Bölüm 7

BLAST

Hey, herkes BLAST'ı seviyor değil mi?" 1 Yani, tanrım, dizilerinizden biri ile bilinen dünyadaki diğer tüm diziler arasında karşılaştırma yapmak nasıl daha kolay olabilir? Ancak, elbette, bu bölüm BLAST'ın ne kadar harika olduğu ile ilgili değil, çünkü bunu zaten biliyoruz. BLAST ile ilgili sorun, büyük koşular tarafından üretilen veri hacmiyle başa çıkmanın ve genel olarak BLAST koşularını otomatikleştirmenin gerçekten zor olabileceğidir.

Neyse ki, Biopython çalışanları bunu çok iyi biliyorlar, bu yüzden BLAST ile başa çıkmak ve işleri çok daha kolay hale getirmek için birçok araç geliştirdiler. Bu bölümde bu araçların nasıl kullanılacağı ve onlarla nasıl faydalı şeyler yapılacağı anlatılmaktadır.

BLAST ile uğraşmak, her ikisi de Biopython içinden yapılabilen iki adıma ayrılabilir. İlk olarak, sorgu dizileriniz için BLAST'ı çalıştırmak ve soms çıktısını gctting. İkincisi, daha fazla analiz için BLAST çıktısını Python'da park etmek.

BLAST'ı çalıştırmaya ilk girişiniz muhtemelen NCBI web hizmeti aracılığıyla olmuştur. Aslında, BLAST'ı çalıştırabileceğiniz birçok yol vardır ve bunlar çeşitli şekillerde kategorize edilebilir. En önemli ayrım BLAST'ı yerel olarak (kendi makinenizde) çalıştırmak ve BLAST'ı uzaktan (başka bir makinede, ï'l*ÏCalle' L)l€' ÜÜ *Ul ' ers) çalıştırmaktır. \\Bu bölümü, NCBI çevrimiçi BLAST hizmetini bir P 't1ir'n xcl'i1 t içinde h' ri i çağırarak başlatabilirsiniz.

.8'"O TE. Aşağıdaki C'liaptci 5 r1csci'ilaes Bi o . Searchlo, Biopython'da bir *e+deneysel* modül. Bu modülü, Biopython'da bir e+deneysel modül olan Bi o . Blast modülü, daha genel bir çerçeve sağladığından, daha genel bir çerçeve sağladığından *- -i " -" *earl:hint t ,ool s us wall. Bununla birlikte, bu kararlı olarak ilan edilene kadar, üretim kodu için lütfen Bio . Blast modülünü kullanmaya devam edin.

7.1 BLAST'ı İnternet Üzerinden Çalıştırma

Bio'da qbl ast ( ) fonksiyonunu yükseltiyoruz. Bl ask . NCBI WWW modülünde BLAST'ın çevrimiçi sürümünü çağırmak için kullanıyoruz. Bunun isteğe bağlı olmayan üç argümanı vardır:

- The first argument is the blast program to use for the search, as a lower case string. The options and descriptions of the programs are available at https://blast.ncbi.nlm.nih.gov/Blast.cgi. Currently qblast only works with blastn, blastp, blastx, tblast and tblastx.
- The second argument specifies the databases to search against. Again, the options for this are available i tlcKtEIGuidriohL.%STItp://ftp.ncbi.nlz.nih.gov/pub/factsheets/houTo_BLASTGuide.pdf .
- TIH iliirrl 'irgunu nt. bir. strinpi ':t'ntn.ining vr'iir sorgu dizisidir. Bu, dizinin kendisi, fasta formatındaki dizi veya GI riurriber gibi bir tanımlayıcı olabilir.

qbl ast fonksiyonu, temelde BLAST web sayfasında ayarlayabileceğiniz farklı parametrelere benzeyen bir dizi başka seçenek argümanı da alır. Burada sadece birkaç tanesini vurgulayacağız:

- url_base argümanı, BLAST'ı internet üzerinden çalıştırmak için temel URL'yi ayarlar. Varsayılan olarak NCBI'ye bağlanır, ancak bunu bulutta çalışan bir NCBI BLAST örneğine bağlanmak için kullanabilirsiniz. Daha fazla ayrıntı için lütfen qblast işlevinin belgelerine bakın.
- qbl ast işlevi BLAST sonuçlarını isteğe bağlı I ornat _type anahtar sözcüğü ile seçebileceğiniz çeşitli biçimlerde döndürebilir: "HTML", "Metin " "ASN . 1". veya "SQL". '1'1it! 'icfa ttlt. "XML "dir. çünkü I llitt, scr:t i' n 7 3 l'clov' bölümünde açıklanan ayrıştırıcı tarafından beklenen biçimdir.
- expect bağımsız değişkeni beklenti veya e-değeri eşiğini ayarlar.

