Machine Translated by Google

Biopython Dərsliyi və Körpə

Jeff Chang, Brad Chapman, Iddo Friedberg, Thomas Hamelryck,
Michiel de Hoon, Peter Cock, Tiago Antao, Eric Taleviç, Bartek Wilczy'nski

Son yeniləmə – 22 dekabr 2023-cü il (Biopython 1.82)

Machine Translated by Google

İçində kilə r

1 Giriş 1.1 Biopython 10
nə dir? . . . 10
1.2 Biopython paketində nə tapa bilə rə m . . . 10
1.3 Biopython-un quraşdırılması . . . 11
1.4 Tez-tez verilə n suallar (FAQ) . . . 11

2 Tez Başlanğıc – Biopython ilə nə edə bilə rsiniz? 15
2.1 Biopython-un nə tə min etdiyinə ümumi baxış . . . 15
2.2 Ardıcıllıqla işlə mə k . . . 15
2.3 İstifadə nümunə si . . . 16
2.4 Ardıcıllıq fayl formatlarının tə hlili . . . 16
2.4.1 Sadə FASTA tə hlil nümunə si . . . 17
2.4.2 Sadə GenBank tə hlil nümunə si . . . 17
2.4.3 Mə n tə hlil etmə yi sevirə m – xahiş edirə m, bu barə də danışmağı dayandırmayın! . . . 18
2.5 Bioloji mə lumat bazaları ilə ə laqə . . . 18
2.6 Bundan sonra nə etmə li . . . 19

3 Ardıcıllıq obyektlə ri 3.1 20
Ardıcıllıqlar simlə r kimi fə aliyyə t göstə rir . . . 20
3.2 Ardıcıllığın dilimlə nmə si . . . 21
3.3 Seq obyektlə rinin simlə rə çevrilmə si . . . 22
3.4 Ardıcıllıqların birlə şdirilmə si və ya ə lavə edilmə si . . . 22
3.5 Korpusun də yişdirilmə si . . . 23
3.6 Nukleotid ardıcıllığı və (ə ks) komplementlə r . . . 23
3.7 Transkripsiya . . . 24
3.8 Tə rcümə . . . 25
3.9 Tə rcümə Cə dvə llə ri . . . 27
3.10 Seq obyektlə rinin müqayisə si . . . 29
3.11 Ardıcıllığın mə zmunu namə lum olan ardıcıllıqlar . . . 29
3.12 Qismə n müə yyə n edilmiş ardıcıllıq mə zmunlu ardıcıllıqlar . . . 29
3.13 MutableSeq obyektlə ri . . . 30
3.14 Alt ardıcıllıqların tapılması . . . 31
3.15 Birbaşa simlə rlə işlə mə k . . . 32

4 Ardıcıllıq annotasiya obyektlə ri 4.1 SeqRecord 34
obyekti . . . 34
4.2 SeqRecord-un yaradılması . . . 35
4.2.1 SeqRecord obyektlə ri sıfırdan . . . 35
4.2.2 FASTA fayllarından SeqRecord obyektlə ri . . . 36
4.2.3 GenBank fayllarından SeqRecord obyektlə ri . . . 37

Machine Translated by Google

4.3 Obyektlə rin xüsusiyyə ti, yeri və mövqeyi 38
4.3.1 SeqFeature obyektlə ri 38
4.3.2 Və zifə lə r və yerlə r 39
4.3.3 Xüsusiyyə t və ya yerlə tə svir olunan ardıcıllıq 41
4.4 Müqayisə 42
4.5 İstinadlar 43
4.6 Formatlaşdırma metodu 43
4.7 SeqRecord-un dilimlə nmə si 44
4.8 SeqRecord obyektlə rinin ə lavə edilmə si 46
4.9 Ə ks tamamlayan SeqRecord obyektlə ri 48

5 Ardıcıllıqla Giriş/Çıxış 5.1 Ardıcıllıqların 50
tə hlili və ya oxunması 50
5.1.1 Ardıcıllıq fayllarının oxunması 50
5.1.2 Ardıcıllıq faylında qeydlə rin tə krarlanması 51
5.1.3 Ardıcıllıq faylında qeydlə rin siyahısının ə ldə edilmə si 52
5.1.4 Mə lumatların çıxarılması 53
5.1.5 Mə lumatların də yişdirilmə si 55
5.2 Sıxılmış fayllardan ardıcıllıqların tə hlili 55
5.3 Şə bə kə də n ardıcıllıqların tə hlili 56
5.3.1 GenBank qeydlə rinin şə bə kə də n tə hlili 57
5.3.2 Şə bə kə də n SwissProt ardıcıllığının tə hlili 58
5.4 Faylları Lüğə t kimi sıralayın 58
5.4.1 Faylları Lüğə t kimi sıralayın – Yaddaşda 59
5.4.2 Faylları Lüğə tlə r kimi ardıcıllıqla sıralayın – İndekslə nmiş fayllar 62
5.4.3 Faylları Lüğə tlə r kimi ardıcıllıqla sıralayın – Verilə nlə r bazası indeksli fayllar 63
5.4.4 Sıxılmış faylların indekslə şdirilmə si 64
5.4.5 Müzakirə 65
5.5 Ardıcıllıq fayllarının yazılması 66
5.5.1 Gediş-gə lişlə r 67
5.5.2 Ardıcıl fayl formatları arasında konvertasiya 68
5.5.3 Ardıcıllıq faylının ə ks tamamlayıcılara çevrilmə si 69
5.5.4 SeqRecord obyektlə rinizi formatlanmış sə tirlə r kimi ə ldə etmə k 70
5.6 Aşağı sə viyyə li FASTA və FASTQ analizatorları 70

6 Ardıcıl düzülmə lə r 72
6.1 Obyektlə rin hizalanması 72
6.1.1 Ardıcıllıq və koordinatlardan Alignment obyektinin yaradılması 72
6.1.2 Hizalanmış ardıcıllıqlardan Alignment obyektinin yaradılması 73
6.1.3 Ümumi düzülmə atributları 74
6.2 Hizalanmanın dilimlə nmə si və indekslə şdirilmə si 75
6.3 Hizalanma haqqında mə lumatın ə ldə edilmə si 77
6.3.1 Düzlə şdirmə forması 77
6.3.2 Düzə lişlə rin müqayisə si 77
6.3.3 Düzlə nmiş ardıcıllıqların indekslə rinin tapılması 77
6.3.4 Eyniliklə rin, uyğunsuzluqların və boşluqların hesablanması 79
6.3.5 Hə rf tezliklə ri 79
6.3.6 Ə və zetmə lə r 79
6.3.7 Massivlə r kimi düzülmə lə r 80
6.4 Düzlə şdirmə üzrə ə mə liyyatlar 81
6.4.1 Düzə lişin çeşidlə nmə si 81
6.4.2 Düzə lişin tə rs tamamlanması 81

Machine Translated by Google

6.4.3 Düzə lişlə rin ə lavə edilmə si 82
6.4.4 Cüt ardıcıllıqla düzülmə nin xə ritə lə şdirilmə si 83
6.4.5 Çox ardıcıl düzülüşün xə ritə lə şdirilmə si 86
6.5 Alignments sinfi 88
6.6 Oxuma və yazma hizaları 89
6.6.1 Oxu düzülmə lə ri 90
6.6.2 Düzə lişlə rin yazılması 90
6.6.3 Çap düzülmə lə ri 90
6.7 Fayl formatlarının hizalanması 92
6.7.1 Aligned FASTA 92
6.7.2 ClustalW 94
6.7.3 Stokholm 96
6.7.4 PHYLIP çıxış faylları 99
6.7.5 QARŞIQ 102
6.7.6 GCG Çox Ardıcıllıq Formatı (MSF) 104
6.7.7 Bə raə tdə n azad etmə k 106
6.7.8 NEXUS 109
6.7.9 BLAST və ya FASTA-dan cə dvə l çıxışı 110
6.7.10 HH-suite çıxış faylları 112
6.7.11 A2M 116
6.7.12 Bə növşə yi Genişlə ndirilmiş Multi-FastA (xmfa) formatı 117
6.7.13 Sequence Alignment/Map (SAM) 122
6.7.14 Genişlə ndirilə bilə n Brauzer Mə lumatı (BED) 126
6.7.15 böyük yataq 128
6.7.16 Pattern Space Layout (PSL) . 6.7.17 bigPsl 131
. 133
6.7.18 Çoxlu Alignment Format (MAF) . 6.7.19 bigMaf 136
. 139
6.7.20 UCSC zə nciri fayl formatı 142

7 Cüt ardıcıllıqla düzülmə 7.1 Ə sas istifadə . 145
. 145
7.2 Cütlü düzlə ndirici obyekti 149
7.3 Ə və zetmə balları 150
7.4 Affin boşluğu balları 151
7.5 Ümumi boşluq balları 152
7.6 Ə vvə lcə də n müə yyə n edilmiş ə və zetmə matrisində n və boşluq ballarından istifadə 154
7.7 Düzə lişlə rin tə krarlanması 155
7.8 Arxa iplə düzülmə 157
7.9 Ə və zetmə matrislə ri 158
7.9.1 Massiv obyektlə ri 158
7.9.2 Cüt ardıcıllığın düzülüşündə n ə və zedici matrisin hesablanması 161
7.9.3 Çox ardıcıl düzülüşündə n ə və zedici matrisin hesablanması 163
7.9.4 Fayldan Massiv obyektlə rinin oxunması 165
7.9.5 Ə vvə lcə də n tə yin edilmiş ə və zedici matrislə rin yüklə nmə si 167
7.10 Nümunə lə r 168
7.11 Ümumilə şdirilmiş qoşa düzülmə lə r 170
7.11.1 Ə və zedici matrisa və ə lifbadan istifadə edə rə k ümumilə şdirilmiş cüt düzülmə lə r . . 170 . . . . . .
7.11.2 Uyğunluq/uyğunsuzluq xallarından və ə lifbadan istifadə etmə klə ümumilə şdirilmiş cüt-cüt düzülmə lə r . . 171 . .
7.11.3 Uyğunluq/uyğunsuzluq xallarından və tam ə də dlə rin ardıcıllığından istifadə etmə klə ümumilə şdirilmiş qoşa düzülmə lə r . 172
7.11.4 Ə və zedici matrisa və tam ə də d ardıcıllığından istifadə edə rə k ümumilə şdirilmiş cüt düzülmə lə r . . 173
7.12 Kodon düzülmə lə ri . . . . 174 . . . . . .

Machine Translated by Google

7.12.1 Nukleotid ardıcıllığının amin turşusu ardıcıllığına uyğunlaşdırılması . . . 174
7.12.2 Kodon ardıcıllığının çoxsaylı ardıcıl düzülmə sinin yaradılması . . . 176
7.12.3 Kodon düzülmə sinin tə hlili . . . 177

8 Çox Ardıcıl Alignment obyektlə ri 182
8.1 Ardıcıllığın uyğunlaşdırılmasının tə hlili və ya oxunması . . . 182
8.1.1 Tə k düzülmə lə r . . . 183
8.1.2 Çoxsaylı Düzə lişlə r . . . 185
8.1.3 Birmə nalı düzülmə lə r . . . 187
8.2 Yazı düzülmə lə ri . . . 189
8.2.1 Ardıcıl uyğunlaşdırma fayl formatları arasında konvertasiya . . . 190
8.2.2 Düzlə şdirmə obyektlə rini formatlanmış sə tirlə r kimi ə ldə etmə k . . . 193
8.3 Düzə lişlə rin manipulyasiyası . . . 194
8.3.1 Dilimlə rin düzülmə si . . . 194
8.3.2 Massivlə r kimi düzülmə lə r . . . 196
8.3.3 Ə və zetmə lə rin hesablanması . . . 197
8.3.4 İcmal mə lumatların hesablanması . . . 197
8.3.5 Tez konsensus ardıcıllığının hesablanması . . . 198
8.3.6 Mövqe Xüsusi Xal Matrislə ri . . . 199
8.3.7 Mə lumat mə zmunu . . . 200
8.4 Yeni üslubda Alignment obyektinin ə ldə edilmə si . . . 202
8.5 Çox ardıcıl düzülüşündə n ə və zedici matrisin hesablanması . . . 202
8.6 Hizalama Alə tlə ri . . . 204
8.6.1 ClustalW . . . 204
8.6.2 Ə ZƏ LƏ . . . 206
8.6.3 İğneyi və suyu QARŞAT . . . 206

9 Pairwise2 istifadə edə rə k cüt-cüt hizalamalar 209

10 BLAST 10.1 212
İnternet üzə rində n BLAST işlə tmə k . . . 212
10.2 BLAST-ın yerli olaraq işə salınması . . . 214
10.2.1 Giriş . . . 214
10.2.2 Müstə qil NCBI BLAST+ . . . 215
10.2.3 BLAST-ın digə r versiyaları . . . 215
10.3 BLAST çıxışının tə hlili . . . 215
10.4 BLAST rekord sinfi . . . 217
10.5 PSI-BLAST ilə işlə mə k . . . 218
10.6 RPS-BLAST ilə işlə mə k . . . 218

11 BLAST və digə r ardıcıl axtarış vasitə lə ri 11.1 SearchIO 221
obyekt modeli . . . 221
11.1.1 Sorğu Nə ticə si . . . 222
11.1.2 Vurun . . . 227
11.1.3 HSP . . . 230
11.1.4 HSPFragment . . . 233
11.2 Standartlar və konvensiyalar haqqında qeyd . . . 234
11.3 Axtarış çıxış fayllarının oxunması . . . 235
11.4 İndekslə mə ilə böyük axtarış çıxışı faylları ilə işlə mə k . . . 236
11.5 Axtarış çıxış fayllarının yazılması və konvertasiyası . . . 236

Machine Translated by Google

12 NCBI-nin Entrez verilə nlə r bazalarına daxil olmaq 238
12.1 Entrez Tə limatları . . . 239
12.2 EInfo: Entrez verilə nlə r bazası haqqında mə lumatların ə ldə edilmə si . . . 240
12.3 ESSearch: Entrez verilə nlə r bazalarının axtarışı . . . 242
12.4 EPost: İdentifikatorların siyahısının yüklə nmə si . . . 243
12.5 EXülasə : Ə sas identifikatorlardan xülasə lə rin alınması . . . 244
12.6 EFetch: Entrez-də n tam qeydlə r endirilir . . . 244
12.7 ELink: NCBI Entrez-də ə laqə li elementlə rin axtarışı . . . 247
12.8 EGQuery: Qlobal Sorğu - axtarış şə rtlə ri üçün hesablanır . . . 248
12.9 ESpell: Orfoqrafiya tə kliflə rinin alınması . . . 249
12.10 Böyük Entrez XML fayllarının tə hlili . . . 249
12.11HTML qaçış simvolları . . . 250
12.12 Sə hvlə rin idarə edilmə si . . . 251
12.13 İxtisaslaşdırılmış tə hlilçilə r . . . 253
12.13.1 Medline qeydlə rinin tə hlili . . . 253
12.13.2 GEO qeydlə rinin tə hlili . . . 255
12.13.3 UniGene qeydlə rinin tə hlili . . . 256
12.14 Proksidə n istifadə . . . 258
12.15 Nümunə lə r . . . 258
12.15.1 PubMed və Medline . . . 258
12.15.2 Entrez Nucleotide qeydlə rinin axtarışı, yüklə nmə si və tə hlili . . . 260
12.15.3 GenBank qeydlə rinin axtarışı, yüklə nmə si və tə hlili . . . 261
12.15.4 Orqanizmlə rin nə sillə rinin tapılması . . . 263
12.16 Tarixçə də n və WebEnv-də n istifadə . . . 264
12.16.1 Tarixçə də n istifadə edə rə k ardıcıllıqların axtarılması və yüklə nmə si . . . 264
12.16.2 Tarixçə də n istifadə edə rə k tezislə rin axtarışı və yüklə nmə si . . . 265
12.16.3 Sitatların axtarışı . . . 266

13 Swiss-Prot və ExpPASy 267
13.1 Swiss-Prot fayllarının tə hlili . . . 267
13.1.1 Swiss-Prot qeydlə rinin tə hlili . . . 267
13.1.2 Swiss-Prot açar sözünün və kateqoriya siyahısının tə hlili . . . 269
13.2 Prosite qeydlə rinin tə hlili . . . 270
13.3 Prosite sə nə dlə ri qeydlə rinin tə hlili . . . 271
13.4 Enzim qeydlə rinin tə hlili . . . 272
13.5 ExpASy serverinə daxil olmaq . . . 273
13.5.1 İsveçrə -Prot rekordunun bə rpası . . . 273
13.5.2 Swiss-Prot axtarışı . . . 274
13.5.3 Prosite və Prosite sə nə dlə rinin ə ldə edilmə si . . . 274
13.6 Prosite mə lumat bazasının skan edilmə si . . . 275

14 Going 3D: PDB modulu 14.1 Kristal struktur 277
fayllarının oxunması və yazılması . . . 277
14.1.1 mmCIF faylının oxunması . . . 277
14.1.2 MMTF formatında faylların oxunması . . . 278
14.1.3 PDB faylının oxunması . . . 278
14.1.4 PQR faylının oxunması . . . 279
14.1.5 PDB XML formatında faylların oxunması . . . 279
14.1.6 mmCIF fayllarının yazılması . . . 279
14.1.7 PDB fayllarının yazılması . . . 279
14.1.8 PQR fayllarının yazılması . . . 280
14.1.9 MMTF fayllarının yazılması . . . 280

Machine Translated by Google

14.2 Struktur tə msili 281
14.2.1 Quruluş 283
14.2.2 Model 283
14.2.3 Zə ncir 283
14.2.4 Qalıq 284
14.2.5 Atom 285
14.2.6 Strukturdan xüsusi Atom/Qalıq/Zə ncirvari/Model çıxarmaq 286
14.3 Bozukluk 286
14.3.1 Ümumi yanaşma 286
14.3.2 Nizamsız atomlar 286
14.3.3 Tə hlükə li qalıqlar 287
14.4 Hetero qalıqlar 287
14.4.1 Ə laqə dar problemlə r 287
14.4.2 Su qalıqları 287
14.4.3 Digə r hetero qalıqlar 288
14.5 Struktur obyekti üzrə naviqasiya 288
14.6 Strukturların tə hlili 291
14.6.1 Mə safə lə rin ölçülmə si 291
14.6.2 Bucaqların ölçülmə si 291
14.6.3 Burulma bucaqlarının ölçülmə si 291
14.6.4 Daxili koordinatlar - mə safə lə r, bucaqlar, burulma bucaqları, mə safə qrafiklə ri və s 291
14.6.5 Atom-atom kontaktlarının müə yyə n edilmə si 299
14.6.6 İki strukturun üst-üstə qoyulması 300
14.6.7 İki ə laqə li strukturun qalıqlarının bir-biri ilə xə ritə lə şdirilmə si 300
14.6.8 Yarım Sfera Ekspozisiyasının Hesablanması 300
14.6.9 İkinci də rə cə li strukturun müə yyə n edilmə si 301
14.6.10 Qalıq də rinliyinin hesablanması 301
14.7 PDB fayllarında ümumi problemlə r 301
14.7.1 Nümunə lə r 302
14.7.2 Avtomatik düzə liş 303
14.7.3 Fatal sə hvlə r 303
14.8 Protein Mə lumat Bankına daxil olmaq 303
14.8.1 Protein Mə lumat Bankından strukturların yüklə nmə si 303
14.8.2 Bütün PDB-nin yüklə nmə si 304
14.8.3 PDB-nin yerli nüsxə sini yeni saxlamaq 304
14.9 Ümumi suallar 304
14.9.1 Bio.PDB nə də rə cə də yaxşı sınaqdan keçirilib? 304
14.9.2 Nə qə də r sürə tlidir? 304
14.9.3 Molekulyar qrafika də stə yi varmı? 305
14.9.4 Bio.PDB-də n kim istifadə edir? 305

15 Bio.PopGen: Populyasiya genetikası 306
15.1 GenePop 306

16 Bio.Phylo ilə Filogenetika 308
16.1 Demo: Ağacda nə var? 308
16.1.1 Ağacın içə risində budaqların rə nglə nmə si 309
16.2 I/O funksiyaları 312
16.3 Ağaclara baxmaq və ixrac etmə k 313
16.4 Tree və Clade obyektlə rində n istifadə 314
16.4.1 Axtarış və keçid üsulları 315
16.4.2 İnformasiya üsulları 316

Machine Translated by Google

16.4.3 Də yişiklik üsulları . . . 317
16.4.4 PhyloXML ağaclarının xüsusiyyə tlə ri . . . 318
16.5 Xarici proqramların işə salınması . . . 318
16.6 PAML inteqrasiyası . . . 319
16.7 Gə lə cə k planlar . . . 319

17 Bio.motiflə rdə n istifadə edə rə k motivlə rin ardıcıl tə hlili 321
17.1 Motif obyektlə ri . . . 321
17.1.1 Nümunə lə rdə n motivin yaradılması . . . 321
17.1.2 Konsensus ardıcıllığının ə ldə edilmə si . . . 323
17.1.3 Motivin tə rs tamamlanması . . . 324
17.1.4 Motifin dilimlə nmə si . . . 324
17.1.5 Nisbi entropiya . . . 324
17.1.6 Ardıcıllıq loqosunun yaradılması . . . 325
17.2 Oxu motivlə ri . . . 325
17.2.1 JASPAR . . . 326
17.2.2 MEME . . . 332
17.2.3 TRANSFAC . . . 335
17.3 Yazı motivlə ri . . . 338
17.4 Mövqe-Çə ki Matrislə ri . . . 339
17.5 Mövqe Xüsusi Qiymə tlə ndirmə Matrislə ri . . . 340
17.6 Nümunə lə rin axtarışı . . . 341
17.6.1 Də qiq uyğunluqların axtarışı . . . 342
17.6.2 PSSM xalından istifadə edə rə k uyğunluqların axtarışı . . . 342
17.6.3 Hesab hə ddinin seçilmə si . . . 343
17.7 Hə r bir motiv obyekti ə laqə li Mövqe Xüsusi Qiymə tlə ndirmə Matrisinə malikdir . . . 344
17.8 Motivlə rin müqayisə si . . . 346
17.9 Yeni motivin tapılması . . . 347
17.9.1 MEME . . . 347
17.10 Faydalı bağlantılar . . . 348

18 Klaster tə hlili 18.1 349
Mə safə funksiyaları . . . 350
18.2 Klaster xassə lə rinin hesablanması . . . 354
18.3 Bölmə alqoritmlə ri . . . 355
18.4 İerarxik qruplaşma . . . 358
18.5 Özünü tə şkil edə n xə ritə lə r . . . 362
18.6 Ə sas Komponent Tə hlili . . . 364
18.7 Cluster/TreeView tipli faylların idarə edilmə si . . . 365
18.8 Hesablama nümunə si . . . 370

19 GenomeDiagram daxil olmaqla qrafika 19.1 372
GenomeDiagram . . . 372
19.1.1 Giriş . . . 372
19.1.2 Diaqramlar, treklə r, xüsusiyyə t-də stlə r və xüsusiyyə tlə r . . . 372
19.1.3 Yuxarıdan aşağıya nümunə . . . 373
19.1.4 Aşağıdan yuxarıya nümunə . . . 374
19.1.5 SeqFeature olmayan funksiyalar . . . 376
19.1.6 Xüsusiyyə t başlıqları . . . 376
19.1.7 Xüsusiyyə t işarə lə ri . . . 377
19.1.8 Ox işarə lə ri . . . 379
19.1.9 Gözə l nümunə . . . 379

Machine Translated by Google

19.1.10 Çoxlu treklə r 384
19.1.11 Treklə r arasında keçidlə r 388
19.1.12 ə lavə seçimlə r 393
19.1.13 Köhnə kodun çevrilmə si 393
19.2 Xromosomlar 394
19.2.1 Sadə Xromosomlar 394
19.2.2 Annotasiyalı Xromosomlar 396

20 KEGG 20.1 398
KEGG qeydlə rinin tə hlili 398
20.2 KEGG API sorğusu 398

21 Bio.fenotip: fenotipik mə lumatların tə hlili 21.1 Fenotip 401
Mikroarraylar 401
21.1.1 Fenotip Mikroarray verilə nlə rinin tə hlili 401
21.1.2 Fenotip Mikroarray verilə nlə rinin manipulyasiyası 402
21.1.3 Fenotip Mikroarray verilə nlə rinin yazılması 405

22 Yemə k də ftə ri – Onunla gözə l işlə r 22.1 Ardıcıllıq faylları 406
ilə işlə mə k 406
22.1.1 Ardıcıllıq faylının süzülmə si 406
22.1.2 Randomize genomların istehsalı 407
22.1.3 CDS qeydlə rinin FASTA faylının tə rcümə si 409
22.1.4 FASTA faylının böyük hə rflə rində ardıcıllığın yaradılması 409
22.1.5 Ardıcıllıq faylının çeşidlə nmə si 410
22.1.6 FASTQ faylları üçün sadə keyfiyyə tli filtrlə mə 411
22.1.7 Astar sıralarının kə silmə si 412
22.1.8 Adapter ardıcıllığının kə silmə si 414
22.1.9 FASTQ fayllarının çevrilmə si 415
22.1.10 FASTA və QUAL fayllarının FASTQ fayllarına çevrilmə si 417
22.1.11 FASTQ faylının indekslə şdirilmə si 417
22.1.12 SFF fayllarının konvertasiyası 418
22.1.13 Açıq oxu çə rçivə lə rinin müə yyə n edilmə si 419
22.2 Ardıcıllığın tə hlili və sadə süjetlə r 421
22.2.1 Ardıcıllığın uzunluqlarının histoqramı 421
22.2.2 Ardıcıllığın qrafiki GC% 422
22.2.3 Nukleotid nöqtə lə rinin qrafiklə ri 424
22.2.4 Oxunma mə lumatlarının ardıcıllaşdırılmasının keyfiyyə t ballarının planlaşdırılması 427
22.3 BioSQL – ardıcıllıqların ə laqə li verilə nlə r bazasında saxlanması 427

23 Biopython test çə rçivə si 23.1 Testlə rin icrası . 429
. 429
23.1.1 Tox istifadə edə rə k testlə rin aparılması 430
23.2 Yazı testlə ri 430
23.2.1 Unittestdə n istifadə edə rə k testin yazılması 431
23.3 Tə limlə rin yazılması 433
23.4 Də rslikdə doktestlə rin yazılması 434

Machine Translated by Google

24 Buradan hara getmə k olar – Biopython-a töhfə 436
24.1 Baq Hesabatları + Xüsusiyyə t Sorğuları 436
24.2 Poçt siyahıları və yeni gə lə nlə rə kömə k etmə k 436
24.3 Də stə klə yici Sə nə dlə r 436
24.4 Mə tbə x kitablarına töhfə verə n nümunə lə r 436
24.5 Platforma üçün paylamanın saxlanması 436
24.6 Kömə kçi Vahid Testlə ri 437
24.7 Töhfə Kodu 437

25 Ə lavə : Python 25.1 haqqında faydalı mə lumatlar 439
Tutacaq nə dir? 439
25.1.1 Sə tirdə n tutacaq yaratmaq 440

Machine Translated by Google

Fə sil 1

Giriş

1.1 Biopython nə dir?

Biopython sə rbə st mövcud Python toplusudur (https://www.python.org) hesablama molekulyar biologiya üçün modullar. Python elmi hesablamalar üçün geniş istifadə olunan obyekt yönümlü, şə rh olunan, çevik bir dildir. Python öyrə nmə k asandır, çox aydın sintaksisə malikdir və C, C++ və ya FORTRAN dillə rində yazılmış modullarla asanlıqla genişlə ndirilə bilə r. 2000-ci ildə yarandığı gündə n [4], Biopython davamlı olaraq bütün dünyada böyük bir könüllülə r qrupu tə rə fində n inkişaf etdirilir və saxlanılır.

Biopython veb saytı (http://www.biopython.org) bioinformatikadan istifadə və tə dqiqat üçün Python ə saslı proqram tə minatının tə rtibatçıları üçün modullar, skriptlə r və veb bağlantılar üçün onlayn resurs tə qdim edir. Biopython-a müxtə lif bioinformatika fayl formatları (BLAST, Clustalw, FASTA, Genbank,...), onlayn xidmə tlə rə çıxış (NCBI, Expasy,...), standart ardıcıllıq sinfi, ardıcıllığın uyğunlaşdırılması və motiv analizi alə tlə ri, klasterlə şdirmə alqoritmlə ri, struktur biologiya modulu və analiz üçün filoqen modulları daxildir.

1.2 Biopython paketində nə tapa bilə rə m

Ə sas Biopython buraxılışları bir çox funksiyaya malikdir, o cümlə də n:

- Bioinformatika fayllarını Python üçün istifadə edilə bilə n mə lumat strukturlarında tə hlil etmə k imkanı, o cümlə də n də stə k aşağıdakı formatlar:
  - Partlayış çıxışı – hə m müstə qil, hə m də WWW Blast-dan
  - Clustalw
  - FASTA
  - GenBank
  - PubMed və Medline
  - Enzyme və Prosite kimi ExPASy faylları
  - "dom" və "lin" faylları daxil olmaqla SCOP
  - UniGene
  - SwissProt
- Də stə klə nə n formatlardakı fayllar qeydlə r üzrə tə krarlana bilə r və ya indekslə şdirilə bilə r. Lüğə t interfeysi.
- Populyar onlayn bioinformatika istiqamə tlə ri ilə mə şğul olmaq üçün kod, mə sə lə n:

Machine Translated by Google

– NCBI – Blast, Entrez və PubMed xidmə tlə ri

– ExpPASy – Swiss-Prot və Prosite girişlə ri, hə mçinin Prosite axtarışları

- Ümumi bioinformatika proqramları üçün interfeyslə r, mə sə lə n:

  - NCBI-də n müstə qil partlayış

  – Clustalw alignment proqramı

  – EMBOSS komanda xə tti alə tlə ri

- Ardıcıllıqlar, ardıcıllıqlardakı idlə r və ardıcıllıq xüsusiyyə tlə ri ilə mə şğul olan standart ardıcıllıq sinfi.
- Tə rcümə , transkripsiya və çə ki kimi ardıcıllıqlar üzə rində ümumi ə mə liyyatları yerinə yetirmə k üçün alə tlə r hesablamalar.
- Ə n Yaxın Qonşular, Naive Bayes və ya Də stə k Vektorundan istifadə edə rə k mə lumatların tə snifatını hə yata keçirmə k üçün kod Maşınlar.
- Ə və zetmə nin yaradılması və öhdə sində n gə lmə k üçün standart üsul daxil olmaqla, uyğunlaşmalarla mə şğul olmaq üçün kod matrislə r.
- Paralellə şdirilə bilə n tapşırıqları ayrı-ayrı proseslə rə bölmə yi asanlaşdıran kod.
- Ə sas ardıcıllıqla manipulyasiyalar, tə rcümə lə r, BLASTing və s. etmə k üçün GUI ə saslı proqramlar.
- Geniş sə nə dlə r və modullardan istifadə ilə bağlı yardım, o cümlə də n bu fayl, on-line wiki-documen-tation, internet saytı və poçt siyahısı.
- BioPerl və BioJava tə rə fində n də stə klə nə n ardıcıl verilə nlə r bazası sxemi olan BioSQL ilə inteqrasiya layihə lə r.

Ümid edirik ki, bu sizə Biopython-u yüklə mə k və istifadə etmə yə başlamaq üçün çoxlu sə bə blə r verə cə k!

1.3 Biopython-un quraşdırılması

Biopython üçün bütün quraşdırma mə lumatları yenilə nmə sini asanlaşdırmaq üçün bu sə nə ddə n ayrılmışdır.

Qısa versiya pip install biopython istifadə edir, ə sas README-ə baxın digə r seçimlə r üçün fayl.

1.4 Tez-tez verilə n suallar (FAQ)

1. Elmi nə şrdə Biopython-a necə istinad edə bilə rə m?
   Lütfə n, ə sas Biopython istinadı kimi ə rizə qeydimizi [5, Cock et al., 2009] göstə rin. Bundan ə lavə , lütfə n, lazım olduqda aşağıdakı siyahıdan hə r hansı nə şrlə rə istinad edin, xüsusə n də Biopython daxilində ki xüsusi modullar üçün istinad kimi (ə traflı mə lumatı vebsaytımızda tapa bilə rsiniz):

   - Layihə nin rə smi elanı üçün: [4, Chapman and Chang, 2000]; • Bio.PDB üçün: [16, Hamelryck

   və Manderick, 2003]; • Bio.Cluster üçün: [10, De Hoon et al., 2004]; •

   Bio.Graphics.GenomeDiagram üçün: [35, Pritchard et al., 2006];

   - Bio.Phylo və Bio.Phylo.PAML üçün: [45, Talevich et al., 2012]; • Biopython, BioPerl,

   BioRuby, BioJava və EMBOSS-da də stə klə nə n FASTQ fayl formatı üçün: [6, Cock et

   al., 2010].

Machine Translated by Google

2. “Biopython” hə rfini necə böyük hə rflə yazmalıyam? "BioPython" yaxşıdır?

Düzgün baş hə rf “BioPython” deyil, “Biopython”dur (baxmayaraq ki, bu, BioPerl, BioJava və BioRuby ilə uyğun olardı).

3. Biopython proqram tə minatı necə lisenziyalaşdırılır?

Biopython, Biopython Lisenziya Müqavilə sinə ə sasə n paylanır. Bununla belə , Biopy-thon 1.69-un buraxılışından bə ri bə zi fayllar seçdiyiniz Biopython Lisenziya Müqavilə si və ya BSD 3 Maddə Lisenziyası ə sasında açıq şə kildə ikili lisenziyaya malikdir. Bu, sonradan bu ikili lisenziyalaşdırma yanaşması altında bütün Biopython-u tə klif etmə k niyyə ti ilə dir.

4. Biopython loqosu nə dir və necə lisenziyalaşdırılır?

2017-ci ilin iyul ayına və Biopython 1.70 buraxılışına olan mə lumata görə , Biopython loqosu kiçik hə rflə yazılmış “biopython” sözünün üstündə ikiqat sarmal tə şkil edə n sarı və mavi ilandır. O, Patrick Kunzmann tə rə fində n hazırlanmışdır və bu loqo sizin seçdiyiniz Biopython Lisenziya Müqavilə si və ya BSD 3 Maddə Lisenziyası ə sasında ikili lisenziyalıdır.

Bundan ə vvə l Biopython loqosu 2003-cü ildə Henrik Vesterqaard və Tomas Hamelrik tə rə fində n açıq müsabiqə çə rçivə sində hazırlanmış “BIOPYTHON” sözünün ə trafında qoşa sarmal meydana gə tirə n iki sarı ilan idi.

5. Hə r buraxılışda nə lə rin yeni olduğunu göstə rə n də yişiklik jurnalınız varmı?

Mə nbə koduna daxil olan NEWS.rst faylına baxın (ə vvə lcə yalnız NEWS adlanır) və ya GitHub-da ə n son XƏ BƏ RLƏ R faylını oxuyun.

6. Çap ə mrlə rimdə nə sə hv olur?

Biopython 1.77-də n etibarə n biz yalnız Python 3-ü də stə klə yirik, ona görə də bu də rslik Python 3 üslubunda çap funksiyasından istifadə edir.

7. Biopython-un hansı versiyasını quraşdırdığımı necə öyrə nə bilə rə m?

Bunu istifadə edin:

```
>>> Bio import >>>
çap (Bio.__versiya__)
```

“İdxal Bio” xə tti uğursuz olarsa, Biopython quraşdırılmayıb. Qeyd edə k ki, bunlar versiyadan ə vvə l və sonra ikiqat alt xə ttdir. İkinci sə tir uğursuz olarsa, versiyanız çox köhnə lmişdir.

Ə gə r versiya sə tri “1.66+” kimi bir artı ilə bitirsə , sizin rə smi buraxılışınız yoxdur, lakin hə min versiya buraxıldıqdan sonra inkişaf kodunun köhnə şə kli var. Bu adlandırma 2016-cı ilin iyun ayına qə də r Biopython 1.68-ə qə də r istifadə edilmişdir.

Machine Translated by Google

Ə gə r versiya sə tri “1.68.dev0” kimi “.dev<number>” ilə başa çatırsa, yenə sizin rə smi buraxılışınız yoxdur, bunun ə və zinə hə min versiyanın buraxılmasından ə vvə l inkişafda olan kodun snapşotu var.

8. Bu sə nə din ə n son versiyası haradadır?

Biopython mə nbə kodu arxivini yüklə sə niz, o, hə m HTML, hə m də PDF formatında müvafiq versiyanı daxil edə cə k. Bu sə nə din ə n son nə şr edilmiş versiyası (hə r buraxılışda yenilə nir) onlayndır:

- http://biopython.org/DIST/docs/tutorial/Tutorial.html • http://biopython.org/DIST/docs/tutorial/Tutorial.pdf

9. Ardıcıl müqayisə lə rimdə sə hv nə dir?

Biopython 1.65-də Seq və MutableSeq siniflə rinin (və alt siniflə rinin) str(seq1) == str(seq2) ilə açıq şə kildə edə bilə cə yiniz sadə sə tir ə saslı müqayisə də n istifadə etmə sinə sə bə b olan ə sas də yişiklik oldu.

Biopython-un köhnə versiyaları id(seq1) == id(seq2) ilə açıq şə kildə edə bilə cə yiniz Seq obyektlə ri üçün nümunə ə saslı müqayisə də n istifadə edə rdi.

Ə gə r hə lə də Biopython-un köhnə versiyalarını də stə klə mə k lazımdırsa, problemlə rin qarşısını almaq üçün bu açıq formalardan istifadə edin. Bölmə 3.10-a baxın .

10. Bio.SeqIO və Bio.AlignIO hansı fayl formatlarını oxuyur və yazır?

Daxili sə nə d sə tirlə rini yoxlayın (Bio import SeqIO-dan, sonra kömə k edin(SeqIO)) və ya http://biopython-a baxın. org/wiki/SeqIO və http://biopython.org/wiki/AlignIO ə n son siyahı üçün vikidə .

11. Nə üçün Bio.SeqIO və Bio.AlignIO funksiyaları fayl adlarını tə hlil etmir, oxumur və yazmır? Onlar tutacaqlarda israr et!

Sizə Biopython 1.54 və ya daha yeni versiya lazımdır və ya sadə cə tutacaqlardan açıq şə kildə istifadə edin (Bölmə 25.1-ə baxın). Mə lumatlarınızı yazdıqdan sonra çıxış tutacaqlarını açıq şə kildə bağlamağı xatırlamaq xüsusilə vacibdir.

12. Nə üçün Bio.SeqIO.write() və Bio.AlignIO.write() funksiyaları tə k qeyd və ya düzülmə ni qə bul etmir? Onlar bir siyahı və ya iteratorda israr edirlə r!

Sizə Biopython 1.54 və ya daha yeni versiya lazımdır və ya bir elementin siyahısını yaratmaq üçün sadə cə elementi [...] ilə sarın.

13. Niyə str(...) mə nə Seq obyektinin tam ardıcıllığını vermir?

Sizə Biopython 1.45 və ya daha yeni versiya lazımdır.

14. Niyə Bio.Blast ə n son düz mə tn NCBI partlayış çıxışı ilə işlə mir?

NCBI BLAST alə tlə rində n düz mə tn çıxışını də yişdirmə yə davam edir və analizatorumuzu yeni saxlamaq davamlı bir mübarizə dir. Biopython-un ə n son versiyasını istifadə etmirsinizsə , tə kmillə şdirmə yə cə hd edə bilə rsiniz. Bununla belə , biz (və NCBI) ə və zinə kompüter proqramı tə rə fində n oxunmaq üçün nə zə rdə tutulmuş XML çıxışından istifadə etmə yi tövsiyə edirik.

15. Bio.Entrez.efetch() istifadə edə n skriptim niyə işlə mə yi dayandırdı?

Bu, 2012-ci ilin fevralında EFetch 2.0-ı tə qdim edə n NCBI də yişikliklə ri ilə ə laqə dar ola bilə r. Birincisi, onlar standart qaytarma rejimlə rini də yişdilə r - yə qin ki, zə nginizə retmode="text" ə lavə etmə k istə rdiniz. İkincisi, indi şə xsiyyə tlə rin siyahısını necə tə qdim etmə k barə də daha sə rtdirlə r – Biopython 1.59-dan sonra siyahını avtomatik olaraq vergüllə ayrılmış sə tirə çevirir.

16. Niyə Bio.Blast.NCBIWWW.qblast() NCBI BLAST veb saytı ilə eyni nə ticə lə ri vermir?

Siz eyni variantları göstə rmə lisiniz – NCBI tez-tez vebsaytdakı standart parametrlə ri tə nzimlə yir və onlar artıq QBLAST standartlarına uyğun gə lmir. Boşluq cə zaları və gözlə mə hə ddi kimi şeylə ri yoxlayın.

17. Niyə mə n SeqRecord obyektlə rini birlikdə ə lavə edə bilmirə m?

Sizə Biopython 1.53 və ya daha yeni versiya lazımdır.

Machine Translated by Google

18. Bio.SeqIO.index_db() niyə işlə mir? Modul yaxşı idxal edir, lakin indeks db funksiyası yoxdur!

Sizə Biopython 1.57 və ya daha yeni versiya (və SQLite3 də stə yi ilə Python) lazımdır.

19. MultipleSeqAlignment obyekti haradadır? Bio.Align modulu yaxşı idxal edir, lakin bu sinif deyil orda!

Sizə Biopython 1.54 və ya daha yeni versiya lazımdır. Alternativ olaraq, köhnə Bio.Align.Generic.Alignment sinfi onun bə zi funksional imkanlarını də stə klə yir, lakin indi bundan istifadə etmə k tövsiyə edilmir.

20. Niyə mə n birbaşa proqram paketlə rində n komanda xə tti alə tlə rini işə sala bilmirə m?

Sizə Biopython 1.55 və ya daha yeni versiya lazımdır, lakin bunlar Biopython 1.78-də köhnə lmişdir. Birbaşa Python alt proses modulundan istifadə etmə yi düşünün.

21. Kod üçün kataloqa baxdım, amma nə sə edə n kodu tapa bilmə dim. Harada gizlə nib?

Bir şeyi bilmə k lazımdır ki, biz kodu __init__.py fayllarına yerlə şdiririk. Bu faylda kodu axtarmağa öyrə şmə misinizsə , bu, çaşqınlıq yarada bilə r. Bunu etmə yimizin sə bə bi idxalı istifadə çilə r üçün asanlaşdırmaqdır. Mə sə lə n, Bio.GenBank idxal GenBank-dan "tə krarlanan" idxal etmə k ə və zinə , sadə cə Bio import GenBank-dan istifadə edə bilə rsiniz.

22. Bio.Fasta niyə işlə mir?

Biz Biopython 1.51-də (Avqust 2009) Bio.Fasta modulunu lə ğv etdik və onu Biopython 1.55-də (Avqust 2010) çıxardıq. DEPRECATED.rst- də ə və zinə Bio.SeqIO-dan istifadə etmə k üçün köhnə kodu necə çevirə cə yini göstə rə n qısa bir nümunə var. fayl.

Daha ümumi suallar üçün, Python FAQ sə hifə lə ri https://docs.python.org/3/faq/index.html faydalı ola bilə r.

Machine Translated by Google

Fə sil 2

Tez Başlanğıc – Biopython ilə nə edə bilə rsiniz?

Bu bölmə sizi Biopython ilə tez bir zamanda işə salmaq və mövcud olanlara və ondan necə istifadə olunacağına dair ümumi mə lumat vermə k üçün nə zə rdə tutulmuşdur. Bu bölmə də ki bütün nümunə lə r Python haqqında bə zi ümumi iş biliklə rinə sahib olduğunuzu və Biopython-u sisteminizdə uğurla quraşdırdığınızı güman edir. Python-da tə zə lə nmə li olduğunuzu düşünürsünüzsə , ə sas Python veb saytı işə başlamaq üçün kifayə t qə də r pulsuz sə nə dlə r tə qdim edir (https://docs.python.org/3/).

Kompüterdə çox bioloji iş internetdə verilə nlə r bazası ilə ə laqə ni nə zə rdə tutduğundan, bə zilə ri nümunə lə rin işlə mə si üçün işlə yə n internet bağlantısı da tə lə b olunacaq.

İndi bunların hamısı yoldan çıxdı, gə lin Biopython ilə nə edə bilə cə yimizə keçə k.

2.1 Biopython-un nə tə min etdiyinə ümumi baxış

Girişdə qeyd edildiyi kimi, Biopython kompüterdə işlə yə n bioloqları maraqlandıran "şeylə r"lə mə şğul olmaq imkanı verə n kitabxanalar toplusudur. Ümumiyyə tlə , bu o demə kdir ki, siz ə n azı bir qə də r proqramlaşdırma tə crübə niz (ə lbə ttə ki, Python-da!) və ya heç olmasa proqramlaşdırmanı öyrə nmə yə marağınız olmalıdır.
Biopython-un işi tə krar istifadə edilə bilə n kitabxanalar tə qdim etmə klə proqramçı kimi işinizi asanlaşdırmaqdır ki, siz xüsusi bir fayl formatının tə hlilinin daxili xüsusiyyə tlə rinə diqqə t yetirmə k ə və zinə , diqqə tinizi xüsusi maraqlandıran sualınıza cavab vermə yə yönə ldə bilə siniz (ə lbə ttə ki, mövcud olmayan tə hlilçi yazıb onu Biopython-a töhfə vermə klə kömə k etmə k istə yirsinizsə , zə hmə t olmasa davam edin!). Belə liklə , Biopython-un işi sizi xoşbə xt etmə kdir!

Biopython haqqında qeyd etmə k lazım olan bir şey odur ki, o, tez-tez "eyni şeyi etmə k" üçün bir neçə üsul tə qdim edir. Son buraxılışlarda işlə r yaxşılaşdı, lakin bu, hə lə də sinir bozucu ola bilə r, çünki Python-da ideal olaraq bir şey etmə k üçün bir düzgün yol olmalıdır. Bununla belə , bu da real fayda ola bilə r, çünki o, sizə kitabxanalar üzə rində çoxlu rahatlıq və nə zarə t imkanı verir. Də rslik sizə hə r şeyi etmə k üçün ümumi və ya asan yolları göstə rmə yə kömə k edir ki, siz sadə cə olaraq işlə ri yerinə yetirə bilə siniz. Alternativ imkanlar haqqında daha çox öyrə nmə k üçün Kulinariya Kitabına baxın (Fə sil 22, burada bə zi sə rinlə şdirmə fə ndlə ri və mə slə hə tlə ri var) və "docstrings"də qurulmuşdur (Python yardım ə mri və ya API sə nə dlə ri vasitə silə) və ya nə ticə də kodun özü.

2.2 Ardıcıllıqla işlə mə k

Mübahisə li (ə lbə ttə !), bioinformatikada mə rkə zi obyekt ardıcıllıqdır. Belə liklə , biz 3-cü Fə sildə daha ə traflı müzakirə edə cə yimiz Seq obyekti olan ardıcıllıqla işlə mə k üçün Biopython mexanizmlə rinə qısa bir girişlə başlayacağıq .

Çox vaxt ardıcıllıqlar haqqında düşünə ndə beynimdə 'AGTACACTGGT' kimi hə rflə r silsilə si olur.
Bu ardıcıllıqla belə Seq obyektini aşağıdakı kimi yarada bilə rsiniz - ">>>" Python ə mrini tə msil edir

Machine Translated by Google

ardınca nə yazacağınız:

```
>>> Bio.Seq -də n idxal Seq >>> my_seq =
Seq("AGTACACTGGT") >>> my_seq

Seq('AGTACACTGGT') >>>
çap (my_seq)
AGTACACTGGT
```

Seq obyekti də stə klə diyi metodlara görə Python sə tirində n fə rqlə nir. Bunu sadə bir simlə edə bilmə zsiniz:

```
>>> my_seq
Seq('AGTACACTGGT') >>>
my_seq.complement()
Seq('TCATGTGACCA') >>>
my_seq.reverse_complement()
Seq('ACCAGTGTACT')
```

Növbə ti ə n vacib sinif SeqRecord və ya Sequence Record-dur. Bu, identifikator, ad və tə svir daxil olmaqla ə lavə annotasiya ilə ardıcıllığı (Seq obyekti kimi) saxlayır. Ardıcıllıq fayl formatlarını oxumaq və yazmaq üçün Bio.SeqIO modulu SeqRecord obyektlə ri ilə işlə yir, bunlar aşağıda tə qdim edilə cə k və 5-ci Fə sildə daha ə traflı ə hatə olunacaq.

Bu, Biopython ardıcıllığı sinfinin ə sas xüsusiyyə tlə rini və istifadə lə rini ə hatə edir. İndi Biopython kitabxanaları ilə qarşılıqlı ə laqə nin nə olduğu haqqında bir az tə sə vvürünüz var, bioloji fayl formatları ilə mə şğul olmağın ə ylə ncə li, ə ylə ncə li dünyasına getmə yin vaxtı gə ldi!

2.3 İstifadə nümunə si

Analizlə rə və Biopython ilə ə laqə li hə r şeyə keçmə zdə n ə vvə l gə lin gördüyümüz hə r şeyi motivasiya etmə k və hə yatı daha maraqlı etmə k üçün bir nümunə yaradaq. Axı bu də rslikdə biologiya olmasaydı, niyə onu oxumağı istə yə rdin?

Mə n bitkilə ri sevdiyim üçün düşünürə m ki, biz sadə cə bitki ə saslı nümunə yə sahib olmalıyıq (orada olan digə r orqanizmlə rin bütün pə rə stişkarlarından üzr istə yirə m!). Yerli istixanamıza bu yaxınlarda etdiyimiz sə fə ri başa vurduqdan sonra birdə n bizdə Lady Slipper Orchids ilə inanılmaz bir və svə sə yarandı (niyə görə sə n, Flickr-da Lady Slipper Orchids fotolarına baxın , və ya Google Şə kil Axtarışını sınayın).

Tə bii ki, sə hlə blə r tə kcə baxmaq üçün gözə l deyil, hə m də tə kamül və sistematika ilə mə şğul olan insanlar üçün son də rə cə maraqlıdır. Belə liklə , tutaq ki, biz Lady Slipper-in tə kamülünün molekulyar tə dqiqi üçün maliyyə tə klifi yazmağı düşünürük və artıq hansı növ tə dqiqatların aparıldığını və buna necə ə lavə edə bilə cə yimizi görmə k istə rdik.

Bir az oxuduqdan sonra biz tapırıq ki, Lady Slipper sə hlə blə ri Orchidaceae ailə sinə və Cypripedioideae alt ailə sinə aiddir və 5 nə sildə n ibarə tdir: Cypripedium, Paphiopedilum, Phragmipedium, Selenipedium və Mexipedium.

Bu, bizə daha çox mə lumat ə ldə etmə yə başlamaq üçün kifayə t qə də r imkan verir. Belə liklə , Biopython alə tlə rinin bizə necə kömə k edə bilə cə yinə baxaq. Biz Bölmə 2.4-də ardıcıllığın tə hlili ilə başlayacağıq , lakin sə hlə blə r də daha sonra geri qayıdacaq - mə sə lə n, biz PubMed-də sə hlə blə r haqqında sə nə dlə ri axtaracağıq və 12-ci Fə sildə GenBank-dan ardıcıllıq mə lumatlarını çıxaracağıq, 13-cü Fə sildə müə yyə n sə hlə b zülallarından Swiss-Prot-dan mə lumat çıxaracağıq və çoxlu ardıcıllıqla ClustalW və ya 62-ci ardıcıllıqla düzülmüş zülallarla işlə yə cə yik .

2.4 Ardıcıllıq fayl formatlarının tə hlili

Bir çox bioinformatika işinin böyük bir hissə si bioloji mə lumatları saxlamaq üçün nə zə rdə tutulmuş bir çox fayl formatı ilə mə şğul olmağı ə hatə edir. Bu fayllar maraqlı bioloji mə lumatlar ilə yüklə nir və xüsusi problem tə hlildir

Machine Translated by Google

bu faylları bir növ proqramlaşdırma dili ilə manipulyasiya edə bildiyiniz formata çevirin. Bununla belə , bu faylları tə hlil etmə k tapşırığı formatların kifayə t qə də r müntə zə m olaraq də yişə bilmə si və formatların ə n yaxşı hazırlanmış tə hlilçilə ri belə poza bilə n kiçik incə liklə rə malik olması ilə mə yus ola bilə r.

İndi biz qısaca Bio.SeqIO modulunu tə qdim edə cə yik – siz 5-ci Fə sildə daha çox mə lumat ə ldə edə bilə rsiniz. Dostlarımız, qadın terlik orkide üçün onlayn axtarışla başlayacağıq. Bu girişi sadə saxlamaq üçün biz sadə cə olaraq NCBI veb-saytından ə l ilə istifadə edirik. Entrez onlayn axtarışından (https://www.ncbi.nlm.nih.gov/nuccore/?term=Cypripedioideae) istifadə edə rə k, NCBI-da nukleotid verilə nlə r bazasına nə zə r salaq. Cypripedioideae mə tnində n bə hs edə n hə r şey üçün (bu, qadın başmaqlı sə hlə blə rin alt ailə sidir).

Bu də rslik ilkin olaraq yazılan zaman bu axtarış bizə yalnız 94 hit verdi, biz onları FASTA formatlı mə tn faylı və GenBank formatlı mə tn faylı (files ls orchid.fasta) kimi saxladıq. və ls orchid.gbk, hə mçinin Doc/examples/ altında Biopython mə nbə koduna daxil edilmişdir.

Bu gün axtarış aparsanız, yüzlə rlə nə ticə ə ldə edə cə ksiniz! Də rsliyi izlə yə rkə n, eyni gen siyahısını görmə k istə yirsinizsə , yuxarıdakı iki faylı endirin və ya onları Biopython mə nbə kodundakı docs/examples/-də n kopyalayın. Bölmə 2.5- də biz Python daxilində bu kimi axtarışın necə aparılacağına baxacağıq.

2.4.1 Sadə FASTA tə hlil nümunə si

Xanım terlik orkide FASTA faylını açsanız ls orchid.fasta sevimli mə tn redaktorunuzda faylın belə başladığını görə cə ksiniz:

```
>gi|2765658|emb|Z78533.1|CIZ78533 C.irapeanum 5.8S rRNA geni və ITS1 və ITS2 DNT
CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGATGAGACCGTGGAATAAACGATCGAGTG
AATCCGGAGGACCGGTGTACTCAGCTCACCGGGGGCATTGCTCCCGTGGTGACCCTGATTTGTTGTTGGG
...
```

O, 94 qeyddə n ibarə tdir, hə r birində “>” (simvoldan böyük) ilə başlayan və ardıcıllıqla gə lə n sə tir var. bir və ya bir neçə sə tirdə . İndi bunu Python-da sınayın:

```
>>> SeqIO.parse("ls_orchid.fasta",
"fasta") - da seq_record üçün SeqIO >>> Bio idxalı üçün :
...         çap (seq_record.id) çap
...         (repr(seq_record.seq)) çap (len(seq_record))
...
...
```

Ekranınızda belə bir şey almalısınız:

```
gi|2765658|emb|Z78533.1|CIZ78533
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGATGAGACCGTGG...CGC') 740

...
gi|2765564|emb|Z78439.1|PBZ78439
Seq('CATTGTTGAGATCACATAATAATTGATCGAGTTAATCTGGAGGATCTGTTTACT...GCC') 592
```

2.4.2 Sadə GenBank tə hlil nümunə si

İndi ls orchid.gbk GenBank faylını yüklə yə–k bunun ə və zinə - qeyd edin ki, bunu etmə k üçün kod yuxarıda FASTA faylı üçün istifadə edilə n parça ilə demə k olar ki, eynidir - yeganə fə rq biz fayl adını və format sə tirini də yişdiririk:

```
>>> SeqIO.parse ("ls_orchid.gbk",
"genbank") -da seq_rekord üçün Bio idxal SeqIO >>>- dan :
```

Machine Translated by Google

```
...         çap (seq_record.id) çap
...         (repr(seq_record.seq)) çap
...         (len(seq_record))
...
```

Bu vermə lidir:

```
Z78533.1
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGATGAGACCGTGG...CGC') 740

...
Z78439.1
Seq('CATTGTTGAGATCACATAATAATTGATCGAGTTAATCTGGAGGATCTGTTTACT...GCC') 592
```

Bu halda seq_record.id kimi daha qısa sə tirdə n istifadə olunduğunu görə cə ksiniz.

2.4.3 Mə n tə hlil etmə yi sevirə m – xahiş edirə m, bu barə də danışmağı dayandırmayın!

Biopython-un çoxlu analizatorları var və hə r birinin tə hlil etdiyi ardıcıllıq formatına və bütün bunlara ə saslanan öz kiçik xüsusi nişlə ri var. Fə sil 5 Bio.SeqIO-nu daha ə traflı ə hatə edir, Fə sil 6 isə ardıcıl düzülmə lə r üçün Bio.Align tə qdim edir.

Ə n populyar fayl formatlarında Bio.SeqIO və /və ya Bio.AlignIO-ya inteqrasiya olunmuş tə hlilediciləə r olsa da, bə zi nadir və sevilmə yə n fayl formatları üçün ya heç bir tə hlilçi yoxdur, ya da hə lə də ə laqə si olmayan köhnə analizator. Zə hmə t olmasa http://biopython.org/wiki/SeqIO viki sə hifə lə rini də yoxlayın və http://biopython.org/wiki/AlignIO ə n son mə lumat üçün və ya poçt siyahısından soruşun. Viki sə hifə lə rində də stə klə nə n fayl növlə rinin ə n son siyahısı və bə zi ə lavə nümunə lə r olmalıdır.

Xüsusi tə hlilçilə r və onlarla gözə l işlə rin necə edilə cə yi haqqında mə lumat axtarmaq üçün növbə ti yer Kulinariya Kitabındadır (bu Tə limatın 22-ci Fə sli). Ə gə r axtardığınız mə lumatı tapa bilmirsinizsə , lütfə n, hə ddə n artıq işlə nmiş kasıb sə nə dlə rinizə kömə k etmə yi və bu barə də yemə k kitabına yazı tə qdim etmə yi düşünün! (bunu necə edə cə yinizi anladıqdan sonra, yə ni!)

2.5 Bioloji mə lumat bazaları ilə ə laqə

Bioinformatikada etmə li olduğunuz çox ümumi işlə rdə n biri bioloji verilə nlə r bazalarından mə lumat çıxarmaqdır. Bu verilə nlə r bazalarına ə l ilə daxil olmaq olduqca yorucu ola bilə r, xüsusə n də çoxlu tə krarlanan işiniz varsa. Biopython bə zi on-line verilə nlə r bazalarını Python skriptlə rində n ə lçatan etmə klə sizə vaxt və enerjiyə qə naə t etmə yə çalışır. Hal-hazırda, Biopython aşağıdakı verilə nlə r bazalarından mə lumat çıxarmaq üçün kodlara malikdir:

- Entrez (və PubMed) NCBI-dan – 12-ci Fə slə baxın.
- ExPASy – 13-cü Fə slə baxın.
- SCOP – Bio.SCOP.search() funksiyasına baxın.

Bu modullardakı kod, ə sasə n, bu sə hifə lə rdə ki CGI skriptlə ri ilə qarşılıqlı ə laqə də olan Python kodunu yazmağı asanlaşdırır, belə liklə , formatla asan başa düşülə n nə ticə lə r ə ldə edə bilə rsiniz. Bə zi hallarda, mə lumatların çıxarılmasını daha da asanlaşdırmaq üçün nə ticə lə r Biopython tə hlilçilə ri ilə sıx şə kildə birlə şdirilə bilə r.

Machine Translated by Google

2.6 Bundan sonra nə etmə li

İndi bu günə qə də r nail olduğunuza görə , ümid edirə m ki, Biopython-un ə saslarını yaxşı başa düşmüsünüz və ondan faydalı iş görmə k üçün istifadə etmə yə hazırsınız. İndi edilə cə k ə n yaxşı şey bu tə limatı oxumağı bitirmə k və sonra mə nbə kodunu araşdırmağa başlamaq və avtomatik yaradılan sə nə dlə rə baxmaqdır.

Nə etmə k istə diyinizin və Biopython-da hansı kitabxanaların bunu edə cə yinə dair bir şə kil ə ldə etdikdə n sonra siz yemə k kitabında (Fə sil 22) zirvə yə qalxmalısınız ki, burada etmə k istə diyinizə bə nzə r bir şey etmə k üçün nümunə kodu ola bilə r.

Ə gə r siz nə etmə k istə diyinizi bilirsinizsə , lakin bunu necə edə cə yinizi anlaya bilmirsinizsə , lütfə n, ə sas Biopython siyahısına suallar göndə rin (bax: http://biopython.org/wiki/Mailing_lists). Bu, tə kcə sualınıza cavab vermə yə kömə k etmə yə cə k, hə m də sə nə dlə ri tə kmillə şdirmə yə imkan verə cə k ki, növbə ti şə xsə etmə k istə diyinizi etmə yə kömə k etsin.

Koddan hə zz alın!

Machine Translated by Google

Fə sil 3

Obyektlə rin ardıcıllığı

Bioloji ardıcıllıqlar, şübhə siz ki, Bioinformatikanın mə rkə zi obyektidir və bu fə sildə biz ardıcıllıqla işlə mə k üçün Biopython mexanizmini, Seq obyektini tə qdim edə cə yik. 4-cü fə sildə ardıcıllıq mə lumatını hə r hansı annotasiya ilə birlə şdirə n müvafiq SeqRecord obyekti tə qdim edilə cə k, bu da 5-ci Fə sildə Ardıcıllıqla Giriş/Çıxış üçün yenidə n istifadə olunur.

Ardıcıllıqlar ə slində AGTACACTGGT kimi hə rf sə tirlə ridir, bu çox tə bii görünür, çünki bu ardıcıllıqların bioloji fayl formatlarında görünmə sinin ə n ümumi yolu.

Seq obyektlə ri ilə standart Python sə tirlə ri arasındakı ə n mühüm fə rq onların müxtə lif üsullara malik olmasıdır. Seq obyekti düz sə tir kimi eyni metodların çoxunu də stə klə sə də , onun translate() metodu bioloji tə rcümə etmə klə fə rqlə nir və reverse_complement() kimi ə lavə bioloji cə hə tdə n uyğun üsullar da var.

3.1 Ardıcıllıqlar simlə r kimi fə aliyyə t göstə rir

Ə ksə r hallarda biz Seq obyektlə ri ilə normal Python sə tirlə ri kimi işlə yə bilə rik, mə sə lə n, uzunluğu ə ldə etmə k və ya elementlə r üzə rində tə krarlamaq:

```
>>> Bio.Seq import Seq >>> my_seq
= Seq("GATCG") >>> indeks üçün ,
hə rf sırala (my_seq): print("%i % s" % (indeks, mə ktub))
...
...
0 G
1 A
2 T
3 C
4 G
>>> çap (len(mə nim_seq))
5
```

Ardıcıllığın elementlə rinə sə tirlə rdə olduğu kimi daxil ola bilə rsiniz (ancaq unutmayın ki, Python sıfırdan hesablanır!):

```
>>> çap (my_seq[0]) # birinci hə rf
G
>>> çap (my_seq[2]) # üçüncü hə rf
T
>>> çap (my_seq[-1]) # son hə rf
G
```

Machine Translated by Google

Seq obyekti sə tir kimi .count() metoduna malikdir. Qeyd edə k ki, bu Python kimi demə kdir string, bu üst-üstə düşmə yə n say verir:

```
>>> Bio.Seq idxal ardıcıllığından >>>
"AAAA".count(" AA ")
2
>>> Seq("AAAA").count("AA") 2
```

Bə zi bioloji istifadə lə r üçün, hə qiqə tə n, üst-üstə düşə n sayı istə yə bilə rsiniz (yə ni bu ə hə miyyə tsiz nümunə də 3). Tə k hə rflə ri axtararkə n bunun heç bir fə rqi yoxdur:

```
>>> Bio.Seq -də n idxal Seq >>> my_seq =
Seq("GATCGATGGGCCTATATAGGATCGAAAATCGC") >>> len(my_seq) 32

>>> my_seq.count("G") 9

>>> 100 * (my_seq.count("G") + my_seq.count("C")) / len(my_seq) 46.875
```

GC% hesablamaq üçün yuxarıdakı kod parçasını istifadə edə bilsə niz də , Bio.SeqUtils modulunun artıq qurulmuş bir neçə GC funksiyasına malikdir. Mə sə lə n:

```
>>> Bio.Seq -də n idxal Seq >>>
Bio.SeqUtils -də n idxal gc_fraction >>> my_seq =
Seq("GATCGATGGGCCTATATAGGATCGAAAATCGC") >>> gc_fraction(my_seq)

0.46875
```

Nə zə rə alın ki, Bio.SeqUtils.gc_fraction() funksiyasından istifadə avtomatik olaraq qarışıq və ziyyə t ardıcıllığı və G və ya C mə nasını verə n qeyri-müə yyə n nukleotid S ilə öhdə sində n gə lmə lidir.

Onu da qeyd edə k ki, adi Python sə tri kimi Seq obyekti müə yyə n mə nada "yalnız oxumaq üçündür". Ardıcıllığınızı redaktə etmə k lazımdırsa, mə sə lə n, bir nöqtə mutasiyasını imitasiya etmə k lazımdırsa, aşağıda MutableSeq obyektində n bə hs edə n Bölmə 3.13- ə baxın.

3.2 Ardıcıllığın dilimlə nmə si

Daha mürə kkə b bir nümunə , ardıcıllığın bir hissə sini ə ldə edə k:

```
>>> Bio.Seq -də n idxal Seq >>> my_seq
= Seq("GATCGATGGGCCTATATAGGATCGAAAATCGC") >>> my_seq[4:12]

Seq('GATGGGCC')
```

Qeyd edə k ki, 'Seq' obyektlə ri Python sə tirlə ri üçün adi indekslə şdirmə konvensiyalarına ə mə l edir, ardıcıllığın ilk elementi 0 ilə nömrə lə nir. Siz dilim etdiyiniz zaman birinci element daxil edilir (yə ni bu halda 4), sonuncu isə xaric edilir (bu halda 12).

Hə mçinin Python sə tri kimi, siz başlanğıc, dayanma və addımla dilimlə r edə bilə rsiniz (defolt olaraq addım ölçüsü bir). Mə sə lə n, bu DNT ardıcıllığının birinci, ikinci və üçüncü kodon mövqelə rini ə ldə edə bilə rik:

Machine Translated by Google

```
>>> my_seq[0::3]
Seq('GCTGTAGTAAG') >>>
my_seq[1::3]
Seq('AGGCATGCATC') >>>
my_seq[2::3]
Seq('TAGCTAAGAC')
```

Python simli ilə gördüyünüz başqa bir addım hiylə si -1 addımdan istifadə etmə kdir. simli. Bunu Seq obyekti ilə də edə bilə rsiniz:

```
>>> my_seq[::-1]
Seq('CGCTAAAAGCTAGGATATATCCGGGTAGCTAG')
```

3.3 Seq obyektlə rinin simlə rə çevrilmə si

ə gə r hə qiqə tə n sadə bir sə tirə ehtiyacınız varsa, mə sə lə n, fayla yazmaq və ya verilə nlə r bazasına daxil etmə k üçün, onda bunu ə ldə etmə k çox asandır:

```
>>> str (my_seq)
 'GATCGATGGGCCTATATAGGATCGAAAATCGC'
```

Seq obyektində str() funksiyasının çağırılması sə tir kimi tam ardıcıllığı qaytardığından, çox vaxt sizdə yoxdur bu çevrilmə ni açıq şə kildə etmə k. Python bunu avtomatik olaraq çap funksiyasında edir:

```
>>> çap (mə nim_seq)
GATCGATGGGCCTATATAGGATCGAAAATCGC
```

Siz hə mçinin Python sə tirinin formatlaşdırılması və ya interpolyasiya operatorundan (%) istifadə edə rkə n Seq obyektini birbaşa %s yertutucusu ilə istifadə edə bilə rsiniz:

```
>>> fasta_format_string = ">Ad\n%s\n" % my_seq >>> çap
(fasta_format_string)
>Ad
GATCGATGGGCCTATATAGGATCGAAAATCGC
<BLANKLINE>
```

Bu kod sə tri sadə FASTA formatı qeydini qurur (sə tirlə rin sarılmasından narahat olmadan). Bölmə 4.6 SeqRecord obyektində n FASTA formatlı sə tir ə ldə etmə yin sə liqə li yolunu tə svir edir, halbuki FASTA formatında ardıcıl faylların oxunması və yazılması ilə bağlı daha ümumi mövzu 5-ci Fə sildə ə hatə olunur.

3.4 Ardıcıllıqların birlə şdirilmə si və ya ə lavə edilmə si

İki Seq obyekti onları ə lavə etmə klə birlə şdirilə bilə r:

```
>>> Bio.Seq import Seq >>> seq1 =
Seq("ACGT") >>> seq2 =
Seq("AACCGG") >>> seq1 + seq2
Seq('ACGTAACCGG' )
```

Biopython ardıcıllığın mə zmununu yoxlamır və mə sə lə n, zülal ardıcıllığını və DNT ardıcıllığını birlə şdirsə niz (bu, sə hv ola bilə r) istisna yaratmayacaq:

Machine Translated by Google

```
>>> Bio.Seq import Seq >>> protein_seq
= Seq( " EVRNAK") >>> dna_seq = Seq("ACGT")
>>> protein_seq + dna_seq
Seq('EVRNAKACGT')
```

Tez-tez bir-birinə ə lavə etmə k üçün bir çox ardıcıllığa sahib ola bilə rsiniz, bunu belə bir for döngə si ilə etmə k olar:

```
>>> Bio.Seq import Seq >>> list_of_seqs =
[Seq("ACGT"), Seq ("AACC"), Seq("GGTT")] >>> concatenated = Seq("") >>> üçün s_of_seqs : concatenated
+= s

...
...
>>> birlə şdirilir
Seq('ACGTAACCGGTT')
```

Python sə tirlə ri kimi, Biopython Seq də .join metoduna malikdir:

```
>>> Bio.Seq import Seq >>> contigs =
[Seq("ATG"), Seq( " ATCCCG"), Seq("TTGCA")] >>> spacer = Seq("N" * 10) >>>
spacer.join(contigs)

Seq('ATGNNNNNNNNNNATCCCCGNNNNNNNNNNTTGCA')
```

3.5 Korpusun də yişdirilmə si

Python sə tirlə rində işi də yişdirmə k üçün çox faydalı yuxarı və aşağı üsullar var. Mə sə lə n,

```
>>> Bio.Seq import Seq >>> dna_seq
= Seq("acgtACGT") >>> dna_seq
Seq('acgtACGT')
>>> dna_seq.upper( )

Seq('ACGTACGT')
>>> dna_seq.lower()
Seq('acgtacgt')
```

Bunlar böyük hə rflə rə hə ssas olmayan uyğunluq etmə k üçün faydalıdır:

```
>>> dna_seq -də "GTAC"
Yalan
>>> dna_seq.upper() -də "GTAC"
Doğrudur
```

3.6 Nukleotid ardıcıllığı və (ə ks) komplementlə r

Nukleotid ardıcıllığı üçün siz onun daxili metodlarından istifadə edə rə k Seq obyektinin tamamlayıcı və ya ə ks tamamlamasını asanlıqla ə ldə edə bilə rsiniz:

```
>>> Bio.Seq -də n idxal Seq >>>
my_seq = Seq("GATCGATGGGCCTATATAGGATCGAAAATCGC")
```

Machine Translated by Google

```
>>> my_seq
Seq('GATCGATGGGCCTATATAGGATCGAAAATCGC') >>>
my_seq.complement()
Seq('CTAGCTACCCGGATATATCCTAGCTTTTAGCG') >>>
my_seq.reverse_complement()
Seq('GCGATTTTCGATCCTATATAGGCCCATCGATC')
```

Daha ə vvə l qeyd edildiyi kimi, Seq obyektini (və ya Python sə tirini) tə rsinə çevirmə yin asan yolu onu -1 addım ilə kə smə kdir:

```
>>> my_seq[::-1]
Seq('CGCTAAAAGCTAGGATATATCCGGGTAGCTAG')
```

Tə sadüfə n zülal ardıcıllığının (ə ks) tamamlayıcısını qə bul etmə k kimi qə ribə bir şey etmə yə cə hd etsə niz, nə ticə lə r bioloji cə hə tdə n mə nasızdır:

```
>>> Bio.Seq idxal ardıcıllığından >>>
protein_seq = Seq ("EVRNAK") >>>
protein_seq.complement( )
Seq('EBYNTM')
```

Burada "E" hə rfi nukleotidlə r üçün etibarlı IUPAC qeyri-müə yyə nlik kodu deyil, ona görə də ə lavə edilmə mişdir. Bununla belə , "V" "A", "C" və ya "G" demə kdir və "B" və s. tamamlayır.

Bölmə 5.5.3- də ki nümunə Seq obyektinin ə ks tamamlama metodunu Bio.SeqIO ilə birlə şdirir. giriş/çıxış ardıcıllığı.

3.7 Transkripsiya

Transkripsiya haqqında danışmazdan ə vvə l mə n strand mə sə lə sini aydınlaşdırmağa çalışmaq istə yirə m. Qısa bir peptidi kodlayan ikiqat zə ncirli DNT-nin aşağıdakı (yaradılmış) uzanmasını nə zə rdə n keçirə k:

```
          DNT kodlaşdıran zə ncir (aka Crick strand, strand +1)
5'        ATGGCCATTGTAATGGGCCGCTGAAAGGGTGCCCGATAG          3'
          |||||||||||||||||||||||||||||||||||||||
3'        TACCGGTAACATTACCCGGCGACTTTCCCACGGGCTATC          5'
       DNT şablon zə ncir (aka Watson strand, strand     1)

                                  |
                             Transkripsiya

5'        AUGGCCAUUGUAAUGGGCCGCUGAAAGGGUGCCCGAUAG          3'
                     Tə k zə ncirli xə bə rçi RNT
```

Hə qiqi bioloji transkripsiya prosesi mRNT-ni vermə k üçün tə rs tamamlayıcı (TCAG CUGA) edə rə k şablon zə ncirində n işlə yir. Bununla belə , Biopython və ümumiyyə tlə bioinformatikada biz adə tə n birbaşa kodlaşdırma zolağı ilə işlə yirik, çünki bu o demə kdir ki, biz T U hə rfini də yişdirmə klə mRNT ardıcıllığını ə ldə edə bilə rik.

İndi gə lin ə slində Biopython-da transkripsiya etmə yə başlayaq. Ə vvə lcə üçün Seq obyektlə ri yaradaq kodlaşdırma və şablon DNT zə ncirlə ri:

Machine Translated by Google

```
>>> Bio.Seq -də n idxal Seq >>> coding_dna
= Seq("ATGGCCATTGTAATGGGCCGCTGAAAGGGTGCCCGATAG") >>> coding_dna

Seq('ATGGCCATTGTAATGGGCCGCTGAAAGGGTGCCCGATAG') >>>
şablon_dna = >> reverse . template_dna

Seq('CTATCGGGCACCCTTTCAGCGGCCCATTACAATGGCCAT')
```

Bunlar yuxarıdakı rə qə mə uyğun olmalıdır - konvensiya ilə yadda saxlayın ki, nukleotid ardıcıllığı adə tə n 5'-də n 3' istiqamə tində oxunur, şə kildə ki şablon zə ncirinin tə rsinə göstə rildiyi halda.

İndi daxili Seq obyektində n istifadə edə rə k kodlaşdırma zə ncirini müvafiq mRNT-yə köçürə k. transkripsiya üsulu:

```
>>> coding_dna
Seq('ATGGCCATTGTAATGGGCCGCTGAAAGGGTGCCCGATAG') >>>
messenger_rna = coding_dna.transcribe() >>> messenger_rna

Seq('AUGGCCAUUGUAAUGGGCCGCUGAAAAGGGUGCCGAUAG')
```

Gördüyünüz kimi, bütün bunlar T-ni U ilə ə və z etmə kdir.

Ə gə r siz şablon zə ncirində n başlayaraq ə sl bioloji transkripsiya etmə k istə yirsinizsə , o zaman bu olur. iki mə rhə lə li proses:

```
>>> template_dna.reverse_complement().transkripsiya()
Seq('AUGGCCAUUGUAAUGGGCCGCUGAAAGGGUGCCCGAUAG')
```

Seq obyekti hə mçinin mRNT-də n DNT-nin kodlaşdırıcı zə ncirinə keçmə k üçün geri transkripsiya metodunu ehtiva edir. Yenə də bu sadə U T ə və zetmə sidir:

```
>>> Bio.Seq -də n idxal Seq >>>
messenger_rna = Seq("AUGGCCAUUGUAAUGGGCCGCUGAAAGGGUGCCCGAUAG") >>> messenger_rna

Seq('AUGGCCAUUGUAAUGGGCCGCUGAAAGGGUGCCCGAUAG') >>> rna
messe
Seq('ATGGCCATTGTAATGGGCCGCTGAAAGGGTGCCCGATAG')
```

Qeyd: Seq obyektinin transkripsiya və back_transcribe üsulları Biopython 1.49-da ə lavə edilmişdir. üçün köhnə buraxılışlarda ə və zində Bio.Seq modulunun funksiyalarından istifadə etmə li olacaqsınız, Bölmə 3.15-ə baxın.

3.8 Tə rcümə

Yuxarıdakı transkripsiya bölmə sində müzakirə edilə n eyni nümunə yə sadiq qalaraq, indi bu mRNT-ni müvafiq zülal ardıcıllığına çevirə k - yenə Seq obyektinin bioloji üsullarından birini istifadə edə rə k:

```
>>> Bio.Seq -də n idxal Seq >>>
messenger_rna = Seq("AUGGCCAUUGUAAUGGGCCGCUGAAAGGGUGCCCGAUAG") >>> messenger_rna

Seq('AUGGCCAUUGUAAUGGGCCGCUGAAAGGGUGCCCGAUAG') >>>
rnamesse.
Seq('MAIVMGR*KGAR*')
```

Siz hə mçinin kodlaşdırma zə ncirinin DNT ardıcıllığından birbaşa tə rcümə edə bilə rsiniz:

Machine Translated by Google

```
>>> Bio.Seq -də n idxal Seq >>>
coding_dna = Seq("ATGGCCATTGTAATGGGCCGCTGAAAGGGTGCCCGATAG") >>> coding_dna

Seq('ATGGCCATTGTAATGGGCCGCTGAAAGGGTGCCCGATAG') >>>
coding_dna.translate
Seq('MAIVMGR*KGAR*')
```

Yuxarıdakı zülal ardıcıllıqlarında qeyd etmə lisiniz ki, son dayanma xarakteri ilə yanaşı, daxili dayanma da var. Bu, düşünülmüş nümunə seçimi idi, çünki o, müxtə lif tə rcümə cə dvə llə ri (Genetik Kodlar) daxil olmaqla, bə zi isteğe bağlı arqumentlə r haqqında danışmaq üçün bə hanə verir.

Biopython-da mövcud olan tə rcümə cə dvə llə ri NCBI-dan olanlara ə saslanır (bu tə limatın növbə ti hissə sinə baxın). Varsayılan olaraq, tə rcümə standart genetik koddan istifadə edə cə k (NCBI cə dvə l id 1). Tutaq ki, biz mitoxondrial ardıcıllıqla mə şğul oluruq. Bunun ə və zinə müvafiq genetik kodu istifadə etmə k üçün tə rcümə funksiyasına demə liyik:

```
>>> coding_dna.translate(cə dvə l="Onurğalı Mitoxondrial")
Seq('MAIVMGRWKGAR*')
```

Siz hə mçinin daha qısa olan və tez-tez cə dvə lə daxil edilə n NCBI cə dvə l nömrə sində n istifadə edə rə k cə dvə li tə yin edə bilə rsiniz GenBank fayllarının xüsusiyyə t annotasiyası:

```
>>> codeding_dna.translate(cə dvə l=2)
Seq('MAIVMGRWKGAR*')
```

İndi siz nukleotidlə ri çə rçivə nin dayanma kodonunda birinciyə çevirmə k və sonra dayandırmaq (tə biə tdə olduğu kimi) istə yə bilə rsiniz:

```
>>> codeding_dna.translate()
Seq('MAIVMGR*KGAR*') >>>
coding_dna.translate(to_stop=Doğru)
Seq('MAIVMGR')
>>> coding_dna.translate(cə dvə l=2)
Seq('MAIVMGRWKGAR*')
>>> coding_dna.translate(cə dvə l=2, to_stop=Doğru)
Seq('MAIVMGRWKGAR')
```

Diqqə t yetirin ki, siz to_stop arqumentində n istifadə etdiyiniz zaman stop kodonun özü tə rcümə olunmur - və stop simvolu zülal ardıcıllığının sonunda daxil edilmir.

Defolt ulduzu bə yə nmə sə niz, hə tta dayanma simvolunu da tə yin edə bilə rsiniz:

```
>>> codeding_dna.translate(cə dvə l=2, stop_simvolu="@")
Seq('MAIVMGRWKGAR@')
```

İndi fə rz edə k ki, sizdə tam kodlaşdırma ardıcıllığı CDS var, yə ni nukleotid ardıcıllığı (mə sə lə n, mRNA – hə r hansı bir birlə şmə də n sonra) kodonların tam sayı (yə ni uzunluq üçə çoxluqdur), başlanğıc kodonu ilə başlayan, dayanma kodonu ilə bitə n və daxili çə rçivə də dayanma kodonları yoxdur. Ümumiyyə tlə , tam CDS verildikdə , standart tə rcümə metodu istə diyinizi edə cə k (bə lkə də to_stop seçimi ilə). Bununla belə , ə gə r ardıcıllığınız qeyri-standart başlanğıc kodonu istifadə edirsə ? Bu, bakteriyalarda çox olur – mə sə lə n, E. coli K12-də yaaX geni:

```
>>> Bio.Seq -də n idxal Seq >>> gen =
Seq(
...         "GTGAAAAAAGATGCAATCTATCGTACTCGCACTTTCCCTGGTTCTGGTCGCTCCCATGGCA"
...         "GCACAGGCTGCGGAAATTACGTTAGTCCCGTCAGTAAAATTACAGATAGGCGATCGTGAT"
```

Machine Translated by Google

```
...         "AATCGTGGCTATTACTGGGATGGAGGTCACTGGCGCGACCACGGCTGGTGGAAACAACAT"
...         "TATGAATGGCGAGGCAATCGCTGGCACCTACGGACCGCCGCCACCGCCGCGCCACCAT"
...         "AAGAAAAGCTCCTCATGATCATCACGGCGGTCATGGTCCAGGCAAACATCACCGCTAA"
... )
>>> gen.translate(cə dvə l="Bakteriya")
Seq('VKKMQSIVLALSLVLVAPMAAQAAEITLVPSVKLQIGDRDNRGYYWDGGHWRDH...HR*',
ProteinAlpabet())
>>> gen.translate(cə dvə l="Bakteriya", to_stop=Doğru)
Seq('VKKMQSIVLALSLVLVAPMAAQAAEITLVPSVKLQIGDRDNRGYYWDGGHWRDH...HHR')
```

Bakterial genetik kodda GTG etibarlı başlanğıc kodonudur və normal olaraq istifadə edildiyi halda Valini kodlayır. başlanğıc kodonu metionin kimi tə rcümə edilmə lidir. Bu, Biopython-a ardıcıllığınızın a olduğunu söylə sə niz baş verir tam CDS:

```
>>> gen.translate(cə dvə l="Bakteriya", cds=Doğru)
Seq('MKKMQSIVLALSLVLVAPMAAQAAEITLVPSVKLQIGDRDNRGYYWDGGHWRDH...HHR')
```

Biopython-a alternativ başlanğıc kodonu metionin kimi tə rcümə etmə sini söylə mə klə yanaşı, bu seçimdə n istifadə edə rə k hə mçinin ardıcıllığınızın hə qiqə tə n etibarlı CDS olduğuna ə min olur (ə gə r deyilsə , istisna ə ldə edə cə ksiniz).

Bölmə 22.1.3- də ki nümunə Seq obyektinin tə rcümə metodunu ardıcıllıq üçün Bio.SeqIO ilə birlə şdirir. giriş/çıxış.

3.9 Tə rcümə Cə dvə llə ri

Ə vvə lki bölmə lə rdə biz Seq obyekt tə rcümə metodundan danışdıq (və ekvivalentini qeyd etdik Bio.Seq modulunda funksiya – Bölmə 3.15-ə baxın). Daxili olaraq bunlar koddan ə ldə edilə n cə dvə l obyektlə rində n istifadə edir NCBI mə lumatı ftp://ftp.ncbi.nlm.nih.gov/entrez/misc/data/gc.prt , https -də də göstə rilir : //www.ncbi.nlm.nih.gov/Taxonomy/Utils/wprintgc.cgi daha oxunaqlı tə rtibatda.

Ə vvə lki kimi, gə lin yalnız iki seçim üzə rində dayanaq: Standart tə rcümə cə dvə li və tə rcümə cə dvə li Onurğalı Mitoxondrial DNT.

```
>>> Bio.Data idxal CodonTable- də n
>>> standart_cə dvə l = CodonTable.birmə nalı_dna_by_name["Standart"]
>>> mito_table = CodonTable.unambiguous_dna_by_name["Onurğalı Mitoxondrial"]
```

Alternativ olaraq, bu cə dvə llə r müvafiq olaraq 1 və 2 ID nömrə lə ri ilə etiketlə nir:

```
>>> Bio.Data idxal CodonTable- də n
>>> standart_cə dvə l = CodonTable.birmə nalı_dna_by_id[1]
>>> mito_table = CodonTable.unambiguous_dna_by_id[2]
```

Hə qiqi cə dvə llə ri çap etmə klə vizual olaraq müqayisə edə bilə rsiniz:

```
>>> çap (standart_cə dvə l)
Cə dvə l 1 Standart, SGC0

     | T          | C          | A          | G          |
--+---------+---------+---------+---------+--
T | TTT F      | TCT S      | TAT Y | TGT C      | T
T | TTC F      | TCC S      | TAC Y | TGC C      | C
T | TTA L      | TCA S      | TAA Stop| TGA Stop| A
T | TTG L(lə r)| TCG S      | TAG Stop| TGG W | G
--+---------+---------+---------+---------+--
```

Machine Translated by Google

```
C | CTT L          | CCT P        | CAT H | CGT R | T
C | CTC L          | CCC P        | CAC H | CGC R | C
C | CTA L          | CCA P        | CAA Q        | CGA R | A
C | CTG L(lə r)| CCG P           | CAG Q        | CGG R | G
--+---------+---------+---------+---------+--
A | ATT I          | ACT T | AAT N | AGT S                    | T
A | ATC I          | ACC T | AAC N | AGC S                    | C
A | ATA I          | ACA T | AAA K | AGA R | A
A | ATG M(lə r)| ACG T | AAG K | AGG R | G
--+---------+---------+---------+---------+--
G | GTT V          | GCT A | GAT D | GGT G                    | T
G | GTC V          | GCC A | GAC D | GGC G                    | C
G | GTA V          | GCA A | GAA E | GGA G                    | A
G | GTG V          | GCG A | GAG E | GGG G                    | G
--+---------+---------+---------+---------+--
```

və :

```
>>> çap (mito_table)
Cə dvə l 2 Onurğalı Mitoxondrial, SGC1

    | T          | C          | A          | G          |
--+---------+---------+---------+---------+--
T | TTT F          | TCT S        | TAT Y | TGT C                | T
T | TTC F          | TCC S        | TAC Y | TGC C                | C
T | TTA L          | TCA S        | TAA Stop| TGA W | A
T | TTG L          | TCG S        | TAG Stop| TGG W | G
--+---------+---------+---------+---------+--
C | CTT L          | CCT P        | CAT H | CGT R | T
C | CTC L          | CCC P        | CAC H | CGC R | C
C | CTA L          | CCA P        | CAA Q        | CGA R | A
C | CTG L          | CCG P        | CAG Q        | CGG R | G
--+---------+---------+---------+---------+--
A | ATT I(lə r)| ACT T | AAT N | AGT S                    | T
A | ATC I(lə r)| ACC T | AAC N | AGC S                    | C
A | ATA M(s)| ACA T | AAA K | AGA Stop| A
A | ATG M(lə r)| ACG T | AAG K | AGG Stop| G
--+---------+---------+---------+---------+--
G | GTT V          | GCT A | GAT D | GGT G                    | T
G | GTC V          | GCC A | GAC D | GGC G                    | C
G | GTA V          | GCA A | GAA E | GGA G                    | A
G | GTG V(lə r)| GCG A | GAG E | GGG G                    | G
--+---------+---------+---------+---------+--
```

Aşağıdakı xüsusiyyə tlə ri faydalı tapa bilə rsiniz - mə sə lə n, öz gen tapıntınızı etmə yə çalışırsınızsa:

```
>>> mito_table.stop_codons
['TAA', 'TAG', 'AGA', 'AGG']
>>> mito_table.start_codons
['ATT', 'ATC', 'ATA', 'ATG', 'GTG']
>>> mito_table.forward_table["ACG"]
'T'
```

Machine Translated by Google

3.10 Seq obyektlə rinin müqayisə si

Ardıcıllığın müqayisə si ə slində çox mürə kkə b bir mövzudur və iki ardıcıllığın bə rabə r olub olmadığına qə rar vermə yin asan yolu yoxdur. Ə sas problem, ardıcıllıqdakı hə rflə rin mə nası kontekstdə n asılıdır - "A" hə rfi DNT, RNT və ya zülal ardıcıllığının bir hissə si ola bilə r. Biopython molekulun növünü izlə yə bilə r, buna görə də iki Seq obyektini müqayisə etmə k bunu da nə zə rə almaq demə k ola bilə r.

DNT fraqmenti "ACG" və RNT fraqmenti "ACG" bə rabə r olmalıdırmı? Bə s "ACG" peptid? Və ya Python sə tri "ACG"? Gündə lik istifadə də ardıcıllığınız ümumiyyə tlə eyni tipdə olacaq (bütün DNT, bütün RNT və ya bütün protein). Yaxşı, Biopython 1.65-də n etibarə n ardıcıllıq müqayisə si yalnız ardıcıllığa baxır və Python sə tri kimi müqayisə edir.

```
>>> Bio.Seq -də n idxal Seq >>> seq1 =
Seq("ACGT")
>>> "ACGT" == seq1 Doğru >>>
seq1
== "ACGT"
Doğrudur
```

Bunun uzantısı olaraq, Python lüğə tində ardıcıllıq obyektlə rinin açar kimi istifadə edilmə si ardıcıllığın açar üçün düz sə tir kimi istifadə edilmə sinə bə rabə rdir. Hə mçinin Bölmə 3.3-ə baxın .

