics/17.9.763>