İsteğe bağlı BLAST argümanları hakkında daha fazla bilgi için sizi NCBI'nin kendi belgelerine veya built into Biopython:

```
>>> from Bio.Blast import NCBIWW
>> help(NCBIWWW qblast)
```

NCBI BLAST web sitesindeki varsayılan ayarların QBLAST'taki varsayılan ayarlarla tam olarak aynı olmadığını unutmayın. Farklı sonuçlar alırsanız, parametreleri kontrol etmeniz gerekir (örneğin, beklenti değeri eşiği ve boşluk değerleri).

Örneğin, BLASTN kullanarak nükleotid veritabanına (nt) karşı aramak istediğiniz bir nükleotid diziniz varsa ve sorgu dizinizin GI numarasını biliyorsanız, şunu kullanabilirsiniz:

```
>>> Bio Blast'tan NCBIWM'yi içe aktar
>>sonuç _hand1e = NCB US . qblast ( "bl ask n" , "nt " , " 833'2 116 " )
```

Alternatif olarak, sorgu dizimiz zaten bir FASTA formatlı dosyada varsa, dosyayı açmamız ve bu kaydı bir dize olarak okumamız ve bunu sorgu argümanı olarak kullanmamız gerekir:

```
>> from Bio.Blast import NCBIWWN
>> Iasta_string = open("m_cold.fasta").read()
>> result_handle - NCBIWW.qblast("blastn", "ut", Iasta_striog)
```

We could also have read in the FASTA file as a `SeqRecord` and then supplied just the sequence itself:

```
>>> Bio.Blast imcort NCBIWWW'den
>> demir Bio inport SeqI0
>> kayıt = SeqID read("m_cold.Pasta", Iormat="fasta")
>>> rezult_handle = NCBI    .qblast("blastn", "nt", record.seq)
```

Yalnızca iiL.fiS'1' sekans menülerini sağlamak, sekansınız için otomatik olarak hava kimliğini belirleyecektir. Bir FASTA dizesi (mevcut tanımlayıcıyı içerecek) oluşturmak için SeqRecord nesnesinin format yöntemini kullanmayı tercih edebilirsiniz:

```
>>> from Bio.Blast import NCBI
"> Irom Bio import SeqI0
"> kayıt = SeqI0.read("m cold.fasta", format="fasta")
>> resu1t_hand1e    NCBIMW.qb1ast("blastn", "nt", record.format("fasta"))
```

Bu ;apJn 'tar li m'ikos there sr net if \ on h'ivt our sequence(s in a non-FASTA file format which you can '-xtr:it Bi o kullanarak . SeqlD (s '- ' --- -t-!-' *)

\4'li:it'iv':r :irguint:iits you give* I ht* qblast ( ) fisuction, you should get back your results in a handle uJ' j':c t (my 'hif'tult in XkIL Bill Illifi). Bundan sonraki adım XhiL çıktısını Python nesnelerine ayrıştırmak olacaktır rt']'l'tscllt.ill¿ tile scitl't'$i i '!sults (Scctioil i ./1) . lrut önce çıktı dosyasının yerel bir kopyasını kaydetmek isteyebilirsiniz. 1 Bu, özellikle BLAST sonuçlarından bilgi çıkaran kodlarda hata ayıklama yaparken yararlıdır

(çünkü çevrimiçi aramayı yeniden çalıştırmak yavaştır ve NCBI bilgisayarının zamanını boşa harcar).