3.11 Ardıcıllığın mə zmunu namə lum olan ardıcıllıqlar

Bə zi hallarda, ardıcıllığın uzunluğu mə lum ola bilə r, lakin onu tə şkil edə n faktiki hə rflə r deyil. Mə sə lə n, GenBank və EMBL faylları ardıcıl mə zmunu açıq şə kildə göstə rmə də n, yalnız konfiqurasiya mə lumatı ilə genomik DNT ardıcıllığını tə msil edə bilə r. Bu cür ardıcıllıqlar heç biri arqumenti və ardınca ardıcıllığın uzunluğu ilə Seq obyekti yaratmaqla tə msil oluna bilə r:

```
>>> Bio.Seq -də n idxal Seq >>>
unknown_seq = Seq(Yox, 10)
```

Belə liklə yaradılmış Seq obyekti də qiq müə yyə n edilmiş uzunluğa malikdir. Ardıcıllığın mə zmununa daxil olmaq üçün hə r hansı cə hd, lakin, UndefinedSequenceError-u qaldıracaq:

```
>>> unknown_seq
Seq(Heç biri, uzunluq=10) >>>
len(namə lum_seq) 10 >>> çap

(namə lum_seq)
Traceback (ə n son zə ng):
 ...
Bio.Seq.UndefinedSequenceError: Ardıcıllığın mə zmunu müə yyə n edilmə yib
>>>
```

3.12 Qismə n müə yyə n edilmiş ardıcıllıq mə zmunlu ardıcıllıqlar

Bə zə n ardıcıllığın mə zmunu yalnız ardıcıllığın hissə lə ri üçün müə yyə n edilir və başqa yerdə tə yin olunmur. Mə sə lə n, MAF (Multiple Alignment Format) faylının aşağıdakı fraqmenti insan, şimpanze, makaka, siçan, siçovul, it və opossum genomlarının ardıcıllığını göstə rir:

```
s hg38.chr7 117512683 36 + 159345973 TTGAAAACCTGAATGTGAGAGTCAGTCAAGGATAGT s panTro4.chr7 119000876 36 + 161824586
TTGAAAACCTTCGAATGTAGAGAGAGAGAG
```

Machine Translated by Google

s rheMac3.chr3 156330991 36 + 198365852 CTGAAATCCTGAATGTGAGAGTCAATCAAGGATGGT s mm10.chr6
18207101 36 + 149736546 CTGAAAATCCTAAGAGAG
s rn5.chr4 42326848 36 + 248343840 CTGAAAACCTAAGTAGGAGAGACAGTTAAAAGATAAT
s canFam3.chr14 56325207 36 + 60966679 TTGAAAAACTGATTATTAGAGGTCAATTAAGGATAGT
s monDom5.chr8 173163865 36 + 312544902 TTAAGAAACTGGAAATGAGGGTTGAATGACAAACTT

Hə r cə rgə də birinci rə qə m xromosomda düzülmüş ardıcıllığın başlanğıc mövqeyini (sıfır ə saslı koordinatlarda), sonra düzülmüş ardıcıllığın ölçüsünü, ipi, tam xromosomun ölçüsünü və düzülmüş ardıcıllığı göstə rir.

Belə qismə n müə yyə n edilmiş ardıcıllığı tə msil edə n Seq obyekti mə lumat arqumenti üçün lüğə tdə n istifadə etmə klə yaradıla bilə r, burada düymə lə r mə lum ardıcıllıq seqmentlə rinin başlanğıc koordinatlarıdır, də yə rlə r isə müvafiq ardıcıllığın mə zmunudur. Mə sə lə n, ilk ardıcıllıq üçün istifadə edə cə yik

```
>>> Bio.Seq -də n idxal Seq >>> seq =
Seq({117512683: "TTGAAAACCTGAATGTGAGAGTCAGTCAAGGATAGT"}, uzunluq=159345973)
```

Qismə n müə yyə n edilmiş ardıcıllıqdan alt ardıcıllığın çıxarılması tam müə yyə n edilmiş ardıcıllığı qaytara bilə r. koordinatlardan asılı olaraq cə rimə edilmiş ardıcıllıq və ya qismə n müə yyə n edilmiş ardıcıllıq:

```
>>> seq[1000:1020]
Seq(Yox, uzunluq=20) >>>
seq[117512690:117512700]
Seq('CCTGAATGTG') >>>
seq[117512670:117512690]
Seq({13: 'TTGAAAA'}, uzunluq=20) >>>
seq[117512700:]
Seq({0: 'AGAGTCAGTCAAGGATAGT'}, uzunluq=41833273)
```

Ardıcıllıqlardan ə n azı biri qismə n və ya tam müə yyə n edilmə dikdə , qismə n müə yyə n edilmiş ardıcıllıqlar ardıcıllıqların ə lavə edilmə si ilə də yaradıla bilə r:

```
>>> seq = Seq("ACGT") >>>
undefined_seq = Seq(Heç biri, uzunluq=10) >>> seq + qeyri-
müə yyə n_seq + seq
Seq({0: 'ACGT', 14: 'ACGT'}, uzunluq=18)
```

3.13 MutableSeq obyektlə ri

Normal Python sə tri kimi, Seq obyekti “yalnız oxumaqdır” və ya Python terminologiyasında də yişmə zdir.
Seq obyektinin sə tir kimi fə aliyyə t göstə rmə sini istə mə kdə n başqa, bu, hə m də faydalı defoltdur, çünki bir çox bioloji tə tbiqlə rdə ardıcıllıq mə lumatlarınızı də yişdirmə diyinizə ə min olmaq istə yirsiniz:

```
>>> Bio.Seq -də n idxal Seq >>> my_seq
= Seq("GCCATTGTAATGGGCCGCTGAAAGGGTGCCCGA")
```

Ardıcıllığı redaktə etmə yə çalışsanız nə baş verdiyini müşahidə edin:

```
>>> my_seq[5] = "G"
Traceback (ə n son zə ng):
 ...
TypeError: 'Seq' obyekti element tə yinatını də stə klə mir
```

Bununla belə , siz onu də yişə n ardıcıllığa (MutableSeq obyekti) çevirə və onunla istə diyiniz hə r şeyi edə bilə rsiniz:

Machine Translated by Google

```
>>> Bio.Seq -də n idxal MutableSeq >>> mutable_seq =
MutableSeq(my_seq) >>> mutable_seq

MutableSeq('GCCATTGTAATGGGCCGCTGAAAGGGTGCCCGA')
```

Alternativ olaraq, birbaşa sə tirdə n MutableSeq obyekti yarada bilə rsiniz:

```
>>> Bio.Seq -də n idxal MutableSeq >>> mutable_seq =
MutableSeq("GCCATTGTAATGGGCCGCTGAAAGGGTGCCCGA")
```

İstə nilə n şə kildə də yişdirilə bilə n ardıcıllıq obyekti verə cə kdir:

```
>>> mutable_seq
MutableSeq('GCCATTGTAATGGGCCGCTGAAAGGGTGCCCGA') >>> mutable_seq[5]
= "C" >>> mutable_seq MutableSeq

('GCCATCGTAATGGGCCGCTGAAAGGGTGCCCGA') >>> mutable_seq .


MutableSeq('GCCACGTAATGGGCCGCTGAAAGGGTGCCCGA') >>>
mutable_seq.reverse() >>> mutable_seq

MutableSeq('AGCCCGTGGGAAAGTCGCCGGGTAATGCACCG')
```

Qeyd edə k ki, MutableSeq obyektinin reverse() metodu, Python siyahısının reverse() metodu kimi, ardıcıllığı yerində tə rsinə çevirir.

Python-da də yişə n və də yişmə z obyektlə r arasındakı mühüm texniki fə rq o demə kdir ki, siz MutableSeq obyektini lüğə t açarı kimi istifadə edə bilmə zsiniz, lakin bu şə kildə Python sə tirində n və ya Seq obyektində n istifadə edə bilə rsiniz.

MutableSeq obyektinizi redaktə etmə yi başa vurduqdan sonra, yalnız oxuna bilə n Seq obyektinə qayıtmaq asandır:

```
>>> Bio.Seq -də n idxal Seq >>> new_seq =
Seq(də yişə n_seq) >>> new_seq

Seq('AGCCCGTGGGAAAGTCGCCGGGTAATGCACCG')
```

Siz hə mçinin Seq obyektində n olduğu kimi MutableSeq obyektində n də sə tir ə ldə edə bilə rsiniz (Bölmə 3.3).

3.14 Alt ardıcıllıqların tapılması

Ardıcıllıq obyektlə ri sadə sə tir obyektlə rində müvafiq metodlarla eyni funksiyanı yerinə yetirə n “find”, “rfind”, “index” və “rindex” metodlarına malikdir. Yeganə fə rq ondadır ki, alt ardıcıllıq sə tir, “bayt”, “bytearray”, “Seq” və ya “MutableSeq” obyekti ola bilə r:

```
>>> Bio.Seq import Seq, MutableSeq >>> seq = Seq
("GCCATTGTAATGGGCCGCTGAAAGGGTGCCCGA") >>> seq.index("ATGGGCCGC") 9


>>> seq.index(b"ATGGGCCGC") 9

>>> seq.index(bytearray(b"ATGGGCCGC")) 9
```

Machine Translated by Google

```
>>> seq.index(Seq("ATGGGCCGC")) 9 >>>

seq.index(MutableSeq("ATGGGCCGC"))
9
```

Ardıcıllıq tapılmadıqda "ValueError" yaranır:

```
>>> seq.index("ACTG") # doctest:+ELLIPSIS Traceback (ə n son zə ng):

 ...
ValueError: ...
```

“tap” metodu isə alt ardıcıllıq tapılmadıqda -1 qaytarır:

```
>>> seq.find("ACTG") -1
```

“rfind” və “rindex” üsulları sağ tə rə fdə n başlayaraq ardıcıllığı axtarır.
ardıcıllıq:

```
>>> seq.find("CC")
1
>>> seq.rfind("CC") 29
```

Eyni zamanda birdə n çox alt ardıcıllığı axtarmaq üçün “axtarış” metodundan istifadə edin. Bu üsul iteratoru qaytarır:

```
>>> indeks üçün , seq.search (["CC", "GGG", "CC"]):
...             çap (indeks, alt)
...
1 CC
11 GGG
14 CC
23 GGG
28 CC
29 CC
```

“Axtarış” metodu hə mçinin düz sə tirlə ri, 'bayt', 'baytarray', 'Seq' və 'MutableSeq' obyektlə rini alt ardıcıllıqlar kimi qə bul edir; eyni alt ardıcıllıqlar yuxarıdakı nümunə də olduğu kimi yalnız bir də fə bildirilir.

3.15 Birbaşa simlə rlə işlə mə k

Bu fə sli bağlamaq üçün, ardıcıllıq obyektlə rində n hə qiqə tə n istifadə etmə k istə mə yə nlə r (və ya funksional proqramlaşdırma tə rzini obyekt yönümlü olana üstünlük verə nlə r) üçün Bio.Seq-də modul sə viyyə li funksiyalar var ki, sadə Python sə tirlə rini, Seq obyektlə rini və ya MutableSeq obyektlə rini qə bul edə cə k:

```
>>> Bio.Seq -də n idxal reverse_complement, transkripsiya, back_transcribe, translate >>> my_string =
"GCTGTTATGGGTCGTTGGAAGGGTGGTCGTGCTGCTGGTTAG" >>> reverse_complement(my_string)

 'CTAACCAGCAGCACGACCACCCTTCCAACGACCCATAACAGC'
>>> transkripsiya (mə nim_sə tirim)
 'GCUGUUAUGGGUCGUUGGAAGGGUGGUCGUGCUGCUGGUUAG' >>>
geri_transkripsiya (mə nim_sim)
```

Machine Translated by Google

```
'GCTGTTATGGGTCGTTGGAAGGGTGGTCGTGCTGCTGGTTAG' >>> tə rcümə
et(my_string)
'AVMGRWKGGRAAG*'
```

Bununla belə , sizə standart olaraq Seq obyektlə ri ilə işlə mə k tövsiyə olunur.

Machine Translated by Google

4-cü fə sil

Annotasiya obyektlə rinin ardıcıllığı

3-cü fə sil ardıcıllıq siniflə rini tə qdim etdi. Seq sinfinin də rhal "yuxarısında" Bio.SeqRecord modulunda müə yyə n edilmiş Sequence Record və ya SeqRecord sinfidir. Bu sinif identifikatorlar və funksiyalar (SeqFeature obyektlə ri kimi) kimi daha yüksə k sə viyyə li xüsusiyyə tlə rin ardıcıllıqla ə laqə lə ndirilmə sinə imkan verir və 5-ci fə sildə tam tə svir olunan Bio.SeqIO giriş/çıxış interfeysi ardıcıllığında istifadə olunur.

Ə gə r siz yalnız FASTA faylları kimi sadə verilə nlə rlə işlə yə cə ksinizsə , yə qin ki, bu fə sli hə lə lik keçə bilə rsiniz. Digə r tə rə fdə n, mə sə lə n, GenBank və ya EMBL fayllarından zə ngin şə rhli ardıcıllıq mə lumatlarından istifadə edə cə ksinizsə , bu mə lumat olduqca vacibdir.

Bu fə sil bu fə sildə SeqRecord və SeqFeature obyektlə ri ilə ə laqə li ə ksə r işlə ri ə hatə etmə li olsa da, siz hə mçinin SeqRecord viki sə hifə sini (http://biopython.org/wiki/SeqRecord) oxumaq istə yə bilə rsiniz. və daxili sə nə dlə r (hə mçinin onlayn – SeqRecord və SeqFeature):

```
>>> Bio.SeqRecord -dan SeqRecord idxalı >>> kömə k
(SeqRecord)
```

4.1 SeqRecord obyekti

SeqRecord (Sequence Record) sinfi Bio.SeqRecord modulunda müə yyə n edilmişdir. Bu sinif identifikatorlar və funksiyalar kimi daha yüksə k sə viyyə li xüsusiyyə tlə rin ardıcıllıqla ə laqə lə ndirilmə sinə imkan verir (Fə sil 3-ə baxın) və Bio.SeqIO sequence giriş/çıxış interfeysi üçün ə sas mə lumat növüdür (bax. Fə sil 5).

SeqRecord sinfinin özü olduqca sadə dir və atributlar kimi aşağıdakı mə lumatları tə qdim edir:

.seq – Ardıcıllığın özü, adə tə n Seq obyekti.

.id – Ardıcıllığı müə yyə n etmə k üçün istifadə olunan ə sas ID – sə tir. Ə ksə r hallarda bu kimi bir şeydir qoşulma nömrə si.

.name – Ardıcıllıq üçün "ümumi" ad/id – sə tir. Bə zi hallarda bu, qoşulma nömrə si ilə eyni olacaq, lakin bu, klon adı da ola bilə r. Mə n bunu GenBank qeydində ki LOCUS id-nin analoqu hesab edirə m.

.description – İnsan tə rə fində n oxuna bilə n tə svir və ya ardıcıllığın ifadə li adı – sə tir.

.letter annotations – Ardıcıllıqdakı hə rflə r haqqında ə lavə mə lumatın (mə hdudlaşdırılmış) lüğə tində n istifadə edə rə k hə r hə rf annotasiyasını saxlayır. Düymə lə r mə lumatın adıdır və mə lumat ardıcıllığın özü ilə eyni uzunluqda Python ardıcıllığı (yə ni siyahı, tuple və ya sə tir) kimi də yə rdə yer alır. Bu, tez-tez keyfiyyə t balları (mə sə lə n, Bölmə 22.1.6) və ya ikinci də rə cə li struktur mə lumatı (mə sə lə n, Stokholm/PFAM uyğunlaşdırma fayllarından) üçün istifadə olunur .

Machine Translated by Google

.annotations – Ardıcıllıqla bağlı ə lavə mə lumatların lüğə ti. Düymə lə r mə lumatın adıdır və mə lumat də yə rin tə rkibində dir. Bu, ardıcıllığa daha çox "strukturlaşdırılmamış" mə lumat ə lavə etmə yə imkan verir.

.features – Ardıcıllıqdakı xüsusiyyə tlə r haqqında daha strukturlaşdırılmış mə lumatı olan SeqFeature obyektlə rinin siyahısı (mə sə lə n, genomdakı genlə rin mövqeyi və ya zülal ardıcıllığında domenlə r). Ardıcıllıq xüsusiyyə tlə rinin strukturu aşağıda Bölmə 4.3-də tə svir edilmişdir.

.dbxrefs - Sə tirlə r kimi verilə nlə r bazası çarpaz istinadlarının siyahısı.

4.2 SeqRecord-un yaradılması

SeqRecord obyektində n istifadə o qə də r də mürə kkə b deyil, çünki bütün mə lumatlar sinfin atributları kimi tə qdim olunur. Adə tə n siz SeqRecord-u "ə llə " yaratmayacaqsınız, ə və zində sizin üçün ardıcıl faylı oxumaq üçün Bio.SeqIO-dan istifadə edin (Fə sil 5 və aşağıdakı nümunə lə rə baxın). Bununla belə , SeqRecord yaratmaq olduqca sadə ola bilə r.

4.2.1 SeqRecord obyektlə rini sıfırdan

Minimum bir SeqRecord yaratmaq üçün sadə cə Seq obyektinə ehtiyacınız var:

```
>>> Bio.Seq import Seq >>> simple_seq =
Seq( " GATC") >>> from Bio.SeqRecord
import SeqRecord >>> simple_seq_r = SeqRecord(simple_seq)
```

Bundan ə lavə , siz hə mçinin id, ad və tə sviri başlanğıc funksiyasına ötürə bilə rsiniz, lakin ə gə r onlar deyilsə namə lum olduqlarını göstə rə n sə tirlə r kimi tə yin olunacaq və sonradan də yişdirilə bilə r:

```
>>> simple_seq_r.id '<namə lum
 id>'
>>> simple_seq_r.id = "AC12345" >>>
simple_seq_r.description = "Uydurma ardıcıllıq, kaş ki, haqqında bir yazı yaza bilə ydim" >>> çap (simple_seq_r.description)

Uydurma ardıcıllığı >>> simple_seq_r.seq Seq('GATC') haqqında mə qalə
yazmaq istə rdim
```

SeqRecord-unuzu fayla çıxarmaq istə yirsinizsə , identifikatorun daxil edilmə si çox vacibdir. edə rdin obyekt yaratarkə n adə tə n bunu daxil edin:

```
>>> Bio.Seq -də n idxal Seq >>> simple_seq
= Seq("GATC") >>> Bio.SeqRecord - dan
idxal SeqRecord >>> simple_seq_r = SeqRecord(simple_seq,
id="AC12345")
```

Yuxarıda qeyd edildiyi kimi, SeqRecord lüğə t atribut annotasiyalarına malikdir. Bu, digə r daha spesifik atributlardan birinə uyğun gə lmə yə n hə r hansı müxtə lif annotasiyalar üçün istifadə olunur. Annotasiyalar ə lavə etmə k asandır və sadə cə annotasiya lüğə ti ilə birbaşa mə şğul olmağı nə zə rdə tutur:

```
>>> simple_seq_r.annotations["delil"] = "Heç biri. Mə n sadə cə uydurdum." >>> çap (simple_seq_r.annotations)
{'sübut': 'Heç biri. Mə n sadə cə uydurdum.'} >>> çap
(simple_seq_r.annotations["sübut"])

Heç biri. Mə n sadə cə uydurdum.
```

Machine Translated by Google

Hə r hə rf üzrə annotasiyalarla işlə mə k oxşardır, letter_annotations ardıcıllıqla eyni uzunluğa malik istə nilə n Python ardıcıllığını (yə ni, sə tir, siyahı və ya də st) tə yin etmə yə imkan verə n lüğə t kimi atributdur:

```
>>> simple_seq_r.letter_annotations["phred_quality"] = [40, 40, 38, 30] >>>
print(simple_seq_r.letter_annotations) {'phred_quality': [40,
40, 38, 30]} >>> print(simple_seq_r.keyfiyyə ti
["phred_keyfiyyə ti"] [40, 40, 38, 30] >>> çap (simple_seq_r.letter_annotations) 40,
 38, 30]
```

dbxrefs və xüsusiyyə tlə rin atributları sadə cə Python siyahılarıdır və sə tirlə ri saxlamaq üçün istifadə edilmə lidir. SeqFeature obyektlə ri (bu fə sildə daha sonra müzakirə olunacaq).

4.2.2 FASTA fayllarından SeqRecord obyektlə ri

Bu nümunə Yersinia pestis biovar Microtus küç. üçün bütün ardıcıllığı ehtiva edə n kifayə t qə də r böyük FASTA faylından istifadə edir. 91001 plazmid pPCP1, ə vvə lcə NCBI-də n endirilib. Bu fayl GenBank qovluğu və ya onlayn NC 005816.fna altında Biopython vahid testlə rinə daxil edilmişdir. saytımızdan.

Fayl belə başlayır - və siz yalnız bir qeydin mövcud olduğunu yoxlaya bilə rsiniz (yə ni simvoldan böyük olan yalnız bir sə tir):

```
>gi|45478711|ref|NC_005816.1| Yersinia pestis biovar Microtus ... pPCP1, tam ardıcıllıq
TGTAACGAACGGTGCAATAGTGATCCACACCCAACGCCTGAAATCAGATCCAGGGGTAATCTGCTCTCC
...
```

2-ci Fə sildə siz SeqRecord obyektlə ri kimi fayldakı bütün qeydlə r üzə rində dövr etmə k üçün istifadə edilə n Bio.SeqIO.parse(...) funksiyasını görə cə ksiniz. Bio.SeqIO modulunun burada istifadə edə cə yimiz yalnız bir qeyddə n ibarə t fayllarda istifadə üçün bacı funksiyası var (ə traflı mə lumat üçün Fə sil 5-ə baxın):

```
>>> Bio import SeqIO- dan >>> rekord =
SeqIO.read("NC_005816.fna", "fasta") >>> rekord

SeqRecord(seq=Seq('TGTAACGAACGGTGCAATAGTGATCCACACCCAACGCCTGAAATCAGATCCAGG...CTG'), id='gi|
45478711|ref|NC_005816.1|', ad='gi|45478711|ref|05'|NC10. description='gi|45478711|ref|
NC_005816.1| Yersinia pestis biovar Microtus küç 91001 plazmid pPCP1, tam ardıcıllıq', dbxrefs=[])
```

İndi gə lin bu SeqRecord-un ə sas atributlarına ayrı-ayrılıqda nə zə r salaq – sizə Seq obyekti verə n seq atributundan başlayaraq:

```
>>> rekord.seq
Seq('TGTAACGAACGGTGCAATAGTGATCCACACCCAACGCCTGAAATCAGATCCAGG...CTG')
```

Sonra, identifikatorlar və tə svirlə r:

```
>>> rekord.id
 'gi|45478711|ref|NC_005816.1|' >>> qeyd.adı

 'gi|45478711|ref|NC_005816.1|' >>>
rekord.description 'gi|45478711|
 ref|NC_005816.1| Yersinia pestis biovar Microtus küç. 91001 plazmid pPCP1, tam ardıcıllıq'
```

Yuxarıda gördüyünüz kimi, FASTA qeydinin başlıq xə ttinin ilk sözü (böyük simvolu sildikdə n sonra) hə m id, hə m də ad atributları üçün istifadə olunur. Yazının tə sviri üçün bütün başlıq xə tti (böyük simvolu sildikdə n sonra) istifadə olunur. Bu, qismə n geriyə uyğunluq sə bə bi ilə qə sdə n edilir, lakin belə bir FASTA faylınız varsa, bu da mə ntiqlidir:

Machine Translated by Google

```
>Yersinia pestis biovar Microtus küç. 91001 plazmid pPCP1
TGTAACGAACGGTGCAATAGTGATCCACACCCAACGCCTGAAATCAGATCCAGGGGGTAATCTGCTCTCC
...
```

Qeyd edə k ki, FASTA faylını oxuyarkə n digə r annotasiya atributlarının heç biri doldurulmur:

```
>>> rekord.dbxrefs
[]
>>> qeyd.annotasiyalar
{}
>>> rekord.letter_annotations {}

>>> qeyd.xüsusiyyə tlə ri
[]
```

Bu halda bizim nümunə FASTA faylımız NCBI-dan idi və onların FASTA xə tlə rini formatlaşdırmaq üçün kifayə t qə də r yaxşı müə yyə n edilmiş konvensiyalar də sti var. Bu o demə kdir ki, bu mə lumatı tə hlil etmə k və mə sə lə n, GI nömrə sini və qoşulmanı çıxarmaq mümkün olacaq. Bununla belə , digə r mə nbə lə rdə n olan FASTA faylları fə rqlidir, ona görə də bu, ümumiyyə tlə mümkün deyil.

4.2.3 GenBank fayllarından SeqRecord obyektlə ri

Ə vvə lki nümunə də olduğu kimi, biz Yersinia pestis biovar Microtus küç. üçün bütün ardıcıllığa baxacağıq. 91001 plazmid pPCP1, ə vvə lcə NCBI-də n endirilib, lakin bu də fə GenBank faylı kimi. Yenə də bu fayl GenBank qovluğu və ya onlayn NC 005816.gb altında Biopython vahid testlə rinə daxil edilmişdir. saytımızdan.

Bu fayl tə k qeyddə n ibarə tdir (yə ni yalnız bir LOCUS xə tti) və başlayır:

```
LOCUS NC_005816 9609 bp TƏ YİF Yersinia pestis biovar Microtus küç. 91001 plazmid DNT dairə vi BCT 21-JUL-2008
pPCP1, tamamlandı
                    ardıcıllıq.
ACCESSION NC_005816 VERSİYA
LAYİHƏ Sİ           NC_005816.1 GI:45478711
                    GenomeProject:10638
...
```

Yenə də bu faylı oxumaq üçün Bio.SeqIO-dan istifadə edə cə yik və kod yuxarıda istifadə olunan kodla demə k olar ki, eynidir FASTA faylı üçün (ə traflı mə lumat üçün Fə sil 5-ə baxın):

```
>>> Bio import SeqIO- dan >>> rekord =
SeqIO.read("NC_005816.gb", "genbank") >>> rekord

SeqRecord(seq=Seq('TGTAACGAACGGTGCAATAGTGATCCACACCCAACGCCTGAAATCAGATCCAGG...CTG'), id='NC_005816.1', ad='NC_005816',
description='Yersinia pestis biovar Microtus küç., 910-ci seqment 9100 dbxrefs=['Layihə :58037'])


>>> rekord.seq
Seq('TGTAACGAACGGTGCAATAGTGATCCACACCCAACGCCTGAAATCAGATCCAGG...CTG')
```

Ad LOCUS xə ttində n gə lir, id isə versiya şə kilçisini ehtiva edir. Tə svir gə lir DEFINITION xə ttində n:

Machine Translated by Google

```
>>> rekord.id
 'NC_005816.1'
>>> qeyd.adı
 'NC_005816' >>>
rekord.description 'Yersinia pestis
 biovar Microtus küç. 91001 plazmid pPCP1, tam ardıcıllıq'
```

GenBank fayllarında hə r hə rf üçün annotasiya yoxdur:

```
>>> rekord.letter_annotations {}
```

Annotasiya mə lumatlarının ə ksə riyyə ti annotasiyalar lüğə tində qeyd olunur, mə sə lə n:

```
>>> len(record.annotations) 13 >>>

record.annotations["mə nbə "]
 'Yersinia pestis biovar Microtus küç. 91001'
```

dbxrefs siyahısı istə nilə n PROJECT və ya DBLINK sə tirində n doldurulur:

```
>>> rekord.dbxrefs
 ['Layihə :58037']
```

Nə hayə t, və bə lkə də ə n maraqlısı, xüsusiyyə tlə r cə dvə lində ki bütün qeydlə r (mə sə lə n, genlə r və ya CDS xüsusiyyə tlə r) xüsusiyyə tlə r siyahısında SeqFeature obyektlə ri kimi qeyd olunsun.

```
>>> len(record.features)
41
```

SeqFeature obyektlə ri haqqında növbə ti, Bölmə 4.3-də danışacağıq .

4.3 Obyektlə rin xüsusiyyə ti, yeri və mövqeyi

4.3.1 SeqFeature obyektlə ri

Ardıcıllıq xüsusiyyə tlə ri ardıcıllığın tə svirinin vacib hissə sidir. Ardıcıllığın özündə n kə nara çıxdıqdan sonra ardıcıllıqla bağlı mə lum olan daha "mücə rrə d" mə lumatı tə şkil etmə k və asanlıqla ə ldə etmə k üçün bir yol lazımdır. Hə r şeyi ə hatə edə cə k ümumi ardıcıllıq xüsusiyyə tlə ri sinfini inkişaf etdirmə k yə qin ki, qeyri-mümkün olsa da, Biopython SeqFeature sinfi ardıcıllıqla bağlı mə lumatı mümkün qə də r ə hatə etmə yə çalışır. Dizayn ə sasə n GenBank/EMBL xüsusiyyə t cə dvə llə rinə ə saslanır, ona görə də onların necə göründüyünü başa düşsə niz, Biopython siniflə rinin strukturunu daha asan başa düşə cə ksiniz.

Hə r bir SeqFeature obyekti haqqında ə sas fikir ana ardıcıllıqda, adə tə n SeqRecord obyektində bölgə ni tə svir etmə kdir. Hə min bölgə yer obyekti ilə tə svir edilir, adə tə n iki mövqe arasındakı diapazon (aşağıda Bölmə 4.3.2- yə baxın).

SeqFeature sinfinin bir sıra atributları var, ona görə də ə vvə lcə onları və onların ümumi xüsusiyyə tlə rini sadalayacağıq, daha sonra fə sildə bunun real hə yat nümunə sinə necə aid olduğunu göstə rmə k üçün nümunə lə r üzə rində işlə yə cə yik. SeqFeature atributları bunlardır:

.type – Bu funksiya növünün mə tn tə sviridir (mə sə lə n, bu "CDS" kimi bir şey olacaq və ya 'gen').

.location – SeqFeature-in mə şğul olduğunuz ardıcıllıqla yeri, aşağıda Bölmə 4.3.2- yə baxın . SeqFeature funksionallığının böyük hissə sini mə kan obyektinə hə valə edir və mə kanın xassə lə ri üçün bir sıra qısa yol atributlarını ehtiva edir:

Machine Translated by Google

.ref – .location.ref üçün stenoqram – yerin istinad etdiyi hə r hansı (fə rqli) istinad ardıcıllığı.
Adə tə n sadə cə Yoxdur.

.ref db – .location.ref_db üçün stenoqrafiya – .ref-də ki hə r hansı identifikatorun istinad etdiyi verilə nlə r bazasını tə yin edir.
Adə tə n sadə cə Yoxdur.

.strand – .location.strand üçün stenoqram – xüsusiyyə tin yerlə şdiyi ardıcıllıq üzrə ip. İkiqat zə ncirli nukleotid ardıcıllığı üçün bu, yuxarı zə ncir üçün 1, alt zə ncir üçün 1, zə ncir vacibdir, lakin namə lumdursa 0 və ya ə hə miyyə ti yoxdursa, heç biri ola bilə r. Bu, zülallar və ya tə k zə ncirli ardıcıllıqlar üçün heç biri deyil.

.qualifiers – Bu xüsusiyyə t haqqında ə lavə mə lumatların Python lüğə tidir. Ə sas də yə rdə olan mə lumatın nə haqqında olduğunu və də yə r faktiki mə lumatın bir növ qısa bir sözlə tə sviridir. Mə sə lə n, seçici üçün ümumi açar "də lil" və də yə r "hesablama (eksperimental olmayan)" ola bilə r. Bu, sadə cə olaraq xüsusiyyə tə baxan şə xsə onun eksperimental olaraq (yə ni yaş laboratoriyada) tə sdiqlə nmə diyini bildirmə yin bir yoludur. Qeyd edə k ki, digə r də yə r sə tirlə rin siyahısı olacaq (yalnız bir sə tir olduqda belə). Bu, GenBank/EMBL fayllarında xüsusiyyə t cə dvə llə rinin ə ksidir.

.sub funksiyalar – Bu, Gen-Bank/EMBL fayllarında "qoşulmalar" kimi mürə kkə b yerlə ri olan xüsusiyyə tlə ri tə msil etmə k üçün istifadə olunurdu. Bu, CompoundLocation obyektinin tə tbiqi ilə köhnə lmişdir və indi nə zə rə alınmamalıdır.

4.3.2 Və zifə lə r və yerlə r

Hə r bir SeqFeature obyekti haqqında ə sas fikir, adə tə n iki mövqe arasındakı diapazonu tə svir edə n bir yer obyektində n istifadə etdiyimiz ana ardıcıllıqda bölgə ni tə svir etmə kdir. İkisi istifadə etdiyimiz terminologiyanı aydınlaşdırmağa çalışır:

mövqe – Bu, qeyri-sə lis və ya qeyri-sə lis ola bilə n ardıcıllıqla tə k mövqeyə aiddir. Mə sə lə n, 5, 20, <100 və >200 bütün mövqelə rdir.

yer – Mə kan bə zi mövqelə rlə mə hdudlaşan ardıcıllığın bölgə sidir. Mə sə lə n, 5..20 (yə ni 5-də n 20-yə qə də r). bir yer.

Sadə cə bunu qeyd edirə m, çünki bə zə n ikisi arasında çaşıb qalıram.

4.3.2.1 SimpleLocation obyekti

Eukaryotik genlə rlə işlə mə sə niz, ə ksə r SeqFeature yerlə ri son də rə cə sadə dir - sizə sadə cə başlanğıc və son koordinatları və bir ip lazımdır. Bu, ə slində SimpleLocation obyektinin etdiyi bütün ə sas şeylə rdir.

Tə crübə də , ə lbə ttə ki, hə r şey daha mürə kkə b ola bilə r. Hə r şeydə n ə vvə l mürə kkə b yerlə ri idarə etmə liyik bir neçə rayondan ibarə tdir. İkincisi, mövqelə rin özlə ri qeyri-sə lis (qeyri-də qiq) ola bilə r.

4.3.2.2 CompoundLocation obyekti

Biopython 1.62, bir neçə bölgə də n ibarə t mürə kkə b yerlə rin necə tə msil olunduğunun yenidə n qurulmasının bir hissə si kimi CompoundLocation-u tə qdim etdi. Ə sas istifadə EMBL/GenBank fayllarında 'qoşulma' yerlə rini idarə etmə k üçündür.

4.3.2.3 Qeyri-sə lis mövqelə r

İndiyə qə də r biz yalnız sadə mövqelə rdə n istifadə etmişik. Xüsusiyyə t yerlə ri ilə mə şğul olmaqda bir mürə kkə blik mövqelə rin özlə rində olur. Biologiyada bir çox hallarda hə r şey tam müə yyə n deyil (bizim nə m laboratoriya bioloqları onları ə min etmə yə çalışdığımız qə də r!). Mə sə lə n, bir dinukleotid hazırlama tə crübə si edə və bunu kə şf edə bilə rsiniz

Machine Translated by Google

mRNT transkriptinin başlanğıcı iki yerdə n birində başlayır. Bu, çox faydalı mə lumatdır, lakin çə tinliyi bunu bir mövqe kimi necə tə qdim etmə kdir. Bununla mə şğul olmağımıza kömə k etmə k üçün qeyri-sə lis mövqelə r konsepsiyasına sahibik. Ə sasə n qeyri-sə lis mövqelə rin bir neçə növü var, ona görə də onlarla mə şğul olan beş sinifimiz var:

Də qiqMövqe – Adından da göründüyü kimi, bu sinif ardıcıllıqla də qiq olaraq tə yin olunan mövqeyi tə msil edir. Bu, sadə cə rə qə m kimi tə qdim olunur və siz obyektin mövqe atributuna baxaraq mövqeyi ə ldə edə bilə rsiniz.

BeforePosition – Bu sinif müə yyə n edilmiş bə zi saytdan ə vvə l baş verə n qeyri-sə lis mövqeyi tə msil edir. Gen-Bank/EMBL notasiyasında bu, `<13' kimi bir şey kimi tə msil olunur və bu, real mövqenin 13-də n kiçik yerdə yerlə şdiyini bildirir. Göstə rilə n yuxarı sə rhə di ə ldə etmə k üçün obyektin mövqe atributuna baxın.

AfterPosition – BeforePosition-dan fə rqli olaraq, bu sinif müə yyə n edilmiş saytdan sonra baş verə n mövqeyi tə msil edir. Bu, GenBank-da `>13' şə klində tə msil olunur və BeforePosition kimi, siz obyektin mövqe atributuna baxaraq sə rhə d nömrə sini ə ldə edirsiniz.

Mövqe daxilində – Bə zə n GenBank/EMBL yerlə ri üçün istifadə edilə n bu sinif, müə yyə n edilmiş iki nukleotid arasında bir yerdə baş verə n mövqeyi modellə şdirir. GenBank/EMBL notasiyasında bu, mövqeyin 1-də n 5-ə qə də r diapazonda olduğunu göstə rmə k üçün '(1.5)' kimi göstə rilmə lidir.

OneOfPosition – Bə zə n GenBank/EMBL yerlə ri üçün istifadə edilə n bu sinif bir neçə mümkün də yə rin mövcud olduğu mövqe ilə mə şğul olur, mə sə lə n, başlanğıc kodonu aydın deyilsə və genin başlaması üçün iki namizə din olduğu halda bundan istifadə edə bilə rsiniz. Alternativ olaraq, bu, iki ə laqə li gen xüsusiyyə ti kimi açıq şə kildə idarə oluna bilə r.

UnknownPosition – Bu sinif namə lum yerin mövqeyi ilə mə şğul olur. Bu Gen-da istifadə edilmir Bank/EMBL, lakin '?' UniProt-da istifadə olunan xüsusiyyə t koordinatı.

Budur, qeyri-sə lis son nöqtə lə ri olan bir yer yaratdığımız bir nümunə :

```
>>> Bio idxaldan SeqFeature >>> start_pos
= SeqFeature.AfterPosition(5) >>> end_pos =
SeqFeature.BetweenPosition(9, sol=8, sağ=9) >>> my_location =
SeqFeature.SimpleLocation(start_pos, end_pos )
```

Nə zə rə alın ki, Biopython 1.59-da də yişdirilmiş bə zi qeyri-sə lis yerlə rin tə fə rrüatları, xüsusə n Mövqe Arası və Mövqe İçi üçün, dilimlə mə və s. üçün hansı tam mövqenin istifadə olunacağını indi aydınlaşdırmalısınız. Başlanğıc mövqeyi üçün bu, ümumiyyə tlə , aşağı (sol) də yə rdir, son mövqe üçün isə bu, ümumiyyə tlə daha yüksə k (sağ) də yə r olacaqdır.

SimpleLocation obyektini çap etsə niz, mə lumatın gözə l tə svirini ə ldə edə bilə rsiniz:

```
>>> çap (mə nim_yerim) [>5:
 (8^9)]
```

Biz yerin başlanğıc və son atributlarından istifadə edə rə k qeyri-sə lis başlanğıc və son mövqelə rə daxil ola bilə rik:

```
>>> my_location.start AfterPosition(5)
>>> çap (my_location.start)
>5

>>> my_location.end
BetweenPosition(9, sol=8, sağ=9) >>> çap
(mə nim_yerim.end) (8^9)
```

Machine Translated by Google

Ə gə r qeyri-sə lis mövqelə rlə mə şğul olmaq istə mirsinizsə və sadə cə rə qə mlə r istə yirsinizsə , onlar ə slində alt siniflə rdir. tam ə də dlə r tam ə də dlə r kimi işlə mə lidir:

```
>>> int(mə nim_yerim.start)
5
>>> int(mə nim_yerim.end)
9
```

Eynilə , qeyri-sə lis mövqelə rdə n narahat olmadan mövqe yaratmağı asanlaşdırmaq üçün sadə cə keçə bilə rsiniz FeaturePosition konstruktorlarına rə qə mlə rlə yazın və siz ExactPosition obyektlə rini geri alacaqsınız:

```
>>> də qiq_yer = SeqFeature.SimpleLocation(5, 9) >>> çap (də qiq_yer) [5:9] >>>
də qiq_yer.start Də qiqMövqe(5) >>>

int(də qiq_yer.start)


5
```

Bu, Biopython-da qeyri-sə lis mövqelə rlə mə şğul olmağın ə n ə sas mə qamlarıdır. O, elə qurulmuşdur ki, qeyri-sə lisliklə mə şğul olmaq də qiq mövqelə rlə mə şğul olmaqdan o qə də r də mürə kkə b deyil və ümid edirə m ki, siz bunu doğru tapacaqsınız!

4.3.2.4 Yerin yoxlanılması

Python açar sözünü SeqFeature və ya yer obyekti ilə birlikdə ana koordinat üçün ə sas/qalığın xüsusiyyə t/yerdə olub-olmadığını görmə k üçün istifadə edə bilə rsiniz.

Mə sə lə n, tutaq ki, sizi maraqlandıran bir SNP var və siz bu SNP-nin hansı xüsusiyyə tlə rin daxilində olduğunu bilmə k istə yirsiniz və bu SNP-nin 4350 indeksində olduğunu düşünə k (Python hesablanır!). Budur, bütün xüsusiyyə tlə ri bir-bir dövrə də yoxladığımız sadə bir kobud güc hə lli:

```
>>> Bio import SeqIO -dan >>> my_snp =
4350 >>> rekord =
SeqIO.read("NC_005816.gb", "genbank") >>> record.features -də ki xüsusiyyə t üçün :

...         my_snp funksiyadadırsa :
...             print("%s %s" % ( feature.type, feature.qualifiers.get("db_xref")))
...
mə nbə ['taxon:229193'] gen
['GeneID:2767712']
CDS ['GI:45478716', 'GeneID:2767712']
```

Qeyd edə k ki, GenBank və ya birlə şmə lə rlə müə yyə n edilmiş EMBL fayllarından gen və CDS xüsusiyyə tlə ri ekzonların birliyidir - onlar heç bir intronu ə hatə etmir.

4.3.3 Xüsusiyyə t və ya yerlə tə svir olunan ardıcıllıq

SeqFeature və ya yer obyekti birbaşa ardıcıllığı ehtiva etmir, bunun ə və zinə yer (Bölmə 4.3.2-yə baxın) bunu ə sas ardıcıllıqdan necə ə ldə etmə yi tə svir edir. Mə sə lə n, ə ks zə ncirdə 5:18 yeri olan (qısa) gen ardıcıllığını nə zə rdə n keçirə k ki, GenBank/ EMBL notasiyasında 1 ə saslı hesablamadan istifadə edə rə k, tamamlayıcı (6..18) olacaq, aşağıdakı kimi:

Machine Translated by Google

```
>>> Bio.Seq -də n idxal Seq >>>
Bio.SeqFeature idxal SeqFeature, SimpleLocation >>> seq =
Seq("ACCGAGACGGCAAAGGCTAGCATAGGTATGAGACTTCCTTCCTGCCAGTGCTGAGGAACTGGGAGCCTAC") >>> xüsusiyyə t =
SeqFeature (5, strand1 ), növü = "gen")
```

Siz ə sas ardıcıllığı götürə , 5:18-i çıxarmaq üçün dilimlə yə və sonra ə ks tamamlayıcıyı götürə bilə rsiniz. Xüsusiyyə t yerinin başlanğıcı və sonu tam ə də də bə nzə yir, buna görə də bu işlə yir:

```
>>> feature_seq = seq[feature.location.start : feature.location.end].reverse_complement() >>> print(feature_seq)

AGCCTTTGCCGTC
```

Bu sadə bir nümunə dir, buna görə də pis deyil – lakin mürə kkə b xüsusiyyə tlə rlə (qoşulmaqla) mə şğul olduqdan sonra bu, olduqca qarışıqdır. Bunun ə və zinə , SeqFeature obyektində bütün bunlara diqqə t yetirmə k üçün çıxarış metodu var (və Biopython 1.78 istinad edilmiş ardıcıllıqların lüğə tini tə min etmə klə trans-splicing ilə mə şğul ola bilə r):

```
>>> xüsusiyyə t_seq = xüsusiyyə t.çıxarış(seq) >>> çap
(xüsusiyyə t_seq)
AGCCTTTGCCGTC
```

SeqFeature və ya yerin uzunluğu onun tə svir etdiyi ardıcıllıq bölgə sinin uzunluğuna uyğun gə lir.

```
>>> print(len(feature_seq)) 13 >>>

print(len(xüsusiyyə t)) 13 >>>

print(len(feature.location))
13
```

SimpleLocation obyektlə ri üçün uzunluq yalnız başlanğıc və son mövqelə r arasındakı fə rqdir. Necə - hə mişə , bir CompoundLocation üçün uzunluq tə rkib hissə lə rinin cə midir.

4.4 Müqayisə

SeqRecord obyektlə ri çox mürə kkə b ola bilə r, lakin burada sadə bir nümunə var:

```
>>> Bio.Seq -də n idxal Seq >>>
Bio.SeqRecord - dan idxal SeqRecord >>> record1 =
SeqRecord(Seq("ACGT"), id="test") >>> record2 =
SeqRecord(Seq("ACGT"), id="test")
```

Bu “eyni” qeydlə ri müqayisə etmə yə çalışdığınız zaman nə baş verir?

```
>>> qeyd1 == qeyd2
```

Ola bilsin ki, Biopython-un tə ə ccüblü köhnə versiyaları SeqRecord üçün Python-un defolt obyekt müqayisə sində n istifadə edə cə k, yə ni record1 == record2 yalnız bu də yişə nlə r yaddaşda eyni obyektə işarə etdikdə True qaytaracaq. Bu misalda record1 == record2 burada False də yə rini qaytarardı!

```
>>> record1 == record2 # Biopython-un köhnə versiyalarında!
Yalan
```

Biopython 1.67-də n etibarə n, record1 == record2 kimi SeqRecord müqayisə si bunun ə və zinə açıq-aydın mə lumat verə cə kdir. insanların buna tutulmaması üçün sə hv:

Machine Translated by Google

```
>>> qeyd1 == qeyd2
Traceback (ə n son zə ng):
 ...
NotImplementedError: SeqRecord müqayisə si qə sdə n hə yata keçirilmir. Maraq atributlarını açıq şə kildə müqayisə edin.
```

Bunun ə və zinə sizi maraqlandıran atributları, mə sə lə n, identifikator və ardıcıllığı yoxlamalısınız:

```
>>> record1.id == record2.id
Doğrudur
>>> record1.seq == record2.seq Doğrudur
```

Diqqə t edin ki, mürə kkə b obyektlə rin müqayisə si tez bir zamanda çə tinlə şir (hə mçinin Bölmə 3.10-a baxın).

4.5 İstinadlar

Ardıcıllıqla bağlı digə r ümumi annotasiya jurnala və ya ardıcıllıqla ə laqə li başqa nə şr edilmiş işə istinaddır. Biopython-da Referansı tə msil etmə yin kifayə t qə də r sadə üsulumuz var – bizdə obyektin atributları kimi istinad haqqında müvafiq mə lumatları saxlayan Bio.SeqFeature.Reference sinifimiz var.

Atributlara jurnal, başlıq və müə lliflə r kimi istinadda görmə yi gözlə diyiniz şeylə r daxildir. Bundan ə lavə , o, hə mçinin medline_id və pubmed_id və istinad haqqında şə rhi saxlaya bilə r.
Bunların hamısına sadə cə obyektin atributları kimi daxil olur.

İstinad hə mçinin yerlə şdirmə obyektinə malikdir ki, o, istinadın istinad etdiyi ardıcıllıqda xüsusi yeri tə yin edə bilsin. Mə sə lə n, BAC-də yerlə şə n müə yyə n bir genlə mə şğul olan bir jurnalınız ola bilə r və onun yalnız bu mövqeyə tam olaraq istinad etdiyini müə yyə n etmə k istə yə bilə rsiniz. Mə kan bölmə 4.3.2- də tə svir olunduğu kimi potensial qeyri-sə lis yerdir .

İstə nilə n istinad obyektlə ri siyahı kimi SeqRecord obyektinin annotasiyalar lüğə tində "istinadlar" açarı altında saxlanılır. Bütün bunlar da var. İstinadların öhdə sində n gə lmə k asan olması nə zə rdə tutulur və ümid edirə m ki, bir çox istifadə hallarını ə hatə edə cə k qə də r ümumidir.

4.6 Formatlaşdırma metodu

SeqRecord sinifinin format() metodu FASTA kimi Bio.SeqIO tə rə fində n də stə klə nə n çıxış fayl formatlarından biri ilə formatlanmış qeydinizi ehtiva edə n sə tir verir:

```
>>> Bio.Seq -də n idxal Seq >>>
Bio.SeqRecord- dan idxal SeqRecord >>> rekord =

 ...
SeqRecord( Seq( "MMYQQGCFAGGTVLRLAKDLAENNRGARVLVVCSEITAVTFRGPSETHLDSMVGQALFGD"
 ...                "GAGAVIVGSDPDLSVERPLYELVWTGATLPDSEGAIDGHLREVGLTFHLLKDVPGLISK"
 ...                "NIEKSLKEAFTPLGISDWNSTFWIAHPGGPAILDQVEAKLGLKEEKMRATREVLSEYGNM"
 ...                "TQDK"
 ...          ),
 ...          id="gi|14150838|gb|AAK54648.1|AF376133_1",
 ...          description="xalkon sintaza [Cucumis sativus]",
 ... )
>>> çap (record.format("fasta"))
```

hansı vermə lidir:

Machine Translated by Google

```
>gi|14150838|gb|AAK54648.1|AF376133_1 xalkon sintaza [Cucumis sativus]
MMYQQGCFAGGTVLRLAKDLAENNRGARVLVVCSEITAVTFRGPSETHLDSMVGQALFGD
GAGAVIVGSDPDLSVERPLYELVWTGATLLPDSEGAIDGHLREVGLTFHLLKDVPGLISK
NIEKSLKEAFTPLGISDWNSTFWIAHPGGPAILDQVEAKLGLKEEKMRATREVLSEYGNM
TQDK
<BLANKLINE>
```

Bu format metodu tə k mə cburi arqument, çıxış formatı kimi Bio.SeqIO tə rə fində n də stə klə nə n kiçik hə rf sə trini götürür (bax. Fə sil 5). Bununla belə , Bio.SeqIO fayl formatlarının bə zilə ri birdə n çox qeyd tə lə b etmə k üçün yaza bilə r (adə tə n bir neçə ardıcıl düzülmə formatları üçün və ziyyə t) və buna görə də bu format() metodu ilə işlə mə yə cə k. Hə mçinin Bölmə 5.5.4-ə baxın .

4.7 SeqRecord-un dilimlə nmə si

Siz ardıcıllığın yalnız bir hissə sini ə hatə edə n yeni SeqRecord vermə k üçün SeqRecord-u kə sə bilə rsiniz. Burada vacib olan odur ki, hə r hə rf üçün hə r hansı annotasiyalar da dilimlə nir və tamamilə yeni ardıcıllıqla daxil olan hə r hansı xüsusiyyə tlə r qorunur (yerlə ri düzə ldilir).

Mə sə lə n, ə vvə llə r istifadə edilə n eyni GenBank faylını götürə k:

```
>>> Bio import SeqIO- dan >>> rekord =
SeqIO.read("NC_005816.gb", "genbank") >>> qeyd

SeqRecord(seq=Seq('TGTAACGAACGGTGCAATAGTGATCCACACCCAACGCCTGAAATCAGATCCAGG...CTG'), id='NC6.'NC80', ad='NC_016',' description='Yersinia
pestis biovar Microtus küç 91001 plazmid pPCP1, tam ardıcıllıq', dbxrefs=['Layihə :58037']) >>> len(record)


9609
>>> len(record.features)
41
```

Bu nümunə üçün biz YP_pPCP05 olan pim geninə diqqə t yetirə cə yik. Birbaşa GenBank faylına nə zə r salsanız, bu genin/CDS-nin 4343..4780 və ya Python-da 4342:4780 sayılan yer sə tirinə malik olduğunu görə bilə rsiniz.
Fayla baxaraq, bunların fayldakı on ikinci və on üçüncü qeydlə r olduğunu öyrə nə bilə rsiniz, buna görə də Python sıfır ə saslı hesablamada onlar xüsusiyyə tlə r siyahısında 11 və 12 girişlə ridir:

```
>>> print(record.features[20]) növü: gen yeri:
[4342:4780](+)
kvalifikatorlar:

    Açar: db_xref, Də yə r: ['GeneID:2767712']
    Açar: gen, Də yə r: ['pim']
    Açar: locus_tag, Də yə r: ['YP_pPCP05']
<BLANKLINE> >>>
print(record.features[21]) növü: CDS yeri: [4342:4780]
(+) kvalifikatorlar:
Açar: codon_start, Də yə r: ['1']


    Açar: db_xref, Də yə r: ['GI:45478716', 'GeneID:2767712']
    Açar: gen, Də yə r: ['pim']
    Açar: locus_tag, Də yə r: ['YP_pPCP05']
```

Machine Translated by Google

Açar: qeyd, Də yə r: ['Yersinia pestis plazmid pPCP-nin ə vvə llə r ardıcıl olaraq pestisin toxunulmazlığı zülal girişlə rinə bə nzə yir, mə sə lə n, gi| 16082683|,ref|NP_395230.1| (NC_003132) , gi|1200166|emb|CAA90861.1| (Z54145) , gi|1488655| emb|CAA63439.1| (X92856) gi|2996219|gb|AAC62543.1| (AF053945) və gi|5763814|emb|CAB531 67.1| , (AL109969)']
,
Açar: mə hsul, Də yə r: ['pesticin immunitet zülalı']
Açar: protein_id, Də yə r: ['NP_995571.1']
Açar: transl_table, Də yə r: ['11']
Açar: tə rcümə , Də yə r: ['MGGGMISKLFCLALIFLSSSSGLAEKNTYTAKDILQNLELNTFGNSLSHGIYGKQTTFK QTEFTNIKSNTKKHIALINKDNSWMISLKILGIKRDEYTVCFEDFSLIRPPTYVAIHPLLIKKVKSGNSIPPTYVAIHPLLIKKVKSGNFIVKETVVC']
<BLANKLINE>

Gə lin bu ana rekordu 4300-də n 4800-ə qə də r kə sə k (pim geni/CDS daxil etmə k üçün kifayə tdir) və görə k necə ə ldə etdiyimiz bir çox xüsusiyyə t:

```
>>> sub_record = rekord[4300:4800] >>> sub_record

SeqRecord(seq=Seq('ATAAATAGATTATTCCAAATAATTTATTTATGTAAGAACAGGATGGGAGGGGGA...TTA'), id='NC_005816.1', name='NC_005816. 91001
plazmid pPCP1, tam ardıcıllıq', dbxrefs=[]) >>> len(alt_rekord) 500 >>> len(alt_record.features)
```

2

Bizim alt qeydimiz sadə cə iki xüsusiyyə tə malikdir, YP_pPCP05 üçün gen və CDS qeydlə ri:

```
>>> print(sub_record.features[0]) növü: gen
yeri: [42:480]
(+) kvalifikatorlar:

Açar: db_xref, Də yə r: ['GeneID:2767712']
Açar: gen, Də yə r: ['pim']
Açar: locus_tag, Də yə r: ['YP_pPCP05']
<BLANKLINE>
>>> print(sub_record.features[1]) növü: CDS
yeri: [42:480]
(+) kvalifikatorlar: Açar:
codon_start,
Də yə r: ['1']
Açar: db_xref, Də yə r: ['GI:45478716', 'GeneID:2767712']
Açar: gen, Də yə r: ['pim']
Açar: locus_tag, Də yə r: ['YP_pPCP05']
Açar: qeyd, Də yə r: ['Yersinia pestis plazmid pPCP-nin ə vvə llə r ardıcıl olaraq pestisin toxunulmazlığı zülal girişlə rinə bə nzə yir, mə sə lə n, gi| 16082683|,ref|NP_395230.1| (NC_003132) , gi|1200166|emb|CAA90861.1| (Z54145 ) gi|2996219|gb| AAC62543.1| (AF053945) , gi|1488655| emb|CAA63439.1| (X92856) və gi| ,
, 5763814|emb|CAB531 67.1| (AL109969)']
Açar: mə hsul, Də yə r: ['pesticin immunitet zülalı']
Açar: protein_id, Də yə r: ['NP_995571.1']
Açar: transl_table, Də yə r: ['11']
Açar: tə rcümə , Də yə r: ['MGGGMISKLFCLALIFLSSSSGLAEKNTYTAKDILQNLELNTFGNSLSHGIYGKQTTFK QTEFTNIKSNTKKHIALINKDNSWMISLKILGIKRDEYTVCFEDFSLIRPPTYVAIHPLLIKKVKSGNSIPPTYVAIHPLLIKKVKSGNFIVKETVVC']
<BLANKLINE>
```

Machine Translated by Google

Diqqə t yetirin ki, onların yerlə ri yeni ana ardıcıllığı ə ks etdirmə k üçün düzə ldilib!

Biopython xüsusiyyə tlə ri (və hə rf üçün hə r hansı annotasiya) ilə mə ntiqli və intuitiv bir iş görsə də , digə r annotasiya üçün bunun hə lə də alt ardıcıllığa aid olub-olmadığını bilmə k mümkün deyil. Molekul tipi istisna olmaqla, tə xmin etmə mə k üçün .annotasiyalar və .dbxrefs alt qeyddə n çıxarılıb və müvafiq olaraq hə r hansı müvafiq mə lumatı ötürmə k sizin ixtiyarınızdadır.

```
>>> sub_record.annotations {'molecule_type':
'DNA'} >>> sub_record.dbxrefs []
```

Siz orqanizm kimi digə r girişlə ri qorumaq istə yə bilə rsinizmi? Bütün annotasiyalar lüğə tini kopyalamaqdan çə kinin, çünki bu halda sizin qismə n ardıcıllığınız artıq dairə vi DNT deyil - o, artıq xə ttidir:

```
>>> sub_record.annotations["topologiya"] = "xə tti"
```

Eyni mə qamı qeyd identifikatoru, adı və tə sviri haqqında da söylə mə k olar, lakin praktiklik üçün bunlar saxlanılır:

```
>>> sub_record.id
 'NC_005816.1' >>>
sub_record.name 'NC_005816'
 >>>
sub_record.description 'Yersinia pestis
 biovar Microtus küç. 91001 plazmid pPCP1, tam ardıcıllıq'
```

Bu problemi gözə l şə kildə göstə rir, baxmayaraq ki, bizim yeni alt qeydimiz plazmidin tam ardıcıllığı deyil, ona görə də tə svir sə hvdir! Gə lin bunu düzə ldə k və sonra yuxarıda Bölmə 4.6-da tə svir edilə n format metodundan istifadə edə rə k alt yazıya azaldılmış GenBank faylı kimi baxaq:

```
>>> sub_record.description = "Yersinia pestis biovar Microtus küç. 91001 plazmid pPCP1, qismə n" >>> çap (sub_record.format("genbank")[:200]
+ "...")

LOCUS NC_005816 500 bp TƏ SVİRİ Yersinia pestis biovar Microtus küç. 91001 plazmid pPCP1, DNT xə tti UNK 01-YAN-1980
qismə n.
ACCESSION NC_005816
VERSİYA NC_0058...
```

Hə r hə rf üçün annotasiyaların (oxu keyfiyyə ti balları) da dilimlə ndiyi bə zi FASTQ nümunə lə ri üçün Bölmə 22.1.7 və 22.1.8 - ə baxın .

4.8 SeqRecord obyektlə rinin ə lavə edilmə si

Siz yeni SeqRecord verə rə k SeqRecord obyektlə rini birlikdə ə lavə edə bilə rsiniz. Burada vacib olan odur ki, hə r hansı bir hə rf üçün ümumi annotasiyalar da ə lavə edilir, bütün funksiyalar qorunur (yerlə ri düzə ldilir) və hə r hansı digə r ümumi annotasiya da saxlanılır (id, ad və tə svir kimi).

Hə r hə rf annotasiyası olan nümunə üçün biz FASTQ faylında ilk qeyddə n istifadə edə cə yik. Fə sil 5 SeqIO funksiyalarını izah edə cə k:

```
>>> Bio import SeqIO >>> rekord =
sonrakı(SeqIO.parse("example.fastq", " fastq")) >>> len(record) 25
```

Machine Translated by Google

```
>>> çap (record.seq)
CCCTTCTTGTCTTCAGCGTTTCTCC >>> çap
edin(record.letter_annotations["phred_keyfiyyə t"]) [26, 26, 18, 26, 26, 26, 26, 26, 26, 26, 26, 26,
 26, 26, 26, 26, 22, 26, 26, 26, 23, 23]
```

Tutaq ki, bu, Roche 454 mə lumatları idi və sizcə , TTT yalnız TT olmalıdır. SeqRecord-u "ə lavə " üçüncü T-də n ə vvə l və sonra dilimlə mə klə yeni redaktə edilmiş rekord yarada bilə rik:

```
>>> sol = qeyd[:20] >>> çap
(left.seq)
CCCTTCTTGTCTTCAGCGTT >>> çap
edin(left.letter_annotations["phred_quality"]) [26, 26, 18, 26, 26, 26, 26, 26, 26, 26, 26, 26,
 26, 26, 26, 66, 26, 26, 66, >>2, sağ ] = qeyd[21:] >>> çap (sağ.seq)

CTCC
>>> çap (sağ.letter_annotations["phred_quality"]) [26, 26, 23, 23]
```

İndi iki hissə ni birlikdə ə lavə edin:

```
>>> redaktə = sol + sağ >>> lent
(redaktə edilmiş)
24
>>> çap (redaktə edilmiş. ardıcıl)
CCCTTCTTGTCTTCAGCGTTCTCC
>>> çap edin(edited.letter_annotations["phred_quality"]) [26, 26, 18, 26, 26,
 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 22, 26, 22, 26, 2, 2, 23]
```

Asan və intuitiv? Ümid edirik ki! Bunu sadə cə olaraq qısalda bilə rsiniz:

```
>>> redaktə edildi = qeyd[:20] + qeyd[21:]
```

İndi xüsusiyyə tlə ri olan bir nümunə üçün GenBank faylından istifadə edə cə yik. Fə rz edə k ki, sizin dairə vi genomunuz var:

```
>>> Bio import SeqIO- dan >>> rekord =
SeqIO.read("NC_005816.gb", "genbank") >>> rekord

SeqRecord(seq=Seq('TGTAACGAACGGTGCAATAGTGATCCACACCCAACGCCTGAAATCAGATCCAGG...CTG'), id='NC_005816.1', ad='NC_005816',
description='Yersinia pestis biovar Microtus küç., 910-ci seqment 9100 dbxrefs=['Layihə :58037']) >>> len(rekord) 9609 >>> len(record.features) 41 >>>
record.dbxrefs ['Project:58037'] >>> record.annotations.keys() dict_keys(['moleculeculecule'_da'fi' 'tarix', 'accessions',
'sequence_version', 'gi',
'açar
sözlə r', 'mə nbə ', 'orqanizm', 'taksonomiya',

'istinadlar', 'şə rh'])
```

Machine Translated by Google

Mə nşə yi belə də yişə bilə rsiniz:

```
>>> də yişdi = rekord[2000:] + rekord[:2000]
>>> də yişdi
SeqRecord(seq=Seq('GATACGCAGTCATATTTTTTACACAATTCTCTAATCCCGACAAGGTCGTAGGTC...GGA'), id='NC_005816.1',
name='NC_005816', description='Yersinia pestis biovar Microtus küç., tam 91000 sıra, pmidCP' dbxrefs=[]) >>> len(köçürülmüş)


9609
```

Nə zə rə alın ki, bu mükə mmə l deyil, çünki verilə nlə r bazası çarpaz istinadları kimi bə zi annotasiyalar, molekul növündə n başqa bütün annotasiyalar və xüsusiyyə tlə rdə n biri (mə nbə xüsusiyyə ti) itirilib:

```
>>> len(keçidilmiş.xüsusiyyə tlə r)
40
>>> shifted.dbxrefs
 []
>>> shifted.annotations.keys()
dict_keys(['molekul_növü'])
```

Bunun sə bə bi, SeqRecord dilimlə mə addımının saxladığı annotasiyada ehtiyatlı olmasıdır (annotasiyanın sə hv yayılması böyük problemlə rə sə bə b ola bilə r). Ə gə r verilə nlə r bazası çarpaz istinadları və ya annotasiyalar lüğə tini saxlamaq istə yirsinizsə , bu, açıq şə kildə edilmə lidir:

```
>>> shifted.dbxrefs = record.dbxrefs[:] >>> shifted.annotations =
record.annotations.copy() >>> shifted.dbxrefs

 ['Layihə :58037'] >>>
shifted.annotations.keys()
dict_keys(['molecule_type', 'topology', 'data_file_division', 'tarix', 'accessions', 'sequence_version', 'gi', 'source''xonism','','or 'istinadlar',
 'şə rh'])
```

Hə m də qeyd edin ki, bu kimi bir nümunə də , ehtimal ki, NCBI-də n bə ri qeyd identifikatorlarını də yişdirmə lisiniz istinadlar orijinal də yişdirilmə miş ardıcıllığa istinad edir.

4.9 Ə ks tamamlayan SeqRecord obyektlə ri

Biopython 1.57-də yeni xüsusiyyə tlə rdə n biri SeqRecord obyektinin reverse_complement metodu idi. Bu, tə rs tamamlanmış qeyddə ki annotasiya ilə nə edə cə yinə dair narahatlıqlarla istifadə si asanlığı balanslaşdırmağa çalışır.

Ardıcıllıq üçün bu, Seq obyektinin tə rs tamamlama metodundan istifadə edir. İstə nilə n xüsusiyyə tlə r yer və ipin yenidə n hesablanması ilə ötürülür. Eyni şə kildə hə r hə rf üçün hə r hansı annotasiya da kopyalanır, lakin tə rsinə çevrilir (bu, keyfiyyə t balları kimi tipik nümunə lə r üçün mə na kə sb edir). Bununla belə , ə ksə r annotasiyaların ötürülmə si problemlidir.

Mə sə lə n, ə gə r qeyd identifikatoru qoşulma idisə , bu qoşulma hə qiqə tə n tə rs tamamlanan ardıcıllığa tə tbiq edilmə mə lidir və identifikatorun defolt olaraq ötürülmə si aşağı axın tə hlilində asanlıqla incə mə lumat korlanmasına sə bə b ola bilə r. Buna görə də , defolt olaraq, SeqRecord-un id, adı, tə sviri, annotasiyaları və verilə nlə r bazası çarpaz istinadlarının hamısı defolt olaraq ötürülmür.

SeqRecord obyektinin reverse_complement metodu qeydin xassə lə rinə uyğun gə lə n bir sıra ə lavə arqumentlə r götürür. Bu arqumentlə ri True olaraq tə yin etmə k köhnə də yə rlə ri kopyalamaq, False isə köhnə də yə rlə ri buraxmaq və standart də yə ri istifadə etmə k demə kdir. Bunun ə və zinə alternativ olaraq yeni istə diyiniz də yə ri tə min edə bilə rsiniz.

Bu qeyd nümunə sini nə zə rdə n keçirin:

Machine Translated by Google

```
>>> Bio import SeqIO - dan >>> rec =
SeqIO.read("NC_005816.gb", "genbank") >>> print(rec.id, len(rec),
len(rec.features), len(rec.dbxrefs), len(rec.annotations))
NC_005816.1 9609 41 1 13
```

Burada ə ks tamamlayıcını götürürük və yeni identifikator tə yin edirik - lakin annotasiyanın ə ksə riyyə tinin necə olduğuna diqqə t yetirin atılır (lakin xüsusiyyə tlə r deyil):

```
>>> rc = rec.reverse_complement(id="TESTING") >>> print(rc.id, len(rc),
len(rc.features), len(rc.dbxrefs), len(rc.annotations))
SINAQ 9609 41 0 0
```

Machine Translated by Google

Fə sil 5

Giriş/Çıxış ardıcıllığı

Bu fə sildə biz 2-ci Fə sildə qısa şə kildə tə qdim edilmiş və hə mçinin 4-cü Fə sildə istifadə olunan Bio.SeqIO modulunu daha ə traflı müzakirə edə cə yik. Bunun mə qsə di müxtə lif ardıcıllıq fayl formatları ilə vahid şə kildə işlə mə k üçün sadə interfeys tə min etmə kdir. Hə mçinin Bio.SeqIO viki sə hifə sinə baxın (http://biopython.org/wiki/SeqIO), və daxili sə nə dlə r (hə mçinin onlayn):

```
>>> Bio import SeqIO- dan >>>
kömə k (SeqIO)
```

"Tuta" ondan ibarə tdir ki, siz Seq obyekti (3-cü Fə silə baxın) və identifikator və tə svir kimi annotasiyadan ibarə t SeqRecord obyektlə ri ilə işlə mə lisiniz (Fə sil 4-ə baxın). Nə zə rə alın ki, çox böyük FASTA və ya FASTQ faylları ilə işlə yə rkə n, bütün bu obyektlə rlə işlə mə k yükü skriptlə ri çox yavaş edə bilə r. Bu halda, hə r bir qeyd üçün sadə cə bir sıra sə tir qaytaran aşağı sə viyyə li SimpleFastaParser və FastqGeneralIterator analizatorlarını nə zə rdə n keçirin (bax: Bölmə 5.6).

5.1 Ardıcıllıqların tə hlili və ya oxunması

İşçi funksiyası Bio.SeqIO.parse() SeqRecord obyektlə ri kimi ardıcıllıqla verilə nlə ri oxumaq üçün istifadə olunur. Bu funksiya iki arqument gözlə yir:

1. Birinci arqument verilə nlə ri oxumaq üçün tutacaq və ya fayl adıdır. Də stə k adə tə n oxumaq üçün açılmış fayldır, lakin komanda xə tti proqramından çıxış və ya internetdə n endirilmiş mə lumat ola bilə r (bax: Bölmə 5.3). Tutacaqlar haqqında ə traflı mə lumat üçün Bölmə 25.1-ə baxın .
2. İkinci arqument ardıcıllıq formatını göstə rə n kiçik hə rfdir – biz sizin üçün fayl formatını tə xmin etmə yə çalışmırıq! Bax http://biopython.org/wiki/SeqIO də stə klə nə n formatların tam siyahısı üçün.

Bio.SeqIO.parse() funksiyası SeqRecord obyektlə rini verə n iteratoru qaytarır. İteratorlar adə tə n aşağıda göstə rildiyi kimi for loopunda istifadə olunur.

Bə zə n özünüzü yalnız bir qeyddə n ibarə t fayllarla mə şğul olursunuz. Bu və ziyyə t üçün eyni arqumentlə ri qə bul edə n Bio.SeqIO.read() funksiyasından istifadə edin. Faylda bir və yalnız bir qeyd olması şə rtilə , bu SeqRecord obyekti kimi qaytarılır. Ə ks halda bir istisna qaldırılır.

5.1.1 Ardıcıllıq fayllarının oxunması

Ümumiyyə tlə , Bio.SeqIO.parse() SeqRecord obyektlə ri kimi ardıcıl faylları oxumaq üçün istifadə olunur və adə tə n belə bir for döngə si ilə istifadə olunur:

Machine Translated by Google

```
Bio idxaldan SeqIO

seq_record üçün SeqIO.parse ("ls_orchid.fasta", "fasta"): print(seq_record.id)
        print(repr(seq_record.seq))
        print(len(seq_record))
```

Yuxarıdakı nümunə Bölmə 2.4- də ki girişdə n tə krarlanır və orxideya DNT ardıcıllığını FASTA formatlı ls orchid.fasta faylına yüklə yə cə k . Bunun ə və zinə ls orchid.gbk kimi GenBank formatlı faylı yüklə mə k istə sə niz onda sizə lazım olan tə-k şey fayl adını və format sə tirini də yişmə kdir:

```
Bio idxaldan SeqIO

seq_record üçün SeqIO.parse ("ls_orchid.gbk", "genbank"): print(seq_record.id)
        print(repr(seq_record.seq))
        print(len(seq_record))
```

Eynilə , ə gə r siz başqa fayl formatında faylı oxumaq istə sə niz, Bio.SeqIO.parse() funksiyasının onu də stə klə diyini fə rz etsə k, sadə cə format sə tirini uyğun olaraq də yişdirmə lisiniz, mə sə lə n, SwissProt faylları üçün "swiss" və ya EMBL mə tn faylları üçün "embl". Wiki sə hifə sində tam siyahı var (http://biopython.org/wiki/ SeqIO) və daxili sə nə dlə rdə (hə mçinin onlayn).

Python iteratorundan istifadə etmə yin başqa bir çox yayılmış yolu siyahının başa düşülmə si (və ya generator ifadə si) daxilində dir. Mə sə lə n, fayldan çıxarmaq istə diyiniz hə r şey qeyd identifikatorlarının siyahısı idisə , biz bunu aşağıdakı siyahı başa düşmə klə asanlıqla edə bilə rik:

```
>>> Bio import SeqIO -dan >>>
identifikatorları = [ SeqIO.parse -də seq_record.id ( "ls_orchid.gbk", "genbank")]
>>> identifikatorları
['Z78533.1', 'Z78532.1', 'Z78531.1', 'Z78530.1', 'Z78529.1', 'Z78527.1', ..., 'Z78439.1']
```

Bölmə 22.2- də bu kimi siyahı başa düşmə k üçün SeqIO.parse() funksiyasından istifadə edə n daha çox nümunə var (mə sə lə n, ardıcıllığın uzunluqlarının və ya GC%-nin planlaşdırılması üçün).

5.1.2 Ardıcıllıq faylında qeydlə rin tə krarlanması

Yuxarıdakı nümunə lə rdə biz adə tə n bütün qeydlə ri bir-bir tə krarlamaq üçün for loopundan istifadə etdik. Siz iterasiya interfeysini də stə klə yə n bütün növ Python obyektlə ri (o cümlə də n siyahılar, tuplelə r və sə tirlə r) ilə for loopundan istifadə edə bilə rsiniz.

Bio.SeqIO tə rə fində n qaytarılan obyekt ə slində SeqRecord obyektlə rini qaytaran iteratordur. Hə r rekordu növbə ilə görə bilə rsiniz, ancaq bir də fə və yalnız bir də fə . Üstün cə hə t odur ki, iterator böyük fayllarla işlə yə rkə n yaddaşınıza qə naə t edə bilə r.

For döngə sində n istifadə etmə k ə və zinə , aşağıdakı kimi girişlə ri keçmə k üçün iteratorda növbə ti () funksiyasından istifadə edə bilə rsiniz:

```
Bio idxaldan SeqIO

rekord_iterator = SeqIO.parse("ls_orchid.fasta", "fasta")

first_record = sonrakı (record_iterator) çap
(ilk_record.id) çap
(ilk_record.description)
```

Machine Translated by Google

```
second_record = növbə ti (record_iterator) çap
(ikinci_rekord.id) çap
(ikinci_rekord.tə svir)
```

Nə zə rə alın ki, next() funksiyasından istifadə etmə yə cə hd etsə niz və başqa nə ticə yoxdursa, xüsusi StopIteration istisnasını ə ldə edə cə ksiniz.

Nə zə rə alınmalı olan xüsusi hallardan biri, ardıcıllıq fayllarınızda çoxlu qeydlə rin olmasıdır, lakin siz yalnız birincisini istə yirsiniz. Bu və ziyyə tdə aşağıdakı kod çox qısadır:

```
Bio idxaldan SeqIO

first_record = sonrakı(SeqIO.parse("ls_orchid.gbk", "genbank"))
```

Burada bir xə bə rdarlıq sözü - bu kimi növbə ti () funksiyasından istifadə fayldakı hə r hansı ə lavə qeydlə ri sə ssizcə rə dd edə cə k. Ə gə r bu fə slin sonrakı bə zi onlayn nümunə lə ri və ya tə k xromosom üçün GenBank faylı kimi fayllarınızda bir və yalnız bir qeyd varsa, bunun ə və zinə yeni Bio.SeqIO.read() funksiyasından istifadə edin. Bu, ə lavə gözlə nilmə z qeydlə rin olmadığını yoxlayacaq.

5.1.3 Ardıcıllıq faylında qeydlə rin siyahısının ə ldə edilmə si

Ə vvə lki bölmə də biz Bio.SeqIO.parse()-nin sizə SeqRecord iteratoru vermə si və qeydlə ri bir-bir ə ldə etmə yiniz haqqında danışdıq. Çox vaxt qeydlə rə istə nilə n qaydada daxil olmaq lazımdır.
Python siyahısı mə lumat növü bunun üçün mükə mmə ldir və biz daxili Python funksiyasının list() funksiyasından istifadə edə rə k qeyd iteratorunu SeqRecord obyektlə rinin siyahısına çevirə bilə rik:

```
Bio idxaldan SeqIO

qeydlə r = siyahı(SeqIO.parse("ls_orchid.gbk", "genbank"))

print(" %i qeyd tapıldı" % len(qeydlə r))

print(" Son qeyd") last_record =
qeydlə r[-1] # Python siyahı hiylə lə rində n istifadə edə rə k
print(last_record.id)
print(repr(last_record.seq))
print(len(last_record))

print(" İlk qeyd") first_record =
qeydlə r[0] # yadda saxla, Python sıfırdan hesablanır print(first_record.id)
print(repr(first_record.seq))
print(len(first_record))
```

Verilmə si:

```
94 qeyd tapıldı
Son rekord
Z78439.1
Seq('CATTGTTGAGATCACATAATAATTGATCGAGTTAATCTGGAGGATCTGTTTACT...GCC') 592

İlk rekord
Z78533.1
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGATGAGACCGTGG...CGC') 740
```

Machine Translated by Google

Siz ə lbə ttə ki, hə lə də SeqRecord obyektlə rinin siyahısı ilə for loopundan istifadə edə bilə rsiniz. Siyahıdan istifadə iteratordan daha çevikdir (mə sə lə n, siyahının uzunluğundan yazıların sayını müə yyə n edə bilə rsiniz), lakin daha çox yaddaş tə lə b olunur, çünki o, bütün qeydlə ri bir anda yaddaşda saxlayacaq.

5.1.4 Mə lumatların çıxarılması

SeqRecord obyekti və onun annotasiya strukturları 4-cü Fə sildə daha ə traflı tə svir edilmişdir. Annotasiyaların necə saxlandığına misal olaraq, GenBank ls orchid.gbk faylında ilk qeydin tə hlilində n ə ldə edilə n nə ticə yə baxacağıq .

```
Bio idxaldan SeqIO

rekord_iterator = SeqIO.parse("ls_orchid.gbk", "genbank") ilk_rekord =
sonrakı(record_iterator) çap(ilk_rekord)
```

Bu belə bir şey vermə lidir:

```
ID: Z78533.1
Adı: Z78533
Tə svir: C.irapeanum 5.8S rRNA geni və ITS1 və ITS2 DNT.
Xüsusiyyə tlə rin sayı: 5
/sequence_version=1 /
source=Cypripedium irapeanum /
taksonomy=['Eukaryota', 'Viridiplantae', 'Streptophyta', ..., 'Cypripedium'] /açar sözlə r=['5.8S ribosomal RNT',
'5.8S', ITNA', ... 'ITS2'] /references=[...] /accessions=['Z78533'] /data_file_division=PLN /tarix=30-NOV-2006 /
organism=Cypripedium
irapeanum /gi=2765658
```

```
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGATGAGACCGTGG...CGC')
```

Bu, SeqRecord üçün annotasiya mə lumatlarının ə ksə riyyə tinin insan tə rə fində n oxuna bilə n xülasə sini verir. Bu nümunə üçün biz sadə cə Python lüğə ti olan .annotations atributundan istifadə edə cə yik. Bu annotasiya lüğə tinin mə zmunu yuxarıdakı qeydi çap etdikdə göstə rildi. Siz hə mçinin onları birbaşa çap edə bilə rsiniz:

```
çap (birinci_record.annotations)
```

İstə nilə n Python lüğə ti kimi siz də açarları asanlıqla ə ldə edə bilə rsiniz:

```
çap (ilk_rekord.annotations.keys())
```

və ya də yə rlə r:

```
çap (birinci_rekord.annotations.də yə rlə r())
```

Ümumiyyə tlə , annotasiya də yə rlə ri sə tirlə r və ya sə tirlə rin siyahısıdır. Xüsusi bir və ziyyə t, hə r hansı bir istinaddır fayl istinad obyektlə ri kimi saxlanılır.

Tutaq ki, siz ls orchid.gbk saytından növlə rin siyahısını çıxarmaq istə yirsiniz GenBank faylı. İstə diyimiz mə lumat, Cypripedium irapeanum, annotasiyalar lüğə tində 'mə nbə ' və 'orqanizm' altında saxlanılır və bu şə kildə ə ldə edə bilə rik:

Machine Translated by Google

```
>>> çap (first_record.annotations["mə nbə "])
Cypripedium irapeanum
```

və ya:

```
>>> çap (first_record.annotations["orqanizm"])
Cypripedium irapeanum
```

Ümumiyyə tlə , "orqanizm" elmi ad üçün istifadə olunur (latınca, mə sə lə n, Arabidopsis thaliana), "mə nbə " isə çox vaxt ümumi ad olacaq (mə sə lə n, thale cress). Bu nümunə də , tez-tez olduğu kimi, iki sahə eynidir.

İndi gə lin bütün qeydlə ri nə zə rdə n keçirə k, hə r bir orkide ardıcıllığının mə nşə yi növlə rinin siyahısını tə rtib edə k:

```
Bio idxaldan SeqIO

bütün_növlə r = []
SeqIO.parse ("ls_orchid.gbk", "genbank")- da seq_rekord üçün :
        all_species.append(seq_record.annotations["orqanizm"]) çap (bütün_növlə r)
```

Bu kodu yazmağın başqa bir yolu siyahı anlayışından istifadə etmə kdir:

```
Bio idxaldan SeqIO

all_species =
        [ seq_record.annotations["orqanizm"] SeqIO.parse
        -də seq_rekord üçün ("ls_orchid.gbk", "genbank")

] çap (bütün_növlə r)
```

Hə r iki halda nə ticə belə dir:

```
['Cypripedium irapeanum', 'Cypripedium californicum', ..., 'Paphiopedilum barbatum']
```

Ə la. Bu olduqca asan idi, çünki GenBank faylları standart şə kildə qeyd olunur.

İndi tutaq ki, siz GenBank faylından çox, FASTA faylından növlə rin siyahısını çıxarmaq istə yirsiniz. Pis xə bə r odur ki, qeydin tə svir xə ttində n istə diyiniz mə lumatları çıxarmaq üçün bə zi kodlar yazmalı olacaqsınız - ə gə r mə lumat ilk növbə də fayldadırsa! Nümunə miz FASTA format faylı ls orchid.fasta belə başlayır:

```
–

>gi|2765658|emb|Z78533.1|CIZ78533 C.irapeanum 5.8S rRNA geni və ITS1 və ITS2 DNT
CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGATGAGACCGTGGAATAAACGATCGAGTG
AATCCGGAGGACCGGTGTACTCAGCTCACCGGGGGCATTGCTCCCGTGGTGACCCTGATTTGTTGTTGGG
...
```

Siz ə l ilə yoxlaya bilə rsiniz, lakin hə r qeyd üçün növün adı tə svir sə tirində ikinci söz kimi göstə rilir. Bu o demə kdir ki, hə r bir qeydin .tə svirini boşluqlara bölsə k, növ orada bir nömrə li sahə kimidir (sıfır sahə rekord identifikatordur). Bu o demə kdir ki, biz bunu edə bilə rik:

```
>>> Bio import SeqIO -dan >>>
all_species = [] >>>
SeqIO.parse ("ls_orchid.fasta", "fasta"): all_species.append(seq_record.description.split()
...             [1]) seq_rekord üçün
...
>>> çap (bütün_növlə r) # doctest:+ELLIPSIS ['C.irapeanum',
 'C.californicum', 'C.fasciculatum', ...,                                    'P.barbatum']
```

Machine Translated by Google

Siyahı anlayışlarından istifadə edə rə k qısa alternativ aşağıdakılar olardı:

```
>>> Bio idxal SeqIO -dan >>>
bütün_növlə r = [
...         SeqIO.parse ("ls_orchid.fasta", "fasta") -da seq_record
...         üçün seq_record.description.split()[1]
... ]
>>> çap (bütün_növlə r) # doctest:+ELLIPSIS ['C.irapeanum',
'C.californicum', 'C.fasciculatum', ..., 'P.barbatum']
```

Ümumiyyə tlə , FASTA tə svir xə ttində n mə lumat çıxarmaq çox gözə l deyil. Ə gə r ardıcıllığınızı GenBank və ya EMBL kimi yaxşı şə rh edilmiş fayl formatında ə ldə edə bilsə niz, bu cür annotasiya mə lumatı ilə mə şğul olmaq daha asandır.

5.1.5 Mə lumatların də yişdirilmə si

Ə vvə lki bölmə də biz SeqRecord-dan mə lumatların necə çıxarılacağını nümayiş etdirdik. Başqa bir ümumi və zifə bu mə lumatları də yişdirmə kdir. SeqRecord-un atributları birbaşa də yişdirilə bilə r, mə sə lə n:

```
>>> Bio import SeqIO- dan >>>
record_iterator = SeqIO.parse("ls_orchid.fasta", "fasta") >>> first_record = next(record_iterator)
>>> first_record.id 'gi|2765658|emb|Z78533.1|CIZ78533' __
record >>> ilk qeyd >>> "ilk
 id . " 'yeni_id'
```

Qeyd edə k ki, ə gə r siz fayla yazılan zaman FASTA-nın çıxış yolunu də yişmə k istə yirsinizsə (Bölmə 5.5-ə baxın), onda siz hə m id, hə m də tə svir atributlarını də yişdirmə lisiniz. Düzgün davranışı tə min etmə k üçün id və istə diyiniz tə svirin ə vvə linə boşluq ə lavə etmə k yaxşıdır:

```
>>> Bio import SeqIO- dan >>> record_iterator =
SeqIO.parse("ls_orchid.fasta", "fasta") >>> first_record = next(record_iterator) >>> first_record.id = "new_id" >>>
first_record.description = first_record.id + >>> print(first_record.id + >>>
print(first_record.id) (first_record.id) >>> print(first_record.id)
>>> print(ilk_rekord.0)at[:"ne2)0 . İstə diyiniz yeni tə svir "" + "arzu olunan yeni tə svir"
CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGATGAGACCGTGGAATAAA

CGATCGAGTGAATCCGGAGGACCGGTGTACTCAGCTCACCGGGGGCATTGCTCCCGTGGT
GACCCTGATTTGTTGTTGGGCCGCCTCGGGAGCGTCCATGGCGGGT
```

5.2 Sıxılmış fayllardan ardıcıllıqların tə hlili

Ə vvə lki bölmə də biz fayldan ardıcıllıq mə lumatlarının tə hlilinə baxdıq. Fayl adından istifadə etmə k ə və zinə siz Bio.SeqIO-ya tutacaq verə bilə rsiniz (bax: Bölmə 25.1) və bu bölmə də biz sıxılmış fayllardan ardıcıllığı tə hlil etmə k üçün tutacaqlardan istifadə edə cə yik.

Yuxarıda gördüyünüz kimi, biz Bio.SeqIO.read() və ya Bio.SeqIO.parse() funksiyasını fayl adı ilə istifadə edə bilə rik - mə sə lə n, bu sürə tli nümunə generator ifadə sində n istifadə edə rə k çoxlu qeyd GenBank faylında ardıcıllıqların ümumi uzunluğunu hesablayır:

Machine Translated by Google

```
>>> Bio import SeqIO -dan >>>
print(sum(len(r) üçün r üçün SeqIO.parse ("ls_orchid.gbk", "gb"))) 67518
```

Burada biz avtomatik olaraq sapı bağlamaq üçün with ifadə sində n istifadə edə rə k, bunun ə və zinə fayl sapından istifadə edirik:

```
>>> Bio import SeqIO -dan >>>
açıq("ls_orchid.gbk") ilə qulp kimi : print(sum(len(r)
...         SeqIO.parse (sap, "gb") üçün r üçün )))
...
67518
```

Və ya sapı ə l ilə bağladığınız köhnə üsul:

```
>>> Bio import SeqIO -dan >>> handle =
open("ls_orchid.gbk") >>> print(sum(len(r) üçün SeqIO.parse
( handle, "gb"))) 67518 >>> handle.close()
```

İndi düşünə k ki, bunun ə və zinə gzip sıxılmış faylımız var? Bunlar Linux-da çox istifadə olunur. Biz edə bilə rik oxumaq üçün sıxılmış faylı açmaq üçün Python-un gzip modulundan istifadə edin - bu bizə bir tutacaq obyekti verir:

```
>>> gzip idxal >>>
Bio import SeqIO >>> ilə
gzip.open("ls_orchid.gbk.gz", "rt") ilə tutacaq olaraq :
...         print(seqIO.parse -də r üçün cə mi(len(r)) (tutacaq, "gb")))
...
67518
```

Eynilə bzip2 sıxılmış faylımız olsaydı:

```
>>> bz2 idxal >>>
Bio import SeqIO >>> ilə
bz2.open("ls_orchid.gbk.bz2", "rt") ilə tutacaq olaraq :
...         print(seqIO.parse -də r üçün cə mi(len(r)) (tutacaq, "gb")))
...
67518
```

BGZF (Bloklu GNU Zip Format) adlı gzip (GNU Zip) variantı var ki, onu oxumaq üçün adi gzip faylı kimi qə bul etmə k olar, lakin tə sadüfi giriş üçün üstünlüklə rə malikdir və bu barə də daha sonra Bölmə 5.4.4-də danışacağıq .

5.3 Şə bə kə də n ardıcıllıqların tə hlili

Ə vvə lki bölmə lə rdə biz fayldan (fayl adı və ya tutacaqdan istifadə etmə klə) və sıxılmış fayllardan (tutacaqdan istifadə etmə klə) ardıcıllıq mə lumatlarının tə hlilinə baxdıq. Burada biz internetdə n ardıcıllıqları yüklə mə k və tə hlil etmə k üçün Bio.SeqIO-dan başqa bir tutacaq növü, şə bə kə bağlantısı ilə istifadə edə cə yik.

Nə zə rə alın ki, ardıcıllıq mə lumatlarını yüklə yə və onu bir gedişdə SeqRecord obyektinə tə hlil edə bildiyiniz üçün bunun yaxşı fikir olduğu demə k deyil. Ümumiyyə tlə , yə qin ki, ardıcıllığı bir də fə endirmə li və tə krar istifadə üçün faylda saxlamalısınız.

Machine Translated by Google

5.3.1 GenBank qeydlə rinin şə bə kə də n tə hlili

Bölmə 12.6 Entrez EFetch interfeysi haqqında daha ə traflı danışır, lakin hə lə lik gə lin NCBI-yə qoşulaq və onların GI nömrə lə rində n istifadə edə rə k GenBank-dan bir neçə Opuntia (tikanlı armud) ardıcıllığı ə ldə edə k.

Hə r şeydə n ə vvə l, yalnız bir rekord gə tirə k. Annotasiyalar və funksiyalar sizi maraqlandırmırsa, FASTA faylını yüklə mə k yaxşı seçimdir, çünki bunlar yığcamdır. İndi yadda saxlayın ki, tutacaqda bir və yalnız bir qeyddə n ibarə t olacağını gözlə diyiniz zaman Bio.SeqIO.read() funksiyasından istifadə edin:

```
Bio idxal Entrez Bio idxal SeqIO


Entrez.email = "ANOther@example.com"
Entrez.efetch( db="nukleotid",
        rettype="fasta", retmode="text", id="6273291" ilə
) tutacaq kimi :
        seq_record = SeqIO.read(tutacaq, "fasta")
print("%s ilə %i xüsusiyyə tlə ri" % (seq_record.id, len(seq_record.features)))
```

Gözlə nilə n nə ticə :

```
gi |6273291 |gb |AF191665.1 |AF191665 0 xüsusiyyə tlə
```

NCBI hə mçinin faylı başqa formatlarda, xüsusə n də GenBank faylı kimi istə mə yinizə icazə verə cə k. 2009-cu il Pasxa bayramına qə də r Entrez EFetch API sizə qaytarılma növü kimi "genbank"dan istifadə etmə yə imkan verir, lakin NCBI indi EFetch for Sequence və digə r Molekulyar Biologiya Verilə nlə r Bazalarında tə svir olunduğu kimi "gb" (və ya zülallar üçün "gp") rə smi qaytarma növlə rində n istifadə etmə kdə israrlıdır. Nə ticə də , Biopython 1.50-də biz Bio.SeqIO-da "genbank" üçün lə qə b kimi "gb"-ni də stə klə yirik.

```
Bio idxal Entrez Bio idxal SeqIO


Entrez.email = "ANOther@example.com"
Entrez.efetch( db="nukleotid",
        rettype="gb", retmode="text", id="6273291" ) ilə idarə edici olaraq :

        seq_record = SeqIO.read(handle, "gb") # "genbank" üçün lə qə b kimi "gb" istifadə edir
print("%s ilə %i xüsusiyyə tlə ri" % (seq_record.id, len(seq_record.features)))
```

Bu nümunə də n gözlə nilə n nə ticə :

```
AF191665.1 3 xüsusiyyə tlə
```

Diqqə t yetirin, bu də fə üç xüsusiyyə timiz var.

İndi bir neçə rekord gə tirə k. Bu də fə tutacaqda çoxlu qeydlə r var, ona görə də Bio.SeqIO.parse() funksiyasından istifadə etmə liyik:

```
Bio idxal Entrez Bio idxal SeqIO


Entrez.email = "ANOther@example.com"
Entrez.efetch( db="nukleotid",
        rettype="gb", retmode="text", id="6273291,6273290,6273289" ilə
) tutacaq kimi :
        SeqIO.parse -də seq_record üçün (tutacaq, "gb"):
```

Machine Translated by Google

```
print("%s %s..." % (seq_record.id, seq_record.description[:50])) print( "Ardıcıllığın uzunluğu %i, %i

xüsusiyyə tlə ri, buradan: %s" % (

        len(seq_record),
        len(seq_record.features),
        seq_record.annotations["mə nbə "],
    )
)
```

Bu, aşağıdakı çıxışı vermə lidir:

```
AF191665.1 Opuntia marenae rpl16 geni; üçün xloroplast geni...
Ardıcıllığın uzunluğu 902, 3 xüsusiyyə t, aşağıdakılardan: xloroplast Opuntia marenae AF191664.1
Opuntia clavata rpl16 geni; üçün xloroplast geni...
Ardıcıllığın uzunluğu 899, 3 xüsusiyyə t: xloroplast Grusonia clavata AF191663.1 Opuntia bradtiana rpl16
geni; üçün xloroplast geni...
Ardıcıllığın uzunluğu 899, 3 xüsusiyyə t: xloroplast Opuntia bradtianaa
```

Bio.Entrez modulu haqqında ə traflı mə lumat üçün Fə sil 12-ə baxın və NCBI tə limatları haqqında oxumağınızdan ə min olun. Entrez-də n istifadə üçün (Bölmə 12.1).

5.3.2 Şə bə kə də n SwissProt ardıcıllığının tə hlili

İndi ExPASy-də n SwissProt faylını endirmə k üçün qulpdan istifadə edə k, 13-cü Fə sildə daha də rində n bə hs edilir. Yuxarıda qeyd edildiyi kimi, tutacaqda yalnız bir qeydin olmasını gözlə diyiniz zaman Bio.SeqIO.read() funksiyasından istifadə edin:

```
Bio idxaldan ExPASy Bio idxaldan
SeqIO

Də stə k kimi ExpPASy.get_sprot_raw("O23729") ilə : seq_record =
    SeqIO.read(handle, "swiss") print(seq_record.id) print
(seq_record.name)
print(seq_record.description)
print( repr(seq_record.seq)( % _ record.seq)seq)
çap(seq_record.annotations["açar sözlə r"])
```

Şə bə kə bağlantınızın yaxşı olduğunu fə rz etsə k, geri qayıtmalısınız:

```
O23729
CHS3_BROFI
RecName: Tam=Xalkon sintaza 3; EC=2.3.1.74; AltName: Tam=Naringenin-xalcone sintaza 3;
Seq('MAPAMEEIRQAQRAEGPAAVLAIGTSTPPNALYQADYPDYYFRITKSEHLTELK...GAE')
Uzunluq 394
 ['Asiltransferaza', 'Flavonoid biosintezi', 'Transferaza']
```

5.4 Faylları Lüğə t kimi sıralayın

SeqIO.parse tə rə fində n qaytarılan iterator üzə rində bir də fə dövrə vurmaq faylı tükə ndirə cə k. TwoBit formatındakı fayllar kimi öz-özünə indekslə nmiş fayllar üçün SeqIO.parse-nin qaytarılması də yə ri də lüğə t kimi istifadə oluna bilə r.

Machine Translated by Google

ardıcıllığın mə zmununa tə sadüfi giriş. Bu halda tə hlil tə lə b ə sasında hə yata keçirildiyi üçün ardıcıllıq mə lumatlarına daxil olana qə də r fayl açıq qalmalıdır:

```
>>> Bio import SeqIO- dan >>> sapı =
open("sequence.bigendian.2bit", "rb") >>> records = SeqIO.parse(handle, "twobit")
>>> records.keys() dict_keys(['seq11111 ' , 'seq222', 'seq353q'3q 'seq6'])
>>> qeydlə r["seq222"]


SeqRecord(seq=Seq('TTGATCGGTGACAAAATTTTTTACAAAGAACTGTAGGACTTGCTACTTCTCCCTC...ACA'), id='seq222', ad='< >>> qeydlə r["seq222"].seq

Seq('TTGATCGGTGACAAAATTTTTTTCAACTTCACACACAC...>>ATTACACATACACAGACT...' ) handle.close() >>>
records["seq222"].seq
Traceback (ə n son zə ng):

 ...
ValueError: ardıcıllığı ə ldə etmə k mümkün deyil: fayl bağlanıb
```

Digə r fayl formatları üçün Bio.SeqIO çox ardıcıllıqlı fayla tə sadüfi giriş kimi lüğə tə imkan verə n üç ə laqə li funksiya modulu tə qdim edir. Burada çeviklik və yaddaş istifadə si arasında bir mübadilə var. Xülasə :

- Bio.SeqIO.to_dict() ə n çevik, eyni zamanda ə n çox yaddaş tə lə b edə n seçimdir (bax. Bölmə 5.4.1). Bu, ə sasə n yaddaşda SeqRecord obyekti kimi saxlanılan hə r bir giriş ilə normal Python lüğə ti yaratmaq üçün kömə kçi funksiyadır və qeydlə ri də yişdirmə yə imkan verir.
- Bio.SeqIO.index() yalnız oxunan lüğə t kimi fə aliyyə t göstə rə n və ardıcıllığı tə hlil edə n faydalı orta yerdir. tə lə b üzrə SeqRecord obyektlə rinə (bax. Bölmə 5.4.2).
- Bio.SeqIO.index_db() hə m də yalnız oxunan lüğə t kimi fə aliyyə t göstə rir, lakin identifikatorları və fayl ofsetlə rini diskdə ki faylda (SQLite3 verilə nlə r bazası kimi) saxlayır, yə ni onun çox aşağı yaddaş tə lə blə ri var (Bölmə 5.4.3-ə baxın), lakin bir qə də r yavaş olacaq.

Geniş icmal üçün müzakirə yə baxın (Bölmə 5.4.5).

5.4.1 Faylları Lüğə t kimi sıralayın – Yaddaşda

Bizim hə r yerdə yayılmış orkide fayllarımızla edə cə yimiz növbə ti iş onları necə indekslə şdirmə yi və Python lüğə t mə lumat növündə n istifadə edə rə k verilə nlə r bazası kimi onlara daxil olmağı göstə rmə kdir (Perl-də hash kimi). Bu, yalnız faylın müə yyə n elementlə rinə daxil olmanız lazım olan orta ölçülü fayllar üçün çox faydalıdır və gözə l sürə tli və çirkli verilə nlə r bazası yaradır. Yaddaşın problemə çevrildiyi daha böyük fayllarla işlə mə k üçün aşağıdakı Bölmə 5.4.2- ə baxın .

SeqRecord lüğə tini (yaddaşda) yaratmaq üçün Bio.SeqIO.to_dict() funksiyasından istifadə edə bilə rsiniz. Defolt olaraq bu, açar kimi hə r bir qeydin identifikatorundan (yə ni .id atributundan) istifadə edə cə k. GenBank faylımızdan istifadə edə rə k bunu sınayaq:

```
>>> Bio idxaldan SeqIO >>> orchid_dict
= SeqIO.to_dict(SeqIO.parse("ls_orchid.gbk", " genbank"))
```

Bio.SeqIO.to_dict(), SeqRecord obyektlə rini verə n siyahı və ya generator üçün yalnız bir tə lə b olunan arqument var. Burada biz indicə SeqIO.parse funksiyasının çıxışından istifadə etdik. Adından da göründüyü kimi, bu Python lüğə tini qaytarır.

Bu orchid_dict də yişə ni adi bir Python lüğə ti olduğundan, ə limizdə olan bütün düymə lə rə baxa bilə rik:

```
>>> len(orchid_dict) 94
```

Machine Translated by Google

```
>>> siyahı(orchid_dict.keys())
['Z78484.1', 'Z78464.1', 'Z78455.1', 'Z78442.1', 'Z78532.1', 'Z78453.1', ..., 'Z78471.1']
```

Python 3-də ".keys()" və ".values()" kimi lüğə t üsulları siyahılardan daha çox iteratorlardır. Ə gə r hə qiqə tə n istə yirsinizsə , hə tta bütün qeydlə rə birdə n baxa bilə rsiniz:

```
>>> list(orchid_dict.values()) # çoxlu nə ticə !
```

Biz düymə lə r vasitə silə tə k SeqRecord obyektinə daxil ola və obyekti normal şə kildə idarə edə bilə rik:

```
>>> seq_record = orchid_dict["Z78475.1"] >>> çap
(seq_record.description)
P.supardii 5.8S rRNA geni və ITS1 və ITS2 DNT >>> seq_record.seq

Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGATCACAT...GGT')
```

Belə liklə , GenBank qeydlə rimizin yaddaşda "mə lumat bazası" yaratmaq çox asandır. Sonra bunun üçün cə hd edə cə yik ə və zinə FASTA faylı.

Qeyd edə k ki, Python tə crübə si olan hə r biriniz bu cür lüğə ti "ə l ilə " yarada bilmə lidir. Bununla belə , tipik lüğə t qurma üsulları tə krar düymə lə rin işini çox gözə l hə ll etmə yə cə kdir. Bio.SeqIO.to_dict() funksiyasından istifadə ikiqat açarları yoxlayacaq və aşkar edilə rsə , istisna qaldıracaq.

5.4.1.1 Lüğə t düymə lə rinin tə yin edilmə si

Yuxarıdakı ilə eyni kodu istifadə edin, lakin ə və zinə FASTA faylı üçün:

```
Bio idxaldan SeqIO

orchid_dict = SeqIO.to_dict(SeqIO.parse("ls_orchid.fasta", "fasta")) çap (orchid_dict.keys())
```

Bu də fə açarlar:

```
['gi|2765596|emb|Z78471.1|PDZ78471', 'gi|2765646|emb|Z78521.1|CCZ78521', ... ..., 'gi|2765613|emb|Z78484.1|PTZ8,'
 'gi|2765583|emb|Z78458.1|PHZ78458']
```

Siz bu sə tirlə ri Bölmə 2.4.1-də ə vvə llə r FASTA faylını tə hlil etdiyimiz vaxtdan tanımalısınız . Tutaq ki, açarlar kimi başqa bir şeyə sahib olmaq istə rdiniz - qoşulma nömrə lə ri kimi. Bu, bizi SeqIO.to_dict()-in isteğe bağlı key_funksiya arqumentinə gə tirir ki, bu da sizə qeydlə riniz üçün lüğə t açarı kimi nə də n istifadə edə cə yinizi müə yyə n etmə yə imkan verir.

Ə vvə lcə SeqRecord obyekti verildikdə istə diyiniz açarı (sə tir kimi) qaytarmaq üçün öz funksiyanızı yazmalısınız. Ümumiyyə tlə , funksiyanın tə fə rrüatları mə şğul olduğunuz giriş qeydlə rinin növündə n asılı olacaq. Amma bizim orkide üçün biz sadə cə olaraq "boru" simvolundan (şaquli xə tt) istifadə edə rə k qeydin identifikatorunu bölmə k və dördüncü girişi (üçüncü sahə) qaytara bilə rik:

```
def get_accession(record): """SeqRecord
    verildikdə , qoşulma nömrə sini sə tir kimi qaytarın.

    mə sə lə n "gi|2765613|emb|Z78488.1|PTZ78488" -> "Z78488.1"
    """

    hissə lə r = record.id.split("|") tə sdiq
    len(hissə lə r) == 5 və hissə lə ri[0] == "gi" və hissə lə ri[2] == "emb" qaytarılması hissə lə ri[3]
```

Machine Translated by Google

Sonra bu funksiyanı lüğə tin yaradılmasında istifadə etmə k üçün SeqIO.to_dict() funksiyasına verə bilə rik:

```
Bio idxaldan SeqIO

orchid_dict = SeqIO.to_dict
        ( SeqIO.parse("ls_orchid.fasta", "fasta"), key_function=get_accession

) çap (orchid_dict.keys())
```

Nə hayə t, istə diyiniz kimi, yeni lüğə t düymə lə ri:

```
>>> çap (orchid_dict.keys())
['Z78484.1', 'Z78464.1', 'Z78455.1', 'Z78442.1', 'Z78532.1', 'Z78453.1', ..., 'Z78471.1']
```

Çox mürə kkə b deyil, ümid edirə m!

5.4.1.2 SEGUID yoxlama cə mində n istifadə edə rə k lüğə tin indekslə şdirilmə si

SeqRecord obyektlə rinin lüğə tlə ri ilə işlə mə k üçün başqa bir nümunə vermə k üçün biz SEGUID checksum funksiyasından istifadə edə cə yik. Bu, nisbə tə n yeni yoxlama cə midir və toqquşmalar çox nadir olmalıdır (yə ni, eyni yoxlama cə mi ilə iki fə rqli ardıcıllıq), CRC64 yoxlama cə mində tə kmillə şdirmə .

Bir daha orkide GenBank faylı ilə işlə yin:

```
Bio idxal SeqIO - dan
Bio.SeqUtils.CheckSum idxal seguid

qeyd üçün SeqIO.parse ("ls_orchid.gbk", "genbank"): print(record.id,
        seguid(record.seq))
```

Bu vermə lidir:

```
Z78533.1 JUEoWn6DPhgZ9nAyowsgtoD9TTto
Z78532.1 MN/s0q9zDoCVEEc+k/IFwCNF2pY
...
Z78439.1 H+JfaShya/4yyAj7IbMqgNkxdxQ
```

İndi Bio.SeqIO.to_dict() funksiyasının key_funksiya arqumenti SeqRecord-u sə tirə çevirə n funksiyanı gözlə yir. Biz seguid() funksiyasından birbaşa istifadə edə bilmə rik, çünki ona Seq obyekti (və ya sə tir) verilə cə yini gözlə yir. Bununla belə , ə və zinə Bio.SeqIO.to_dict()-ə vermə k üçün "birdə fə lik" funksiyası yaratmaq üçün Python-un lambda xüsusiyyə tində n istifadə edə bilə rik:

```
>>> Bio import SeqIO >>>
Bio.SeqUtils.CheckSum idxal seguid >>> seguid_dict =
SeqIO.to_dict( SeqIO.parse("ls_orchid.gbk", "
...             genbank"), lambda rec : seguid(rec.seq)
... )
>>> rekord = seguid_dict["MN/s0q9zDoCVEEc+k/IFwCNF2pY"] >>> çap (record.id)

Z78532.1
>>> çap (rekord.tə svir)
C.californicum 5.8S rRNA geni və ITS1 və ITS2 DNT
```

Bu, fayldakı ikinci giriş olan Z78532.1 rekordunu götürmə li idi.

Machine Translated by Google

5.4.2 Faylları Lüğə tlə r kimi sıralayın – İndekslə nmiş fayllar

Ə vvə lki bir neçə misal göstə rmə yə çalışdığı kimi, Bio.SeqIO.to_dict() funksiyasından istifadə etmə k çox çevikdir. Bununla belə , hə r şeyi yaddaşda saxladığı üçün işlə yə bilə cə yiniz faylın ölçüsü kompüterinizin RAM-ı ilə mə hdudlaşır. Ümumiyyə tlə , bu, yalnız kiçik və orta ölçülü fayllarda işlə yə cə k.

Daha böyük fayllar üçün bir az fə rqli işlə yə n Bio.SeqIO.index() funksiyasını nə zə rdə n keçirmə lisiniz. O, hə lə də obyekt kimi bir lüğə t qaytarsa da, bu, hə r şeyi yaddaşda saxlamır. Bunun ə və zinə , o, sadə cə olaraq, hə r bir qeydin fayl daxilində harada olduğunu qeyd edir – siz müə yyə n bir qeyd tə lə b etdikdə , o, tə lə b ə sasında tə hlil edir.

Nümunə olaraq, ə vvə lki kimi eyni GenBank faylını istifadə edə k:

```
>>> Bio idxaldan SeqIO >>> orchid_dict =
SeqIO.index("ls_orchid.gbk", " genbank") >>> len(orchid_dict) 94



>>> orchid_dict.keys()
 ['Z78484.1', 'Z78464.1', 'Z78455.1', 'Z78442.1', 'Z78532.1', 'Z78453.1', ...,                                    'Z78471.1']

>>> seq_record = orchid_dict["Z78475.1"] >>> çap
(seq_record.description)
P.supardii 5.8S rRNA geni və ITS1 və ITS2 DNT >>> seq_record.seq

Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGATCACAT...GGT') >>>
orchid_dict.close()
```

Nə zə rə alın ki, Bio.SeqIO.index() funksiyası deyil, yalnız fayl adı olacaq. Bunun yaxşı sə bə blə ri var, lakin bu bir az texnikidir. İkinci arqument fayl formatıdır (digə r Bio.SeqIO funksiyalarında istifadə edilə n kiçik hə rf sə tri). Siz FASTA və FASTQ faylları daxil olmaqla bir çox digə r sadə fayl formatlarından istifadə edə bilə rsiniz (Bölmə 22.1.11-də ki nümunə yə baxın). Bununla belə , PHYLIP və ya Clustal kimi hizalama formatları də stə klə nmir.
Nə hayə t, isteğe bağlı arqument olaraq siz ə sas funksiyanı tə qdim edə bilə rsiniz.

Budur, FASTA faylından istifadə edə n eyni nümunə - də yişdirdiyimiz yalnız fayl adı və format adıdır:

```
>>> Bio idxaldan SeqIO >>>
orchid_dict = SeqIO.index("ls_orchid.fasta", " fasta") >>> len(orchid_dict)

94
>>> orchid_dict.keys() ['gi |
 2765596 |emb |Z78471.1 |PDZ78471', 'gi |2765646 |emb |Z78521.1 |CCZ78521', ... ..., 'gi |2765613 |2765613 |emb |
   'gi |2765583 |emb |Z78458.1 |PHZ78458']
```

5.4.2.1 Lüğə t düymə lə rinin tə yin edilmə si

Tutaq ki, siz ə vvə lki kimi eyni düymə lə rdə n istifadə etmə k istə yirsiniz? Bölmə 5.4.1.1-də ki Bio.SeqIO.to_dict() misalında olduğu kimi , siz FASTA identifikatorundan (sə tir kimi) istə diyiniz açara xə ritə çə kmə k üçün kiçik funksiya yazmalısınız:

```
def get_acc(identifikator):
        """SeqRecord identifikator sə trini nə zə rə alaraq, qoşulma nömrə sini sə tir kimi qaytarın.

        mə sə lə n "gi |2765613 |emb |Z78488.1 |PTZ78488" -> "Z78488.1"

        """

        hissə lə r = identifikator.split("|") tə sdiq
        len(hissə lə r) == 5 və hissə lə ri[0] == "gi" və hissə lə ri[2] == "emb" qaytarılması hissə lə ri[3]
```

Machine Translated by Google

Sonra bu funksiyanı lüğə tin yaradılmasında istifadə etmə k üçün Bio.SeqIO.index() funksiyasına verə bilə rik:

```
>>> Bio idxaldan SeqIO >>>
orchid_dict = SeqIO.index("ls_orchid.fasta", " fasta", key_function=get_acc) >>> çap (orchid_dict.keys())

['Z78484.1', 'Z78464.1', 'Z78455.1', 'Z78442.1', 'Z78532.1', 'Z78453.1', ..., 'Z78471.1']
```

Necə olduğunu bilə ndə asan?

5.4.2.2 Qeyd üçün ilkin mə lumatların ə ldə edilmə si

Bio.SeqIO.index()-də n olan lüğə tə bə nzə r obyekt sizə hə r bir girişi SeqRecord obyekti kimi verir. Bununla belə , bə zə n orijinal xam mə lumatları birbaşa fayldan ə ldə etmə k faydalıdır. Bunun üçün tə k arqument (qeyd identifikatoru) götürə n və bayt sə tirini qaytaran get_raw() metodundan istifadə edin (də yişiklik olmadan fayldan çıxarılıb).

Hə və slə ndirici misal, Bio.SeqIO.write() funksiyasının (hə lə) çıxış fayl formatını də stə klə mə diyi (mə sə lə n, adi mə tn SwissProt fayl formatı) və ya mə tni tam olaraq qorumağınız lazım olduğu (mə sə lə n, GenBank və ya Biopython-dan EMBL çıxışı hə lə annotasiyanın hə r son bitini saxlamayan) böyük fayldan qeydlə rin alt də stinin çıxarılmasıdır.

Tutaq ki, siz bütün UniProt-u FTP saytından (ftp://ftp.uniprot.org/pub/databases/uniprot/current_release/knowledgebase/complete/uniprot_sprot.dat.gz) düz mə tn SwissPort fayl formatında endirmisiniz. və onu uniprot_sprot.dat faylı kimi sıxışdırmayın və siz ondan yalnız bir neçə qeyd çıxarmaq istə yirsiniz:

```
>>> Bio import SeqIO -dan >>>
uniprot = SeqIO.index("uniprot_sprot.dat", "swiss") >>> ilə out_handle kimi
open("selected.dat", "wb") ilə :
...         acc üçün ["P33487", "P19801", " P13689 ", "Q8JZQ5", "Q9TRC7"]:
...             out_handle.write(uniprot.get_raw(acc))
...
```

Qeyd edə k ki, Python 3-də n sonra biz faylı binar rejimdə yazmaq üçün açmalıyıq, çünki get_raw() metodu bayt sə tirlə rini qaytarır.

Bölmə 22.1.5-də böyük ardıcıllıq faylını çeşidlə mə k üçün SeqIO.index() funksiyasından istifadə edə n daha uzun bir nümunə var (hə r şeyi bir anda yaddaşa yüklə mə də n).

5.4.3 Faylları Lüğə tlə r kimi sıralayın – Verilə nlə r bazası indeksli fayllar

Biopython 1.57 alternativi Bio.SeqIO.index_db() tə qdim etdi, o, hə tta çox böyük fayllarda işlə yə bilir, çünki o, qeyd mə lumatını yaddaşda deyil, diskdə fayl kimi saxlayır (SQLite3 verilə nlə r bazasından istifadə etmə klə). Hə mçinin, siz bir neçə faylı birlikdə indekslə şdirə bilə rsiniz (bütün qeyd identifikatorlarının unikal olmasını tə min etmə klə).

Bio.SeqIO.index() funksiyası üç tə lə b olunan arqument alır:

- İndeks fayl adı, biz .idx ilə bitə n bir şeydə n istifadə etmə yi tə klif edirik. Bu indeks faylı ə slində SQLite3-dir verilə nlə r bazası.
- İndekslə nə cə k fayl adlarının siyahısı (və ya tə k fayl adı)
- Fayl formatı (SeqIO modulunun qalan hissə sində istifadə edilə n kiçik hə rf sə tri).

Nümunə olaraq, NCBI FTP saytından, ftp://ftp.ncbi.nih-də n GenBank düz fayl buraxılışlarını nə zə rdə n keçirin. gov/genbank/, gzip sıxılmış GenBank fayllarıdır.

GenBank-ın 210-cu buraxılışına ə sasə n, viral ardıcıllığı tə şkil edə n 38 fayl var, gbvrl1.seq, . . . , gbvrl38.seq, bir də fə sıxılmış diskdə tə xminə n 8 GB yer tutur və cə mi iki milyona yaxın qeyddə n ibarə tdir.

Ə gə r viruslarla maraqlanırsınızsa, rsync ə mri ilə bütün virus fayllarını komanda xə ttində n çox asanlıqla yüklə yə və sonra onları gunzip ilə aça bilə rsiniz:

Machine Translated by Google

```
# Yalnız tə svir üçün, aşağı salınmış nümunə yə baxın $ rsync -avP
"ftp.ncbi.nih.gov::genbank/gbvrl*.seq.gz" . $ gunzip gbvrl*.seq.gz
```

Viruslarla maraqlanmırsınızsa, bu, yalnız bu nümunə üçün yüklə mə k üçün çoxlu mə lumatdır - buna görə də gə lin yalnız ilk dörd parçanı (hə r biri sıxılmış tə xminə n 25 MB) endirə k və onları açaq (tə xminə n 1 GB yer tutaraq):

```
# Azaldılmış nümunə , yalnız ilk dörd parçanı endirin $ curl -O ftp://ftp.ncbi.nih.gov/
genbank/gbvrl1.seq.gz $ curl -O ftp://ftp.ncbi.nih.gov/genbank/gbvrl2.seq.gz $ curl -O
ftp://ftp.ncbi.nih.gov/genbank/gbvrl3.seq.gz $ curl -O ftp://ftp.ncbi.nih.gov/genbank/
gbvrl4.seq.gz $ gunzip gbvrl*.seq.gz
```

İndi Python-da bu GenBank fayllarını aşağıdakı kimi indekslə şdirin:

```
>>> import glob >>> Bio
import SeqIO >>> files = glob.glob("gbvrl*.seq")
>>> print ("%i faylları indeksə " % len(fayllar)) 4 >>> gb_vrl =
SeqIO.index_db("gbvrl.idx", fayllar, "genbank") >>> " qu print(%i ) len(gb_vrl))

272960 ardıcıllıq indekslə nmişdir
```

Virus GenBank fayllarının tam də stinin indekslə şdirilmə si mə nim maşınımda tə xminə n on də qiqə çə kdi, yalnız ilk dörd fayllar tə xminə n bir də qiqə çə kdi.

Bununla belə , bir də fə bunu tə krarlamaq gbvrl.idx indeks faylını saniyə nin bir hissə sində yenidə n yüklə yə cə k.

Siz indeksi yalnız oxunan Python lüğə ti kimi istifadə edə bilə rsiniz - hansı faylın olması barə də narahat olmadan ardıcıllıq gə lir, mə sə lə n

```
>>> çap(gb_vrl["AB811634.1"].tə svir)
At ensefalozu virusu NS3 geni, tam CD, tə crid: Kimron1.
```

5.4.3.1 Qeyd üçün ilkin mə lumatların ə ldə edilmə si

Bölmə 5.4.2.2-də yuxarıda müzakirə edilə n Bio.SeqIO.index() funksiyasında olduğu kimi , lüğə tə bə nzə r obyekt də sizə hə r qeydin xam baytlarını ə ldə etmə yə imkan verir:

```
>>> çap (gb_vrl.get_raw("AB811634.1"))
LOCUS AB811634 723 bp TƏ SVİRİ At ensefalozu virusu NS3 geni, tam CD, tə crid: Kimron1. RNT xə tti VRL 17-İYUN-2015

QOŞULMA AB811634
 ...
//
```

5.4.4 Sıxılmış faylların indekslə şdirilmə si

Çox tez-tez bir ardıcıllıq faylını indekslə şdirə rkə n, o, olduqca böyük ola bilə r - buna görə də onu diskdə sıxmaq istə yə bilə rsiniz. Tə ə ssüf ki, sə mə rə li tə sadüfi giriş gzip və bzip2 kimi daha çox yayılmış fayl formatları ilə çə tindir. Bu parametrdə BGZF (Bloklanmış GNU Zip Format) çox faydalı ola bilə r. Bu, BAM fayl formatı, samtools ilə populyarlaşan gzip variantıdır (və standart gzip alə tlə rində n istifadə etmə klə açıla bilə r) . və tabix.

Machine Translated by Google

BGZF sıxılmış fayl yaratmaq üçün samtools ilə birlikdə gə lə n bgzip ə mr xə tti alə tində n istifadə edə bilə rsiniz. Nümunə lə rimizdə fayl adı uzantısından *.bgz istifadə edirik, ona görə də onları normal gziplə nmiş fayllardan (*.gz adlı) fə rqlə ndirmə k olar. Siz hə mçinin Python daxilində n BGZF fayllarını oxumaq və yazmaq üçün Bio.bgzf modulundan istifadə edə bilə rsiniz.

Bio.SeqIO.index() və Bio.SeqIO.index_db() hə r ikisi BGZF sıxılmış faylları ilə istifadə edilə bilə r. Mə sə lə n, sıxılmamış GenBank faylı ilə başlamışsınızsa:

```
>>> Bio idxaldan SeqIO >>>
orchid_dict = SeqIO.index("ls_orchid.gbk", " genbank") >>> len(orchid_dict)

94
>>> orchid_dict.close()
```

Bunu (orijinal faylı saxlayarkə n) aşağıdakı ə mrdə n istifadə edə rə k ə mr satırında sıxışdıra bilə rsiniz – lakin narahat olmayın, sıxılmış fayl artıq digə r nümunə faylları ilə birlikdə dir:

```
$ bgzip -c ls_orchid.gbk > ls_orchid.gbk.bgz
```

Sıxılmış faylı eyni şə kildə istifadə edə bilə rsiniz:

```
>>> Bio import SeqIO >>> orchid_dict =
SeqIO.index("ls_orchid.gbk.bgz", "genbank") >>> len(orchid_dict) 94 >>> orchid_dict.close ()
```

və ya:

```
>>> Bio import SeqIO >>> orchid_dict
= SeqIO.index_db("ls_orchid.gbk.bgz.idx", "ls_orchid.gbk.bgz", "genbank") >>> len ( orchid_dict)

94
>>> orchid_dict.close()
```

SeqIO indekslə mə avtomatik olaraq BGZF sıxılmasını aşkar edir. Eynisini istifadə edə bilmə yə cə yinizi unutmayın sıxılmamış və sıxılmış fayllar üçün indeks faylı.

5.4.5 Müzakirə

Yaxşı, bu üsullardan hansını istifadə etmə lisiniz və niyə ? Bu, nə etmə yə çalışdığınızdan (və nə qə də r mə lumatla mə şğul olduğunuzdan) asılıdır. Bununla belə , ümumi olaraq Bio.SeqIO.index() seçimi yaxşı başlanğıc nöqtə sidir. Ə gə r siz milyonlarla qeyd, çoxlu fayl və ya tə krar analizlə rlə mə şğul olursunuzsa, Bio.SeqIO.index_db() saytına baxın.

Bio.SeqIO.to_dict()-i Bio.SeqIO.index() və ya Bio.SeqIO.index_db() ə və zinə seçmə k üçün sə bə blə r onun yüksə k yaddaş ehtiyaclarına baxmayaraq, çeviklik ehtiyacına çevrilir. SeqRecord obyektlə rinin yaddaşda saxlanmasının üstünlüyü ondan ibarə tdir ki, onlar öz istə yi ilə də yişdirilə , ə lavə oluna və ya silinə bilə r. Yüksə k yaddaş istehlakının mə nfi tə rə finə ə lavə olaraq, indekslə şdirmə daha uzun çə kə bilə r, çünki bütün qeydlə r tam tə hlil edilmə lidir.

Hə m Bio.SeqIO.index(), hə m də Bio.SeqIO.index_db() yalnız tə lə b ə sasında qeydlə ri tə hlil edir. İndekslə mə zamanı onlar hə r bir qeydin başlanğıcını axtarırlar və identifikatoru çıxarmaq üçün mümkün qə də r az iş görürlə r.

Bio.SeqIO.index_db() ə və zinə Bio.SeqIO.index() seçmə yin sə bə blə ri bunlardır:

- İndeksi daha sürə tli qurmaq (sadə fayl formatlarında daha nə zə rə çarpır)
- SeqRecord obyektlə ri kimi bir az daha sürə tli giriş (lakin fə rq yalnız sadə üçün nə zə rə çarpır fayl formatlarını tə hlil edin).

Machine Translated by Google

- Lüğə t açarları kimi hə r hansı də yişmə z Python obyektində n istifadə edə bilə r (mə sə lə n, sə tirlə r də sti və ya dondurulmuş də st) tə kcə simlə r deyil.
- Ə gə r indekslə şdirilə n ardıcıllıq faylı varsa, indeks verilə nlə r bazasının köhnə lmə sində n narahat olmaq lazım deyil. də yişdi.

Bio.SeqIO.index_db() funksiyasını Bio.SeqIO.index() ə və zinə seçmə k üçün sə bə blə rə aşağıdakılar daxildir:

- Yaddaş mə hdud deyil – bu, 10 milyonlarla ardıcıllığın ümumi olduğu ikinci nə sil ardıcıllıq faylları üçün artıq vacibdir və Bio.SeqIO.index() istifadə etmə k 4GB-dan çox RAM və buna görə də Python-un 64bit versiyası tə lə b edə bilə r.
- İndeks diskdə saxlandığı üçün onu tə krar istifadə etmə k olar. İndeks verilə nlə r bazası faylının yaradılması daha uzun vaxt tə lə b etsə də , gə lə cə kdə eyni mə lumat fayllarında yenidə n işlə nə cə k bir skriptiniz varsa, bu, uzun müddə tdə vaxta qə naə t edə bilə r.
- Çoxlu faylların birlikdə indekslə şdirilmə si
- Get_raw() metodu daha sürə tli ola bilə r, çünki ə ksə r fayl formatları üçün hə r bir qeydin uzunluğudur hə m də onun ofsetini saxlayır.

5.5 Ardıcıllıqla faylların yazılması

Ardıcıllıqla daxiletmə (faylların oxunması) üçün Bio.SeqIO.parse() funksiyasından istifadə haqqında danışdıq və indi ardıcıllıqla çıxış (faylların yazılması) üçün olan Bio.SeqIO.write()-ə baxacağıq. Bu, üç arqument götürə n funksiyadır: bə zi SeqRecord obyektlə ri, yazmaq üçün tutacaq və ya fayl adı və ardıcıllıq formatı.

Budur, çə tin bir şə kildə bir neçə SeqRecord obyekti yaratmaqla başladığımız bir nümunə (daha doğrusu ə l ilə onları fayldan yüklə mə kdə nsə):

```
Bio.Seq -də n idxal Seq- də n
Bio.SeqRecord- dan SeqRecord idxalı

rec1 = SeqRecord(
    Seq(
        "MMYQQGCFAGGTVLRLAKDLAENNRGARVLVVCSEITAVTFRGPSETHLDSMVGQALFGD"
        "GAGAVIVGSDPDLSVERPLYELVWTGATLPDSEGAIDGHLREVGLTFHLLKDVPGLISK"
        "NIEKSLKEAFTPLGISDWNSTFWIAHPGGPAILDQVEAKLGLKEEKMRATREVLSEYGNM"
        "TQDK",
    ),
    id="gi|14150838|gb|AAK54648.1|AF376133_1", description="xalkon
    sintaza [Cucumis sativus]",
)

rec2 = SeqRecord(
    Seq(
        "YPDYYFRITNREHKAELKEKFQRMCDKSMIKKRYMYLTEEILKENPSMCEYMAPSLDARQ"
        "DMVVVEIPKLGKEAAVKAIKEWGQ",
    ),
    id="gi|13919613|gb|AAK33142.1|",
    description="xalkon sintaza [Fragaria vesca subsp. bracteata]",
)

rec3 = SeqRecord(
```

Machine Translated by Google

```
        Seq(
            "MVTVEEFRRAQCAEGPATVMAIGTATPSNCVDQSTYPDYYFRITNSEHKVELKEKFKRMC"
            "EKSMIKKRYMHLTEEILKENPNICAYMAPSLDARQDIVVVEVPKLGKEAAQKAIKEWGQP"
            "KSKITHLVFCTTSGVDMPGCDYQLTKLLGLRPSVKRFMMYQQGCFAGGTVLRMAKDLAEN"
            "NKGARVLVVCSEITAVTFRGPNDTHLDSLVGQALFGDGAAAVIIGSDPIPEVERPLFELV"
            "SAAQTLLPDSEGAIDGHLREVGLTFHLLKDVPGLISKNIEKSLVEAFQPLGISDWNSLFW"
            "IAHPGGPAILDQVELKLGLKQEKLKATRKVLSNYGNMSSACVLFILDEMRKASAKEGLGT"
            "TGEGLEWGVLFGFGPGLTVETVVLHSVAT",
        ),
        id="gi|13925890|gb|AAK49457.1|",
        description="xalkon sintaza [Nicotiana tabacum]",
)

my_records = [rec1, rec2, rec3]
```

İndi bizdə SeqRecord obyektlə rinin siyahısı var, biz onları FASTA format faylına yazacağıq:

```
Bio idxaldan SeqIO

SeqIO.write(mə nim_qeydlə rim, "my_example.faa", "fasta")
```

Və bu faylı sevimli mə tn redaktorunuzda açsanız, bu belə görünmə lidir:

```
>gi|14150838|gb|AAK54648.1|AF376133_1 xalkon sintaza [Cucumis sativus]
MMYQQGCFAGGTVLRLAKDLAENNRGARVLVVCSEITAVTFRGPSETHLDSMVGQALFGD
GAGAVIVGSDPDLSVERPLYELVWTGATLLPDSEGAIDGHLREVGLTFHLLKDVPGLISK
NIEKSLKEAFTPLGISDWNSTFWIVEAHPGLGISDWNSTFWIVEAHPGLMGNG TQDK >gi|13919613|gb|AAK33142.1|
xalkon
sintaza [Fragaria vesca subsp. bracteata]
YPDYYFRITNREHKAELKEKFQRMCDKSMIKKRYMYLTEEILKENPSMCEYMAPSLDARQ
DMVVVEIPKLGKEAAVKAIKEWGQ
>gi|13925890|gb|AAK49457.1| xalkon sintaza [Nicotiana tabacum]
MVTVEEFRRAQCAEGPATVMAIGTATPSNCVDQSTYPDYYFRITNSEHKVELKEKFKRMC
EKSMIKKRYMHLTEEILKENPNICAYMAPSLDARQDIVVVEVPKLGKEAAQKAIKEWGQP
KSKITHLVFCTTSGVDMPGCDYQLTKLLGLRPSVKRFMMYQQGCFAGGTVLRMAKDLAEN
NKGARVLVVCSEITAVTFRGPNDTHLDSLVGQALFGDGAAAVIIGSDPIPEVERPLFELV
SAAQTLLPDSEGAIDGHLREVSLVESDWLTFFNSKHLLK
IAHPGGPAILDQVELKLGLKQEKLKATRKVLSNYGNMSSACVLFILDEMRKASAKEGLGT
TGEGLEWGVLFGFGPGLTVETVVLHSVAT
```

Tutaq ki, siz Bio.SeqIO.write() funksiyasının sapa neçə qeyd yazdığını bilmə k istə yirsiniz? Ə gə r qeydlə riniz siyahıda olsaydı, sadə cə olaraq len(my_records) istifadə edə bilə rsiniz, lakin qeydlə riniz generatordan/iteratordan gə lə ndə bunu edə bilmə zsiniz. Bio.SeqIO.write() funksiyası fayla yazılmış SeqRecord obyektlə rinin sayını qaytarır.

Qeyd - Bio.SeqIO.write() funksiyasına artıq mövcud olan fayla yazmağı ə mr etsə niz, köhnə fayl heç bir xə bə rdarlıq edilmə də n üzə rinə yazıla bilə r.

5.5.1 Dairə vi sə fə rlə r

Bə zi insanlar öz analizatorlarının "dönə -dönüşlü" olmasını xoşlayırlar, yə ni faylda oxuyub yenidə n yazsanız, o, də yişmə z qalır. Bu, tə hlilçinin orijinal faylı tam olaraq tə krar etmə k üçün kifayə t qə də r mə lumat çıxarmasını tə lə b edir. Bio.SeqIO bunu etmə k mə qsə di daşımır.

Önə msiz bir misal olaraq, FASTA fayllarında ardıcıllıq mə lumatlarının hə r hansı bir sə tir sarılmasına icazə verilir. Eyni SeqRecord yalnız sə tir fasilə lə rində fə rqlə nə n aşağıdakı iki nümunə nin tə hlilində n verilə cə k:

Machine Translated by Google

>YAL068C-7235.2170 Tə xmin edilə n promotor ardıcıllığı
TACGAGAATAATTTCTCATCATCCAGCTTTAACACAAAATTCGCACAGTTTTCGTTAAGA
GAACTTAACATTTTCTTATGACGTAAATGAAGTTTATATATAAATTTCCTTTTTATTGGA

>YAL068C-7235.2170 Tə xmin edilə n promotor ardıcıllığı
TACGAGAATAATTTCTCATCATCCAGCTTTAACACAAAATTCGCA
CAGTTTTCGTTAAGAGAACTTAACATTTTCTTATGACGTAAATGA
AGTTTATATATAAATTTCCTTTTTTTTGGA

Dairə vi-dönüşlü FASTA analizatoru yaratmaq üçün siz ardıcıllıq xə tti qırılmalarının harada baş verdiyini izlə mə lisiniz və bu ə lavə mə lumat adə tə n mə nasızdır. Bunun ə və zinə Biopython çıxışda 60 simvoldan ibarə t standart sə tirdə n istifadə edir. Ağ boşluqla bağlı eyni problem bir çox digə r fayl formatlarına da aiddir. Bə zi hallarda başqa bir mə sə lə odur ki, Biopython (hə lə) annotasiyaların hə r bir hissə sini (mə sə lə n, GenBank və EMBL) qoruyub saxlamır.

Bə zə n orijinal tə rtibatı qorumaq (var ola bilə cə k qə ribə liklə rlə) vacibdir. Bölmə 5.4.2.2- yə baxın bir potensial hə ll üçün Bio.SeqIO.index() lüğə tə bə nzə r obyektin get_raw() metodu haqqında.

5.5.2 Ardıcıl fayl formatları arasında konvertasiya

Ə vvə lki nümunə də Bio.SeqIO.write() funksiyasına giriş kimi SeqRecord obyektlə rinin siyahısını istifadə etdik, lakin o, Bio.SeqIO.parse()-də n ə ldə etdiyimiz kimi SeqRecord iteratorunu da qə bul edə cə k – bu, bizə bu iki funksiyanı birlə şdirə rə k faylın çevrilmə sini hə yata keçirmə yə imkan verir.

Bu nümunə üçün GenBank formatında ls orchid.gbk faylında oxuyacağıq və FASTA formatında yazın:

```
Bio idxaldan SeqIO

qeydlə r = SeqIO.parse("ls_orchid.gbk", "genbank") sayı =
SeqIO.write(qeydlə r, "my_example.fasta", "fasta") print("Çevrilmiş %i qeydlə r" %
sayı)
```

Yenə də bu bir az mürə kkə bdir. Belə liklə , faylın çevrilmə si ümumi bir iş olduğundan, bir kömə kçi var funksiyası onu sadə cə olaraq ə və z etmə yə imkan verir:

```
Bio idxaldan SeqIO

count = SeqIO.convert("ls_orchid.gbk", "genbank", "my_example.fasta", "fasta") print("Çevrilmiş %i qeydlə ri" %
sayı)
```

Bio.SeqIO.convert() funksiyası tutacaqlar və ya fayl adları alacaq. Baxmayaraq ki, diqqə tli olun – çıxış varsa fayl artıq mövcuddur, onun üzə rinə yazacaq! Ə traflı mə lumat üçün daxili yardıma baxın:

```
>>> Bio import SeqIO- dan >>>
kömə k (SeqIO.convert)
```

Prinsipcə , sadə cə fayl adlarını və format adlarını də yişdirmə klə , bu kod Biopython-da mövcud olan istə nilə n fayl formatları arasında çevirmə k üçün istifadə edilə bilə r. Bununla belə , bə zi formatların yazılması digə r fayl formatlarında olmayan mə lumat (mə sə lə n, keyfiyyə t balları) tə lə b edir. Mə sə lə n, FASTQ faylını FASTA faylına çevirə bildiyiniz halda, bunun ə ksini edə bilmə zsiniz. Müxtə lif FASTQ formatları arasında konvertasiyaya baxan yemə klə r kitabının 22.1.9 və 22.1.10-cu bölmə lə rinə də baxın .

Nə hayə t, Bio.SeqIO.convert() funksiyasından istifadə etmə k üçün ə lavə stimul olaraq (kodunuz daha qısa olacaq), bunu bu şə kildə etmə k daha sürə tli ola bilə r! Bunun sə bə bi, çevirmə funksiyasının bir neçə fayl formatına xüsusi optimallaşdırma və tövsiyə lə rdə n istifadə edə bilmə sidir.

Machine Translated by Google

5.5.3 Ardıcıllıq faylının ə ks tamamlayıcılara çevrilmə si

Tutaq ki, sizdə nukleotid ardıcıllığı faylı var və siz onu onların ə ksini ehtiva edə n fayla çevirmə k istə yirsiniz. tamamlayan ardıcıllıqlar. Bu də fə ə ldə etdiyimiz SeqRecord obyektlə rini çevirmə k üçün bir az iş tə lə b olunur giriş faylımızdan çıxış faylımıza saxlamaq üçün uyğun bir şeyə çevirin.

Başlamaq üçün fayldan bə zi nukleotid ardıcıllığını yüklə mə k üçün Bio.SeqIO.parse() funksiyasından istifadə edə cə yik, sonra çap edə cə yik. Seq obyektinin quraşdırılmış .reverse_complement() metodundan istifadə edə rə k onların tə rs tamamlamaları (bax: Bölmə 3.6):

```
>>> Bio idxal SeqIO- dan
>>> qeyd üçün SeqIO.parse ("ls_orchid.gbk", "genbank"):
...         çap (record.id)
...         çap (record.seq.reverse_complement())
...
```

İndi bu tə rs tamamlayıcıları faylda saxlamaq istə yiriksə , SeqRecord obyektlə rini yaratmalıyıq. Biz SeqRecord obyektinin quraşdırılmış .reverse_complement() metodundan istifadə edə bilə r (Bölmə 4.9-a baxın) , lakin biz bunu etmə liyik yeni rekordlarımızı necə adlandıracağımıza qə rar verin.

Bu, yaddaşda bir siyahı yaradan siyahı anlamalarının gücünü nümayiş etdirmə k üçün ə la yerdir:

```
>>> Bio idxal SeqIO- dan
>>> qeydlə r = [
...         rec.reverse_complement(id="rc_" + rec.id, tə sviri="ə ks tamamlayıcı")
...         Rec üçün SeqIO.parse ("ls_orchid.fasta", "fasta")
... ]
>>> len (rekordlar)
94
```

İndi siyahı anlayışlarının qolları qə də r gözə l hiylə var, şə rti bir ifadə ə lavə edə bilə rsiniz:

```
>>> qeydlə r = [
...         rec.reverse_complement(id="rc_" + rec.id, tə sviri="ə ks tamamlayıcı")
...         Rec üçün SeqIO.parse ("ls_orchid.fasta", "fasta")
...         ə gə r len(rec) < 700
... ]
>>> len (rekordlar)
18
```

Bu, ardıcıllığın uzunluğunun altında olduğu ə ks tamamlayıcı qeydlə rin yaddaşda siyahısını yaradacaq 700 ə sas cüt. Bununla belə , generator ifadə si ilə eyni şeyi edə bilə rik - lakin üstünlüyü ilə bu, bir anda yaddaşdakı bütün qeydlə rin siyahısını yaratmır:

```
>>> qeydlə r = (
...         rec.reverse_complement(id="rc_" + rec.id, tə sviri="ə ks tamamlayıcı")
...         Rec üçün SeqIO.parse ("ls_orchid.fasta", "fasta")
...         ə gə r len(rec) < 700
... )
```

Tam nümunə olaraq:

```
>>> Bio idxal SeqIO- dan
>>> qeydlə r = (
...         rec.reverse_complement(id="rc_" + rec.id, tə sviri="ə ks tamamlayıcı")
...         Rec üçün SeqIO.parse ("ls_orchid.fasta", "fasta")
...         ə gə r len(rec) < 700
```

Machine Translated by Google

```
... )
>>> SeqIO.write(qeydlə r, "rev_comp.fasta", "fasta") 18
```

Bölmə 22.1.3- də FASTA faylındakı hə r bir qeydi nukleotidlə rdə n amin turşularına çevirə n müvafiq nümunə var .

5.5.4 SeqRecord obyektlə rinizi formatlanmış sə tirlə r kimi ə ldə etmə k

Tutaq ki, siz hə qiqə tə n qeydlə rinizi bir fayla və ya idarə yə yazmaq istə mirsiniz – bunun ə və zinə müə yyə n fayl formatında qeydlə ri ehtiva edə n sə tir istə yirsiniz. Bio.SeqIO interfeysi tutacaqlara ə saslanır, lakin Python simli ə saslı tutacaq tə min edə n faydalı daxili modula malikdir.

Bunu necə istifadə edə bilə cə yiniz nümunə si üçün gə lin orkidelə rimizdə n bir də stə SeqRecord obyektini yüklə yə k. GenBank faylını açın və FASTA formatında qeydlə ri ehtiva edə n sə tir yaradın:

```
Bio idxalı SeqIO- dan io idxalı
StringIO- dan

qeydlə r = SeqIO.parse("ls_orchid.gbk", "genbank") out_handle = StringIO()

SeqIO.write(qeydlə r, out_handle, "fasta") fasta_data =
out_handle.getvalue() print(fasta_data)
```

İlk də fə gördüyünüz zaman bu, tamamilə sadə deyil! Müə yyə n bir fayl formatında bir qeyddə n ibarə t sə tir istə diyiniz xüsusi hal üçün, SeqRecord sinifinin format() metodundan istifadə edin (Bölmə 4.6-a baxın).

Qeyd edə k ki, biz bunu tə şviq etmə sə k də , fayla yazmaq üçün format() metodundan istifadə edə bilə rsiniz, mə sə lə n, bu kimi bir şey:

```
Bio idxaldan SeqIO

out_handle kimi open("ls_orchid_long.tab", "w") ilə :
    qeyd üçün SeqIO.parse ("ls_orchid.gbk", "genbank"): ə gə r len(record) > 100:

                out_handle.write(record.format("tab"))
```

Bu kod üslubu FASTA kimi sadə ardıcıl fayl formatı və ya burada istifadə olunan sadə nişanla ayrılmış format üçün işlə sə də , daha mürə kkə b və ya bir-birinə qarışmış fayl formatları üçün işlə mə yə cə k. Buna görə də biz hə lə də aşağıdakı nümunə də olduğu kimi Bio.SeqIO.write() istifadə etmə yi tövsiyə edirik:

```
Bio idxaldan SeqIO

qeydlə r = (SeqIO.parse("ls_orchid.gbk", "genbank")-da qeyd üçün qeyd , ə gə r len ( rec ) > 100)
SeqIO.write(qeydlə r, "ls_orchid.tab", "tab")
```

SeqIO.write(...)-a tə k zə ng etmə k SeqRecord.format(...) metoduna çoxsaylı zə nglə rdə n daha tezdir.

5.6 Aşağı sə viyyə li FASTA və FASTQ analizatorları

Aşağı sə viyyə li SimpleFastaParser və ya FastqGeneralIterator ilə işlə mə k tez-tez Bio.SeqIO.parse-də n daha praktikdir, burada sürə ti yüksə k olan böyük yüksə k mə hsuldarlıqlı FASTA və ya FASTQ ardıcıllıq faylları ilə mə şğul olursunuz.

Machine Translated by Google

mə sə lə lə r. Bu fə slin girişində qeyd olunduğu kimi, fayl formatlı neytral Bio.SeqIO interfeysi hə tta FASTA kimi sadə formatlar üçün də çoxlu obyektlə rin yaradılmasına ə lavə yük daşıyır.

FASTA fayllarını tə hlil edə rkə n, daxildə Bio.SeqIO.parse() fayl də stə yi ilə aşağı sə viyyə li SimpleFastaParser-i çağırır. Siz bundan birbaşa istifadə edə bilə rsiniz - o, hə r bir qeydi iki sə tir, başlıq sə tiri (> simvolundan sonra hə r şey) və ardıcıllığı (düz sə tir kimi) kimi qaytaran fayl sapı üzə rində tə krarlanır:

```
>>> Bio.SeqIO.FastaIO- dan SimpleFastaParser idxalı >>> sayı = 0

>>> total_len = 0 >>> ilə
open("ls_orchid.fasta") in_handle :
...         başlıq üçün SimpleFastaParser(in_handle) -də ardıcıllıq: += 1 sayın
...
...                 total_len += len(seq)
...
>>> print("%i ümumi ardıcıllıq uzunluğu olan qeydlə r %i" % (count, total_len)) Ümumi ardıcıllıq uzunluğu ilə 94 qeyd
67518
```

Nə qə də r ki, sə tirlə rin sarılması ilə maraqlanmırsınız (və çox güman ki, siz yüksə k ötürmə qabiliyyə tinə malik mə lumatları qısa oxumaq istə mirsiniz), onda bu sə tirlə rdə n FASTA formatının çıxarılması da çox sürə tlidir:

```
...
out_handle.write(">%s\n%s\n" % (başlıq, ardıcıllıq))
...
```

Eyni şə kildə , FASTQ fayllarını tə hlil edə rkə n, daxildə Bio.SeqIO.parse() aşağı sə viyyə li FastqGeneralIterator-u çağırır. fayl də stə yi ilə . Ə gə r tam ə də dlə rə çevrilmiş keyfiyyə t ballarına ehtiyacınız yoxdursa və ya onlarla ASCII sə tirlə ri kimi işlə yə bilirsinizsə , bu idealdır:

```
>>> Bio.SeqIO.QualityIO idxalı FastqGeneralIterator >>> say = 0

>>> total_len = 0 >>> ilə
open("example.fastq") in_handle kimi : başlıq , seq, qual üçün
...         FastqGeneralIterator(in_handle): say += 1
...
...                 total_len += len(seq)
...
>>> print("%i ümumi ardıcıllıq uzunluğu olan qeydlə r %i" % (count, total_len)) Ümumi ardıcıllıq uzunluğu 75 olan 3
qeyd
```

Yemə k Kitabında (Fə sil 22), FASTQ-nun necə sə mə rə li şə kildə çıxarılması da daxil olmaqla, buna dair daha çox nümunə var bu kod parçasını istifadə edə rə k sə tirlə rdə n:

```
...
out_handle.write("@%s\n%s\n+\n%s\n" % (başlıq, ardıcıllıq, keyfiyyə t))
...
```

Machine Translated by Google

Fə sil 6

Ardıcıl düzülmə lə r

Ardıcıl düzülmə lə r bir-birinə uyğunlaşdırılmış iki və ya daha çox ardıcıllığın toplusudur – adə tə n boşluqların daxil edilmə si və qabaqcıl və ya arxadakı boşluqların ə lavə edilmə si ilə – bütün ardıcıllıq sə tirlə ri eyni uzunluqda olsun.

Düzə lişlə r hə r bir ardıcıllığın tam uzunluğuna uzana bilə r və ya hə r bir ardıcıllığın alt bölmə si ilə mə hdudlaşa bilə r. Biopython-da bütün ardıcıl düzülmə lə r bölmə 6.1-də tə svir edilə n Alignment obyekti ilə tə msil olunur . Alignment obyektlə ri Clustal və ya BLAT (bölmə 6.6- da tə svir edilmişdir.) kimi hizalama proqramının çıxışını tə hlil etmə klə və ya iki ardıcıllığı bir-birinə uyğunlaşdıra bilə n Biopython-un cüt ardıcıllıqla düzlə ndiricisində n istifadə etmə klə ə ldə edilə bilə r (Fə sil 7-də tə svir edilmişdir).

Bio.AlignIO-da köhnə MultipleSeqAlignment sinfinin və tə hlilçilə rin tə sviri üçün Fə sil 8-ə baxın. MultipleSeqAlignment obyektlə rini yaradan ardıcıllıqla uyğunlaşdırma proqramının çıxışını tə hlil edə n.

6.1 Obyektlə rin hizalanması

Alignment sinfi Bio.Align-də müə yyə n edilmişdir. Adə tə n siz hizalama proqramlarının çıxışını tə hlil etmə klə (bölmə 6.6) və ya Biopython-un ikili düzlə ndiricisini işə salmaqla (Fə sil 7) Alignment obyektini ə ldə edə rdiniz. Bu bölmə nin faydası üçün biz sıfırdan Alignment obyekti yaradacağıq.

6.1.1 Ardıcıllıq və koordinatlardan Alignment obyektinin yaradılması

Tutaq ki, üç ardıcıllığınız var:

```
>>> seqA = "CCGGTTTTT" >>>
seqB = "AGTTTAA" >>> seqC
= "AGGTTT" >>> ardıcıllıqlar
= [seqA, seqB, seqC]
```

Alignment obyekti yaratmaq üçün ardıcıllığın bir-birinə necə düzüldüyünü müə yyə n edə n koordinatlara da ehtiyacımız var. Bunun üçün NumPy massivində n istifadə edirik:

```
>>> numpy-ni np olaraq idxal
edin >>> koordinatları = np.array([[1, 3, 4, 7, 9], [0, 2, 2, 5, 5], [0, 2, 3, 6, 6]])
```

Bu koordinatlar aşağıdakı ardıcıllıq seqmentlə ri üçün düzülüşü müə yyə n edir:

- SeqA[1:3], SeqB[0:2] və SeqC[0:2] bir-birinə uyğunlaşdırılıb;
- SeqA[3:4] və SeqC[2:3] seqB-də bir nukleotid boşluğu ilə bir-birinə uyğunlaşdırılıb;
- SeqA[4:7], SeqB[2:5] və SeqC[3:6] bir-birinə uyğunlaşdırılıb;

Machine Translated by Google

- SeqA[7:9] seqB və ya seqC ilə uyğunlaşdırılmayıb.

Qeyd edə k ki, düzülmə seqA-nın birinci nukleotidini və seqB-nin son iki nukleotidini ə hatə etmir.
İndi Alignment obyektini yarada bilə rik:

```
>>> Bio.Align import Alignment >>> alignment =
Alignment(ardıcıllıqlar, koordinatlar) >>> alignment # doctest:
+ELLIPSIS < Hazırlama obyekti (3 sıra x 8 sütun) ...>
```

Hizalama obyektində bu düzülmə yə daxil olan ardıcıllıqlara işarə edə n atribut ardıcıllığı var:

```
>>> alignment.sequences
 ['CCGGTTTTT', 'AGTTTAA', 'AGGTTT']
```

və bir atribut hizalanma koordinatları ilə ə laqə lə ndirir:

```
>>> alignment.koordinatlar
massivi([[1, 3, 4, 7, 9], [0, 2, 2, 5,
            5], [0, 2, 3, 6, 6]])
```

Hizalanmanı açıq şə kildə göstə rmə k üçün Alignment obyektini çap edin:

```
>>> çap (düzlə şdirmə )
                                1 CGGTTTTT 9
                                0 AG-TTT-- 5
                                0 AGGTTT-- 6
<BLANKLINE>
```

hə r bir ardıcıllığın başlanğıc və son koordinatları uyğun olaraq, hizalanmanın solunda və sağında göstə rilir.

6.1.2 Hizalanmış ardıcıllıqlardan Alignment obyektinin yaradılması

Boşluqları tə msil edə n tirelə rlə düzlə nmiş ardıcıllıqla başlasanız, infer_coordinates sinif metodundan istifadə edə rə k koordinatları hesablaya bilə rsiniz. Bu üsul, ilk növbə də , Biopython-un hizalama analizatorlarında istifadə olunur (bax. Bölmə 6.6), lakin o, başqa mə qsə dlə r üçün faydalı ola bilə r. Mə sə lə n, Alignment obyektini düzülmüş ardıcıllıqlardan aşağıdakı kimi qura bilə rsiniz:

```
>>> aligned_sequences = ["CGGTTTTT", "AG-TTT--", "AGGTTT--"] >>> sequences =
[aligned_sequence.replace("-", "")
 ...                            düzülmüş_ardıcıllıq üçün ] # fmt: keçin
 ...
>>> ardıcıllığı
 ['CGGTTTTT', 'AGTTT', 'AGGTTT'] >>> koordinatları =
Alignment.infer_coordinates(aligned_sequences) >>> koordinatları

massiv([[0, 2, 3, 6, 8], [0, 2, 2, 5,
            5], [0, 2, 3, 6, 6]])
```

seqA-nın ilkin G nukleotidi və seqB-nin son CC nukleotidlə ri uyğunlaşmaya daxil edilmə yib və buna görə də burada yoxdur. Ancaq bunu düzə ltmə k asandır:

Machine Translated by Google

```
>>> sequences[0] = "C" + sequences[0] >>>
sequences[1] = sequences[1] + "AA" >>> sequences

 ['CCGGTTTTT', 'AGTTTAA', 'AGGTTT'] >>> koordinatları[0, :]
+= 1 >>> koordinatları

massiv([[1, 3, 4, 7, 9], [0, 2, 2, 5,
        5], [0, 2, 3, 6, 6]])
```

İndi Alignment obyektini yarada bilə rik:

```
>>> hizalama = Hizalama (ardıcıllıq, koordinatlar) >>> çap
(düzlə şdirmə )
                    1 CGGTTTTT 9
                    0 AG-TTT-- 5
                    0 AGGTTT-- 6
<BLANKLINE>
```

yuxarıda 6.1.1 bölmə sində yaradılmış Alignment obyekti ilə eyni olan .

Varsayılan olaraq, Alignment başlatıcısının koordinat arqumenti hizalanmada heç bir boşluq olmadığını güman edə n Heç biri deyil. Boş olmayan düzülmə də ki bütün ardıcıllıqlar eyni uzunluğa malik olmalıdır. Ə gə r koordinatlar arqumenti Yoxdursa, onda başlatıcı Alignment obyektinin koordinat atributunu sizin üçün dolduracaq:

```
>>> unified_alignment = Alignment(["ACGTACGT", "AAGTACGT", "ACGTACCT"]) >>> unpped_alignment
# doctest: +ELLIPSIS <Alignment obyekti (3 sə tir x 8 sütun) ...>
>>> unpped_alignment.koordinatlar massivi([[0, 8], [0], [0] > çap
(açıq_düzlə şdirmə )
```

```
                    0 ACGTACGT 8
                    0 AAGTACGT 8
                    0 ACGTACCT 8
<BLANKLINE>
```

6.1.3 Ümumi düzülmə atributları

Aşağıdakı atributlar adə tə n Alignment obyektlə rində olur:

- ardıcıllıqlar: Bu, bir-birinə uyğunlaşdırılmış ardıcıllıqların siyahısıdır. Düzə lişin necə olmasından asılı olaraq yaradılmış, ardıcıllıqlar aşağıdakı növlə rə malik ola bilə r:
  - düz Python sə tri;
  - Seq;
  - MutableSeq;
  - SeqRecord;
  - bayt;
  - bytearray;
  - numpy.int32 mə lumat növü ilə NumPy massivi;

Machine Translated by Google

– "c", "B", "i" və ya "I" formatlı bitişik buferi olan hər hansı digər obyekt; – PairwiseAligner obyektinin ələ lifba atributunda müəyyən edilmiş obyektlərin siyahıları və ya dəstləri hizalanma yaratdı (bax bölmə 7.11).

Cütlü düzülmələr üçün (iki ardıcıllığın düzülməsi nəzərdə tutulur) xassələrin hədəfi və sorğusu müvafiq olaraq ardıcıllıqlar[0] və ardıcıllıqlar[1] üçün ləqəblərdir.

- koordinatlar: ardıcıllıqların necə olduğunu müəyyən edən ardıcıllıq indekslərini saxlayan tam ədədlərin NumPy massivi bir-birinə uyğunlaşdırılmış;
- xal: hizalama faylında təhlilçi tərəfindən tapılan və ya proqram tərəfindən hesablanan uyğunlaşma hesabı PairwiseAligner (bax bölmə 7.1);
- annotasiyalar: Düzlənşdirmə ilə əlaqəli əksər digər annotasiyaları saxlayan lüğət;
- sütun_annotasiyaları: Düzlənşdirmə boyunca uzanan və düzülmə ilə eyni uzunluğa malik olan annotasiyaları saxlayan lüğət, məsələn, konsensus ardıcıllığı (nümunə üçün bölmə 6.7.2-yə baxın).

Bio.Align-də təhlilçi tərəfindən yaradılmış Alignment obyekti hizalanmanın oxunduğu hizalanma fayl formatından asılı olaraq əlavə atributlara malik ola bilər.

6.2 Düzlənşdirmənin dilimlənməsi və indekslənşdirilməsi

Formanın düzülməsi[k, i:j] dilimləri, burada k tam ədəddir və i və j tam ədəddir və ya yoxdur, hədəf (əgər k=0) və ya sorğu (əgər k=1) üçün düzlənmiş ardıcıllığı (boşluqlar daxil olmaqla) göstərən sətri qaytarır.

Bunu göstərmək üçün aşağıdakı nümunə dəçap edilmiş düzülmə 8 sütundan ibarətdir:

```
>>> çap (düzlənşdirmə )
                    1 CGGTTTTT 9
                    0 AG-TTT-- 5
                    0 AGGTTT-- 6
<BLANKLINE>
>>> hizalama.uzunluq 8
```

Ayrı-ayrılıqda düzülmüş ardıcıllıq sətirlərini əldə etmək üçün istifadə edin

```
>>> düzülmə [0]
'CGGTTTTT'
>>> düzülmə [1]
'AG-TTT--'
>>> düzülmə [2]
'AGGTTT--'
>>> hizalama[0, :]
'CGGTTTTT'
>>> hizalama[1, :]
'AG-TTT--'
>>> düzülmə [0, 1:-1]
'GGTTTT'
>>> hizalama[1, 1:-1]
'G-TTT-'
```

Daxil edilən cək sütunlar tam ədədlər üzərində təkrarlanandan istifadə etmək lə də seçilə bilər:

Machine Translated by Google

```
>>> düzülmə [0, (1, 2, 4)]
'GGT'
>>> düzülmə [1, diapazon(0, 5, 2)]
'AT'
```

Çap edilmiş düzülmə nin [i, j] mövqeyində hə rfi almaq üçün hizalama[i, j] istifadə edin; bu "-" qaytaracaq hə min mövqedə boşluq aşkar edilə rsə :

```
>>> düzülmə [0, 2]
'G'
>>> düzülmə [2, 6]
'-'
```

Hizalanmada xüsusi sütunlar ə ldə etmə k üçün istifadə edin

```
>>> düzülmə [:, 0]
'CAA'
>>> düzülmə [:, 1]
'GGG'
>>> düzülmə [:, 2]
'GG'
```

Formanın düzülmə si[i:j:k] dilimlə ri düzlə şdirmə nin yalnız ardıcıllığı [i:j:k] daxil olmaqla yeni Alignment obyektini qaytarır:

```
>>> alignment[1:] # doctest:+ELLIPSIS <Alignment obyekti (2
sə tir x 6 sütun) ...> >>> çap (alignment[1:]) hə də fində

                    0 AG-TTT 5
                    0 ||-||| 6
                    0 AGGTTT 6
sorğu
<BLANKLINE>
```

Formanın düzülmə si[:, i:j] dilimlə ri, burada i və j tam ə də dlə rdir və ya yoxdur, çap edilmiş düzülmə də yalnız i-də n j-ə qə də r sütunları ehtiva edə n yeni Alignment obyektini qaytarır.

Yuxarıdakı uyğunlaşdırma nümunə si üçün ilk 4 sütunu çıxarmaq:

```
>>> alignment[:, :4] # doctest:+ELLIPSIS <Hazırlama obyekti (3
sə tir x 4 sütun) ...> >>> çap (alignment[:, :4])

                    1 CGGT 5
                    0 AG-T 3
                    0 AGGT 4
<BLANKLINE>
```

Eynilə , son 6 sütunu çıxarmaq:

```
>>> alignment[:, -6:] # doctest:+ELLIPSIS <Alignment obyekti (3
sə tir x 6 sütun) ...> >>> çap (alignment[:, -6:])

                    3 GTTTTT 9
                    2 -TTT-- 5
                    2 GTTT-- 6
<BLANKLINE>
```

Sütun indeksi hə mçinin tam ə də dlə rin iterativi ola bilə r:

Machine Translated by Google

```
>>> çap (düzlə şdirmə [:, (1, 3, 0)])
                    0 GTC 3
                    0 GTA 3
                    0 GTA 3
<BLANKLINE>
```

Alignment çağırışı[:, :] hizalanmanın surə tini qaytarır.

6.3 Hizalanma haqqında mə lumat ə ldə etmə k

6.3.1 Düzlə şdirmə forması

Düzə ldilmiş ardıcıllıqların sayı len(alignment) ilə qaytarılır:

```
>>> lent (düzlə şdirmə )
3
```

Hizalanma uzunluğu çap edilmiş hizalanmadakı sütunların sayı kimi müə yyə n edilir. Bu, hə r bir ardıcıllıqdakı uyğunluqların sayının, uyğunsuzluqların sayının və boşluqların ümumi uzunluğunun cə minə bə rabə rdir:

```
>>> hizalama.uzunluq 8
```

Forma xassə si hizalanmanın uzunluğundan və sütunların sayından ibarə t olan də ftə ri qaytarır çap edildiyi kimi düzülür:

```
>>> alignment.forma (3, 8)
```

6.3.2 Düzə lişlə rin müqayisə si

Alignment1.sequences və alignment2.sequences-də ki ardıcıllıqların hə r biri bir-birinə bə rabə rdirsə və alignment1.coordinates və alignment2.coordinates eyni koordinatları ehtiva edirsə , iki düzülmə bir-birinə bə rabə rdir (yani alignment1 == alignment2 True olaraq qiymə tlə ndirir). Bu şə rtlə rdə n hə r hansı biri yerinə yetirilmə zsə , alignment1 == alignment2 False olaraq qiymə tlə ndirilir. İki düzülmə nin bə rabə rsizliyi (mə sə lə n, düzlə şdirmə 1 < düzülmə 2) ə vvə lcə alignment1.sequences və alignment2.sequences müqayisə edilmə klə və ə gə r onlar bə rabə rdirsə , alignment1.coordinates with alignment2.coordinates müqayisə edilmə klə müə yyə n edilir.

6.3.3 Düzlə nmiş ardıcıllıqların indekslə rinin tapılması

Cütlü düzülmə lə r üçün, hizalanmanın uyğunlaşdırılmış xüsusiyyə ti bir-birinə uyğunlaşdırılmış hə də f və sorğu ardıcıllığında alt ardıcıllıqların başlanğıc və son indekslə rini qaytarır. Hə də f (t) və sorğu (q) arasında düzülmə N parçadan ibarə tdirsə , siz N uzunluğunda iki də st alırsınız:

```
(((t_start1, t_end1), (t_start2, t_end2), ..., (t_startN, t_endN)), ((q_start1, q_end1), (q_start2,
  q_end2), ..., (q_startN, q_endN)))
```

Mə sə lə n,

```
>>> cütlü_alignment = hizalama[:2, :] >>> çap
(cütlü_alignment) hə də fi
                    1 CGGTTTTT 9
                    0 .|-|||-- 8 0 AG-TTT--
sorğu               5
<BLANKLINE>
```

Machine Translated by Google

```
>>> pairwise_alignment.aligned massiv([[[1, 3], [4, 7]],
<BLANKLINE>


            [[0, 2], [2,
              5]]])
```

Qeyd edə k ki, müxtə lif düzülmə lə r bir-birinə uyğunlaşdırılmış eyni alt ardıcıllığa malik ola bilə r. Xüsusilə , bu, düzülmə lə r yalnız boşluqların yerlə şdirilmə si baxımından bir-birində n fə rqlə ndikdə

baş verə bilə r:

```
>>> pairwise_alignment1 = Alignment(["AAACAAA", "AAAAGAAA"],
...                                 np.array([[0, 3, 4, 4, 7], [0, 3, 3, 4, 7]])) # fmt: atla
...
>>> pairwise_alignment2 = Hizalama(["AAACAAA", "AAAAGAAA"],
...                                 np.array([[0, 3, 3, 4, 7], [0, 3, 4, 4, 7]])) # fmt: keçin
...
>>> çap (cüt şə kildə _alignment1)
0 |||--|||          0 AAAC-AAA 7 hə də f
        8
sorğu               0 AAA-GAAA 7
<BLANKLINE> >>>
çap (cütlü_alignment2)
0 |||--|||          0 AAA-CAAA 7 hə də f
        8
sorğu               0 AAAG-AAA 7
<BLANKLINE>
>>> pairwise_alignment1.aligned massiv([[[0, 3], [4, 7]],
<BLANKLINE>


            [[0, 3], [4,
              7]]])
>>> pairwise_alignment2.aligned massiv([[[0, 3], [4,
7]],

<BLANKLINE>
            [[0, 3], [4,
              7]]])
```

Ə mlak indekslə ri düzülmə də ki hə r hə rfin ardıcıllıq indeksi ilə 2D NumPy massivi qaytarır, -1 ilə göstə rilə n boşluqlarla:

```
>>> çap (düzlə şdirmə )
                    1 CGGTTTTT 9
                    0 AG-TTT-- 5
                    0 AGGTTT-- 6
<BLANKLINE>
>>> alignment.indices massivi([[ 1,
2, 3, 4, 5, 6, 7, 8], [ 0, 1, -1, 2, 3, 4, -1, -1], [ 0, 1, 2, 3, 4, 5, -1, -1]])
```

inverse_indices xassə si ardıcıllıqdakı hə r hə rf üçün düzülmə də sütun indeksi ilə düzülmüş ardıcıllıqların hə r biri üçün bir olan 1D NumPy massivlə rinin siyahısını qaytarır. Düzə lişdə olmayan hə rflə r -1 ilə işarə lə nir:

Machine Translated by Google

```
>>> alignment.sequences ['CCGGTTTTT',
 'AGTTTAA', 'AGGTTT'] >>> alignment.inverse_indices #
doktest: +NORMALIZE_WHITESPACE [massiv([-1, 0, 1, 2, 3, 4, 5, 6, , , 7, [3, 7], -1, -1]), massiv([0, 1, 2, 3, 4,
 5])]
```

6.3.4 Eyniliklə rin, uyğunsuzluqların və boşluqların hesablanması

Sayma metodu ikili düzülmə nin eyniliklə rinin, uyğunsuzluqlarının və boşluqlarının sayını hesablayır. İkidə n çox ardıcıllığın düzülüşü üçün eyniliklə rin, uyğunsuzluqların və boşluqların sayı düzülmə də ki bütün ardıcıllıq cütlə ri üçün hesablanır və cə mlə nir. Üç ə də d AlignmentCounts obyekti kimi qaytarılır, hansı ki, sahə lə r boşluqları, identifikasiyaları və uyğunsuzluqları ilə adlandırılmış bir də stdir. Bu üsul hazırda heç bir arqument tə lə b etmir, lakin gə lə cə kdə onun davranışını fə rdilə şdirmə yə imkan verə n ə lavə arqumentlə ri qə bul etmə k üçün çox güman ki, də yişdirilə cə k.

```
>>> hə də fi çap edin
                                1 CGGTTTTT 9
                                0 .|-|||-- 8
sorğu                           0 AG-TTT-- 5
<BLANKLINE>
>>> pairwise_alignment.counts()
AlignmentCounts(boşluqlar=3, eyniliklə r=4, uyğunsuzluqlar=1) >>> çap (alignment)

                                1 CGGTTTTT 9
                                0 AG-TTT-- 5
                                0 AGGTTT-- 6
<BLANKLINE>
>>> alignment.counts()
AlignmentCounts(boşluqlar=8, şə xsiyyə tlə r=14, uyğunsuzluqlar=2)
```

6.3.5 Hə rf tezliklə ri

Tezliklə r metodu hə r hə rfin düzülmə nin hə r sütununda nə qə də r tez-tez göründüyünü hesablayır:

```
>>> alignment.tezliklə r # doktest: +NORMALIZE_WHITESPACE {'C': massiv([1., 0., 0., 0., 0., 0., 0.,
0.]), 'G': massiv([0., 3., 2., 0.., 0.', 0., 0.'), 0. massiv([0., 0., 0., 3., 3., 3., 1., 1.]),
   'A': massiv([2., 0., 0., 0., 0., 0., 0.]), '-': massiv([0., 0., ., ., ., 1., 0. 2.])}
```

6.3.6 Ə və zetmə lə r

Hə r bir cüt nukleotid arasındakı ə və zetmə lə rin sayını tapmaq üçün ə və zetmə metodundan istifadə edin:

```
>>> m = hizalama.ə və zetmə lə r >>> çap (m)

        A       CGT
A 1,0 0,0 0,0 0,0
C 2.0 0.0 0.0 0.0
G 0,0 0,0 4,0 0,0 T 0,0 0,0
0,0 9,0 <BLANKLINE>
```

Machine Translated by Google

Qeyd edə k ki, matris simmetrik deyil: R sə tir hə rfi və C sütun hə rfi üçün hesablamalar ardıcıllıqdakı R hə rfinin onun altında görünə n ardıcıllıqla C hə rfi ilə ə və zlə nmə sinin sayıdır. Mə sə lə n, sonrakı ardıcıllıqla A-ya düzülə n C-lə rin sayı

```
>>> m["C", "A"] 2.0
```

sonrakı ardıcıllıqla C-yə düzülə n A-ların sayı isə

```
>>> m["A", "C"]
0.0
```

Simmetrik matris ə ldə etmə k üçün istifadə edin

```
>>> m += m.transpose() >>> m /=
2.0 >>> çap (m)

        A       CGT
A 1,0 1,0 0,0 0,0
C 1,0 0,0 0,0 0,0 G 0,0 0,0
4,0 0,0 T 0,0 0,0 0,0 9,0
<BLANKLINE> >>> m["A",
"C"] 1,0 >>>
m["C", "A"] 1,0
```

Düzə lişdə A və T arasında ə və zetmə lə rin ümumi sayı 1,0 + 1,0 = 2-dir.

6.3.7 Massivlə r kimi düzülmə lə r

NumPy-də n istifadə edə rə k, hizalanma obyektini hə rflə r massivinə çevirə bilə rsiniz. Xüsusilə , bu, hizalanma mə zmunu üzrə sürə tli hesablamalar üçün faydalı ola bilə r.

```
>>> align_array = np.array(alignment) >>> align_array.shape
(3, 8) >>> align_array # doctest:

+NORMALIZE_WHITESPACE massivi([[b'C', b'G', b'G', b'T', b'T', b'T, b'''T', b'T',
b''T' b'G', b'-', b'T', b'T', b'T', b'-', b'-'], [b'A', b'G', b'G', b'T', b'T', b'T', b'-', b'-']], dtype='|S1')
```

Varsayılan olaraq, bu sizə bayt simvolları massivi verə cə k (dtype='|S1' mə lumat növü ilə). Siz dtype='U' istifadə edə rə k Unicode (Python sə tri) simvolları massivi yarada bilə rsiniz:

```
>>> align_array = np.array(alignment, dtype="U") >>> align_array # doctest:
+NORMALIZE_WHITESPACE massivi([['C', 'G', 'G', 'T', 'T', 'T', 'T', 'T', 'T',', 'T', 'T',',',
'A,',', 'T', 'T', '-', '-'], ['A', 'G', 'G', 'T', 'T', 'T', '-', '-']], dtype='<U1')
```

Qeyd edə k ki, hizalama obyekti və NumPy massivi align_array yaddaşda olan ayrı obyektlə rdir - birinin redaktə si digə rini yenilə mə yə cə k!

Machine Translated by Google

6.4 Düzlə şdirmə üzrə ə mə liyyatlar

6.4.1 Düzə lişin çeşidlə nmə si

Çeşidlə mə üsulu hizalanma ardıcıllığını çeşidlə yir. Varsayılan olaraq, çeşidlə mə ə gə r varsa, hə r bir ardıcıllığın id atributuna və ya ə ks halda ardıcıllığın mə zmununa ə sasə n hə yata keçirilir.

```
>>> çap (düzlə şdirmə )
                                    1 CGGTTTTT 9
                                    0 AG-TTT-- 5
                                    0 AGGTTT-- 6
<BLANKLINE>
>>> alignment.sort() >>> çap
(alignment)
                                    0 AGGTTT-- 6
                                    0 AG-TTT-- 5
                                    1 CGGTTTTT 9
<BLANKLINE>
```

Alternativ olaraq, çeşidlə mə sırasını müə yyə n etmə k üçün ə sas funksiyanı tə min edə bilə rsiniz. Mə sə lə n, GC mə zmununu artırmaqla ardıcıllıqları çeşidlə yə bilə rsiniz:

```
>>> Bio.SeqUtils -də n import gc_fraction >>>
alignment.sort(key=gc_fraction) >>> print(alignment) #
CHEEMPIE 0 AG-TTT-- 5

                                    0 AGGTTT-- 6
                                    1 CGGTTTTT 9
<BLANKLINE>
```

Nə zə rə alın ki, ə sas funksiya tə kcə düzlə nmiş hissə yə deyil, tam ardıcıllığa (o cümlə də n seqB-nin ilkin A və son GG nukleotidlə rinə) tə tbiq edilir.

Ə ks arqument GC mə zmununun azaldılmasında ardıcıllığı ə ldə etmə k üçün çeşidlə mə sırasını tə rsinə çevirmə yə imkan verir:

```
>>> alignment.sort(açar=gc_fraction, tə rs=Doğru) >>> çap (alignment)

                                    1 CGGTTTTT 9
                                    0 AGGTTT-- 6
                                    0 AG-TTT-- 5
<BLANKLINE>
```

6.4.2 Düzə lişin tə rs tamamlanması

Düzə lişin tə rs tamamlanması hə r bir ardıcıllığın ə ks tamamlayıcısını götürə cə k və koordinatları yenidə n hesablayacaq:

```
>>> alignment.sequences
 ['CCGGTTTTT', 'AGGTTT', 'AGTTTAA'] >>> rc_alignment
= alignment.reverse_complement() >>> print(rc_alignment.sequences)

 ['AAAAACCGG', 'AAACCT', 'TTAAACT'] >>> çap
(rc_alignment)
                                    0 AAAAACCG 8
                                    0 --AAACCT 6
```

Machine Translated by Google

```
                                    2 --AAA-CT 7
<BLANKLINE>
>>> alignment[:, :4].ardıcıllıqlar ['CCGGTTTTT',
 'AGGTTT', 'AGTTTAA'] >>> çap edin(alignment[:, :4])

                                    1 CGGT 5
                                    0 AGGT 4
                                    0 AG-T 3
<BLANKLINE>
>>> rc_alignment = alignment[:, :4].reverse_complement() >>> rc_alignment[:, :4].sequences
['AAAAACCGG', 'AAACCT', 'TTAAACT'] >>> çap
 (rc_alignment[:, :4])

                                    4 ACCG 8
                                    2 ACCT 6
                                    4 A-CT 7
<BLANKLINE>
```

Düzə lişin tə rs tamamlanması onun sütun annotasiyalarını qoruyur (ə ks ardıcıllıqla), lakin bütün digə r annotasiyaları rə dd edir.

6.4.3 Düzə lişlə rin ə lavə edilmə si

Düzə lişlə r eyni sayda cə rgə yə malikdirsə , uzadılmış düzülmə yaratmaq üçün bir-birinə ə lavə edilə bilə r. Nümunə olaraq, ə vvə lcə iki düzülmə yaradaq:

```
>>> Bio.Seq - də n idxal Seq >>>
Bio.SeqRecord import SeqRecord >>> a1 =
SeqRecord(Seq("AAAAC"), id="Alpha") >>> b1 = SeqRecord(Seq("AAAC"),
id="Beta") >>> c1 = SeqRecord(Seq("AAAAGam"), = >> " id
SeqRecord(Seq("GTT"), id="Alpha") >>> b2 = SeqRecord(Seq("TT"),
id="Beta") >>> c2 = SeqRecord(Seq("GT"), id="Qamma") >>> sol =
Alignment(


...             [a1, b1, c1], koordinatlar=np.array([[0, 3, 4, 5], [0, 3, 3, 4], [0, 3, 4, 5]])
... ) >>>
left.annotations = {"alə t": "demo", "ad": "start"} >>> left.column_annotations = {"stats":
"CCCXC"} >>> sağ = Hizalama(


...             [a2, b2, c2], koordinatlar=np.massiv([[0, 1, 2, 3], [0, 0, 1, 2], [0, 1, 1, 2]])
... ) >>>
right.annotations = {"alə t": "demo", "ad": "son"} >>> right.column_annotations =
{"stats": "CXC"}
```

İndi bu iki uyğunlaşmaya baxaq:

```
>>> çap (solda)
Alfa                    0 AAAAC 5
Beta                    0 AAA-C 4
Qamma                   0 AAAAG 5
<BLANKLINE> >>>
```

çap (sağda)

Machine Translated by Google

```
Alfa                    0 GTT 3
Beta                    0 -TT 2
Qamma                   0 GT 2
<BLANKLINE>
```

İki düzülmə nin ə lavə edilmə si iki düzülmə ni sıra üzrə birlə şdirə cə k:

```
>>> birlə şdirilmiş = sol + sağ >>> çap
(birlə şdirilmiş)
Alfa                    0 AAAACGTT 8
Beta                    0 AAA-C-TT 6
Qamma                   0 AAAAGG-T 7
<BLANKLINE>
```

Bunun işlə mə si üçün hə r iki düzülmə eyni sayda ardıcıllığa malik olmalıdır (burada onların hə r ikisində 3 sıra var):

```
>>> len (solda)
3
>>> len(sağda) 3 >>>

len(birlə şdirilmiş) 3
```

Ardıcıllıqlar birlikdə ə lavə edilə bilə n SeqRecord obyektlə ridir. Annotasiyanın necə işlə ndiyi barə də tə fə rrüatlar üçün Fə sil 4- ə baxın. Bu misal xüsusi haldır ki, hə r iki orijinal düzülmə eyni adları bölüşür, yə ni sə tirlə r ə lavə edildikdə onlar da eyni adı alırlar.

İstə nilə n ümumi annotasiya qorunur, lakin fə rqli annotasiya itirilir. Bu istifadə edilə n eyni davranışdır SeqRecord annotasiyalarında və qeyri-münasib də yə rlə rin tə sadüfə n yayılmasının qarşısını almaq üçün nə zə rdə tutulub:

```
>>> birlə şdirilmiş.annotasiyalar
{'alə t': 'demo'}
```

Eynilə , sütun başına hə r hansı ümumi annotasiyalar birlə şdirilir:

```
>>> combined.column_annotations {'stats':
'CCCXCCXC'}
```

6.4.4 Cüt ardıcıllıqla düzülmə nin xə ritə lə şdirilmə si

Tutaq ki, bir xromosoma transkriptinin ikili düzülüşü var:

```
>>> xromosom = "AAAAAAAAACCCCCCCAAAAAAAAAAAGGGGGGAAAAAAAAA"
>>> transkript = "CCCCCCGGGGGG" >>> sequences1
= [xromosom, transkript] >>> koordinatlar1 = np.array([[8, 15, 26,
32], [0, 7, 7, 13]]) >>> alignment1 = Alignment, çap (koordinatlar>> 1 )1 )


                        8 CCCCCAAAAAAAAAAAGGGGGG 32
                        0 |||||||----------|||||| 24
sorğu                   0 CCCCCCC----------GGGGGG 13
<BLANKLINE>
```

və transkript və ardıcıllıq arasında cüt uyğunlaşma (mə sə lə n, RNT-seq ilə ə ldə edilir):

Machine Translated by Google

```
>>> rnaseq = "CCCCGGGG" >>>
sequences2 = [transkript, rnaseq] >>> koordinatlar2 = np.array([[3,
11], [0, 8]]) >>> alignment2 = Alignment(sequences2, coordinates2) >>>
print(alignment2) hə  də  f

                                    3 CCCCGGGG 11
                                    0 | | | | | | | | 8
sorğu                               0 CCCCGGGG 8
<BLANKLINE>
```

RNT ardıcıllığının genomla düzülmə sini tapmaq üçün arqument olaraq alignment2 ilə alignment1-də xə ritə metodundan istifadə edin:

```
>>> alignment3 = alignment1.map(alignment2) >>> print(alignment3) target

                                   11 CCCCAAAAAAAAAAAAGGGG 30
                                    0 | | | |------| | | | 19
sorğu                               0 CCCC----------GGGG 8
<BLANKLINE> >>>
alignment3.coordinates massivi([[11, 15, 26,
30], [ 0, 4, 4, 8]]) >>> formatı(alignment3,
                "psl") '8\t0\t0\t0\t0\t0\t1\t11\t+
\tquery\t8\t0\t8\ttarget\t40\t11\t30\t2\t4,4,\t0,4,\t11,26,\n'
```

Ardıcıllıqları çap edə bilmə k üçün bu nümunə də biz müə yyə n ardıcıllıqla mə zmunu olan ardıcıllıqlardan istifadə edə rə k alignment1 və alignment2 qurduq. Bununla belə , hizalanmanın xə ritə lə şdirilmə si ardıcıllığın mə zmununda asılı deyil; hizalanma3 üçün koordinatları qurmaq üçün yalnız alignment1 və alignment2 koordinatlarından istifadə olunur.

Xə ritə metodu, hə mçinin müxtə lif genom yığıncaqları arasında uyğunluğu qaldırmaq üçün istifadə edilə bilə r. Bu halda, mə nlik iki genom yığıncağı arasında bir DNT uyğunlaşmasıdır və arqument genom birlə şmə lə rində n birinə qarşı transkriptin uyğunlaşdırılmasıdır:

```
>>> Bio import Align >>> chain =
Align.read("panTro5ToPanTro6.over.chain", "chain") >>> chain.sequences[0].id ' chr1'

>>> len(chain.sequences[0].seq) 228573443

>>> chain.sequences[1].id 'chr1'

>>> len(chain.sequences[1].seq) 224244399 >>> import
numpy as np
>>> np.set_printoptions(threshold=5)
# hə r sə tirə 5 massiv elementi çap edin >>> print(chain.coordinates) # doctest:+ELLIPSIS [[12002020202 122250400 ...
122909818 122909819 122909835]

   [111776384 111776784 111776785 ... 112019962 112019962 112019978]]
```

şimpanze genom montajı panTro5-də chr1 diapazonunun 122250000:122909835 diapazonunun şimpanze genom yığım panTro6-nın chr1-in 111776384:112019978 diapazonuna uyğunlaşdığını göstə rir. Zə ncir fayl formatı haqqında ə traflı mə lumat üçün bölmə 6.7.20-ə baxın.

Machine Translated by Google

```
>>> transkript = Align.read("est.panTro5.psl", "psl") >>> transcript.sequences[0].id 'chr1'

>>> len(transkript.sequences[0].seq)
228573443
>>> transcript.sequences[1].id 'DC525629'

>>> len(transkript.sequences[1].seq) 407

>>> çap (transkript.koordinatlar) [[122835789
 122835847 122840993 122841145 122907212 122907314] 344]]
   [              32                 90                 90              242              242
```

Bu göstə rir ki, DC525629 ifadə edilmiş ardıcıllıq etiketinin 32:344 nukleotid diapazonu panTro5 şimpanze genom yığımının chr1 diapazonunun 122835789:122907314 diapazonuna uyğun gə lir. Nə zə rə alın ki, hə də f ardıcıllığı chain.sequences[0].seq və hə də f ardıcıllığı transkript.sequences[0] eyni uzunluğa malikdir:

```
>>> len(zə ncir.sequences[0].seq) == len(transkript.sequences[0].seq)
Doğrudur
```

Zə ncirin hə də fi və sorğusunu elə də yişirik ki, zə ncirin sorğusu transkriptin hə də finə uyğun olsun:

```
>>> zə ncir = zə ncir[::-1] >>>
chain.sequences[0].id 'chr1'

>>> len(chain.sequences[0].seq) 224244399

>>> chain.sequences[1].id 'chr1'

>>> len(chain.sequences[1].seq) 228573443 >>>

print(chain.coordinates) # doctest:+ELLIPSIS [[111776384 111776784 111776785 ...
 112019962 91291917]
   [122250000 122250400 122250400 ... 122909818 122909819 122909835]]
>>> np.set_printoptions(threshold=1000) # çap seçimlə rini sıfırlayın
```

İndi biz arqument olaraq transkriptlə chain.map-a zə ng edə rə k şimpanze genomunun panTro6-ya qarşı DC525629 koordinatlarını ə ldə edə bilə rik:

```
>>> lifted_transcript = chain.map(transkript) >>> lifted_transcript.sequences[0].id
'chr1' >>> len(lifted_transcript.sequences[0].seq) 224244399 >>>

lifted_transcript.sequences[1].id


 'DC525629'
>>> len(lifted_transcript.sequences[1].seq) 407

>>> çap (lift_transkript.koordinatlar) [[111982717
 111982775 111987921 111988073 112009200 112009302] 344]]
   [              32                 90                 90              242              242
```

Bu göstə rir ki, DC525629 ifadə edilmiş ardıcıllıq etiketinin 32:344 nukleotid diapazonu panTro6 şimpanze genom yığımının chr1 diapazonunun 111982717:112009302 diapazonuna uyğun gə lir. Qeyd edə k ki, DC525629 genomu şimpanze üzə rində dir

Machine Translated by Google

genom montajı panTro5 122907314 - 122835789 = 71525 bp, panTro6-da isə genom aralığı 112009302 - 111982717 = 26585 bp-dir.

6.4.5 Çox ardıcıl düzülmə nin xə ritə lə şdirilmə si

Şimpanze, insan, makaka, marmoset, siçan və siçovulların genomik ardıcıllığının çoxlu düzülmə sini nə zə rdə n keçirə k:

```
>>> Bio import Align >>> path =
"panTro5.maf" >>> genome_alignment
= Align.read(path, "maf") >>> genome_alignment.sequences -də qeyd üçün :
print (record.id, len(record.seq))
 ...
 ...
panTro5.chr1 228573443 hg19.chr1
249250621 rheMac8.chr1
225584828
calJac3.chr18 47448759
mm10.chr3 160039680
rn6.chr2 266435125
>>> genom_alignment.koordinatlar massivi([[133922962,
133922962, 133922970, 133922970, 133922972, 133922972,
                133922995, 133922998, 133923010], [155784573,
              155784573, 155784581, 155784581, 155784583, 155784587, 155781458 155784609, 155784621], [130383910,
                130383910, 130383918, 130383918, 130383920,
              130383920, 13038934, 13038194 130383958], [ 9790455, 9790455, 9790463, 9790463, 9790465, 9790465,
                9790488, 9790491, 9790588], [888], [86, 889 88858028,
              88858026, 88858024, 88858020, 88857997, 88857997, 88857985], [188162970, 188162967, 959, 1891,
                    188162957, 188162953, 188162930, 188162930,
              188162918]])
```

```
>>> çap (genom_alignment) panTro5.c
133922962 ---ACTAGTTA--CA----GTAACAGAAAATAAAATTTAAATAGAAACTTAAAggcc hg19.chr1 155784573 ---ACTAGTTA--CA----GTATACAGAAAcc
rheMac8.c 130383910 ---ACTAGTTA--CA----GTAACAGAAAATAAAATTTAAATAGAAACTTAAAggcc calJac3.c 9790455 ---ACTAGTTA--CA----
GTAACAGAAAATAAAATTTAAATAGAAGCTTAAAgg83 mm8980 TATAATAATTGTATATGTCACAGAAAAAAAATGAATTTTTCAAT---GACTTAATAGCC


rn6.chr2 188162970 TACAATAATTG--TATGTCATAGAAAAAAATGAATTTTCAAT---AACTTAATAGCC
<BLANKLINE>
panTro5.c 133923010 hg19.chr1
155784621 rheMac8.c 130383958

calJac3.c 9790503
mm10.chr3 88857985 rn6.chr2
188162918 <BLANKLINE>
```

Tutaq ki, biz genom birlə şmə lə rinin köhnə versiyalarını (panTro5, hg19, rheMac8, calJac3, mm10 və rn6) cari versiyaları (panTro6, hg38, rheMac10, calJac4, mm39 və rn7) ilə ə və z etmə k istə yirik. Bunu etmə k üçün bizə hə r növ üçün köhnə və yeni montaj versiyası arasında cüt uyğunlaşma lazımdır. Bunlar adə tə n UCSC-nin liftOver alə ti üçün istifadə olunan zə ncir faylları kimi UCSC tə rə fində n tə min edilir. İçində ki .zə ncir faylları

Machine Translated by Google

Biopython mə nbə paylamasında Testlə r/Align alt kataloqu yalnız müvafiq genomik bölgə ni daxil etmə k üçün UCSC-nin .zə ncir fayllarından çıxarılıb. Mə sə lə n, panTro5-də n panTro6-ya qaldırmaq üçün biz aşağıdakı mə zmunlu panTro5ToPanTro6.chain faylından istifadə edirik:

```
zə ncir 1198066 chr1 228573443 + 133919957 133932620 chr1 224244399 + 130607995 130620657 1
4990            0            2
1362            3            0
6308
```

Hə r bir növ üçün genom birlə şmə sini qaldırmaq üçün müvafiq .zə ncir faylında oxuyuruq:

```
>>> yollar = [
...         "panTro5ToPanTro6.chain",
...         "hg19ToHg38.chain",
...         "rheMac8ToRheMac10.chain",
...         "calJac3ToCalJac4.chain",
...         "mm10ToMm39.chain",
...         "rn6ToRn7.chain",
... ]
>>> liftover_alignments = [ Yollardakı yol üçün align.read(yol, "zə ncir") ] >>> liftover_alignments -də
liftover_alignment üçün :
...         çap(liftover_alignment.target.id, liftover_alignment.coordinates[0, :])
...
chr1 [133919957 133924947 133924947 133926309 133926312 133932620] chr1 [155184381
156354347 15631547 157128497 157137496] chr1 [130382477 130383872 130383872 130384222
130384222 130388520] chr18 [97170887 9788508 9795062 9795065 9795737] chr3 [66807541
74196805 74196831 94707528 94707528 94708176 947087] 87181 [188111581 188158351
188158351 188171225 188171225 188228261 188228261

  188236997]
```

Qeyd edə k ki, növlə rin sırası liftover_alignments və genome_alignment.sequences-də eynidir.
İndi biz çoxlu ardıcıllığı yeni genom montaj versiyalarına qaldıra bilə rik:

```
>>> genom_alignment = genome_alignment.mapall(liftover_alignments) >>>
genome_alignment.sequences -də qeyd üçün : print(record.id,
...         len(record.seq))
...
chr1 224244399
chr1 248956422
chr1 223616942
chr18 47031477
chr3 159745316
chr2 249053267
>>> genom_alignment.koordinatlar
massivi([[130611000, 130611000, 130611008, 130611008, 130611010, 130611010,
          130611033, 130611036, 130611048], [155814782,
         155814782, 155814790, 155814790, 155814792, 155814795, 152,81185 155814818, 155814830],
          [ 95186253, 95186253, 95186245, 95186245,
         95186243, 95186243, 95186286, 915, 915, 915, 9758318, 9758318, 9758326, 9758326, 9758328,
           9758328, 9758351, 9758354, 9758366],
```

Machine Translated by Google

```
[ 88765346, 88765343, 88765335, 88765333, 88765331, 88765327, 88765304, 88765304,
    88765292], [172461, 174256691, 174256691,
174256689, 174256685, 174256662, 174256662, 174256650]])
```

.zə ncir fayllarına ardıcıllıq mə zmunu daxil olmadığı üçün biz ardıcıl düzülüşü birbaşa çap edə bilmirik. Ə və zində hə r növ üçün genomik ardıcıllığı ayrıca oxuyuruq (tə nbə l yüklə mə yə imkan verdiyi üçün .2 bit fayl kimi; bölmə 5.4-ə baxın) :

```
>>> Bio import SeqIO -dan >>> adlar
= ("panTro6", "hg38", "rheMac10", "calJac4", "mm39", "rn7") >>> i üçün , sadalanan ad ( adlar): fayl
adı = f"{ name }.2bit" genom = SeqIO.parse(fayl adı),
...             " twobit"
...             genome_alignment.sequences[i].id assert
...             len(genome_alignment.sequences[i]) == len(genome[xromosom])
...             genom_alignment.sequences[i] = genom[xromosom] genom_alignment.sequences[i].id {isim} ==
...             f."ch"
...
...
>>> çap (genom_alignment) panTro6.c
130611000 ---ACTAGTTA--CA----GTAACAGAAAATAAAATTTAAATAGAAACTTAAAggcc hg38.chr1 155814782 ---ACTAGTTA--CA----
GTATACAGAAAcc rheMac10. 95186253 ---ACTAGTTA--CA----GTAACAGAAAATAAAATTTAAATAGAAACTTAAAggcc calJac4.c 9758318 ---
ACTAGTTA--CA----GTAACAGAaaataaaatttaaatagaagcttaaaggct mm384.chr TATAATAATTGTATATGTCACAGAAAAAAAATGAATTTTTCAAT---
GACTTAATAGCC

rn7.chr2 174256702 TACAATAATTG--TATGTCATAGAAAAAAATGAATTTTCAAT---AACTTAATAGCC
<BLANKLINE>
panTro6.c 130611048 hg38.chr1
155814830 rheMac10. 95186205

calJac4.c 9758366
mm39.chr3 88765292
rn7.chr2 174256650
<BLANKLINE>
```

Mapall metodu hə mçinin müvafiq amin turşusu ardıcıllığının çoxsaylı ardıcıl düzülüşündə n kodon ardıcıllığının çoxsaylı düzülmə sini yaratmaq üçün istifadə edilə bilə r (ə traflı mə lumat üçün Bölmə 7.12.2-yə baxın).

6.5 Alignments sinfi

Alignments (cə m) sinfi AlignmentsAbstractBaseClass-dan və siyahıdan miras qalır və Alignment obyektlə rini saxlamaq üçün siyahı kimi istifadə edilə bilə r. Alignments obyektlə rinin davranışı siyahı obyektlə rinin davranışından üç mühüm şə kildə fə rqlə nir:

- Alignments obyekti öz iteratorudur, halbuki siyahı obyektində iterin çağırılması hə r də fə yeni iterator yaradır. Buna görə də Alignments obyekti iteratoru Bio.Align.parse tə rə fində n qaytarılan iterator (bax. bölmə 6.6.1) və ya cütlü düzlə şdirici tə rə fində n qaytarılan iterator (Bölmə 7-ə baxın) kimi davranacaq .

  ```
  >>> Bio.Align idxal Alignments- də n

  >>> alignment_list = [alignment1, alignment2, alignment3] >>> alignment_list
  -də ki element üçün : print(repr(item)) #
  ...             doctest: +ELLIPSIS
  ```

Machine Translated by Google

```
...
<Hizalanma obyekti (2 sə tir x 24 sütun) ... at> <Hizalanma obyekti
(2 sə tir x 8 sütun) ... ünvanında> <Hazırlama obyekti (2 sə tir x 19
sütun) ...>

>>> hizalama_siyahısındakı element
...         üçün : çap (repr(item)) # doctest: +ELLIPSIS
...
<Hizalanma obyekti (2 sə tir x 24 sütun) ... at> <Hizalanma obyekti
(2 sə tir x 8 sütun) ... ünvanında> <Hazırlama obyekti (2 sə tir x 19
sütun) ...>

>>> alignments = Alignments([alignment1, alignment2, alignment3]) >>> >>> in the
element for alignments:
...         print(repr(item)) # doctest: +ELLIPSIS
...
<Hizalanma obyekti (2 sə tir x 24 sütun) ... at> <Hizalanma obyekti
(2 sə tir x 8 sütun) ... ünvanında> <Hazırlama obyekti (2 sə tir x 19
sütun) ...>

>>> düzülmə lə rdə ki element
...         üçün : print(repr(item)) # doctest: +ELLIPSIS
...
```

Son döngə heç nə çap etmir, çünki iterator artıq tükə nmişdir.

- Alignments sinfi iteratoru ilk elementinə sıfırlayan geri sarma metodunu müə yyə n edir və bizə imkan verir onu yenidə n döndə rin:

```
>>> alignments.rewind() >>>
alignmentlə rdə ki element üçün :
...         print(repr(item)) # doctest: +ELLIPSIS
...
<Hizalanma obyekti (2 sə tir x 24 sütun) ... at> <Hizalanma obyekti
(2 sə tir x 8 sütun) ... ünvanında> <Hazırlama obyekti (2 sə tir x 19
sütun) ...>
```

- Metadata Alignments obyektində atributlar kimi saxlanıla bilə r, halbuki düz siyahı qə bul etmir atributlar:

```
>>> alignment_list.score = 100 # doctest: +ELLIPSIS Traceback (ə n son
zə ng):
  ...
AttributeError: 'list' obyektinin 'balı' atributu yoxdur... >>> alignments.score = 100 >>> alignments.score
100
```

6.6 Oxuma və yazma hizaları

Clustal kimi ardıcıllıqla uyğunlaşdırma proqramından ə ldə edilə n çıxış Bio.Align.read və Bio.Align.parse funksiyaları ilə Alignment obyektlə rinə tə hlil edilə bilə r. Onların istifadə si Bio.SeqIO-da oxumaq və tə hlil etmə k funksiyalarına bə nzə yir (Bölmə 5.1-ə baxın): Oxu funksiyası tə k düzülmə ni ehtiva edə n çıxış faylını oxumaq üçün istifadə olunur.

Machine Translated by Google

və Alignment obyektini qaytarır, tə hlil funksiyası isə bir və ya daha çox düzülmə ni ehtiva edə n çıxış faylında saxlanılan düzülmə lə ri tə krarlamaq üçün iteratoru qaytarır. Bölmə 6.7 Bio.Align-də tə hlil edilə bilə n hizalama formatlarını tə svir edir. Bio.Align hə mçinin bu formatların ə ksə riyyə tində düzülmə lə ri yaza bilə n yazma funksiyasını tə min edir.

6.6.1 Oxu düzülmə lə ri

Bio.Align.parse tə rə fində n qaytarılan alignments iteratoru AlignmentsAbstractBaseClass baza sinfində n miras alır. Bu sinif ə vvə ldə n hizalamalar üzə rində dövrə vurmaq üçün iteratoru sıfırlayan geri çə kmə metodunu müə yyə n edir. Siz hə mçinin hizalanmaların sayını ə ldə etmə k üçün lin-ə zə ng edə bilə rsiniz. Fayl formatından asılı olaraq, düzülmə lə rin sayı açıq şə kildə faylda saxlanıla bilə r (mə sə lə n, bigBed, bigPsl və bigMaf faylları və ziyyə tində) və ya ə ks halda düzülmə lə rin sayı onların üzə rində bir də fə dövrə vurmaqla (və iteratoru orijinal və ziyyə tinə qaytarmaqla) sayılır. Ə gə r fayl böyükdürsə , len-in geri qaytarılması xeyli vaxt tə lə b edə bilə r. Düzə lişlə rin sayı keşlə ndikcə len-ə sonrakı zə nglə r tez qayıdacaq.

Fayl formatından asılı olaraq, Bio.Align.parse tə rə fində n qaytarılan alignmentlə r obyekti faylda tapılan metadatanı saxlayan atributları, mə sə lə n, hizalamaları yaratmaq üçün istifadə edilmiş proqram tə minatının versiya nömrə sini ehtiva edə bilə r. Hə r bir fayl formatı üçün saxlanılan xüsusi atributlar Bölmə 6.7-də tə svir edilmişdir.

6.6.2 Yazı düzülmə lə ri

Fayla düzülmə lə r yazmaq üçün istifadə edin

```
>>> Bio idxaldan Align >>> target
= " myfile.txt"
>>> Align.write (düzlə şdirmə , hə də f, "klastal")
```

düzülmə lə rin tə k düzülmə və ya düzülmə lə rin siyahısı olduğu halda, hə də f fayl adı və ya açıq fayla bə nzə r obyektdir və "klastal" istifadə edilə cə k fayl formatıdır. Bə zi fayl formatları metamə lumatların düzülmə lə rlə saxlanmasına icazə verdiyi və ya tə lə b etdiyinə görə , düzlə şdirmə lə rin düz siyahısı ə və zinə Alignments (cə m) sinfində n istifadə etmə k istə yə bilə rsiniz (bax. Bölmə 6.5), metadata lüğə tini alignmentlə r obyektində atribut kimi saxlamağa imkan verir:

```
>>> Bio idxaldan Align >>>
alignments = Align.Alignments(alignments) >>> metadata =
{"Proqram": " Biopython", "Versiya": "1.81"} >>> alignments.metadata =
metadata >>> target = "myfile.txt"

>>> Align.write (düzlə şdirmə , hə də f, "klastal")
```

6.6.3 Çap düzülmə lə ri

Mə tn (qeyri-ikili) formatlar üçün, tə lə b olunan formatda hizalanmanı göstə rə n sə tir ə ldə etmə k üçün uyğunlaşdırmada Python-un daxili format funksiyasına zə ng edə və ya formatlanmış (f-) sə tirlə rdə Alignment obyektlə rində n istifadə edə bilə rsiniz. Arqumentsiz çağırılırsa, format funksiyası hizalanmanın sə tir tə svirini qaytarır:

```
>>> str (düzlə şdirmə )
'                    1 CGGTTTTT 9\n                    0 AGGTTT-- 6\n                    0 AG-TTT-- 5\n'
>>> formatı (düzlə şdirmə )
'                    1 CGGTTTTT 9\n                    0 AGGTTT-- 6\n                    0 AG-TTT-- 5\n'
>>> çap (format (düzlə şdirmə ))
                    1 CGGTTTTT 9
                    0 AGGTTT-- 6
```

Machine Translated by Google

```
                    0 AG-TTT-- 5
<BLANKLINE>
```

Bölmə 6.7-də göstərilən formatlardan birini təyin etmək, format uyğunlaşmanı göstərən sətir yaradacaq. tələb olunan format:

```
>>> formatı (düzləşdirmə, "klastal")
'sequence_0                    CGGTTTTT\nardıcıllıq_1                    AGGTTT--\nardıcıllıq_2
>>> çap (format(düzləşdirmə, "klastal"))
sequence_0                    CGGTTTTT
sequence_1                    AGGTTT--
sequence_2                    AG-TTT--
<BLANKLINE>
<BLANKLINE>
<BLANKLINE>
>>> print(f"*** bu, Clustal formatında düzülmə dir: ***\n{alignment:clustal}\n***")
*** bu, Clustal formatında düzülmə dir: ***
ardıcıllıq_0                  CGGTTTTT
ardıcıllıq_1                  AGGTTT--
ardıcıllıq_2                  AG-TTT--
<BLANKLINE>
<BLANKLINE>
<BLANKLINE>
***
>>> formatı (düzləşdirmə, "maf")
'a\ns sequence_0 1 8 + 9 CGGTTTTT\ns sequence_1 0 6 + 6 AGGTTT--\ns sequence_2 0 5 + 7 AG-TTT--\n\n'
>>> çap (format (düzləşdirmə, "maf"))
a
s sequence_0 1 8 + 9 CGGTTTTT
s ardıcıllığı_1 0 6 + 6 AGGTTT--
s ardıcıllığı_2 0 5 + 7 AG-TTT--
<BLANKLINE>
<BLANKLINE>
```

İsteğe bağlı açar söz arqumentləri Python-un daxili format funksiyası ilə və ya formatlanmış ilə istifadə edilə bilməz strings, Alignment sinfi hizalamağı fərdiləşdirmək üçün isteğe bağlı arqumentləri olan format metoduna malikdir aşağıdakı alt bölmələrdə təsvir olunduğu kimi format. Məsələn, biz düzləşdirməni BED formatında çap edə bilərik (bax bölmə 6.7.14) müəyyən sayda sütunla:

```
>>> çap (cütlü_hizalama)
hədəf                 1 CGGTTTTT 9
                      0 .|-|||-- 8
sorğu                 0 AG-TTT-- 5
<BLANKLINE>
>>> çap (format(cüt-cüt hizalama, "yataq")) # doctest: +NORMALIZE_WHITESPACE
hədəf    1    7    sorğu    0    +    1    7    0    2    2,3,
<BLANKLINE>
>>> çap (pairwise_alignment.format("yataq")) # doctest: +NORMALIZE_WHITESPACE
hədəf    1    7    sorğu    0    +    1    7    0    2    2,3,
<BLANKLINE>
>>> çap(pairwise_alignment.format("yataq", yataqN=3)) # doctest: +NORMALIZE_WHITESPACE
hədəf    1    7
<BLANKLINE>
```

Machine Translated by Google

```
>>> çap(pairwise_alignment.format("yataq", yataqN=6)) # doctest: +NORMALIZE_WHITESPACE
hə də f 1 7 sorğu 0 +
<BLANKLINE>
```

6.7 Fayl formatlarının hizalanması

Aşağıdakı cə dvə l Bio.Align-də tə hlil edilə bilə n hizalama formatlarını göstə rir. Format arqumenti fmt Fayl formatını müə yyə n etmə k üçün Bio.Align funksiyalarında istifadə olunur. Bu fayl formatlarının ə ksə riyyə ti də ola bilə r cə dvə ldə göstə rildiyi kimi Bio.Align tə rə fində n yazılmalıdır.

Fayl formatı fmt	Tə sviri	mə tn / ikili	Yazma	Alt Bölmə si tə rə fində n	də stə klə nir
a2m	A2M		bə li		6.7.11
	Brauzer Genişlə ndirilə bilə n Mə lumat (BED)		bə li		6.7.14
çarpayısı bigbed	bigBed	mə tn	bə li		6.7.15
bigmaf	bigMaf	mə tni	bə li		6.7.19
bigpsl	bigPsl	binar	bə li		6.7.17
	UCSC zə nciri fayl zə nciri	binar ikili mə tn	bə li		6.7.20
klastal	ClustalW	mə tn	bə li		6.7.2
kabartma	QABAR	mə tn	yox		6.7.5
bə raə t qazandırmaq	bə raə t almaq	mə tn	bə li		6.7.7
fasta	Düzə ldilmiş FASTA	mə tn	bə li		6.7.1
hhr	HH-suite çıxış faylları	mə tn	yox		6.7.10
maf	Çoxlu Alignment Format (MAF)	mə tn	bə li		6.7.18
bə növşə yi	Bə növşə yi Genişlə ndirilmiş Multi-FastA (xmfa) formatı	mə tn	bə li		6.7.12
msf	GCG Çox Ardıcıllıq Formatı (MSF)	mə tn	yox		6.7.6
ə laqə	NEXUS	mə tn	bə li		6.7.8
phylip psl	PHYLIP çıxış faylları	mə tn	bə li		6.7.4
	Pattern Space Layout (PSL)	mə tn	bə li		6.7.16
sam	Ardıcıllığın Hizalanması/Xə ritə si (SAM)	mə tn	bə li		6.7.13
stokholm	Stokholm	mə tn			6.7.3
cə dvə lli	BLAST və ya FASTA-dan cə dvə l çıxışı	mə tn	yox		6.7.9

6.7.1 Aligned FASTA

Düzə ldilmiş FASTA formatındakı fayllar tam olaraq bir (cüt və ya çoxlu) ardıcıl düzülüşü saxlayır. hizalanmadakı boşluqlar tire (-) ilə tə msil olunur. Faylları düzülmüş şə kildə oxumaq və ya yazmaq üçün fmt="fasta" istifadə edin FASTA formatı. Qeyd edə k ki, bu, William Pearson-un FASTA uyğunlaşdırılması ilə yaradılan çıxışdan fə rqlidir proqram (belə çıxışın tə hlili ə və zinə bölmə 6.7.9 -da tə svir edilmişdir).

Biopython-un test paketində ki probcons.fa faylı uyğunlaşdırılmış FASTA-da bir çoxlu düzülmə ni saxlayır. format. Bu faylın mə zmunu aşağıdakı kimidir:

```
>plas_horvu
D-VLLGANGGVLVFEPNDFSVKAGETITFKNNAGYPHNVVFDEDAVPSG-VD-VSKISQEEYLTAPGETFSVTLTV---PGTYGFYCEPHAGAGMVGKVTV
>plas_chlre
--VKLGADSGALEFVPKTLTIKSGETVNFVNNAGFPHNIVFDEDAIPSG-VN-ADAISRDDYLNAPGETYSVKLTA---AGEYGYYCEPHQGAGMVGKIIV
>plas_anava
--VKLGSDKGLLVFEPAKLTIKPGDTVEFLNNKVPPHNVVFDAALNPAKSADLAKSLSHKQLLMSPGQSTSTTFPADAPAGEYTFYCEPHRGAGMVGKITV
>plas_proho
VQIKMGTDKYAPLYEPKALSISAGDTVEFVMNKVGPHNVIFDK--VPAG-ES-APALSNTKLRIAPGSFYSVTLGT---PGTYSFYCTPHRGAGMVGTITV
```

Machine Translated by Google

```
>azup_achcy
VHMLNKGKDGAMVFEPASLKVAPGDTVTFIPTDK-GHNVETIKGMIPDG-AE-A-------FKSKINENYKVTFTA---PGVYGVKCTPHYGMGMVGVVEV
```

Bu faylı oxumaq üçün istifadə edin

```
>>> Bio import Align- də n
>>> hizalama = Align.read("probcons.fa", "fasta")
>>> alignment # doctest: +ELLIPSIS
<Hazırlama obyekti (5 sə tir x 101 sütun) ...>
```

Defolt tə svirini görmə k üçün düzülüşü çap edə bilə rik:

```
>>> çap (düzlə şdirmə )
plas_horv          0 D-VLLGANGGVLVFEPNDFSVKAGETITFKNNAGYPHNVVFDEDAVPSG-VD-VSKISQE
plas_chlr          0 --VKLGADSGALEFVPKTLTIKSGETVNFVNNAGFPHNIVFDEDAIPSG-VN-ADAISRD
plas_anav          0 --VKLGSDKGLLVFEPAKLTIKPGDTVEFLNNKVPPHNVVFDAALNPAKSADLAKSLSHK
plas_proh          0 VQIKMGTDKYAPLYEPKALSISAGDTVEFVMNKVGPHNVIFDK--VPAG-ES-APALSNT
azup_achc          0 VHMLNKGKDGAMVFEPASLKVAPGDTVTFIPTDK-GHNVETIKGMIPDG-AE-A------
<BLANKLINE>
plas_horv         57 EYLTAPGETFSVTLTV---PGTYGFYCEPHAGAGMVGKVTV 95
plas_chlr         56 DYLNAPGETYSVKLTA---AGEYGYYCEPHQGAGMVGKIIV 94
plas_anav         58 QLLMSPGQSTSTTFPADAPAGEYTFYCEPHRGAGMVGKITV 99
plas_proh         56 KLRIAPGSFYSVTLGT---PGTYSFYCTPHRGAGMVGTITV 94
azup_achc         51 -FKSKINENYKVTFTA---PGVYGVKCTPHYGMGMVGVVEV 88
<BLANKLINE>
```

və ya onu düzlə nmiş FASTA formatında çap edə bilə rik:

```
>>> çap (format (düzlə şdirmə , "fasta"))
>plas_horvu
D-VLLGANGGVLVFEPNDFSVKAGETITFKNNAGYPHNVVFDEDAVPSG-VD-VSKISQEEYLTAPGETFSVTLTV---PGTYGFYCEPHAGAGMVGKVTV
>plas_chlre
--VKLGADSGALEFVPKTLTIKSGETVNFVNNAGFPHNIVFDEDAIPSG-VN-ADAISRDDYLNAPGETYSVKLTA---AGEYGYYCEPHQGAGMVGKIIV
>plas_anava
--VKLGSDKGLLVFEPAKLTIKPGDTVEFLNNKVPPHNVVFDAALNPAKSADLAKSLSHKQLLMSPGQSTSTTFPADAPAGEYTFYCEPHRGAGMVGKITV
>plas_proho
VQIKMGTDKYAPLYEPKALSISAGDTVEFVMNKVGPHNVIFDK--VPAG-ES-APALSNTKLRIAPGSFYSVTLGT---PGTYSFYCTPHRGAGMVGTITV
>azup_achcy
VHMLNKGKDGAMVFEPASLKVAPGDTVTFIPTDK-GHNVETIKGMIPDG-AE-A-------FKSKINENYKVTFTA---PGVYGVKCTPHYGMGMVGVVEV
<BLANKLINE>
```

və ya hə r hansı digə r mövcud format, mə sə lə n, Clustal (bax bölmə 6.7.2):

```
>>> çap (format (düzlə şdirmə , "klastal"))
plas_horvu         D-VLLGANGGVLVFEPNDFSVKAGETITFKNNAGYPHNVVFDEDAVPSG-
plas_chlre         --VKLGADSGALEFVPKTLTIKSGETVNFVNNAGFPHNIVFDEDAIPSG-
plas_anava         --VKLGSDKGLLVFEPAKLTIKPGDTVEFLNNKVPPHNVVFDAALNPAKS
plas_proho         VQIKMGTDKYAPLYEPKALSIAGDTVEFVMNKVGPHNVIFDK--VPAG-
azup_achcy         VHMLNKGKDGAMVFEPASLKVAPGDTVTFIPTDK-GHNVETIKGMIPDG-
<BLANKLINE>
plas_horvu         VD-VSKISQEEYLTAPGETFSVTLTV---PGTYGFYCEPHAGAGMVGKVT
plas_chlre         VN-ADAISRDDYLNAPGETYSVKLTA---AGEYGYYCEPHQGAGMVGKII
plas_anava         ADLAKSLSHKQLLMSPGQSTSTTFPADAPAGEYTFYCEPHRGAGMVGKIT
plas_proho         ES-APALSNTKLRIAPGSFYSVTLGT---PGTYSFYCTPHRGAGMVGTIT
azup_achcy         AE-A-------FKSKINENYKVTFTA---PGVYGVKCTPHYGMGMVGVVE
```

Machine Translated by Google

```
<BLANKLINE>
plas_horvu                                    V
plas_chlre                                    V
plas_anava                                    V
plas_proho                                    V
azup_achcy                                    V
<BLANKLINE>
<BLANKLINE>
<BLANKLINE>
```

Düzlə şdirmə ilə ə laqə li ardıcıllıqlar SeqRecord obyektlə ridir:

```
>>> alignment.sequences
 [SeqRecord(seq=Seq('DVLLGANGGVLVFEPNDFSVKAGETITFKNNAGYPHNVVFDEDAVPSGVDVSKI...VTV'), id='plas_horvu', na
```

Qeyd edə k ki, bu ardıcıllıqlarda boşluqlar (”-” simvolları yoxdur, çünki düzülmə mə lumatı ə və zinə koordinatlar atributunda saxlanılır:

```
>>> alignment.koordinatlar massivi([[ 0,
1, 1, 33, 34, 42, 44, 48, 48, 50, 50, 51, 58, 73, 73, 95], [ 0, 0, 0, 32, 33, 7, 4, 4, 49, 50, 57, 72, 72, 94], [ 0, 0, 0, 32, 33, 41,
        43, 47, 48, 50, 51, 52, 59, 74, 77, 99], [ 0, 1, 4, 4, 3, 2, 3 47, 49, 49, 50, 57, 72, 72, 94], [ 0, 1, 2, 34, 34, 42,
        44, 48, 48, 50, 50, 51, 51, 66, 66, 88]])
```

Bu uyğunlaşdırmanı fayla yazmaq üçün Align.write istifadə edin (burada fayl ə və zinə StringIO obyektində n istifadə edə cə yik):

```
>>> io idxalından StringIO >>> axın =
StringIO ()
>>> Align.write(alignment, stream, "FASTA")
1
>>> print(stream.getvalue()) >plas_horvu D-

VLLGANGGVLVFEPNDFSVKAGETITFKNNAGYPHNVVFDEDAVPSG-VD-VSKISQEEYLTAPGETFSVTLTV---PGTYGFYCEPHAGAGMVGKVTV >plas_chlre --

VKLGADSGALEFVPKTLTIKSGETVNFVNNAGFPHNIVFDEDAIPSG-VN-ADAISRDDYLNAPGETYSVKLTA---AGEYGYYCEPHQGAGMVGKIIV >plas_anava --

VKLGSDKGLLVFEPAKLTIKPGDTVEFLNNKVPPHNVVFDAALNPAKSADLAKSLSHKQLLMSPGQSTSTTFPADAPAGEYTFYCEPHRGAGMVGKITV >plas_proho

VQIKMGTDKYAPLYEPKALSISAGDTVEFVMNKVGPHNVIFDK--VPAG-ES-APALSNTKLRIAPGSFYSVTLGT---PGTYSFYCTPHRGAGMVGTITV >azup_achcy VHMLNKGKDGAMVFEPASLKVAPGDTVTFIPTD
GHNVETIKGMIPDG-
AE-A-------FKSKINENYKVTFTA---PGVYGVKCTPHYGMGMVGVVEV
<BLANKLINE>
```

Qeyd edə k ki, Align.write yazılmış düzülmə lə rin sayını qaytarır (bu halda 1).

6.7.2 ClustalW

Clustal hə m müstə qil proqramlar, hə m də veb serverlə r kimi mövcud olan çoxlu ardıcıllıqla uyğunlaşdırma proqramları toplusudur. opuntia.aln faylı (onlayn və ya Biopython mə nbə kodunun Doc/examples alt kataloqunda mövcuddur) Clustal tə rə fində n yaradılan çıxış faylıdır. Onun ilk bir neçə sə tiridir

CLUSTAL 2.1 çox ardıcıl düzülmə

Machine Translated by Google

```
gi|6273285|gb|AF191659.1|AF191 gi|
6273284|gb|AF191658.1|AF191 gi|6273287|
gb|AF191661.1|AF191 gi|6273286|AF19|
gb6. gi|6273290|gb|AF191664.1|AF191 gi|
6273289|gb|AF191663.1|AF191 gi|6273291|
gb|AF191665.1|AF191
```

```
TATACATTAAAGAAGGGGGATGCGGATAAATGGAAAGGCGAAAGAAAGAA
TATACATTAAAGAAGGGGGATGCGGATAAATGGAAAGGCGAAAGAAAGAA
TATACATTAAAGAAGGGGGATGCGGATAAATGGAAAGGCGAAAGAAAGAA
TATACATAAAAAGAAGGGGGATGCGGATAAATGGAAAGGCGAAAGAAAGAA
TATACATTAAAGGAGGGGGATGCGGATAAATGGAAAGGCGAAAGAAAGAA
TATACATTAAAGGAGGGGGATGCGGATAAATGGAAAGGCGAAAGAAAGAA
TATACATTAAAGGAGGGGGATGCGGATAAATGGAAAGGCGAAAGAAAGAA
******* **** ************************************
```

...

Bu faylı tə hlil etmə k üçün istifadə edin

```
>>> Bio import Align- də n
>>> alignments = Align.parse("opuntia.aln", "clustal")
```

Düzə lişlə rdə ki metadata atributu fayl başlığında göstə rilə n mə lumatları saxlayır:

```
>>> alignments.metadata
{'Proqram': 'CLUSTAL', 'Versiya': '2.1'}
```

İlk (və yeganə) hizalanmanı çıxarmaq üçün hizalamalarda növbə ti zə ng edə bilə rsiniz:

```
>>> hizalama = sonrakı (düzlə şdirmə )
>>> çap(alignment) # doctest: +ELLIPSIS
gi|627328 gi|          0 TATACATTAAAGAAGGGGGATGCGGATAAATGGAAAGGCGAAAGAAAAAAAAAAATGAAT
627328 gi|             0 TATACATTAAAGAAGGGGGATGCGGATAAATGGAAAGGCGAAAGAAAAAAAAAAATGAAT
627328 gi|             0 TATACATTAAAGAAGGGGGATGCGGATAAATGGAAAGGCGAAAGAAAAAAAAAAATGAAT
627328 gi|             0 TATACATAAAAAGAAGGGGGATGCGGATAAATGGAAAGGCGAAAGAAAAAAAAAAATGAAT
627329 gi|             0 TATACATTAAAGGAGGGGGATGCGGATAAATGGAAAGGCGAAAGAAAAAAAAAAATGAAT
627328 gi|             0 TATACATTAAAGGAGGGGGATGCGGATAAATGGAAAGGCGAAAGAAAAAAAAAAATGAAT
627329                 0 TATACATTAAAGGAGGGGGATGCGGATAAATGGAAAGGCGAAAGAAAAAAAAAAATGAAT
<BLANKLINE>
gi|627328 gi|         60 CTAAATGATATACGATTCCACTATGTAAGGTCTTTGAATCATATCATAAAAAGACAATGTA
627328 gi|            60 CTAAATGATATACGATTCCACTATGTAAGGTCTTTGAATCATATCATAAAAAGACAATGTA
627328 gi|            60 CTAAATGATATACGATTCCACTATGTAAGGTCTTTGAATCATATCATAAAAAGACAATGTA
627328                60 CTAAATGATATACGATTCCACTA...
```

Ə gə r metadata sizi maraqlandırmırsa, o zaman Align.read funksiyasından istifadə etmə k daha rahatdır hə r halda hə r bir Clustal faylı yalnız bir uyğunlaşdırma ehtiva edir:

```
>>> Bio import Align- də n
>>> alignment = Align.read("opuntia.aln", "clustal")
```

Clustal çıxış faylında hə r bir düzülmə blokunun altındakı konsensus xə ttində ulduz işarə si var. hə r mövqedə qorunur. Bu mə lumat sütun_annotasiya atributunda saxlanılır düzülmə :

```
>>> alignment.column_annotations # doctest: +ELLIPSIS
{'clustal_consensus': '************ ************************************...
```

Düzə ltmə nin klastal formatda çap edilmə si ardıcıl düzülmə ni göstə rə cə k, lakin daxil deyil metadata:

```
>>> çap (format(alignment, "clustal")) # doctest: +ELLIPSIS
gi|6273285|gb|AF191659.1|AF191 TATACATTAAAGAAGGGGGATGCGGATAAATGGAAAGGCGAAAGAAAGAA
gi|6273284|gb|AF191658.1|AF191 TATACATTAAAGAAGGGGGATGCGGATAAATGGAAAGGCGAAAGAAAGAA
gi|6273287|gb|AF191661.1|AF191 TATACATT...
```

Machine Translated by Google

Düzə lişlə rin klastal formatda yazılması hə m metadata, hə m də ardıcıllıqla düzülmə ni ə hatə edə cə k:

```
>>> io importundan StringIO >>> axın =
StringIO() >>> alignments.rewind
()
>>> Align.write(alignments, stream, "clustal") 1 >>> print(stream.getvalue()) #

doctest: +ELLIPSIS CLUSTAL 2.1 çoxsaylı ardıcıl düzülmə <BLANKLINE>


<BLANKLINE>
gi |6273285 |gb |AF191659.1 |AF191 gi |6273284 |
gb |AF191658.1 |AF191 gi |6273287 |gb |
AF191661.1 |AF191 gi |6273286 |AF19 |gb6.
TATACATTAAAGAAGGGGGATGCGGATAAATGGAAAGGCGAAAGAAAGAA
TATACATTAAAGAAGGGGGATGCGGATAAATGGAAAGGCGAAAGAAAGAA
TATACATTAAAGAAGGGGGATGCGGATAAATGGAAAGGCGAAAGAAAGAA
TATACATAAAAAGAAG...
```

Düzə lişlə ri ə l ilə yaradırsınızsa və çıxışa metadata mə lumatını daxil etmə k istə yirsinizsə , Alignments (cə m) obyektində n istifadə edin (bax: Bölmə 6.5).

6.7.3 Stokholm

Bu, PFAM tə rə fində n istifadə edilə n Stokholm fayl formatında zülal ardıcıllığının uyğunlaşdırılmasına bir nümunə dir:

```
# STOKHOLM 1.0
#=GF ID 7kD_DNA_bağlama #=GF AC
PF02294.20
#=GF DE 7kD DNT bağlayan domen #=GF AU
Mian N;0000-0003-4284-4749 #=GF AU Bateman
A;0000-0002-6982-4660 #=GF SE Pfam-B_8148 (buraxılış 5)

#=GF GA 25.00 25.00; #=GF TC 26,60
46,20; #=GF NC 23.20 19.20; #=GF BM
hmmbuild HMM.ann SEED.ann #=GF
SM hmmsearch -Z 57096847 -E 1000 --cpu 4 HMM pfamseq
#=GF TP Domain

#=GF CL CL0049
#=GF RN [1]
#=GF RM 3130377
#=GF RT DNT bağlayan zülalların 7a, 7b və 7e mikrosekans analizi #=Sulfolobus acidocaldarius archaebacterium-dan GF RT.

#=GF RA Choli T, Wittmann-Liebold B, Reinhardt R; #=GF RL J Biol Chem
1988;263:7087-7093.
#=GF DR INTERPRO; IP003212; #=GF DR
SCOP; 1sso; fa; #=GF DR SO; 0000417;
polipeptid_domen; #=GF CC Bu ailə yə hipertermofil #=GF CC
archaebacterium 7kD DNT-ni bağlayan/endoribonukleaza P2 ailə sinin üzvlə ri daxildir.

#=GF CC Hüceyrə də #=GF CC monomerlə ri kimi tapılan beş 7kD DNT bağlayıcı zülal, 7a-7e var.
Protein 7e ə n sıx DNT bağlama #=GF CC qabiliyyə tini göstə rir.

#=GF SQ 3 #=GS
DN7_METS5/4-61 AC A4YEA2.1
```

Machine Translated by Google

```
#=GS DN7A_SACS2/3-61 AC P61991.2 #=GS
DN7A_SACS2/3-61 DR PDB; 1SSO A; 2-60; #=GS DN7A_SACS2/3-61
DR PDB; 1JIC A; 2-60; #=GS DN7A_SACS2/3-61 DR PDB; 2CVR A; 2-60;
#=GS DN7A_SACS2/3-61 DR PDB; 1B4O A; 2-60; #=GS DN7E_SULAC/
3-60 AC P13125.2 DN7_METS5/4-61 DN7A_SACS2/3-61 #=GR
DN7A_SACS2/3-61 SS DN7E_SULAC/3-60 #=GC
SS_cons #=GC seq                    KIKFKYKGQDLEVDISKVKKVWKVGKMVSFTYDD.NGKTGRGAVSEKDAPKELLNMIGK
                                    TVKFKYKGEEKQVDISKIKKVWRVGKMISFTYDEGGGKTGRGAVSEKDAPKELLQMLEK
                                    EEEEESSEEEEETTTEEEESSEEEEE-SSSSEEEEEEETTTTS-CHHHHHTT
                                    KVRFKYKGEEKEVDTSKIKKVWRVGKMVSFTYDD.NGKTGRGAVSEKDAPKELMDMLAR
                                    EEEEESSEEEEETTTEEEESSEEEEE-SSSSEEEEEEETTTTS-CHHHHHTT
                                    KVKFKYKGEEKEVDISKIKKVWRVGKMVSFTYDD.NGKTGRGAVSEKDAPKELLsMLuK
```

Bu, InterPro veb saytından (https://www.ebi.ac.uk/interpro/) endirilmiş 7kD DNT bağlaması (PF02294.20) PFAM girişi üçün toxum düzülüşüdür . PFAM girişinin bu versiyası Biopython mə nbə paylamasında Tests/Stockholm/ alt kataloqunda pfam2.seed.txt faylı kimi də mövcuddur. Bu faylı aşağıdakı kimi yüklə yə bilə rik:

```
>>> Bio importdan Align >>> alignment
= Align.read("pfam2.seed.txt", " stockholm") >>> alignment # doctest: +ELLIPSIS <Alignment
obyekti (3 sıra x 59 sütun) ...>
```

Düzə lişin xülasə sini çap edə bilə rik:

```
>>> çap (düzlə şdirmə )
DN7_METS5          0 KIKFKYKGQDLEVDISKVKKVWKVGKMVSFTYDD-NGKTGRGAVSEKDAPKELLNMIGK
DN7A_SACS          0 TVKFKYKGEEKQVDISKIKKVWRVGKMISFTYDEGGGKTGRGAVSEKDAPKELLQMLEK
DN7E_SULA          0 KVRFKYKGEEKEVDTSKIKKVWRVGKMVSFTYDD-NGKTGRGAVSEKDAPKELMDMLAR
<BLANKLINE>
DN7_METS5         58
DN7A_SACS         59
DN7E_SULA         58
<BLANKLINE>
```

Siz hə mçinin onu müə yyə n bir obyektdə göstə rmə k üçün uyğunlaşdırma obyektində Python-un daxili format funksiyasına zə ng edə bilə rsiniz fayl formatı (ə traflı mə lumat üçün bölmə 6.6.3-ə baxın), mə sə lə n, faylı yenidə n yaratmaq üçün Stokholm formatında:

```
>>> çap (format(düzlə şdirmə , "stokholm"))
# STOKHOLM 1.0
#=GF ID 7kD_DNA_bağlayıcı #=GF AC
PF02294.20 #=GF DE 7kD DNT-
mə cburi domen #=GF AU Mian N;0000-0003-4284-4749
#=GF AU Bateman A;0000-0002-SEF6- Pfam-B_8148 (buraxılış 5.2)


#=GF GA 25.00 25.00; #=GF TC
26,60 46,20; #=GF NC 23.20 19.20;
#=GF BM hmmbuild HMM.ann
SEED.ann
#=GF SM hmmsearch -Z 57096847 -E 1000 --cpu 4 HMM pfamseq
#=GF TP Domeni
```

Machine Translated by Google

```
#=GF CL CL0049
#=GF RN [1]
#=GF RM 3130377
#=GF RT Archaebacterium Sulfolobus acidocaldarius olan #=GF RT-də n DNT bağlayan zülalların 7a, 7b və 7e mikrosekans analizi.

#=GF RA Choli T, Wittmann-Liebold B, Reinhardt R; #=GF RL J Biol Chem
1988;263:7087-7093.
#=GF DR INTERPRO; IP003212; #=GF DR SCOP;
1sso; fa; #=GF DR SO; 0000417;
polipeptid_domen; #=GF CC Bu ailə yə hipertermofil #=GF CC
archaebacterium 7kD DNT-ni bağlayan/endoribonukleaza P2 ailə sinin üzvlə ri daxildir. Orada #=GF
CC, #=GF CC hüceyrə sində monomerlə r kimi tapılan beş 7kD DNT bağlayıcı zülaldır, 7a-7e. Protein 7e ə n sıx DNT
bağlama qabiliyyə tini göstə rir.

#=GF SQ 3 #=GS
DN7_METS5/4-61 AC A4YEA2.1 #=GS DN7A_SACS2/3-61
AC P61991.2 #=GS DN7A_SACS2/3-61 DR PDB; 1SSO
A; 2-60; #=GS DN7A_SACS2/3-61 DR PDB; 1JIC A; 2-60; #=GS
DN7A_SACS2/3-61 DR PDB; 2CVR A; 2-60; #=GS DN7A_SACS2/3-61
DR PDB; 1B4O A; 2-60; #=GS DN7E_SULAC/3-60 AC P13125.2
DN7_METS5/4-61 DN7A_SACS2/3-61 #=GR DN7A_SACS2/3-61 SS
DN7E_SULAC/3-60 #=GC SS_cons #=GC seq
                                                KIKFKYKGQDLEVDISKVKKVWKVGKMVSFTYDD.NGKTGRGAVSEKDAPKELLNMIGK
                                                TVKFKYKGEEKQVDISKIKKVWRVGKMISFTYDEGGGKTGRGAVSEKDAPKELLQMLEK
                                                EEEEESSEEEEETTTEEEESSEEEEEE-SSSSEEEEEEETTTTS-CHHHHHTT
                                                KVRFKYKGEEKEVDTSKIKKVWRVGKMVSFTYDD.NGKTGRGAVSEKDAPKELMDMLAR
                                                EEEEESSEEEEETTTEEEESSEEEEEE-SSSSEEEEEEETTTTS-CHHHHHTT
                                                KVKFKYKGEEKEVDISKIKKVWRVGKMVSFTYDD.NGKTGRGAVSEKDAPKELLsMLuK

<BLANKLINE>
```

və ya alternativ olaraq uyğunlaşdırılmış FASTA kimi (bax bölmə 6.7.1):

```
>>> çap (format (düzlə şdirmə , "fasta"))
>DN7_METS5/4-61
KIKFKYKGQDLEVDISKVKKVWKVGKMVSFTYDD-NGKTGRGAVSEKDAPKELLNMIGK
>DN7A_SACS2/3-61
TVKFKYKGEEKQVDISKIKKVWRVGKMISFTYDEGGGKTGRGAVSEKDAPKELLQMLEK
>DN7E_SULAC/3-60
KVRFKYKGEEKEVDTSKIKKVWRVGKMVSFTYDD-NGKTGRGAVSEKDAPKELMDMLAR
<BLANKLINE>
```

və ya PHYLIP formatında (bax bölmə 6.7.4):

```
>>> çap (format(alignment, "phylip")) 3 59

DN7_METS5/KIKFKYKGQDLEVDISKVKKVWKVGKMVSFTYDD-NGKTGRGAVSEKDAPKELLNMIGK
DN7A_SACS2TVKFKYKGEEKQVDISKIKKVWRVGKMISFTYDEGGGKTGRGAVSEKDAPKELLQMLEK
DN7E_SULACKVRFKYKGEEKEVDTSKIKKVWRVGKMVSFTYDD-NGKTGRGAVSEKDAPKELMDMLAR
<BLANKLINE>
```

Hizalanma haqqında ümumi mə lumat, mə sə lə n, Alignment obyektinin annotasiya atributunda saxlanılır

Machine Translated by Google

```
>>> alignment.annotations["identifier"] '7kD_DNA_binding'
 >>>
alignment.annotations["klan"]
 'CL0049'
>>> alignment.annotations["verilə nlə r bazasına istinadlar"] [{'reference':
 'INTERPRO; IPR003212;'}, {'istinad': 'SCOP; 1sso; fa;'}, {'reference': 'SO; 0000417; sə h
```

Bu düzülmə də ki fə rdi ardıcıllıqlar alignment.sequences altında SeqRecords kimi saxlanılır, o cümlə də n hə r bir ardıcıllıq qeydi ilə ə laqə li annotasiyalar:

```
>>> alignment.sequences -də qeyd üçün : print("%s %s %s"
 ...              % (record.id, record.annotations["accession"], record.dbxrefs))
 ...
DN7_METS5/4-61 A4YEA2.1 []
DN7A_SACS2/3-61 P61991.2 ['PDB; 1SSO A; 2-60;', 'PDB; 1JIC A; 2-60;', 'PDB; 2CVR A; 2-60;', 'PDB; 1B4O DN7E_SULAC/3-60 P13125.2 []
```

İkinci ardıcıllığın ikinci strukturu (DN7A_SACS2/3-61) SeqRecord-un letter_annotations atributunda saxlanılır:

```
>>> alignment.sequences[0].letter_annotations {} >>>

alignment.sequences[1].letter_annotations {'ikinci struktur':
'EEEEESSSSEEEEETTTEEEEEESSSSEEEEEEE-SSSSEEEEEEETTTTS-CHHHHHHTT'} >>> alignment.sequences[1 ]
```

Konsensus ardıcıllığı və ikinci də rə cə li struktur bütövlükdə ardıcıl düzülmə ilə ə laqə lə ndirilir və buna görə də Alignment obyektinin column_annotations atributunda saxlanılır:

```
>>> alignment.column_annotations # doctest: +NORMALIZE_WHITESPACE {'konsensusun ikinci də rə cə li
strukturu': 'EEEEESSSSEEEEETTTEEEEEESSSSEEEEEEE-SSSSEEEEEEETTTTS-CHHHHHHTT',
   'konsensus ardıcıllığı': 'KVKFKYKGEEKEVDISKIKKVWRVGKMVSFTYDD.NGKTGRGAVSEKDAPKELLsMLuK'}
```

6.7.4 PHYLIP çıxış faylları

Ardıcıl düzülmə lə r üçün PHYLIP formatı Joe Felsenstein tə rə fində n PHYLogeny müdaxilə paketində n ə ldə edilmişdir. PHYLIP formatında fayllar çap edilmiş düzülmə də sə tir və sütunların sayı üçün iki rə qə mlə başlayır. Ardıcıllıqla düzülmə nin özü ardıcıl formatda və ya interleaved formatında ola bilə r.

Birincisinə misal olaraq sequential.phy faylıdır (Biopython mə nbə paylanmasında Tests/Phylip/-də verilmişdir):

```
  3 384
CYS1_DICDI -----MKVIL LFVLAVFTVF VSS------- --------RG IPPEEQ---- --------SQ
                    FLEFQDKFNK KY-SHEEYLE RFEIFKSNLG KIEELNLIAI NHKADTKFGV NKFADLSSDE
                    FKNYYLNNKE AIFTDDLPVA DYLDDEFINS IPTAFDWRTR G-AVTPVKNQ GQCGSCWSFS
                    TTGNVEGQHF ISQNKLVSLS EQNLVDCDHE CMEYEGEEAC DEGCNGGLQP NAYNYIIKNG
                    GIQTESSYPY TAETGTQCNF NSANIGAKIS NFTMIP-KNE TVMAGYIVST GPLAIAADAV
                    E-WQFYIGGV F-DIPCN--P NSLDHGILIV GYSAKNTIFR KNMPYWIVKN SWGADWGEQG
                    YIYLRRGKNT CGVSNFVSTS II--
ALEU_HORVU MAHARVLLLA LAVLATAAVA VASSSSFADS NPIRPVTDRA ASTLESAVLG ALGRTRHALR
                    FARFAVRYGK SYESAAEVRR RFRIFSESLE EVRSTN---- RKGLPYRLGI NRFSDMSWEE
                    FQATRL-GAA QTCSATLAGN HLMRDA--AA LPETKDWRED G-IVSPVKNQ AHCGSCWTFS
                    TTGALEAAYT QATGKNISLS EQQLVDCAGG FNNF------ --GCNGGLPS QAFEYIKYNG
```

Machine Translated by Google

```
                    GIDTEESYPY KGVNGV-CHY KAENAAVQVL DSVNITLNAE DELKNAVGLV RPVSVAFQVI
                    DGFRQYKSGV YTSDHCGTTP DDVNHAVLAV GYGVENGV-- ---PYWLIKN SWGADWGDNG
                    YFKMEMGKNM CAIATCASYP VVAA
CATH_HUMAN ------MWAT LPLLCAGAWL LGV------- -PVCGAAELS VNSLEK---- --------FH
                    FKSWMSKHRK TY-STEEYHH RLQTFASNWR KINAHN---- NGNHTFKMAL NQFSDMSFAE
                    IKHKYLWSEP QNCSAT--KS NYLRGT--GP YPPSVDWRKK GNFVSPVKNQ GACGSCWTFS
                    TTGALESAIA IATGKMLSLA EQQLVDCAQD FNNY------ --GCQGGLPS QAFEYILYNK
                    GIMGEDTYPY QGKDGY-CKF QPGKAIGFVK DVANITIYDE EAMVEAVALY NPVSFAFEVT
                    QDFMMYRTGI YSSTSCHKTP DKVNHAVLAV GYGEKNGI-- ---PYWIVKN SWGPQWGMNG
                    YFLIERGKNM CGLAACASYP IPLV
```

Ardıcıl formatda, növbə ti ardıcıllığa keçmə zdə n ə vvə l bir ardıcıllığın tam uyğunlaşdırılması göstə rilir. Interleaved formatında, müxtə lif ardıcıllıqlar üçün düzülmə lə r bir-birinin yanındadır, mə sə lə n, interlaced.phy faylında (Biopython mə nbə paylanmasında Tests/Phylip/-də verilmişdir):

```
  3 384
CYS1_DICDI -----MKVIL LFVLAVFTVF VSS------- --------RG IPPEEQ---- --------SQ
ALEU_HORVU MAHARVLLLA LAVLATAAVA VASSSSFADS NPIRPVTDRA ASTLESAVLG ALGRTRHALR
CATH_HUMAN ------MWAT LPLLCAGAWL LGV------- -PVCGAAELS VNSLEK---- --------FH

                    FLEFQDKFNK KY-SHEEYLE RFEIFKSNLG KIEELNLIAI NHKADTKFGV NKFADLSSDE
                    FARFAVRYGK SYESAAEVRR RFRIFSESLE EVRSTN---- RKGLPYRLGI NRFSDMSWEE
                    FKSWMSKHRK TY-STEEYHH RLQTFASNWR KINAHN---- NGNHTFKMAL NQFSDMSFAE

                    FKNYYLNNKE AIFTDDLPVA DYLDDEFINS IPTAFDWRTR G-AVTPVKNQ GQCGSCWSFS
                    FQATRL-GAA QTCSATLAGN HLMRDA--AA LPETKDWRED G-IVSPVKNQ AHCGSCWTFS
                    IKHKYLWSEP QNCSAT--KS NYLRGT--GP YPPSVDWRKK GNFVSPVKNQ GACGSCWTFS

                    TTGNVEGQHF ISQNKLVSLS EQNLVDCDHE CMEYEGEEAC DEGCNGGLQP NAYNYIIKNG
                    TTGALEAAYT QATGKNISLS EQQLVDCAGG FNNF------ --GCNGGLPS QAFEYIKYNG
                    TTGALESAIA IATGKMLSLA EQQLVDCAQD FNNY------ --GCQGGLPS QAFEYILYNK

                    GIQTESSYPY TAETGTQCNF NSANIGAKIS NFTMIP-KNE TVMAGYIVST GPLAIAADAV
                    GIDTEESYPY KGVNGV-CHY KAENAAVQVL DSVNITLNAE DELKNAVGLV RPVSVAFQVI
                    GIMGEDTYPY QGKDGY-CKF QPGKAIGFVK DVANITIYDE EAMVEAVALY NPVSFAFEVT

                    E-WQFYIGGV F-DIPCN--P NSLDHGILIV GYSAKNTIFR KNMPYWIVKN SWGADWGEQG
                    DGFRQYKSGV YTSDHCGTTP DDVNHAVLAV GYGVENGV-- ---PYWLIKN SWGADWGDNG
                    QDFMMYRTGI YSSTSCHKTP DKVNHAVLAV GYGEKNGI-- ---PYWIVKN SWGPQWGMNG

                    YIYLRRGKNT CGVSNFVSTS II--
                    YFKMEMGKNM CAIATCASYP VVAA
                    YFLIERGKNM CGLAACASYP IPLV
```

Bio.Align-də ki tə hlilçi fayl mə zmunundan onun ardıcıl və ya interleaved formatda olub-olmadığını müə yyə n edir və sonra onu müvafiq şə kildə tə hlil edir.

```
>>> Bio importdan Align >>>
alignment = Align.read("sequential.phy", "phylip") >>> alignment # doctest:
+ELLIPSIS <Alignment obyekti (3 sıra x 384
sütun) ...> >>> alignment2 = Align.read ("interlaced.phy", " phylip2
") doELL #IP:test
```

Machine Translated by Google

```
<Hizalanma obyekti (3 sə tir x 384 sütun) ...>
>>> hizalama == hizalama2
Doğrudur
```

Burada iki düzülmə eyni ardıcıllıq mə zmununa malikdirsə və eynidirsə bə rabə r sayılır düzülmə koordinatları.

```
>>> hizalama.forma
(3, 384)
>>> çap (düzlə şdirmə )
CYS1_DICD          0 -----MKVILLFVLAVFTVFVSS---------------RGIPPEEQ------------SQ
ALEU_HORV          0 MAHARVLLLALAVLATAAVAVASSSSFADSNPIRPVTDRAASTLESAVLGALGRTRHALR
CATH_HUMA          0 ------MWATLPLLCAGAWLLGV--------PVCGAAELSVNSLEK------------FH
<BLANKLINE>
CYS1_DICD         28 FLEFQDKFNKKY-SHEEYLERFEIFKSNLGKIEELNLIAINHKADTKFGVNKFADLSSDE
ALEU_HORV         60 FARFAVRYGKSYESAAEVRRRFRIFSESLEEVRSTN----RKGLPYRLGINRFSDMSWEE
CATH_HUMA         34 FKSWMSKHRKTY-STEEYHHRLQTFASNWRKINAHN----NGNHTFKMALNQFSDMSFAE
<BLANKLINE>
CYS1_DICD         87 FKNYYLNNKEAIFTDDLPVADYLDDEFINSIPTAFDWRTRG-AVTPVKNQGQCGSCWSFS
ALEU_HORV        116 FQATRL-GAAQTCSATLAGNHLMRDA--AALPETKDWREDG-IVSPVKNQAHCGSCWTFS
CATH_HUMA         89 IKHKYLWSEPQNCSAT--KSNYLRGT--GPYPPSVDWRKKGNFVSPVKNQGACGSCWTFS
<BLANKLINE>
CYS1_DICD        146 TTGNVEGQHFISQNKLVSLSEQNLVDCDHECMEYEGEEACDEGCNGGLQPNAYNYIIKNG
ALEU_HORV        172 TTGALEAAYTQATGKNISLSEQQLVDCAGGFNNF--------GCNGGLPSQAFEYIKYNG
CATH_HUMA        145 TTGALESAIAIATGKMLSLAEQQLVDCAQDFNNY--------GCQGGLPSQAFEYILYNK
<BLANKLINE>
CYS1_DICD        206 GIQTESSYPYTAETGTQCNFNSANIGAKISNFTMIP-KNETVMAGYIVSTGPLAIAADAV
ALEU_HORV        224 GIDTEESYPYKGVNGV-CHYKAENAAVQVLDSVNITLNAEDELKNAVGLVRPVSVAFQVI
CATH_HUMA        197 GIMGEDTYPYQGKDGY-CKFQPGKAIGFVKDVANITIYDEEAMVEAVALYNPVSFAFEVT
<BLANKLINE>
CYS1_DICD        265 E-WQFYIGGVF-DIPCN--PNSLDHGILIVGYSAKNTIFRKNMPYWIVKNSWGADWGEQG
ALEU_HORV        283 DGFRQYKSGVYTSDHCGTTPDDVNHAVLAVGYGVENGV-----PYWLIKNSWGADWGDNG
CATH_HUMA        256 QDFMMYRTGIYSSTSCHKTPDKVNHAVLAVGYGEKNGI-----PYWIVKNSWGPQWGMNG
<BLANKLINE>
CYS1_DICD        321 YIYLRGKNTCGVSNFVSTSII-- 343
ALEU_HORV        338 YFKMEMGKNMCAIATCASYPVVAA 362
CATH_HUMA        311 YFLIERGKNMCGLAACASYPIPLV 335
<BLANKLINE>
```

Düzə ltmə ni PHYLIP formatında çıxararkə n Bio.Align düzülmüş ardıcıllığın hə r birini birində yazır. xə tt:

```
>>> çap (format (düzlə şdirmə , "filip"))
3 384
CYS1_DICDI-----MKVILLFVLAVFTVFVSS---------------RGIPPEEQ------------SQFLEFQDKFNKKY-SHEEYLERFEIFKSNLGKIE
ALEU_HORVUMAHARVLLLALAVLATAAVAVASSSFADSNPIRPVTDRAASTLESAVLGALGRTRHALRFARFAVRYGKSYESAAEVRRRFRIFSESLEEVR
CATH_HUMAN------MWATLPLLCAGAWLLGV--------PVCGAAELSVNSLEK------------FHFKSWMSKHRKTY-STEEYHHRLQTFASNWRKIN
<BLANKLINE>
```

Hizalanmanı PHYLIP formatında yaza, nə ticə ni tə hlil edə və onun orijinalla eyni olduğunu tə sdiqlə yə bilə rik. hizalama obyekti:

```
>>> io import StringIO- dan
>>> axın = StringIO()
```

Machine Translated by Google

```
>>> Align.write(alignment, stream, "phylip")
1
>>> stream.seek(0)
0
>>> alignment3 = Align.read(axın, "phylip")
>>> hizalama == hizalama3
Doğrudur
>>> [alignment.sequences -də  qeyd üçün rekord.id ]
['CYS1_DICDI', 'ALEU_HORVU', 'CATH_HUMAN']
>>> [alignment3.sequences -də  qeyd üçün rekord.id ]
['CYS1_DICDI', 'ALEU_HORVU', 'CATH_HUMAN']
```

6.7.5 QARŞIQ

EMBOSS (Avropa Molekulyar Biologiya Açıq Proqram Tə minatı) üçün açıq mə nbə li proqram alə tlə ri toplusudur. molekulyar biologiya və bioinformatika [37]. Buraya cütlük üçün iynə və su kimi proqramlar daxildir ardıcıllıqla düzülmə . Bu, Smith-Waterman üçün su proqramı tə rə fində n yaradılan çıxış nümunə sidir yerli cüt ardıcıl düzülmə (Biopython-un Testlə r/Qabartma kataloqunda water.txt kimi mövcuddur) paylanması):

```
########################################
# Proqram: su
# İş vaxtı: Çə rşə nbə 16 Yanvar 17:23:19 2002
# Hesabat_faylı: stdout
########################################
#=======================================
#
# Hizalanmış_ardıcıllıqlar: 2
№ 1: IXI_234
№ 2: IXI_235
# Matris: EBLOSUM62
# Boşluq_cə zası: 10.0
# Cə zanın uzadılması: 0.5
#
# Uzunluq: 131
# Şə xsiyyə t: #          112/131 (85,5%)
Oxşarlıq: 112/131 (85,5%)
# Boşluqlar:              19/131 (14,5%)
# Xal: 591.5
#
#
#=======================================

IXI_234                  1 TSPASIRPPAGPSSRPAMVSSRRTRPSPPGPRRPTGRPCCSAAPRRPQAT    50
                           ||||||||||||||| ||||||||||||||||||||||||||
IXI_235                  1 TSPASIRPPAGPSSR---------RPSPPGPRRPTGRPCCSAAPRRPQAT    41

IXI_234                 51 GGWKTCSGTCTTSSTTRHRGRSGWSARTTTAACLRASRKSMRAACSRSAG   100
                           |||||||||||||||||||||||||                |||||||||||||||||
IXI_235                 42 GGWKTCSGTCTTSSTTRHRGRSGW----------RASRKSMRAACSRSAG    81

IXI_234                101 SRPNRFAPTLMSSCITSTTGPAWAGDRSHE                 131
```

Machine Translated by Google

```
                                  |||||||||||||||||||||||||||||||
IXI_235                    82 SRPNRFAPTLMSSCITSTTGPAWAGDRSHE                112


#-------------------------------------
#-------------------------------------
```

Bu çıxış faylında yalnız bir düzülmə olduğundan, onu birbaşa çıxarmaq üçün Align.read istifadə edə bilə rik. Bunun ə və zinə biz Align.parse istifadə edə cə yik ki, bu su axınının metadatasını görə bilə k:

```
>>> Bio idxaldan Align >>> alignments
= Align.parse ("su.txt", "qabartma")
```

Düzə lişlə rin metadata atributu faylın başlığında göstə rilə n mə lumatları, o cümlə də n çıxışı yaratmaq üçün istifadə olunan proqram, proqramın işlə dildiyi tarix və vaxt, çıxış faylının adı və istifadə edilmiş xüsusi uyğunlaşdırma faylı formatını saxlayır (defolt olaraq srspair olduğu güman edilir):

```
>>> alignments.metadata
{'Align_format': 'srspair', 'Proqram': 'su', 'Rundate': 'Çə rşə nbə 16 Yanvar 17:23:19 2002', 'Report_file': '
```

Hizalanmanı çıxarmaq üçün istifadə edirik

```
>>> alignment = next(alignments) >>>
alignment # doctest: +ELLIPSIS <Alignment obyekti
(2 sıra x 131 sütun) ...> >>> alignment.shape (2, 131) >>> print(alignment)



IXI_234                     0 TSPASIRPPAGPSSRPAMVSSRRTRPSPPGPRRPTGRPCCSAAPRQATGGWKTCSGTC 0
                            |||||||||||||||||||||||||||||---------||||||||||||||||||||||||||||||||||||
IXI_235                     0 TSPASIRPPAGPSSR---------RPSPPGPRRPTGRPCCSAAPRRPQATGGWKTCSGTC
<BLANKLINE>
IXI_234                    60 TTTSTRHRGRSGWSARTTTAACLRASRKSMRAACSRSAGSRPNRFAPTLMSSCITSTTTG
                           60 |||||||||||||||----------||||||||||||||||||||||||||||||||||||||||||
IXI_235                    51 TTTSTRHRGRSGW----------RASRKSMRAACSRSAGSRPNRFAPTLMSSCITSTTTG
<BLANKLINE>
IXI_234                   120 PPAWAGDRSHE 131
                          120 ||||||||||| 131
IXI_235                   101 PPAWAGDRSHE 112
<BLANKLINE>
>>> alignment.koordinatlar
massivi([[ 0, 15, 24, 74, 84, 131],
         [ 0, 15, 15, 65, 65, 112]])
```

Hizalanmanın müə yyə n hissə lə rini çıxarmaq üçün indekslə rdə n istifadə edə bilə rik:

```
>>> düzülmə [0]
 'TSPASIRPPAGPSSRPAMVSSRRTRPSPPGPRRPTGRPCCSAAPRRPQATGGWKTCSGTCTTSTSTRHRGRSGWSARTTTAACLRASRKSMRAACSRSAGSR >>>
uyğunlaşdırma[1]
 'TSPASIRPPAGPSSR---------RPSPPGPRRPTGRPCCSAAPRQATGGWKTCSGTCTTSTSTRHRGRSGW----------RASRKSMRAACSRSAGSR >>> alignment[1, 10:30]

 'GPSSR---------RPSPPG'
```

Düzə lişin annotasiya atributu bu düzülmə ilə ə laqə li mə lumatları xüsusi olaraq saxlayır:

Machine Translated by Google

```
>>> alignment.annotations
{'Matrix': 'EBLOSUM62', 'Gap_penalty': 10.0, 'Geniş_cə rimə ': 0.5, 'Şə xsiyyə t': 112, 'Oxşarlıq': 112,
```

Boşluqların, identifikasiyaların və uyğunsuzluqların sayını hə mçinin hesablama metoduna zə ng etmə klə də ə ldə etmə k olar. hizalama obyekti:

```
>>> alignment.counts()
AlignmentCounts(boşluqlar=19, şə xsiyyə tlə r=112, uyğunsuzluqlar=0)
```

burada AlignmentCounts, Python-un standart kitabxanasındakı kolleksiyalar modulunda adlandırılmış nüsxə dir.

İki ardıcıllıq arasında göstə rilə n konsensus xə tti column_annotations atributunda saxlanılır:

```
>>> alignment.column_annotations
{'emboss_consensus': '|||||||||||||||                                    |||||||||||||||||||||||||||||||||||||||||||||||||||||||||
```

Hizalanmanı digə r formatlarda, mə sə lə n, formatda çap etmə k üçün format funksiyasından (və ya format metodundan) istifadə edin PHYLIP formatı (bax bölmə 6.7.4):

```
>>> çap (format (düzlə şdirmə , "filip"))
2 131
IXI_234 TSPASIRPPAGPSSRPAMVSSRRTRPSPPGPRRPTGRPCCSAAPRRPQATGGWKTCSGTCTTSTSTRHRGRSGWSARTTTAACLRASRKSMRA
IXI_235 TSPASIRPPAGPSSR---------RPSPPGPRRPTGRPCCSAAPRRPQATGGWKTCSGTCTTSTSTRHRGRSGW----------RASRKSMRA
<BLANKLINE>
```

Fə rdi ardıcıllıqları ə ldə etmə k üçün alignment.sequences istifadə edə bilə rik. Lakin bu, ikili düzülmə olduğundan, hə də f və sorğu ardıcıllığını ə ldə etmə k üçün alignment.target və alignment.query istifadə edə bilə rik:

```
>>> hizalama.hə də f
SeqRecord(seq=Seq('TSPASIRPPAGPSSRPAMVSSRRTRPSPPGPRRPTGRPCCSAAPRQATGGWK...SHE'), id='IXI_234', name=' >>> alignment.query

SeqRecord(seq=Seq('TSPASIRPPAGPSSRRPSPPGPRRPTGRPCCSAAPRRPQATGGWKTCSGTCTTS...SHE'), id='IXI_235', ad='
```

Hal-hazırda, Biopython EM-BOSS tə rə fində n müə yyə n edilmiş çıxış formatlarında yazı ardıcıllığının uyğunlaşdırılmasını də stə klə mir.

6.7.6 GCG Çox Ardıcıllıq Formatı (MSF)

Çox Ardıcıllıq Formatı (MSF) tə rə fində n yaradılan çoxsaylı ardıcıl düzülmə lə ri saxlamaq üçün yaradılmışdır GCG (Genetics Computer Group) proqramlar toplusu. W_prot.msf faylı Tests/msf qovluğunda Biopython paylanması MSF formatında ardıcıl düzülmə faylına bir nümunə dir Bu fayl bir 11 protein ardıcıllığının uyğunlaşdırılması:

```
!!AA_ÇOXLU_HIZLI

    MSF: 99 Növ: P 18 oktyabr 2017 11:35 Yoxla: 0 ..

 Adı: W*01:01:01:01          Len:    99 Yoxlayın: 7236 Çə ki: 1.00
 Adı: W*01:01:01:02          Len:    99 Yoxlayın: 7236 Çə ki: 1.00
 Adı: W*01:01:01:03          Len:    99 Yoxlayın: 7236 Çə ki: 1.00
 Adı: W*01:01:01:04          Len:    99 Yoxlayın: 7236 Çə ki: 1.00
 Adı: W*01:01:01:05          Len:    99 Yoxlayın: 7236 Çə ki: 1.00
 Adı: W*01:01:01:06          Len:    99 Yoxlayın: 7236 Çə ki: 1.00
 Adı: W*02:01                Len:    93 Yoxlama: 9483 Çə ki: 1.00
 Adı: W*03:01:01:01          Len:    93 Yoxlama: 9974 Çə ki: 1.00
 Adı: W*03:01:01:02          Len:    93 Yoxlama: 9974 Çə ki: 1.00
```

Machine Translated by Google

```
Adı: W*04:01                    Len:      93 Yoxlayın: 9169 Çə ki: 1.00
Adı: W*05:01 //                 Len:      99 Yoxlayın: 7331 Çə ki: 1.00


  W*01:01:01:01 GLTPFNGYTA ATWTRTAVSS VGMNIPYHGA SYLVRNQELR SWTAADKAAQ
  W*01:01:01:02 GLTPFNGYTA ATWTRTAVSS VGMNIPYHGA SYLVRNQELR SWTAADKAAQ
  W*01:01:01:03 GLTPFNGYTA ATWTRTAVSS VGMNIPYHGA SYLVRNQELR SWTAADKAAQ
  W*01:01:01:04 GLTPFNGYTA ATWTRTAVSS VGMNIPYHGA SYLVRNQELR SWTAADKAAQ
  W*01:01:01:05 GLTPFNGYTA ATWTRTAVSS VGMNIPYHGA SYLVRNQELR SWTAADKAAQ
  W*01:01:01:06 GLTPFNGYTA ATWTRTAVSS VGMNIPYHGA SYLVRNQELR SWTAADKAAQ
            W*02:01 GLTPSNGYTA ATWTRTAASS VGMNIPYDGA SYLVRNQELR SWTAADKAAQ
  W*03:01:01:01 GLTPSSGYTA ATWTRTAVSS VGMNIPYHGA SYLVRNQELR SWTAADKAAQ
  W*03:01:01:02 GLTPSSGYTA ATWTRTAVSS VGMNIPYHGA SYLVRNQELR SWTAADKAAQ
            W*04:01 GLTPSNGYTA ATWTRTAASS VGMNIPYDGA SYLVRNQELR SWTAADKAAQ
            W*05:01 GLTPSSGYTA ATWTRTAVSS VGMNIPYHGA SYLVRNQELR SWTAADKAAQ

  W*01:01:01:01 MPWRRNRQSC SKPTCREGGR SGSAKSLRMG RRGCSAQNPK DSHDPPPHL
  W*01:01:01:02 MPWRRNRQSC SKPTCREGGR SGSAKSLRMG RRGCSAQNPK DSHDPPPHL
  W*01:01:01:03 MPWRRNRQSC SKPTCREGGR SGSAKSLRMG RRGCSAQNPK DSHDPPPHL
  W*01:01:01:04 MPWRRNRQSC SKPTCREGGR SGSAKSLRMG RRGCSAQNPK DSHDPPPHL
  W*01:01:01:05 MPWRRNRQSC SKPTCREGGR SGSAKSLRMG RRGCSAQNPK DSHDPPPHL
  W*01:01:01:06 MPWRRNRQSC SKPTCREGGR SGSAKSLRMG RRGCSAQNPK DSHDPPPHL
            W*02:01 MPWRRNMQSC SKPTCREGGR SGSAKSLRMG RRRCTAQNPK RLT
  W*03:01:01:01 MPWRRNRQSC SKPTCREGGR SGSAKSLRMG RRGCSAQNPK RLT
  W*03:01:01:02 MPWRRNRQSC SKPTCREGGR SGSAKSLRMG RRGCSAQNPK RLT
            W*04:01 MPWRRNMQSC SKPTCREGGR SGSAKSLRMG RRGCSAQNPK RLT
            W*05:01 MPWRRNRQSC SKPTCREGGR SGSAKSLRMG RRGCSAQNPK DSHDPPPHL
```

Bu faylı Biopython ilə tə hlil etmə k üçün istifadə edin

```
>>> Bio import Align- də n
>>> hizalama = Align.read("W_prot.msf", "msf")
```

Parser "MSF:" ilə başlayan sə tir daxil olmaqla, bütün sə tirlə ri ötür. Aşağıdakı sə tirlə r ("//" demarkasiyası) hə r bir ardıcıllığın uzunluğunu yoxlamaq üçün tə hlilçi tə rə fində n oxunur. Hizalama bölmə si (sonra "//" demarkasiyası) tə hlilçi tə rə fində n oxunur və Alignment obyekti kimi saxlanılır:

```
>>> alignment # doctest: +ELLIPSIS
<Hazırlama obyekti (11 sə tir x 99 sütun) ...>
>>> çap (düzlə şdirmə )
W*01:01:0          0 GLTPFNGYTAATWTRTAVSSVGMNIPYHGASYLVRNQELRSWTAADKAAQMPWRRNRQSC
W*01:01:0          0 GLTPFNGYTAATWTRTAVSSVGMNIPYHGASYLVRNQELRSWTAADKAAQMPWRRNRQSC
W*01:01:0          0 GLTPFNGYTAATWTRTAVSSVGMNIPYHGASYLVRNQELRSWTAADKAAQMPWRRNRQSC
W*01:01:0          0 GLTPFNGYTAATWTRTAVSSVGMNIPYHGASYLVRNQELRSWTAADKAAQMPWRRNRQSC
W*01:01:0          0 GLTPFNGYTAATWTRTAVSSVGMNIPYHGASYLVRNQELRSWTAADKAAQMPWRRNRQSC
W*01:01:0          0 GLTPFNGYTAATWTRTAVSSVGMNIPYHGASYLVRNQELRSWTAADKAAQMPWRRNRQSC
W*02:01            0 GLTPSNGYTAATWTRTAASSVGMNIPYDGASYLVRNQELRSWTAADKAAQMPWRRNMQSC
W*03:01:0          0 GLTPSSGYTAATWTRTAVSSVGMNIPYHGASYLVRNQELRSWTAADKAAQMPWRRNRQSC
W*03:01:0          0 GLTPSSGYTAATWTRTAVSSVGMNIPYHGASYLVRNQELRSWTAADKAAQMPWRRNRQSC
W*04:01            0 GLTPSNGYTAATWTRTAASSVGMNIPYDGASYLVRNQELRSWTAADKAAQMPWRRNMQSC
W*05:01            0 GLTPSSGYTAATWTRTAVSSVGMNIPYHGASYLVRNQELRSWTAADKAAQMPWRRNRQSC
<BLANKLINE>
W*01:01:0         60 SKPTCREGGRSGSAKSLRMGRRGCSAQNPKDSHDPPPHL 99
```

Machine Translated by Google

```
W*01:01:0          60 SKPTCREGGRSGSAKSLRMGRRGCSAQNPKDSHDPPPHL 99
W*01:01:0          60 SKPTCREGGRSGSAKSLRMGRRGCSAQNPKDSHDPPPHL 99
W*01:01:0          60 SKPTCREGGRSGSAKSLRMGRRGCSAQNPKDSHDPPPHL 99
W*01:01:0          60 SKPTCREGGRSGSAKSLRMGRRGCSAQNPKDSHDPPPHL 99
W*01:01:0          60 SKPTCREGGRSGSAKSLRMGRRGCSAQNPKDSHDPPPHL 99
W*02:01            60 SKPTCREGGRSGSAKSLRMGRRRCTAQNPKRLT------ 93
W*03:01:0          60 SKPTCREGGRSGSAKSLRMGRRGCSAQNPKRLT------ 93
W*03:01:0          60 SKPTCREGGRSGSAKSLRMGRRGCSAQNPKRLT------ 93
W*04:01            60 SKPTCREGGRSGSAKSLRMGRRGCSAQNPKRLT------ 93
W*05:01            60 SKPTCREGGRSGSAKSLRMGRRGCSAQNPKDSHDPPPHL 99
<BLANKLINE>
```

Ardıcıllıqlar və onların adları alignment.sequences atributunda saxlanılır:

```
>>> len(alignment.sequences)
11
>>> alignment.sequences[0].id
 'W*01:01:01:01'
>>> alignment.sequences[0].seq
Seq('GLTPFNGYTAATWTRTAVSSVGMNIPYHGASYLVRNQELRSWTAADKAAQMPWR...PHL')
```

Hizalama koordinatları alignment.coordinates atributunda saxlanılır:

```
>>> hizalama.koordinatlar
massiv([[ 0, 93, 99],
        [ 0, 93, 99],
        [ 0, 93, 99],
        [ 0, 93, 99],
        [ 0, 93, 99],
        [ 0, 93, 99],
        [ 0, 93, 93],
        [ 0, 93, 93],
        [ 0, 93, 93],
        [ 0, 93, 93],
        [ 0, 93, 99]])
```

Hal-hazırda, Biopython MSF formatında ardıcıl düzülmə lə rin yazılmasını də stə klə mir.

6.7.7 Bə raə tdə n azad etmə k

Exonerate cüt ardıcıllıqla düzülmə üçün ümumi proqramdır [42]. Ardıcıllıq düzülmə lə ri tapıldı
Exonerate tə rə fində n "siqar" (Compact Idiosyncratic Gapped) insan tə rə fində n oxuna bilə n formada çıxarıla bilə r.
Alignment Report) formatında və ya "vulqar" (Verbose Faydalı Etiketli Gapped Alignment Report) formatında.
İstifadə çi bu formatlardan birini və ya bir neçə sini çıxışa daxil etmə yi tə lə b edə bilə r. Bio.Align-də ki tə hlilçi edə bilə r
yalnız siqar və ya vulqar formatlarda düzülmə lə ri tə hlil edin və uyğunlaşdırmaları ehtiva edə n çıxışı tə hlil etmə yə cə k
insan tə rə fində n oxuna bilə n format.

Biopython test paketində ki exn_22_m_cdna2genome_vulgar.exn faylı Exonerate nümunə sidir
vulqar formatda düzülmə lə ri göstə rə n çıxış faylı:

Komanda xə tti: [exonerate -m cdna2genome ../scer_cad1.fa /media/Waterloo/Downloads/genomes/scer_s288c/sc Host adı: [blackbriar]

vulqar: gi|296143771|ref|NM_001180731.1| 0 1230 + gi|330443520|ref|NC_001136.10| 1319275 1318045 - 6146 vulqar: gi|296143771|ref|
NM_001180731.1| 1230 0 - gi|330443520|ref|NC_001136.10| 1318045 1319275 + 6146 vulqar: gi|296143771|ref|NM_001180731.1| 0 516 + gi|33044368
ref|NC_001145.3| 85010 667216 + 518 M 11 -- tamamlanmış bə raə t analizi

Machine Translated by Google

Bu fayla üç hizalanma daxildir. Bu faylı tə hlil etmə k üçün istifadə edin

```
>>> Bio import Align- də n
>>> alignments = Align.parse("exn_22_m_cdna2genome_vulgar.exn", "exonerate")
```

Lüğə t alignments.metadata yuxarıda göstə rilə n bu düzülmə lə r haqqında ümumi mə lumatları saxlayır çıxış faylı:

```
>>> alignments.metadata # doctest: +NORMALIZE_WHITESPACE
{'Proqram': 'bağımsızlıq',
  'Ə mr xə tti': 'exonerate -m cdna2genome ../scer_cad1.fa /media/Waterloo/Downloads/genomes/scer_s288c 'Hostname': 'blackbriar'}
```

İndi biz hizalamalar üzə rində tə krarlaya bilə rik. Düzə liş ə msalı 6146.0 olan birinci düzülmə də boşluq yoxdur:

```
>>> hizalama = sonrakı (düzlə şdirmə )
>>> alignment.score
6146.0
>>> hizalama.koordinatlar
massiv([[1319275, 1319274, 1319271, 1318045],
        [       0,       1,       4, 1230]])
>>> çap(alignment) # doctest: +ELLIPSIS
gi |330443 1319275 ??????????????
                   0 |||||||||||||||||||||||||||||||||||||||||||||||||||||
gi |296143          0 ??????????????
...
gi |330443 1318075 ??????????????? 1318045
                1200 |||||||||||||||||||||||||||||||||||||||
```

Machine Translated by Google

```
>>> hizalama.ə mə liyyatlar
bytearray(b'MCM')
```

Digə r ə mə liyyat kodlarının müə yyə n edilmə si üçün Exonerate sə nə dlə rinə baxın.

Eynilə , fayl yazmaq üçün Bio.Align.write-ə zə ng edə rkə n "vulqar" və ya "siqar" arqumentində n istifadə etmə k olar. vulqar və ya siqar hizalama xə tlə ri ilə .

Düzə ltmə ni BED və PSL formatında çap edə bilə rik:

```
>>> çap (format(düzlə şdirmə , "yataq")) # doctest: +NORMALIZE_WHITESPACE
gi |330443520 |ref |NC_001136.10 | gi |296143771 |ref |NM_001180731.1 | 61 1318045 1319275 <BLANKLINE>

>>> çap (format(alignment, "psl")) # doctest: +NORMALIZE_WHITESPACE
1230 0 0 0 0 0 0 0 - gi |296143771 |ref |NM
<BLANKLINE>
```

SAM format analizatoru özünün (isteğe bağlı) ə mə liyyatlar atributunu müə yyə n edir (bölmə 6.7.13), bu, tam deyil. Exonerate format analizatorunda müə yyə n edilmiş ə mə liyyatlar atributuna uyğundur. Ə mə liyyatlar kimi atribut isteğe bağlıdır, hizalanmanı SAM formatında çap etmə zdə n ə vvə l onu silirik:

```
>>> del alignment.operations
>>> çap (format(alignment, "sam")) # doctest: +NORMALIZE_WHITESPACE
gi |296143771 |ref |NM_001180731.1 | 16 gi |330443520 |ref |NC_001136.10 | 1318046 <BLANKLINE>
```

Üçüncü düzülmə dörd uzun boşluqdan ibarə tdir:

```
>>> alignment = sonrakı(alignments) # ikinci düzülmə
>>> alignment = sonrakı(alignments) # üçüncü düzülmə
>>> çap(alignment) # doctest: +ELLIPSIS
gi |330443 85010 ???????????-????????????????--????-?-????????----???????????
0 |||||||||||-||||||||||||||||||--||||-|-||||||||||||||||||||
gi |296143 0 ???????????????
<BLANKLINE>
gi |330443 85061 ???????????????
60 |||||----------
gi |296143 60 ??????----------
...
gi |330443 666990 ????????????
582000 ----------||||||||||
gi |296143 346 ----------??????????
<BLANKLINE>
gi |330443 667050 ?????????-?????????
582060 ||--|||||-|||||||--|-||||||||||||||||||||||||||||
gi |296143 356 ???--?????????????--?-?????????????
<BLANKLINE>
gi |330443 667109 ????????????
582120 |||||||||||||||||||||||||||||||||||||||||||||||||||||||-|
gi |296143 411 ???????????????
<BLANKLINE>
gi |330443 667168 ????????? 667216
582180 ||-|||-||||||||||||||||||||||||||||||||||||||| 582228
gi |296143 470 ???-???-????????????? 516
<BLANKLINE>
>>> çap (format(düzlə şdirmə , "tə mizlə mə ")) # doctest: +NORMALIZE_WHITESPACE
```

Machine Translated by Google

vulqar: gi|296143771|ref|NM_001180731.1| 0 516 + gi|330443688|ref|NC_001145.3| 85010 667216 + 518 M 11 11 G 1 0 M 15 15 G 2 0 M 4 4 G 1 0 M 1 1 G 1 0 M 8 8
G 4 0 M 17 17 5 0 2 I 0 168904 3 0 2 M 4 4 G 0 1 M 8 8 G 2 0 M 3 3 G 1 0
M 33 33 G 0 2 M 7 7 G 0 1 M 102 102 5 0 2 I 0 96820 3 0 2 M 14 14 G 0 2 M 10 10
G 2 0 M 5 5 G 0 2 M 10 10 G 2 0 M 4 4 G 0 1 M 20 20 G 1 0 M 15 15 G 0 1 M 5 5
G 3 0 M 4 4 5 0 2 I 0 122114 3 0 2 M 20 20 G 0 5 M 6 6 5 0 2 I 0 193835 3 0 2
M 12 12 G 0 2 M 5 5 G 1 0 M 7 7 G 0 2 M 1 1 G 0 1 M 12 12 C 75 75 M 6 6 G 1 0
M 4 4 G 0 1 M 2 2 G 0 1 M 3 3 G 0 1 M 41 41
<BLANKLINE>

6.7.8 NEXUS

NEXUS fayl formatı [30] filogenetik mə lumatları saxlamaq üçün bir neçə proqram tə rə fində n istifadə olunur. Bu, NEXUS formatında fayl nümunə sidir (Biopython paylanmasında Tests/Nexus alt kataloqunda codonposset.nex kimi mövcuddur):

```
#NEXUS
 [MacClade 4.05 Hesablama Bioloqu, Universitetdə qeydiyyatdan keçdi]


MƏ LUMATLARA BAŞLAYIN;
        ÖLÇÜLƏ R NTAX=2 NCHAR=22; MƏ LUMAT
        TİPİNİ FORMAT EDİN=DNT YOXDUR=? GAP=- ;
MATRİS
                                10          20 ]]
 [ [                            .           .

Aegotheles      AAAAAGGCATTGTGGTGGGAAT [22]
Aerodramus      ?????????TTGTGGTGGGAAT [13]
 ;
END;


CODONS BAŞLAYIN;
        CODONPOSSET * KodonMövqelə ri =
                N: 1-10, 1:
                11-22\3, 2:
                12-22\3, 3: 13-22\3;

        CODESET * UNTITLED = Universal: hamısı ;
END;
```

Ümumiyyə tlə , NEXUS formatında olan fayllar daha mürə kkə b ola bilə r. Bio.Align , NEXUS fayllarından Alignment obyektlə rini çıxarmaq üçün Bio.Nexus-da NEXUS tə hlilçisinə (bax. Fə sil 16) çox etibar edir.

Bu NEXUS faylında düzülüşü oxumaq üçün istifadə edin

```
>>> Bio idxaldan Align >>> alignment =
Align.read ("codonposset.nex", "nexus") >>> print(alignment)

Agothele        0 AAAAAGGCATTGTGGTGGGAAT 22
                0 .........|||||||||||||| 22 0 ?????????
Aerodram        TTGTGGTGGGAAT 22
<BLANKLINE>
```

Machine Translated by Google

```
>>> hizalama.forma
(2, 22)
```

Ardıcıllıqlar ardıcıllıq atributunun altında saxlanılır:

```
>>> alignment.sequences[0].id
'Aegotheles'
>>> alignment.sequences[0].seq
Seq('AAAAAGGCATTGTGGTGGGAAT')
>>> alignment.sequences[0].annotations
{'molekul_növü': 'DNT'}
>>> alignment.sequences[1].id
"Aerodramus"
>>> alignment.sequences[1].seq
Seq('?????????TTGTGGTGGGAAT')
>>> alignment.sequences[1].annotations
{'molekul_növü': 'DNT'}
```

Bu düzülüşü NEXUS formatında çap etmə k üçün istifadə edin

```
>>> çap (format (düzlə şdirmə , "nexus"))
#NEXUS

mə lumatlara başlamaq;
ölçülə r ntax=2 nchar=22;
format datatype=dna itkin=? boşluq=-;
matris
Aegotheles AAAAAGGCATTGTGGTGGGAAT
Aerodramus ?????????TTGTGGTGGGAAT
;
son;
<BLANKLINE>
```

Eynilə , siz align.write(alignment, "myfilename.nex", "nexus") ilə düzülüşü yazmaq üçün istifadə edə bilə rsiniz. NEXUS formatını myfilename.nex faylına köçürün.

6.7.9 BLAST və ya FASTA-dan cə dvə l çıxışı

Cə dvə l çıxışında hizalama çıxışı -m 8CB və ya -m 8CC ilə işlə yə n FASTA hizalayıcı [34] tə rə fində n yaradılır. arqument və ya BLAST [1] ilə -outfmt 7 arqumenti ilə işlə yin.

Biopython mə nbə paylanmasının Tests/Fasta alt kataloqunda nucleotide_m8CC.txt faylı -m 8CC arqumenti ilə FASTA tə rə fində n yaradılan çıxış faylının nümunə si:

```
# fasta36 -m 8CC seq/mgstm1.nt seq/gst.nlib
# FASTA 36.3.8h May, 2020
# Sorğu: pGT875 - 657 nt
# Verilə nlə r bazası: seq/gst.nlib
# Sahə lə r: sorğu id, mövzu identifikatoru, % identifikasiya, düzülmə uzunluğu, uyğunsuzluqlar, boşluq açılır, q. başlamaq, q. son, # 12 xit tapılıb

pGT875    pGT875    100.00    657    0    0    1    657    38    69
pGT875    RABGLTR    79.10    646    135    0    1    646    34
pGT875    BTGST    59.56    413    167    21    176    594    228
pGT875    RABGSTB    66.93    127    42    8    159    289    157
pGT875    OCDHPR    91.30    23    2    1    266    289    2303
...
# FASTA 1 sorğunu emal etdi
```

Machine Translated by Google


```
>>> Bio idxaldan Align >>>
alignments = Align.parse ("nucleotide_m8CC.txt", "cə dvə l")
```

Fayl başlığında göstə rilə n mə lumat düzülmə lə rin metadata atributunda saxlanılır:

```
>>> alignments.metadata # doctest: +NORMALIZE_WHITESPACE {'Command line':
'fasta36 -m 8CC seq/mgstm1.nt seq/gst.nlib', 'Proqram': 'FASTA', 'Versiya': '36.3.8h
  May':'tabase', 2 'seq/
  gst.nlib'}
```

Düzə lişlə r üzə rində tə krarlamaqla müə yyə n bir düzülmə çıxarın. Nümunə olaraq dördüncü sıraya keçə k:

```
>>> hizalama = sonrakı (düzlə şdirmə )
>>> düzlə şdirmə = sonrakı (düzlə şdirmə )
>>> hizalama = sonrakı (düzlə şdirmə )
>>> düzlə şdirmə = sonrakı (düzlə şdirmə )
>>> çap (düzlə şdirmə )
RABGSTB          156 ?????????--????????????
                   0 |||||||||||||||--||||||||||||||||||--||||||||||||||||||||||||
pGT875           158 ???????????????--??????????????????-????????????
<BLANKLINE>
RABGSTB          214 ??????????
                  60 ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||-|
pGT875           215 ?????????????????????-?????????????
<BLANKLINE>
RABGSTB          273 ??????-????????? 287
                 120 ||||||-||||||||| 135
pGT875           274 ??????????????? 289
<BLANKLINE>
>>> alignment.coordinates # doktest: +NORMALIZE_WHITESPACE massivi([[156, 171,
173, 190, 190, 201, 202, 260, 261, 272, 272, 279, 8, 271, [[[156, 171, 173, 190, 190, 201] 173, 190, 192, 203, 203,
          261, 261, 272, 273, 280, 281, 289]])
>>> alignment.aligned massiv([[[156,
171], [173, 190], [190, 201], [202,
            260], [261, 272],
            [272, 279], [279,
            287]], <BLANK>
```

```
          [[158, 173], [173,
            190], [192,
            203], [203,
            261], [261,
            272], [273,
            280], [281,
            289]]])
```

Hə də f və sorğu ardıcıllığının ardıcıllıq mə lumatları hə də f və sorğu atributlarında (hə mçinin alignment.sequences altında) saxlanılır:

Machine Translated by Google

```
>>> hizalama.hə də f
SeqRecord(seq=Seq(Yox, uzunluq=287), id='RABGSTB', ad='<namə lum ad>', description='<namə lum tə svir >>> alignment.query

SeqRecord(seq=Seq(Heç biri, uzunluq=657), id='pGT875', ad='<namə lum ad>', description='<namə lum tə svir
```

Düzə liş haqqında mə lumat hizalanmanın annotasiya atributunda saxlanılır:

```
>>> alignment.annotations # doctest: +NORMALIZE_WHITESPACE
{'% eynilik': 66.93,
 'uyğunsuzluqlar': 42,
 'boşluq açılır': 8,
 'qiymə tlə ndirmə ': 3.2e-07,
 'bit hesabı': 45.0}
```

Xüsusilə BLAST xüsusi sütunları daxil etmə klə və ya istisna etmə klə cə dvə lli çıxışı fə rdilə şdirmə k üçün bir çox variant tə klif edir; tə fə rrüatlar üçün BLAST sə nə dlə rinə baxın. Bu mə lumatlar lüğə tlə rdə saxlanılır uyğun olaraq alignment.annotations, alignment. target.annotations, və ya alignment.query.annotations.

6.7.10 HH-suite çıxış faylları

Hhr formatında hizalanma faylları HH-suite-də hhsearch və ya hhblits tə rə fində n yaradılır [44]. Mə sə lə n, Biopython test paketində 2uvo_hhblits.hhr faylına baxın:

```
Sorğu                    2UVO:A |PDBID |ZƏ NCİR |Ardıcıllıq
Sütunları uyğunlaşdırın 171
Seqslə rin_sayı          4005-də n 1560
Nef                      8.3
Axtarılan_HMM-lə r 34
Tarix                    Cümə 15 Fevral 16:34:13 2019
Ə mr                     hhblits -i 2uvoAh.fasta -d /pdb70

 No Hit Prob E-də yə r P-də yə ri Hesab SS Cols Sorğu HMM Şablonu HMM
   1 2uvo_A Aqqlütinin izolektin 1; 100,0 3,7E-34 4,8E-38 210,3 0,0 171 1-171 1-171 (171)
   2 2wga; lektin (aqqlutinin); 99,9 1,1E-30 1,4E-34 190,4 0,0 162 2-169 2-163 (164)
   3 1ulk_A Lektin-C; xitin-bindin 99,8 5,2E-24 6,6E-28 148,2 0,0 120                    1-124      2-121 (126)
...
  31 4z8i_A BBTPGRP3, peptidoqlikan 79.6 32 1wga ; lektin  0,12 1,5E-05 36,1 0,0 37      1-37       9-54 (236)
  (aqqlutinin); 40.4                                        2,6 0,00029 25,9 0,0 106 54-163          11-116 (164)

№1
>2uvo_A Aqqlütinin izolektin 1; karbohidrat bağlayan zülal, hevein domeni, xitin bağlayan, GERM agglu Ehtimal=99.95 E-də yə r=3.7e-34 Qiymə t=210.31
Aligned_cols=171 Eyniliklə r=100% Oxşarlıq=2.050 Sum_p

Q 2UVO:A |PDBID |C      1 ERCGEQGSNMECPNNLCCSQYGYCGMGGDYCGKGCQNGACWTSKRCGSQAGGATCTNNQCCSQYGYCGFGAEYCGAGCQG 1
Q Konsensus             ~~cg~~~~~~~c~~~~CCs~~g~CG~~~~c~~~c~~~~c~~~~~Cg~~~~~~~c~~~~CCs~~g~CG~~~~~c~~~c |||||++.++..||++.|||+|+|||.+.+||++
                          +||.+.|++..+|+++++.++....
T Konsensus             1 ~~cg~~~~~~c~~~~CCS~~g~Cg~~~~~Cg~gC~~~~c~~~~~cg~~~~~~c~~~~CCs~~g~Cg~~~~~c~~~c~~ 1
T 2uvo_A                ERCGEQGSNMECPNNLCCSQYGYCGMGGDYCGKGCQNGACWTSKRCGSQAGGATCTNNQCCSQYGYCGFGAEYCGAGCQG
T ss_dssp                 CBCBGGGTTBBCGGGCEECTTSBEEBSHHHHSTTCCBSSCSSCCBCBGGGTTBCCSTTCCECTTSBEEBSHHHHSTTCCB
T ss_pred                 CCCCCCCCCCCCCCCeeCCCCeECCCcccccccccccccccccccccccccccccCCCCceeCCCCccCCCCcc
Güvə n                    799999999999999999999999999999999999999999999999999999999999999999999999999999999999999
```

Machine Translated by Google

```
Q 2UVO:A|PDBID|C 81 GPCRADIKCGSQAGGKLCPNLCCSQWGFCGLGSEFCGGGCQSGACSTDKPCGKDAGGRVCTNNYCCSKWGSCGIGPGYC Q Konsensus 81
~~~~~~~~Cg~~~~~~~c~~~~CCS~~G~CG~~~~~C~~~Cq~~~c~~~~~Cg~~~~~~~c~~~~CCS~~G~CG~~~~~C +++++|+.|+...+++.||++.||||.||||...+||+.+||+++|++|.+|++.++++|..
                                        +.||+++-||+...||
T Konsensus                    81 ~~~~~~~~cg~~~~~~~c~~~~CCs~~g~CG~~~~~C~~gCq~~~c~~~~~cg~~~~~~~c~~~~ccs~~g~Cg~~~~~C 81
T 2uvo_A                       GPCRADIKCGSQAGGKLCPNLCCSQWGFCGLGSEFCGGGCQSGACSTDKPCGKDAGGRVCTNNYCCSKWGSCGIGPGYC
T ss_dssp                          SSCSSCCBCBGGGTTBCCGGGCEECTTSBEEBSHHHHSTTCCBSSCSSCCCCBTTTTBCSSTTCEECTT
T ss_pred                          ccccccccccccCCCCCCccccCCCCCCCCccccCCCCccccccccccccccccccCCCCccCCCCCEecCchhhc
Güvə n                             9999999999998899999999999
```

Machine Translated by Google

- Ə ldə edilmiş düzülmə lə rin hə r biri üçün bir sə tirdə n ibarə t xülasə ;
- Ardıcıl olaraq ə traflı şə kildə göstə rilə n hizalamalar.

Bu faylı tə hlil etmə k üçün istifadə edin

```
>>> Bio idxaldan Align >>> alignments
= Align.parse ("2uvo_hhblits.hhr", "hhr")
```

Başlıq mə lumatlarının ə ksə riyyə ti düzlə şdirmə lə rin metadata atributunda saxlanılır:

```
>>> alignments.metadata # doctest: +NORMALIZE_WHITESPACE {'Uyğun_sütunlar': 171,
'Ardıcılların_sayı': (1560, 4005),
 'Neff': 8.3, 'Axtarılan_HMM'lə r': 34,
 'Fe:363' 2019',
 'Ə mr xə tti': 'hhblits -i
 2uvoAh.fasta -d /pdb70'}
```

atribut kimi saxlanılan sorğu adı istisna olmaqla:

```
>>> alignments.query_name '2UVO:A|
 PDBID|CHAIN|SEQUENCE'
```

çünki düzülmə lə rin hə r birində yenidə n görünə cə k.
Düzə lişlə r üzə rində tə krarlayın:

```
>>> düzülmə lə rdə düzülmə üçün :
...          print(alignment.target.id) # doctest: +ELLIPSIS
...
2uvo_A
2wga
1ulk_A
...
4z8i_A
1wga
```

Birinci hizaya daha ə traflı baxaq:

```
>>> alignments.rewind() >>>
alignment = next(alignments) >>> alignment #
doctest: +ELLIPSIS <Alignment obyekti (2 sə tir x 171
sütun) ...> >>> çap (alignment) 2uvo_A

                    0 ERCGEQGSNMECPNNLCCSQYGYCGMGGDYCGKGCQNGACWTSKRCGSQAGGATCTNNQC 0
                    ||||||||||||||||||||||||||||||||||||||||||||||||||||
2UVO:A|PD           0 ERCGEQGSNMECPNNLCCSQYGYCGMGGDYCGKGCQNGACWTSKRCGSQAGGATCTNNQC
<BLANKLINE>
2uvo_A             60 CSQYGYCGFGAEYCGAGCQGGPCRADIKCGSQAGGKLCPNLCCSQWGFCGLGSEFCGGG 60
                   ||||||||||||||||||||||||||||||||||||||||||||||||||||
2UVO:A|PD          60 CSQYGYCGFGAEYCGAGCQGGPCRADIKCGSQAGGKLCPNLCCSQWGFCGLGSEFCGGG
<BLANKLINE>
2uvo_A            120 CQSGACSTDKPCGKDAGGRVCTNNYCCSKWGSCGIGPGYCGAGCQSGGCDG 171 120
                  ||||||||||||||||||||||||||||||||||||||||||||||||||||| 171
2UVO:A|PD         120 CQSGACSTDKPCGKDAGGRVCTNNYCCSKWGSCGIGPGYCGAGCQSGGCDG 171
<BLANKLINE>
```

Machine Translated by Google

Hə də f və sorğu ardıcıllığı alignment.sequences-də saxlanılır. Bunlar ikili düzülmə lə r olduğundan, biz onlara alignment.target və alignment.query vasitə silə də daxil ola bilə rik:

```
>>> alignment.hə də f alignment.sequencesdir [0]
Doğrudur
>>> alignment.query alignment.sequencesdir [1]
Doğrudur
```

Sorğunun identifikatoru alignments.query_name-də n tə yin edilir (qeyd edə k ki, hhr faylında düzülmə də çap edilmiş sorğu ID-si qısaldılmışdır):

```
>>> alignment.query.id '2UVO:A|
 PDBID|CHAIN|SEQUENCE'
```

Hə də fin identifikatoru ardıcıllıqla düzülmə blokundan götürülür (T 2uvo_A ilə başlayan xə tt):

```
>>> alignment.target.id '2uvo_A'
```

Hə də fin və sorğunun ardıcıl mə zmunu bu uyğunlaşdırmada mövcud olan mə lumatlardan doldurulur:

```
>>> alignment.target.seq
Seq('ERCGEQGSNMECPNNLCCSQYGYCGMGGDYCGKGCQNGACWTSKRCGSQAGGAT...CDG') >>>
alignment.query.seq
Seq('ERCGEQGSNMECPNNLCCSQYGYCGMGGDYCGKGCQNGACWTSKRCGSQAGGAT...CDG')
```

Ə gə r düzülmə tam ardıcıllıqla uzanmazsa, ardıcıllığın mə zmunu natamam olacaq (qismə n müə yyə n edilmiş ardıcıllıq; Bölmə 3.12-yə baxın) .

Bu hizalama blokunun ">" ilə başlayan ikinci sə tirində hə də f ardıcıllığının götürüldüyü Gizli Markov Modelinin adı və tə sviri göstə rilir. Bunlar alignment.target.annotations lüğə tində "hmm_name" və "hmm_description" düymə lə ri altında saxlanılır:

```
>>> alignment.target.annotations # doctest: +NORMALIZE_WHITESPACE {'hmm_name': '2uvo_A',

   'hmm_description': 'Aqqlütinin izolektin 1; karbohidrat bağlayan zülal, hevein domeni, xitin-bindin
```

Lüğə t alignment.target.letter_annotations hə də f alignment konsensus ardıcıllığını, PSIPRED tə rə fində n proqnozlaşdırıldığı kimi ikinci də rə cə li strukturu və DSSP tə rə fində n müə yyə n edilə n hə də f ikinci də rə cə li strukturu saxlayır:

```
>>> alignment.target.letter_annotations # doctest: +NORMALIZE_WHITESPACE {'Konsensus':
'~~cg~~~~~~c~~~~CCS~~g~Cg~~~~~Cg~gC~~~~c~~~~~cg~~~~~~~c~~~~CCs~~g~Cg~~~~~c~~~c~~~~~~~~~~ 'ss_pred':
   'CCCCCCCCCCCCCCCCeeCCCCeECCCcccccCCcccccccccccccccccccccccCCCCceeCCCCccCCCCcccccccccccccc 'ss_dssp':
   'CBCBGGGTTBBCGGGCEECTTSBEEBSHHHHSTTCCBSSCSSCCCCBBTCGGTCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCBBCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

Bu nümunə də sorğu ardıcıllığı üçün yalnız konsensus ardıcıllığı mövcuddur:

```
>>> alignment.query.letter_annotations {'Konsensus':
'~~cg~~~~~~~c~~~~CCs~~g~CG~~~~~c~~~c~~~~c~~~~~Cg~~~~~~~c~~~~CCs~~g~CG~~~~~~~
```

alignment.annotations lüğə ti hizalama blokunun üçüncü sə tirində göstə rilə n düzülmə haqqında mə lumatı saxlayır:

```
>>> alignment.annotations # doctest: +NORMALIZE_WHITESPACE {'Ehtimal': 99.95,

   'E-də yə r': 3.7e-34, 'Xal':
   210.31, 'Şə xsiyyə tlə r':
   100.0, 'Oxşarlıq': 2.05,
   'Cə mi_probs': 166.9}
```

Machine Translated by Google

Cütlü düzülmə üçün etibarlılıq də yə rlə ri alignment.column_annotations lüğə tində "Güvə n" düymə si altında saxlanılır. Bu lüğə t hə mçinin sorğu və hə r bir uyğunlaşdırma blokunun hə də f bölmə si arasında göstə rilə n hə r sütun üçün hesabı saxlayır:

```
>>> alignment.column_annotations # doctest: +NORMALIZE_WHITESPACE {'sütun hesabı': '|||++.
++..||++.|||+|+|||.+.+||+++||.+.|++..+|+++++.++.++..+|+++++.++....
 'Güvə n': '79999999999
```

Machine Translated by Google

```
plas_proh          56 KLRIAPGSFYSVTLGT---PGTYSFYCTPHRGAGMVGTITV 94
azup_achc          51 -FKSKINENYKVTFTA---PGVYGVKCTPHYGMGMVGVVEV 88
<BLANKLINE>
```

Tə hlilçi sütunun uyğunluq/uyğunsuzluq/silinmə ("D") və ya ə lavə ("I") olduğunu müə yyə n etmə k üçün A2M faylında tire, nöqtə və kiçik və böyük hə rflə rin nümunə sini tə hlil edir. Bu mə lumat alignment.column_annotations lüğə tinin uyğun düymə si altında saxlanılır:

```
>>> alignment.column_annotations {'state':
'DIDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD
```

Və ziyyə t mə lumatı düzülmə də saxlandığından, hizalanmanı A2M formatında çap edə bilə rik:

```
>>> çap (format(alignment, "a2m")) >plas_horvu

D.VLLGANGGVLVFEPNDFSVKAGETITFKNNAGYPHNVVFDEDAVPSG.VD.VSKISQEEYLTAPGETFSVTLTV...PGTYGFYCEPHAGAGMVGKVTV >plas_chlre

-.VKLGADSGALEFVPKTLTIKSGETVNFVNNAGFPHNIVFDEDAIPSG.VN.ADAISRDDYLNAPGETYSVKLTA...AGEYGYYCEPHQGAGMVGKIIV >plas_anava

-.VKLGSDKGLLVFEPAKLTIKPGDTVEFLNNKVPPHNVVFDAALNPAKsADlAKSLSHKQLLMSPGQSTSTTFPAdapAGEYTFYCEPHRGAGMVGKITV >plas_proho

VqIKMGTDKYAPLYEPKALSISAGDTVEFVMNKVGPHNVIFDK--VPAG.ES.APALSNTKLRIAPGSFYSVTLGT...PGTYSFYCTPHRGAGMVGTITV >azup_achcy VhMLNKGKDGAMVFEPASLKVAPGDTVTFIPTDK

GHNVETIKGMIPDG.AE.A-------FKSKINENYKVTFTA...PGVYGVKCTPHYGMGMVGVVEV
<BLANKLINE>
```

Eynilə , hizalanma Align.write istifadə edə rə k çıxış faylına A2M formatında yazıla bilə r (bax bölmə 6.6.2).

6.7.12 Mauve eXtended Multi-FastA (xmfa) formatı

Mauve [8] çoxlu genom düzüşlə rinin qurulması üçün proqram paketidir. Bu hizalamalar genişlə nmiş Multi-FastA (xmfa) formatında saxlanılır. ProqressivMauve (Mauve-də aligner proqramı) tam olaraq necə adlandırıldığından asılı olaraq, xmfa formatı bir qə də r fə rqlidir.

Ə gə r progressiveMauve-də olduğu kimi tə k ardıcıl giriş faylı ilə çağırılırsa

```
progressiveMauve combined.fasta --output=combined.xmfa ...
```

burada birlə şdirilmiş.fasta genom ardıcıllığını ehtiva edir:

```
>equCab1
GAAAAGGAAAGTACGGCCCGGCCACTCCGGGTGTGTGCTAGGAGGGCTTA
>mm9
GAAGAGGAAAAGTAGATCCCTGGCGTCCGGAGCTGGGACGT
>canFam2
CAAGCCCTGCGCGCTCAGCCGGAGTGTCCCGGGCCCTGCTTTCCTTTTC
```

onda combined.xmfa çıxış faylı aşağıdakı kimidir:

```
#FormatVersiya Mauve1
#Sequence1Fayl          birlə şdirilmiş.fa
#Ardıcıllıq1Giriş         1
#Sequence1Format           FastA
#Sequence2Fayl          birlə şdirilmiş.fa
#Ardıcıllıq2Giriş         2
```

Machine Translated by Google

```
#Sequence2Format        FastA
#Sequence3Fayl          birlə şdirilmiş.fa
#Ardıcıllıq3Giriş       3
#Sequence3Format        FastA
#BackboneFile           birlə şdirilmiş.xmfa.bbcols
> 1:2-49 - birlə şdirilmiş.fa
AAGCCCTCCTAGCACACACCCGGAGTGG-CCGGGCCGTACTTTCCTTTT
> 2:0-0 + birlə şdirilmiş.fa
-------------------------------------------------
> 3:2-48 + birlə şdirilmiş.fa
AAGCCCTGC--GCGCTCAGCCGGAGTGTCCCGGGCCCTGCTTTCCTTTT
=
> 1:1-1 + birlə şdirilmiş.fa
G
=
> 1:50-50 + birlə şdirilmiş.fa
A
=
> 2:1-41 + birlə şdirilmiş.fa
GAAGAGGAAAAGTAGATCCCTGGCGTCCGGAGCTGGGACGT
=
> 3:1-1 + birlə şdirilmiş.fa C

=
> 3:49-49 + birlə şdirilmiş.fa
C
=
```

müvafiq olaraq at (equCab1), siçan (mm9) və it (canFam2) üçün giriş genom ardıcıllığına istinad edə n rə qə mlə rlə (1, 2, 3). Bu xmfa faylı = simvolu ilə ayrılmış altı düzülmə blokundan ibarə tdir. Bu düzülmə lə ri çıxarmaq üçün Align.parse istifadə edin:

```
>>> Bio idxaldan Align >>> alignments
= Align.parse ("combined.xmfa", "mauve")
```

Fayl başlığı mə lumatları metadata atributunda saxlanılır:

```
>>> alignments.metadata # doctest: +NORMALIZE_WHITESPACE {'FormatVersion': 'Mauve1',

   'BackboneFile': 'combined.xmfa.bbcols', 'Fayl': 'combined.fa'}
```

İdentifikatorlar atributu üç ardıcıllıq üçün ardıcıllıq identifikatorlarını saxlayır, bu halda bu üç rə qə mdir:

```
>>> alignments.identifiers ['0', '1', '2']
```

Bu identifikatorlar fə rdi düzülmə lə rdə istifadə olunur:

```
>>> düzülmə lə rdə düzülmə üçün : çap
...             ([ alignment.sequences -də qeyd üçün qeyd.id ]) çap(alignment) çap("********")
...
...
...
```

Machine Translated by Google

```
['0', '1', '2']
0                    49 AAGCCCTCCTAGCACACACCCGGAGTGG-CCGGGCCGTACTTTCCTTTT 1
1                     0 ------------------------------------------------- 0
2                     1 AAGCCCTGC--GCGCTCAGCCGGAGTGTCCCGGGCCCTGCTTTCCTTTT 48
<BLANKLINE>
********
['0']
0                     0 G 1
<BLANKLINE>
********
['0']
0                    49 A 50
<BLANKLINE>
********
['1']
1                     0 GAAGAGGAAAAGTAGATCCCTGGCGTCCGGAGCTGGGACGT 41
<BLANKLINE>
********
['2']
2                     0 C 1
<BLANKLINE>
********
['2']
2                    48 C 49
<BLANKLINE>
********
```

Qeyd edə k ki, yalnız birinci blok real uyğunlaşmadır; digə r bloklar yalnız bir ardıcıllığı ehtiva edir. By bu bloklar daxil olmaqla, xmfa faylı combined.fa girişində tə min edilmiş tam ardıcıllığı ehtiva edir. fayl.

Ə gə r progressiveMauve hə r bir genom üçün ayrıca giriş faylı ilə çağırılırsa, mə sə lə n

progressiveMauve equCab1.fa canFam2.fa mm9.fa --output=separate.xmfa ...

burada hə r bir Fasta faylı yalnız bir növ üçün genom ardıcıllığını ehtiva edir, sonra çıxış faylı separat.xmfa aşağıdakı kimidir:

```
#FormatVersiya Mauve1
#Sequence1Fayl          equCab1.fa
#Sequence1Format          FastA
#Sequence2Fayl          canFam2.fa
#Sequence2Format          FastA
#Sequence3Fayl          mm9.fa
#Sequence3Format          FastA
#BackboneFile separat.xmfa.bbcols
> 1:1-50 - equCab1.fa
TAAGCCCTCCTAGCACACACCCGGAGTGGCC-GGGCCGTAC-TTTCCTTTTC
> 2:1-49 + canFam2.fa
CAAGCCCTGC--GCGCTCAGCCGGAGTGTCCCGGGCCCTGC-TTTCCTTTTC
> 3:1-19 - mm9.fa
--------------------------------GGATCTACTTTTCCTCTTC
=
> 3:20-41 + mm9.fa
```

Machine Translated by Google

```
CTGGCGTCCGGAGCTGGGACGT
=
```

At üçün equCab1, siçan üçün mm9 və it üçün canFam2 identifikatorları indi çıxışda açıq şə kildə göstə rilir. fayl. Bu xmfa faylı = simvolu ilə ayrılmış iki hizalama blokundan ibarə tdir. Çıxarmaq üçün Align.parse istifadə edin bu uyğunlaşmalar:

```
>>> Bio import Align- də n
>>> alignments = Align.parse("separate.xmfa", "mauve")
```

Fayl başlığı datasına indi daxiletmə faylının adı daxil deyil:

```
>>> alignments.metadata # doctest: +NORMALIZE_WHITESPACE
{'FormatVersion': 'Mauve1',
  'BackboneFile': 'separate.xmfa.bbcols'}
```

İdentifikatorlar atributu üç ardıcıllıq üçün ardıcıllıq identifikatorlarını saxlayır:

```
>>> alignments.identifiers
 ['equCab1.fa', 'canFam2.fa', 'mm9.fa']
```

Bu identifikatorlar fə rdi düzülmə lə rdə istifadə olunur:

```
>>> düzülmə lə rdə düzülmə üçün :
 ...          çap ([ alignment.sequences -də qeyd üçün qeyd.id ])
 ...          çap (düzlə şdirmə )
 ...          çap ("******")
 ...
 ['equCab1.fa', 'canFam2.fa', 'mm9.fa']
equCab1.f          50 TAAGCCCTCCTAGCACACACCCGGAGTGGCC-GGGCCGTAC-TTTCCTTTTC 0
canFam2.f           0 CAAGCCCTGC--GCGCTCAGCCGGAGTGTCCCGGGCCCTGC-TTTCCTTTTC 49
mm9.fa             19 --------------------------------GGATCTACTTTTCCTCTTC 0
<BLANKLINE>
********
 ['mm9.fa']
mm9.fa             19 CTGGCGTCCGGAGCTGGGACGT 41
<BLANKLINE>
********
```

Düzlə şdirmə lə ri bə növşə yi formatda çıxarmaq üçün Align.write istifadə edin:

```
>>> io import StringIO- dan
>>> axın = StringIO()
>>> alignments = Align.parse("separate.xmfa", "mauve")
>>> Align.write (düzlə şdirmə lə r, axın, "bə növşə yi")
2
>>> print(stream.getvalue()) # doctest: +NORMALIZE_WHITESPACE
#FormatVersiya Mauve1
#Sequence1Fayl          equCab1.fa
#Sequence1Format            FastA
#Sequence2Fayl          canFam2.fa
#Sequence2Format            FastA
#Sequence3Fayl          mm9.fa
#Sequence3Format            FastA
#BackboneFile           ayrı.xmfa.bbcols
```

Machine Translated by Google

```
> 1:1-50 - equCab1.fa
TAAGCCCTCCTAGCACACACCCGGAGTGGCC-GGGCCGTAC-TTTCCTTTTC
> 2:1-49 + canFam2.fa
CAAGCCCTGC--GCGCTCAGCCGGAGTGTCCCGGGCCCTGC-TTTCCTTTTC
> 3:1-19 - mm9.fa
--------------------------------GGATCTACTTTTCCTCTTC
=
> 3:20-41 + mm9.fa
CTGGCGTCCGGAGCTGGGACGT
=
<BLANKLINE>
```

Burada yazıçı bu formatı yaratmaq üçün alignments.metadata və alignments.identifiers-də saxlanan mə lumatlardan istifadə edir. Alignments obyektinizdə bu atributlar yoxdursa, onları Align.write-a açar söz arqumentlə ri kimi tə qdim edə bilə rsiniz:

```
>>> axın = StringIO() >>>
alignments = Align.parse("separate.xmfa", "mauve") >>> metadata =
alignments.metadata >>> identifiers =
alignments.identifiers >>> alignments = list(alignments) #
bu, atributları aşağı salır >>> alignments.metadata :+backtestELL son doc):


  ...
AttributeError: 'list' obyektinin 'metadata' atributu yoxdur >>> alignments.identifiers
# doctest: +ELLIPSIS Traceback (ə n son zə ng):

  ...
AttributeError: 'siyahı' obyektində 'identifikator' atributu yoxdur
>>> Align.write(düzlə şdirmə lə r, axın, "bə növşə yi", metadata=metadata, identifikatorlar=identifikatorlar) 2

>>> print(stream.getvalue()) # doctest: +NORMALIZE_WHITESPACE #FormatVersion Mauve1

#Sequence1Fayl          equCab1.fa
#Sequence1Format            FastA
#Sequence2Fayl          canFam2.fa
#Sequence2Format            FastA
#Sequence3Fayl          mm9.fa
#Sequence3Format            FastA
#BackboneFile           separat.xmfa.bbcols > 1:1-50 -
equCab1.fa
TAAGCCCTCCTAGCACACACCCGGAGTGGCC-GGGCCGTAC-TTTCCTTTTC > 2:1-49 + canFam2.fa
CAAGCCCTGC--
GCGCTCAGCCGGAGTGTCCCGTTTC1 --GCGCTCAGCCGGAGTGTCCCGTTTC11 mm9.fa
--------------------------------
GGATCTACTTTTCCTCTTC
=
> 3:20-41 + mm9.fa
CTGGCGTCCGGAGCTGGGACGT
=
<BLANKLINE>
```

Python, bu misalda olduğu kimi, bu atributları alignments obyektinə birbaşa ə lavə etmə yə imkan vermir

Machine Translated by Google

sadə siyahıya çevrildi. Bununla belə , siz Alignments obyekti yarada və ona atributlar ə lavə edə bilə rsiniz (Bölmə 6.5-ə baxın):

```
>>> alignments = Align.Alignments(alignments)
>>> alignments.metadata = metadata
>>> alignments.identifiers = identifikatorlar
>>> axın = StringIO()
>>> Align.write(düzlə şdirmə lə r, axın, "bə növşə yi", metadata=metadata, identifikatorlar=identifikatorlar)
2
>>> print(stream.getvalue()) # doctest: +NORMALIZE_WHITESPACE
#FormatVersiya Mauve1
#Sequence1Fayl equCab1.fa
#Sequence1Format FastA
#Sequence2Fayl canFam2.fa
#Sequence2Format FastA
#Sequence3Fayl mm9.fa
#Sequence3Format FastA
#BackboneFile ayrı.xmfa.bbcols
> 1:1-50 - equCab1.fa
TAAGCCCTCCTAGCACACACCCGGAGTGGCC-GGGCCGTAC-TTTCCTTTTC
> 2:1-49 + canFam2.fa
CAAGCCCTGC--GCGCTCAGCCGGAGTGTCCCGGGCCCTGC-TTTCCTTTTC
> 3:1-19 - mm9.fa
--------------------------------GGATCTACTTTTCCTCTTC
=
> 3:20-41 + mm9.fa
CTGGCGTCCGGAGCTGGGACGT
=
<BLANKLINE>
```

Bə növşə yi formatda tə k düzülüşü çap edə rkə n, metadata və mə lumatı tə min etmə k üçün açar söz arqumentlə rində n istifadə edin identifikatorlar:

```
>>> hizalama = hizalamalar[0]
>>> çap(alignment.format("bə növşə yi", metadata=metadata, identifikatorlar=identifikatorlar))
> 1:1-50 - equCab1.fa
TAAGCCCTCCTAGCACACACCCGGAGTGGCC-GGGCCGTAC-TTTCCTTTTC
> 2:1-49 + canFam2.fa
CAAGCCCTGC--GCGCTCAGCCGGAGTGTCCCGGGCCCTGC-TTTCCTTTTC
> 3:1-19 - mm9.fa
--------------------------------GGATCTACTTTTCCTCTTC
=
<BLANKLINE>
```

6.7.13 Ardıcıllıqla Alignment/Map (SAM)

Sequence Alignment/Map (SAM) formatında [29] fayllar , adə tə n, ikili ardıcıl düzülmə lə ri saxlayır. bir istinad genomuna qarşı yeni nə sil ardıcıllıq mə lumatları. Biopython test paketində ki ex1.sam faylıdır SAM formatında minimal fayl nümunə si. Onun ilk bir neçə sə tri belə dir:

```
EAS56_57:6:190:289:82 69    chr1  100  0   *   =  100  0   CTCAAGGTTGTTGCA
EAS56_57:6:190:289:82 137   chr1  100  73  35M =  100  0   AGGGGTGCAGAGCCG
EAS51_64:3:190:727:308 99   chr1  103  99  35M =  263  195 GGTGCAGAGCCGAGT
...
```

Machine Translated by Google


```
>>> Bio idxaldan Align >>>
alignments = Align.parse("ex1.sam", " sam") >>> alignment =
next(alignments)
```

Birinci sə tirin bayrağı 69-dur. SAM/BAM fayl formatının spesifikasiyasına ə sasə n, bayraqda bit istiqamə tində 4 bayrağı olan sə tirlə r xə ritə də n çıxarılıb. 69-da bu mövqeyə uyğun bit doğru olaraq tə yin edildiyi üçün bu ardıcıllıq xə ritə sizdir və genomla uyğunlaşdırılmayıb (chr1-i göstə rə n birinci sə tirə baxmayaraq). Bu uyğunlaşdırmanın hə də fi (və ya alignment.sequences-də ki ilk element) Yoxdur:

```
>>> alignment.flag 69 >>> bin(69)


'0b1000101'
>>> zibil(4)
'0b100'
>>> if alignment.flag & 4:
...             print("xə ritə siz")
... başqa:
...             çap ("xə ritə lə nmiş")
...
unmapped
>>> alignment.sequences [Yox,
SeqRecord(seq=Seq('CTCAAGGTTGTTGCAAGGGGGTCTATGTGAACAAA'), id='EAS56_57:6:190:289:82', name='< unk >>> alignment.target Doğru deyil
```

İkinci sə tir 1-ci xromosoma uyğunluğu tə msil edir:

```
>>> alignment = next(alignments) >>> if alignment.flag &
4: print("unmapped") ... else:
...

...             çap ("xə ritə lə nmiş")
...

xə ritə lə nmiş >>>
alignment.target SeqRecord(seq=Yox, id='chr1', ad='<namə lum ad>', tə svir='', dbxrefs=[])
```

Bu SAM faylı hə r düzülmə üçün genom ardıcıllığı mə lumatını saxlamadığından, düzülüşü çap edə bilmirik. Bununla belə , biz hizalama mə lumatını SAM formatında və ya hə də f ardıcıllığı mə lumatını tə lə b etmə yə n hə r hansı digə r formatda (mə sə lə n, BED, 6.7.14-cü bölmə yə baxın) çap edə bilə rik:

```
>>> formatı (düzlə şdirmə , "sam")
'EAS56_57:6:190:289:82\t137\tchr1\t100\t73\t35M\t=\t100\t0\tAGGGGTGCAGAGCCGAGTCACGGGGTTGCCAGCAC\t<<<<<< >>> " formatlanmış "),
'chr1\t99\t134\tEAS56_57:6:190:289:82\t0\t+
\t99\t134\t0\t1\t35,\t0,\n'
```

Bununla belə , biz düzülüşü PSL formatında çap edə bilmə rik (bax. bölmə 6.7.16), çünki bunun üçün chr1 hə də f ardıcıllığının ölçüsünü bilmə k lazımdır:

```
>>> formatı(alignment, "psl") # doctest: +ELLIPSIS
Traceback (ə n son zə ng):
  ...
Tip xə tası: ...
```

Machine Translated by Google

Hə də f ardıcıllığının ölçüsünü bilirsinizsə , onları ə l ilə tə yin edə bilə rsiniz:

```
>>> Bio.Seq -də n idxal Seq
>>> Bio.SeqRecord -dan SeqRecord idxalı
>>> hə də f = SeqRecord (Seq(Heç biri, uzunluq=1575), id="chr1")
>>> alignment.target = hə də f
>>> formatı(alignment, "psl") # doctest: +ELLIPSIS
'35\t0\t0\t0\t0\t0\t0\t0\t+\tEAS56_57:6:190:289:82\t35\t0\t35\tchr1\t1575\t99\t134\t1\t35,\t0,\t99,\n
```

Biopython test paketində ki ex1_header.sam faylı eyni düzülmə lə ri ehtiva edir, lakin indi də daxildir başlıq. Onun ilk bir neçə sə tri belə dir:

```
@HD\tVN:1.3\tSO:koordinat
@SQ\tSN:chr1\tLN:1575
@SQ\tSN:chr2\tLN:1584
EAS56_57:6:190:289:82 69 chr1 100 0 * = 100 0 CTCAAGGTTGTTGCA
...
```

Başlıq hə də f xromosomların ölçüsü daxil olmaqla, düzülmə lə r haqqında ümumi mə lumatları saxlayır.
Hə də f mə lumatı alignments obyektinin targets atributunda saxlanılır:

```
>>> Bio import Align- də n
>>> alignments = Align.parse("ex1_header.sam", "sam")
>>> len(alignments.targets)
2
>>> alignments.targets[0]
SeqRecord(seq=Seq(Heç biri, uzunluq=1575), id='chr1', ad='<namə lum ad>', tə svir='', dbxrefs=[])
>>> alignments.targets[1]
SeqRecord(seq=Seq(Heç biri, uzunluq=1584), id='chr2', ad='<namə lum ad>', tə sviri='', dbxrefs=[])
```

Başlıqda göstə rilə n digə r mə lumatlar metadata atributunda saxlanılır:

```
>>> alignments.metadata
{'HD': {'VN': '1.3', 'SO': 'koordinat'}}
```

Hə də f mə lumatı ilə biz indi düzülüşü PSL formatında da çap edə bilə rik:

```
>>> alignment = next(alignments) # xə ritə siz ardıcıllıq; keçin
>>> hizalama = sonrakı (düzlə şdirmə )
>>> formatı (düzlə şdirmə , "psl")
'35\t0\t0\t0\t0\t0\t0\t0\t+\tEAS56_57:6:190:289:82\t35\t0\t35\tchr1\t1575\t99\t134\t1\t35,\t0,\t99,\n
```

Biz indi hizalanmanı insan tə rə fində n oxuna bilə n formada da çap edə bilə rik, lakin nə zə rə alın ki, hə də f ardıcıllığı mə zmunu bu faylda mövcud deyil:

```
>>> çap (düzlə şdirmə )
chr1                99 ????????? 134
                     0 ............................... 35
EAS56_57:           0 AGGGGTGCAGAGCCGAGTCACGGGGTTGCCAGCAC 35
<BLANKLINE>
```

Biopython test də stində ki sam1.sam faylında düzülmə lə rin necə olduğunu göstə rə n ə lavə MD etiketi var.
sorğu ardıcıllığı hə də f ardıcıllığından fə rqlə nir:

```
@SQ SN:1 LN: 239940
@PG ID:bwa PN:bwa VN:0.6.2-r126
HWI-1KL120:88:D0LRBACXX:1:1101:1780:2146 77 * 0 0 * * 0
...
HWI-1KL120:88:D0LRBACXX:1:1101:2852:2134 137 1 136186 25 101M = 136186
```

Machine Translated by Google

Ayrışdırıcı MD etiketində n yerli genom ardıcıllığını yenidə n qurur və bizə hə də f ardıcıllığı görmə yə imkan verir. düzülmə ni çap edə rkə n açıq şə kildə :

```
>>> Bio import Align- də n
>>> hizalamalar = Align.parse("sam1.sam", "sam")
>>> düzülmə lə rdə düzülmə üçün :
...         alignment.flag & 4 deyilsə : # Xə ritə siz sə tirlə ri keçin
...             fasilə
...
>>> alignment # doctest: +ELLIPSIS
<Hazırlama obyekti (2 sə tir x 101 sütun) ...>
>>> çap (düzlə şdirmə )
1                   136185 TCACGGTGGCCTGTTGAGGCAGGGGTCACGCTGACCTCTGTCCGCGTGGGAGGGGCCGG
                         0 ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
HWI-1KL12                0 TCACGGTGGCCTGTTGAGGCAGGGGCTCACGCTGACCTCTCGGCGTGGGAGGGGCCGG
<BLANKLINE>
1                   136245 TGTGAGGCAAGGGCTCACACTGACCTCTCTCAGCGTGGGAG 136286
                        60 ||||||||||||||||||||||||||||||||||||||||||| 60                101
HWI-1KL12               TGTGAGGCAAGGGCTCACGCTGACCTCTCTCGGCGTGGGAG                         101
<BLANKLINE>
```

SAM faylları sadə ardıcıllıqla ə lavə lə ri və silinmə lə ri ayırd etmə k üçün ə lavə mə lumatı ehtiva edə bilə r genomun atlanmış bölgə lə rində n (mə sə lə n, intronlar), sə rt və yumşaq kə smə və yastıqlı ardıcıllıq bölgə lə rində n. kimi bu mə lumat Alignment obyektinin koordinat atributunda saxlanıla bilmə z və a-da saxlanılır ə və zinə xüsusi ə mə liyyatlar atributu. Nümunə olaraq bu SAM faylında üçüncü düzülüşü istifadə edə k:

```
>>> Bio import Align- də n
>>> alignments = Align.parse("dna_rna.sam", "sam")
>>> hizalama = sonrakı (düzlə şdirmə )
>>> hizalama = sonrakı (düzlə şdirmə )
>>> hizalama = sonrakı (düzlə şdirmə )
>>> çap (format(düzlə şdirmə , "SAM")) # doctest: +NORMALIZE_WHITESPACE
NR_111921.1          0          chr3          48663768          0          46M1827N82M3376N76M12H          *
<BLANKLINE>
>>> hizalama.koordinatlar
massiv([[48663767, 48663813, 48665640, 48665722, 48669098, 48669174],
        [          0, 46, 204]])   46,                    128,          128,
>>> alignment.operations
bytearray(b'MNMNM')
>>> alignment.query.annotations["hard_clip_right"]
12
```

Bu düzülmə də 63M1062N75M468N43M siqar simi 46 düzülmüş nukleotidi, 1827 nukleotiddə n ibarə t bir intronu, 82 düzülmüş nukleotidi, 3376 nukleotiddə n ibarə t bir intronu, 76 düzlə nmiş, 76 düzlə nmiş nukleotidlə ri və 1827 nukleotidlə ri müə yyə n edir. nu-kleotidlə r. Saxlanılan sə rt kə smə istisna olmaqla, bu ə mə liyyatlar ə mə liyyatlar atributunda göstə rilir alignment.query.annotations["hard_clip_right"] (və ya alignment.query.annotations["hard_clip_left"], ə gə r varsa) ə və zinə .

Sıfırdan yaradılmış hizalamalarla SAM faylını yazmaq üçün Alignments (cə m) obyektində n istifadə edin (bax. Bölmə 6.5) düzülmə lə ri, hə mçinin metadata və hə də flə ri saxlamaq üçün:

```
>>> io import StringIO- dan
>>> np kimi idxal numpy

>>> Bio import Align- də n
```

Machine Translated by Google

```
>>> Bio.Seq -də n idxal Seq
>>> Bio.SeqRecord -dan SeqRecord idxalı

>>> alignments = Align.Alignments()

>>> seq1 = Seq(Heç biri, uzunluq=10000)
>>> target1 = SeqRecord(seq1, id="chr1")
>>> seq2 = Seq(Heç biri, uzunluq=15000)
>>> target2 = SeqRecord(seq2, id="chr2")
>>> alignments.targets = [hə də f1, hə də f2]
>>> alignments.metadata = {"HD": {"VN": "1.3", "SO": "koordinat"}}

>>> seqA = Seq(Heç biri, uzunluq=20)
>>> queryA = SeqRecord(seqA, id="readA")
>>> ardıcıllıqlar = [hə də f1, sorğuA]
>>> koordinatları = np.array([[4300, 4320], [0, 20]])
>>> alignment = Align.Alignment(ardıcıllıqlar, koordinatlar)
>>> alignments.append(alignment)

>>> seqB = Seq(Heç biri, uzunluq=25)
>>> queryB = SeqRecord(seqB, id="readB")
>>> ardıcıllıqlar = [hə də f1, sorğuB]
>>> koordinatları = np.array([[5900, 5925], [25, 0]])
>>> alignment = Align.Alignment(ardıcıllıqlar, koordinatlar)
>>> alignments.append(alignment)

>>> seqC = Seq(Heç biri, uzunluq=40)
>>> queryC = SeqRecord (seqC, id="readC")
>>> ardıcıllıqlar = [hə də f2, sorğuC]
>>> koordinatları = np.array([[12300, 12318], [0, 18]])
>>> alignment = Align.Alignment(ardıcıllıqlar, koordinatlar)
>>> alignments.append(alignment)

>>> axın = StringIO()
>>> Align.write (düzlə şdirmə , axın, "sam")
3
>>> print(stream.getvalue()) # doctest: +NORMALIZE_WHITESPACE
@HD VN:1.3 SO:koordinat
@SQ	SN: chr1	LN: 10000
@SQ	SN: chr2	LN: 15000
readA	0	chr1	4301	255	20M	*	0	0	*
oxu B	16	chr1	5901	255	25M	*	0	0	*
oxu C	0	chr2	12301	255	18M22S	*	0	0	*
<BLANKLINE>
```

6.7.14 Genişlə ndirilə bilə n Brauzer Mə lumatı (BED)

BED (Browser Extensible Data) faylları adə tə n gen transkriptlə rinin uyğunlaşmalarını saxlamaq üçün istifadə olunur. genom. UCSC-də n tə svirə baxın BED formatının tam izahı üçün.

BED fayllarında üç tə lə b olunan sahə və doqquz ə lavə sahə var. Alt kataloqda bed12.bed faylı Testlə r/Blat 12 sahə li BED faylının nümunə sidir:

Machine Translated by Google

```
chr22    1000    5000    mRNA1    960    +    1200    4900    255,0,0
chr22    2000    6000    mRNA2    900    -    2300    5960    0,255,0
```

Bu faylı tə hlil etmə k üçün istifadə edin

```
>>> Bio import Align- də n
>>> alignments = Align.parse("yataq12.yataq", "yataq")
>>> lent (düzlə şdirmə )
2
>>> düzülmə lə rdə düzülmə üçün :
...         çap (alignment.coordinates)
...
[[1000 1567 4512 5000]
  [ 0 567 567 1055]]
[[2000 2433 5601 6000]
  [ 832 399 399 0]]
```

Qeyd edə k ki, birinci ardıcıllıq (“mRNA1”) irə li zə ncirlə , ikinci ardıcıllıq isə (“mRNA2”) ə ks zə ncirlə ə laqə lə ndirildi.

BED faylı hə r bir xromosomun uzunluğunu saxlamadığı üçün hə də f ardıcıllığın uzunluğu onun maksimumu:

```
>>> hizalama.hə də f
SeqRecord(seq=Seq(Heç biri, uzunluq=9223372036854775807), id='chr22', ad='<namə lum ad>', tə svir='',
```

Sorğu ardıcıllığının uzunluğu onun düzülmə mə lumatından ə ldə edilə bilə r:

```
>>> alignment.query
SeqRecord(seq=Seq(Yox, uzunluq=832), id='mRNA2', ad='<namə lum ad>', tə sviri='', dbxrefs=[])
```

Uyğunlaşma hesabı (sahə 5) və 7-9 sahə lə rində saxlanılan mə lumat (qalın Başlanğıc, qalın Son və itemRgb BED formatının spesifikasiyası) hizalama obyektində atributlar kimi saxlanılır:

```
>>> alignment.score
900.0
>>> alignment.thickStart
2300
>>> alignment.thickEnd
5960
>>> alignment.itemRgb
'0,255,0'
```

BED formatında düzülmə ni çap etmə k üçün Python-un daxili format funksiyasından istifadə edə bilə rsiniz:

```
>>> çap (format(düzlə şdirmə , "yataq")) # doctest: +NORMALIZE_WHITESPACE
chr22    2000    6000    mRNA2    900    -    2300    5960    0,255,0
<BLANKLINE>
```

və ya hizalanma obyektinin format metodundan istifadə edə bilə rsiniz. Bu, sahə lə rin sayını tə yin etmə yə imkan verir bedN açar söz arqumenti kimi yazılacaq:

```
>>> çap (alignment.format("yataq")) # doctest: +NORMALIZE_WHITESPACE
chr22 2000 900    6000    mRNA2    -    2300    5960    0,255,0
<BLANKLINE>
>>> çap (alignment.format("yataq", 3)) # doctest: +NORMALIZE_WHITESPACE
chr22    2000    6000
```

Machine Translated by Google

```
<BLANKLINE>
>>> çap (alignment.format("yataq", 6)) # doctest: +NORMALIZE_WHITESPACE
chr22 900          2000          6000          mRNA2                    -
<BLANKLINE>
```

Eyni açar söz arqumenti Align.write ilə istifadə edilə bilə r:

```
>>> alignments.rewind()
>>> Align.write(alignments, "mybed3file.bed", "yataq", bedN=3)
2
>>> alignments.rewind()
>>> Align.write(alignments, "mybed6file.bed", "yataq", bedN=6)
2
>>> alignments.rewind()
>>> Align.write(alignments, "mybed12file.bed", "yataq")
2
```

6.7.15 böyük yataq

bigBed fayl formatı BED faylının 6.7.14 indeksli ikili versiyasıdır. bigBed faylı yaratmaq üçün siz ya UCSC-də n bedToBigBed proqramından istifadə edə bilə r (). və ya zə ng edə rə k bunun üçün Biopython istifadə edə bilə rsiniz Bio.Align.write funksiyası fmt="bigbed" ilə . İki üsul eyni bigBed ilə nə ticə lə nmə lidir faylları, bedToBigBed-də n istifadə etmə k daha sürə tli və daha etibarlı ola bilə r, çünki bu, qızıl standartdır. BigBed faylları kimi daxili indekslə gə lir, bu, müə yyə n bir genomik bölgə ni tez bir zamanda axtarmağa imkan verir.

Nümunə olaraq, Tests/Blat alt kataloqunda mövcud olan dna_rna.bb bigBed faylını tə hlil edə k. Biopython paylanması:

```
>>> Bio import Align- də n
>>> alignments = Align.parse("dna_rna.bb", "bigbed")
>>> lent (düzlə şdirmə )
4
>>> çap (alignments.declaration) # doctest: +NORMALIZE_WHITESPACE
stolüstü çarpayı
"Brauzer Genişlə ndirilə bilə n Mə lumat"
(
    string uint        xrom;                    "İstinad ardıcıllığı xromosomu və ya iskele"
                       xromStart;               "Xromosomda başlanğıc mövqeyi"
    uint               xromEnd;                 "Xromosomda son mövqe"
    string uint        ad;                      "Elementin adı."
                       xal;                     "Xal (0-1000)"
    simvol[1]          ip; qalın                "+ və ya - ip üçün"
    uint               başlanğıc; qalın         "Ekranın qalın olması lazım olan yerdə n başlanğıc (kodon başlanğıcı)"
    uint               son; qorunur;            "Ekranın qalın olması lazım olan yerin sonu (stop kodonu)"
    uint               blockCount;              "22-11-2004 tarixində n etibarə n itemRgb kimi istifadə edilmişdir"
    int                int[blockCount]          "Blokların sayı"
    blockÖlçülə ri; int[blockCount] chromStarts; "Blok ölçülə rinin vergüllə ayrılmış siyahısı"
                                                 "ChromStart-a nisbə tə n başlanğıc mövqelə ri"
)
<BLANKLINE>
```

Bə yannamə də sütunların yaradılması üçün istifadə edilə n AutoSql formatında spesifikasiya var bigBed faylı. Hə də f ardıcıllıqları (adə tə n, ardıcıllığın düzüldüyü xromosomlar) hə də flə r atributunda saxlanılır. BigBed formatında yalnız hə r bir hə də fin identifikatoru və ölçüsü var saxlanılır. Bu nümunə də yalnız bir xromosom var:

Machine Translated by Google

```
>>> alignments.hə də flə r
[SeqRecord(seq=Seq(Heç biri, uzunluq=198295559), id='chr3', ad='<namə lum ad>', tə sviri='<namə lum des
```

Fə rdi uyğunlaşmalara baxaq. Hizalanma mə lumatı BED üçün olduğu kimi saxlanılır fayl (bax bölmə 6.7.14):

```
>>> hizalama = sonrakı (düzlə şdirmə )
>>> alignment.target.id
'chr3'
>>> alignment.query.id
'NR_046654.1'
>>> hizalama.koordinatlar
massiv([[42530895, 42530958, 42532020, 42532095, 42532563, 42532606],
        [       181, 118,      118,                          43,          43,              0]])
>>> alignment.thickStart
42530895
>>> alignment.thickEnd
42532606
>>> çap(alignment) # doctest: +ELLIPSIS
chr3 42530895 ????????????????
                          0 ||||||||||||||||||||||||||||||||||||||||||||||||||||||||
NR_046654              181 ????????????????
<BLANKLINE>
chr3 42530955 ????????????????
                         60 |||------------------------------------------------------------
NR_046654              121 ???------------------------------------------------------------
...
chr3            42532515 ????????????????
                       1620 ----------------------------------------------||||||||||||
NR_046654               43 ---------------------------------------------- ????????????
<BLANKLINE>
chr3            42532575 ?????????????????? 42532606
                       1680 |||||||||||||||||||||||||||||||||||                    1711
NR_046654               31 ??????????????????                                          0
<BLANKLINE>
```

Defolt bigBed formatı hə də f və sorğunun ardıcıl mə zmununu saxlamır. Bunlar varsa başqa yerdə (mə sə lə n, Fasta faylı) göstə rmə k üçün alignment.target.seq və alignment.query.seq tə yin edə bilə rsiniz. hizalamanı çap edə rkə n mə zmunun ardıcıllığını və ya hizalamağı tə lə b edə n formatlarda yazmaq üçün ardıcıllığın mə zmunu (mə sə lə n, Clustal, bölmə 6.7.2-yə baxın). Testlə rdə test_Align_bigbed.py test skripti Biopython paylanmasındakı alt kataloq bunun necə edilə cə yinə dair bə zi nümunə lə r verir.

İndi ardıcıllıq bölgə sini necə axtaracağımıza baxaq. Bunlar bigBed faylında saxlanılan ardıcıllıqlardır, BED formatında çap edilmiş (bax bölmə 6.7.14):

```
>>> alignments.rewind()
>>> düzülmə lə rdə düzülmə üçün :
...         çap(format(düzlə şdirmə , "yataq")) # doctest: +NORMALIZE_WHITESPACE
...
chr3        42530895        42532606        NR_046654.1        1000        -        42530895        425
<BLANKLINE>
chr3        42530895        42532606        NR_046654.1_də yişdirildi        978        -        42530895
<BLANKLINE>
chr3        48663767        48669174        NR_111921.1        1000        +        48663767        486
```

Machine Translated by Google

```
<BLANKLINE>
chr3            48663767            48669174            NR_111921.1_modified            972            +            48663767
<BLANKLINE>
```

48000000 mövqelə ri arasında chr3-də bölgə lə ri tapmaq üçün hizalama obyektində axtarış metodundan istifadə edin və 49000000. Bu üsul iteratoru qaytarır:

```
>>> seçilmiş_alignments = alignments.search("chr3", 48000000, 49000000)
>>> seçilmiş_alignments -də düzülmə üçün :
...         çap (alignment.query.id)
...
NR_111921.1
NR_111921.1_modified
```

Xromosom adı bütün xromosomları ehtiva etmə k üçün Yox ola bilə r və başlanğıc və son mövqelə r ola bilə r. 0 mövqeyində n axtarışa başlamaq və ya müvafiq olaraq xromosomun sonuna qə də r axtarışa davam etmə k üçün heç biri.

BigBed formatında düzülmə lə rin yazılması Bio.Align.write-ə zə ng etmə k qə də r asandır:

```
>>> Align.write(alignments, "output.bb", "bigbed")
```

Siz bigBed faylına daxil edilə cə k BED sahə lə rinin sayını tə yin edə bilə rsiniz. Mə sə lə n, BED6 yazmaq üçün fayl, istifadə

```
>>> Align.write(alignments, "output.bb", "bigbed", bedN=6)
```

bedToBigBed üçün olduğu kimi, bigBed çıxışına ə lavə sütunlar daxil edə bilə rsiniz. Tutaq ki, fayl bedExample2.as (Biopython paylanmasının Tests/Blat alt kataloqunda mövcuddur) daxil edilmiş BED sahə lə rinin bə yannamə sini AutoSql formatında saxlayır. Bu bə yannamə ni belə oxuya bilə rik:

```
>>> Bio.Align idxal bigbed -də n
>>> axın olaraq open("bedExample2.as") ilə :
...         autosql_data = stream.read()
...
>>> bə yannamə = bigbed.AutoSQLTable.from_string(autosql_data)
>>> növü (bə yannamə )
<sinfi 'Bio.Align.bigbed.AutoSQLTable'>
>>> çap (bə yannamə )
masa hg18KGchr7
"Rə ng və GeneSymbol və SwissProtID ilə chr7 üçün UCSC Genlə ri"
(
    simli xrom; uint chromStart;            "İstinad ardıcıllığı xromosomu və ya iskele"
    xromEnd; sə tir adı; uint bal; char[1]  "Xromosomdakı xüsusiyyə tin başlanğıc mövqeyi"
    uint        ip; qalın                   "Xromosomdakı xüsusiyyə tin son mövqeyi"
    Başlanğıc; qalın son;                   "Genin adı"
    qorunur; string genSymbol;              "Qal"
    string spID;                            "+ və ya - ip üçün"
                                            "Kodlaşdırma bölgə sinin başlanğıcı"
                                            "Kodlaşdırma bölgə sinin sonu"
    uint                                    "Yaşıl + iplik, Qırmızı iplik"
                                            "Gen simvolu"
                                            "SWISS-PROT protein qoşulma nömrə si"
)
<BLANKLINE>
```

İndi biz 9 BED sahə lə ri və ə lavə geneSymbol və spID sahə lə ri ilə bigBed faylı yaza bilə rik. zə ng edir

Machine Translated by Google

```
>>> Align.write(
...         hizalamalar,
...         "output.bb",
...         "böyük yataq",
...         çarpayıN=9,
...         bə yannamə =bə yannamə ,
...         extraIndex=["ad", "geneSymbol"],
... )
```

Burada biz hə mçinin bigBed faylına ad və genSymbol üzrə ə lavə indekslə rin daxil edilmə sini xahiş etdik.
Align.write geneSymbol və spID açarlarını alignment.annotations lüğə tində tapmağı gözlə yir.
BigBed formatında hizalama fayllarının yazılması ilə bağlı daha çox nümunə üçün Biopython paylamasındakı Testlə r alt kataloqunda test_Align_bigbed.py test skriptinə müraciə t edin.

6.7.16 Pattern Space Layout (PSL)

PSL (Pattern Space Layout) faylları BLAST-Like Alignment Tool BLAT [24] tə rə fində n yaradılır . kimi
BED faylları (bax bölmə 6.7.14), PSL faylları adə tə n transkriptlə rin genomlara uyğunlaşdırılmasını saxlamaq üçün istifadə olunur.
Bu, qısa BLAT faylının nümunə sidir (bunun Tests/Blat alt kataloqunda dna_rna.psl kimi mövcuddur)
Biopython paylanması), 5 sə tirdə n ibarə t standart PSL başlığı ilə :

psLayout versiyası 3

uyğun	sə hv uyğun	rep. uyğun	N	Q boşluq sayı	Q boşluq ə sasları	T boşluq sayı	T boşluq ə sasları	s
165	0	39	0	0	0	2	5203	+	NR_111921.1
175	0	6	0	0	0	2	1530	-	NR_046654.1
162	2	39	0	1	2	3	5204	+	NR_111921.1_modi
172	1	6	0	1	3	3	1532	-	NR_046654.1_modif

Bu faylı tə hlil etmə k üçün istifadə edin

```
>>> Bio import Align- də n
>>> alignments = Align.parse("dna_rna.psl", "psl")
>>> alignments.metadata
{'psLayout versiyası': '3'}
```

Hə r sə tir üçün bir Alignment obyekti ə ldə etmə k üçün hizalanmaları tə krarlayın:

```
>>> düzülmə lə rdə düzülmə üçün :
...         çap (alignment.target.id, alignment.query.id)
...
chr3 NR_046654.1
chr3 NR_046654.1_modified
chr3 NR_111921.1
chr3 NR_111921.1_modified
```

Son hizalanmaya daha ə traflı baxaq. PSL faylındakı ilk dörd sütunun sayını göstə rir
uyğunsuzluqların sayı, tə krar bölgə lə rə uyğunlaşdırılmış nukleotidlə rin sayı və sayı
N (namə lum) simvola uyğunlaşdırılmış nukleotidlə rin. Bu də yə rlə r Alignment atributları kimi saxlanılır
obyekt:

```
>>> alignment.matches
162
```

Machine Translated by Google

```
>>> alignment.misMatches
2
>>> alignment.repMatches
39
>>> alignment.nCount
0
```

Hə də f və sorğunun ardıcıllıq mə lumatları PSL faylında açıq şə kildə saxlanmadığından, ardıcıllığın mə zmunu alignment.target və alignment.query müə yyə n edilmə mişdir. Bununla belə , onların ardıcıllıq uzunluğu mə lumdur:

```
>>> alignment.target # doctest: +ELLIPSIS
SeqRecord(seq=Seq(Heç biri, uzunluq=198295559), id='chr3', ...)
>>> alignment.query # doctest: +ELLIPSIS
SeqRecord(seq=Seq(Heç biri, uzunluq=220), id='NR_111921.1_modified', ...)
```

Düzə ltmə ni BED və ya PSL formatında çap edə bilə rik:

```
>>> çap (format(düzlə şdirmə , "yataq")) # doctest: +NORMALIZE_WHITESPACE
48663767 48669174 chr3 NR_111921.1_modified 0 + 48663767
<BLANKLINE>
>>> çap (format(alignment, "psl")) # doctest: +NORMALIZE_WHITESPACE
162 2 5204 39 0 1 2 3 + NR_111921.1_modi
<BLANKLINE>
```

Burada uyğunluqların sayı, uyğunsuzluqlar, tə krar bölgə uyğunluqları və namə lum nukleotidlə rə uyğunluqlar Alignment obyektinin müvafiq atributlarından götürülmüşdür. Bu atributlar mövcud deyilsə , mə sə lə n, düzülmə PSL faylından gə lmə yibsə , bu rə qə mlə r istifadə edə rə k hesablanır varsa, ardıcıllığın mə zmunu. Hə də f ardıcıllığında tə krar bölgə lə r ardıcıllığın maskalanması ilə göstə rilir maska açar sözü arqumenti üçün aşağıdakı də yə rlə rlə müə yyə n edildiyi kimi kiçik və ya böyük hə rf simvolları kimi:

- Yanlış (defolt): Uyğunluqları maskalı ardıcıllıqla ayrıca saymayın;
- "aşağı": kiçik hə rflə rlə uyğunluqları tə krar bölgə lə rə uyğunluq kimi sayın və hesabat verin;
- "yuxarı": Böyük hə rflə rlə uyğunluqları tə krar bölgə lə rə uyğunluq kimi sayın və hesabat verin;

Namə lum nukleotidlə r üçün istifadə olunan xarakter wildcard arqumenti ilə müə yyə n edilir. İlə uyğunluq üçün BLAT, joker simvol standart olaraq "N"-dir. Uyğunluqları saymaq istə mirsinizsə , wildcard=None istifadə edin hə r hansı namə lum nukleotidlə r ayrıca.

```
>>> idxal numpy
>>> Bio import Align- də n
>>> sorğu = "GGTGGGGG"
>>> hə də f = "AAAAAAAggggGGNGAAAAA"
>>> koordinatları = numpy.array([[0, 7, 15, 20], [0, 0, 8, 8]])
>>> alignment = Align.Alignment([hə də f, sorğu], koordinatlar)
>>> çap (düzlə şdirmə )
hə də f 0 AAAAAAggggGGNGAAAAA 20
        0 -------....||.|----- 20
sorğu   0 -------GGTGGGGG----- 8
<BLANKLINE>
>>> xə tt = alignment.format("psl")
>>> print(line) # doctest: +NORMALIZE_WHITESPACE
6 1 0 0 1 0 0 0 8 hə də f 20 7 15 1 8, 0, 7,
sorğu 8 0 >>> sə tir = alignment.format("psl", mask="aşağı")
```

Machine Translated by Google

```
>>> print(line) # doctest: +NORMALIZE_WHITESPACE
3 1 0 0 0 1 + sorğu 3 8 0 8 hə də f 20 7 0
>>> line = alignment.format("psl", mask="aşağı", joker simvol=Yox)
15 1 8, 0, 7,

>>> print(line) # doctest: +NORMALIZE_WHITESPACE
3 2 3 0 0 0 0 0 + sorğu 8 0 8 hə də f 20 7 15 1 8, 0, 7,
```

Eyni arqumentlə r Bio.Align.write istifadə edə rə k PSL formatında çıxış faylına düzülmə lə r yazarkə n istifadə edilə bilə r. Bu funksiyanın PSL başlığının olub olmadığını göstə rə n ə lavə açar söz başlığı (defolt olaraq doğrudur) var yazılıb.

Formatlama metoduna ə lavə olaraq, Python-un daxili format funksiyasından istifadə edə bilə rsiniz:

```
>>> çap (format(alignment, "psl")) # doctest: +NORMALIZE_WHITESPACE
6 0 0 0 1 + sorğu 0 8 8 hə 1 lə f 20 7 0 0 15 1 8, 0, 7,
```

Alignment obyektlə rinin Python-da formatlanmış (f-) sə tirlə rdə istifadə edilmə sinə icazə verir:

```
>>> line = f"PSL formatında düzülmə '{alignment:psl}'-dir."
>>> print(line) # doctest: +NORMALIZE_WHITESPACE
PSL formatında düzülmə '6 1 0-dır 0 1 0 0 0 + sorğu 8 0 8 hə də f 20 7
'
```

Qeyd edə k ki, ə lavə açar söz arqumentlə ri format funksiyası və ya formatlanmış sə tirlə rlə istifadə edilə bilmə z.

6.7.17 bigPsl

bigPsl faylı ə vvə lcə də n tə yin edilmiş 12 BED sahə sində n və BED12+13 formatlı bigBed faylıdır. 13 xüsusi sahə AutoSql bigPsl.as faylında müə yyə n edilmişdir PSL faylının indekslə şdirilmiş ikili versiyasını yaratmaqla UCSC tə rə fində n tə min edilir (bax. bölmə 6.7.16). BigPsl faylı yaratmaq üçün pslToBigPsl və ya istifadə edə bilə rsiniz UCSC-də n bedToBigBed proqramları. və ya Bio.Align.write funksiyasına zə ng edə rə k Biopython-dan istifadə edə bilə rsiniz fmt="bigpsl" ilə . İki üsul eyni bigPsl faylları ilə nə ticə lə nsə də , UCSC alə tlə ri çoxdur daha sürə tli və daha etibarlı ola bilə r, çünki bu, qızıl standartdır. bigPsl faylları bigBed faylları olduğundan, onlar ilə birlikdə gə lirlə r müə yyə n bir genomik bölgə ni tez bir zamanda axtarmağa imkan verə n daxili indeks.

Nümunə olaraq, Tests/Blat alt kataloqunda mövcud olan dna_rna.psl.bb bigBed faylını tə hlil edə k. Biopython paylanması. Bu fayl bigBed dna_rna.bb faylının bigPsl ekvivalentidir (bax. bölmə 6.7.15) və PSL faylının dna_rna.psl (bax bölmə 6.7.16).

```
>>> Bio import Align- də n
>>> alignments = Align.parse("dna_rna.psl.bb", "bigpsl")
>>> lent (düzlə şdirmə )
4
>>> çap (alignments.declaration) # doctest: +NORMALIZE_WHITESPACE
masa bigPsl
"bigPsl ikili düzülmə "
(
    string uint     xrom;              "İstinad ardıcıllığı xromosomu və ya iskele"
                    xromStart;         "Xromosomda başlanğıc mövqeyi"
    uint            xromEnd;           "Xromosomda son mövqe"
    string uint     ad;                "Elementin adı və ya ID-si, ideal olaraq hə m insan tə rə fində n oxuna bilə n, hə m də unikaldır"
                    xal;               "Xal (0-1000)"
    simvol[1]       ip; qalın          "+ və ya - sorğunun + və ya - xə ttinə uyğun olub-olmadığını göstə rir
    uint            başlanğıc; qalın   "Ekranın qalın olması lazım olan yerdə n başlanğıc (kodon başlanğıcı)"
    uint            son; qorunur;      "Ekranın qalın olması lazım olan yerin sonu (stop kodonu)"
    uint            blockCount;        "RGB də yə ri (giriş faylında R, G, B sə tirində n istifadə edin)"
    int                                "Blokların sayı"
```

Machine Translated by Google

```
    int[blockCount] blockÖlçülə ri; int[blockCount]          "Blok ölçülə rinin vergüllə ayrılmış siyahısı"
    chromStarts; oChromStart uint; oChromEnd; oStrand;      "ChromStart-a nisbə tə n başlanğıc mövqelə ri"
    oChromSize; int[blockCount] oChromStarts; lstring       "Başqa xromosomda başlanğıc mövqeyi"
    uint                oSequence; simli                    "Başqa xromosomda son mövqe"
    simvol[1]           oCDS; uint                          "+ və ya -, - o demə kdir ki, psl BED-ə uyğun koordinata çevrilib
    uint                xromSize; uyğunluq;                 "Başqa xromosomların ölçüsü."
    uyğunsuzluq; repMatch; nCount; seqType;                 "OChromStart-a və ya oChromStart+oChrom-a nisbə tə n başlanğıc mövqelə ri
                                                            "Digə r xromda ardıcıllıq (və ya redaktə siyahısı və ya boş)"
                                                            "NCBI formatında CDS"
                                                            "Hə də f xromosomunun ölçüsü"
    uint                                                    "Uyğunlaşan bazaların sayı."
    uint                                                    "Uyğun olmayan bazaların sayı"
    uint                                                    "Uyğun gə lə n, lakin tə krarların bir hissə si olan ə sasların sayı"
    uint                                                    "'N' ə sasların sayı"
    uint                                                    "0 = boş, 1 = nukleotid, 2 = amin turşusu"
)
<BLANKLINE>
```

Bə yannamə də bigPsl.as AutoSql faylı tə rə fində n müə yyə n edilmiş sütunların spesifikasiyası var. UCSC. Hə də f ardıcıllıqları (adə tə n, ardıcıllığın düzüldüyü xromosomlar) saxlanılır. hə də flə r atributunda. BigBed formatında hə r bir hə də fin yalnız identifikatoru və ölçüsü saxlanılır. In bu misalda yalnız bir xromosom var:

```
>>> alignments.hə də flə r
 [SeqRecord(seq=Seq(Heç biri, uzunluq=198295559), id='chr3', ad='<namə lum ad>', tə sviri='<namə lum des
```

Düzə lişlə r üzə rində tə krarlama hə r sə tir üçün bir Alignment obyekti verir:

```
>>> düzülmə lə rdə düzülmə üçün :
...         çap (alignment.target.id, alignment.query.id)
...
chr3 NR_046654.1
chr3 NR_046654.1_modified
chr3 NR_111921.1
chr3 NR_111921.1_modified
```

Fə rdi uyğunlaşmalara baxaq. Hizalama mə lumatı ilə eyni şə kildə saxlanılır müvafiq PSL faylı (bax bölmə 6.7.16):

```
>>> hizalama.koordinatlar
massiv([[48663767, 48663795, 48663796, 48663813, 48665640, 48665716,
         48665716, 48665722, 48669098, 48669174],
        [        3,        31,        31,        48,        48,       124,
               126,       132,       132,       208]])
>>> alignment.thickStart
48663767
>>> alignment.thickEnd
48669174
>>> alignment.matches
162
>>> alignment.misMatches
2
>>> alignment.repMatches
39
```

Machine Translated by Google

```
>>> alignment.nCount
0
```

Düzə ltmə ni BED və ya PSL formatında çap edə bilə rik:

```
>>> çap (format(düzlə şdirmə , "yataq")) # doctest: +NORMALIZE_WHITESPACE
chr3 48663767 48669174 NR_111921.1_modified 1000 + 48663767
<BLANKLINE>
>>> çap (format(alignment, "psl")) # doctest: +NORMALIZE_WHITESPACE
162 5204 2 39 0 1 2 3 + NR_111921.1_modi
<BLANKLINE>
```

bigPsl faylı bigBed faylının xüsusi halı olduğundan, siz hizalamalarda axtarış metodundan istifadə edə bilə rsiniz. xüsusi genomik bölgə lə rə uyğunlaşma tapmaq üçün obyekt. Mə sə lə n, chr3-də regionları axtara bilə rik mövqelə r 48000000 və 49000000:

```
>>> seçilmiş_alignments = alignments.search("chr3", 48000000, 49000000)
>>> seçilmiş_alignments -də düzülmə üçün :
...     çap (alignment.query.id)
...
NR_111921.1
NR_111921.1_modified
```

Xromosom adı bütün xromosomları ehtiva etmə k üçün Yox ola bilə r və başlanğıc və son mövqelə r ola bilə r. 0 mövqeyində n axtarışa başlamaq və ya müvafiq olaraq xromosomun sonuna qə də r axtarışa davam etmə k üçün heç biri.

Biopython ilə bigPsl faylı yazmaq üçün Bio.Align.write(alignments, "myfilename.bb", fmt="bigpsl") istifadə edin. burada myfilename.bb çıxış bigPsl faylının adıdır. Alternativ olaraq, üçün (ikili) axın istifadə edə bilə rsiniz çıxış. Ə lavə variantlar var

- sıxışdırın: Doğrudursa (defolt), zlib istifadə edə rə k mə lumatları sıxın; Ə gə r False, mə lumatları sıxmayın.
- extraIndex: İndekslə nə cə k ə lavə sütunların adları ilə sə tirlə rin siyahısı.
- cds: Doğrudursa, CDS tipli sorğu funksiyasını axtarın və onu PSL faylında NCBI üslubunda yazın (defolt: Yalan).
- fa: Doğrudursa, sorğu ardıcıllığını PSL faylına daxil edin (defolt: Yanlış).
- maska: Hə də f ardıcıllığında tə krarlanan bölgə lə rin maskalandığını və hesabatda bildirilmə li olduğunu göstə rin matçlar sahə sinin ə və zinə repMatches sahə si. Mə qbul də yə rlə rdir
  - Heç biri: maskalanma yoxdur (defolt);
  - “aşağı”: kiçik hə rflə rlə maskalanma;
  - “yuxarı”: böyük hə rflə rlə maskalanma.
- joker simvol: Cə hə nnə mdə ki joker xarakterə (namə lum nukleotidlə ri tə msil edə n) uyğunlaşmaları bildirin uyğunluqlar, uyğunsuzluqlar və ya repatches ə və zinə nCount sahə sində hə də f və ya sorğu ardıcıllığı sahə lə r. Defolt də yə r "N"-dir.

Uyğunluqların sayı, uyğunsuzluqlar, tə krar bölgə uyğunluğu , və namə lum nukleotidlə rə uyğunluqlar alınır.

Machine Translated by Google

6.7.18 Çoxlu Alignment Format (MAF)

MAF (Multiple Alignment Format) faylları insan tə rə fində n oxuna bilə n bir sıra çoxlu ardıcıl düzülmə lə ri saxlayır. format. MAF faylları adə tə n genomların bir-birinə uyğunlaşdırılmasını saxlamaq üçün istifadə olunur. Fayl ucsc_test.maf Biopython paylanmasının Tests/MAF alt kataloqunda sadə MAF faylının nümunə sidir:

```
trek adı=euArc görünürlüğü=paket mafDot=off frames="multiz28wayFrames" növOrder="hg16 panTro1 baboo ##maf version=1 scoring=tba.v8

# tba.v8 (((insan şimpanze) babun) (siçan siçovulu))
# multiz.v7
# maf_project.v5 _tba_right.maf3 siçan _tba_C
# single_cov2.v4 single_cov2 /dev/stdin

xal = 23262.0
s hg16.chr7 s        27578828 38 + 158545518 AAA-GGGAATGTTAACCAAATGA---ATTGTCTCTTACGGTG
panTro1.chr6 28741140 38 + 161576975 AAA-GGGAATGTTAACCAAATGA---ATTGTCTCTTACGGTG
s baboon 116834 38 + 4622798 AAA-GGGAATGTTAACCAAATGA---GTTGTCTCTTATGGTG
s mm4.chr6           53215344 38 + 151104725 -AATGGGAATGTTAAGCAAACGA---ATTGTCTCTCAGTGTG
s rn3.chr4           81344243 40 + 187371129 -AA-GGGGATGCTAAGCCAATGAGTTGTTGTCTCTCAATGTG

xal = 5062.0
s hg16.chr7 27699739 6 + 158545518 TAAAGA
s panTro1.chr6 28862317 6 + 161576975 TAAAGA
s baboon 241163 6 + 4622798 TAAAGA
s mm4.chr6 53303881 6 + 151104725 TAAAGA
s rn3.chr4 81444246 6 + 187371129 taagga

xal = 6636.0
s hg16.chr7 27707221 13 + 158545518 gcagctgaaaaca
s panTro1.chr6 28869787 13 + 161576975 gcagctgaaaaca
s baboon 249182 13 + 4622798 gcagctgaaaaca
s mm4.chr6           53310102 13 + 151104725 ACAGCTGAAAATA
```

Bu faylı tə hlil etmə k üçün istifadə edin

```
>>> Bio import Align- də n
>>> alignments = Align.parse("ucsc_test.maf", "maf")
```

Fayl başlığında göstə rilə n mə lumat (trek xə tti və "#" ilə başlayan sonrakı sə tirlə r) alignments obyektinin metadata atributu:

```
>>> alignments.metadata # doctest: +NORMALIZE_WHITESPACE
{'ad': 'euArc',
  'görünürlük': 'paket',
  'mafDot': 'off',
  'çə rçivə lə r': 'multiz28wayFrames',
  'speciesOrder': ['hg16', 'panTro1', 'baboon', 'mm4', 'rn3'],
  'tə svir': 'Nümunə uyğunluğu',
  'MAF Versiyası': '1',
  'Qal': 'tba.v8',
  'Şə rhlə r': ['tba.v8 (((insan şimpanze) babun) (siçan siçovulu))',
                'multiz.v7',
                'maf_project.v5 _tba_right.maf3 siçan _tba_C',
                'single_cov2.v4 single_cov2 /dev/stdin']}
```

Machine Translated by Google

Düzə lişlə ri tə krarlayaraq, MAF faylında hə r bir düzülmə bloku üçün bir Alignment obyekti ə ldə edirik:

```
>>> hizalama = sonrakı (düzlə şdirmə )
>>> alignment.score
23262.0
>>> {seq.id: alignment.sequences ardıcıllığı üçün len(seq) } # sə nə d testi: +NORMALIZE_WHITESPACE
{'hg16.chr7': 158545518,
  'panTro1.chr6': 161576975,
  'babun': 4622798,
  'mm4.chr6': 151104725,
  'rn3.chr4': 187371129}
>>> alignment.coordinates # doctest: +NORMALIZE_WHITESPACE
massiv([[27578828, 27578829, 27578831, 27578831, 27578850, 27578850, 27578866],
        [28741140, 28741141, 28741143, 28741143, 28741162, 28741162, 28741178],
        [ 116834, 116835, 116837, 116837, 116856, 116856, 116872],
        [53215344, 53215344, 53215346, 53215347, 53215366, 53215366, 53215382],
        [81344243, 81344243, 81344245, 81344245, 81344264, 81344267, 81344283]])
>>> çap (düzlə şdirmə )
hg16.chr7 27578828 AAA-GGGAATGTTAACCAAATGA---ATTGTCTCTTACGGTG 27578866
panTro1.c 28741140 AAA-GGGAATGTTAACCAAATGA---ATTGTCTCTTACGGTG 28741178
baboon 116834 AAA-GGGAATGTTAACCAAATGA---GTTGTCTCTTATGGTG 116872
mm4.chr6 53215344 -AATGGGAATGTTAAGCAAACGA---ATTGTCTCTCAGTGTG 53215382
rn3.chr4 81344243 -AA-GGGGATGCTAAGCCAATGAGTTGTTGTCTCTCAATGTG 81344283
<BLANKLINE>
>>> çap (format (düzlə şdirmə , "filip"))
5 42
hg16.chr7 AAA-GGGAATGTTAACCAAATGA---ATTGTCTCTTACGGTG
panTro1.chAAA-GGGAATGTTAACCAAATGA---ATTGTCTCTTACGGTG
baboon AAA-GGGAATGTTAACCAAATGA---GTTGTCTCTTATGGTG
mm4.chr6 -AATGGGAATGTTAAGCAAACGA---ATTGTCTCTCAGTGTG
rn3.chr4 -AA-GGGGATGCTAAGCCAATGAGTTGTTGTCTCTCAATGTG
<BLANKLINE>
```

“a” (düzlə şdirmə bloku) və “s” (ardıcıllıq) sə tirlə rinə ə lavə olaraq, MAF fayllarında “i” sə tirlə ri ola bilə r. Bu blokdan ə vvə l və sonra genom ardıcıllığı haqqında mə lumat, "e" sə tirlə ri haqqında mə lumat boşdur düzülmə hissə lə ri və hə r düzülmüş ə sasın keyfiyyə tini göstə rə n “q” xə tlə ri. Bu bir nümunə dir belə xə tlə r daxil olmaqla hizalama bloku:

```
xal=19159.000000
s mm9.chr10 3014644 45 + 129993255 CCTGTACC---CTTTGGTGAGAATTTTTGTTTCAGTGTTAAAAG
s hg18.chr6 i 15870786 46 - 170899992 CCTATACCTTTCTTTTATGAGAA-TTTTGTTTTAATCCTAAAC-
hg18.chr6 s Mə n 9085 C 0
panTro2.chr6 q 16389355 46 - 173908612 CCTATACCTTTCTTTTATGAGAA-TTTTGTTTTAATCCTAAAC-
panTro2.chr6 və 99999999999999999999-999999999999999999-
panTro2.chr6 s I 9106 C 0
calJac1.Contig6394 i 6182 46 + 133105 CCTATACCTTTCTTTCATGAGAA-TTTTGTTTGAATCCTAAAC-
calJac1.Contig6394 s N 0 C 0
loxAfr1.scaffold_75561f. i 1167 34 - 10574 ----------TTTGGTTAGAA-TTATGCTTTAATTCAAAAC-
loxAfr1.scaffold_75566 e ----------99999699899-9999999999999869998-
tupBel1.scaffold_114895.1-498454 167376 N 0 C 0
4145 - e echTel1.scaffold_288249 87661 7564 + e otoGar1.33441. 181217 2931 - 498454 I
100002 I
359464 I
```

Machine Translated by Google

e ponAbe2.chr6 16161448 8044 - 174210431 İ

Bu, ucsc_mm9_chr10.maf faylındakı 10-cu uyğunlaşdırma blokudur (Testlə r/MAF alt kataloqunda mövcuddur) Biopython paylanması):

```
>>> Bio import Align- də n
>>> alignments = Align.parse("ucsc_mm9_chr10.maf", "maf")
>>> diapazondakı i üçün (10):
...         hizalama = sonrakı (düzlə şdirmə )
...
>>> alignment.score
19159.0
>>> çap (düzlə şdirmə )
mm9.chr10 3014644 CCTGTACC---CTTTGGTGAGAATTTTTGTTTCAGTGTTAAAAGTTG 3014689
hg18.chr6 155029206 CCTATACCTTTCTTTTATGAGAA-TTTTGTTTTAATCCTAAAC-TTTT 155029160
panTro2.c 157519257 CCTATACCTTTCTTTTATGAGAA-TTTTGTTTTAATCCTAAAC-TTTT 157519211
calJac1.C 6228 6182 CCTATACCTTTCTTTCATGAGAA-TTTTGTTTGAATCCTAAAC-TTTT
loxAfr1.s 9407 ----------TTGGTTAGAA-TTATGCTTTAATTCAAAAC-TTCC 9373
<BLANKLINE>
```

"i" sə tirlə ri cari uyğunlaşdırma blokundakı ardıcıllıqla içində kilə r arasındakı ə laqə ni göstə rir ə vvə lki və sonrakı hizalama bloku. Bu mə lumat annotasiya atributunda saxlanılır müvafiq ardıcıllıq:

```
>>> alignment.sequences[0].annotations
{}
>>> alignment.sequences[1].annotations
{'leftStatus': 'Mə n', 'leftCount': 9085, 'rightStatus': 'C', 'rightCount': 0}
```

bu blokla ə vvə lki blok arasında 9085 ə sas daxil edilmiş ("I") olduğunu göstə rir, blok isə sonrakı ilə bitişikdir ("C"). UCSC sə nə dlə rinə baxın bunların tam tə sviri üçün sahə lə r və status simvolları.

"q" sə tirlə ri annotasiyaların "keyfiyyə t" lüğə t açarı altında saxlanılan ardıcıl keyfiyyə ti göstə rir. müvafiq ardıcıllığın atributu:

```
>>> alignment.sequences[2].annotations["keyfiyyə t"]
'99999999999999999999999999999999999999999999'
>>> alignment.sequences[4].annotations["keyfiyyə t"]
'99999698999999999999999698699989997'
```

"e" sə tirlə ri bu düzülmə də n ə vvə l və sonra bitişik ardıcıllıqla növlə r haqqında mə lumatı göstə rir blok, lakin bu düzülmə blokunda heç bir hizalama nukleotidlə ri yoxdur. Bu "boş" düymə si altında saxlanılır alignment.annotations lüğə ti:

```
>>> alignment.annotations["empty"] # doctest: +NORMALIZE_WHITESPACE
[(SeqRecord(seq=Seq(Heç biri, uzunluq=498454), id='tupBel1.scaffold_114895.1-498454', ad='', tə svir='
  (SeqRecord(seq=Seq(Yox, uzunluq=100002), id='echTel1.scaffold_288249', name='', description='', dbxref (SeqRecord(seq=Seq(Heç biri,
  uzunluq=359464), id='otoGar1.scaffold_3.3944'.) ad='', tə svir='', (SeqRecord(seq=Seq(Heç biri, uzunluq=174210431), id='ponAbe2.chr6', ad='',
  tə svir='', dbxrefs=[]), (
```

Bu, mə sə lə n, 158040939 mövqeyində n ("I") daxil edilmiş düzlə nmə yə n ə sasların olduğunu göstə rir. 158048983 ponAbe2.chr6 genomik ardıcıllığının ə ks zolağında. Yenə də UCSC sə nə dinə baxın "e" xə tlə rinin tam tə rifi üçün.

MAF formatında düzülmə ni çap etmə k üçün Python-un daxili format funksiyasından istifadə edə bilə rsiniz:

Machine Translated by Google

```
>>> çap (format(düzlə şdirmə , "MAF"))
xal=19159.000000
s mm9.chr10                                                    3014644 45 + 129993255 CCTGTACC---CTTTGGTGAGAATTTTTGTTTCAGTGTTAAA
s hg18.chr6 i                                                  15870786 46 - 170899992 CCTATACCTTTCTTTTATGAGAA-TTTTGTTTTAATCCTAAA
hg18.chr6 s                                                    Mə n 9085 C 0
panTro2.chr6 q                                                 16389355 46 - 173908612 CCTATACCTTTCTTTTATGAGAA-TTTTGTTTTAATCCTAAA
panTro2.chr6 və                                                                          99999999999999999999-99999999999999999
panTro2.chr6 s                                                 I 9106 C 0
calJac1.Contig6394 i calJac1.Contig6394                        6182 46 +          133105 CCTATACCTTTCTTTCATGAGAA-TTTTGTTTGAATCCTAAA
s loxAfr1.scaffold_75561f. i                                   N 0 C 0
loxAfr1.scaffold_75566 e                                       1167 34 -          10574 ------------TTGGTTAGAA-TTATGCTTTAATTCAAAA
tupBel1.scaffold_114895.1-498454 167376                                                  ------------99999699899-999999999999986999
4145 - e echTel1.scaffold_288249 87661                         N 0 C 0
7564 + e otoGar1.scaffold_e3344. ponAbe2.chr6 <BLANKLINE>                    498454 I
                                                                              100002 I
                                                               181217 2931 -  359464 I
                                                               16161448 8044 - 174210431 İ

<BLANKLINE>
```

Tam MAF faylı yazmaq üçün Bio.Align.write(alignments, "myfilename.maf", fmt="maf") istifadə edin, burada myfilename.maf çıxış MAF faylının adıdır. Alternativ olaraq, çıxış üçün (mə tn) axını istifadə edə bilə rsiniz. Fayl başlığı mə lumatı alignments obyektinin metadata atributundan götürülə cə k. Ə gə r siz düzlə şdirmə lə ri sıfırdan yaradırsınızsa, siyahıya bə nzə r düzülmə lə r yaratmaq üçün Alignments (cə m) sinfində n istifadə edə bilə rsiniz. obyekt (bax. Bölmə 6.5) və ona metadata atributunu verin.

6.7.19 bigMaf

BigMaf faylı 3 tə lə b olunan BED sahə sində n və xüsusi bir yataqdan ibarə t BED3+1 formatlı bigBed faylıdır. MAF faylının indekslə nmiş ikili versiyasını yaradan, MAF uyğunlaşdırma blokunu sə tir kimi saxlayan sahə (bax: bölmə 6.7.19). Ə laqə dar AutoSql faylı bigMaf.as UCSC tə rə fində n tə min edilir. bigMaf faylı yaratmaq üçün edə bilə rsiniz ya UCSC-nin mafToBigMaf və bedToBigBed proqramlarından istifadə edin. və ya zə ng edə rə k Biopython istifadə edə bilə rsiniz fmt="bigmaf" ilə Bio.Align.write funksiyası. İki üsul eyni bigMaf ilə nə ticə lə nmə lidir UCSC alə tlə ri qızıl standart olduğundan daha sürə tli və daha etibarlı ola bilə r. bigMaf faylları kimi bigBed faylları, onlar müə yyə n bir istinad bölgə sində tez axtarış etmə yə imkan verə n daxili indekslə gə lir genom.

Biopython paylanmasının Tests/MAF alt kataloqunda ucsc_test.bb faylı bir nümunə dir. bigMaf faylı. Bu fayl ucsc_test.maf MAF faylına ekvivalentdir (bax bölmə 6.7.18). Bu faylı tə hlil etmə k üçün, istifadə edin

```
>>> Bio import Align- də n
>>> alignments = Align.parse("ucsc_test.bb", "bigmaf")
>>> lent (düzlə şdirmə )
3
>>> çap (alignments.declaration) # doctest: +NORMALIZE_WHITESPACE
masa çarpayıMaf
 "MAF bloklu çarpayı 3"
 (
     simli xrom; uint                    "İstinad ardıcıllığı xromosomu və ya iskele"
     chromStart; xromEnd; lstring        "Xromosomda başlanğıc mövqeyi"
     uint          mafBlock;             "Xromosomda son mövqe"
                                         "MAF bloku"
```

Machine Translated by Google

```
)
<BLANKLINE>
```

Bə yannamə də UCSC-də n bigMaf.as AutoSql faylı tə rə fində n müə yyə n edilmiş sütunların spesifikasiyası var.

bigMaf faylı MAF faylında olan başlıq mə lumatını saxlamır, lakin o, istinad tə yin edir genom. Müvafiq SeqRecord alignments obyektinin targets atributunda saxlanılır:

```
>>> alignments.reference 'hg16' >>>

alignments.targets # doctest: +ELLIPSIS [SeqRecord(seq=Seq(Heç biri,
 uzunluq=158545518), id='hg16.chr7', ...)]
```

Düzə lişlə ri tə krarlamaqla biz bigMaf faylında hə r bir hizalama bloku üçün bir Alignment obyekti ə ldə edirik:

```
>>> alignment = next(alignments) >>> alignment.score
23262.0 >>> {seq.id: len(seq) in

alignment.sequences } # doctest: +NORMALIZE_WHITESPACE {'hg16.chr7': 158545518,

   'panTro1.chr6': 161576975, 'baboon':
   4622798, 'mm4.chr6':
   151104725, 'rn3.chr4': 187371129}

>>> alignment.coordinates # doctest: +NORMALIZE_WHITESPACE massivi([[27578828,
27578829, 27578831, 27578831, 27578850, 27578850, 2757, 261]882 28741141, 28741143, 28741143, 28741162, 28741162,
                28741178], [ 116834, 116835, 116837, 116837, 156, 116 116872], [53215344, 53215344, 53215346, 53215347,
                53215366, 53215366, 53215382], [81344243, 813444343 81344245, 81344264, 81344267, 81344283]])


>>> çap (alignment) hg16.chr7
27578828 AAA-GGGAATGTTAACCAAATGA---ATTGTCTCTTACGGTG 27578866 panTro1.c 28741140 AAA-GGGAATGTTAACCAAATGA--
ATTGGTC18 ba 116834 AAA-GGGAATGTTAACCAAATGA---GTTGTCTCTTATGGTG 116872

mm4.chr6 53215344 -AATGGGAATGTTAAGCAAACGA---ATTGTCTCTCAGTGTG 53215382
rn3.chr4 81344243 -AA-GGGGATGCTAAGCCAATGAGTTGTTGTCTCTCAATGTG 81344283
<BLANKLINE>
>>> çap (format(alignment, "phylip")) 5 42

hg16.chr7 AAA-GGGAATGTTAACCAAATGA---ATTGTCTCTTACGGTG panTro1.chAAA-
GGGAATGTTAACCAAATGA---ATTGTCTCTTACGGTG AAA-GGGAATGTTAACCAAATGA---
                GTTGTCTCTTATGGTG. mmchboon -AATGGGAATGTTAAGCAAACGA---
ATTGTCTCTCAGTGTG rn3.chr4 -AA-GGGGATGCTAAGCCAATGAGTTGTTGTCTCTCAATGTG

<BLANKLINE>
```

“i”, “e” və “q” sə tirlə rində ki mə lumatlar müvafiq MAF faylında olduğu kimi saxlanılır (bax. bölmə 6.7.18):

```
>>> Bio importdan Align >>> alignments
= Align.parse("ucsc_mm9_chr10.bb", " bigmaf") >>> diapazondakı i üçün (10): hizalama =
sonrakı(düzlə şdirmə )
 ...
```

Machine Translated by Google

```
...
>>> alignment.score
19159.0
>>> çap (düzlə şdirmə )
mm9.chr10 3014644 CCTGTACC---CTTTGGTGAGAATTTTTGTTTCAGTGTTAAAAGTTG 3014689
hg18.chr6 155029206 CCTATACCTTTCTTTTATGAGAA-TTTTGTTTTAATCCTAAAC-TTTT 155029160
panTro2.c 157519257 CCTATACCTTTCTTTTATGAGAA-TTTTGTTTTAATCCTAAAC-TTTT 157519211
calJac1.C 6228        6182 CCTATACCTTTCTTTCATGAGAA-TTTTGTTTGAATCCTAAAC-TTTT
loxAfr1.s             9407 ------------TTGGTTAGAA-TTATGCTTTAATTCAAAAC-TTCC                    9373
<BLANKLINE>
>>> çap (format(düzlə şdirmə , "MAF"))
xal=19159.000000
s mm9.chr10                                  3014644 45 + 129993255 CCTGTACC---CTTTGGTGAGAATTTTTGTTTCAGTGTTAAA
s hg18.chr6 i                                15870786 46 - 170899992 CCTATACCTTTCTTTTATGAGAA-TTTTGTTTTAATCCTAAA
hg18.chr6 s                                  Mə n 9085 C 0
panTro2.chr6 q                               16389355 46 - 173908612 CCTATACCTTTCTTTTATGAGAA-TTTTGTTTTAATCCTAAA
panTro2.chr6 və                                                      99999999999999999999-99999999999999999
panTro2.chr6 s                               I 9106 C 0
calJac1.Contig6394 i calJac1.Contig6394          6182 46 +          133105 CCTATACCTTTCTTTCATGAGAA-TTTTGTTTGAATCCTAAA
s loxAfr1.scaffold_75561f. i                 N 0 C 0
loxAfr1.scaffold_75566 e                         1167 34 -           10574 ------------TTGGTTAGAA-TTATGCTTTAATTCAAAA
tupBel1.scaffold_114895.1-498454 167376                                      ------------99999699899-999999999999986999
4145 - e echTel1.scaffold_288249 87661       N 0 C 0
7564 + e otoGar1.scaffold_e3344. ponAbe2.chr6 <BLANKLINE>   498454 I
                                             100002 I
                                                 181217 2931 -   359464 I
                                             16161448 8044 - 174210431 İ

<BLANKLINE>
>>> alignment.sequences[1].annotations
{'leftStatus': 'Mə n', 'leftCount': 9085, 'rightStatus': 'C', 'rightCount': 0}
>>> alignment.sequences[2].annotations["keyfiyyə t"]
'99999999999999999999999999999999999999999'
>>> alignment.sequences[4].annotations["keyfiyyə t"]
'99999699899999999999999998699989997'
>>> alignment.annotations["empty"] # doctest: +NORMALIZE_WHITESPACE
[(SeqRecord(seq=Seq(Heç biri, uzunluq=498454), id='tupBel1.scaffold_114895.1-498454', ad='', tə svir='
  (SeqRecord(seq=Seq(Yox, uzunluq=100002), id='echTel1.scaffold_288249', name='', description='', dbxref (SeqRecord(seq=Seq(Heç biri,
  uzunluq=359464), id='otoGar1.scaffold_3.3944'.) ad='', tə svir='', (SeqRecord(seq=Seq(Heç biri, uzunluq=174210431), id='ponAbe2.chr6', ad='',
  tə svir='', dbxrefs=[]), (
```

Tam bigMaf faylı yazmaq üçün Bio.Align.write(alignments, "myfilename.bb", fmt="bigMaf") istifadə edin. burada myfilename.bb çıxış bigMaf faylının adıdır. Alternativ olaraq, üçün (ikili) axın istifadə edə bilə rsiniz çıxış. Ə gə r düzlə şdirmə lə ri sıfırdan yaradırsınızsa, yaratmaq üçün Alignments (cə m) sinfində n istifadə edə bilə rsiniz siyahıya bə nzə r düzlə şdirmə obyekti (bax. Bölmə 6.5) və ona hə də flə r atributunu verin. Sonuncu siyahı olmalıdır Referans növlə r üçün xromosomlar üçün SeqRecord obyektlə ri göründüklə ri ardıcıllıqla hizalamalar. Alternativ olaraq, Bio.Align.write-ə zə ng edə rkə n hə də flə r açar sözü arqumentində n istifadə edə bilə rsiniz. The hə r SeqRecord-un id-si istinad istinada istinad etdiyi referans.xromosom şə klində olmalıdır növlə r. Bio.Align.write ə lavə açar söz arqumentini sıxışdırır (standart olaraq True) mə lumatların zlib istifadə edə rə k sıxılmalı olub-olmaması.

bigMaf faylı bigBed faylının xüsusi halı olduğundan, hizalamalarda axtarış metodundan istifadə edə bilə rsiniz. istinad növlə rinin xüsusi bölgə lə rinə uyğunlaşma tapmaq üçün obyekt. Mə sə lə n, biz bölgə lə ri axtara bilə rik

Machine Translated by Google

chr10-da 10-cu xromosomda 3018000 və 3019000 mövqelə ri arasında:

```
>>> seçilmiş_alignments = alignments.search("mm9.chr10", 3018000, 3019000) >>>
seçilmiş_alignmentlə rdə düzülmə üçün :
...         start, end = alignment.coordinates[0, 0], alignment.coordinates[0, -1] print(start, end)
...
...
3017743 3018161
3018161 3018230
3018230 3018359
3018359 3018482
3018482 3018644
3018644 3018822
3018822 3018932
3018932 3019271
```

Xromosom adı bütün xromosomları ehtiva etmə k üçün None ola bilə r və başlanğıc və son mövqelə r müvafiq olaraq 0 mövqeyində n axtarışa başlamaq və ya xromosomun sonuna qə də r axtarışa davam etmə k üçün None ola bilə r. Qeyd edə k ki, biz yalnız istinad növlə ri üçün genomik mövqedə axtarış edə bilə rik.

6.7.20 UCSC zə nciri fayl formatı

Zə ncir faylları iki nukleotid ardıcıllığı arasında cüt uyğunlaşmanı tə svir edir və hə r iki ardıcıllıqda boşluqlara imkan verir. Yalnız hə r bir düzülmüş alt ardıcıllığın uzunluğu və boşluq uzunluqları bir zə ncir faylında saxlanılır; ardıcıllıqların özlə ri saxlanmır. Zə ncir faylları adə tə n iki genom montaj versiyası arasında uyğunlaşmaları saxlamaq üçün istifadə olunur ki, bu da bir genom montaj versiyasına uyğunlaşmaların digə r genom birlə şmə sinə qaldırılmasına imkan verir. Bu, zə ncir faylının nümunə sidir (Biopython paylanmasının Tests/Blat alt kataloqunda psl_34_001.chain kimi mövcuddur):

```
zə ncir 16 chr4 191154276 + 61646095 61646111 hg18_dna 33 + 11 27 1 16

zə ncir 33 chr1 249250621 + 10271783 10271816 hg18_dna 33 + 0 33 2 33

zə ncir 17 chr2 243199373 + 53575980 53575997 hg18_dna 33 - 8 25 3 17

zə ncir 35 chr9 141213431 + 85737865 85737906 hg19_dna 50 + 9 50 4 41

zə ncir 41 chr8 146364022 + 95160479 95160520 hg19_dna 50 + 8 49 5 41

zə ncir 30 chr22 51304566 + 42144400 42144436 hg19_dna 50 + 11 47 6 36

zə ncir 41 chr2 243199373 + 183925984 183926028 hg19_dna 50 + 1 49 7 6 4 38 zə ncir 31 chr19 59128983 + 35483340
510d + 310g 46 8 25 0
0

134
11
zə ncir 39 chr18 78077248 + 23891310 23891349 hg19_dna 50 + 10 49 9 39

...
```

Bu fayl psl_34_001.psl PSL faylında UCSC-nin pslToChain proqramını işlə tmə klə yaradılıb. Zə ncirvari fayl formatının spesifikasiyasına görə , hə r bir zə ncir blokundan sonra boş bir xə tt olmalıdır, lakin bə zi alə tlə r (pslToChain daxil olmaqla) yə qin ki, bu qaydaya ə mə l etmir.

Machine Translated by Google

Bu faylı tə hlil etmə k üçün istifadə edin

```
>>> Bio import Align- də n
>>> alignments = Align.parse("psl_34_001.chain", "zə ncir")
```

Hə r zə ncir üçün bir Alignment obyekti ə ldə etmə k üçün hizalamaları tə krarlayın:

```
>>> düzülmə lə rdə düzülmə üçün :
...         çap (alignment.target.id, alignment.query.id) # doctest: +ELLIPSIS
...
chr4 hg18_dna
chr1 hg18_dna
chr2 hg18_dna
chr9 hg19_dna
chr8 hg19_dna
chr22 hg19_dna
chr2 hg19_dna
...
chr1 hg19_dna
```

Düzə lişlə ri geri sarın və yeddinci düzülmə yə çatana qə də r ə vvə ldə n tə krarlayın:

```
>>> alignments.rewind()
>>> diapazondakı i üçün (7):
...         hizalama = sonrakı (düzlə şdirmə )
...
```

Düzə liş xalını və zə ncir identifikatorunu yoxlayın (hə r birinin başlıq sə tirində müvafiq olaraq birinci və sonuncu rə qə m zə ncir bloku) yeddinci düzülmə ni ə ldə etdiyimizi tə sdiqlə mə k üçün:

```
>>> alignment.score
41.0
>>> alignment.annotations["id"]
'7'
```

Düzə ltmə ni zə ncir fayl formatında çap edə bilə rik. Hizalanma koordinatları ilə uyğundur
6 nukleotiddə n ibarə t düzülmüş bölmə , 4 nukleotidlik boşluq və zə ncir blokunda mə lumat
38 nukleotidin uyğunlaşdırılmış bölmə si:

```
>>> çap (format(düzlə şdirmə , "zə ncir")) # doctest: +NORMALIZE_WHITESPACE
zə ncir 41 chr2 243199373 + 183925984 183926028 hg19_dna 50 + 1 49 7
6           0           4
38
<BLANKLINE>
<BLANKLINE>
>>> hizalama.koordinatlar
massiv([[183925984, 183925990, 183925990, 183926028],
        [          1,          7,          11, 49]])
>>> çap (düzlə şdirmə )
chr2 183925984 ??????----????????? 183926028
                    0 |||||----||||||
```

Machine Translated by Google

```
>>> çap (format(düzlə şdirmə , "YATAK")) # doctest: +NORMALIZE_WHITESPACE
chr2 183925984 hg19_dna 183926028 41 + 183925984 18392
<BLANKLINE>
>>> çap (format(düzlə şdirmə , "PSL")) # doctest: +NORMALIZE_WHITESPACE
44 0 0 0 1 4 0 0 + hg19_dna 50
<BLANKLINE>
>>> çap (format (düzlə şdirmə , "tə mizlə mə "))
vulqar: hg19_dna 1 49 + chr2 183925984 183926028 + 41 M 6 6 G 4 0 M 38 38
<BLANKLINE>
>>> çap (alignment.format("exonerate", "siqar"))
siqar: hg19_dna 1 49 + chr2 183925984 183926028 + 41 M 6 I 4 M 38
<BLANKLINE>
>>> çap (format(alignment, "sam")) # doctest: +NORMALIZE_WHITESPACE
hg19_dna 0 chr2 183925985 255 1S6M4I38M1S * 0 0
<BLANKLINE>
```

Machine Translated by Google

Fə sil 7

Cüt ardıcıllıqla düzülmə

Cüt ardıcıllıqla düzülmə iki ardıcıllığın aralarındakı oxşarlıq xalını optimallaşdırmaqla bir-birinə uyğunlaşdırma prosesidir. Bio.Align modulu Needleman-Wunsch, Smith-Waterman, Gotoh (üç ştatlı) və Waterman-Smith-Beyer qlobal və yerli ikili düzülmə alqoritmlə rində n istifadə etmə klə qlobal və yerli hizalamalar üçün PairwiseAligner sinfini ehtiva edir və hizalanma parametrlə rini də yişdirmə k üçün çoxsaylı seçimlə r var.
Biz Durbin və başqalarına istinad edirik. [11] ardıcıllıqla düzülmə alqoritmlə ri haqqında ə traflı mə lumat üçün.

7.1 Ə sas istifadə

Cüt şə kildə düzülmə yaratmaq üçün ə vvə lcə PairwiseAligner obyekti yaradın:

```
>>> Bio idxaldan Align >>> aligner
= Align.PairwiseAligner ()
```

PairwiseAligner obyekt düzlə ndiricisi (bax. Bölmə 7.2) qoşa düzülmə lə r üçün istifadə olunacaq hizalanma parametrlə rini saxlayır. Bu atributlar obyektin konstruktorunda tə yin edilə bilə r:

```
>>> hizalayıcı = Align.PairwiseAligner(uyğun_score=1.0)
```

və ya obyekt edildikdə n sonra:

```
>>> aligner.match_score = 1.0
```

İki ardıcıllıq arasında düzülmə xalını hesablamaq üçün aligner.score metodundan istifadə edin:

```
>>> hə də f = "GAACT" >>> sorğu =
"GAT" >>> xal = aligner.score(hə də f,
sorğu) >>> xal
```

```
3.0
```

aligner.align metodu hə r biri iki ardıcıllıq arasında bir düzülmə ni tə msil edə n Alignment obyektlə rini qaytarır:

```
>>> alignments = aligner.align(hə də f, sorğu) >>> alignment =
alignments[0] >>> alignment # doctest:
+ELLIPSIS <Hazırlama obyekti (2 sıra x 5 sütun) ...>
```

Alignment obyektlə ri üzə rində tə krarlayın və düzülmə lə ri görmə k üçün onları çap edin:

Machine Translated by Google

```
>>> hizalamalarda hizalama üçün : çap (alignment)
 ...
 ...
hə də f                    0 GAACT 5
                           0 ||--| 5
sorğu                      0 GA--T 3
<BLANKLINE>
hə də f                    0 GAACT 5
                           0 |-|-| 5
sorğu                      0 GAT 3
<BLANKLINE>
```

Hə r düzülmə uyğunlaşma xalını saxlayır:

```
>>> alignment.score 3.0
```

elə cə də düzülmüş ardıcıllığa işarə lə r:

```
>>> alignment.target 'GAACT' >>>

alignment.query 'GAT'
```

Daxili olaraq, düzülmə ardıcıllıq koordinatları baxımından saxlanılır:

```
>>> alignment = alignments[0] >>>
alignment.coordinates array([[0, 2, 4, 5],
[0, 2, 2, 3]])
```

Burada iki sıra hə də f və sorğu ardıcıllığına istinad edir. Bu koordinatlar hizalanma olduğunu göstə rir aşağıdakı üç blokdan ibarə tdir:

- hə də f[0:2] sorğuya uyğunlaşdırılmışdır[0:2];
- target[2:4] boşluğa uyğunlaşdırılıb, çünki sorğu[2:2] boş sə tirdir;
- hə də f[4:5] sorğuya uyğunlaşdırılmışdır[2:3].

Cütlü düzülmə üçün düzülmüş ardıcıllıqların sayı hə mişə 2-dir:

```
>>> lent (düzlə şdirmə ) 2
```

Hizalanma uzunluğu çap edilmiş hizalanmadakı sütunların sayı kimi müə yyə n edilir. Bu, hə də f və sorğuda uyğunluqların sayının, uyğunsuzluqların sayının və boşluqların ümumi uzunluğunun cə minə bə rabə rdir:

```
>>> hizalama.uzunluq 5
```

Düzə liş edilmiş alt ardıcıllıqların başlanğıc və son indekslə rini qaytaran aligned xassə , ilk düzülmə üçün uzunluğu 2 olan iki də sti qaytarır:

Machine Translated by Google

```
>>> alignment.aligned massiv([[[0, 2],
[4, 5]], <BLANKLINE>


                [[0, 2], [2,
                  3]]])
```

alternativ düzülmə üçün isə uzunluğu 3 olan iki də st qaytarılır:

```
>>> alignment = hizalamalar[1] >>> çap
(alignment) hə də fi
                                0 GAACT 5
                                0 |-|-| 5
sorğu                           0 GAT 3
<BLANKLINE>
>>> alignment.aligned massiv([[[0,
1], [2, 3], [4, 5]],


<BLANKLINE>
                [[0, 1], [1,
                  2], [2, 3]]])
```

Qeyd edə k ki, müxtə lif düzülmə lə r bir-birinə uyğunlaşdırılmış eyni alt ardıcıllığa malik ola bilə r. Xüsusilə , bu, düzülmə lə r yalnız boşluqların yerlə şdirilmə si baxımından bir-birində n fə rqlə ndikdə baş verə bilə r:

```
>>> aligner.mode = "qlobal" >>>
aligner.mismatch_score = -10 >>> alignments =
aligner.align("AAACAAA", "AAAGAAA") >>> len(alignments) 2 >>> print(alignments[0])



0 |||--|||                       0 AAAC-AAA 7 hə də f
          8 0 AAA-GAAA 7
sorğu
<BLANKLINE>
>>> alignments[0].aligned massiv([[[0, 3], [4,
7]], <BLANKLINE>


                [[0, 3], [4,
                  7]]]) >>>
çap(alignments[1]) hə də f
                                0 AAA-CAAA 7
                                0 |||--||| 8
sorğu                           0 AAAG-AAA 7
<BLANKLINE>
>>> alignments[1].aligned massiv([[[0, 3], [4,
7]], <BLANKLINE> [[0, 3],
                  [4, 7]]])
```

Machine Translated by Google

Xə ritə metodu düzlə şdirmə 2 sorğusunun düzlə şdirmə 1 hə də finə qoşa düzülmə sini tapmaq üçün ikili düzülmə də tə tbiq oluna bilə r, burada düzlə şdirmə 2 və düzlə şdirmə 1 sorğusu eynidir. Tipik bir misal, düzülmə 1-in xromosom və transkript arasında qoşa düzülmə olduğu, düzülmə 2-nin transkript və ardıcıllıq (mə sə lə n, oxunan RNT-seq) arasında ikili düzülmə olduğu və biz ardıcıllığın xromosoma uyğunluğunu tapmaq istə yirik:

```
>>> aligner.mode = "yerli" >>>
aligner.open_gap_score = -1 >>> aligner.extend_gap_score
= 0 >>> xromosom =
"AAAAAAAACCCCCCCAAAAAAAAAGGGGGGAAAAAAAAA" >>> transkript = "CCCCCCGAG
alignner.open_gap_score = (" CCCCCCGAG aligner.>
transkript) >>> len(alignments1) 1 >>> alignment1 = alignments1[0] >>> print(alignment1) target


                                8 CCCCCAAAAAAAAAAAGGGGGG 32
                                0 |||||||----------|||||| 24
sorğu                           0 CCCCCCC----------GGGGGG 13
<BLANKLINE>
>>> sequence = "CCCCGGGG" >>> alignments2
= aligner.align(transkript, ardıcıllıq) >>> len(alignments2) 1


>>> alignment2 = alignments2[0] >>> print(alignment2)
hə də f
                                3 CCCCGGGG 11
                                0 |||||||| 8
sorğu                           0 CCCCGGGG 8
<BLANKLINE>
>>> mapped_alignment = alignment1.map(alignment2) >>> print(mapped_alignment)
hə də fi
                               11 CCCCAAAAAAAAAAAGGGG 30 0
                                  ||||----------|||| 19
sorğu                           0 CCCC----------GGGG 8
<BLANKLINE> >>>
format(mapped_alignment, "psl") '8\t0\t0\t0\t0\t1\t11\t+
 \tquery\t8\t0\t8\ttarget\t40\t11\t30\t2\t4,4,\t0,2,\t11
```

Hizalanmanın xə ritə lə şdirilmə si ardıcıllığın mə zmunundan asılı deyil. Ardıcıllığın mə zmununu silsə k, eyni düzülmə PSL formatında tapıla bilə r (baxmayaraq ki, ardıcıllığı çap etmə k qabiliyyə tini itiririk):

```
>>> Bio.Seq import Seq >>> alignment1.target =
Seq(None, len(alignment1.target)) >>> alignment1.query = Seq (None, len(alignment1.query)) >>>
alignment2.target = Seq (None, len (alignment2.target2)) >>> ry len(alignment2.query)) >>>
mapped_alignment = alignment1.map(alignment2) >>> format(mapped_alignment, "psl")
'8\t0\t0\t0\t0\t1\t11\t+\tquery\t8\t0\t8\ttarget\t40\t11\t30\t2\t4,4,\t0,4,\t11,26,\n'
```

Varsayılan olaraq, bütövlükdə optimal uyğunlaşmanı tapan qlobal cütlü hizalama hə yata keçirilir

Machine Translated by Google

hə də f və sorğunun uzunluğu. Bunun ə və zinə , yerli düzülmə hə də f və sorğunun ə n yüksə k uyğunlaşma balı ilə ardıcıllığını tapacaq. Yerli hizalamalar aligner.mode-u "yerli" olaraq tə yin etmə klə yaradıla bilə r:

```
>>> aligner.mode = "yerli" >>> hə də f = "AGAACTC"
>>> sorğu = "GAACT" >>> xal = aligner.score
(hə də f, sorğu) >>> xal


5.0
>>> alignments = aligner.align(hə də f, sorğu) >>> hizalamalarda düzülmə
üçün : çap (alignment)
 ...
 ...
hə də f                    1 GAACT 6
                           0 | | | | | 5
sorğu                      0 GAACT 5
<BLANKLINE>
```

Nə zə rə alın ki, ə n yaxşı yerli düzülmə lə rin tə rifində müə yyə n qeyri-müə yyə nlik var, ə gə r 0 ballı seqmentlə r düzülmə yə ə lavə edilə bilə r. Biz Waterman & Eggert-in [50] tə klifinə ə mə l edirik və buna icazə vermirik uzantılar.

7.2 Cütlü düzlə şdirici obyekt

PairwiseAligner obyekti qoşa düzülmə lə r üçün istifadə edilə cə k bütün hizalama parametrlə rini saxlayır. Bütün parametrlə r üçün də yə rlə rin icmalına baxmaq üçün istifadə edin

```
>>> Bio importdan Align >>> aligner =
Align.PairwiseAligner(match_score=1.0, mode ="local") >>> print(aligner)

Parametrlə ri ilə cüt ardıcıl düzlə ndirici
    Joker: Yoxdur
    match_score: 1.000000
    uyğunsuzluq_score: 0.000000
    target_daxili_açıq_boşluq_score: 0.000000 target_daxili_uzatma_boşluğu:
    0.000000 target_left_open_bop_score: 0.000ftsend 0.000000
    target_right_open_gap_score: 0.000000 target_right_extend_gap_score:
    0.000000 query_internal_open_boşluq_score: 0.000000
    query_daxili_uzatma_boşluğu_score: 0.0000000
    query_daxili_uzatma_boşluğu: 0.0000000_query 0,000000
    query_left_extend_boşluq_score: 0,000000 query_right_open_gap_score:
    0,000000 query_right_extend_boşluq_score: 0,000000 rejim: yerli




<BLANKLINE>
```

Bu parametrlə rin tə rifi üçün aşağıdakı Bölmə 7.3, 7.4 və 7.5- ə baxın . Atribut rejimi (yuxarıda Bölmə 7.1-də tə svir edilmişdir) müvafiq olaraq qlobal və ya yerli cütlü düzülüşü müə yyə n etmə k üçün "qlobal" və ya "yerli"yə bə rabə r tə yin edilə bilə r.

Machine Translated by Google

Boşluğun hesablanması parametrlə rində n (7.4 və 7.5-ci bölmə lə rə baxın) və rejimdə n asılı olaraq, PairwiseAligner obyekti avtomatik olaraq cüt ardıcıllıqla düzülmə üçün istifadə etmə k üçün müvafiq alqoritmi seçir. Seçilmiş alqoritmi yoxlamaq üçün istifadə edin

```
>>> aligner.alqoritmi 'Smith-Waterman'
```

Bu atribut yalnız oxumaq üçündür.

PairwiseAligner obyekti eyni zamanda hizalama zamanı istifadə edilə cə k ϵ də qiqliyini saxlayır. ϵ də yə ri aligner.epsilon atributunda saxlanılır və defolt olaraq 10 6- ya bə rabə rdir :

```
>>> aligner.epsilon 1e-06
```

Aralarındakı mütlə q fə rq ϵ -də n az olarsa, düzülmə mə qsə di ilə iki xal bir-birinə bə rabə r hesab edilə cə k.

7.3 Ə və zetmə balları

Ə və zetmə balları iki hə rf (nukleotidlə r və ya amin turşuları) bir-birinə düzüldükdə ümumi bala ə lavə olunacaq də yə ri müə yyə n edir. PairwiseAligner tə rə fində n istifadə edilə cə k ə və zetmə balları iki yolla müə yyə n edilə bilə r:

- Eyni hə rflə r üçün uyğunluq xalını və uyğun gə lmə yə n hə rflə r üçün uyğunsuzluq xallarını tə yin etmə klə . Nu-kleotid ardıcıllığının uyğunlaşdırılması adə tə n uyğunluq və uyğunsuzluq xallarına ə saslanır. Mə sə lə n, defolt olaraq BLAST [1] 2.5 boşluq cə zası ilə meqablast tə rə fində n nukleotid düzülüşü üçün +1 uyğunluq hesabından və 2 uyğunsuzluq hesabından istifadə edir (boşluq xalları haqqında ə lavə mə lumat üçün bölmə 7.4-ə baxın). Uyğunluq və uyğunsuzluq xalları PairwiseAligner obyektinin uyğunluq və uyğunsuzluq atributlarını tə yin etmə klə müə yyə n edilə bilə r:

```
>>> Bio import Align >>> aligner =
Align.PairwiseAligner() >>> aligner.match_score 1.0 >>>
aligner.mismatch_score 0.0 >>> ball =

aligner.score ("ACGT", "ACAT") >>> print(score)
3.0
>>> aligner.match1score = _aligner.match_score = _aligner.match_0 -2.0
>>> aligner.gap_score = -2.5
>>>
xal = aligner.score("ACGT", "ACAT") >>> print(bal) 1.0
```

Uyğunluq və uyğunsuzluq xallarından istifadə edə rkə n namə lum hə rflə r üçün joker simvol (standart olaraq heç biri) tə yin edə bilə rsiniz. Uyğunluğun də yə rində n və ya uyğunsuzluq hesabından asılı olmayaraq, bunlar düzülmə lə rdə sıfır xal qazanacaq:

```
>>> aligner.wildcard = "?" >>> xal =
aligner.score("ACGT", "AC?T") >>> çap (hesab) 3.0
```

Machine Translated by Google

- Alternativ olaraq, ə və zetmə matrisini tə yin etmə k üçün PairwiseAligner obyektinin substitution_matrix atributundan istifadə edə bilə rsiniz. Bu, müxtə lif uyğun gə lə n və uyğun gə lmə yə n hə rflə r üçün müxtə lif ballar tə tbiq etmə yə imkan verir. Bu, adə tə n, amin turşusu ardıcıllığının uyğunlaşdırılması üçün istifadə olunur. Mə sə lə n, defolt olaraq BLAST [1] blastp ilə zülal uyğunlaşdırılması üçün BLOSUM62 ə və zetmə matrisində n istifadə edir. Bu ə və zetmə matrisi Biopython-dan ə ldə edilə bilə r:

```
>>> Bio.Align import ə və zedici_matrislə rində n >>>
substitution_matrices.load () # doctest: +ELLIPSIS ['BENNER22', 'BENNER6',
 'BENNER74', 'BLASTN', 'BLASTP', 'BLOSUM45', 'BLOSUM50', ', >>2, BLOSUM50', ' >> 2 tri substitution_matrices.load("BLOSUM62") >>> çap(matris) #
doctest: +ELLIPSIS # Matblas tə rə fində n blosum62.iij saytından hazırlanmış
matris

 ...
        A       R       N       D       C       Q...
A 4,0 -1,0 -2,0 -2,0 0,0 -1,0 ...
R -1,0 5,0 0,0 -2,0 -3,0 1,0 ...
N -2,0 0,0 6,0 1,0 -3,0 0,0 ...
D -2,0 -2,0 1,0 6,0 -3,0 0,0 ...
C 0,0 -3,0 -3,0 -3,0 9,0 -3,0 ...
Q -1,0 1,0 0,0 0,0 -3,0 5,0 ...
 ...
>>> aligner.substitution_matrix = matris >>> xal =
aligner.score("ACDQ", "ACDQ")
>>> xal
24.0
>>> xal = aligner.score("ACDQ", "ACNQ")
>>> xal
19.0
```

Ə və zedici matrisadan istifadə edə rkə n X namə lum simvol kimi şə rh edilmir. Ə və zində ə və zetmə matrisi tə rə fində n verilə n xal istifadə olunacaq:

```
>>> matrisi["D", "X"] -1.0 >>> xal =

aligner.score("ACDQ", "ACXQ") >>> xal

17.0
```

Defolt olaraq, aligner.substitution_matrix Yoxdur. aligner.match_score və aligner.mismatch_scor atributları aligner.substitution_matrix heç biri deyilsə nə zə rə alınmır. aligner.match_score və ya aligner.mismatch_score-u etibarlı də yə rlə rə tə yin etmə k aligner.substitution_matrix-i Heç biri kimi sıfırlayacaq.

7.4 Affin boşluğu balları

Affin boşluq xalları boşluğu açmaq üçün xal və mövcud boşluğu genişlə ndirmə k üçün hesabla müə yyə n edilir: boşluq balı = açıq boşluq balı + (n - 1) × boşluq balı, burada n boşluğun uzunluğudur. Biopython-un cüt ardıcıllıqla düzlə ndiricisi üzə rində incə də nə li idarə etmə yə imkan verir PairwiseAligner obyektinin aşağıdakı on iki atributunu tə yin etmə klə boşluq hesablama sxemi:

Bu atributlar daxili boşluqlar üçün və ardıcıllığın hə r iki ucunda müxtə lif boşluq xallarına imkan verir bu nümunə də göstə rilir:

Rahatlıq üçün PairwiseAligner obyektlə ri Cə dvə l 7.1- də göstə rildiyi kimi (ierarxik olaraq) bu də yə rlə rin bir sırasına toplu olaraq istinad edə n ə lavə atributlara malikdir .

Machine Translated by Google

Açılış xalları	Xalların uzadılması
sorğu_sol_açıq_boşluq_xalı sorğusu_sol_boşluğu_uzatmaq	
sorğu_daxili_açıq_boşluq_xalı sorğusu_daxili_uzatma_boşluğu_balı	
query_sağ_açıq_boşluq_xalı sorğusu_sağ_uzatmaq_boşluq_balı	
target_left_open_boşluq_score target_left_uzatmaq_boşluq_score	
target_daxili_açıq_boşluq_score target_daxili_uzatma_boşluğu_balı	
target_sağ_açıq_boşluq_score target_right_uzatmaq_boşluq_score	

hə də f sorğu balı		
A	-	sorğu sol açıq boşluq hesabı
C	-	sorğu sola boşluq hesabını uzat
C	-	sorğu sola boşluq hesabını uzat
G	G matç hesabı	
G	T uyğunsuzluğu hesabı	
G	-	sorğu daxili açıq boşluq hesab
A	-	sorğu daxili uzadılmış boşluq hesabı
A	-	sorğu daxili uzadılmış boşluq hesabı
T	T matç hesabı	
A	Bir matç hesabı	
G	-	sorğu daxili açıq boşluq hesab
C	C matçı hesabı	
-	C hə də f daxili açıq boşluq hesabı	
-	C hə də f daxili uzadılmış boşluq hesabı	
C	C matçı hesabı	
T	G uyğunsuzluğu hesabı	
C	C matçı hesabı	
-	C hə də f daxili açıq boşluq hesabı	
A	Bir matç hesabı	
-	T hə də f sağ açıq boşluq hesab	
-	Hə də f sağdakı boşluq hesabını genişlə ndirir	
-	Hə də f sağdakı boşluq hesabını genişlə ndirir	

7.5 Ümumi boşluq balları

Boşluq balları üzə rində daha də qiq nə zarə t üçün, siz boşluq hesablama funksiyasını tə yin edə bilə rsiniz. Mə sə lə n, Aşağıdakı boşluq hesablama funksiyası sorğu ardıcıllığında iki nukleotiddə n sonra boşluğa icazə vermir:

```
>>> Bio import Align- də n
>>> düzlə şdirici = Align.PairwiseAligner()
>>> def my_boşluq_score_funksiyam (başlanğıc, uzunluq):
...         ə gə r başlanğıc == 2:
...             geri - 1000
...         başqa:
...             qayıtmaq -1 * uzunluq
...
>>> aligner.query_gap_score = my_boşluq_score_funksiyam
>>> hizalamalar = aligner.align("AACTT", "AATT")
>>> düzülmə lə rdə düzülmə üçün :
...         çap (düzlə şdirmə )
...
hə də f                 0 AACTT 5
```

Machine Translated by Google

Cə dvə l 7.1: PairwiseAligner obyektlə rinin metaatributları.

Meta atribut	Xə ritə lə ri atributları
boşluq_xalı	hə də f_boşluq_balı, sorğu_boşluğu_balı
açıq_boşluq_xalı	hə də f_açıq_boşluq_balı, sorğu_açıq_boşluq_balı
uzadılması_boşluq_xalı	hə də f_uzatma_boşluğu_balı, sorğu_uzatmaq_boşluq_balı
daxili_boşluq_xalı	hə də f_daxili_boşluq_balı, sorğu_daxili_boşluq_balı
daxili_açıq_boşluq_xalı	hə də f_daxili_açıq_boşluq_balı, sorğu_daxili_açıq_boşluq_balı
sağ_uzatma_boşluğu_hesabı_açıq_boşluq_hesabı	hə də f_daxili_uzatma_boşluğu_balı, sorğu_daxili_uzatma_boşluğu_balı
	hə də f_end_boşluq_balı, sorğu_sonu_boşluq_balı
	hə də f_end_açıq_boşluq_xalı, sorğu_sonu_açıq_boşluq_balı
	hə də f_end_uzatma_boşluğu_xalı, sorğu_sonu_uzatmaq_boşluq_balı
	hə də f_sol_boşluq_balı, sorğu_sol_boşluq_balı
	hə də f_sağ_boşluq_balı, sorğu_sağ_boşluq_balı
	hə də f_sol_açıq_boşluq_xalı, sorğu_sol_açıq_boşluq_xalı
	hə də f_sol_uzatma_boşluğu_xalı, sorğu_sol_uzatmaq_boşluq_balı
	hə də f_sağ_açıq_boşluq_balı, sorğu_sağ_açıq_boşluq_balı
	hə də f_sağ_uzatma_boşluq_balı, sorğu_sağ_uzatmaq_boşluq_balı
	hə də f_daxili_açıq_boşluq_balı, hə də f_sol_açıq_boşluq_xalı,
	hə də f_sağ_açıq_boşluq_balı
hə də f_uzatmaq_boşluq_balı	hə də f_daxili_uzatma_boşluğu_balı, hə də f_sol_uzatmaq_boşluq_balı,
	hə də f_sağ_uzatmaq_boşluq_balı
hə də f_boşluğu_hesabı	hə də f_açıq_boşluq_balı, hə də f_uzatmaq_boşluq_balı
sorğusu_açıq_boşluq_hesabı	sorğu_daxili_açıq_boşluq_balı, sorğu_solu_açıq_boşluq_xalı,
	sorğu_sağ_açıq_boşluq_balı
sorğu_uzatmaq_boşluq_balı	sorğu_daxili_uzatma_boşluğu_balı, sorğu_sol_uzatmaq_boşluq_balı,
	sorğu_sağ_uzatmaq_boşluq_balı
sorğu_boşluğu_score	sorğu_açıq_boşluq_xalı, sorğu_uzatmaq_boşluğu_xalı
target_daxili_boşluq_score	hə də f_daxili_açıq_boşluq_balı, hə də f_daxili_uzatma_boşluğu_balı
target_end_boşluq_score	hə də f_end_açıq_boşluq_balı, hə də f_end_uzatmaq_boşluq_balı
target_end_open_bour_score	hə də f_sol_açıq_boşluq_xalı, hə də f_sağ_açıq_boşluq_balı
target_end_end_uzatmaq_boşluq_score	target_left_uzatmaq_boşluq_score, target_sağ_uzatmaq_boşluq_score
sol_boşluq_balı	hə də f_sol_açıq_boşluq_balı, sol_boşluğu_uzatmaq_balı
hə də f_sağ_boşluq_balı	hə də f_sağ_açıq_boşluq_balı, hə də f_sağ_boşluğu_uzatmaq_balı
sorğu_end_boşluğu_xalı	sorğu_sonu_açıq_boşluq_xalı, sorğu_sonu_boşluq_xalı
sorğu_sonu_açıq_boşluq_xalı	sorğusu_sol_açıq_boşluq_xalı, sorğu_sağ_açıq_boşluq_xalı
query_end_uzatmaq_boşluq_xalı	sorğu_sol_uzatmaq_boşluq_balı, sorğu_sağ_uzatmaq_boşluq_balı
sorğu_daxili_boşluq_xalı	sorğu_daxili_açıq_boşluq_balı, sorğu_daxili_boşluq_balı
sorğusu_sol_boşluq_xalı	sorğu_sol_açıq_boşluq_xalı, sorğu_sol_boşluğu_uzatmaq_balı
sorğusu_sağ_boşluq_xalı	query_sağ_açıq_boşluq_balı, sorğu_sağ_uzatmaq_boşluq_balı

Machine Translated by Google

```
                    0 -|.|| 5
sorğu               0 -AATT 4
<BLANKLINE>
hə də f             0 AACTT 5
                    0 |-.|| 5
sorğu               0 A-ATT 4
<BLANKLINE>
hə də f             0 AACTT 5
                    0 ||.-| 5
sorğu               0 AAT-T 4
<BLANKLINE>
hə də f             0 AACTT 5
                    0 ||.|- 5
sorğu               0 AATT- 4
<BLANKLINE>
```

7.6 Ə vvə lcə də n müə yyə n edilmiş ə və zetmə matrisində n və boşluq ballarından istifadə

Varsayılan olaraq, PairwiseAligner obyekti +1.0 uyğunluq balı, 0.0 uyğunsuzluğu hesabı ilə işə salınır. və bütün boşluq balları 0.0-a bə rabə rdir, baxmayaraq ki, bu, ümumiyyə tlə , sadə bir qiymə tlə ndirmə sxemi olmağın üstünlüyünə malikdir. ə n yaxşı performansı vermir. Bunun ə və zinə , ə vvə lcə də n tə yin edilmiş arqumenti seçmə k üçün arqument hesabından istifadə edə bilə rsiniz PairwiseAligner obyektini işə salarkə n qiymə tlə ndirmə sxemi. Hal-hazırda, tə qdim edilə n bal sxemlə ri blastn və meqablast, nukleotid düzülüşü üçün uyğundur və zülal üçün uyğun olan blastp hizalamalar. Bu qiymə tlə ndirmə sxemlə rinin seçilmə si PairwiseAligner obyektini defolt hesabla işə salacaq müvafiq olaraq BLASTN, MegaBLAST və BLASTP tə rə fində n istifadə edilə n parametrlə r.

```
>>> Bio import Align- də n
>>> düzlə şdirici = Align.PairwiseAligner(qol = "blastn")
>>> çap(aligner) # doctest:+ELLIPSIS
Parametrlə ri ilə cüt ardıcıl düzlə ndirici
    ə və zedici_matris: <...-də massiv obyekti>
    hə də f_daxili_açıq_boşluq_balı: -7.000000
    hə də f_daxili_uzatma_boşluğu_balı: -2.000000
    sol_açıq_boşluq_balı: -7.000000
    target_left_uzatmaq_boşluq_balı: -2.000000
    hə də f_sağ_açıq_boşluq_balı: -7.000000
    hə də f_sağ_uzatma_boşluğu_score: -2.000000
    sorğu_daxili_açıq_boşluq_balı: -7.000000
    sorğu_daxili_uzatma_boşluğu_balı: -2.000000
    query_left_open_boşluq_balı: -7.000000
    query_left_uzatma_boşluğu_balı: -2.000000
    query_sağ_açıq_boşluq_balı: -7.000000
    query_right_extend_boşluq_balı: -2.000000
    rejim: qlobal
<BLANKLINE>
>>> çap (aligner.ə və zedici_matris[:, :])
        A    T    G    C    S    w    R    Y    K    M    B    V    H    D    N
A 2,0 -3,0 -3,0 -3,0 -3,0 -1,0 -1,0 -3,0 -3,0 -1,0 -3,0 -1,0 -1,0 -1,0 -2,0
T -3,0 2,0 -3,0 -3,0 -3,0 -1,0 -3,0 -1,0 -1,0 -3,0 -1,0 -3,0 -1,0 -1,0 -2,0
G -3,0 -3,0 2,0 -3,0 -1,0 -3,0 -1,0 -3,0 -1,0 -3,0 -1,0 -1,0 -3,0 -1,0 -2,0
C -3,0 -3,0 -3,0 2,0 -1,0 -3,0 -3,0 -1,0 -3,0 -1,0 -1,0 -1,0 -1,0 -3,0 -2,0
```

Machine Translated by Google

S -3.0 -3.0 -1.0 -1.0 -1.0 -3.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -2.0 W -1.0 -1.0 -3.0 -3.0 -3.0 -1.0 -1.0 -1.0 -1.0.0. -1,0 -2,0

R -1,0 -3,0 -1,0 -3,0 -1,0 -1,0 -1,0 -3,0 -1,0 -1,0 -1,0 -1,0 -1,0 -1,0 -2,0
Y -3,0 -1,0 -3,0 -1,0 -1,0 -1,0 -3,0 -1,0 -1,0 -1,0 -1,0 -1,0 -1,0 -1,0 -2,0
K -3,0 -1,0 -1,0 -3,0 -1,0 -1,0 -1,0 -1,0 -1,0 -3,0 -1,0 -1,0 -1,0 -1,0 -2,0
M -1,0 -3,0 -3,0 -1,0 -1,0 -1,0 -1,0 -1,0 -3,0 -1,0 -1,0 -1,0 -1,0 -1,0 -2,0
B -3,0 -1,0 -1,0 -1,0 -1,0 -1,0 -1,0 -1,0 -1,0 -1,0 -1,0 -1,0 -1,0 -1,0 -2,0
V -1,0 -3,0 -1,0 -1,0 -1,0 -1,0 -1,0 -1,0 -1,0 -1,0 -1,0 -1,0 -1,0 -1,0 -2,0
H -1,0 -1,0 -3,0 -1,0 -1,0 -1,0 -1,0 -1,0 -1,0 -1,0 -1,0 -1,0 -1,0 -1,0 -2,0
D -1,0 -1,0 -1,0 -3,0 -1,0 -1,0 -1,0 -1,0 -1,0 -1,0 -1,0 -1,0 -1,0 -1,0 -2,0
N -2,0 -2,0 -2,0 -2,0 -2,0 -2,0 -2,0 -2,0 -2,0 -2,0 -2,0 -2,0 -2,0 -2,0 -2,0
<BLANKLINE>

7.7 Düzə lişlə r üzə rində tə krarlama

aligner.align tə rə fində n qaytarılan düzülmə lə r bir növ də yişmə z tə krarlana bilə n obyektlə rdir (diapazona bə nzə r). Onlar Alignment obyektlə rinin silsilə sinə və ya siyahısına bə nzə sə lə r də , hə r bir Alignment obyektinin lazım olduqda dinamik şə kildə yaradılması mə nasında fə rqlə nirlə r. Bu yanaşma ona görə seçilmişdir ki, hizalanmaların sayı son də rə cə böyük ola bilə r, xüsusə n də zə if düzülmə lə r üçün (misal üçün Bölmə 7.10- a baxın).

Hizalamalarda aşağıdakı ə mə liyyatları yerinə yetirə bilə rsiniz: •

len(alignments) saxlanan düzlə şdirmə lə rin sayını qaytarır. Düzə lişlə rin sayı çox olsa belə , bu funksiya tez qayıdır. Düzə lişlə rin sayı olduqca böyükdürsə (adə tə n, 9,223,372,036,854,775,807-də n böyükdür, bu, 64 bitlik maşınlarda uzun int kimi saxlanıla bilə n ə n böyük tam ə də ddir), len(alignments) OverflowError-u qaldıracaq. Çox sayda hizalanma hizalanma keyfiyyə tinin aşağı olduğunu göstə rir.

```
>>> Bio importdan Align >>> aligner =
Align.PairwiseAligner() >>> alignments = aligner.align("AAA", "AA")
>>> len(alignments ) 3
```

- Siz indeks üzrə xüsusi düzülüşü çıxara bilə rsiniz:

```
>>> Bio importdan Align >>> aligner =
Align.PairwiseAligner() >>> alignments = aligner.align("AAA",
" AA") >>> print(alignments[2])

hə də f 0                    0 AAA 3
        -| | 3 0 -AA 2
sorğu
<BLANKLINE>
>>> çap (alignments[0]) hə də fi
                             0 AAA 3
                             0 ||- 3
sorğu                        0 AA- 2
<BLANKLINE>
```

- Siz alignmentlə r üzə rində tə krarlaya bilə rsiniz, mə sə lə n, aşağıdakı kimi

```
>>> hizalamalarda hizalama üçün : çap (alignment)
 ...
 ...
```

Machine Translated by Google

alignments.rewind çağırmaqla, siz alignments iteratorunu birinci hizalamaya və hizaları ə vvə ldə n tə krarlayın:

```
>>> Bio import Align- də n
>>> düzlə şdirici = Align.PairwiseAligner()
>>> hizalamalar = aligner.align("AAA", "AA")
>>> düzülmə lə rdə düzülmə üçün :
...     çap (düzlə şdirmə )
...
hə də f          0 AAA 3
                 0 ||- 3
sorğu            0 AA- 2
<BLANKLINE>
hə də f          0 AAA 3
                 0 |-| 3
sorğu            0 AA 2
<BLANKLINE>
hə də f          0 AAA 3
                 0 -|| 3
sorğu            0 -AA 2
<BLANKLINE>
>>> alignments.rewind()
>>> düzülmə lə rdə düzülmə üçün :
...     çap (düzlə şdirmə )
...
hə də f          0 AAA 3
                 0 ||- 3
sorğu            0 AA- 2
<BLANKLINE>
hə də f          0 AAA 3
                 0 |-| 3
sorğu            0 AA 2
<BLANKLINE>
hə də f          0 AAA 3
                 0 -|| 3
sorğu            0 -AA 2
<BLANKLINE>
```

Siz hə mçinin hizalama iteratorunu siyahıya və ya də ftə rə çevirə bilə rsiniz:

```
>>> hizalamalar = siyahı (düzlə şdirmə lə r)
```

Zə ng etmə yə cə hd etmə zdə n ə vvə l len(alignments)-a zə ng edə rə k hizalanmaların sayını yoxlamaq müdrikdir bütün düzülmə lə ri siyahı kimi saxlamaq üçün siyahı(alignments).

- Hizalanma hesabı (hizalamalarda hə r düzülmə üçün eyni də yə rə malikdir) atribut. Bu, fə rdi düzülmə lə ri çıxarmağa davam etmə zdə n ə vvə l hizalanma xalını yoxlamağa imkan verir:

```
>>> çap (alignments.score)
2.0
```

Machine Translated by Google

7.8 Arxa ipə uyğunlaşma

Varsayılan olaraq, ikili hizalayıcı sorğunun irə li telini hə də fin irə li xə ttinə uyğunlaşdırır.
Sorğu üçün hə də fin ə ks xə ttinə uyğunlaşma xalını hesablamaq üçün strand="-" istifadə edin:

```
>>> Bio import Align- də n
>>> Bio.Seq import reverse_complement- də n
>>> hə də f = "AAAACCC"
>>> sorğu = "AACC"
>>> düzlə şdirici = Align.PairwiseAligner()
>>> aligner.mismatch_score = -1
>>> aligner.internal_boşluq_score = -1
>>> aligner.score(hə də f, sorğu) # strand defolt olaraq "+"-dır
4.0
>>> aligner.score(hə də f, tə rs_tamamlayıcı(sorğu), strand="-")
4.0
>>> aligner.score(hə də f, sorğu, strand="-")
0.0
>>> aligner.score(hə də f, tə rs_tamamlayıcı(sorğu))
0.0
```

Zə ng edə rkə n strand="-" işarə si ilə ə ks tellə uyğunlaşma ə ldə edilə bilə r.
aligner.align:

```
>>> alignments = aligner.align (hə də f, sorğu)
>>> lent (düzlə şdirmə )
1
>>> çap (düzlə şdirmə lə r [0])
hə də f                    0 AAAACCC 7
                           0 --|||| - 7
sorğu                      0 --AACC- 4
<BLANKLINE>
>>> çap (alignments[0].format("yataq")) # doctest: +NORMALIZE_WHITESPACE
hə də f 2 4, 0,      6 sorğu 4          +      2      6 0 1
<BLANKLINE>
>>> alignments = aligner.align(hə də f, tə rs_tamamlayıcı(sorğu), strand="-")
>>> lent (düzlə şdirmə )
1
>>> çap (düzlə şdirmə lə r [0])
hə də f                    0 AAAACCC 7
        0 --|||| - 7
sorğu                      4 --AACC- 0
<BLANKLINE>
>>> çap (alignments[0].format("yataq")) # doctest: +NORMALIZE_WHITESPACE
hə də f 2 4, 0,      6 sorğu 4          -      2      6 0 1
<BLANKLINE>
>>> alignments = aligner.align(hə də f, sorğu, strand="-")
>>> lent (düzlə şdirmə )
2
>>> çap (düzlə şdirmə lə r [0])
hə də f                    0 AAAACCC---- 7
                           0 ---------- 11
```

Machine Translated by Google

```
sorğu                    4 -------GGTT 0
<BLANKLINE>
>>> çap (alignments[1]) hə də fi
                         0 ----AAACCC 7
                         0 ---------- 11
sorğu                    4 GGTT------- 0
<BLANKLINE>
```

Nə zə rə alın ki, sorğunun hə də fin ə ks zolağına uyğunlaşdırılması üçün xal, sol və sağ boşluq xalları fə rqli olduqda, sorğunun ə ks tamamlayıcısını hə də fin irə li zolağına uyğunlaşdırmaq üçün baldan fə rqli ola bilə r:

```
>>> aligner.left_gap_score = -0.5 >>> aligner.right_gap_score
= -0.2 >>> aligner.score (hə də f, sorğu) 2.8
```

```
>>> alignments = aligner.align(hə də f, sorğu) >>> len(alignments) 1 >>> çap
(alignments[0])
```

```
hə də f 0                0 AAAACCC 7
        --|||- 7 0 --AACC- 4
sorğu
<BLANKLINE>
>>> aligner.score(hə də f, tə rs_tamamlayıcı(sorğu), strand="-") 3.1
```

```
>>> alignments = aligner.align(hə də f, tə rs_tamamlayıcı(sorğu), strand="-") >>> len(düzlə şdirmə )

1
>>> çap (düzlə şdirmə lə r [0])
hə də fi                 0 AAAACCC 7
                         0 --|||- 7
sorğu                    4 --AACC- 0
<BLANKLINE>
```

7.9 Ə və zetmə matrislə ri

Ə və zetmə matrislə ri [11] iki müxtə lif qalığın bir-birini ə və z etmə ehtimalını tə snif etmə k üçün qiymə tlə ndirmə şə rtlə rini tə min edir. Ardıcıl müqayisə lə r apararkə n bu vacibdir. Biopython mə şhur PAM və BLOSUM matrislə ri də daxil olmaqla bir çox ümumi ə və zetmə matrislə ri tə qdim edir və hə mçinin öz ə və zetmə matrislə rinizi yaratmaq üçün funksionallıq tə min edir.

7.9.1 Massiv obyektlə ri Siz ə və zedici

matrislə ri indekslə rin tam ə də dlə r deyil, hə rflə r (nukleotidlə r və ya amin turşuları) olduğu 2D massivlə r kimi düşünə bilə rsiniz. Bio.Align.substitution_matrices-də ki Array sinfi hə m tam ə də dlə r, hə m də xüsusi sə tirlə r üzrə indekslə şdirmə ni də stə klə yə n numpy massivlə rin alt sinfidir. Massiv nümunə si bir ölçülü massiv və ya kvadrat iki ölçülü massiv ola bilə r. Mə sə lə n, birölçülü Array obyekti DNT ardıcıllığının nukleotid tezliyini saxlamaq üçün istifadə oluna bilə r, ikiölçülü Array obyekti isə ardıcıl düzülmə lə r üçün hesablama matrisini tə msil etmə k üçün istifadə edilə bilə r.

Birölçülü Massiv yaratmaq üçün yalnız icazə verilə n hə rflə rin ə lifbası göstə rilmə lidir:

Machine Translated by Google

```
>>> Bio.Align.substitution_matrices -də n idxal Massivi >>> sayar = Array("ACGT") >>> çap
(sayır)

A 0.0
C 0.0
G 0.0
T 0.0
<BLANKLINE>
```

İcazə verilə n hə rflə r ə lifba mülkiyyə tində saxlanılır:

```
>>> sayar. ə lifba 'ACGT'
```

Bu ə mlak yalnız oxumaq üçündür; ə sas _alphabet atributunun də yişdirilmə si gözlə nilmə z nə ticə lə rə sə bə b ola bilə r. Elementlə rə hə m hə rflə , hə m də tam indekslə daxil olmaq olar:

```
>>> sayar["C"] = -3 >>> sayar[2] =
7 >>> çap edir(sayır)

A 0.0
C -3.0
G 7.0 T 0.0

<BLANKLINE> >>>
sayar[1] -3.0
```

Ə lifbada olmayan hə rfdə n və ya hə ddə n kə narda olan indeksdə n istifadə IndexError-a sə bə b olacaq:

```
>>> sayar["U"]
Traceback (ə n son zə ng):
        ...
İndeks xə tası: 'U'
>>> sayır["X"] = 6
Traceback (ə n son zə ng):
        ...
İndeks xə tası: 'X'
>>> sayır[7]
Traceback (ə n son zə ng):
        ...
IndexError: 7 indeksi 4 ölçülü 0 oxu üçün hüdudlardan kə nardır
```

İki ölçülü massiv dims=2 göstə rilmə klə yaradıla bilə r:

```
>>> Bio.Align.substitution_matrices import Massivi >>> sayar = Array("ACGT", dims=2)
>>> çap (sayır )

        ACGT
A 0,0 0,0 0,0 0,0
C 0,0 0,0 0,0 0,0
G 0,0 0,0 0,0 0,0
T 0,0 0,0 0,0 0,0 <BLANKLINE>
```

Machine Translated by Google

Yenə də indekslə şdirmə üçün hə m hə rflə rdə n, hə m də tam ə də dlə rdə n istifadə edilə bilə r və ə lifbada olmayan hə rfin göstə rilmə si IndexError-a sə bə b olacaq:

```
>>> sayar["A", "C"] = 12,0 >>> sayar[2, 1]
= 5,0 >>> sayar[3, "T"] = -2 >>>
çap edir(sayır)

      A       C      G       T
A 0,0 12,0 0,0 0,0
C 0,0 0,0 0,0 0,0
G 0,0 5,0 0,0 0,0
T 0,0 0,0 0,0 -2,0
<BLANKLINE>
>>> sayar["X", 1]
Traceback (ə n son zə ng):
        ...
İndeks xə tası: 'X'
>>> sayır["A", 5]
Traceback (ə n son zə ng):
        ...
IndexError: indeks 5 4 ölçülü 1 ox üçün hüdudlardan kə nardır
```

İki ölçülü massivdə n sə tir və ya sütun seçmə k bir ölçülü massivi qaytaracaq:

```
>>> sayar = Array("ACGT", dims=2) >>> sayar["A",
"C"] = 12.0 >>> sayar[2, 1] = 5.0 >>>
sayar[3, "T"] = -2


>>> sayar["G"]
Massiv([0., 5., 0., 0.], ə lifba='ACGT')
            >>> sayar[:, "C"]

Massiv([12., 0., 5., 0.], ə lifba='ACGT')
```

Belə liklə , massiv obyektlə ri massiv və lüğə t kimi istifadə edilə bilə r. Onlar düz numpy çevrilə bilə r massivlə r və ya sadə lüğə t obyektlə ri:

```
>>> np kimi idxal numpy >>> x =
Array("ACGT") >>> x["C"] = 5


>>> x
Massiv([0., 5., 0., 0.], ə lifba='ACGT')

>>> a = np.array(x) # adi sayda massiv yaradın
>>> a
massiv([0., 5., 0., 0.]) >>> d = dict(x)
# sadə lüğə t yaradın
>>> d
{'A': 0,0, 'C': 5,0, 'G': 0,0, 'T': 0,0}
```

Machine Translated by Google

Massivin ə lifbası adə tə n sə tir olsa da, siz (də yişmə z) obyektlə rdə n ibarə t də stdə n də istifadə edə bilə rsiniz. Bu, mə sə lə n, açarların fə rdi nukleotidlə r və ya amin turşuları deyil, üç nukleotidli kodonlar olduğu kodon ə və zetmə matrisi üçün istifadə olunur.

Massivin ə lifba xassə si də yişmə z olsa da, seçə rə k yeni Array obyekti yarada bilə rsiniz. ə lifbadan maraqlandığınız hə rflə r. Mə sə lə n,

```
>>> a = Array("ABCD", dims=2, data=np.arange(16).reshape(4, 4)) >>> print(a)

        ABCDA 0,0 1,0 2,0 3,0 B
4,0 5,0 6,0 7,0

C 8.0 9.0 10.0 11.0
D 12,0 13,0 14,0 15,0
<BLANKLINE>
>>> b = a.select("CAD") >>> çap(b)

        C       A       D
C 10.0 8.0 11.0
A 2.0 0.0 3.0
D 14,0 12,0 15,0
<BLANKLINE>
```

Qeyd edə k ki, bu, hə m də ə lifbanı yenidə n sıralamağa imkan verir.

Ə lifbada olmayan hə rflə r üçün mə lumatlar sıfıra tə yin edilir:

```
>>> c = a.select("DEC") >>> çap(c)

        DECD 15,0 0,0
14,0 E 0,0 0,0 0,0

C 11.0 0.0 10.0
<BLANKLINE>
```

Array sinfi numpy massivinin alt sinfi olduğu üçün ondan belə istifadə etmə k olar. Ə gə r bir ValueError tetiklenir Riyazi ə mə liyyatda görünə n massiv obyektlə ri, mə sə lə n, müxtə lif ə lifbalara malikdir

```
>>> Bio.Align.substitution_matrices import Massiv >>> d = Array ( "ACGT") >>> r = Array("ACGU")
>>> d + r


Traceback (ə n son zə ng):
        ...
ValueError: ə lifbalar uyğun deyil
```

7.9.2 Ə və zedici matrisin cüt ardıcıl düzülüşündə n hesablanması

Array numpy massivin alt sinfi olduğundan, siz eyni şə kildə Array obyektində riyazi ə mə liyyatları tə tbiq edə bilə rsiniz. Burada biz bunu Escherichia coli və Bacillus subtilis-in 16S ribosomal RNT gen ardıcıllığının düzülüşündə n hesablama matrisini hesablayaraq göstə ririk. Ə vvə lcə PairwiseAligner obyekti yaradırıq (Fə sil 7-ə baxın) və onu blastn tə rə fində n istifadə edilə n standart xallarla işə salırıq:

```
>>> Bio.Align -də n idxal PairwiseAligner >>> aligner =
PairwiseAligner(scoring="blastn") >>> aligner.mode = "yerli"
```

Machine Translated by Google

Sonra, Escherichia coli və Bacillus subtilis-in 16S ribosomal RNT gen ardıcıllığını oxuyuruq (Tests/Align/ecoli.fa və Tests/Align/bsubtilis.fa-da verilmişdir) və onları bir-birinə uyğunlaşdırırıq:

```
>>> Bio idxaldan SeqIO >>> sequence1
= SeqIO.read("ecoli.fa", " fasta") >>> sequence2 = SeqIO.read("bsubtilis.fa",
"fasta") >>> alignments = aligner.align(sequence1, sequence2)
```

Yaradılan hizalanmaların sayı çox böyükdür:

```
>>> len(düzlə şdirmə ) 1990656
```

Bununla belə , onlar bir-birində n çox az fə rqlə ndiyinə görə , biz özbaşına olaraq birinci düzülüşü seçirik və hə r ə və zetmə nin sayını hesablayırıq:

```
>>> alignment = hizalamalar[0] >>>
ə və zlə mə lə r = alignment.ə və zetmə lə r >>> çap
(ə və zetmə lə r)
            ACGTA 307.0 19.0 34.0 19.0 C
15.0 280.0 25.0 29.0 G 34.0 24.0 401.0
20.0 T 24.0 36.0 20.0 228.0


<BLANKLINE>
```

Hə r bir ə və zetmə ehtimalını tapmaq üçün ümumi ə də də qarşı normallaşdırırıq və müşahidə olunan tezliklə rin simmetrik matrisini yaradırıq:

```
>>> müşahidə olunan_tezliklə r = ə və zetmə lə r / ə və zetmə lə r.sum() >>> müşahidə
olunan_tezliklə r = (müşahidə edilə n_tezliklə r + müşahidə edilə n_tezliklə r.transpose()) / 2.0 >>> çap (format(müşahidə edilə n_tezliklə r,
"%.4f"))
            A            C            G            T
A 0,2026 0,0112 0,0224 0,0142
C 0,0112 0,1848 0,0162 0,0215
G 0,0224 0,0162 0,2647 0,0132
T 0,0142 0,0215 0,0132 0,1505
<BLANKLINE>
```

Fon ehtimalı hə r bir ardıcıllıqla ayrı-ayrılıqda A, C, G və ya T nukleotidlə rinin tapılma ehtimalıdır. Bu, hə r bir sə tir və ya sütunun cə mi kimi hesablana bilə r:

```
>>> fon = müşahidə olunan_tezliklə r.sum(0) >>> çap (format(fon,
"%.4f"))
A 0,2505 C
0,2337 G
0,3165 T
0,1993
<BLANKLINE>
```

Tə sadüfi gözlə nilə n ə və zetmə lə rin sayı sadə cə fon paylanmasının özü ilə mə hsuludur:

Machine Translated by Google

```
>>> gözlə nilə n_tezliklə r = fon[:, Yoxdur].nöqtə (fon[Yox, :])
>>> çap (format(gözlə nilə n_tezliklə r, "%.4f"))
          A          C          G          T
A 0,0627 0,0585 0,0793 0,0499
C 0,0585 0,0546 0,0740 0,0466
G 0,0793 0,0740 0,1002 0,0631
T 0,0499 0,0466 0,0631 0,0397
<BLANKLINE>
```

Hesablama matrisi daha sonra müşahidə edilə n və gözlə nilə n ehtimalların nisbə t nisbə tinin loqarifmi kimi hesablana bilə r. ehtimallar:

```
>>> oddsratios = müşahidə olunan_tezliklə r / gözlə nilə n_tezliklə r
>>> np kimi idxal numpy
>>> xal_matrisi = np.log2(tə hlükə nisbə ti)
>>> çap (bal_matrisası)
        A       C       G       T
A 1,7 -2,4 -1,8 -1,8
C -2,4 1,8 -2,2 -1,1
G -1,8 -2,2 1,4 -2,3
T -1,8 -1,1 -2,3 1,9
<BLANKLINE>
```

Matris ikili düzlə şdirici üçün ə və zetmə matrisini tə yin etmə k üçün istifadə edilə bilə r (bax. Fə sil 7):

```
>>> aligner.ə və zetmə _matris = xal_matris
```

7.9.3 Çox ardıcıl düzülüşündə n ə və zedici matrisin hesablanması

Bu nümunə də biz ə vvə lcə protein.aln Clustalw faylından zülal ardıcıllığının düzülmə sini oxuyacağıq . (hə mçinin mövcuddur burada onlayn)

```
>>> Bio import Align- də n
>>> fayl adı = "protein.aln"
>>> alignment = Align.read (fayl adı, "clustal")
```

Bölmə 6.7.2-də bunun yerinə yetirilmə si ilə bağlı ə lavə mə lumat var.

Hizalanmanın ə və zetmə xüsusiyyə ti müxtə lif qalıqların neçə də fə ə və zlə ndiyini saxlayır bir-biri üçün:

```
>>> ə və zetmə lə r = hizalama.ə və zetmə lə r
```

Mə sə lə ni daha oxunaqlı etmə k üçün biz yalnız qütb yüklü yan zə ncirlə ri olan amin turşularını seçə cə yik:

```
>>> ə və zetmə lə r = ə və zetmə lə r.select("DEHKR")
>>> çap (ə və zetmə lə r)
          D          E          H          K          R
D 2360,0 270,0 15,0                    1,0 48,0
E 241,0 3305,0 15,0 45,0                          2.0
H      0,0 18,0 1235,0                 8.0        0.0
K      0.0        9,0 24,0 3218,0 130,0
R      2.0        2,0 17,0 103,0 2079,0
<BLANKLINE>
```

Machine Translated by Google

Digə r amin turşuları üçün cə rgə lə r və sütunlar matrisdə n çıxarıldı.

Sonra, matrisi normallaşdırırıq və simmetrik edirik.

```
>>> müşahidə olunan_tezliklə r = ə və zetmə lə r / ə və zetmə lə r.sum()
>>> müşahidə olunan_tezliklə r = (müşahidə edilə n_tezliklə r + müşahidə edilə n_tezliklə r.transpose()) / 2.0
>>> çap (format(müşahidə edilə n_tezliklə r, "%.4f"))
          D       E       H       K       R
D 0,1795 0,0194 0,0006 0,0000 0,0019
E 0,0194 0,2514 0,0013 0,0021 0,0002
H 0,0006 0,0013 0,0939 0,0012 0,0006
K 0,0000 0,0021 0,0012 0,2448 0,0089
R 0,0019 0,0002 0,0006 0,0089 0,1581
<BLANKLINE>
```

Sə tirlə r və ya sütunlar üzə rində cə mlə mə hə r bir qalığın nisbi tezliyini verir:

```
>>> fon = müşahidə olunan_tezliklə r.sum(0)
>>> çap (format(fon, "%.4f"))
D 0.2015
E 0.2743
H 0.0976
K 0.2569
R 0.1697
<BLANKLINE>
>>> background.sum()
1.0
```

Qalıq cütlə rinin gözlə nilə n tezliyi o zamandır

```
>>> gözlə nilə n_tezliklə r = fon[:, Yoxdur].nöqtə (fon[Yox, :])
>>> çap (format(gözlə nilə n_tezliklə r, "%.4f"))
          D       E       H       K       R
D 0,0406 0,0553 0,0197 0,0518 0,0342
E 0,0553 0,0752 0,0268 0,0705 0,0465
H 0,0197 0,0268 0,0095 0,0251 0,0166
K 0,0518 0,0705 0,0251 0,0660 0,0436
R 0,0342 0,0465 0,0166 0,0436 0,0288
<BLANKLINE>
```

Burada, background[:, None] gözlə nilə n_tezliklə rin də yə rlə rində n ibarə t tə k sütundan ibarə t 2D massivi yaradır, və rxpected_frequencies[Yox, :] bu də yə rlə ri tə k cə rgə də olan 2D massivi. Onların nöqtə mə hsulunu alaraq (daxili mə hsul) hə r bir girişin iki gözlə nilə n_tezlikdə n ibarə t olduğu gözlə nilə n tezliklə rin matrisini yaradır də yə rlə r bir-birinə vurulur. Mə sə lə n, gözlə nilə n_tezliklə r['D', 'E'] qalıq_tezliklə rinə ['D'] bə rabə rdir

İndi müşahidə olunan tezliklə ri gözlə nilə n tezliklə rə bölmə k yolu ilə log-otds matrisini hesablaya bilə rik. və loqarifmi götürün:

```
>>> np kimi idxal numpy
>>> xal_matrisası = np.log2(müşahidə edilə n_tezliklə r / gözlə nilə n_tezliklə r)
>>> çap (bal_matrisası)
          DE          H       K       R
D 2,1 -1,5 -5,1 -10,4 -4,2
E -1,5 1,7 -4,4 -5,1 -8,3
H -5,1 -4,4 3,3 -4,4 -4,7
K -10,4 -5,1 -4,4 1,9 -2,3
R -4,2 -8,3 -4,7 -2,3 2,5
<BLANKLINE>
```

Machine Translated by Google

Bu matris hizalamaları yerinə yetirə rkə n ə və zetmə matrisi kimi istifadə edilə bilə r. Mə sə lə n,

```
>>> Bio.Align -də n PairwiseAligner idxalı
>>> düzlə şdirici = PairwiseAligner()
>>> aligner.ə və zetmə _matris = xal_matris
>>> aligner.gap_score = -3.0
>>> alignments = aligner.align("DEHEK", "DHHKK")
>>> çap (düzlə şdirmə lə r [0])
hə də f                    0 DEHEK 5
                           0 |.|.| 5
sorğu                      0 DHHKK 5
<BLANKLINE>
>>> çap ("%.2f" % alignments.score)
-2.18
>>> xal = (
...         xal_matrisası["D", "D"]
...         + xal_matrisası["E", "H"]
...         + xal_matrisası["H", "H"]
...         + xal_matrisası["E", "K"]
...         + xal_matrisası["K", "K"]
... )
>>> çap ("%.2f" % xal)
-2.18
```

(ə traflı mə lumat üçün Fə sil 7-ə baxın).

7.9.4 Fayldan Massiv obyektlə rinin oxunması

Bio.Align.substitution_matrices-də n bir və iki ölçülü Array obyektlə rini oxumaq üçün tə hlilçi daxildir. fayl. Birölçülü massivlə r sadə iki sütunlu formatla tə msil olunur, birinci sütunda açar və ikinci sütun müvafiq də yə r. Mə sə lə n, hg38.chrom.sizes faylı (ə ldə edilmiş UCSC-də n), Biopython paylanmasının Testlə r/Align alt kataloqunda mövcuddur, ölçüsünü ehtiva edir İnsan genomunun hə r bir xromosomunun nukleotidlə ri hg38:

```
chr1            248956422
chr2            242193529
chr3            198295559
chr4            190214555
...
chrUn_KI270385v1        990
chrUn_KI270423v1        981
chrUn_KI270392v1        971
chrUn_KI270394v1        970
```

Bu faylı tə hlil etmə k üçün istifadə edin

```
>>>- dan Bio.Align idxal ə və zedici_matrislə ri
>>> tutacaq olaraq open("hg38.chrom.sizes") ilə :
...         cə dvə l = ə və zetmə _matrislə ri.oxu (tutacaq)
...
>>> çap (cə dvə l) # doctest: +ELLIPSIS
chr1 248956422.0
chr2 242193529.0
chr3 198295559.0
```

Machine Translated by Google

```
chr4 190214555.0
...
chrUn_KI270423v1        981.0
chrUn_KI270392v1        971,0
chrUn_KI270394v1        970.0
<BLANKLINE>
```

Də yə rlə ri tam ə də dlə r kimi oxumaq üçün dtype=int istifadə edin:

```
>>> qulp kimi open("hg38.chrom.sizes") ilə : table =
...         substitution_matrices.read(handle, int)
...
>>> çap (cə dvə l) # doctest: +ELLIPSIS chr1 248956422

chr2 242193529
chr3 198295559
chr4 190214555
...
chrUn_KI270423v1        981
chrUn_KI270392v1        971
chrUn_KI270394v1        970
<BLANKLINE>
```

İki ölçülü massivlə r üçün biz NCBI tə rə fində n tə min edilə n ə və zetmə matrislə rinin fayl formatına ə mə l edirik. Mə sə lə n, NCBI-nin protein-protein BLAST [1] proqramı blastp üçün standart ə və zetmə matrisi olan BLOSUM62 matrisi aşağıdakı kimi saxlanılır:

```
# Blosum62.iij saytından matblas tə rə fində n hazırlanmış matris # * sütunu
minimum xaldan istifadə edir
# BLOSUM 1/2 Bit Vahidlə rdə Kümelə nmiş Qiymə tlə ndirmə Matrisi #
Bloklar Verilə nlə r Bazası = /data/blocks_5.0/blocks.dat # Klaster Faiz:
>= 62 # Entropiya = 0,6979, Gözlə nilə n =
-0,5209 ARNDCQEGHILVKMFPST *

A 4 -1 -2 -2 0 -1 -1 0 -2 -1 -1 -1 -1 -2 -1 1 0 -3 -2 0 -2 -1 0 -4
R -1 5 0 -2 -3 1 0 -2 0 -3 -2 2 -1 -3 -2 -1 -1 -3 -2 -3 -1 0 -1 -4
N -2 0 6 1 -3 0 0 0 1 -3 -3 0 -2 -3 -2 1 0 -4 -2 -3 3 0 -1 -4
D -2 -2 1 6 -3 0 2 -1 -1 -3 -4 -1 -3 -3 -1 0 -1 -4 -3 -3 4 1 -1 -4
C 0 -3 -3 -3 9 -3 -4 -3 -3 -1 -1 -3 -1 -2 -3 -1 -1 -2 -2 -1 -3 -3 -2 -4
Q -1 1 0 0 -3 5 2 -2 0 -3 -2 1 0 -3 -1 0 -1 -2 -1 -2 0 3 -1 -4
E -1 0 0 2 -4 2 5 -2 0 -3 -3 1 -2 -3 -1 0 -1 -3 -2 -2 1 4 -1 -4
G 0 -2 0 -1 -3 -2 -2 6 -2 -4 -4 -2 -3 -3 -2 0 -2 -2 -3 -3 -1 -2 -1 -4
H -2 0 1 -1 -3 0 0 -2 8 -3 -3 -1 -2 -1 -2 -1 -2 -2 2 -3 0 0 -1 -4
...
```

Bu fayl Bio/Align/substitution_matrices/data altında Biopython paylanmasına daxildir. Bu faylı tə hlil etmə k üçün istifadə edin

```
>>> Bio-dan.İdxal ə və zetmə _matrislə rini uyğunlaşdırın >>> tutacaq kimi
open("BLOSUM62") ilə : matris =
...         substitution_matrices.read(handle)
...
>>> çap (matris.ə lifba)
```

Machine Translated by Google

```
ARNDCQEGHILKMFPSTWYVBZX*
>>> çap (matris["A", "D"])
-2.0
```

ilə başlayan başlıq sə tirlə ri atribut başlığında saxlanılır:

```
>>> matrix.header[0]
 'Matrix blosum62.iij saytından matblas tə rə fində n hazırlanmışdır'
```

İndi bu matrisi hizalayıcı obyektdə ə və zetmə matrisi kimi istifadə edə bilə rik:

```
>>> Bio.Align -də n PairwiseAligner idxalı
>>> düzlə şdirici = PairwiseAligner()
>>> aligner.ə və zedici_matris = matris
```

Array obyektini saxlamaq üçün ə vvə lcə sə tir yaradın:

```
>>> mə tn = str(matris)
>>> çap (mə tn) # doctest: +ELLIPSIS
# Matblas tə rə fində n blosum62.iij saytından hazırlanmış matris
# * sütunu minimum xaldan istifadə edir
1/2 Bit vahidlə rində # BLOSUM Clustered Scoring Matrix
# Bloklar verilə nlə r bazası = /data/blocks_5.0/blocks.dat
# Klaster Faiz: >= 62
# Entropiya = 0,6979, Gözlə nilə n = -0,5209
        A     RDQ    N          C          E     G     H     I     L     K     M     F     P     S ...
A 4,0 -1,0 -2,0 -2,0 0,0 -1,0 -1,0 0,0 -2,0 -1,0 -1,0 -1,0 -1,0 -2,0 -1,0 1,0 ...
R -1,0 5,0 0,0 -2,0 -3,0 1,0 0,0 -2,0 0,0 -3,0 -2,0 2,0 -1,0 -3,0 -2,0 -1,0 ...
N -2,0 0,0 6,0 1,0 -3,0 0,0 0,0 0,0 1,0 -3,0 -3,0 0,0 -2,0 -3,0 -2,0 1,0 ...
D -2,0 -2,0 1,0 6,0 -3,0 0,0 2,0 -1,0 -1,0 -3,0 -4,0 -1,0 -3,0 -3,0 -1,0 0,0 ...
C 0,0 -3,0 -3,0 -3,0 9,0 -3,0 -4,0 -3,0 -3,0 -1,0 -1,0 -3,0 -1,0 -2,0 -3,0 -1,0 ...
 ...
```

və mə tni fayla yazın.

7.9.5 Ə vvə lcə də n tə yin edilmiş ə və zetmə matrislə rinin yüklə nmə si

Biopython BLOSUM (Bloklar) daxil olmaqla ə də biyyatda müə yyə n edilmiş çoxlu ə və zedici matrislə ri ehtiva edir. Ə və zetmə Matrisi) [19] və PAM (Nöqtə Qə bul Edilə n Mutasiya) matrislə ri [9]. Bu matrislə r kimi mövcuddur Bio/Align/substitution_matrices/data qovluğunda düz fayllar və istifadə edə rə k Python-a yüklə nə bilə r ə və zetmə _matrislə ri alt modulunda yüklə mə funksiyası. Mə sə lə n, BLOSUM62 matrisi ola bilə r qaçaraq yüklə nir

```
>>>- dan Bio.Align idxal ə və zedici_matrislə ri
>>> m = ə və zetmə _matrislə ri.load("BLOSUM62")
```

Bu ə və zetmə matrisinin genetik kodda istifadə olunan 20 amin turşusundan ibarə t ə lifbası var. birmə nalı olmayan amin turşuları B (asparagin və ya aspartik turşu), Z (qlutamin və ya qlutamik turşu) və X (tə msil edir) hə r hansı bir amin turşusu) və ulduz işarə si ilə tə msil olunan dayanma kodonu:

```
>>> m.ə lifbası
 'ARNDCQEGHILKMFPSTWYVBZX*'
```

Mövcud ə və zetmə matrislə rinin tam siyahısını ə ldə etmə k üçün arqumentsiz yükdə n istifadə edin:

Machine Translated by Google

```
>>> substitution_matrices.load() # test: +ELLIPSIS ['BENNER22', 'BENNER6', 'BENNER74',
 'BLASTN', 'BLASTP', 'BLOSUM45', 'BLOSUM50', ...,                                    'TRANS']
```

Qeyd edə k ki, Schneider et al tə rə fində n verilmiş ə və zetmə matrisi. [40] üç nukleotidli kodonlardan ibarə t ə lifbadan istifadə edir:

```
>>> m = substitution_matrices.load("SCHNEIDER") >>> m.alphabet # doctest:
+ELLIPSIS ('AAA', 'AAC', 'AAG', 'AAT', 'ACA', 'ACC', 'ACG',
 'ACT', ..., 'TTG'),
```

7.10 Nümunə lə r

Tutaq ki, alpha.faa və beta.faa-da saxlanılan yuxarıdan eyni iki hemoglobin ardıcıllığı (HBA HUMAN, HBB HUMAN) arasında qlobal cüt-cüt uyğunlaşma etmə k istə yirsiniz: –

```
>>> Bio idxaldan Align >>> Bio idxaldan
SeqIO >>> seq1 = SeqIO.read("alpha.faa " , "
fasta") >>> seq2 = SeqIO.read("beta.faa", "fasta") >>> aligner =
Align.PairwiseAligner() >>> xal = aligner.score (score)( seq.sq1 ) 72.0
```

72.0 uyğunlaşma balı göstə rir. Fə rdi uyğunlaşmaları görmə k üçün bunu edin

```
>>> alignments = aligner.align(seq1.seq, seq2.seq)
```

Bu misalda optimal hizalanmaların ümumi sayı böyükdür (4×1037-də n çox) və len (düzlə şdirmə) çağırır.
Daşma xə tası yaradacaq:

```
>>> lent (düzlə şdirmə )
Traceback (ə n son zə ng):
 ...
OverflowError: optimal düzülmə lə rin sayı 9223372036854775807-də n çoxdur
```

Gə lin birinci sıralamaya nə zə r salaq:

```
>>> hizalama = hizalamalar[0]
```

Hizalama obyekti hizalanma hesabını, elə cə də düzlə şdirmə nin özünü saxlayır:

```
>>> çap (alignment.score) 72.0 >>> çap

(alignment) hə də fi
                          0 MV-LS-PAD--KTN--VK-AA-WGKV-----GAHAGEYGAEALE-RMFLSF----P-TTK 0 ||-|--|----|----|--|--||||-----|---||--|----|--|

sorğu                     0 MVHL-TP--EEK--SAV-TA-LWGKVNVDEVG---GE--A--L-GR--L--LVVYPWT--
<BLANKLINE>
hə də f                  41 TY--FPHF----DLSHGS---AQVK-G------HGKKV--A--DA-LTNAVAHV-DDMPN 60
                           ----|--|----|||------|-|--|------||||--|--|--|--|---|-
sorğu                    39 --QRF--FESFGDLS---TPDA-V-MGNPKVKAHGKKVLGAFSD-GL--A--H-LD---N
<BLANKLINE>
```

Machine Translated by Google

```
hə də f          79 ALS----A-LSD-LHAH--KLR-VDPV-NFK-LLSHC---LLVT--LAAHLPA----EFT 120 -|-----|-||--||----||--|||--||--|-|------|-|-|--|

sorğu            81 -L-KGTFATLS-ELH--CDKL-HVDP-ENF-RLL---GNVL-V-CVLA-H---HFGKEFT
<BLANKLINE>
hə də f         119 PA-VH-ASLDKFLAS---VSTV------LTS--KYR- 142
                180 |--|--|------|----|--|------|----||-- 217
sorğu           124 P-PV-QA------A-YQKV--VAGVANAL--AHKY-H 147
<BLANKLINE>
```

Daha yaxşı uyğunlaşmalar adə tə n boşluqların cə zalandırılması ilə ə ldə edilir: boşluğun açılması üçün daha yüksə k xə rclə r və mövcud boşluğu genişlə ndirmə k üçün daha az xə rclə r. Amin turşusu ardıcıllığı üçün uyğunluq xalları adə tə n PAM və ya BLOSUM kimi matrislə rdə kodlanır. Belə liklə , nümunə miz üçün daha mə nalı uyğunlaşma BLOSUM62 matrisində n istifadə edə rə k, 10 boşluq cə zası və 0,5 boşluq uzadılması cə zası ilə ə ldə edilə bilə r:

```
>>> Bio idxaldan Align >>> Bio idxaldan SeqIO
>>> Bio.Align idxal ə və zedici_matrislə rində n
>>> seq1 = SeqIO.read("alpha.faa", "fasta") >>> seq2 = SeqIO.read("beta.faa", "fasta")
>>> aligner = Align.PairwiseAligner = >> core aligner( -_ 0 ) op . >>>
aligner.extend_gap_score = -0.5 >>> aligner.substitution_matrix =
substitution_matrices.load ( "BLOSUM62") >>> hesab =
aligner.score(seq1.seq, seq2.seq) >>> print(score) 292.5
>>> alignments ( seq.seq.seq1) = aligner . len(düzlə şdirmə ) 2
```

```
>>> çap (alignments[0].hesab)
292.5
>>> çap (alignments[0]) hə də fi
                 0 MV-LSPADKTNVKAAWGKVGAHAGEYGAEALERMFLSFPTTKTYFPHF-DLS-----HGS
                 0 ||-|.|..|..|.|.||||--...|.|.|||.|.....|.|...|..|-|||-----.|.
sorğu            0 MVHLTPEEKSAVTALWGKV--NVDEVGGEALGRLLVVYPWTQRFFESFGDLSTPDAVMGN
<BLANKLINE>
hə də f         53 AQVKGHGKKVADALTNAVAHVDDMPNALSALSDLHAHKLRVDPVNFKLLSHCLLVTLAAH 60 ..||.|||||..|.....||.

sorğu           58 PKVKAHGKKVLGAFSDGLAHLDNLKGTFATLSELHCDKLHVDPENFRLLGNVLVCVLAHH
<BLANKLINE>
hə də f        113 LPAEFTPAVHASLDKFLASVSTVLTSKYR 142
               120 ...||||.|.|...|..|.|...|..||. 149 118
sorğu          FGKEFTPPVQAAYQKVVAGVANALAHKYH 147
<BLANKLINE>
```

Bu uyğunlaşma ə vvə llə r eyni ardıcıllıqlardan istifadə edə rə k EMBOSS iynə si ilə ə ldə etdiyimiz eyni nə ticə yə malikdir və eyni parametrlə r.

Yerli uyğunlaşdırmanı yerinə yetirmə k üçün aligner.mode parametrini 'yerli' olaraq tə yin edin:

```
>>> aligner.mode = "yerli" >>>
aligner.open_gap_score = -10 >>>
aligner.extend_gap_score = -1 >>> alignments =
aligner.align("LSPADKTNVKAA", "PEEKSAV")
```

Machine Translated by Google

```
>>> print(len(düzlə şdirmə lə r)) 1

>>> alignment = alignments[0] >>>
print(alignment) hə də fi
                                    2 PADKTNV 9
                                    0 |..|..| 7
sorğu                               0 PEEKSAV 7
<BLANKLINE>
>>> çap (alignment.score) 16.0
```

7.11 Ümumilə şdirilmiş qoşa düzülmə lə r

Ə ksə r hallarda PairwiseAligner tə k hə rfli nukleotidlə rdə n və ya amin turşularından ibarə t ardıcıllıqların (sə tirlə r və ya Seq obyektlə ri) hizalanmasını hə yata keçirmə k üçün istifadə olunur. Ümumiyyə tlə , PairwiseAligner ixtiyari obyektlə rin siyahılarına və ya də stlə rinə də tə tbiq oluna bilə r. Bu bölmə də bu cür ümumilə şdirilmiş cüt-cüt düzülmə lə rin bə zi nümunə lə ri tə svir olunacaq.

7.11.1 Ə və zedici matrisa və alfa-qısadan istifadə edə rə k ümumilə şdirilmiş cüt düzülmə lə r mə rc

Schneider və başqaları. [40] üç nukleotidli kodonların uyğunlaşdırılması üçün ə və zedici matrisa yaratmışdır (ə traflı mə lumat üçün aşağıda bölmə 7.9- a baxın). Bu ə və zetmə matrisi bütün üç hə rfli kodonlardan ibarə t ə lifba ilə ə laqə lə ndirilir:

```
>>> Bio.Align idxal ə və zetmə _matrislə rində n >>> m =
substitution_matrices.load("SCHNEIDER") >>> m.alphabet # doktest: +ELLIPSIS
(' AAA ', 'AAC', 'AAG', 'AAT', 'ACA', 'ACC', 'ACA', 'ACC', 'ACT'TT',
 'ACT'TT', ...
```

Kodon ardıcıllığını bir-birinə uyğunlaşdırmaq üçün bu matrisadan istifadə edə bilə rik:

```
>>> Bio import Align >>> aligner =
Align.PairwiseAligner() >>> aligner.substitution_matrix = m >>>
aligner.gap_score = -1.0 >>> s1 = ("AAT", "CTG", "TTT", "TTT")
>>> s2 = ("AAT", "TTA" aign >>> s2 = ( "AAT", "TTA"
aign . s2) >>> len(düzlə şdirmə ) 2
```

```
>>> çap (düzlə şdirmə lə r [0])
AAT CTG TTT TTT
 ||| ... ||| ---
AAT TTA TTT ---
<BLANKLINE>
>>> çap (düzlə şdirmə lə r [1])
AAT CTG TTT TTT
 ||| ...        --- |||
AAT TTA --- TTT
<BLANKLINE>
```

Qeyd edə k ki, bu nümunə də olduğu kimi TTT-ni TTA-ya uyğunlaşdırın:

Machine Translated by Google

```
AAT CTG TTT TTT ||| --- ...
 |||
AAT --- TTA TTT
```

daha aşağı bal alacaq:

```
>>> çap (m["CTG", "TTA"]) 7.6

>>> çap (m["TTT", "TTA"]) -0,3
```

güman ki, ona görə ki, CTG və TTA hə r ikisi lösini kodlayır, TTT isə fenilalanini kodlayır. Üç hə rfli kodon ə və zetmə matrisi də eyni amin turşusunu tə msil edə n kodonlar arasında üstünlük verir. Mə sə lə n, TTA CTC ilə müqayisə də CTG-yə üstünlük verir, baxmayaraq ki, hə r üç lösin kodu:

```
>>> s1 = ("AAT", "CTG", "CTC", "TTT") >>> s2 = ("AAT", "TTA", "TTT")
>>> alignments = aligner.align(s1, s2) >>>
len(alignments) 1
```

```
>>> çap (düzlə şdirmə lə r [0])
AAT CTG CTC TTT
 ||| ...          --- |||
AAT TTA --- TTT
<BLANKLINE>
>>> çap (m["CTC", "TTA"]) 6.5
```

7.11.2 Uyğunluq/uyğunsuzluq hesablarından istifadə edə rə k ümumilə şdirilmiş cüt-cüt düzülmə lə r ə lifba

Üç hə rfli amin turşusu simvollarından istifadə edə rə k, yuxarıdakı ardıcıllıqlar çevrilir

```
>>> s1 = ("Asn", "Leu", "Leu", "Phe") >>> s2 = ("Asn", "Leu", "Phe")
```

Üç hə rfli amin turşusu ə lifbasından istifadə etmə klə bu ardıcıllıqları birbaşa bir-birinə uyğunlaşdıra bilə rik:

```
>>> Bio import Align >>> aligner =
Align.PairwiseAligner() >>> aligner.alphabet = ['Ala', 'Arg', 'Asn',
'Asp', 'Cys', 'Gln', ' Glu', 'Gly', 'His' , ''et', I'le', 'L'et', 'Phe', 'Pro', 'Ser', 'Thr', 'Trp', 'Tyr', 'Val'] # fmt: atla
 ...
 ...
 ...
 ...
```

Biz BLOSUM62 matrisinin tə xmini olaraq +6/-1 uyğunluq və uyğunsuzluq xallarından istifadə edirik və bu ardıcıllıqları bir-birinə uyğunlaşdırırıq:

```
>>> aligner.match = +6 >>>
aligner.mismatch = -1 >>> alignments =
aligner.align(s1, s2) >>> print(len(alignments)) 2
```

Machine Translated by Google

```
>>> çap (düzlə şdirmə lə r [0])
Asn Leu Leu Phe
 ||| ||| --- |||
Asn Leu --- Phe
<BLANKLINE>
>>> çap (düzlə şdirmə lə r [1])
Asn Leu Leu Phe
 ||| --- ||| |||
Asn --- Leu Phe
<BLANKLINE>
>>> çap (alignments.score) 18.0
```

7.11.3 Uyğunluq/uyğunsuzluq xallarından və tam ə də dli ardıcıllıqlardan istifadə edə rə k ümumilə şdirilmiş cüt-cüt düzülmə lə r

Daxili olaraq, hizalama hə yata keçirə rkə n ilk addım, hizalayıcı ilə ə laqə li ə lifbadakı hə r bir ardıcıllıqdakı hə r hə rfin indekslə rində n ibarə t tam ə də d massivlə ri ilə iki ardıcıllığı ə və z etmə kdir. Bu addım birbaşa tam massivlə ri ötürmə klə yan keçə bilə r:

```
>>> numpy-ni np olaraq idxal edin >>>
Bio import Align >>> aligner =
Align.PairwiseAligner() >>> s1 = np.array([2, 10, 10, 13 ] , np.int32)
>>> s2 = np.array([2, 10, a.inttch.3 ) neri ] >>> >>> aligner.mismatch = -1 >>>
alignments = aligner.align(s1, s2) >>> print(len(alignments)) 2 >>>
print(alignments[0]) 2 10 10 13 | || --
|| 2 10 -- 13
<BLANKLINE>
>>> çap (düzlə şdirmə lə r [1])
2 10 10 13
 | -- || ||
2 -- 10 13
<BLANKLINE>
>>> çap (alignments.score) 18.0
```

Qeyd edə k ki, indekslə r bu nümunə də numpy.int32 tə rə fində n göstə rildiyi kimi 32 bitlik tam ə də dlə rdə n ibarə t olmalıdır.

Namə lum hə rflə r yenidə n joker simvol tə yin etmə klə və uyğun gə lə ndə n istifadə etmə klə daxil edilə bilə r İndeks kimi Unicode kod nöqtə si nömrə si:

```
>>> aligner.wildcard = "?" >>>
ord(aligner.wildcard) 63

>>> s2 = np.array([2, 63, 13], np.int32) >>> aligner.gap_score
= -3
```

Machine Translated by Google

```
>>> alignments = aligner.align(s1, s2) >>> print(len(alignments)) 2
```

```
>>> çap (düzlə şdirmə lə r [0])
2 10 10 13
 | ..        -- | |
2 63 -- 13
<BLANKLINE>
>>> çap edin(düzlə şdirmə lə r[1]) 2 10 10
13
 | --         .. | |
2 -- 63 13
<BLANKLINE>
>>> çap (alignments.score) 9.0
```

7.11.4 Ə və zedici matrisa və tam ə də ddə n istifadə edə rə k ümumilə şdirilmiş cüt düzülmə lə r ardıcıllıqlar

Tam ə də d ardıcıllığı, eyni zamanda, ə və zetmə matrisində n istifadə etmə klə düzülə bilə r, bu halda onunla ə laqə li ə lifbası olmayan bir ə də d kvadrat massiv. Bu halda, bütün indeks də yə rlə ri mə nfi olmamalı və ə və zedici matrisin ölçüsündə n kiçik olmalıdır:

```
>>> Bio importdan Align >>> import numpy
as np >>> aligner = Align.PairwiseAligner()
>>> m = np.eye(5) >>> m[0, 1 :] = m[1:, 0] = -2 >>> m[2, 2] = 3 >>>
çap (m) [.2] -[ 1. -2. -2. -2.] [-2.
1. 0. 0. 0.] [-2. 0. 3. 0. 0.] [-2. 0. 0. 1. 0.] [-2. 0. 0. 0.
1.]] >>>

aligner.ə və zetmə _matrisi = m >>>
   aligner.gap_score = -1 >>> s1 =
   np.array([0, 2, 3, 4], np.int32) >>> s2
   = np.array([0, >> 1 , 3 , al ] =
   aligner.align(s1, s2) >>>
print(len(alignments)) 2
```

```
>>> çap edin(alignments[0]) 0 - 2 3 4

 | - |          .    -
0 3 2 1 -
<BLANKLINE>
>>> çap edin(düzlə şdirmə lə r[1]) 0 - 2 3
4
 | - | - 0 3 2      .
- 1
<BLANKLINE>
```

Machine Translated by Google

Cə dvə l 7.2: CodonAligner obyektlə rinin metaatributları.

Meta atribut	Onun atributları
frameshift_minus_score frameshift_minus_two_score, frameshift_minus_one_score frameshift_plus_score frameshift_plus_two_score, frameshift_plus_one_score frameshift_two_score frameshift_minus_two_scoreplcoreuscore_score, frameshift_minus_score. frameshift_minus_bir_xal, frameshift_plus_bir_score frameshift_score frameshift_minus_two_score, frameshift_minus_bir_score, frameshift_plus_bir_score, frameshift_plus_score_score	

```
>>> çap (alignments.score) 2.0
```

7.12 Kodon düzülmə lə ri

Bio.Align modulundakı CodonAligner sinfi nukleotid ardıcıllığını kodlaşdırdığı amin turşusu ardıcıllığına uyğunlaşdırmaq üçün xüsusi düzlə şdirici tə tbiq edir. Tə rcümə zamanı çə rçivə də yişikliyi baş verə rsə , belə düzülmə lə r ə hə miyyə tsizdir.

7.12.1 Nukleotid ardıcıllığının amin turşusu ardıcıllığına uyğunlaşdırılması

Nukleotid ardıcıllığını amin turşusu ardıcıllığına uyğunlaşdırmaq üçün ə vvə lcə CodonAligner obyekti yaradın:

```
>>> Bio idxaldan Align >>> aligner =
Align.CodonAligner ()
```

CodonAligner obyekt düzlə ndiricisi hizalamalar üçün istifadə olunacaq hizalanma parametrlə rini saxlayır:

```
>>> çap (düzlə şdirici)
Parametrlə r joker işarə si ilə kodon hizalayıcısı: 'X'

    match_score: 1.0
    uyğunsuzluq_score: 0.0
    frameshift_minus_two_score: -3.0 frameshift_minus_one_score:
    -3.0 frameshift_plus_one_score: -3.0
    frameshift_plus_two_score: -3.0 <BLANKLINE>
```

Joker simvol, match_score və mismatch_score parametrlə ri yuxarıda tə svir edilə n PairwiseAligner sinfi ilə eyni şə kildə müə yyə n edilir (bax: Bölmə 7.2). Frameshift_minus_two_score, frameshift_minus_one_score, frameshift_plus_one_score və frameshift_plus_two_score parametrlə ri ilə müə yyə n edilmiş də yə rlə r düzülmə də müvafiq olaraq -2, -1, +1 və ya +2 çə rçivə sürüşmə si baş verdikdə , düzlə şdirmə hesabına ə lavə edilir. Varsayılan olaraq, çə rçivə sürüşmə balları -3.0 olaraq tə yin olunur. PairwiseAligner sinfinə bə nzə r (Cə dvə l 7.1), CodonAligner sinfi Cə dvə l 7.2- də göstə rildiyi kimi bu də yə rlə rin bir sırasına toplu olaraq istinad edə n ə lavə atributları müə yyə n edir . İndi iki nukleotid ardıcıllığını və onların kodlaşdırdıqları amin turşusu ardıcıllığını nə zə rdə n keçirə k:

```
>>> Bio.Seq -də n idxal Seq >>>
Bio.SeqRecord - dan idxal SeqRecord >>> nuc1 =
Seq("TCAGGGACTGCGAGAACCAAGCTACTGCTGCTGCTGGGCTGCGCTCTGCGCCGCAGGTGGGGCGCTGGAG") >>> rna1 = SeqRecord(nucRecord
(nuc) = >> 1, nuc = 1 ) "
Seq("TCAGGGACTTCGAGAACCAAGCGCTCCTGCTGCTGGCTGCGCTCGGCGCCGCAGGTGGAGCACTGGAG")
```

Machine Translated by Google

```
>>> rna2 = SeqRecord(nuc2, id="rna2") >>> aa1 =
Seq("SGTARTKLLLLLAALCAAGGALE") >>> aa2 =
Seq("SGTSRTKRLLLLAALGAAGGALE") >>> pro1 =
SeqRecord(aa1, id="pro1") >>> pro1 = SeqRecord(aa1,
id="pro1") >>> record (2)
```

İki zülal ardıcıllığı hə r ikisi 23 amin turşusundan ibarə t olsa da, birinci nukleotid ardıcıllığı 3 × 23 = 69 nukleotiddə n ibarə tdir, ikinci nükleotid ardıcıllığı isə yalnız 68 nukleotiddə n ibarə tdir:

```
>>> len(pro1) 23

>>> len(pro2) 23

>>> len(rna1)
69
>>> len(rna2)
68
```

Bu, ikinci nukleotid ardıcıllığının tə rcümə si zamanı -1 kadr sürüşmə si hadisə si ilə bağlıdır. rna1-i pro1-ə , rna2-ni pro2-yə uyğunlaşdırmaq üçün CodonAligner.align istifadə edin, Alignment obyektlə rinin iteratorunu qaytarın:

```
>>> alignments1 = aligner.align(pro1, rna1) >>> len(alignments1) 1


>>> hizalama1 = sonrakı(düzlə şdirmə 1) >>> çap(alignment1)
pro1 rna1
                    0 SGTARTKLLLLLAALCAAGG
                    0 TCAGGGACTGCGAGAACCAAGCTACTGCTGCTGCTGGCTGCGCTCTGCGCCGCAGGTGGG
<BLANKLINE>
pro1                20 ALE 23
rna1                60 GCGCTGGAG 69
<BLANKLINE>
>>> alignment1.coordinates
massivi([[ 0, 23], [ 0,
        69]]) >>>
alignment1[0]
 'SGTARTKLLLLLAALCAAGGALE'
>>> düzülmə 1[1]
 'TCAGGGACTGCGAGAACCAAGCTACTGCTGCTGCTGGCTGCGCTCTGCGCCGCAGGTGGGGCGCTGGAG'
>>> alignments2 = aligner.align(pro2, rna2) >>> len(alignments2) 1 >>>
alignment2 = next(alignments2)


>>> print(alignment2) pro2 rna2

                    0 SGTSRTKR 8
                    0 TCAGGGACTTCGAGAACCAAGCGC 24
<BLANKLINE>
pro2                8 LLLLAALGAAGGALE 23
rna2               23 CTCCTGCTGCTGGCTGCGCTCGGCGCCGCAGGTGGAGCACTGGAG 68
<BLANKLINE>
>>> alignment2[0]
 'SGTSRTKRLLLLAALGAAGGALE'
>>> düzülmə 2[1]
```

Machine Translated by Google

```
'TCAGGGACTTCGAGAACCAAGCGCCTCCTGCTGCTGGCTGCGCTCGGCGCCGCAGGTGGAGCACTGGAG' >>> alignment2.koordinatlar
massivi([[ 0, 8, 8, 23], [ 0, 24, 23, 68]])
```

Düzə liş1 69 nukleotidin 23 amin turşusuna davamlı uyğunlaşması olsa da, düzülmə 2-də 24 nukleotiddə n sonra -1 kadr sürüşmə si tapırıq. Alignment2[1] -1 kadr sürüşmə sini tə tbiq etdikdə n sonra nukleotid ardıcıllığını ehtiva etdiyinə görə , o, nuc2-də n bir nukleotid daha uzundur və birbaşa tə rcümə oluna bilə r, nə ticə də aa2 amin turşusu ardıcıllığı yaranır:

```
>>> Bio.Seq -də n idxal tə rcümə si >>> len(nuc2)
68
>>> len(alignment2[1]) 69

>>> tə rcümə et(alignment2[1])
'SGTSRTKRLLLLAALGAAGGALE'
>>> _ == aa2
Doğrudur
```

Hizalama hesabı alignments1 və alignments2 iteratorlarında və alignment1 və alignment2 fə rdi düzülmə lə rində atribut kimi saxlanılır:

```
>>> alignments1.score 23.0 >>>

alignment1.score 23.0 >>>

alignments2.score 20.0

>>> alignment2.score 20.0
```

burada rna1-pro1 düzülmə sinin balı düzülmüş amin turşularının sayına bə rabə rdir və rna2-pro2 düzülmə sinin balı çə rçivə də yişikliyinə görə cə zaya görə 3 azdır. Düzə lişin özünü hesablamadan hizalanma xalını hesablamaq üçün hesab metodundan istifadə etmə k olar:

```
>>> xal = aligner.score(pro1, rna1) >>> çap (bal) 23.0


>>> xal = aligner.score(pro2, rna2) >>> çap (bal) 20.0
```

7.12.2 Kodon ardıcıllığının çoxsaylı ardıcıl düzülmə sinin yaradılması

Tutaq ki, üçüncü ə laqə li amin turşusu ardıcıllığı və onunla ə laqə li nukleotid ardıcıllığı var:

```
>>> aa3 = Seq("MGTALLLLLAALCAAGGALE") >>> pro3
= SeqRecord(aa3, id="pro3") >>> nuc3 =
Seq("ATGGGAACCGCGCTGCTTTTGCTACTGGCCGCGCTCTGCGCCGCAGGTGGGGCCCTGGArna ( 3 >> nSeq ) id="rna3")
>>> nuc3.translate() == aa3
Doğrudur
```

Yuxarıda olduğu kimi, nukleotid ardıcıllığını amin turşusu ardıcıllığına uyğunlaşdırmaq üçün CodonAligner-də n istifadə edirik:

Machine Translated by Google

```
>>> alignments3 = aligner.align(pro3, rna3)
>>> lent (düzlə şdirmə 3)
1
>>> alignment3 = sonrakı(alignments3)
>>> çap (düzlə şdirmə 3)
pro3                0 MGTALLLLLAALCAAGGALE
rna3                0 ATGGGAACCGCGCTGCTTTTGCTACTGGCCGCGCTCTGCGCCGCAGGTGGGGCCCTGGAG
<BLANKLINE>
pro3               20
rna3               60
<BLANKLINE>
```

Üç amin turşusu ardıcıllığı, mə sə lə n, ClustalW istifadə edə rə k, bir-birinə uyğunlaşdırıla bilə r. Budur, biz hizalanmanı ə l ilə yaradın:

```
>>> np kimi idxal numpy
>>> Bio.Align import Alignment- də n
>>> ardıcıllıqlar = [pro1, pro2, pro3]
>>> protein_alignment = Hizalama(
...         ardıcıllıqlar, koordinatlar=np.array([[0, 4, 7, 23], [0, 4, 7, 23], [0, 4, 4, 20]])
... )
>>> çap (protein_alignment)
pro1                0 SGTARTKLLLLLAALCAAGGALE 23
pro2                0 SGTSRTKRLLLLAALGAAGGALE 23
pro3                0 MGTA---LLLLLAALCAAGGALE 20
<BLANKLINE>
```

İndi nükleotid-zülal cüt-cüt ilə zülal düzülüşündə mapall metodundan istifadə edə bilə rik. uyğun kodon düzülmə sini ə ldə etmə k üçün arqument kimi hizalamalar:

```
>>> kodon_alignment = protein_alignment.mapall([alignment1, alignment2, alignment3])
>>> çap (kodon_alignment)
rna1                0 TCAGGGACTGCGAGAACCAAGCTA 24
rna2                0 TCAGGGACTTCGAGAACCAAGCGC 24
rna3                0 ATGGGAACCGCG---------CTG 15
<BLANKLINE>
rna1               24 CTGCTGCTGCTGGCTGCGCTCTGCGCCGCAGGTGGGGCGCTGGAG 69
rna2               23 CTCCTGCTGCTGGCTGCGCTCGGCGCCGCAGGTGGAGCACTGGAG 68
rna3               15 CTTTTGCTACTGGCCGCGCTCTGCGCCGCAGGTGGGGCCCTGGAG 60
<BLANKLINE>
```

7.12.3 Kodon düzülmə sinin tə hlili

Sayt başına sinonim olmayan və sinonim ə və zetmə lə rin sayının hesablanması

Kodon düzülmə sinin ə n mühüm tə tbiqi hə r sahə üzrə qeyri-sinonim ə və zetmə lə rin (dN) və hə r sahə üzrə sinonim ə və zetmə lə rin (dS) sayını tə xmin etmə kdir. Bunlar account_dn_ds ilə hesablana bilə r Bio.Align.analysis-də funksiya. Bu funksiya kodonların cüt-cüt düzülmə sini və daxil edilmə sini, hə mçinin hesablama metodunu tə yin edə n isteğe bağlı arqumentlə r metodu, codon_table (standart olaraq standartdır Kod), keçid və keçid sürə tlə rinin k nisbə ti və tarazlıq kodonunu müə yyə n etmə k üçün cfreq tezlik. Biopython hazırda üç hesablama ə saslı metodu (NG86, LWL85, YN00), elə cə də dN və dS-i qiymə tlə ndirmə k üçün maksimum ehtimal metodu (ML):

- NG86: Nei və Qojobori (1986) [33] (standart). Bu üsulla siz hə mçinin nisbə tini tə yin edə bilə rsiniz k arqumenti vasitə silə keçid və keçid də rə cə lə ri, default olaraq 1.0.

Machine Translated by Google

- LWL85: Li və başqaları. (1985) [28].
- YN00: Yang və Nielsen (2000) [51].
- ML: Qoldman və Yang (1994) [13]. Bu üsulla siz tarazlıq kodonunu da tə yin edə bilə rsiniz cfreq arqumenti vasitə silə aşağıdakı seçimlə rlə tezlik:
  - F1x4: tə qdim olunan kodon ardıcıllığında nukleotid tezliyini hesablayın və ondan istifadə edin fon kodon tezliyi;
  - F3x4: (standart) birinci, ikinci və üçüncü mövqe üçün nukleotid tezliyini ayrıca hesablayın tə qdim olunan kodonlarda və ondan fon kodon tezliyini hesablamaq üçün istifadə edin;
  - F61: psevdocount ilə tə min edilmiş kodon ardıcıllıqlarından kodonların tezliyini hesablayın 0.1.

Hesablama_dN_dS metodu ikili kodon düzülüşünə tə tbiq edilə bilə r. Ümumiyyə tlə , fə rqli hesablama üsulları dN və dS üçün bir qə də r fə rqli tə xminlə rlə nə ticə lə nə cə k:

```
>>> Bio.Align idxal tə hlilində n
>>> pairwise_codon_alignment = codon_alignment[:2]
>>> çap (ikili_kodon_alignment)
rna1                0 TCAGGGACTGCGAGAACCAAGCTA 24
                    0 |||||||||.||||||||||..
rna2                0 TCAGGGACTTCGAGAACCAAGCGC 24
<BLANKLINE>
rna1               24 CTGCTGCTGCTGGCTGCGCTCTGCGCCGCAGGTGGGGCGCTGGAG 69
                   24 ||.|||||||||||||||||||||||||||.||||||||||| 69
rna2               23 CTCCTGCTGCTGGCTGCGCTCGGCGCCGCAGGTGGAGCACTGGAG 68
<BLANKLINE>
>>> dN, dS = analiz.hesabla_dn_ds(cütlü_kodon_alignment, metod="NG86")
>>> print(dN, dS) # doctest: +ELLIPSIS
0,067715... 0,201197...
>>> dN, dS = analiz.hesabla_dn_ds(cütlük_kodon_alignment, metod="LWL85")
>>> print(dN, dS) # doctest: +ELLIPSIS
0.068728... 0.207551...

>>> dN, dS = analiz.hesabla_dn_ds(cütlü_kodon_alignment, metod="YN00")
>>> print(dN, dS) # doctest: +ELLIPSIS
0,081468... 0,127706...
>>> dN, dS = analiz.hesabla_dn_ds(cütlü_kodon_alignment, metod="ML")
>>> print(dN, dS) # doctest: +ELLIPSIS
0,069475... 0,205754...
```

Kodon ardıcıllığının çoxlu düzülüşü üçün dN və dS də yə rlə rinin matrisini hesablaya bilə rsiniz:

```
>>> dN, dS = analiz.hesabla_dn_ds_matris(kodon_alignment, metod="NG86")
>>> çap (dN)
rna1        0.000000
rna2        0,067715        0.000000
rna3        0.060204        0.145469        0.000000
      rna1        rna2 rna3
>>> çap (dS)
rna1        0.000000
rna2        0.201198        0.000000
rna3 0,664268 0,798957 rna1 rna2 rna3        0.000000
```

Machine Translated by Google

account_dn_ds_matrix tə rə fində n qaytarılan dN və dS obyektlə ri Bio.Phylo.TreeConstruction-da DistanceMatrix sinifinin nümunə lə ridir. Bu funksiya yalnız isteğe bağlı arqument kimi codon_table alır.

Bu iki ardıcıllıqdan Bio.Phylo.TreeConstruction istifadə edə rə k dN ağacı və dS ağacı yarada bilə rsiniz:

```
>>> Bio.Phylo.TreeConstruction -dan idxal DistanceTreeConstructor >>> dn_constructor =
DistanceTreeConstructor() >>> ds_constructor =
DistanceTreeConstructor() >>> dn_tree = dn_constructor.upgma(dN)
>>> ds_tree = ds_constructorS)pee > _print(print) <class
'Bio.Phylo.BaseTree.Tree'> >>> print(dn_tree) # doctest:
+ELLIPSIS Tree(rooted=True)



    Clade(branch_length=0, name='Inner2')
        Clade(budaq_uzunluğu=0,053296..., ad='rna2')
        Clade(branch_length=0,023194..., name='Inner1')
            Clade(budaq_uzunluğu=0,0301021..., ad='rna3')
            Clade(budaq_uzunluğu=0,0301021..., ad='rna1')
>>> print(ds_tree) # doctest: +ELLIPSIS Tree(köklü=Doğru)

    Clade(branch_length=0, name='Inner2')
        Clade(budaq_uzunluğu=0,365806..., ad='rna3')
        Clade(branch_length=0,265207..., name='Inner1')
            Clade(budaq_uzunluğu=0,100598..., ad='rna2')
            Clade(budaq_uzunluğu=0,100598..., ad='rna1')
```

McDonald-Kreitman testini yerinə yetirmə k

McDonald-Kreitman testi, eyni tə kamül prosesində n olub-olmadığını görmə k üçün növ daxilində ki sinonim ə və zetmə lə ri və qeyri-sinonim ə və zetmə lə ri növlə r arasında sinonim ə və zetmə lə r və qeyri-sinonim ə və zetmə lə ri müqayisə edə rə k uyğunlaşma tə kamülün miqdarını qiymə tlə ndirir. Test eyni növün müxtə lif fə rdlə rində n nümunə götürülmüş gen ardıcıllığını tə lə b edir. Aşağıdakı nümunə də meyvə milçə yinin Adh genində n istifadə edə cə yik. Mə lumata Drosophila melanogasterdə n 11 fə rd, Drosophila simulansdan 4 fə rd və Drosophila yakubadan 12 fə rd daxildir. Protein uyğunlaşdırılması mə lumatları və nukleotid ardıcıllıqları Tests/codonalign kataloqunda Biopython paylanmasında müvafiq olaraq adh.aln və drosophila.fasta faylları kimi mövcuddur. Bio.Align.analysis-də ki mktest funksiyası Mcdonald-Kreitman testini hə yata keçirir.

```
>>> Bio import SeqIO >>> from Bio
import Align >>> from Bio.Align
import CodonAligner >>> from Bio.Align.analysis import
import mktest >>> aligner = CodonAligner() >>>
nucleotide_records = SeqIO.index
("drosophila.fasta", "fasta") >>> nucleotide._record ( nukleotid._re ) üçün :

...         çap(nucleotide_record.description) # doctest: +ELLIPSIS
...
gi |9097 |emb |X57361.1 | Drosophila simulans (fə rdi c) ... gi |9099 |emb |X57362.1 |
Drosophila simulans (fə rdi d) ... gi |9101 |emb |X57363.1 | Drosophila simulans (fə rdi
e) ... gi |9103 |emb |X57364.1 | Drosophila simulans (fə rdi f) ... gi |9217 |emb |X57365.1 |
Drosophila yakuba (fə rdi a) ... gi |9219 |emb |X57366.1 | Drosophila yakuba (fə rdi b) ...
```

Machine Translated by Google

```
gi|9221|emb|X57367.1| Drosophila yakuba (fə rdi c) ...
gi|9223|emb|X57368.1| Drosophila yakuba (fə rdi d) ...
gi|9225|emb|X57369.1| Drosophila yakuba (fə rdi e) ...
gi|9227|emb|X57370.1| Drosophila yakuba (fə rdi f) ...
gi|9229|emb|X57371.1| Drosophila yakuba (fə rdi g) ...
gi|9231|emb|X57372.1| Drosophila yakuba (fə rdi h) ...
gi|9233|emb|X57373.1| Drosophila yakuba (fə rdi i) ...
gi|9235|emb|X57374.1| Drosophila yakuba (fə rdi j) ...
gi|9237|emb|X57375.1| Drosophila yakuba (fə rdi k) ...
gi|9239|emb|X57376.1| Drosophila yakuba (fə rdi l) ...
gi|156879|gb|M17837.1|DROADHCK D.melanogaster (Ja-F ştammı) ...
gi|156863|gb|M19547.1|DROADHCC D.melanogaster (Af-S ştammı) ...
gi|156877|gb|M17836.1|DROADHCJ D.melanogaster (Af-F ştammı) ...
gi|156875|gb|M17835.1|DROADHCI D.melanogaster (Wa-F ştammı) ...
gi|156873|gb|M17834.1|DROADHCH D.melanogaster (Fr-F ştammı) ...
gi|156871|gb|M17833.1|DROADHCG D.melanogaster (Fl-F ştammı) ...
gi|156869|gb|M17832.1|DROADHCF D.melanogaster (Ja-S ştammı) ...
gi|156867|gb|M17831.1|DROADHCE D.melanogaster (Fl-2S ştammı) ...
gi|156865|gb|M17830.1|DROADHCD D.melanogaster (Fr-S ştammı) ...
gi|156861|gb|M17828.1|DROADHCB D.melanogaster (Fl-1S ştammı) ...
gi|156859|gb|M17827.1|DROADHCA D.melanogaster (Wa-S ştammı) ...
>>> protein_alignment = Align.read("adh.aln", "clustal")
>>> len(zülalların düzülmə si)
27
>>> print(protein_alignment) # doctest: +ELLIPSIS
gi|9217|e gi|          0 MAFTLTNKNVVFVAGLGGIGLDTSKELVKRDLKNLVILDRIENPAAIAELKAINPKVTVT
9219|e gi|             0 MAFTLTNKNVVFVAGLGGIGLDTSKELVKRDLKNLVILDRIENPAAIAELKAINPKVTVT
9221|e                 0 MAFTLTNKNVVFVAGLGGIGLDTSKELVKRDLKNLVILDRIENPAAIAELKAINPKVTVT
...
gi|156859              0 MSFTLTNKNVIFVAGLGGIGLDTSKELLKRDLKNLVILDRIENPAAIAELKAINPKVTVT
<BLANKLINE>
...
<BLANKLINE>
gi|9217|e gi|        240 GTLEAIQWSKHWDSGI 256
9219|e gi|           240 GTLEAIQWSKHWDSGI 256
9221|e               240 GTLEAIQWSKHWDSGI 256
...
gi|156859            240 GTLEAIQWTKHWDSGI 256
<BLANKLINE>
>>> kodon_alignments = []
protein_alignment.sequences -də protein_rekordu üçün >>> :
...     nucleotide_record = nucleotide_records[protein_record.id]
...     alignments = aligner.align(protein_record, nucleotide_record)
...     assert len(düzlə şdirmə lə r) == 1
...     codon_alignment = sonrakı (düzlə şdirmə )
...     codon_alignments.append(kodon_alignment)
...
>>> print(codon_alignment) # doctest: +ELLIPSIS
gi|156859              0 MSFTLTNKNVIFVAGLGGIG
gi|156859              0 ATGTCGTTTACTTTGACCAACAAGAACGTGATTTTCGTTGCCGGTCTGGGAGGCATTGGT
<BLANKLINE>
```

Machine Translated by Google

```
gi|156859          20 LDTSKELLKRDLKNLVILDR
gi|156859          60 CTGGACACCAGCAAGGAGCTGCTCAAGCGCGATCTGAAGAACCTGGTGATCCTCGACCGC
<BLANKLINE>
...
<BLANKLINE>
gi|156859         240 GTLEAIQWTKHWDSGI 256
gi|156859         720 GGCACCCTGGAGGCCATCCAGTGGACCAAGCACTGGGACTCCGGCATC 768
<BLANKLINE>
>>> nucleotide_records.close() # İndekslə nmiş FASTA faylını bağlayın
>>> alignment = protein_alignment.mapall(kodon_alignments)
>>> çap(alignment) # doctest: +ELLIPSIS
gi|9217|e          0 ATGGCGTTTACCTTGACCAACAAGAACGTGGTTTTCGTGGCCGGTCTGGGAGGCATTGGT
gi|9219|e          0 ATGGCGTTTACCTTGACCAACAAGAACGTGGTTTTCGTGGCCGGTCTGGGAGGCATTGGT
gi|9221|e          0 ATGGCGTTTACCTTGACCAACAAGAACGTGGTTTTCGTGGCCGGTCTGGGAGGCATTGGT
...
gi|156859          0 ATGTCGTTTACTTTGACCAACAAGAACGTGATTTTCGTTGCCGGTCTGGGAGGCATTGGT
<BLANKLINE>
...
<BLANKLINE>
gi|9217|e        720 GGCACCCTGGAGGCCATCCAGTGGTCCAAGCACTGGGACTCCGGCATC 768
gi|9219|e        720 GGCACCCTGGAGGCCATCCAGTGGTCCAAGCACTGGGACTCCGGCATC 768
gi|9221|e        720 GGTACCCTGGAGGCCATCCAGTGGTCCAAGCACTGGGACTCCGGCATC 768
...
gi|156859        720 GGCACCCTGGAGGCCATCCAGTGGACCAAGCACTGGGACTCCGGCATC 768
<BLANKLINE>
>>> unikal_növ = ["Drosophila simulans", "Drosophila yakuba", "D.melanogaster"]
>>> növ = []
>>> alignment.sequences -də qeyd üçün :
...         tə svir = qeyd.tə svir
...         unikal_növlə rdə s üçün :
...             tə svirdə varsa :
...                 fasilə
...         başqa:
...             İstisna yüksə ltmə k (f" {tə svir} üçün növ tapmaq alınmadı")
...         növ.ə lavə (lə r)
...
>>> çap (növ)
['Drosophila yakuba', 'Drosophila yakuba', 'Drosophila yakuba', 'Drosophila yakuba', 'Drosophila yakuba >>> pvalue = mktest(düzlə şmə , növlə r)

>>> print(pvalue) # doctest: +ELLIPSIS
0,00206457...
```

Çoxlu kodon düzülüşünə ə lavə olaraq, mktest funksiyası düzülmə də ki hə r ardıcıllığın aid olduğu növlə ri daxil edir. Kodon cə dvə li isteğe bağlı arqument codon_table kimi tə qdim edilə bilə r.

Machine Translated by Google

Fə sil 8

Çox Ardıcıl Alignment obyektlə ri

Bu fə sildə MultipleSeqAlignment obyektlə ri yaradaraq ardıcıl düzlə şdirmə proqramının çıxışını tə hlil edə n köhnə MultipleSeqAlignment sinfi və Bio.AlignIO-da tə hliledicilə r tə svir olunur. Çoxlu Ardıcıl Düzə lişlə r dedikdə , biz bir-birinə uyğunlaşdırılmış çoxlu ardıcıllığın toplusunu nə zə rdə tuturuq - adə tə n boşluq simvollarının daxil edilmə si və aparıcı və ya arxadakı boşluqların ə lavə edilmə si ilə - bütün ardıcıllıq sə tirlə ri eyni uzunluqda olsun. Belə bir düzülmə hə rflə rin matrisi kimi qə bul edilə bilə r, burada hə r bir sıra daxili olaraq SeqRecord obyekti kimi saxlanılır.

Biz bu cür mə lumatları saxlayan MultipleSeqAlignment obyektini və onları müxtə lif fayl formatları kimi oxumaq və yazmaq üçün Bio.AlignIO modulunu tə qdim edə cə yik (ə vvə lki fə sildə ki Bio.SeqIO modulunun dizaynından sonra). Nə zə rə alın ki, hə m Bio.SeqIO, hə m də Bio.AlignIO ardıcıl uyğunlaşdırma fayllarını oxuya və yaza bilə r. Müvafiq seçim ə sasə n mə lumatlarla nə etmə k istə diyinizdə n asılı olacaq.

Bu fə slin son hissə si ClustalW kimi ümumi çoxlu ardıcıl düzlə şdirmə alə tlə rində n istifadə haqqındadır və Python-dan MUSCLE və nə ticə lə ri Biopython ilə tə hlil edin.

8.1 Ardıcıllığın uyğunlaşdırılmasının tə hlili və ya oxunması

Ardıcıllıqla düzülmə lə ri oxumaq üçün iki funksiyamız var, Bio.AlignIO.read() və Bio.AlignIO.parse() bunlar Bio.SeqIO-da tə qdim olunan konvensiyaya uyğun olaraq bir və ya bir neçə düzülmə ni ehtiva edə n fayllar üçündür.

Bio.AlignIO.parse() funksiyasından istifadə MultipleSeqAlignment obyektlə rini verə n iteratoru qaytaracaq. It-eratorlar adə tə n for loopunda istifadə olunur. Çoxsaylı müxtə lif düzülmə lə rə malik olacağınız hallara misal olaraq PHYLIP alə t seqboot-dan yenidə n nümunə lə nmiş düzlə şdirmə lə r və ya EMBOSS alə tlə ri su və ya iynə də n və ya Bill Pearson-un FASTA alə tlə rində n çoxlu cüt düzülmə lə r daxildir.

Bununla belə , bir çox hallarda siz yalnız bir düzülmə ni ehtiva edə n fayllarla mə şğul olacaqsınız. Bu halda, siz tə k MultipleSeqAlignment obyektini qaytaran Bio.AlignIO.read() funksiyasından istifadə etmə lisiniz.

Hə r iki funksiya iki mə cburi arqument gözlə yir:

1. Birinci arqument adə tə n açıq fayldan (Bölmə 25.1-ə baxın) və ya fayl adı.

2. İkinci arqument hizalanma formatını tə yin edə n kiçik hə rf sə tiridir. Bio.SeqIO-da olduğu kimi, biz sizin üçün fayl formatını tə xmin etmə yə çalışmırıq! Bax http://biopython.org/wiki/AlignIO də stə klə nə n formatların tam siyahısı üçün.

Birdə n çox uyğunlaşdırma ehtiva edə bilə n qeyri-müə yyə n fayl formatları ilə işlə mə k üçün aşağıdakı Bölmə 8.1.3- də müzakirə edilə n ə lavə seq_count arqumenti də var .

Machine Translated by Google

8.1.1 Tə k düzülmə lə r

Nümunə olaraq, PFAM və ya Stokholm faylında aşağıdakı annotasiya zə ngin zülal uyğunlaşmasını nə zə rdə n keçirin format:

```
# STOKHOLM 1.0
#=GS COATB_BPIKE/30-81 AC P03620.1
#=GS COATB_BPIKE/30-81 DR PDB; 1ifl; 1-52;
#=GS Q9T0Q8_BPIKE/1-52 AC Q9T0Q8.1
#=GS COATB_BPI22/32-83 AC P15416.1
#=GS COATB_BPM13/24-72 AC P69541.1
#=GS COATB_BPM13/24-72 DR PDB; 2cpb; 1-49;
#=GS COATB_BPM13/24-72 DR PDB; 2cps; 1-49;
#=GS COATB_BPZJ2/1-49 AC P03618.1
#=GS Q9T0Q9_BPFD/1-49 AC Q9T0Q9.1
#=GS Q9T0Q9_BPFD/1-49 DR PDB; 1nh4 A; 1-49;
#=GS COATB_BPIF1/22-73 AC P03619.2
#=GS COATB_BPIF1/22-73 DR PDB; 1ifk; 1-50;
COATB_BPIKE/30-81 #=GR          AEPNAATNYATEAMDSLKTQAIDLISQTWPVVTTVVVAGLVIRLFKKFSSKA
COATB_BPIKE/30-81 SS Q9T0Q8_BPIKE/  -HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH----
1-52 COATB_BPI22/32-83          AEPNAATNYATEAMDSLKTQAIDLISQTWPVVTTVVVAGLVIKLFKKFVSRA
COATB_BPM13/24-72               DGTSTATSYATEAMNSLKTQATDLIDQTWPVVTSVAVAGLAIRLFKKFSSKA
#=GR COATB_BPM13/24-72          AEGDDP...AKAAFNSLQASATEYIGYAWAMVVVIVGATIGIKLFKKFTSKA
SS COATB2-B1- Q9T0Q9_BPFD/1-49 #=GR  ---ST...CHCHHHHCCCCTCCCTTCHHHHHHHHHHHHHHHHHCTT--
Q9T0Q9_BPFD/1-49 SS             AEGDDP...AKAAFDSLQASATEYIGYAWAMVVVIVGATIGIKLFKKFASKA
COATB_BPIF1/22-73               AEGDDP...AKAAFDSLQASATEYIGYAWAMVVVIVGATIGIKLFKKFTSKA
#=GR COATB_BPIF1/22-73 SS #=GC  ------...-HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH--
SS_eksiler #=GC                 FAADDATSQAKAAFDSLTAQATEMSGYAWALVVLVVGATVGIKLFKKFVSRA
seq_eksiler //                  XX-HHHH--HHHHHH--HHHHHHH--HHHHHHHHHHHHHHHHHHHHHHH
                                XHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHC--
                                AEssss...AptAhDSLpspAT-hIu.sWshVsslVsAsluIKLFKKFsSKA
```

Bu, indidə n endirilmiş Phage Coat Gp8 (PF05371) PFAM girişi üçün toxum düzülüşüdür. https://pfam.xfam.org/ saytından PFAM-ın buraxılış tarixi . Bu faylı aşağıdakı kimi yüklə yə bilə rik (fə rz edə k ki, onda cari iş qovluğunda "PF05371 seed.sth" kimi diskdə saxlanılıb):

```
>>> Bio import AlignIO- dan
>>> hizalama = AlignIO.read("PF05371_seed.sth", "stockholm")
```

Bu kod hizalanmanın xülasə sini çap edə cə k:

```
>>> çap (düzlə şdirmə )
7 sə tir və 52 sütunla düzülmə
AEPNAATNYATEAMDSLKTQAIDLISQTWPVVTTVVVAGLVIRL...SKA COATB_BPIKE/30-81
AEPNAATNYATEAMDSLKTQAIDLISQTWPVVTTVVVAGLVIKL...SRA Q9T0Q8_BPIKE/1-52
DGTSTATSYATEAMNSLKTQATDLIDQTWPVVTSVAVAGLAIRL...SKA COATB_BPI22/32-83
AEGDDP---AKAAFNSLQASATEYIGYAWAMVVVIVGATIGIKL...SKA COATB_BPM13/24-72
AEGDDP---AKAAFDSLQASATEYIGYAWAMVVVIVGATIGIKL...SKA COATB_BPZJ2/1-49
AEGDDP---AKAAFDSLQASATEYIGYAWAMVVVIVGATIGIKL...SKA Q9T0Q9_BPFD/1-49
FAADDATSQAKAAFDSLTAQATEMSGYAWALVVLVVGATVGIKL...SRA COATB_BPIF1/22-73
```

Yuxarıdakı çıxışda ardıcıllığın kə sildiyini görə cə ksiniz. Bunun ə və zinə özümüzü yaza bilə rdik Sə tirlə r üzə rində SeqRecord obyektlə ri kimi tə krarlamaqla bunu istə diyimiz kimi formatlaşdırmaq üçün kodu daxil edin:

Machine Translated by Google

```
>>> Bio import AlignIO- dan >>> alignment
= AlignIO.read("PF05371_seed.sth", "stockholm") >>> print("Alignment length %i" %
alignment.get_alignment_length())
Hizalama uzunluğu 52 >>>
düzülmüş qeyd üçün : çap ("%s - %s" %
...             (record.seq, record.id))
...
AEPNAATNYATEAMDSLKTQAIDLISQTWPVVTTVVVAGLVIRLFKKFSSKA - COATB_BPIKE/30-81
AEPNAATNYATEAMDSLKTQAIDLISQTWPVVTTVVVAGLVIKLFKKFVSRA - Q9T0Q8_BPIKE/1-52
DGTSTATSYATEAMNSLKTQATDLIDQTWPVVTSVAVAGLAIRLFKKFSSKA - COATB_BPI22/32-83
AEGDDP---AKAAFNSLQASATEYIGYAWAMVVVIVGATIGIKLFKKFTSKA - COATB_BPM13/24-72
AEGDDP---AKAAFDSLQASATEYIGYAWAMVVVIVGATIGIKLFKKFASKA - COATB_BPZJ2/1-49
AEGDDP---AKAAFDSLQASATEYIGYAWAMVVVIVGATIGIKLFKKFTSKA - Q9T0Q9_BPFD/1-49
FAADDATSQAKAAFDSLTAQATEMSGYAWALVVLVVGATVGIKLFKKFVSRA - COATB_BPIF1/22-73
```

Siz hə mçinin müə yyə n fayl formatında göstə rmə k üçün uyğunlaşdırma obyektində Python-un daxili format funksiyasına zə ng edə bilə rsiniz – tə fə rrüatlar üçün Bölmə 8.2.2-ə baxın.

Yuxarıdakı xam faylda qeyd etdinizmi ki, ardıcıllıqların bir neçə sində verilə nlə r bazası ilə çarpaz istinadlar var. PDB və ə laqə li mə lum ikincili struktur? Bunu sınayın:

```
>>> düzülmüş qeyd üçün : ə gə r record.dbxrefs:
...
...                     çap ("%s %s" % (record.id, record.dbxrefs))
...
COATB_BPIKE/30-81 ['PDB; 1ifl; 1-52;']
COATB_BPM13/24-72 ['PDB; 2cpb; 1-49;', 'PDB; 2cps; 1-49;']
Q9T0Q9_BPFD/1-49 ['PDB; 1nh4 A; 1-49;']
COATB_BPIF1/22-73 ['PDB; 1ifk; 1-50;']
```

Bütün ardıcıllıq annotasiyalarına baxmaq üçün bunu sınayın:

```
>>> düzülmüş qeyd üçün : çap (qeyd)
...
...
```

PFAM http://pfam.xfam.org/family/PF05371 ünvanında gözə l veb interfeys tə qdim edir. bu, ə slində bu uyğunlaşdırmanı bir neçə başqa formatda yüklə mə yinizə imkan verə cə kdir. Fayl FASTA fayl formatında belə görünür:

```
>COATB_BPIKE/30-81
AEPNAATNYATEAMDSLKTQAIDLISQTWPVVTTVVVAGLVIRLFKKFSSKA
>Q9T0Q8_BPIKE/1-52
AEPNAATNYATEAMDSLKTQAIDLISQTWPVVTTVVVAGLVIKLFKKFVSRA
>COATB_BPI22/32-83
DGTSTATSYATEAMNSLKTQATDLIDQTWPVVTSVAVAGLAIRLFKKFSSKA
>COATB_BPM13/24-72
AEGDDP---AKAAFNSLQASATEYIGYAWAMVVVIVGATIGIKLFKKFTSKA
>COATB_BPZJ2/1-49
AEGDDP---AKAAFDSLQASATEYIGYAWAMVVVIVGATIGIKLFKKFASKA
>Q9T0Q9_BPFD/1-49
AEGDDP---AKAAFDSLQASATEYIGYAWAMVVVIVGATIGIKLFKKFTSKA
>COATB_BPIF1/22-73
FAADDATSQAKAAFDSLTAQATEMSGYAWALVVLVVGATVGIKLFKKFVSRA
```

Machine Translated by Google

Qeyd edə k ki, vebsaytda boşluqları nöqtə lə r (nöqtə lə r) və ya tire kimi göstə rmə k üçün seçim olmalıdır, biz tire göstə rmişik yuxarıda. Bunu "PF05371 seed.faa" faylı olaraq yüklə yib saxladığınızı fə rz etsə k, demə k olar ki, onu yüklə yə bilə rsiniz. tam eyni kod:

```
>>> Bio import AlignIO- dan
>>> hizalama = AlignIO.read("PF05371_seed.faa", "fasta")
>>> çap (düzlə şdirmə )
```

Bu kodda də yişə nlə rin hamısı fayl adı və format sə tridir. Eyni çıxışı ə ldə edə cə ksiniz ə vvə l ardıcıllıqlar və qeyd identifikatorları eynidir. Ancaq gözlə diyiniz kimi, hə r birini yoxlasanız SeqRecord-da heç bir annotasiya və verilə nlə r bazası çarpaz istinadları yoxdur, çünki bunlar FASTA-ya daxil deyil fayl formatı.

Nə zə rə alın ki, orijinalı çevirmə k üçün Sanger veb-saytından istifadə etmə k ə və zinə Bio.AlignIO-dan istifadə edə bilə rdiniz Stokholm formatlı faylı özünüz FASTA faylına çevirin (aşağıya baxın).

İstə nilə n də stə klə nə n fayl formatı ilə siz sadə cə olaraq də yişdirmə klə düzülmə ni eyni şə kildə yüklə yə bilə rsiniz format sə tri. Mə sə lə n, PHYLIP faylları üçün "phylip", NEXUS faylları üçün "nexus" və ya "qabartma" istifadə edin. hizalamalar EMBOSS alə tlə ri tə rə fində n çıxarılır. Wiki sə hifə sində tam siyahı var (http://biopython.org/wiki/AlignIO) və daxili sə nə dlə rdə (hə mçinin onlayn):

```
>>> Bio import AlignIO- dan
>>> kömə k (AlignIO)
```

8.1.2 Çoxsaylı Düzə lişlə r

Ə vvə lki bölmə tə k düzülmə ni ehtiva edə n faylları oxumağa yönə lmişdir. Ümumiyyə tlə , fayllar ola bilə r birdə n çox düzülmə ehtiva edir və bu faylları oxumaq üçün Bio.AlignIO.parse() funksiyasından istifadə etmə liyik.

Fə rz edə k ki, PHYLIP formatında kiçik bir düzülmə niz var:

```
 5      6
Alfa      AACAAC
Beta      AACCCC
Qamma     ACCAAC
Delta     CCACCA
Epsilon CCAAAC
```

Ə gə r siz PHYLIP alə tlə rində n istifadə edə rə k filogenetik ağacı yüklə mə k istə yirsinizsə , addımlardan biri bootseq alə tində n istifadə edə rə k bir çox yenidə n nümunə vi düzülmə lə r toplusunu yaradın. Bu kimi bir çıxış verə cə kdir qısalıq üçün qısaldılmış bu:

```
 5      6
Alfa      AAACCA
Beta      AAACCC
Qamma     ACCCCA
Delta     CCCAAC
Epsilon CCCAAA
 5      6
Alfa      AAACAA
Beta      AAACCC
Qamma     ACCCAA
Delta     CCCACC
Epsilon CCCAAA
 5      6
Alfa      AAAAAC
```

Machine Translated by Google

```
Beta        AAACCC
Qamma       AACAAC
Delta       CCCCCA
Epsilon CCCAAC
...
    5       6
Alfa        AAAACC
Beta        ACCCCC
Qamma       AAAACC
Delta       CCCCAA
Epsilon CAAACC
```

Bunu Bio.AlignIO istifadə edə rə k oxumaq istə yirsinizsə , istifadə edə bilə rsiniz:

```
>>> Bio import AlignIO -dan >>>
alignments = AlignIO.parse("resampled.phy", "phylip") >>> alignmentlə rdə
düzülmə üçün : print(alignment) print()
...
...
...
```

Bu, nümayiş üçün yenidə n qısaldılmış aşağıdakı çıxışı verə cə kdir:

```
5 sə tir və 6 sütunla düzülmə
AAACCA Alfa
AAACCC Beta
ACCCCA Qamma
CCCAAC Delta
CCCAAA Epsilon

5 sə tir və 6 sütunla düzülmə
AAACAA Alfa
AAACCC Beta
ACCCAA Qamma
CCCACC Delta
CCCAAA Epsilon

5 sə tir və 6 sütunla düzülmə
AAAAAC Alfa
AAACCC Beta
AACAAC Qamma
CCCCCA Deltası
CCCAAC Epsilon

...

5 sə tir və 6 sütunla düzülmə
AAAACC Alfa
ACCCCC Beta
AAAACC Qamma
CCCCAA Deltası
CAAACC Epsilon
```

Machine Translated by Google

Bio.SeqIO.parse() funksiyasında olduğu kimi, Bio.AlignIO.parse() funksiyasından istifadə iteratoru qaytarır. İstə nilə n ardıcıllıqla onlara daxil olmağa imkan verə n bütün düzülmə lə ri bir anda yaddaşda saxlamaq istə yirsinizsə , iteratoru siyahıya çevirin:

```
>>> Bio import AlignIO- dan >>>
alignments = list(AlignIO.parse("resampled.phy", "phylip")) >>> last_align = alignments[-1]
>>> first_align = alignments [0]
```

8.1.3 Birmə nalı düzülmə lə r

Bir çox hizalama fayl formatları birdə n çox düzülmə ni açıq şə kildə saxlaya bilə r və hə r düzülmə arasındakı bölgü aydındır. Lakin, ümumi ardıcıllıq fayl formatından istifadə edildikdə belə blok strukturu yoxdur. Ə n çox rast gə linə n belə və ziyyə t, hizalanmaların FASTA fayl formatında saxlanmasıdır.
Mə sə lə n, aşağıdakıları nə zə rdə n keçirin:

```
>Alfa
ACTACGACTAGCTCAG--G
>Beta
ACTACCGCTAGCTCAGAAG
> Qamma
ACTACGGCTAGCACAGAAG
>Alfa
ACTACGACTAGCTCAGG--
>Beta
ACTACCGCTAGCTCAGAAG
> Qamma
ACTACGGCTAGCACAGAAG
```

Bu, altı ardıcıllıqdan (tə krarlanan identifikatorlarla) ibarə t tə k düzülmə ola bilə r. Və ya identifikatorlara ə sasə n, bu, yə qin ki, hə r biri eyni uzunluğa malik olan üç ardıcıllıqla iki fə rqli düzülmə dir.

Bu növbə ti nümunə haqqında nə demə k olar?

```
>Alfa
ACTACGACTAGCTCAG--G
>Beta
ACTACCGCTAGCTCAGAAG
>Alfa
ACTACGACTAGCTCAGG--
> Qamma
ACTACGGCTAGCACAGAAG
>Alfa
ACTACGACTAGCTCAGG--
> Delta
ACTACGGCTAGCACAGAAG
```

Yenə də bu altı ardıcıllıqla tə k düzülmə ola bilə r. Ancaq bu də fə identifikatorlara ə saslanaraq, bunun tə sadüfə n hamısının eyni uzunluğa malik olan üç cüt düzülmə olduğunu tə xmin edə bilə rik.
Bu son nümunə oxşardır:

```
>Alfa
ACTACGACTAGCTCAG--G
```

Machine Translated by Google

```
>XXX
ACTACCGCTAGCTCAGAAG
>Alfa
ACTACGACTAGCTCAGG
>YYY
ACTACGGCAAGCACAGG
>Alfa
--ACTACGAC--TAGCTCAGG
>ZZZ
GGACTACGACAATAGCTCAGG
```

Bu üçüncü misalda, müxtə lif uzunluqlara görə , bu, bütün altı qeyddə n ibarə t tə k düzülmə kimi qə bul edilə bilmə z. Bununla belə , bu, üç cüt düzülmə ola bilə r.

Aydındır ki, bir FASTA faylında birdə n çox hizalanma saxlamağa çalışmaq ideal deyil. Bununla belə , siz bunları giriş faylları kimi qə bul etmə yə mə cbur olsanız, Bio.AlignIO bütün alignmentlə rin eyni sayda qeydlə rə malik olduğu ə n ümumi və ziyyə tin öhdə sində n gə lə bilə r. Bunun bir nümunə si EMBOSS alə tlə ri iynə si və su ilə hazırlana bilə n cüt düzülmə lə r toplusudur – baxmayaraq ki, bu və ziyyə tdə Bio.AlignIO format sə tri kimi "qabartma" istifadə edə rə k öz doğma çıxışını başa düşə bilmə lidir.

Bu FASTA nümunə lə rini bir neçə ayrı düzülmə kimi şə rh etmə k üçün Bio.AlignIO.parse() funksiyasından hə r düzülmə də neçə ardıcıllığın gözlə nildiyini müə yyə n edə n ə lavə seq_count arqumenti ilə istifadə edə bilə rik (bu nümunə lə rdə müvafiq olaraq 3, 2 və 2). Mə sə lə n, üçüncü nümunə ni giriş mə lumatları kimi istifadə edə rə k:

```
>>> AlignIO.parse -də düzülmə üçün (tutacaq, "fasta", seq_count=2):
...         hizalanmada qeyd üçün print("Alignment length %i" % alignment.get_alignment_length()) :
...
...                 print("%s - %s" % (record.seq, record.id)) print()
...
...
```

verilmə si:

```
Düzə liş uzunluğu 19
ACTACGACTAGCTCAG--G - Alfa
ACTACCGCTAGCTCAGAAG - XXX

Düzə liş uzunluğu 17
ACTACGACTAGCTCAGG - Alfa
ACTACGGCAAGCACAGG - YYY

Düzə liş uzunluğu 21
--ACTACGAC--TAGCTCAGG - Alpha
GGACTACGACAATAGCTCAGG - ZZZ
```

Seq_count arqumenti olmadan Bio.AlignIO.read() və ya Bio.AlignIO.parse() istifadə ilk iki nümunə üçün altı qeydin hamısını ehtiva edə n tə k düzülmə verə cə kdir. Üçüncü misal üçün, bir istisna qaldırılacaq, çünki uzunluqlar fə rqlidir və onların tə k düzülmə sinə mane olur.

Fayl formatının özü Bio.AlignIO-ya hə r düzülmə də ardıcıllıqların sayını birbaşa tə yin etmə yə imkan verə n blok strukturuna malikdirsə , o zaman seq_count arqumentinə ehtiyac yoxdur. Ə gə r o verilirsə və faylın mə zmunu ilə uyğun gə lmirsə , xə ta yaranır.

Qeyd edə k ki, bu isteğe bağlı seq_count arqumenti fayldakı hə r düzülmə nin eyni sayda ardıcıllığa malik olduğunu güman edir. Fə rziyyə olaraq siz qə ribə və ziyyə tlə rlə rastlaşa bilə rsiniz, mə sə lə n, hə r biri müxtə lif sıra ardıcıllığı olan bir neçə düzülmə ni ehtiva edə n FASTA faylı – baxmayaraq ki, bunun real dünya nümunə sini eşitmə k istə rdim. Mə lumatları daha gözə l bir fayl formatında ə ldə edə bilmə yə cə yinizi fə rz etsə k, düz irə lilə mə yolu yoxdur

Machine Translated by Google

Bio.AlignIO istifadə edə rə k bununla mə şğul olmaq. Bu və ziyyə tdə , ardıcıllıqla oxumağı düşünə bilə rsiniz Bio.SeqIO-dan istifadə edə rə k və uyğun olaraq hizalamalar yaratmaq üçün onları bir araya yığaraq.

8.2 Yazı düzülmə lə ri

Hizalama daxiletmə si (oxumaq) üçün Bio.AlignIO.read() və Bio.AlignIO.parse() istifadə etmə k haqqında danışdıq. faylları) və indi biz Bio.AlignIO.write()-a baxacağıq ki, bu da hizalanma çıxışı (faylların yazılması) üçündür. Bu a funksiya üç arqument götürür: bə zi MultipleSeqAlignment obyektlə ri (və ya geriyə uyğunluq üçün köhnə lmiş Alignment obyektlə ri), yazmaq üçün tutacaq və ya fayl adı və ardıcıllıq formatı.

Budur, çə tin bir şə kildə bir neçə MultipleSeqAlignment obyekti yaratmaqla başladığımız bir nümunə (onları fayldan yüklə mə k ə və zinə ə llə). Nə zə rə alın ki, biz bə zi SeqRecord obyektlə ri yaradırıq -də n uyğunlaşma.

```
>>> Bio.Seq -də n idxal Seq
>>> Bio.SeqRecord -dan SeqRecord idxalı
>>>- dan Bio.Align idxal MultipleSeqAlignment
>>> align1 = MultipleSeqAlignment(
...         [
...             SeqRecord(Seq("ACTGCTAGCTAG"), id="Alfa"),
...             SeqRecord(Seq("ACT-CTAGCTAG"), id="Beta"),
...             SeqRecord(Seq("ACTGCTAGDTAG"), id="Qamma"),
...         ]
... )
>>> align2 = MultipleSeqAlignment(
...         [
...             SeqRecord(Seq("GTCAGC-AG"), id="Delta"),
...             SeqRecord(Seq("GACAGCTAG"), id="Epsilon"),
...             SeqRecord(Seq("GTCAGCTAG"), id="Zeta"),
...         ]
... )
>>> align3 = MultipleSeqAlignment(
...         [
...             SeqRecord(Seq("ACTAGTACAGCTG"), id="Eta"),
...             SeqRecord(Seq("ACTAGTACAGCT-"), id="Theta"),
...             SeqRecord(Seq("-CTACTACAGGTG"), id="Iota"),
...         ]
... )
>>> my_alignments = [align1, align2, align3]
```

İndi Alignment obyektlə rinin siyahısı var, onları PHYLIP formatlı fayla yazacağıq:

```
>>> Bio import AlignIO- dan
>>> AlignIO.write(my_alignments, "my_example.phy", "phylip")
```

Və bu faylı sevimli mə tn redaktorunuzda açsanız, bu belə görünmə lidir:

```
 3 12
Alfa        ACTGCTAGCT AG
Beta        ACT-CTAGCT AG
Qamma       ACTGCTAGDT AG
 3 9
Delta       GTCAGC-AG
```

Machine Translated by Google

```
Epislon        GACAGCTAG
Zeta           GTCAGCTAG
  3 13
Eta            ACTAGTACAG CTG
Teta           ACTAGTACAG CT-
Iota           -CTACTACAG GTG
```

Mövcud uyğunlaşdırmanı yüklə mə k və bə lkə də bir qə də r sadə də n sonra onu saxlamaq istə mə k daha çox yayılmışdır müə yyə n satır və ya sütunların çıxarılması kimi manipulyasiya.

Tutaq ki, siz Bio.AlignIO.write() funksiyasının qulp üçün neçə düzülmə yazdığını bilmə k istə yirsiniz? Düzə lişlə riniz yuxarıdakı misal kimi siyahıda olsaydı, sadə cə olaraq len(my_alignments) istifadə edə bilə rsiniz, lakin qeydlə riniz generatordan/iteratordan gə lə ndə bunu edə bilmə zsiniz. Buna görə də Bio.AlignIO.write() funksiyası fayla yazılmış düzülmə lə rin sayını qaytarır.

Qeyd - Bio.AlignIO.write() funksiyasına artıq mövcud olan fayla yazmağı desə niz, heç bir xə bə rdarlıq edilmə də n köhnə faylın üzə rinə yazılacaq.

8.2.1 Ardıcıl uyğunlaşdırma fayl formatları arasında konvertasiya

Bio.AlignIO ilə ardıcıl uyğunlaşdırma fayl formatları arasında konvertasiya Bio.SeqIO ilə ardıcıl fayl formatları arasında çevirmə ilə eyni şə kildə işlə yir (Bölmə 5.5.2). Biz ümumiyyə tlə düzlə şdirmə lə ri Bio.AlignIO.parse() istifadə edə rə k yüklə yirik və sonra Bio.AlignIO.write() istifadə edə rə k onları saxlayırıq – və ya sadə cə Bio.AlignIO.convert() kömə kçi funksiyasından istifadə edirik.

Bu misal üçün, biz ə vvə llə r istifadə olunmuş PFAM/Stockholm format faylını yüklə yə cə yik və onu Clustal W formatlı fayl kimi saxlayacağıq:

```
>>> Bio import AlignIO- dan >>> count
= AlignIO.convert("PF05371_seed.sth", "stockholm", "PF05371_seed.aln", "clustal") >>> print("Çevrilmiş %i düzülmə lə r" % sayı)

1 düzülmə çevrildi
```

Və ya Bio.AlignIO.parse() və Bio.AlignIO.write() istifadə edə rə k:

```
>>> Bio import AlignIO - dan >>>
alignments = AlignIO.parse("PF05371_seed.sth", "stockholm") >>> count =
AlignIO.write(alignments, "PF05371_seed.aln", "clustal") >>> print("Converted %i alignments" %
count)
1 düzülmə çevrildi
```

Bio.AlignIO.write() funksiyası çoxsaylı hizalama obyektlə rinin verilmə sini gözlə yir. Yuxarıdakı nümunə də biz ona Bio.AlignIO.parse() tə rə fində n qaytarılan hizalanma iteratorunu verdik.

Bu halda, biz bilirik ki, faylda yalnız bir düzülmə var, ona görə də Bio.AlignIO.read() funksiyasından istifadə edə bilə rdik. bunun ə və zinə , lakin nə zə rə alın ki, biz bu uyğunlaşdırmanı tə k element siyahısı kimi Bio.AlignIO.write()-ə ötürmə liyik:

```
>>> Bio import AlignIO- dan >>>
alignment = AlignIO.read("PF05371_seed.sth", "stockholm")
>>> AlignIO.write([alignment], "PF05371_seed.aln", "clustal")
```

İstə nilə n halda, siz aşağıdakı mə zmunlu eyni yeni Clustal W format faylı “PF05371 seed.aln” ilə başa çatmalısınız:

```
CLUSTAL X (1.81) çox ardıcıl düzülmə
```

Machine Translated by Google

```
COATB_BPIKE/30-81          AEPNAATNYATEAMDSLKTQAIDLISQTWPVVTTVVVAGLVIRLFKKFSS
Q9T0Q8_BPIKE/1-52          AEPNAATNYATEAMDSLKTQAIDLISQTWPVVTTVVVAGLVIKLFKKFVS
COATB_BPI22/32-83          DGTSTATSYATEAMNSLKTQATDLIDQTWPVVTSVAVAGLAIRLFKKFSS
COATB_BPM13/24-72          AEGDDP---AKAAFNSLQASATEYIGYAWAMVVVIVGATIGIKLFKKFTS
COATB_BPZJ2/1-49           AEGDDP---AKAAFDSLQASATEYIGYAWAMVVVIVGATIGIKLFKKFAS
Q9T0Q9_BPFD/1-49           AEGDDP---AKAAFDSLQASATEYIGYAWAMVVVIVGATIGIKLFKKFTS
COATB_BPIF1/22-73          FAADDATSQAKAAFDSLTAQATEMSGYAWALVVLVVGATVGIKLFKKFVS

COATB_BPIKE/30-81          KA
Q9T0Q8_BPIKE/1-52          RA
COATB_BPI22/32-83          KA
COATB_BPM13/24-72          KA
COATB_BPZJ2/1-49           KA
Q9T0Q9_BPFD/1-49           KA
COATB_BPIF1/22-73          RA
```

Alternativ olaraq, “PF05371 seed.phy” adlandıracağımız PHYLIP formatlı fayl yarada bilə rsiniz:

```
>>> Bio import AlignIO- dan
>>> AlignIO.convert("PF05371_seed.sth", "stockholm", "PF05371_seed.phy", "phylip")
```

Bu də fə çıxış belə görünür:

```
 7 52
COATB_BPIK AEPNAATNYA TEAMDSLKTQ AIDLISQTWP VVTTVVVAGL VIRLFKKFSS
Q9T0Q8_BPI AEPNAATNYA TEAMDSLKTQ AIDLISQTWP VVTTVVVAGL VIKLFKKFVS
COATB_BPI2 DGTSTATSYA TEAMNSLKTQ ATDLIDQTWP VVTSVAVAGL AIRLFKKFSS
COATB_BPM1 AEGDDP---A KAAFNSLQAS ATEYIGYAWA MVVVIVGATI GIKLFKKFTS
COATB_BPZJ AEGDDP---A KAAFDSLQAS ATEYIGYAWA MVVVIVGATI GIKLFKKFAS
Q9T0Q9_BPF AEGDDP---A KAAFDSLQAS ATEYIGYAWA MVVVIVGATI GIKLFKKFTS
COATB_BPIF FAADDATSQA KAAFDSLTAQ ATEMSGYAWA LVVLVVGATV GIKLFKKFVS

                   KA
                   RA
                   KA
                   KA
                   KA
                   KA
                   RA
```

Orijinal PHYLIP uyğunlaşdırma fayl formatının böyük çatışmazlıqlarından biri də ardıcıllıq identifikatorlarının on simvol ilə ciddi şə kildə kə silir. Bu nümunə də , gördüyünüz kimi, ortaya çıxan adlar hə lə də unikaldır - amma çox oxunaqlı deyillə r. Nə ticə də , orijinal PHYLIP formatının daha rahat variantı indi olduqca geniş istifadə olunur:

```
>>> Bio import AlignIO- dan
>>> AlignIO.convert("PF05371_seed.sth", "stockholm", "PF05371_seed.phy", "filip-rahat")
```

Bu də fə çıxış bütün identifikatorların daxil olmasına icazə vermə k üçün daha uzun abzasdan istifadə edə rə k belə görünür tam:

```
 7 52
COATB_BPIKE/30-81 AEPNAATNYA TEAMDSLKTQ AIDLISQTWP VVTTVVVAGL VIRLFKKFSS
```

Machine Translated by Google

```
Q9T0Q8_BPIKE/1-52 AEPNAATNYA TEAMDSLKTQ AIDLISQTWP VVTTVVVAGL VIKLFKKFVS
COATB_BPI22/32-83 DGTSTATSYA TEAMNSLKTQ ATDLIDQTWP VVTSVAVAGL AIRLFKKFSS
COATB_BPM13/24-72 AEGDDP---A KAAFNSLQAS ATEYIGYAWA MVVVIVGATI GIKLFKKFTS
COATB_BPZJ2/1-49 AEGDDP---A KAAFDSLQAS ATEYIGYAWA MVVVIVGATI GIKLFKKFAS
Q9T0Q9_BPFD/1-49 AEGDDP---A KAAFDSLQAS ATEYIGYAWA MVVVIVGATI GIKLFKKFTS
COATB_BPIF1/22-73 FAADDATSQA KAAFDSLTAQ ATEMSGYAWA LVVLVVGATV GIKLFKKFVS

                                  KA
                                  RA
                                  KA
                                  KA
                                  KA
                                  KA
                                  RA
```

Orijinal ciddi PHYLIP formatı ilə işlə mə lisinizsə , identifikatorları bir şə kildə sıxışdırmalı və ya öz adlarınızı və ya nömrə lə mə sisteminizi tə yin etmə lisiniz. Aşağıdakı kod biti çıxışı saxlamazdan ə vvə l qeyd identifikatorlarını manipulyasiya edir:

```
>>> Bio import AlignIO -dan >>>
alignment = AlignIO.read("PF05371_seed.sth", "stockholm") >>> name_mapping =
{} >>> i üçün , siyahıda qeyd
edin ( alignment): name_mapping[i] = record.id record.id
...            = "seq%i" % i
...
...
>>> çap (ad_mapping)
{0: 'COATB_BPIKE/30-81', 1: 'Q9T0Q8_BPIKE/1-52', 2: 'COATB_BPI22/32-83', 3: 'COATB_BPM13/24-72', 4: 'CO >>> (AlignIO. "PF05371_seed.phy"
"phylip")
```

Bu kod yeni ardıcıllıq sistemində n orijinal identifikatora sadə xə ritə çə kmə ni qeyd etmə k üçün Python lüğə tində n istifadə etdi:

```
{
    0: "COATB_BPIKE/30-81",
    1: "Q9T0Q8_BPIKE/1-52",
    2: "COATB_BPI22/32-83",
    #...
}
```

Budur yeni (sə rt) PHYLIP format çıxışı:

```
 7 52
seq0        AEPNAATNYA TEAMDSLKTQ AIDLISQTWP VVTTVVVAGL VIRLFKKFSS
seq1        AEPNAATNYA TEAMDSLKTQ AIDLISQTWP VVTTVVVAGL VIKLFKKFVS
seq2        DGTSTATSYA TEAMNSLKTQ ATDLIDQTWP VVTSVAVAGL AIRLFKKFSS
seq3        AEGDDP---A KAAFNSLQAS ATEYIGYAWA MVVVIVGATI GIKLFKKFTS
seq4        AEGDDP---A KAAFDSLQAS ATEYIGYAWA MVVVIVGATI GIKLFKKFAS
seq5        AEGDDP---A KAAFDSLQAS ATEYIGYAWA MVVVIVGATI GIKLFKKFTS
seq6        FAADDATSQA KAAFDSLTAQ ATEMSGYAWA LVVLVVGATV GIKLFKKFVS

            KA
            RA
```

Machine Translated by Google

KA
KA
KA
KA
RA

Ümumiyyə tlə , identifikator mə hdudiyyə tinə görə , ciddi PHYLIP fayl formatları ilə işlə mə k ilk seçiminiz olmamalıdır. Digə r tə rə fdə n PFAM/Stokholm formatından istifadə sizə çoxlu ə lavə annotasiyalar da yazmağa imkan verir.

8.2.2 Düzlə şdirmə obyektlə rini formatlanmış sə tirlə r kimi ə ldə etmə k

Bio.AlignIO interfeysi tutacaqlara ə saslanır, yə ni müə yyə n bir fayl formatında sə tirdə düzülmə (lə r)i ə ldə etmə k istə yirsinizsə , bir az daha çox iş görmə lisiniz (aşağıya baxın). Bununla belə , yə qin ki, hizalanma obyektində Python-un daxili format funksiyasına zə ng etmə yə üstünlük verə cə ksiniz. Bu, çıxış formatı kimi Bio.AlignIO tə rə fində n də stə klə nə n kiçik hə rf sə tirini tə k arqument kimi qə bul edir. Mə sə lə n:

```
>>> Bio import AlignIO- dan >>>
alignment = AlignIO.read("PF05371_seed.sth", "stockholm") >>> çap
(format(düzlə şdirmə , "klastal"))
CLUSTAL X (1.81) çox ardıcıl düzülmə

COATB_BPIKE/30-81          AEPNAATNYATEAMDSLKTQAIDLISQTWPVVTTVVVAGLVIRLFKKFSS
Q9T0Q8_BPIKE/1-52          AEPNAATNYATEAMDSLKTQAIDLISQTWPVVTTVVVAGLVIKLFKKFVS
COATB_BPI22/32-83          DGTSTATSYATEAMNSLKTQATDLIDQTWPVVTSVAVAGLAIRLFKKFSS
...
```

Çıxış formatı spesifikasiyası olmadan format str ilə eyni çıxışı qaytarır.

Bölmə 4.6-da tə svir olunduğu kimi, SeqRecord obyekti Bio.SeqIO tə rə fində n də stə klə nə n çıxış formatlarından istifadə etmə klə oxşar üsula malikdir.

Daxili format, StringIO sapı ilə Bio.AlignIO.write() çağırır. Bunu özünüz edə bilə rsiniz mə sə lə n, Biopython-un köhnə versiyasından istifadə edirsinizsə kodu:

```
>>> io- dan StringIO import >>> Bio
import AlignIO- dan >>> alignments
= AlignIO.parse("PF05371_seed.sth", "stockholm") >>> out_handle = StringIO()

>>> AlignIO.write(alignments, out_handle, "clustal") 1

>>> clustal_data = out_handle.getvalue() >>> çap
(clustal_data)
CLUSTAL X (1.81) çox ardıcıl düzülmə

COATB_BPIKE/30-81          AEPNAATNYATEAMDSLKTQAIDLISQTWPVVTTVVVAGLVIRLFKKFSS
Q9T0Q8_BPIKE/1-52          AEPNAATNYATEAMDSLKTQAIDLISQTWPVVTTVVVAGLVIKLFKKFVS
COATB_BPI22/32-83          DGTSTATSYATEAMNSLKTQATDLIDQTWPVVTSVAVAGLAIRLFKKFSS
COATB_BPM13/24-72          AEGDDP---AKAAFNSLQASATEYIGYAWAMVVVIVGATIGIKLFKKFTS
...
```

Machine Translated by Google

8.3 Düzə lişlə rin manipulyasiyası

Yüklə mə və saxlama hizalarını ə hatə etdikdə n sonra onlarla başqa nə edə bilə cə yinizi nə zə rdə n keçirə cə yik.

8.3.1 Düzlə şdirmə lə rin dilimlə nmə si

Hə r şeydə n ə vvə l, bə zi mə nalarda hizalama obyektlə ri SeqRecord obyektlə rinin (sə tirlə r) Python siyahısı kimi fə aliyyə t göstə rir. Bu modeli nə zə rə alaraq ümid edirə m ki, len() (sə tirlə rin sayı) və iterasiya (hə r sıra SeqRecord kimi) hə rə kə tlə ri mə na kə sb edir:

```
>>> Bio import AlignIO- dan >>>
alignment = AlignIO.read("PF05371_seed.sth", "stockholm") >>> print("Sıraların sayı:
%i" % len(alignment))
Sə tirlə rin sayı: 7
>>> düzülmüş qeyd üçün : çap ("%s -
...             %s" % (record.seq, record.id))
...
AEPNAATNYATEAMDSLKTQAIDLISQTWPVVTTVVVAGLVIRLFKKFSSKA - COATB_BPIKE/30-81
AEPNAATNYATEAMDSLKTQAIDLISQTWPVVTTVVVAGLVIKLFKKFVSRA - Q9T0Q8_BPIKE/1-52
DGTSTATSYATEAMNSLKTQATDLIDQTWPVVTSVAVAGLAIRLFKKFSSKA - COATB_BPI22/32-83
AEGDDP---AKAAFNSLQASATEYIGYAWAMVVVIVGATIGIKLFKKFTSKA - COATB_BPM13/24-72
AEGDDP---AKAAFDSLQASATEYIGYAWAMVVVVIVGATIGIKLFKKFASKA - COATB_BPZJ2/1-49
AEGDDP---AKAAFDSLQASATEYIGYAWAMVVVIVGATIGIKLFKKFTSKA - Q9T0Q9_BPFD/1-49
FAADDATSQAKAAFDSLTAQATEMSGYAWALVVLVVGATVGIKLFKKFVSRA - COATB_BPIF1/22-73
```

Siz hə mçinin hizalanmaya daha çox sə tir ə lavə etmə k üçün siyahıya bə nzə r ə lavə və genişlə ndirmə üsullarından istifadə edə bilə rsiniz (SeqRecord obyektlə ri kimi). Siyahı metaforasını nə zə rə alaraq, düzülmə nin sadə dilimlə nmə si də mə na kə sb etmə lidir - o, başqa düzlə şdirmə obyektini geri qaytaran bə zi cə rgə lə ri seçir:

```
>>> çap (düzlə şdirmə )
7 cə rgə və 52 sütun ilə düzülmə
AEPNAATNYATEAMDSLKTQAIDLISQTWPVVTTVVVAGLVIRL...SKA COATB_BPIKE/30-81
AEPNAATNYATEAMDSLKTQAIDLISQTWPVVTTVVVAGLVIKL...SRA Q9TKEQ82-B
DGTSTATSYATEAMNSLKTQATDLIDQTWPVVTSVAVAGLAIRL...SKA COATB_BPI22/32-83 AEGDDP---
AKAAFNSLQASATEYIGYAWAMVVVIVGATIGIKL...SKA COATB_BPM13/24-72 AEGDDP---
AKAAFDSLQASATEYIGYAWAMVVVVIVGATIGIKL...SKA COATB_BPZJ2/1-49 AEGDDP---
AKAAFDSLQASATEYIGYAWAMVVVIVGATIGIKL...SKA Q9T0Q9_BPFD/1-49
FAADDATSQAKAAFDSLTAQATEMSGYAWALVVLVVGATVGIKL...SRA COATB_BPIF1/22-73 >>> çap
(alignment[3:7])
4 cə rgə və 52 sütunla düzülmə AEGDDP---
AKAAFNSLQASATEYIGYAWAMVVVIVGATIGIKL...SKA COATB_BPM13/24-72 AEGDDP---
AKAAFDSLQASATEYIGYAWAMVVVVIVGATIGIKL...SKAZ COATB24/1 AEGDDP---
AKAAFDSLQASATEYIGYAWAMVVVIVGATIGIKL...SKA Q9T0Q9_BPFD/1-49
FAADDATSQAKAAFDSLTAQATEMSGYAWALVVLVVGATVGIKL...SRA COATB_BPIF1/22-73
```

Sütunla seçmə k istə sə niz nə olacaq? NumPy matrisində n və ya massiv obyektlə rində n istifadə edə nlə r Buna tə ə ccüblə nmə yə cə ksiniz - ikiqat indeksdə n istifadə edirsiniz.

```
>>> çap (düzlə şdirmə [2, 6])
T
```

İki tam indeksdə n istifadə etmə k bunun üçün bir hə rf, qısa ə li çıxarır:

Machine Translated by Google

```
>>> çap (alignment[2].seq[6])
T
```

Tə k bir sütunu belə bir sə tir kimi çıxara bilə rsiniz:

```
>>> çap (düzlə şdirmə [:, 6])
TTT---T
```

Siz hə mçinin bir sıra sütunlar seçə bilə rsiniz. Mə sə lə n, çıxardığımız eyni üç sıranı seçmə k üçün daha ə vvə l, lakin onların yalnız ilk altı sütununu götürün:

```
>>> çap (düzlə şdirmə [3:6, :6])
3 cə rgə və 6 sütunla düzülmə
AEGDDP COATB_BPM13/24-72
AEGDDP COATB_BPZJ2/1-49
AEGDDP Q9T0Q9_BPFD/1-49
```

Birinci indeksi : kimi tə rk etmə k bütün sə tirlə ri götürmə k demə kdir:

```
>>> çap (düzlə şdirmə [:, :6])
7 sə tir və 6 sütun ilə düzülmə
AEPNAA COATB_BPIKE/30-81
AEPNAA Q9T0Q8_BPIKE/1-52
DGTSTA COATB_BPI22/32-83
AEGDDP COATB_BPM13/24-72
AEGDDP COATB_BPZJ2/1-49
AEGDDP Q9T0Q9_BPFD/1-49
FAADDA COATB_BPIF1/22-73
```

Bu, bizi bölmə ni silmə k üçün sə liqə li bir yola gə tirir. Yeddi ardıcıllığın üçündə boşluq olan 7, 8 və 9-cu sütunlara diqqə t yetirin:

```
>>> çap (düzlə şdirmə [:, 6:9])
7 sə tir və 3 sütunla düzülmə
TNY COATB_BPIKE/30-81
TNY Q9T0Q8_BPIKE/1-52
TSY COATB_BPI22/32-83
--- COATB_BPM13/24-72
--- COATB_BPZJ2/1-49
--- Q9T0Q9_BPFD/1-49
TSQ COATB_BPIF1/22-73
```

Yenə doqquzuncu sütundan sonra hə r şeyi ə ldə etmə k üçün dilimlə yə bilə rsiniz:

```
>>> çap (düzlə şdirmə [:, 9:])
7 sə tir və 43 sütunla düzülmə
ATEAMDSLKTQAIDLISQTWPVVTTVVVAGLVIRLFKKFSSKA COATB_BPIKE/30-81
ATEAMDSLKTQAIDLISQTWPVVTTVVVAGLVIKLFKKFVSRA Q9T0Q8_BPIKE/1-52
ATEAMNSLKTQATDLIDQTWPVVTSVAVAGLAIRLFKKFSSKA COATB_BPI22/32-83
AKAAFNSLQASATEYIGYAWAMVVVIVGATIGIKLFKKFTSKA COATB_BPM13/24-72
AKAAFDSLQASATEYIGYAWAMVVVIVGATIGIKLFKKFASKA COATB_BPZJ2/1-49
AKAAFDSLQASATEYIGYAWAMVVVIVGATIGIKLFKKFTSKA Q9T0Q9_BPFD/1-49
AKAAFDSLTAQATEMSGYAWALVVLVVGATVGIKLFKKFVSRA COATB_BPIF1/22-73
```

Machine Translated by Google

İndi maraqlısı odur ki, hizalama obyektlə rinin ə lavə edilmə si sütunla işlə yir. Bu, sütunlar blokunu silmə k üçün bunu etmə yə imkan verir:

```
>>> redaktə edildi = hizalama[:, :6] + düzlə şdirmə [:, 9:] >>> çap
(redaktə )
7 sə tir və 49 sütunla düzülmə
AEPNAAATEAMDSLKTQAIDLISQTWPVVTTVVVAGLVIRLFKKFSSKA COATB_BPIKE/30-81
AEPNAAATEAMDSLKTQAIDLISQTWPVVTTVVVAGLVIKLFKKFVSRA Q9T0Q8_BPIKE/1-52
DGTSTAATEAMNSLKTQATDLIDQTWPVVTSVAVAGLAIRLFKKFSSKA COATB_BPI22/32-83
AEGDDPAKAAFNSLQASATEYIGYAWAMVVVIVGATIGIKLFKKFTSKA COATB_BPM13/24-72
AEGDDPAKAAFDSLQASATEYIGYAWAMVVVIVGATIGIKLFKKFASKA COATB_BPZJ2/1-49
AEGDDPAKAAFDSLQASATEYIGYAWAMVVVIVGATIGIKLFKKFTSKA Q9T0Q9_BPFD/1-49
FAADDAAKAAFDSLTAQATEMSGYAWALVVLVVGATVGIKLFKKFVSRA COATB_BPIF1/22-73
```

Düzə liş ə lavə sinin başqa bir ümumi istifadə si bir neçə fə rqli gen üçün düzülmə lə ri meta-laynlaşdırmada birlə şdirmə k olardı. Baxmayaraq ki, diqqə tli olun - identifikatorlar üst-üstə düşmə lidir (SeqRecord obyektlə rinin ə lavə edilmə si üçün Bölmə 4.8-ə baxın). Ə vvə lcə düzülmə sə tirlə rini id-ə görə ə lifba sırası ilə çeşidlə mə k faydalı ola bilə r:

```
>>> edited.sort() >>>
çap (redaktə edilmiş)
7 cə rgə və 49 sütunla düzülmə
DGTSTAATEAMNSLKTQATDLIDQTWPVVTSVAVAGLAIRLFKKFSSKA COATB_BPI22/32-83
FAADDAAKAAFDSLTAQATEMSGYAWALVLVVGATVGIKLFKKFVSRA COATB3_2/
AEPNAAATEAMDSLKTQAIDLISQTWPVVTTVVVAGLVIRLFKKFSSKA COATB_BPIKE/30-81
AEGDDPAKAAFNSLQASATEYIGYAWAMVVVIVGATIGIKLFKKFTSKA COATB_BPM13/24-72
AEGDDPAKAAFDSLQASATEYIGYAWAMVVVIVGATIGIKLFKKFASKA COATB_BPZJ2/1-49
AEPNAAATEAMDSLKTQAIDLISQTWPVVTTVVVAGLVIKLFKKFVSRA Q9T0Q8_BPIKE/1-52
AEGDDPAKAAFDSLQASATEYIGYAWAMVVVIVGATIGIKLFKKFTSKA Q9T0Q9_BPFD/1-49
```

Nə zə rə alın ki, yalnız eyni sayda cə rgə olduqda iki düzülmə ə lavə edə bilə rsiniz.

8.3.2 Massivlə r kimi düzülmə lə r

Etdiyiniz işdə n asılı olaraq, hizalama obyektini hə rflə r massivinə çevirmə k daha faydalı ola bilə r – və siz bunu NumPy ilə edə bilə rsiniz:

```
>>> numpy kimi np kimi idxal
numpy >>> Bio import AlignIO- dan
>>> alignment = AlignIO.read("PF05371_seed.sth", "stockholm") >>> align_array =
np.array(alignment) >>> print("Masiv forması %i by
%i" % align_array.shape)
Massiv forması 7 x 52 >>>
align_array[:, :10] # doctest:+ELLIPSIS massivi([['A', 'E', 'P', 'N', 'A',
'A', 'T', 'N', 'Y', 'A'], ['A'''', E 'A', 'T', 'N', 'Y', 'A'], ['D', 'G', 'T', 'S', 'T', 'A', 'T', 'S', 'Y', 'A'], ['A',
            'E', 'G', 'D','','- '-', 'A'], ['A', 'E', 'G', 'D', 'D', 'P', '-', '-', '-', 'A'], ['A', 'E', 'G', 'D', 'D',
            'P', '-', '-,' 'A', 'A', 'D', 'D', 'A', 'T', 'S', 'Q', 'A']],...
```

Qeyd edə k ki, bu, orijinal Biopython uyğunlaşdırma obyektini və NumPy massivini yaddaşda ayrı qoyur obyektlə r - birini redaktə etmə k digə rini yenilə mə yə cə kdir!

Machine Translated by Google

8.3.3 Ə və zetmə lə rin hesablanması

Düzə lişin ə və zetmə xüsusiyyə ti düzülmə də ki hə rflə rin bir-birini nə qə də r tez-tez ə və z etdiyini bildirir. Bu, düzülmə də bütün sə tir cütlə rini götürmə klə , iki hə rfin bir-birinə düzülmə sayını hesablamaqla və bunun bütün cütlə r üzə rində cə mlə nmə si ilə hesablanır. Mə sə lə n,

```
>>> Bio.Seq -də n idxal Seq >>>
Bio.SeqRecord - dan idxal SeqRecord >>> Bio.Align -də n
idxal MultipleSeqAlignment >>> msa = MultipleSeqAlignment( [ [

...
...                 SeqRecord(Seq("ACTCCTA"), id="seq1"),
...                 SeqRecord(Seq("AAT-CTA"), id="seq2"),
...                 SeqRecord(Seq("CCTACT-"), id="seq3"),
...                 SeqRecord(Seq("TCTCCTC"), id="seq4"),
...             ]
... )
>>> çap (msa)
4 sıra və 7 sütun ilə düzülmə ACTCCTA seq1 AAT-CTA seq2
CCTACT- seq3
TCTCCTC seq4 >>>
ə və zetmə lə r =
msa.ə və zetmə

>>> çap (ə və zetmə lə r)
    A       C       T
A 2.0 4.5 1.0
C 4,5 10,0 0,5
T 1,0 0,5 12,0
<BLANKLINE>
```

Cütlə rin sıralanması ixtiyari olduğundan saylar diaqonaldan yuxarı və aşağı bə rabə r bölünür. Mə sə lə n, A-dan C-yə 9 düzülmə ['A', 'C'] mövqeyində 4.5 və ['C', 'A'] mövqeyində 4.5 olaraq saxlanılır. Bu tə nzimlə mə Bölmə 7.9 -da tə svir olunduğu kimi bu saylardan ə və zedici matrisin hesablanması zamanı riyaziyyatı asanlaşdırmağa kömə k edir .

Qeyd edə k ki, msa.substitutions yalnız düzülmə də görünə n hə rflə r üçün qeydlə ri ehtiva edir. edə bilə rsə n mə sə lə n, çatışmayan hə rflə r üçün qeydlə r ə lavə etmə k üçün seçim metodundan istifadə edin

```
>>> m = ə və zetmə lə r.select("ATCG") >>> çap (m)

    A       T       CG
A 2,0 1,0 4,5 0,0
T 1,0 12,0 0,5 0,0
C 4,5 0,5 10,0 0,0
G 0,0 0,0 0,0 0,0
<BLANKLINE>
```

Bu, hə m də ə lifbadakı hə rflə rin sırasını də yişmə yə imkan verir.

8.3.4 İcmal mə lumatların hesablanması

Bir uyğunlaşma ə ldə etdikdə n sonra, çox güman ki, bu barə də mə lumat tapmaq istə yə cə ksiniz. Hizalanma obyektinin özündə bir düzülmə haqqında mə lumat yarada bilə cə k bütün funksiyalara sahib olmağa çalışmaq ə və zinə , biz funksionallığı hizalamada fə aliyyə t göstə rə n ayrıca siniflə rə ayırmağa çalışdıq.

Machine Translated by Google

Obyekt haqqında ümumi mə lumatı hesablamağa hazırlaşmaq tezdir. Tutaq ki, alignment adlı hizalama obyektimiz var, mə sə lə n, 8-ci Fə sildə tə svir edildiyi kimi Bio.AlignIO.read(...) istifadə edə rə k oxuyun. Xülasə mə lumatını hesablayacaq obyekt ə ldə etmə k üçün bizə lazım olan hə r şey:

```
>>> Bio.Align -də n idxal AlignInfo >>> summary_align
= AlignInfo.SummaryInfo(msa)
```

summary_align obyekti çox faydalıdır və sizin üçün aşağıdakı sə liqə li işlə ri görə cə k:

1. Tez konsensus ardıcıllığını hesablayın – bölmə 8.3.5- ə baxın
2. Düzə liş üçün mövqeyə xas xal matrisini ə ldə edin – 8.3.6-cı bölmə yə baxın
3. Hizalanma üçün mə lumat mə zmununu hesablayın – bölmə 8.3.7-ə baxın
4. Düzə lişdə ə və zetmə lə r haqqında mə lumat yaradın – a. yaratmaq üçün bundan istifadə edə rə k bölmə 7.9 tə fə rrüatları ə və zetmə matrisi.

8.3.5 Tez konsensus ardıcıllığının hesablanması

Bölmə 8.3.4- də tə svir edilə n SummaryInfo obyekti uyğunlaşmanın sürə tli konsensusunu hesablamaq üçün funksionallığı tə min edir. summary_align adlı SummaryInfo obyektimiz olduğunu fə rz etsə k, konsensus hesablaya bilə rik:

```
>>> konsensus = summary_align.dumb_consensus()
>>> konsensus
Seq('XCTXCTX')
```

Adından da göründüyü kimi, bu, hə qiqə tə n sadə konsensus kalkulyatorudur və yalnız konsensusun hə r bir nöqtə sində bütün qalıqları ə lavə edə cə k və ə gə r ə n ümumi də yə r bə zi hə dd də yə rində n yüksə kdirsə , ümumi qalığı konsensusa ə lavə edə cə k. Ə gə r hə ddi çatmazsa, konsensusa qeyri-müə yyə nlik xarakteri ə lavə edir. Qaytarılan konsensus obyekti Seq obyektidir.

Siz isteğe bağlı parametrlə ri ötürmə klə dumb_consensus-un necə işlə diyini tə nzimlə yə bilə rsiniz:

eşik Bu, müə yyə n bir qalığın bir mövqedə nə qə də r ümumi olmasını tə yin edə n hə dddir ə lavə edilmə zdə n ə vvə l. Defolt 0,7-dir (70% demə kdir).

qeyri-müə yyə n xarakter Bu istifadə edilə cə k qeyri-müə yyə nlik xarakteridir. Standart 'N'dir.

Alternativ olaraq, alignment atributundan istifadə etmə klə msa çox ardıcıl düzülmə obyektini yeni üslublu Alignment obyektinə çevirə bilə rsiniz (bax. bölmə 6.1) (bax bölmə 8.4):

```
>>> alignment = msa.alignment
```

Daha sonra siz Motif obyekti yarada bilə rsiniz (bax. bölmə 17.1):

```
>>> Bio.motifs -də n idxal Motif >>> motiv =
Motif("ACGT", düzülmə )
```

və tez konsensus ardıcıllığı ə ldə edin:

```
>>> motiv.konsensus
Seq('ACTCCTA')
```

motif.counts.calculate_consensus metodu (bax bölmə 17.1.2) konsensus ardıcıllığının necə hesablanacağını ə traflı şə kildə müə yyə n etmə yə imkan verir. Mə sə lə n,

```
>>> motif.counts.calculate_consensus(şə xsiyyə t=0,7)
 'NCTNCTN'
```

Machine Translated by Google

8.3.6 Mövqe Xüsusi Xal Matrislə ri Mövqe xüsusi xal

matrislə ri (PSSM) uyğunlaşma mə lumatını konsensusdan fə rqli şə kildə ümumilə şdirir və müxtə lif tapşırıqlar üçün faydalı ola bilə r. Ə sasə n, PSSM bir sayma matrisidir. Düzə lişdə ki hə r bir sütun üçün hə r ə lifba hə rfinin sayı sayılır və ümumilə şdirilir. Cə mlə r sol ox boyunca bə zi tə msilçi ardıcıllığa nisbə tə n göstə rilir. Bu ardıcıllıq konsensus ardıcıllığı ola bilə r, lakin düzülmə də istə nilə n ardıcıllıq da ola bilə r.

Mə sə lə n, yuxarıdakı uyğunlaşma üçün:

```
>>> çap (msa)
4 sə tir və 7 sütunla düzülmə
ACTCCTA seq1
AAT-CTA seq2
CCTACT- seq3
TCTCCTC seq4
```

istifadə tə rə fi boyunca konsensus ardıcıllığı ilə PSSM alırıq

```
>>> my_pssm = summary_align.pos_specific_score_matrix(consensus, chars_to_ignore=["N"]) >>> print(my_pssm)

      ACT
X 2.0 1.0 1.0
C 1.0 3.0 0.0
T 0,0 0,0 4,0
X 1.0 2.0 0.0
C 0,0 4,0 0,0
T 0,0 0,0 4,0
X 2.0 1.0 0.0
<BLANKLINE>
```

burada PSSM-i hesablayarkə n hə r hansı N qeyri-müə yyə nlik qalıqlarına mə hə l qoymuruq.
Bununla bağlı iki qeyd edilmə lidir:

1. Ə lifbalara ciddi riayə t etmə k üçün siz yalnız PSSM-nin yuxarı hissə si boyunca düzülmə obyektinin ə lifbasında olan simvolları daxil edə bilə rsiniz. PSSM-in yuxarı oxu boyunca boşluqlar daxil edilmir.

2. Oxun sol tə rə fində nümayiş etdirilmə k üçün ötürülə n ardıcıllığın konsensus olması lazım deyil.
   Mə sə lə n, bu ox boyunca düzülmə də ikinci ardıcıllığı göstə rmə k istə yirsinizsə , aşağıdakıları etmə lisiniz:

```
>>> second_seq = msa[1] >>>
my_pssm = summary_align.pos_specific_score_matrix(ikinci_seq, chars_to_ignore=["N"]) >>> çap (my_pssm)

      ACT
A 2.0 1.0 1.0
A 1,0 3,0 0,0
T 0,0 0,0 4,0
- 1,0 2,0 0,0
C 0,0 4,0 0,0
T 0,0 0,0 4,0
A 2.0 1.0 0.0
<BLANKLINE>
```

Machine Translated by Google

Yuxarıdakı ə mr PSSM obyektini qaytarır. Siz öz_pssm[sequence_number][residue_count_name] kimi abunə olmaqla PSSM-nin istə nilə n elementinə daxil ola bilə rsiniz. Mə sə lə n, yuxarıdakı PSSM-nin ikinci elementində 'A' qalığının hesablarını ə ldə etmə k üçün aşağıdakıları edə rdiniz:

```
>>> çap (my_pssm[5]["T"]) 4.0
```

PSSM sinifinin strukturu inşallah hə m elementlə rə daxil olmağı, hə m də matrisin kifayə t qə də r çapını asanlaşdırır.

Alternativ olaraq, alignment atributundan istifadə etmə klə msa çox ardıcıl düzülmə obyektini yeni üslublu Alignment obyektinə çevirə bilə rsiniz (bax. bölmə 6.1) (bax bölmə 8.4):

```
>>> alignment = msa.alignment
```

Daha sonra siz Motif obyekti yarada bilə rsiniz (bax. bölmə 17.1):

```
>>> Bio.motifs -də n idxal Motif >>> motiv =
Motif("ACGT", düzülmə )
```

və hə r mövqedə hə r nukleotidin sayını ə ldə edin:

```
>>> saylar = motiv.sayılar
>>> çap (sayır) 0
                    1          2          3          4          5
6 A: 2,00 1,00 0,00 1,00 0,00 0,00 2,00 C: 1,00 3,00 0,00 2,00 4,00 0,00 1,00 G:
0,00 0,00 0,00 0,00 0,00 T 1.00 0.00 4.00 0.00 0.00 4.00 0.00


<BLANKLINE>
>>> çap (sayır["T"][5])
4.0
```

8.3.7 Mə lumat mə zmunu

Tə kamül mühafizə sinin potensial faydalı ölçüsü ardıcıllığın informasiya mə zmunudur.

Molekulyar bioloqlara yönə lmiş mə lumat nə zə riyyə sinə faydalı girişi http://www.lecb.ncifcrf.gov/~toms/paper/primer/ saytında tapa bilə rsiniz . Mə qsə dlə rimiz üçün biz konsensus ardıcıllığının mə lumat mə zmununa və ya konsensus ardıcıllığının bir hissə sinə baxacağıq. Aşağıdakı düsturdan istifadə edə rə k, müə yyə n bir sütundakı mə lumat mə zmununu çox ardıcıllıqla uyğunlaşdıraraq hesablayırıq:

$$ICj = \sum_{i=1}^{Na} Pij \log \frac{Pij}{Qi}$$

harada:

- ICj – düzülmə də j-ci sütun üçün mə lumat mə zmunu.
- Na – Ə lifbadakı hə rflə rin sayı.
- Pij – j-ci sütunda müə yyə n i hə rfinin tezliyi (yə ni, ə gə r G 6 də fə də n 3-də baş veribsə) hizalama sütunu, bu 0,5 olacaq)
- Qi – i hə rfinin gözlə nilə n tezliyi. Bu, isteğe bağlı bir arqumentdir, istifadə si istifadə çinin ixtiyarındadır. Varsayılan olaraq, zülal ə lifbası üçün avtomatik olaraq 0,05 = 1/20, nuklein turşusu ə lifbası üçün isə 0,25 = 1/4 olaraq tə yin edilir. Bu, ə vvə lcə də n paylanma ehtimalı olmadan mə lumat mə zmununu ə ldə etmə k üçündür. Prinsiplə ri qə bul edə rkə n və ya qeyri-standart ə lifbadan istifadə edə rkə n Qi üçün də yə rlə r tə qdim etmə lisiniz .

Machine Translated by Google

Yaxşı, indi Biopython-da hansı mə lumat mə zmununun hesablanması barə də bir fikrimiz var, gə lin baxaq hizalanmanın müə yyə n bir bölgə si üçün onu necə ə ldə etmə k olar.

Birincisi, biz alignment xülasə obyekti ə ldə etmə k üçün hizalamamızdan istifadə etmə liyik, bunun necə ə ldə edilə cə yinə dair tə limatlar üçün summary_align (bax. bölmə 8.3.4) adlandırılacağını güman edirik . Bu obyekti ə ldə etdikdə n sonra, bir bölgə üçün mə lumat mə zmununu hesablamaq bu qə də r asandır:

```
>>> e_freq_table = {"A": 0,3, "G": 0,2, "T": 0,3, "C": 0,2} >>> info_content =
summary_align.information_content(
...             2, 6, e_freq_table=e_freq_table, chars_to_ignore=["N"]
... )
>>> info_content # doctest:+ELLIPSIS
6.3910647...
```

İndi info_content gözlə nilə n tezliklə rlə ə laqə dar olaraq region üzrə [2:6] nisbi mə lumat mə zmununu ehtiva edə cə k.

Də yə r qaytarılması yuxarıdakı düsturda loqarifm bazası kimi 2 bazasından istifadə etmə klə hesablanır. Də yişdirə bilə rsiniz log_base parametrini istə diyiniz baza kimi ötürmə klə bunu:

```
>>> info_content = summary_align.information_content(
...             2, 6, e_freq_table=e_freq_table, log_base=10, chars_to_ignore=["N"]
... ) >>>
info_content # doctest:+ELLIPSIS 1.923902...
```

Defolt olaraq, sütunda 0 tezliyi olan nukleotid və ya amin turşusu qalıqları hə min sütun üçün nisbi mə lumat sütunu hesablanarkə n nə zə rə alınmır. Ə gə r bu istə diyiniz nə ticə deyilsə , bunun ə və zinə pseudo_count istifadə edə bilə rsiniz.

```
>>> info_content = summary_align.information_content( 2, 6,
...             e_freq_table=e_freq_table, chars_to_ignore=["N", "-"], pseudo_count=1
... )
>>> info_content # doctest:+ELLIPSIS
4.299651...
```

Bu halda j-ci sütunda müə yyə n i hə rfinin müşahidə olunan Pij tezliyi aşağıdakı kimi hesablanır:

$$Pij = \frac{nij + k \times Qi}{Nj + k}$$

harada:

- k – arqument kimi verdiyiniz psevdo hesab.
- k – arqument kimi verdiyiniz psevdo hesab.
- Qi – Yuxarıda tə svir olunduğu kimi i hə rfinin gözlə nilə n tezliyi.

Yaxşı, indi mə lumat mə zmununu hesablamağa hazırsınız. Bunu bə zi real hə yat problemlə rinə tə tbiq etmə yə cə hd etmə k istə yirsinizsə , ondan necə istifadə olunduğuna dair fikir ə ldə etmə k üçün mə lumat mə zmunu ilə bağlı ə də biyyata nə zə r salmaq çox güman ki, yaxşı olardı. Ümid edirik ki, qazma işlə riniz bu funksiyanın kodlaşdırılmasında edilə n sə hvlə ri aşkar etmə yə cə k!

Machine Translated by Google

8.4 Yeni üslubda Alignment obyektinin ə ldə edilmə si

Köhnə üslubdan yeni üslubda Alignment obyekti yaratmaq üçün alignment xassə sində n istifadə edin (bax. bölmə 6.1). MultipleSeqAlignment obyekti:

```
>>> növü (msa)
<sinfi 'Bio.Align.MultipleSeqAlignment'>
>>> çap (msa)
4 sə tir və 7 sütunla düzülmə
ACTCCTA seq1
AAT-CTA seq2
CCTACT- seq3
TCTCCTC seq4
>>> alignment = msa.alignment
>>> növü (düzlə şdirmə )
<sinfi 'Bio.Align.Alignment'>
>>> çap (düzlə şdirmə )
seq1                    0 ACTCCTA 7
seq2                    0 AAT-CTA 6
seq3                    0 CCTACT- 6
seq4                    0 TCTCCTC 7
<BLANKLINE>
```

Qeyd edə k ki, alignment xassə si uyğun olan yeni Alignment obyekti yaradır və qaytarır Alignment obyekti yaradılan zaman MultipleSeqAlignment obyektində saxlanılan mə lumat. Alignment xassə sinə zə ng etdikdə n sonra MultipleSeqAlignment-də edilə n hə r hansı də yişiklik, bu xüsusiyyə tə yayılmayacaq Hizalama obyekti. Bununla belə , yeni Alignment yaratmaq üçün ə lbə ttə ki, alignment xassə sinə yenidə n zə ng edə bilə rsiniz yenilə nmiş MultipleSeqAlignment obyektinə uyğun olan obyekt.

8.5 Çox ardıcıllıqdan ə və zedici matrisin hesablanması hizalanma

Hizalamadan öz ə və zetmə matrisini yarada bilə rsiniz. Bu nümunə də ə vvə lcə bir zülal oxuyacağıq protein.aln Clustalw faylından ardıcıl düzülmə (hə mçinin burada onlayn mövcuddur)

```
>>> Bio import AlignIO- dan
>>> fayl adı = "protein.aln"
>>> msa = AlignIO.read (fayl adı, "klastal")
```

Bölmə 8.6.1-də bunun edilmə si ilə bağlı daha çox mə lumat var.

Hizalanmanın ə və zetmə xüsusiyyə ti müxtə lif qalıqların neçə də fə ə və zlə ndiyini saxlayır bir-biri üçün:

```
>>> müşahidə olunan_tezliklə r = msa.ə və zetmə lə r
```

Mə sə lə ni daha oxunaqlı etmə k üçün biz yalnız qütb yüklü yan zə ncirlə ri olan amin turşularını seçə cə yik:

```
>>> müşahidə olunan_tezliklə r = müşahidə edilə n_tezliklə r.select("DEHKR")
>>> çap (müşahidə edilə n_tezliklə r)
          DE                H          K          R
D 2360.0 255.5              7.5        0,5 25,0
E 255,5 3305,0 16,5 27,0                          2.0
```

Machine Translated by Google

```
H        7,5 16,5 1235,0 16,0 8,5
K        0,5 27,0 16,0 3218,0 116,5
R 25.0                2.0          8,5 116,5 2079,0
<BLANKLINE>
```

Digə r amin turşuları üçün cə rgə lə r və sütunlar matrisdə n çıxarıldı.

Sonra matrisi normallaşdırırıq:

```
>>> np kimi idxal numpy
>>> müşahidə edilə n_tezliklə r /= np.sum(müşahidə edilə n_tezliklə r)
```

Sə tirlə r və ya sütunlar üzə rində cə mlə mə hə r bir qalığın nisbi tezliyini verir:

```
>>> qalıq_tezliklə ri = np.sum(müşahidə edilə n_tezliklə r, 0)
>>> çap (qalıq_tezliklə r.format("%.4f"))
D 0.2015
E 0.2743
H 0.0976
K 0.2569
R 0.1697
<BLANKLINE>
>>> np.sum(qalıq_tezliklə r)
1.0
```

Qalıq cütlə rinin gözlə nilə n tezliyi o zamandır

```
>>> gözlə nilə n_tezliklə r = np.dot(
...          qalıq_tezliklə ri[:, Yoxdur], qalıq_tezliklə ri[Yox, :]
... )
>>> çap (gözlə nilə n_tezliklə r.format("%.4f"))
          D          E          H          K          R
D 0,0406 0,0553 0,0197 0,0518 0,0342
E 0,0553 0,0752 0,0268 0,0705 0,0465
H 0,0197 0,0268 0,0095 0,0251 0,0166
K 0,0518 0,0705 0,0251 0,0660 0,0436
R 0,0342 0,0465 0,0166 0,0436 0,0288
<BLANKLINE>
```

Burada residue_frequencies[:, None] də yə rlə ri olan tə k sütundan ibarə t 2D massivi yaradır.
qalıq_tezliklə ri və qalıq_tezliklə ri[Yox, :] tə k cə rgə kimi bu də yə rlə ri olan 2D massivi.
Onların nöqtə mə hsulunu (daxili mə hsul) götürə rə k, hə r bir giriş aşağıdakılardan ibarə t olduğu gözlə nilə n tezliklə rin matrisini yaradır.
iki qalıq_tezlik də yə ri bir-biri ilə vurulur. Mə sə lə n, gözlə nilə n_tezliklə r['D', 'E']
qalıq_tezliklə rinə ['D'] * qalıq_tezliklə rinə ['E'] bə rabə rdir.

İndi müşahidə olunan tezliklə ri gözlə nilə n tezliklə rə bölmə k yolu ilə log-otds matrisini hesablaya bilə rik.
və loqarifmi götürün:

```
>>> m = np.log2(müşahidə edilə n_tezliklə r / gözlə nilə n_tezliklə r)
>>> çap (m)
          DE          H          K          R
D 2,1 -1,5 -5,1 -10,4 -4,2
E -1,5 1,7 -4,4 -5,1 -8,3
H -5,1 -4,4 3,3 -4,4 -4,7
K -10,4 -5,1 -4,4 1,9 -2,3
R -4,2 -8,3 -4,7 -2,3 2,5
<BLANKLINE>
```

Machine Translated by Google

Bu matris hizalamaları yerinə yetirə rkə n ə və zetmə matrisi kimi istifadə edilə bilə r. Mə sə lə n,

```
>>> Bio.Align -də n idxal PairwiseAligner >>> aligner =
PairwiseAligner() >>> aligner.substitution_matrix
= m >>> aligner.gap_score = -3.0 >>> alignments =
aligner.align("DEHEK", "DHHKK") >>>
çap(alignments[0])

hə də f                    0 DEHEK 5
                           0 |.|.| 5
sorğu                      0 DHHKK 5
<BLANKLINE>

>>> print("%.2f" % alignments.score) -2.18

>>> xal = m["D", "D"] + m["E", "H"] + m["H", "H"] + m["E", "K"] + m["K", "K"] >>> çap("%.2f" % xal) -2.18
```

8.6 Hizalama Alə tlə ri

Ardıcıllıqları uyğunlaşdırmaq üçün çoxlu alqoritmlə r var, hə m cütlü düzülmə lə r, hə m də çoxlu ardıcıl düzülmə lə r. Bu hesablamalar nisbə tə n yavaşdır və siz ümumiyyə tlə Python-da belə bir alqoritm yazmaq istə mə zsiniz. Cütlü düzülmə lə r üçün siz C dilində hə yata keçirilə n və buna görə də sürə tli olan Biopython-un PairwiseAligner-də n istifadə edə bilə rsiniz (Fə sil 7-ə baxın). Alternativ olaraq, Python-dan istifadə edə rə k xarici uyğunlaşdırma proqramını işlə də bilə rsiniz. Normalda siz:

1. Hizalanmamış ardıcıllıqlarınızın daxil faylını hazırlayın, adə tə n bu, FASTA faylı olacaq.
   Bio.SeqIO-dan istifadə edə rə k yaradın (5-ci Fə silə baxın).
2. Python-un alt proses modulundan istifadə edə rə k onun ə mrini yerinə yetirə rə k hizalama proqramını işə salın.
3. Tipik olaraq Bio.AlignIO-dan istifadə edə rə k, alə tdə n çıxışı, yə ni düzlə nmiş ardıcıllığınızı oxuyun (ə vvə lki bax. bu fə sildə).

Burada bu iş prosesinin bir neçə nümunə sini göstə rə cə yik.

8.6.1 ClustalW

ClustalW çox ardıcıllıqla düzülmə üçün mə şhur komanda xə tti alə tidir (ClustalX adlı qrafik interfeys də mövcuddur). ClustalW-ni Python daxilində istifadə etmə yə çalışmazdan ə vvə l, digə r seçimlə rlə tanış olmaq üçün ə vvə lcə komanda xə ttində ClustalW alə tini özünüz işə salmalısınız.

Ə n ə sas istifadə üçün sizə lazım olan tə k şey opuntia.fasta kimi FASTA giriş faylına sahib olmaqdır. (onlayn və ya Biopython mə nbə kodunun Doc/examples alt kataloqunda mövcuddur). Bu, yeddi tikanlı armud DNT ardıcıllığını (Opuntia kaktus ailə sində n) ehtiva edə n kiçik bir FASTA faylıdır. Defolt olaraq ClustalW daxil olan FASTA faylına, bu halda opuntia.aln və opuntia.dnd-ə ə saslanan adlarla uyğunlaşdırma və bə lə dçi ağac faylı yaradacaq, lakin siz bunu lə ğv edə və ya açıqlaya bilə rsiniz:

```
>>> idxal alt prosesi >>> cmd
= "clustalw2 -infile=opuntia.fasta" >>> nə ticə lə r =
subprocess.run(cmd, shell=Doğru, stdout=subprocess.PIPE, mə tn=Doğru)
```

Diqqə t yetirin ki, burada icra edilə bilə n adı clustalw2 kimi vermişik, bu, bizdə birinci versiyadan fə rqli fayl adına malik olan ikinci versiyanın quraşdırıldığını göstə rir (clustalw, standart). Xoşbə xtlikdə n hə r iki versiya komanda xə ttində eyni arqumentlə r də stini də stə klə yir (və hə qiqə tə n də funksional olaraq eyni olmalıdır).

Machine Translated by Google

Siz görə bilə rsiniz ki, sizdə ClustalW quraşdırılsa da, yuxarıdakı ə mr işlə mir –"ə mr tapılmadı" (xüsusilə Windows-da) haqqında mesaj ala bilə rsiniz. Bu, ClustalW icraedicinin PATH-də olmadığını göstə rir (mühit də yişə ni, axtarılacaq kataloqların siyahısı). Siz ya ClustalW alə tlə rinin surə tinizin yerini daxil etmə k üçün PATH parametrinizi yenilə yə bilə rsiniz (bunu necə edə cə yiniz OS-də n asılı olacaq) və ya sadə cə alə tin tam yolunu yazın. Unutmayın ki, Python-da \n və \t sə tirlə ri standart olaraq yeni sə tir və nişan kimi şə rh olunur –buna görə də biz bu şə kildə tə rcümə olunmayan xam sə tir üçün başlanğıcda "r" hə rfi qoyuruq. Bu, ümumiyyə tlə Windows stili fayl adını tə yin edə rkə n yaxşı tə crübə dir.

```
>>> import
ə mə liyyat sistemi >>> clustalw_exe = r"C:\Program Files\new clustal\clustalw2.exe" >>> assert
os.path.isfile(clustalw_exe), "Clustal W icra edilə bilə n yoxdur" >>> cmd = clustalw_exe + >>> nə ticə lə r =
subprocess.run(cmd, shells.PI=out(cmd, " -infile=opuntia.fasta"
shells.PI) mə tn=Doğru)
```

İndi, bu nöqtə də ə mr satırı vasitə lə rinin necə "işlə diyini" bilmə k kömə k edir. Komanda xə ttində alə t işlə tdiyiniz zaman o, tez-tez mə tn çıxışını birbaşa ekrana çap edə cə k. Bu mə tn standart çıxış (normal nə ticə lə r) və standart xə ta (sə hv mesajları və debug mesajları üçün) adlanan iki "boru" vasitə silə tutula və ya yönlə ndirilə bilə r. Alə tə daxil olan istə nilə n mə tn olan standart giriş də var. Bu adlar stdin, stdout və stderr kimi qısaldılır. Alə t başa çatdıqda, onun qaytarma kodu (tam ə də d) olur, bu, şə rti olaraq müvə ffə qiyyə t üçün sıfırdır, sıfırdan fə rqli qaytarma kodu isə xə tanın baş verdiyini göstə rir.

Yuxarıdakı ClustalW nümunə sində , ə mr satırında işlə dildikdə , bütün vacib çıxışlar birbaşa çıxış fayllarına yazılır. Siz gözlə diyiniz zaman ekrana çap edilə n hə r şey results.stdout və results.stderr-də , qaytarma kodu isə results.returncode-da saxlanılır.

Bizi maraqlandıran iki çıxış faylıdır, hizalama və bə lə dçi ağacı. Biz ClustalW-a hansı fayl adlarından istifadə edə cə yini demə dik, lakin o, giriş faylına ə sasə n adları seçir. Bu halda çıxış opuntia.aln faylında olmalıdır. İndiyə qə də r Bio.AlignIO-dan istifadə edə rə k düzülmə də necə oxumağı öyrə nə bilmə lisiniz:

```
>>> Bio import AlignIO- dan >>> align =
AlignIO.read("opuntia.aln", "clustal") >>> çap (align)

7 cə rgə və 906 sütun ilə düzülmə
TATACATTAAAGAAGGGGGATGCGGATAAATGGAAAGGCGAAAG...AGA gi|6273285|gb|AF191659.1|AF191
TATACATTAAAGAAGGGGGATGCGAGAGAGAAAAG... gi|6273284|gb|AF191658.1|AF191
TATACATTAAAGAAGGGGGATGCGGATAAATGGAAAGGCGAAAG...AGA gi|6273287|gb|AF191661.1|AF191
TATACATAAAAAGAAGGGGGATGCGGATAAATGGAAAGGCGAAAG...AGA gi|6273286|gb|AF191660.1|AF191
TATACATTAAAGGAGGGGATGCGGATAAATGGAAAGGCGAAAG...AGA gi|6273290|gb|AF191664.1|AF191
TATACATTAAAGGAGGGGATGCGGATAAATGGAAAGGCGAAAG...AGA gi|6273289|gb|AF191663.1|AF191
TATACATTAAAGGAGGGGATGCGGATAAATGGAAAGGCGAAAG...AGA gi|6273291|gb|AF191665.1|AF191
```

Ə gə r maraqlanırsınızsa (və bu, bu fə slin ə sas mə qsə dində n başqa), opuntia.dnd ClustalW faylı sadə cə standart Newick ağac faylıdır və Bio.Phylo bunları tə hlil edə bilə r:

```
>>> Bio import Phylo-dan >>> ağac =
Phylo.read ("opuntia.dnd", "newick")
>>> Phylo.draw_ascii(ağac)
                                                  _____ gi|6273291|gb|AF191665.1|AF191665 _____|  |
    | _____ gi|6273290|gb|AF191664.1|
  AF191664 |  |__|  |  |____ gi|6273289|gb|AF191663.1|AF191663 |
```

Machine Translated by Google

_ |_____ gi |6273287 |gb |AF191661.1 |AF191661

| |_____ gi |6273286 |gb |AF191660.1 |AF191660 | | __ gi |6273285 |gb |AF191659.1 |

AF191659 |___| | gi |6273284 |gb |AF191658.1 |AF191658 <BLANKLINE>

16-cı fə sil Biopython-un filogenetik ağaclara də stə yini daha də rində n ə hatə edir.

8.6.2 Ə ZƏ LƏ

MUSCLE, ClustalW-də n daha yeni bir çox ardıcıl düzülmə vasitə sidir. Ə vvə lki kimi, Python-dan işə salmağa cə hd etmə zdə n ə vvə l MUSCLE-dan ə mr satırından istifadə etmə yi tövsiyə edirik.

Ə n ə sas istifadə üçün sizə lazım olan tə k şey opuntia.fasta kimi FASTA giriş faylına sahib olmaqdır. (onlayn və ya Biopython mə nbə kodunun Doc/examples alt kataloqunda mövcuddur). Daha sonra siz MUSCLE-a bu FASTA faylını oxumağı və uyğunlaşdırmanı opuntia.txt adlı çıxış faylına yazmağı söylə yə bilə rsiniz:

```
>>> idxal alt prosesi >>> cmd
= "muscle -align opuntia.fasta -output opuntia.txt" >>> nə ticə lə r =
subprocess.run(cmd, shell=True, stdout=subprocess.PIPE, text=True)
```

MUSCLE uyğunlaşdırmanı FASTA faylı kimi çıxaracaq (boşluqlu ardıcıllıqlardan istifadə etmə klə). Bio.AlignIO modulu format="fasta" istifadə edə rə k bu düzülüşü oxuya bilir:

```
>>> Bio import AlignIO- dan >>> align
= AlignIO.read("opuntia.txt", "fasta") >>> çap (align)

7 cə rgə və 906 sütunla düzülmə
TATACATTAAAGGAGGGGATGCGGATAAATGGAAAGGCGAAAG...AGA gi |6273289 |gb |AF191663.1 |AF191663
TATACATTAAAGGAGGGAGAGAGATGCAGAGAGAGA... gi |6273291 |gb |AF191665.1 |AF191665
TATACATTAAAGGAGGGGATGCGGATAAATGGAAAGGCGAAAG...AGA gi |6273290 |gb |AF191664.1 |AF191664
TATACATTAAAGAAGGGGGATGCGGATAAATGGAAAGGCGAAAG...AGA gi |6273287 |gb |AF191661.1 |AF191661
TATACATAAAAAGAAGGGGATGCGGATAAATGGAAAGGCGAAAAG...AGA gi |6273286 |gb |AF191660.1 |AF191660
TATACATTAAAGAAGGGGGATGCGGATAAATGGAAAGGCGAAAG...AGA gi |6273285 |gb |AF191659.1 |AF191659
TATACATTAAAGAAGGGGGATGCGGATAAATGGAAAGGCGAAAG...AGA gi |6273284 |gb |AF191658.1 |AF191658
```

Siz hə mçinin digə r isteğe bağlı parametrlə ri tə yin edə bilə rsiniz; tə fə rrüatlar üçün MUSCLE-ın daxili yardımına baxın.

8.6.3 İğneyi və suyu EMBOSS

EMBOSSsuite Smith-Waterman alqoritminin yerli uyğunlaşdırılması və Needleman-Wunsch qlobal düzülmə si üçün su və iynə alə tlə rini ehtiva edir. Alə tlə r eyni stil interfeysini paylaşır, ona görə də ikisi arasında keçid mə nasızdır – biz burada sadə cə iynə də n istifadə edə cə yik.

Tutaq ki, siz FASTA formatında hazırlanmış iki ardıcıllıq arasında qlobal ikili düzülmə etmə k istə yirsiniz aşağıdakı kimi:

```
>HBA_HUMAN
MVLSPADKTNVKAAWGKVGAHAGEYGAEALERMFLSFPTTKTYFPHFDLSHGSAQVKGHG
KKVADALTNAVAHVDDMPNALSALSDLHAHKLRVDPVNFKLLSHCLLVTLAAHLPAEFTP
AVHASLDKFLASVSTVLTSKYR
```

alpha.faa faylında, ikincisi isə beta.faa faylında:

Machine Translated by Google

```
>HBB_HUMAN
MVHLTPEEKSAVTALWGKVNVDEVGGEALGRLLVVYPWTQRFFESFGDLSTPDAVMGNPK
VKAHGKKVLGAFSDGLAHLDNLKGTFATLSELHCDKLHVDPENFRLLGNVLVCVLAHHFG
KEFTPPVQAAYQKVVAGVANALAHKYH
```

Siz Doc/examples/ kataloqu altında Biopython mə nbə kodu ilə bu nümunə faylların surə tlə rini tapa bilə rsiniz.

Bu iki ardıcıllığı iynə də n istifadə edə rə k bir-birinə uyğunlaşdırmaq ə mri aşağıdakı kimidir:

```
iynə -outfile=needle.txt -asequence=alpha.faa -bsequence=beta.faa -gapopen=10 -gapextend=0.5
```

Nə üçün bunu ə mr sorğusunda ə l ilə işə salmağa cə hd etmirsiniz? Bunun ikili müqayisə apardığını görmə lisiniz və çıxışı needle.txt faylında qeyd edir (standart EMBOSS hizalama fayl formatında).

Sizdə EMBOSS quraşdırılmış olsa belə , bu ə mri işə salmaq işlə mə yə bilə r –"ə mr tapılmadı" (xüsusilə Windows-da) haqqında mesaj ala bilə rsiniz. Bu, yə qin ki, o demə kdir ki, EMBOSS alə tlə ri sizin PATH mühit də yişə ninizdə deyil. Siz ya PATH parametrinizi yenilə yə bilə rsiniz, ya da sadə cə alə tin tam yolundan istifadə edə bilə rsiniz, mə sə lə n:

```
C:\EMBOSS\needle.exe -outfile=needle.txt -asequence=alpha.faa -bsequence=beta.faa -gapopen=10 -gapexten
```

Sonra bu ə mri bizim üçün yerinə yetirmə k üçün Python-dan istifadə etmə k istə yirik. Yuxarıda izah edildiyi kimi, tam nə zarə t üçün Python-un daxili alt proses modulundan istifadə etmə yi tövsiyə edirik:

```
>>> idxal sistemi >>>
idxal alt prosesi >>> cmd =
"iynə -outfile=needle.txt -asequence=alpha.faa -bsequence=beta.faa -gapopen=10 -gapextend=0 >>> nə ticə lə r = subprocess.run(

...         cmd,
...         stdout=subprocess.PIPE,
...         stderr=subprocess.PIPE,
...         mə tn=Doğru,
...         shell=(sys, platforma != "win32"),
... )
>>> çap (results.stdout)

>>> çap (nə ticə lə r.stderr)
Needleman-Wunsch iki ardıcıllığın qlobal uyğunlaşması
```

Bundan sonra, bu fə sildə ə vvə llə r müzakirə edildiyi kimi Bio.AlignIO ilə çıxış faylını qabartma formatı kimi yüklə yə bilə rik:

```
>>> Bio import AlignIO- dan >>> align =
AlignIO.read("needle.txt", "qabartma") >>> çap (align)

2 sə tir və 149 sütunla düzülmə MV-
LSPADKTNVKAAWGKVGAHAGEYGAEALERMFLSFPTTKTY...KYR HBA_HUMAN
MVHLTPEEKSAVTALWGKV--NVDEVGGEALGRLLVVYPWTQRF...KYH HBB_HUMAN
```

Bu misalda biz EMBOSS-a çıxışı fayla yazmağı söylə dik, lakin siz ona çıxışı stdout-a yazmasını söylə yə bilə rsiniz (müvə qqə ti çıxış faylından xilas olmaq istə mirsinizsə faydalıdır –outfile=stdout arqumentində n istifadə edin):

```
>>> cmd = "iynə -outfile=stdout -asequence=alpha.faa -bsequence=beta.faa -gapopen=10 -gapextend=0.5" >>> uşaq = alt proses.Popen(

...         cmd,
```

Machine Translated by Google

```
...         stdout=subprocess.PIPE,
...         stderr=alt proses.BORU,
...         mə tn=Doğru,
...         shell=(sys.platform != "win32"),
... )
>>> align = AlignIO.read(child.stdout, "qabartma")
>>> çap (düzlə şdirmə )
2 sə tir və 149 sütunla düzülmə
MV-LSPADKTNVKAAWGKVGAHAGEYGAEALERMFLSFPTTKTY...KYR HBA_HUMAN
MVHLTPEEKSAVTALWGKV--NVDEVGGEALGRLLVVYPWTQRF...KYH HBB_HUMAN
```

Eynilə , stdin-də n daxil olan girişlə rdə n birini oxumaq mümkündür (mə sə lə n, asequence="stdin").

Bu, yalnız iynə və su ilə edə bilə cə yiniz şeylə rin sə thini cızdı. Bir faydalı hiylə budur ikinci faylda çoxlu ardıcıllıq ola bilə r (deyə k ki, beş) və sonra EMBOSS beş cüt düzülmə edə cə k.

Machine Translated by Google

Fə sil 9

Pairwise2 istifadə edə rə k cüt-cüt hizalamalar

Nə zə rə alın ki, Bio.pairwise2 buraxılış 1.80-də köhnə lmişdir. Alternativ olaraq, Bio.Align.PairwiseAligner istifadə etmə yi düşünün (7-ci Fə sildə tə svir edilmişdir).

Bio.pairwise2 EMBOSS- dan su (yerli) və iynə (qlobal) ilə eyni alqoritmlə ri ehtiva edir. suite (yuxarıya bax) və eyni nə ticə lə ri qaytarmalıdır. Pairwise2 modulu bu yaxınlarda sürə t və yaddaş istehlakı ilə bağlı bə zi optimallaşdırmalara mə ruz qalmışdır (Biopython versiyaları >1.67) belə liklə , qısa ardıcıllıqlar üçün (qlobal düzülmə lə r: ~2000 qalıq, yerli düzülmə lə r ~600 qalıq) EM22 vasitə silə su ə mr xə tti ilə zə ng etmə kdə n daha sürə tli (və ya eyni də rə cə də sürə tli) olur.

Tutaq ki, siz eyni iki hemoglobin ardıcıllığı arasında qlobal ikili uyğunlaşma etmə k istə yirsiniz. yuxarıda (HBA-HUMAN, HBB-HUMAN) alpha.faa və beta.faa-da saxlanılır:

```
>>> Bio import pairwise2 >>> from Bio
import SeqIO >>> seq1 =
SeqIO.read("alpha.faa", " fasta") >>> seq2 = SeqIO.read("beta.faa",
"fasta") >>> alignments = pairwise2.align.globalxx(seq1.seq,
seq1.seq)
```

Gördüyünüz kimi align.globalxx ilə alignment funksiyasını çağırırıq. Çə tin hissə funksiya adının son iki hə rfidir (burada: xx), bunlar matçlar (və uyğunsuzluqlar) və boşluqlar üçün xalların və cə zaların şifrə sini açmaq üçün istifadə olunur. İlk hə rf matç hesabını deşifrə edir, mə sə lə n, x o demə kdir ki, uyğunsuzluqların heç bir xə rci yoxdur, mə sə lə n, matç 1 sayılır. m ilə uyğunluqlar və ya uyğunsuzluqlar üçün ümumi də yə rlə r müə yyə n edilə bilə r (daha çox seçim üçün Biopython API-ə baxın). İkinci mə ktub boşluqların də yə rini deşifrə edir; x heç bir boşluq xə rclə rinin olmaması demə kdir, s boşluğun açılması və genişlə ndirilmə si üçün müxtə lif cə zalar tə yin edilə bilə r. Belə liklə , qlobalxx o demə kdir ki, yalnız hə r iki ardıcıllıq arasındakı uyğunluqlar sayılır.

Də yişə n düzülmə lə rimiz indi verilmiş şə rtlə r üçün eyni optimal hesaba malik düzülmə lə rin siyahısını (ə n azı bir) ehtiva edir. Bizim nümunə mizdə bu, 72 xalla 80 müxtə lif düzülmə dir (Bio.pairwise2 1000-ə qə də r düzülmə ni qaytaracaq). Bu uyğunlaşmalardan birinə nə zə r salın:

```
>>> lent (düzlə şdirmə )
80
>>> print(alignments[0]) # doctest:+ELLIPSIS Alignment(seqA='MV-
LSPADKTNV---KA--A-WGKVGAHAG...YR-', seqB='MVHL-----T--PEEKSAVTALWGKV----...YH', sc
```

Hə r bir düzülmə iki düzülmüş ardıcıllıqdan, hesabdan, düzülmə nin başlanğıc və son mövqelə rində n (qlobal düzülmə lə rdə başlanğıc hə mişə 0, son isə hizalanmanın uzunluğundan) ibarə t adlandırılmış də stdir. Bio.pairwise2 daha gözə l çap üçün format_alignment funksiyasına malikdir:

```
>>> print(pairwise2.format_alignment(*alignments[0])) # doctest:+ELLIPSIS MV-LSPADKTNV---KA--A-WGKVGAHAG---EY-GA-EALE-
RMFLSF----PTTK-TY--F...YR-
```

Machine Translated by Google

```
 | | | | MVHL---┼T--        |          | | | | |                    |     |     | | |      |     |             |         |       |...|
PEEKSAVTALWGKV-----NVDE-VG-GEAL-GR--L--LVVYP---WT-QRF...YH
    Xal = 72
<BLANKLINE>
```

Biopython 1.77-də n etibarə n tə lə b olunan parametrlə r açar sözlə rlə tə min edilə bilə r. Son nümunə indi ola bilə r belə də yazılsın:

```
>>> alignments = pairwise2.align.globalxx(sequenceA=seq1.seq, sequenceB=seq2.seq)
```

Daha yaxşı uyğunlaşmalar adə tə n boşluqların cə zalandırılması ilə ə ldə edilir: boşluğun açılması üçün daha yüksə k xə rclə r və aşağı xə rclə r mövcud boşluğu genişlə ndirmə k üçün. Amin turşusu ardıcıllığı üçün uyğunluq xalları adə tə n kimi matrislə rdə kodlanır PAM və ya BLOSUM. Belə liklə , BLOSUM62-də n istifadə etmə klə nümunə miz üçün daha mə nalı uyğunlaşma ə ldə edilə bilə r matris, 10 boşluq cə zası və 0,5 boşluq uzadılması cə zası ilə birlikdə (qlobaldan istifadə etmə klə):

```
>>> Bio- dan cüt-cüt idxal2
>>> Bio idxal SeqIO- dan
>>>- dan Bio.Align idxal ə və zedici_matrislə ri
>>> blosum62 = substitution_matrices.load("BLOSUM62")
>>> seq1 = SeqIO.read("alpha.faa", "fasta")
>>> seq2 = SeqIO.read("beta.faa", "fasta")
>>> alignments = pairwise2.align.globalds(seq1.seq, seq2.seq, blosum62, -10, -0.5)
>>> lent (düzlə şdirmə )
2
>>> çap (pairwise2.format_alignment(*alignments[0]))
MV-LSPADKTNVKAAWGKVGAHAGEYGAEALERMFLSFPTTKTY...KYR
 | | |.|..|..|.|.| | | | .....|.............|.......| |.
MVHLTPEEKSAVTALWGKV-NVDEVGGEALGRLLVVYPWTQRFF...KYH
    Bal=292.5
```

Bu uyğunlaşma ə vvə llə r eyni ardıcıllıqlardan istifadə edə rə k EMBOSS iynə si ilə ə ldə etdiyimiz eyni nə ticə yə malikdir və eyni parametrlə r.

Yerli düzülmə lə r align.localXX funksiyası ilə eyni şə kildə çağırılır, burada yenə XX iki demə kdir uyğunluq və boşluq funksiyaları üçün hə rf kodu:

```
>>> Bio- dan cüt-cüt idxal2
>>>- dan Bio.Align idxal ə və zedici_matrislə ri
>>> blosum62 = substitution_matrices.load("BLOSUM62")
>>> alignments = pairwise2.align.localds("LSPADKTNVKAA", "PEEKSAV", blosum62, -10, -1)
>>> çap (pairwise2.format_alignment(*alignments[0]))
3 PADKTNV
     |..|..|
1 PEEKSAV
    Xal=16
<BLANKLINE>
```

Son Biopython versiyalarında format_alignment yalnız yerli düzlə şdirmə nin düzlə nmiş hissə sini çap edə cə k (yuxarıdakı nümunə də göstə rildiyi kimi, 1 ə saslı notasiyada başlanğıc mövqelə ri ilə birlikdə). Ə gə r siz də ardıcıllığın düzülmə miş hissə lə ri ilə maraqlanırsınızsa, açar sözündə n istifadə edin-parametr full_sequences=Doğru:

```
>>> Bio- dan cüt-cüt idxal2
>>>- dan Bio.Align idxal ə və zedici_matrislə ri
```

Machine Translated by Google

```
>>> blosum62 = substitution_matrices.load("BLOSUM62") >>> alignments =
pairwise2.align.localds("LSPADKTNVKAA", "PEEKSAV", blosum62, -10, -1) >>>
print(pairwise2.format_alignment(*alignments[0] )=T_rusequence
LSPADKTNVKAA
    |..|..|
--PEEKSAV---
   Xal=16
<BLANKLINE>
```

Qeyd edə k ki, Smith & Waterman tə rə fində n müə yyə n edildiyi kimi yerli düzülmə lə r müsbə t nə ticə yə (>0) malik olmalıdır. Belə liklə , pairwise2 heç bir >0 xal ə ldə edilmə dikdə heç bir düzülmə qaytara bilmə z. Hə mçinin, pairwise2 hə r iki saytda sıfır xallı genişlə ndirmə lə rin ə lavə edilmə sinin nə ticə si olan düzülmə lə ri bildirmə yə cə k. Növbə ti misalda, serin/aspartik turşu (S/D) və lizin/asparagin (K/N) cütlə rinin hə r ikisinin uyğunluğu 0-dır. Gördüyünüz kimi, düzlə nmiş hissə uzadılmayıb:

```
>>> Bio- dan pairwise2 >>> Bio.Align idxal
ə və zetmə _matrislə rində n >>> blosum62 =
substitution_matrices.load("BLOSUM62") >>> alignments =
pairwise2.align.localds ("LSSPADKTNVKKAA", "DDPEEKSAVNN", blosum62, -10 , >>> - çap(pairwise2.format_alignment(*alignments[0]))

4 PADKTNV
    |..|..| 3
PEEKSAV
   Hesabı=16
<BLANKLINE>
```

Tam uyğunluq/uyğunsuzluq matrisini tə min etmə k ə və zinə , uyğunluq kodu m ümumi uyğunluq/uyğunsuzluq də yə rlə rini asanlıqla müə yyə n etmə yə imkan verir. Növbə ti misalda 5/-4 uyğunluq/uyğunsuzluq xalları və yerlilə rdə n istifadə etmə klə 2/0,5 boşluq cə zaları (açıq/uzatma) istifadə olunur:

```
>>> alignments = pairwise2.align.localms("AGAACT", "GAC", 5, -4, -2, -0.5) >>>
print(pairwise2.format_alignment(*alignments[0]))
2 GAAC
    | ||
1 G-AC
   Xal=13
<BLANKLINE>
```

Bio.pairwise2.align funksiyalarının bir faydalı açar söz arqumenti yalnız hesabdır. True olaraq tə yin edildikdə , o, yalnız ə n yaxşı düzülmə (lə r)in xalını qaytaracaq, lakin ə hə miyyə tli də rə cə də daha qısa müddə tdə . O, hə mçinin yaddaş xə tası qaldırılmadan ə vvə l daha uzun ardıcıllıqların uyğunlaşdırılmasına imkan verə cə k. Başqa bir faydalı açar söz arqumenti yalnız bir uyğunlaşmadır=Doğru, bu da müə yyə n sürə t artımı ilə nə ticə lə nə cə k.

Tə ə ssüf ki, Bio.pairwise2 Biopython-un çoxsaylı ardıcıl düzülmə obyektlə ri ilə işlə mir (hə lə). Bununla belə , modulun bir sıra maraqlı qabaqcıl xüsusiyyə tlə ri var: siz öz uyğunluq və boşluq funksiyalarınızı tə yin edə bilə rsiniz (affin loqarifmik boşluq xə rclə rini sınamaq istə yirsiniz?), boşluq cə zaları və son boşluq cə zaları hə r iki ardıcıllıq üçün fə rqli ola bilə r, ardıcıllıqlar siyahılar şə klində tə qdim edilə bilə r (birdə n çox simvol ilə kodlanmış qalıqlarınız varsa faydalıdır) bütün digə r funksiyaları yerinə yetirmə k üçün alə tlə rlə) və s.
Ə traflı mə lumat üçün Biopython API- də modul sə nə dlə rinə baxın .

Machine Translated by Google

Fə sil 10

PARTİ

Hey, hamı BLAST-ı sevir, elə deyilmi? Demə k istə yirə m ki, hey, sizin ardıcıllıqlarınızdan biri ilə mə lum dünyadaki bütün digə r ardıcıllıqlar arasında müqayisə aparmaq necə asanlaşa bilə r? Amma tə bii ki, bu bölmə BLAST-ın nə qə də r sə rin olması ilə bağlı deyil, çünki biz bunu artıq bilirik. Söhbə t BLAST ilə bağlı problemdə n gedir – böyük qaçışlarla yaradılan mə lumatların hə cmi ilə mə şğul olmaq və ümumilikdə BLAST ə mə liyyatlarını avtomatlaşdırmaq hə qiqə tə n çə tin ola bilə r.

Xoşbə xtlikdə n, Biopython xalqı bunu çox yaxşı bilir, ona görə də onlar BLAST ilə mübarizə aparmaq və işlə ri asanlaşdırmaq üçün çoxlu alə tlə r hazırlayıblar. Bu bölmə də bu vasitə lə rdə n necə istifadə etmə k və onlarla faydalı işlə r görmə k barə də ə traflı mə lumat verilir.

BLAST ilə mə şğul olmaq iki mə rhə lə yə bölünə bilə r, hə r ikisi də Biopython daxilində edilə bilə r. Birincisi, sorğu ardıcıllığı(lar) üçün BLAST-ı işə salmaq və müə yyə n bir nə ticə ə ldə etmə k. İkincisi, ə lavə tə hlil üçün Python-da BLAST çıxışının tə hlili.

BLAST ilə ilk tanışlığınız çox güman ki, NCBI veb-xidmə ti vasitə silə olub. Ə slində , bir neçə şə kildə tə snif edilə bilə n BLAST-ı idarə etmə yin bir çox yolu var. Ə n mühüm fə rq BLAST-ı yerli olaraq (öz maşınınızda) və BLAST-ı uzaqdan idarə etmə kdir (başqa bir maşında, adə tə n NCBI serverlə rində). Biz bu fə sli Python skripti daxilində n NCBI onlayn BLAST xidmə tinə müraciə t etmə klə başlayacağıq.

QEYD: Aşağıdakı 11-ci Fə sil Bio.SearchIO-nu tə svir edir. Biz bunu son nə ticə də köhnə Bio.Blast modulunu ə və z etmə k niyyə tində yik, çünki o, digə r ə laqə li ardıcıllıqla axtarış alə tlə ri ilə də işlə mə k üçün daha ümumi çə rçivə tə min edir. Bununla belə , hə lə lik siz NCBI BLAST ilə mə şğul olmaq üçün ya ondan, ya da köhnə Bio.Blast modulundan istifadə edə bilə rsiniz.

10.1 İnternet üzə rində n BLAST işlə tmə k

BLAST-ın onlayn versiyasını çağırmaq üçün Bio.Blast.NCBIWWW modulunda qblast() funksiyasından istifadə edirik. Bunun üç qeyri-opsional arqumenti var:

- Birinci arqument kiçik hə rf sə tir kimi axtarış üçün istifadə edilə cə k partlayış proqramıdır. Proqramların variantları və tə svirlə ri https://blast.ncbi.nlm.nih.gov/Blast.cgi ünvanında mövcuddur. Hazırda qblast yalnız blastn, blastp, blastx, tblast və tblastx ilə işlə yir.

- İkinci arqument axtarış edilə cə k verilə nlə r bazalarını tə yin edir. Yenə də bunun üçün seçimlə r BLAST ftp://ftp.ncbi.nlm.nih.gov/pub/factsheets/HowTo_BLASTGuide üçün NCBI Bə lə dçisində mövcuddur . pdf.

- Üçüncü arqument sorğu ardıcıllığını ehtiva edə n sə tirdir. Bu ya ardıcıllığın özü ola bilə r, fasta formatında ardıcıllıq və ya GI nömrə si kimi identifikator.

NCBI tə limatları, https://blast.ncbi.nlm.nih.gov/Blast.cgi?CMD=Web&PAGE_TYPE=BlastDocs&DOC_TYPE=DeveloperInfo dövlə t:

Machine Translated by Google

1. Server ilə hə r 10 saniyə də bir də fə də n çox ə laqə saxlamayın.

2. Hə r hansı bir RID üçün də qiqə də bir də fə də n çox sorğu keçirmə yin.

3. Problem olduqda NCBI sizinlə ə laqə saxlaya bilmə si üçün URL parametri e-poçtu və alə tində n istifadə edin.

4. Skriptlə ri hə ftə sonları və ya 50-də n çox axtarış olarsa, hə ftə içi Şə rq vaxtı ilə 21:00 və 5:00 arasında işlə din tə qdim olunacaq.

Üçüncü bə ndi yerinə yetirmə k üçün NCBIWWW.email də yişə nini tə yin etmə k olar.

```
>>> Bio.Blast - dan idxal NCBIWWW >>>
NCBIWWW.email = "ANOther@example.com"
```

Qblast funksiyası hə mçinin bir sıra digə r seçim arqumentlə rini də götürür ki, bunlar da ə sasə n analojidir BLAST veb sə hifə sində tə yin edə bilə cə yiniz müxtə lif parametrlə r. Onlardan bir neçə sini burada qeyd edə cə yik:

- Arqument url_base BLAST-ı internet üzə rində n idarə etmə k üçün ə sas URL-i tə yin edir. Varsayılan olaraq o, NCBI-yə qoşulur, lakin bundan buludda işlə yə n NCBI BLAST nümunə sinə qoşulmaq üçün istifadə edilə bilə r. Ə lavə mə lumat üçün qblast funksiyası üçün sə nə dlə rə müraciə t edin.
- Qblast funksiyası BLAST nə ticə lə rini müxtə lif formatlarda qaytara bilə r, siz isteğe bağlı format_type açar sözü ilə seçə bilə rsiniz: "HTML", "Mə tn", "ASN.1" və ya "XML". Aşağıdakı bölmə 10.3- də tə svir edilə n tə hlilçi tə rə fində n gözlə nilə n format olduğu üçün defolt "XML"dir .
- Gözlə mə arqumenti gözlə mə və ya e-də yə r hə ddini tə yin edir.

Ə lavə BLAST arqumentlə ri haqqında ə traflı mə lumat üçün biz sizə NCBI-nin öz sə nə dlə rinə istinad edirik və ya Biopython-da qurulmuşdur:

```
>>> Bio.Blast - dan idxal NCBIWWW >>>
yardım(NCBIWWW.qblast)
```

Qeyd edə k ki, NCBI BLAST veb-saytındakı standart parametrlə r QBLAST-dakı standart parametrlə rlə tamamilə eyni deyil. Fə rqli nə ticə lə r ə ldə etsə niz, parametrlə ri yoxlamalı olacaqsınız (mə sə lə n, gözlə nilə n də yə r hə ddi və boşluq də yə rlə ri).

Mə sə lə n, bir nukleotid ardıcıllığınız varsa, nukleotid verilə nlə r bazasına (nt) qarşı axtarmaq istə yirsiniz. BLASTN istifadə edə rə k və sorğu ardıcıllığınızın GI nömrə sini bilirsinizsə , istifadə edə bilə rsiniz:

```
>>> Bio.Blast - dan idxal NCBIWWW >>> result_handle
= NCBIWWW.qblast("blastn", "nt", "8332116")
```

Alternativ olaraq, sorğu ardıcıllığımız artıq FASTA formatlı fayldadırsa, sadə cə olaraq faylı açıb bu qeydi sə tir kimi oxumalı və ondan sorğu arqumenti kimi istifadə etmə liyik:

```
>>> Bio.Blast - dan idxal NCBIWWW >>> fasta_string
= open("m_cold.fasta").read() >>> result_handle = NCBIWWW.qblast("blastn",
"nt", fasta_string)
```

Biz də FASTA faylını SeqRecord kimi oxuya bilə rdik və yalnız ardıcıllığın özünü tə qdim edə bilə rdik:

```
>>> Bio.Blast - dan idxal NCBIWWW >>> Bio import
SeqIO-dan >>> rekord = SeqIO.read
("m_cold.fasta", format="fasta") >>> result_handle = NCBIWWW.qblast("blastn", "nt",
record.seq)
```

Machine Translated by Google

Sadə cə ardıcıllığı tə min etmə k o demə kdir ki, BLAST avtomatik olaraq ardıcıllığınıza identifikator tə yin edə cə k. Siz FASTA sə trini (mövcud identifikatoru ehtiva edə cə k) yaratmaq üçün SeqRecord obyektinin format metodundan istifadə etmə yi üstün tuta bilə rsiniz:

```
>>> Bio.Blast - dan idxal NCBIWWW >>> Bio
import SeqIO- dan >>> rekord =
SeqIO.read("m_cold.fasta", format="fasta") >>> result_handle =
NCBIWWW.qblast("blastn", "nt", record.format("fasta"))
```

Bio.SeqIO-dan istifadə edə rə k çıxara bilə cə yiniz FASTA olmayan fayl formatında ardıcıllıq(lar)ınız varsa, bu yanaşma daha mə na kə sb edir (bax. Fə sil 5).

Qblast() funksiyasına verdiyiniz arqumentlə rdə n asılı olmayaraq, nə ticə lə rinizi tutacaq obyektində geri almalısınız (standart olaraq XML formatında). Növbə ti addım XML çıxışını axtarış nə ticə lə rini tə msil edə n Python obyektlə rində tə hlil etmə k olacaq (Bölmə 10.3), lakin siz ə vvə lcə çıxış faylının yerli surə tini saxlamaq istə yə bilə rsiniz. Mə n bunu xüsusilə BLAST nə ticə lə rində n mə lumat çıxaran kodumu sazlayarkə n faydalı hesab edirə m (çünki onlayn axtarışın tə krar icrası yavaş olur və NCBI kompüter vaxtını sə rf edir).

Yalnız BLAST çıxışını oxumaq üçün result_handle.read() funksiyasından istifadə edə bildiyimiz üçün bir az diqqə tli olmalıyıq. bir də fə –result_handle.read() çağırmaq yenidə n boş sə tir qaytarır.

```
>>> out_handle kimi open("my_blast.xml", "w") ilə :
...         out_handle.write(result_handle.read())
...
>>> result_handle.close()
```

Bunu etdikdə n sonra nə ticə lə r my_blast.xml faylındadır və orijinal tutacaqda bütün mə lumatlar çıxarılıb (biz onu bağladıq). Bununla belə , BLAST analizatorunun tə hlil funksiyası (10.3-də tə svir edilmişdir) fayl tutacağına bə nzə r bir obyekti götürür, ona görə də biz sadə cə olaraq daxil etmə k üçün yadda saxlanılan faylı aça bilə rik:

```
>>> result_handle = open("my_blast.xml")
```

İndi BLAST nə ticə lə rini yenidə n tutacaqda ə ldə etdikdə n sonra biz onlarla nə sə etmə yə hazırıq, ona görə də bu, bizi birbaşa tə hlil bölmə sinə aparır (aşağıda Bölmə 10.3- ə baxın). İndi bunun üçün irə lilə mə k istə yə bilə rsiniz. . . .

10.2 BLAST-ın yerli olaraq işə salınması

10.2.1 Giriş

BLAST-ın yerli olaraq (internet üzə rində n fə rqli olaraq, Bölmə 10.1-ə baxın) ə n azı iki ə sas üstünlüyü var:

- Yerli BLAST internet üzə rində n BLAST-dan daha sürə tli ola bilə r;
- Local BLAST sizə qarşı ardıcıllıqları axtarmaq üçün öz verilə nlə r bazanızı yaratmağa imkan verir.

Mülkiyyə t və ya də rc edilmə miş ardıcıllıq mə lumatları ilə mə şğul olmaq BLAST-ı yerli olaraq işə salmaq üçün başqa bir sə bə b ola bilə r. Ardıcıllıqları yenidə n bölüşdürmə yə icazə verilmə yə bilə r, ona görə də onları BLAST sorğusu kimi NCBI-yə tə qdim etmə k seçim olmayacaq.

Tə ə ssüf ki, bə zi böyük çatışmazlıqlar da var - bütün bitlə ri quraşdırmaq və düzgün quraşdırmaq müə yyə n sə y tə lə b edir:

- Local BLAST ə mr xə tti alə tlə rinin quraşdırılmasını tə lə b edir.
- Yerli BLAST (böyük) BLAST verilə nlə r bazalarının qurulmasını (və potensial olaraq yenilə nmə sini) tə lə b edir.

Mə sə lə lə ri daha da çaşdırmaq üçün bir neçə fə rqli BLAST paketi mövcuddur və BLAT kimi imitasiya BLAST çıxış faylları yarada bilə n digə r alə tlə r də mövcuddur.

Machine Translated by Google

10.2.2 Müstə qil NCBI BLAST+

"Yeni" NCBI BLAST+ suite 2009-cu ildə buraxıldı. Bu, köhnə NCBI "miras" BLAST paketini ə və z edir (aşağıya bax).

Bu bölmə Python daxilində bu alə tlə ri necə istifadə edə cə yinizi qısaca göstə rə cə k. Ə gə r siz artıq Bölmə 8.6-dakı düzlə şdirmə alə ti nümunə lə rini oxumusunuzsa və ya sınamısınızsa, bunların hamısı olduqca sadə görünmə lidir. Birincisi, biz bir komanda xə tti sə trini qururuq (müstə qil BLAST-ı ə l ilə işlə dirsinizsə , ə mr satırı sorğusunda yazacağınız kimi). Sonra bu ə mri Python daxilində n yerinə yetirə bilə rik.

Mə sə lə n, gen nukleotid ardıcıllığının FASTA faylını götürə rə k, artıq olmayan (NR) protein verilə nlə r bazasına qarşı BLASTX (tə rcümə) axtarışını hə yata keçirmə k istə yə bilə rsiniz. Siz (və ya sistem administratorunuz) NR verilə nlə r bazasını endirdiyinizi və quraşdırdığınızı fə rz etsə k, aşağıdakıları işlə də bilə rsiniz:

```
$ blastx -query opuntia.fasta -db nr -out opuntia.xml -qiymə tlə ndirmə 0.001 -outfmt 5
```

Bu, BLASTX-i NR verilə nlə r bazasına qarşı 0,001 gözlə nti kə sim də yə rində n istifadə edə rə k işlə tmə li və göstə rilə n fayla XML çıxışı çıxarmalıdır (bunu sonra tə hlil edə bilə rik). Mə nim kompüterimdə bu, tə xminə n altı də qiqə çə kir - nə ticə ni faylda saxlamaq üçün yaxşı sə bə bdir ki, lazım olduqda istə nilə n tə hlili tə krarlayasınız.

Python daxilində komanda xə tti sə tirini qurmaq üçün alt proses modulundan istifadə edə və onu işə sala bilə rik:

```
>>> idxal alt prosesi >>> cmd = "blastx
-query opuntia.fasta -db nr -out opuntia.xml" >>> cmd +=
                        -qiymə tlə ndirmə 0.001 -outfmt 5"
>>> subprocess.run(cmd, shell=Doğru)
```

Bu nümunə də BLASTX-də n terminala heç bir çıxış olmamalıdır. Yoxlamaq istə yə bilə rsiniz opuntia.xml çıxış faylı yaradılmışdır.

Də rslikdə ki ə vvə lki nümunə lə rdə n xatırladığınız kimi, opuntia.fasta yeddi ardıcıllığı ehtiva edir, ona görə də BLAST XML çıxışında çoxlu nə ticə lə r olmalıdır. Buna görə də aşağıda Bölmə 10.3-də tə svir olunduğu kimi onu tə hlil etmə k üçün Bio.Blast.NCBIXML.parse() istifadə edin.

10.2.3 BLAST-ın digə r versiyaları

NCBI BLAST+ (C++ dilində yazılmışdır) ilk də fə 2009-cu ildə orijinal NCBI “mirası” ə və z etmə k üçün buraxılmışdır. Artıq yenilə nmə yə n BLAST (C dilində yazılmışdır). Çoxlu də yişikliklə r oldu – köhnə versiyada çoxlu müxtə lif BLAST axtarış növlə rini ə hatə edə n (indi BLAST+-da ayrı-ayrı ə mrlə rdir) bir ə sas ə mr xə tti alə ti blastall var idi və bütün ə mr xə tti seçimlə rinin adı də yişdirildi. NCBI "miras" BLAST alə tlə ri üçün Biopython sarğıları köhnə lib və gə lə cə k buraxılışda silinə cə k.

Qarışıqlığın qarşısını almaq üçün bu də rslikdə Biopython-dan bu köhnə alə tlə ri çağırmağı ə hatə etmirik.

Vaşinqton Universitetinin BLAST ilə də rastlaşa bilə rsiniz (WU-BLAST) və onun varisi, Advanced Biocomputing BLAST (AB-BLAST, 2009-cu ildə buraxılmışdır, pulsuz/açıq mə nbə deyil). Bu paketlə rə NCBI “mirası”ndan blastall-ı tə qlid edə n wu-blastall və ab-blastall komanda xə tti alə tlə ri daxildir.

BLAST də sti. Biopython hazırda bu alə tlə ri çağırmaq üçün sarğı tə min etmir, lakin onlardan istə nilə n NCBI uyğun çıxışı tə hlil edə bilmə lidir.

10.3 BLAST çıxışının tə hlili

Yuxarıda qeyd edildiyi kimi, BLAST XML, HTML və düz mə tn kimi müxtə lif formatlarda çıxış yarada bilə r. Ə vvə lcə Biopython-da BLAST düz mə tn və HTML çıxışı üçün tə hliledicilə r var idi, çünki bunlar o zaman tə klif edilə n yeganə çıxış formatları idi. Tə ə ssüf ki, bu formatlarda BLAST çıxışı hə r də fə Biopython analizatorlarını pozaraq də yişmə yə davam edirdi. Bizim HTML BLAST analizatorumuz silindi, köhnə lmiş adi mə tn BLAST tə hlili isə indi yalnız Bio.SearchIO vasitə silə ə lçatandır. Onu öz riskinizlə istifadə edin, hansı BLAST versiyasından istifadə etdiyinizdə n asılı olaraq işlə yə bilə r və ya olmaya bilə r.

Machine Translated by Google

BLAST-da də yişikliklə rlə ayaqlaşmaq, xüsusə n də müxtə lif BLAST versiyaları ilə işlə yə n istifadə çilə rlə ümidsiz bir cə hdə çevrildiyinə görə , biz indi BLAST-ın son versiyaları tə rə fində n yaradıla bilə n çıxışı XML formatında tə hlil etmə yi tövsiyə edirik. Yalnız XML çıxışı düz mə tn və HTML çıxışından daha sabit deyil, hə m də avtomatik tə hlil etmə k çox asandır və Biopython-u tamamilə daha sabit edir.

XML formatında BLAST çıxışını müxtə lif yollarla ə ldə edə bilə rsiniz. Çıxışın XML formatında olması şə rti ilə tə hlilçi üçün onun necə yaradıldığı önə mli deyil.

- Bölmə 10.1-də tə svir olunduğu kimi, BLAST-ı internet üzə rində n idarə etmə k üçün Biopython-dan istifadə edə bilə rsiniz .
- Bölmə 10.2-də tə svir olunduğu kimi BLAST-ı yerli olaraq işə salmaq üçün Biopython-dan istifadə edə bilə rsiniz .
- Siz veb brauzeriniz vasitə silə NCBI saytında özünüz BLAST axtarışını edə və sonra nə ticə lə ri saxlaya bilə rsiniz. Nə ticə lə ri almaq üçün format kimi XML-i seçmə lisiniz və ə ldə etdiyiniz son BLAST sə hifə sini (bilirsiniz, bütün maraqlı nə ticə lə rin olduğu sə hifə !) faylda saxlamalısınız.
- Siz hə mçinin Biopython istifadə etmə də n BLAST-ı yerli olaraq işlə də və nə ticə ni faylda saxlaya bilə rsiniz. Yenə də nə ticə lə ri almaq üçün format kimi XML seçmə lisiniz.

Ə hə miyyə tli mə qam ondan ibarə tdir ki, mə lumatları tə hlil etmə k üçün onu götürmə k üçün Biopython skriptlə rində n istifadə etmə li deyilsiniz. Bu yollardan birində işlə r görsə niz, nə ticə lə rə diqqə t yetirmə lisiniz. Python-da tutacaq hə r hansı bir mə lumat mə nbə yinə girişi tə svir etmə yin gözə l ümumi üsuludur ki, mə lumat read() və readline() funksiyalarından istifadə edilə rə k götürülə bilsin (bax: Bölmə 25.1).

Skript vasitə silə BLAST ilə qarşılıqlı ə laqə də olmaq üçün yuxarıdakı koda ə mə l etmisinizsə , onda sizdə artıq result_handle, BLAST nə ticə lə rinin idarə edicisi var. Mə sə lə n, onlayn axtarış etmə k üçün GI nömrə sində n istifadə etmə klə :

```
>>> Bio.Blast - dan idxal NCBIWWW >>>
result_handle = NCBIWWW.qblast("blastn", "nt", "8332116")
```

Ə və zində BLAST-ı başqa yolla işlə tmisinizsə və faylda BLAST çıxışı (XML formatında) varsa my_blast.xml, sizə lazım olan tə k şey oxumaq üçün faylı açmaqdır:

```
>>> result_handle = open("my_blast.xml")
```

İndi ə limizdə bir qolu var, biz çıxışı tə hlil etmə yə hazırıq. Onu tə hlil etmə k üçün kod hə qiqə tə n çox kiçikdir. Tə k BLAST nə ticə sini gözlə yirsinizsə (yə ni, tə k sorğudan istifadə etmisiniz):

```
>>> Bio.Blast - dan idxal NCBIXML >>>
blast_record = NCBIXML.read(result_handle)
```

və ya çoxlu nə ticə niz varsa (yə ni, çoxlu sorğu ardıcıllığı):

```
>>> Bio.Blast - dan idxal NCBIXML >>>
blast_records = NCBIXML.parse(result_handle)
```

Bio.SeqIO və Bio.Align kimi (5 və 6-cı fə sillə rə bax) bizdə bir cüt daxiletmə funksiyası var, oxumaq və tə hlil etmə k, burada oxumaq tam olaraq bir obyektə malik olduqda, tə hlil isə çoxlu obyektlə rə malik olduğunuz zaman üçün iteratordur – lakin SeqRecord və ya MultipleSeqAlignment obyektlə rini ə ldə etmə k ə və zinə biz BLAST qeyd obyektlə rini ə ldə edirik.

BLAST faylının minlə rlə nə ticə də n ibarə t böyük ola bilə cə yi və ziyyə ti idarə etmə k üçün NCBIXML.parse() iteratoru qaytarır. Sadə ingilis dilində , iterator hə r BLAST axtarış nə ticə si üçün BLAST qeydlə rini bir-bir ə ldə edə rə k BLAST çıxışında addım atmağa imkan verir:

```
>>> Bio.Blast - dan idxal NCBIXML >>>
blast_records = NCBIXML.parse(result_handle) >>> blast_record =
next(blast_records)
```

Machine Translated by Google

```
# ... partlayış_rekordu ilə bir şey edin >>> partlayış_rekordu
= sonrakı (blast_rekordları) # ... partlayış_rekordu ilə bir şey
edin >>> blast_record = sonrakı (partlayış_rekordları) # ...
partlayış_rekordu ilə bir şey edin >>> partlayış_rekordu =
sonrakı (blast_rekordları)

Traceback (ə n son zə ng):
    Fayl "<stdin>", sə tir 1, <modul>-da
İterasiyanı dayandırın
# Ə lavə qeyd yoxdur
```

Və ya for-loop istifadə edə bilə rsiniz:

```
>>> blast_records -da blast_record üçün :
...             pass # blast_record ilə bir şey edin
...
```

Nə zə rə alın ki, siz BLAST qeydlə rində n yalnız bir də fə keçə bilə rsiniz. Adə tə n, hə r BLAST qeydində n sizi maraqlandıran mə lumatı saxlayardınız. Ə gə r siz bütün qaytarılmış BLAST qeydlə rini saxlamaq istə yirsinizsə , iteratoru siyahıya çevirə bilə rsiniz:

```
>>> blast_records = siyahı (blast_records)
```

İndi siz hə mişə ki kimi siyahıdakı hə r BLAST qeydinə indekslə daxil ola bilə rsiniz. Ə gə r BLAST faylınız böyük olsa da, onların hamısını siyahıda saxlamağa çalışarkə n yaddaş problemlə ri ilə üzlə şə bilə rsiniz.

Adə tə n, hə r də fə bir BLAST axtarışı hə yata keçirə cə ksiniz. Sonra, sizə lazım olan tə k şey birincini götürmə kdir (və yalnız) blast_records-da BLAST rekordu:

```
>>> Bio.Blast - dan idxal NCBIXML >>> blast_records
= NCBIXML.parse(result_handle) >>> blast_record = next(blast_records)
```

və ya daha zə rif:

```
>>> Bio.Blast - dan idxal NCBIXML >>> blast_record
= NCBIXML.read(result_handle)
```

Güman edirə m ki, indi siz BLAST rekordunda nə olduğunu merak edirsiniz.

10.4 BLAST rekord sinfi

BLAST Record, BLAST çıxışından çıxarmaq istə diyiniz hə r şeyi ehtiva edir. Hal-hazırda biz BLAST hesabatından bə zi mə lumatları necə ə ldə etmə yin bir nümunə sini göstə rə cə yik, lakin burada tə svir olunmayan xüsusi bir şey istə yirsinizsə , qeyd sinfi haqqında mə lumatı ə traflı nə zə rdə n keçirin və koda və ya avtomatik yaradılan sə nə dlə rə nə zə r salın – sə nə d sə tirlə rində hə r bir mə lumatda nə yin saxlandığı barə də çoxlu yaxşı mə lumatlar var.

Nümunə mizə davam etmə k üçün partlayış hesabatımızdakı bütün hitlə r haqqında bə zi xülasə mə lumatları çap edə k xüsusi hə dddə n böyükdür. Aşağıdakı kod bunu edir:

```
>>> E_VALUE_THRESH = 0,04

>>> blast_record.alignments- də hizalama üçün : alignment.hsps -də
...             hsp üçün : ə gə r hsp.expect <
...                     E_VALUE_THRESH:
```

Machine Translated by Google

```
...                     print("****Alignment****")
...                     print("sequence:", alignment.title) print("length:",
...                     alignment.length) print("e value:", hsp.expect)
...                     print(hsp.query[0:75] + "...")
...                     print(hsp.match[0:75] + "...".) print(75) " s..."
...
...
...
```

Bu, aşağıdakı kimi xülasə hesabatları çap edə cə k:

****Alignment**** ardıcıllığı:

>gb|AF283004.1|AF283004 Arabidopsis thaliana soyuq uyğunlaşma proteini WCOR413 kimi zülal alfa forması mRNA, tam CD uzunluğu: 783 e də yə ri: 0,034

tacttgttgatattggatcgaacaaactggagaaccaacatgctcacgtcacttttagtcccttacatattcctc... |||||||||| | ||||||||||||| || ||||
|| || |||||||| |||||| | | |||||||| ||| ||... tacttgttggtgttggatcgaaccaattggaagacgaatatgctcacatcacttctcattccttacatcttcttc...

Ə sasə n, siz onu tə hlil etdikdə n sonra BLAST hesabatındakı mə lumatla istə diyiniz hə r şeyi edə bilə rsiniz. Bu, ə lbə ttə ki, ondan nə üçün istifadə etmə k istə diyinizdə n asılı olacaq, lakin ümid edirə m ki, bu sizə lazım olanı etmə yə başlamağa kömə k edə cə k!

BLAST hesabatından mə lumat çıxarmaq üçün vacib olan mə qam mə lumatın saxlandığı obyektlə rin növüdür. Biopython-da tə hlilçilə r tə hlil etdiyinizdə n asılı olaraq Blast və ya PSIBlast Record obyektlə rini qaytarır. Bu obyektlə r Bio.Blast.Record-da müə yyə n edilib və kifayə t qə də r tamdır.

Şə killə r 10.1 və 10.2 və Blast və PSIBlast rekord siniflə ri üçün UML sinif diaqramlarında mə nim cə hdlə rimdir. PSIBlast qeyd obyekti oxşardır, lakin PSIBlast-ın iterasiya addımlarında istifadə olunan raundları də stə klə yir.

Ə gə r UML-də yaxşısınızsa və edilə bilə cə k sə hvlə ri/tə kmillə şdirmə lə ri görürsünüzsə , zə hmə t olmasa mə nə bildirin.

10.5 PSI-BLAST ilə işlə mə k

Siz PSI-BLAST-ın müstə qil versiyasını (keçmiş NCBI komanda xə tti alə ti blastpgp və ya onun ə və zedici psiblastı) birbaşa komanda xə ttində n və ya python-un alt proses modulundan istifadə edə rə k işlə də bilə rsiniz.

Yazı zamanı NCBI internet vasitə silə PSI-BLAST axtarışı aparan alə tlə ri də stə klə mir.

Qeyd edə k ki, Bio.Blast.NCBIXML analizatoru PSI-BLAST-ın cari versiyalarından XML çıxışını oxuya bilə r, lakin hə r iterasiyada hansı ardıcıllığın yeni və ya tə krar istifadə edildiyi kimi mə lumat XML faylında yoxdur.

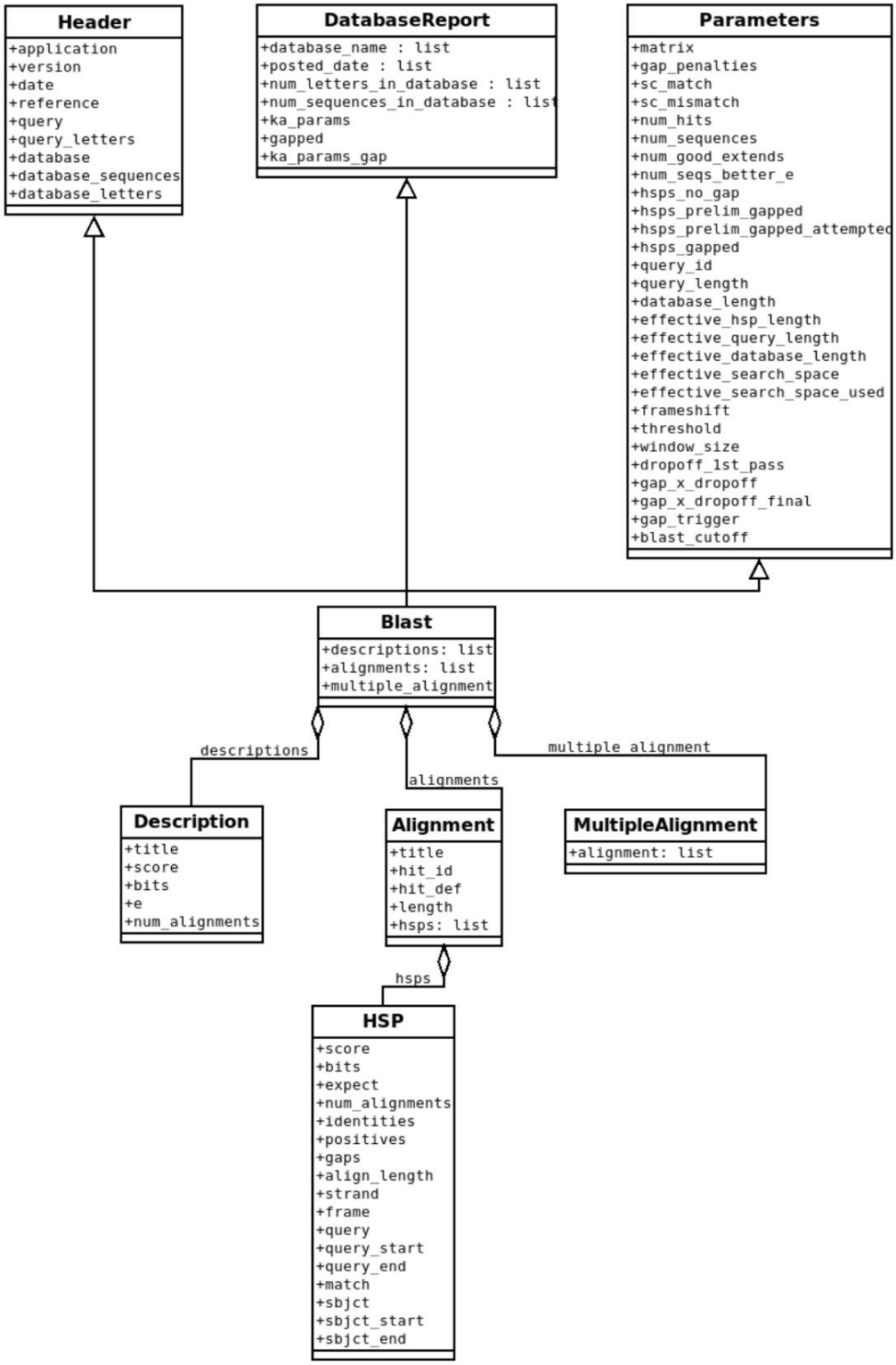
10.6 RPS-BLAST ilə işlə mə k

Siz RPS-BLAST-ın müstə qil versiyasını (istə r köhnə NCBI komanda xə tti alə ti rpsblast, ya da onun eyni adla ə və zlə nmə si) birbaşa komanda xə ttində n və ya python-un alt proses modulundan istifadə edə rə k işlə də bilə rsiniz.

Yazı zamanı NCBI internet vasitə silə RPS-BLAST axtarışı aparan alə tlə ri də stə klə mir.

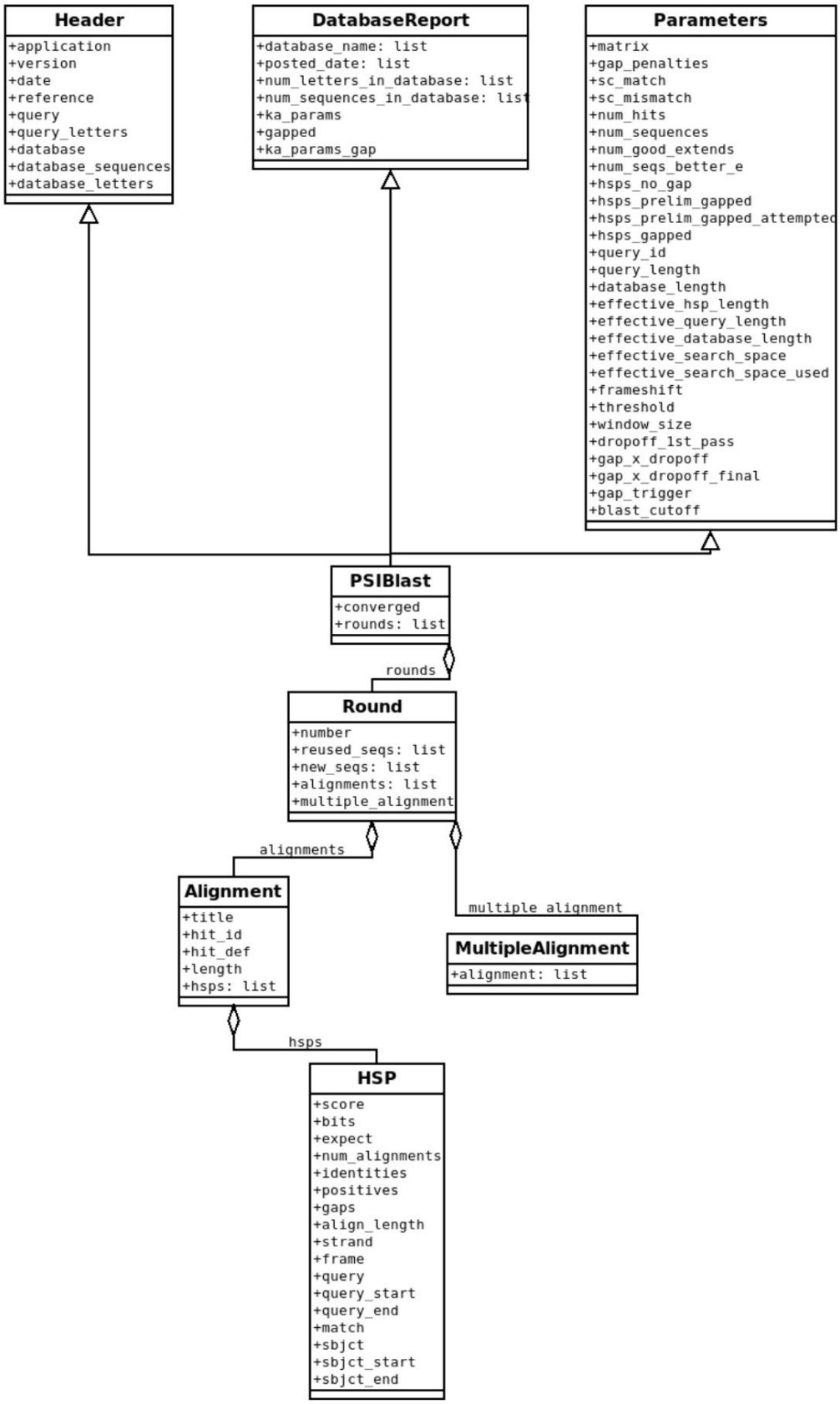
RPS-BLAST-ın cari versiyalarından XML çıxışını oxumaq üçün Bio.Blast.NCBIXML analizatorundan istifadə edə bilə rsiniz.

Machine Translated by Google



Şə kil 10.1: BLAST hesabatındakı bütün mə lumatları ə ks etdirə n Blast Record sinfi üçün sinif diaqramı

Machine Translated by Google



Şə kil 10.2: PSIBlast Record sinfi üçün sinif diaqramı.

Machine Translated by Google

Fə sil 11

BLAST və digə r ardıcıl axtarış vasitə lə ri

Bioloji ardıcıllığın identifikasiyası bioinformatikanın tə rkib hissə sidir. Bunun üçün hə r birinin öz alqoritmlə ri və yanaşmaları olan bir neçə alə t mövcuddur, mə sə lə n, BLAST (bə lkə də ə n populyar), FASTA, HMMER və daha çox. Ümumiyyə tlə , bu alə tlə r adə tə n potensial uyğunluqların verilə nlə r bazasını axtarmaq üçün ardıcıllığınızı istifadə edir.
Mə lum ardıcıllıqların sayının artması ilə (buna görə də potensial uyğunluqların sayının artması) nə ticə lə ri şə rh etmə k getdikcə çə tinlə şir, çünki yüzlə rlə və hə tta minlə rlə potensial uyğunluq ola bilə r.
Tə bii ki, bu axtarışların nə ticə lə rinin ə l ilə şə rhində n söhbə t gedə bilmə z. Bundan ə lavə , tez-tez hə r birinin öz statistikası, konvensiyaları və çıxış formatı olan bir neçə ardıcıl axtarış alə tlə ri ilə işlə mə k lazımdır.
Tə sə vvür edin ki, birdə n çox axtarış alə tində n istifadə edə rə k bir neçə ardıcıllıqla işlə mə k lazım olduqda nə qə də r qorxulu olardı.

Biz bunu özümüz də yaxşı bilirik, ona görə də Biopython-da Bio.SearchIO alt modulunu yaratmışıq.
Bio.SearchIO sizə müxtə lif axtarış alə tlə ri tə rə fində n istifadə edilə n müxtə lif standartlar və konvensiyalarla mə şğul olmaqla yanaşı, axtarış nə ticə lə rinizdə n mə lumatı rahat şə kildə çıxarmağa imkan verir. SearchIO adı BioPerl-in eyniadlı moduluna hörmə tdir.

Bu fə sildə sizin üçün nə edə bilə cə yini göstə rmə k üçün Bio.SearchIO-nun ə sas xüsusiyyə tlə rini nə zə rdə n keçirə cə yik.
Yol boyu iki mə şhur axtarış alə tində n istifadə edə cə yik: BLAST və BLAT. Onlar yalnız illüstrativ mə qsə dlə r üçün istifadə olunur və siz iş axınını Bio.SearchIO tə rə fində n də stə klə nə n istə nilə n digə r axtarış alə tlə rinə uyğunlaşdıra bilmə lisiniz. İstifadə edə cə yimiz axtarış çıxışı faylları ilə birlikdə izlə mə yə çox xoş gə lmisiniz. BLAST çıxış faylı buradan endirilə bilə r , və BLAT çıxış faylı burada və ya Doc/ examples/ qovluğu altında Biopython mə nbə koduna daxil edilir. Hə r iki çıxış faylı bu ardıcıllıqla yaradılıb:

```
>gizli_seq
CCCTCTACAGGGAAGCGCTTTCTGTTGTCTGAAAGAAAGAAAGTGCTTCCTTTTAGAGGG
```

BLAST nə ticə si NCBI refseq_rna verilə nlə r bazasına qarşı blastn istifadə edə rə k yaradılan XML faylıdır. BLAT üçün ardıcıllıq verilə nlə r bazası fevral 2009 hg19 insan genomunun layihə si idi və çıxış formatı PSL-dir.

Biz Bio.SearchIO obyekt modelinə girişdə n başlayacağıq. Model axtarış nə ticə lə rinizin tə msilidir, ona görə də Bio.SearchIO-nun özüdür. Bundan sonra biz Bio.SearchIO-da tez-tez istifadə edə bilə cə yiniz ə sas funksiyaları yoxlayacağıq.

İndi hamımız hazırıq, gə lin ilk addıma keçə k: ə sas obyekt modelini tə qdim edirik.

11.1 SearchIO obyekt modeli

Bir çox ardıcıl axtarış alə tlə ri arasında və hşicə sinə fə rqlə nə n çıxış üslublarına baxmayaraq, onların ə sas konsepsiyasının oxşar olduğu ortaya çıxır:

Machine Translated by Google

- Çıxış faylında bir və ya bir neçə axtarış sorğusunun nə ticə lə ri ola bilə r.
- Hə r bir axtarış sorğusunda siz verilmiş axtarış verilə nlə r bazasından bir və ya bir neçə hit görə cə ksiniz.
- Hə r verilə nlə r bazası vuruşunda siz sorğu ardıcıllığınız və verilə nlə r bazası ardıcıllığı arasında faktiki ardıcıllığın uyğunlaşdırılmasını ehtiva edə n bir və ya bir neçə bölgə ni görə cə ksiniz.
- BLAT və ya Exonerate kimi bə zi proqramlar bu bölgə lə ri bir neçə uyğunlaşdırma fraqmentinə (yaxud BLAT-da bloklara və ola bilsin ki, exonerate-də eksonlara) daha da ayıra bilə r. Bu, hə mişə gördüyünüz bir şey deyil, çünki BLAST və HMMER kimi proqramlar bunu etmir.

Bu ümumiliyi də rk edə rə k, ondan Bio.SearchIO obyekt modelini yaratmaq üçün ə sas kimi istifadə etmə k qə rarına gə ldik. Obyekt modeli Python obyektlə rinin iç-içə iyerarxiyasından ibarə tdir, hə r biri yuxarıda göstə rilə n bir konsepsiyanı tə msil edir. Bu obyektlə r bunlardır:

- Tə k axtarış sorğusunu tə msil etmə k üçün QueryResult.
- Hit, vahid verilə nlə r bazasını tə msil etmə k üçün vuruş. Hit obyektlə ri QueryResult daxilində dir və hə r QueryResult-da sıfır və ya daha çox Hit obyekti var.
- HSP (yüksə k xallı cüt üçün qısa), sorğu və hit ardıcıllığı arasında ə hə miyyə tli uyğunlaşmaların region(lar)ını tə msil etmə k üçün. HSP obyektlə ri Hit obyektlə rinin içə risində dir və hə r Hitin bir və ya daha çox HSP obyekti var.
- HSPFragment, sorğu və hit ardıcıllığı arasında tə k bitişik düzülüşü tə msil etmə k üçün. HSPFragment obyektlə ri HSP obyektlə ri içə risində dir. BLAST və HMMER kimi ardıcıl axtarış vasitə lə rinin ə ksə riyyə ti HSP və HSPFragment obyektlə rini birlə şdirir, çünki hə r bir HSP yalnız bir HSPFragmentə malik olacaqdır. Bununla belə , çoxlu HSPFragment ehtiva edə n HSP istehsal edə n BLAT və Exonerate kimi alə tlə r var. Ə gə r bu, indi bir az çaşqın görünsə , narahat olmayın, sonra bu iki obyekt haqqında daha ə traflı mə lumat verə cə yik.

Bu dörd obyekt Bio.SearchIO-dan istifadə etdiyiniz zaman qarşılıqlı ə laqə quracağınız obyektlə rdir. Onlar ə sas Bio.SearchIO metodlarından biri ilə yaradılmışdır: oxumaq, tə hlil etmə k, indeks və ya index_db. Bu üsulların tə fə rrüatları sonrakı bölmə lə rdə verilir. Bu bölmə üçün biz yalnız oxu və tə hlildə n istifadə edə cə yik. Bu funksiyalar öz Bio.SeqIO və Bio.AlignIO analoqları ilə eyni şə kildə davranır:

- oxu bir sorğu ilə çıxış fayllarının axtarışı üçün istifadə olunur və QueryResult obyektini qaytarır
- parse çoxlu sorğulu axtarış çıxış faylları üçün istifadə olunur və QueryResult verə n generatoru qaytarır obyektlə r

Bunu hə ll etdikdə n sonra QueryResult ilə başlayaraq hə r Bio.SearchIO obyektini araşdırmağa başlayaq.

11.1.1 Sorğu Nə ticə si

QueryResult obyekti tə k axtarış sorğusunu tə msil edir və sıfır və ya daha çox Hit obyektlə rini ehtiva edir. Ə limizdə olan BLAST faylından istifadə nin necə göründüyünə baxaq:

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

Proqram: blastn (2.2.27+)
    Sorğu: 42291 (61)
                mystery_seq
  Hə də f: refseq_rna
      Baxışlar: ---- ------ --------------------------------------------------------
                        # # HSP ID + tə sviri
```