BLAST çıktısını okumak için re su1t;_handle . read( ) işlevini yalnızca bir kez kullanabileceğimiz ve result _handle .read( ) işlevini çağırdığımızda yine boş bir dize döneceği için biraz dikkatli olmamız gerekir.

```
>'> ile open("my_blast.rml", "w") ae out_handle:
 .       out_handle.write(result_handle.read())

>> sonuç hamdle.close()
```

Bunu yaptıktan sonra, sonuçlar ny_blast . xcl dosyasındadır ve orijinal tutamacın tüm verileri çıkarılmıştır (bu yüzden onu kapattık). Bununla birlikte, BLAST ayrıştırıcısının ayrıştırma işlevi (7.3'te açıklanmıştır) dosya tutamacı benzeri bir nesne alır, bu nedenle kaydedilen dosyayı giriş için açabiliriz:

```
>> result_handle = open("benim blast.xml")
```

Artık BLAST sonuçlarını tekrar bir tutamaç haline getirdiğimize göre, onlarla bir şeyler yapmaya hazırız, bu yüzden bu bizi doğrudan ayrıştırma bölümüne götürür (aşağıdaki Bölüm 7.3'e bakın). Şimdi buna geçmek isteyebilirsiniz ....

7.2 BLAST'ı yerel olarak çalıştırma

7.2.1 Giriş

BLAST'ı yerel olarak çalıştırmanın (internet üzerinden çalıştırmanın aksine, bkz. Bölüm 7.1) en azından iki önemli avantajı vardır:

- Yerel BLAST, internet üzerinden BLAST'tan daha hızlı olabilir;
- Yerel BLAST, dizileri aramak için kendi veritabanınızı oluşturmanıza olanak tanır.

Tescilli veya yayınlanmamış dizi verileriyle uğraşmak, BLAST'ı yerel olarak çalıştırmak için başka bir neden olabilir. Dizileri yeniden dağıtmanıza izin verilmeyebilir, bu nedenle bunları NCBI'ye BLAST sorgusu olarak göndermek bir seçenek olmayacaktır.

Ne yazık ki, bazı önemli dezavantajlar da var Tüm parçaları takmak ve doğru şekilde ayarlamak biraz çaba gerektiriyor:

- Yerel BLAST, komut satırı araçlarının yüklenmesini gerektirir.
- Yerel BLAST, (büyük) BLAST veritabanlarının kurulmasını (ve potansiyel olarak güncel tutulmasını) gerektirir.

İşleri daha da karıştırmak için birkaç farklı BLAST paketi mevcuttur ve BLAT gibi taklit BLAST çıktı dosyaları üretebilen başka araçlar da vardır.

7.2.2 Bağımsız NCBI BLAST-}-

"Yeni" NCBI BLAST-i- paketi 2009 yılında piyasaya sürülmüştür. Bu, eski NCBI "legacy" BLAST paketinin yerini almıştır (aşağıya bakınız).

Bu bölümde bu araçların Python içinden nasıl kullanılacağı kısaca gösterilecektir. Eğer Bölüm 6.5'teki hizalama aracı örneklerini daha önce okuduysanız veya denediyseniz, bunların hepsi oldukça basit görünecektir. İlk olarak, bir komut satırı atringi oluşturuyoruz (bağımsız BLAST'ı elle çalıştırıyorsanız komut satırı istemine yazacağınız bir komut). Daha sonra bu komutu Python içinden çalıştırabiliriz.

Örneğin, gen nükleotid dizilerinin bir FASTA dosyasını alarak, yedekli olmayan (NR) protein veritabanına karşı bir BLASTX (çeviri) araması yapmak isteyebilirsiniz. Sizin (veya sistem yöneticinizin) NR veritabanını indirip yüklediğinizi varsayarsak, şu komutu çalıştırabilirsiniz:

```
$ blaBtx -query opuotia.Pasta -db nr -out opuntia.xml -evalue 0.001 -outfmt 5
```

Bu, 0,001'lik bir beklenti kesme değeri kullanarak NR veritabanına karşı BLASTX'i çalıştırmalı ve belirtilen dosyaya (daha sonra ayrıştırabileceğimiz) XML çıktısı üretmelidir. Benim bilgisayarımda bu işlem yaklaşık altı dakika sürüyor - çıktıyı bir dosyaya kaydetmek için iyi bir neden, böylece gerektiğinde herhangi bir analizi tekrarlayabilirsiniz.

Biopython içinden NCBI BLASTX sarmalayıcısını Bio .Bl ask . Appli cat ions modülünden NCBI BLASTX sarmalayıcısını komut satırı dizesini oluşturmak ve çalıştırmak için kullanabiliriz: