Bevezetés a Bioinformatikába Programozás (gyakorlat)

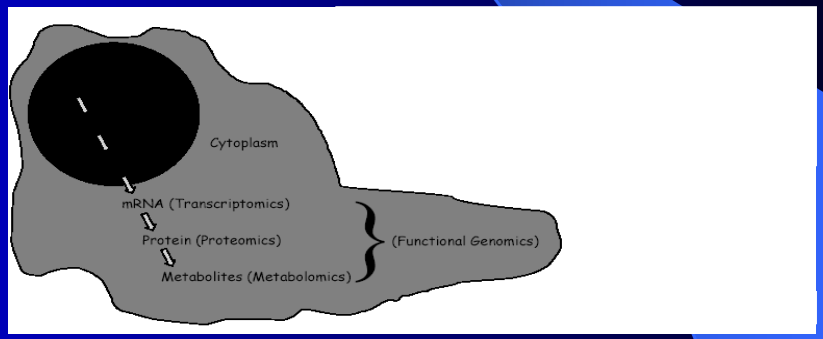
Kozlovszky Miklós

kozlovszky.miklos@nik.uni-obuda.hu

12. Előadás

Az „omics” világ

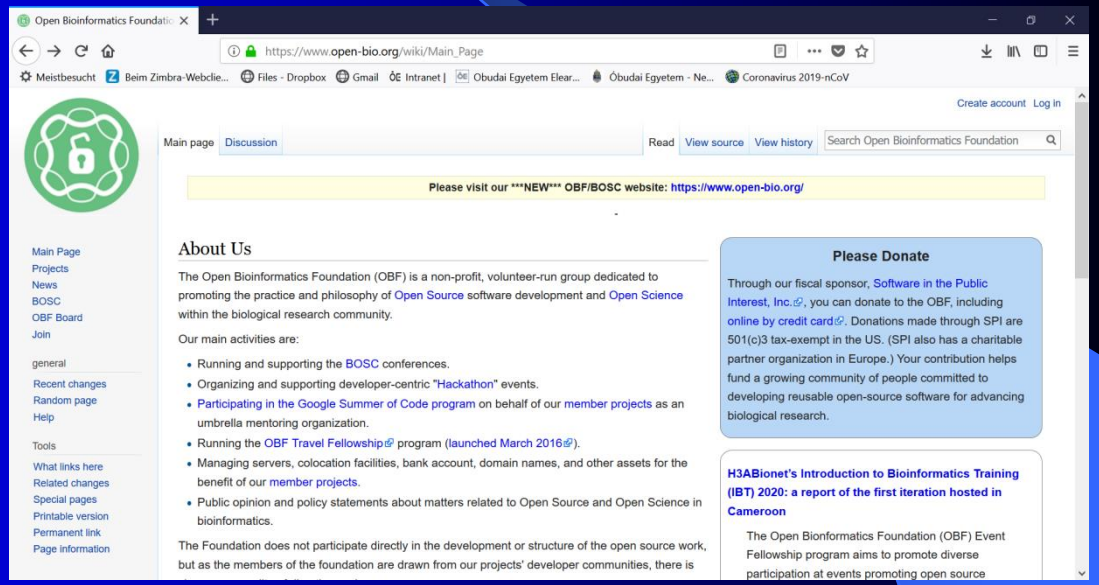
- ✓ Proteomics
  - Vizsgálja, hogy hol keletkeznek, valamint milyen szerepet játszanak a fehérjék az élő szervezetben
  - Structural proteomics
    - Szekvenciákból térszerkezetet vizsgál/jósol, fehérjéket osztályokba sorol, stb. (lásd előző két előadás)
- Genomics
  - Foglalkozik:
    - génstruktúrákkal
    - szabályozó szekvenciákkal
    - nem kódolódó részekkel

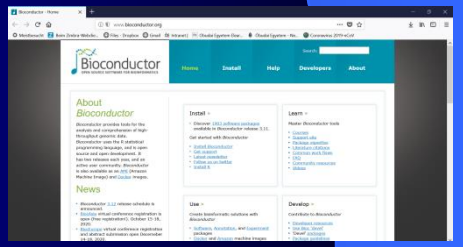


  Megpróbálja leírni az élőlényt a genom szekvenciája alapján
- Transcriptomics
  - Az átírás törvényszerűségeit vizsgálja
- Metabolomics
  - Az élő sejt számára fontos kis molekulasúlyú molekulák vizsgálata

Elterjedtebb bioinformatikai szoftverek

- Open Bioinformatics Foundation (OBF)
  - Nyilt
  - Főbb projektjei
    - BioJava
    - BioPerl
    - Biopython
    - BioRuby
    - BioSQL
    - DAS és Global Sequence Identifiers lista
    - EMBOSS
- Bioconductor - R





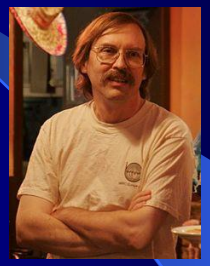
Telepítés

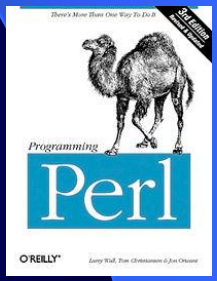
- Windows 10 (vmware) https://developer.microsoft.com/hu-hu/windows/downloads/virtual-machines/
- Bioperl
  - https://bioperl.org/INSTALL.html
- Biopython
  - https://biopython.org/wiki/Download
- Bioconductor (R nyelv)
  - http://www.bioconductor.org/

Perl/BioPerl
Programozás gyakorlat

Perl

- Elérhetősége
  - www.perl.org/
  - www.cpan.org/
- Története
  - Larry Wall (Unisys), 1987 („Pearl”)
  - vagy:
    - **P**ractical **E**xtracting and **R**eporting **L**anguage
    - **P**athologically **E**clectic and **R**ubbis **L**ister
    - …
  - C + shell scriptek (sh)+AWK + Lisp+ …
  - 2010 : Perl 5 (5.12)
- Irodalom
  - „Tevés” könyv → Programming Perl
  - WEB
    - Magyarul: http://www.szabilinux.hu/verhas/perl/index.html

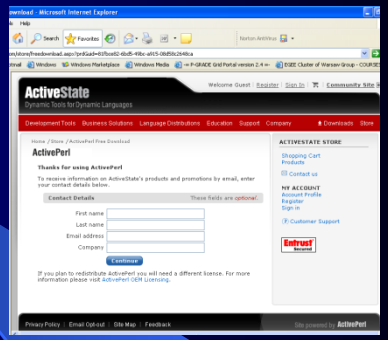




Perl bevezetés

- Alapvetően script alapú programozási nyelv
- Nagyon jól támogatja a reguláris kifejezéseket
  - kiválóan alkalmas nagy méretű szöveg- vagy adatfile-ok feldolgozására, hatékony szövegkezelés.
- Elterjedt, egyszerű, gyors
- További előnyök
  - Gyors nemcsak futtatásban, hanem fejlesztésnél is
  - Használatra optimalizált, és nem szépségre
  - Moduláris
  - Használhatók az elterjedt programozási paradigmák

Perl telepítés



- Windows
  - ActiveState
- Linux
  - Disztribúció függő
    - érdemes valami csomagkezelővel felrakni (yum, apt-get, stb…)
  - Egyéb módszerekkel
    - http://www.cpan.org/ ~14K modul!
    - www.**perl**.com/download.csp
    - www.**perl**.org/get.html
    - …

Perl gyorstalpaló

- Építőkövek:

```
$skalar = "Ez egy szoveg"; # skalár változó
$szamskalar = 123.456; # skalár szám értékkel
@tomb = (1,2,3,4); # egy tömb
$i = $tomb[0]; # a tömb legelső eleme
```

Perl gyorstalpaló (alapok)

- Tömbök

  @tomb = ('Laci', ‘Teri', ‘Gyuri'); #a teljes tömb

  $tomb[0]='Laci‘ #első elem

  $#tomb #a tomb utolsó indexe

- shift
  - @tomb = ('Laci', ‘Teri', ‘Gyuri');
  - $valtozo = shift(@tomb);
    - (kiszedi a legkisebb elemet a tömb-ből és beteszi $valtozo-ba ($valtozo= 'Laci‘, @tomb = (‘Teri', ‘Gyuri');

- Indítási paraméterek kiolvasása:

@ARGV tömbbe kerülnek a paraméterek

Perl gyorstalpaló (folyt.)

while (kifejezés) blokk

while (kifejezés) blokk continue blokk

for (kifejezés1 ; utasítás2 ; utasítás3) blokk

foreach változó (lista) blokk

foreach változó (lista) blokk continue blokk

if (kifejezés) blokk

else blokk1

Perl gyorstalpaló (Szubrutin használat)

```
&sajatsubrutin;
#ez itt egy comment
sub sajatsubrutin
{
    print "Bent vagyok a szubrutinban!!!"; return;
}
print (“hello vilag\n”);
```

Modulok

- -Szubrutinok gyűjteménye
- -Objektum hierarchia

Perl gyorstalpaló (Értékadás szubrutinnak)

```
&sajatsubrutin2(1,2,3);
sub sajatsubrutin2
{
   print (“$_[0],$_[1],$_[2] \n“);
}
```

BioPerl

- Biológiai problémák megoldására optimalizált perl alapú API
- Nyílt forráskód
- Kollaboratív projekt
- Elérhetősége
  - www.bioperl.org

BioPerl (folyt)

- Történelem
  - 1995 óta
  - Első stabil verzió: 2002

- 100+ fejlesztő, nemzetközi fejlesztés
- Kis core programozó csapat
- Elterjedten használják genomikai központokban és kis laborokban egyaránt
- Értékesíthető tudást ad a piacon

A BioPerl mire lett kitalálva

- Szekvenciák beolvasása szabványos file-okból (FAST, GenBank, EMBL,SwissPr SwissProt,...)
- Szekvencia manipuláció, átfordítás, stb.
- BLAST report-okban szövegbányászat
- Főbb területek:
  - Szekvencia illesztés (egyszeres/többszörös)
  - Bibliografikus adatok feldolgozása
  - Adatbázis kapcsolatok
  - Reguláris kifejezések használata, illesztések/keresések (pl:blast) használat

Perl/bioperl szintaxis

- Első sor: #!/usr/bin/perl –w

(megadjuk hogy hol van a perl és használjuk a debug-ot)

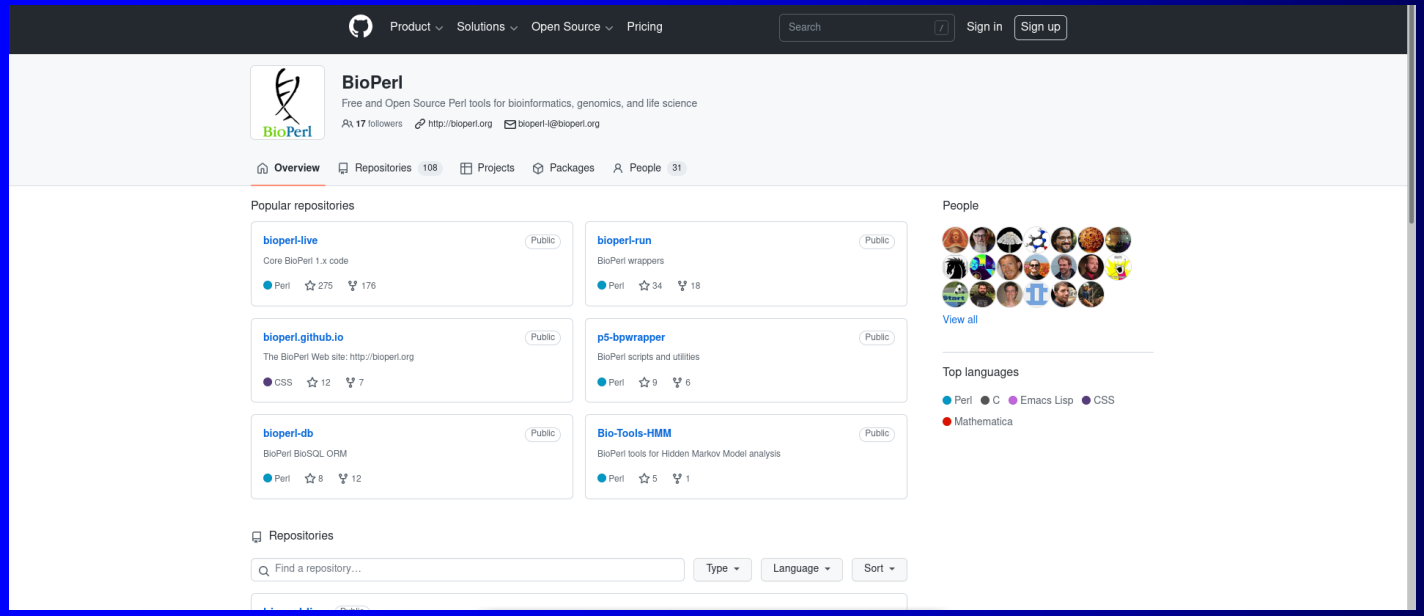
- #ez itt egy comment rész (csak 2. sortól!)
- Minden utasítás/sor végén “;” kell hogy álljon
- use lib „/xx/xx/bioperl_modules”; ha szükségünk van külső modulokra
- use Bio::Seq;

Megadjuk, hogy milyen modulokat használunk (use x::y)

- $x=1 („x” nevű változó létrehozása és értékadás)
- unixon: chmod +x *.pl (ha nem automatikus a futtatás)
- Parancssorból indítsuk: perl -version

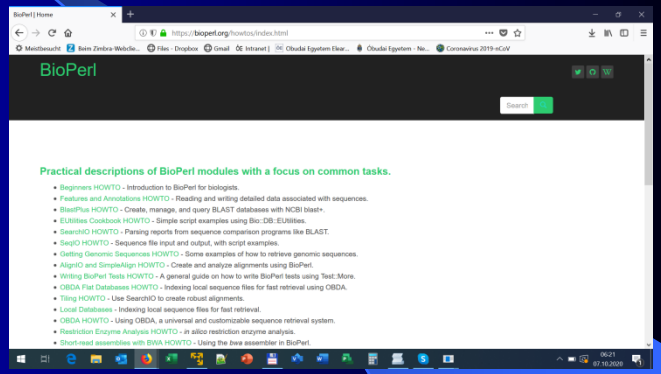
BioPerl kód és háttéranyagok

- Aktuális code release: BioPerl 1.7.2 (2018. szept.)
- https://bioperl.org/howtos/index.html

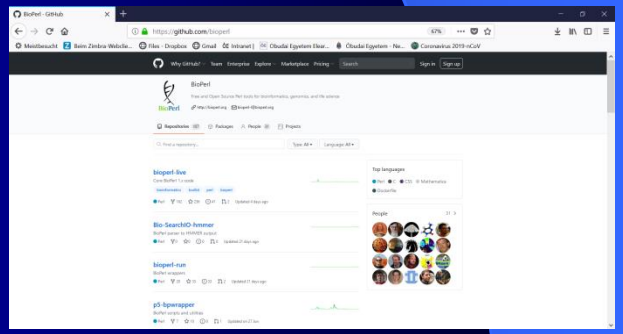


Információk a BioPerl csomagokról

https://bioperl.org/howtos/index.html



https://github.com/bioperl



Perl használat Windows-on/Linux-on

- Kell hozzá:
  - egy kényelmes szövegszerkesztő
    - Pl.: Notepad++
  - parancssor (command/cmd)
  - telepített perl futtatási környezet

Néhány segítség a feladatokhoz

- Ha szeretnénk külső parancsot indítani perl-ből (pl.: shell scriptet)

  system(”date”);
- a tomb-höz tartozó legnagyobb aktuális index értéke

  $#tomb
- Kilépés jól, vagy hibával

  exit (0), vagy exit(1)
- a sortörés/speciális karakterek levágására

  chop() (mindíg vág) és chomp () (csak új sor jelet vág)
- file meglétének ellenőrzése:

  open(in,"<$input_file") || die "$0: cannot open $input_file file\n";

Ha szeretnénk használni egy csomagot, de nincs feltelepítve:

Szoftver Repository: http://bioperl.org
Grafikusan (Perl Package Manager) / vagy parancssorból (ppm-shell
search bioperl
install)

Perl/bioperl szintaxis (gyakorlat start)

- Első sor: #!/usr/bin/perl –w

(megadjuk hogy hol van a perl és használjuk a debug-ot)
#ez itt egy megjegyzés (comment) rész (csak 2. sortól!)

- Minden utasítás/sor végén “;” kell hogy álljon

use lib „/xx/xx/bioperl_modules”; ha szükségünk van külső modulokra
use Bio::Seq;
Megadjuk, hogy milyen modulokat használunk (use x::y)

- Unix-on: chmod +x *.pl (ha nem automatikus a futtatás)

Első lépések
Perl Helloworld…

- Feladat: Parancssorból indítsuk: perl –version

```
#!/usr/bin/perl
print(“Hello World”);
```

```
$text=“Ez magyar Hello Világ”;
print($text);
```

```
$txt1=“két részes”;
$txt2=“szöveg összeragasztása”;
print($txt1.$txt2);
```

Rövid feladatok
Értékadás változóknak

```
$a = 12;
$b = 23;
$c = $a + $b;
$d = ”Haliho\n";
---------------------------
```

- Írjuk ki a többi változókat is a képernyőre

Rövid feladatok (folyt.)

- Értékadás
  - @tomb = (1, 2 , 3);
  - Írjuk ki a tömb első elemét: $tomb[0]
- Egyszerűbb program szerkezetek

```
$valtozo1=1;
if( $valtozo1 ==1 )
{
    print("egyenlo\n”);
    $valtozo2= 2;
}else
{
    print("nem egyenlo\n“);
    $valtozo1=3;
}
print ("$valtozo1,$valtozo2\n");
```

Perl gyorstalpaló (alapok)

- Ciklusok konkrétan

  for ($i = 1; $i < 10; $i++) {…}

  ,vagy

  $i = 1;

  while($i < 10) {$i++;….}

- Operátorok (néhány fontosabb)
  - Szöveg és szám összehasonlítás
    - <,>,<=,>=, lt, gt, le, ge
    - ==,!=,<=>, eq, ne, cmp
    - Logikai és: &&
    - Logikai vagy:||

Perl gyorstalpaló
(I/O - File kezelés)

- I/O operátor a „<„ és a „>”.
- Nyitás
  - Olvasni fájlból

  open(alias_olvasni_név,"<filenév1.txt");
  - Írni fájlba

  open(alias_irni_név,">filenév2.txt");

  Lehet még >> (hozzáírás)
- Példa:

```
while( $line = <alias_olvasni_név> )
{
    print alias_írni_név $line;
}
```

- Bezárás

```
close alias_irni_név;
close alias_olvasni_név;
```

- Inputok a billentyűzetről: <STDIN>

Megjegyzés: Adott helyre direkt ugrás (seek handler,pozíció, honnan)

Feladatok

- Írassuk ki a program indítási paramétereit

```
foreach $parameter (@ARGV)
{
        print ($parameter);
}
```

- Adjunk meg számokat a program indításánál és keressük meg/írjuk ki a legnagyobbat.

```
$max = pop(@ARGV);
foreach $elem (@ARGV)
{
        $max = $elem if $max < $elem;
}
print $max;
```

Feladatok – közvetlen adatbetöltés (BioPerl)

```
use Bio::Seq;
$seq_obj = Bio::Seq->new(-seq=> 'actgtaact',
                         -description  => 'Sample Bio::Seq object',
                         -display_id   => 'something',
                         -accession_number => 'accnum',
                         -alphabet     => 'dna' );
print $seq_obj->seq;
```

- A szekvenciákat objektumként kell feltölteni (több különféle van: **Seq**, PrimarySeq, LocatableSeq, RelSegment, LiveSeq, stb.)

I/O - File kezelés

- I/O operátor a „<„ és a „>”.
- Nyitás
  - Olvasni fájlból

  open(alias_olvasni_név,"<filenév1.txt");
  - Írni fájlba

  open(alias_irni_név,">filenév2.txt");

  Lehet még >> (hozzáírás)
- Példa:

```
while( $line = <alias_olvasni_név> )
{
    print alias_írni_név $line;
}
```

- Bezárás

```
close alias_irni_név;
close alias_olvasni_név;
```

- Inputok a billentyűzetről: <STDIN>

Megjegyzés: Adott helyre direkt ugrás (seek handler,pozíció, honnan)

Feladatok – file beolvasás (BioPerl)

- Olvassunk be egy szekvenciát (pl.: a haemoglobin A láncát)!
  - FASTA formátumban… (HBA_HUMAN)

```
use Bio::SeqIO;
$seq_obj  = Bio::SeqIO->new(-file => "inputfilename",
                    -format => 'Fasta');
```

- Ha több szekvenciát tartalmaz a FASTA fájl:

```
$seq = $seq_obj->next_seq;
print $seq->seq;
```

Feladat
Írassuk ki a betöltött szekvenciákat a képernyőre

```
while ($seq_obj = $in->next_seq){
    # print the sequence
    print $seq_obj->seq,"\n";
}
```

Feladat – file kiírása (fasta formátumba)

```
use Bio::Seq;
use Bio::SeqIO;
$seq = Bio::Seq->new(-seq   =>'actgtggcgtcaact',
                -description    => ‘ez egy sample Bio::Seq tipusu object',
                -display_id       => 'something',
                 -accession_number => 'accnum',
                -alphabet         => 'dna' );
$seqio_obj = Bio::SeqIO->new(-file => '>sequence.fasta', -format => 'fasta' );
$seqio_obj->write_seq($seq);
```

Formátum konvertálás

```
use strict;
my $infile = "bemenetifilenev.xx";
my $infileformat = „bemeneti formatum";
my $outfile = "outputfilenev.xxx";
my $outfileformat = „kimenoformatum";   # Egy-egy SeqIO objektum
    irasra olvasasra
my $seq_in = Bio::SeqIO->new('-file' => "<$infile",
                                        '-format' => $infileformat);
my $seq_out = Bio::SeqIO->new('-file' => ">$outfile",
                                          '-format' => $outfileformat);
while (my $inseq = $seq_in->next_seq)
{ #addig olvasok amig van a file-ban valami
          $seq_out->write_seq($inseq);
}
```

Műveletek szekvenciákon

- Translate: bármit, ami nem protein abc-ből épül fel, le tud fordítani protein szekvenciára
- Lehet paraméterezni:
  - milyen codon táblát használjon (pl:mitokondriálist, standard, bacterial, stb.)
  - Mit csináljon STOP és START jelek esetén

```
$seq = Bio::Seq->new(-seq    => 'actgtggcgtcaact',
                     -description  => 'Sample Bio::Seq object',
                     -display_id       => 'something',
                     -accession_number => 'accnum',
                     -alphabet         => 'dna' );
$prot_obj=$seq->translate;
print $prot_obj->seq;
```

Egyszerű statisztikák készítése

- A szekvencia alkotó részeiről egyszerű statisztikákat nyerhetünk (pl: mol. tömeg, monomerek/kodonok száma

```
use Bio::Tools::SeqStats;   #ezt a csomagot kell használni
$prot_obj=$seq->translate;
$seq_stats  = Bio::Tools::SeqStats->new($seq);#$seqobj);
$weight = $seq_stats->get_mol_wt();
$codon_ref = $seq_stats->count_codons();  # aminosavakhoz
$hash_ref = $seq_stats->count_monomers();  # nukleinsavakhoz
```

Példa az adatok kiírására:

```
 foreach $base (sort keys %$hash_ref) {
     print "Number of bases of type ", $base, "= ", %$hash_ref->{$base},"\n";  }
```

Feladat:

Nézzük/keressük meg az egyszerű statisztikákhoz tartozó API leírásokat.

BioPerl Deobfuscator : http://bioperl.org/cgi-bin/deob_interface.cgi

Adatbázis hozzáférés

```
$gb = new Bio::DB::GenBank();

$seq1 = $gb->get_Seq_by_id('MUSIGHBA1'); #kapunk vissza egy szekvenciát ID alapján

$seq2 = $gb->get_Seq_by_acc('AF303112'); # kapunk vissza egy szekvenciát acc. szám alapján:

$seqio = $gb->get_Stream_by_id(["J00522","AF303112","2981014"]);
$seq3 = $seqio->next_seq; # ez ugyanolyan seqio objektumot ad vissza, mint amkor fájlból olvasunk
```

(hasonlóan lehet elérni: GenBank, GenPept, RefSeq, SwissProt, EMBL)

Adatbázis hozzáférés (Genbank hozzáférés)

Feladat:

- Töltsünk le egy szekvenciát a Genbank-ból
- Írjuk ki a képernyőre

Feladat

- Indításnál olvassunk be paraméterként egy fájl nevét és egy fájl formátumot.
- Ha nem kapunk indításkor két paramétert lépjünk ki és írjunk hibaüzenetet.
- Olvassuk be a fájlból az ID-t és keressük meg a hozzátartozó szekvenciát a Genbank-en.
- Írjuk ki a szekvenciát az indításkor megadott adott formátumba(pl.:Genbank).

Példa külső program futtatására

Szignál helyek keresése fehérje szekvenciákban

- Sigcleave - gyakorta használt program
- Bioperl-ben is implementálva lett.
- A program érzékenységét a threshold-al lehet változtatni
- Eredetileg az EGCG csomagban volt (Von Heijne féle algoritmus)
- Használható pro-, és eukariótákra (paraméterezni kell)
- A jó threshold érték:3.5
- 95%-os hatásfokú a szignál és nem szignál szekvenciák, valamint 75%-os a cleavage helyek jóslásában
- Bár minden részen jósol kötőhelyeket, csak az N-terminális oldalon releváns

Sigcleave futtatás

```
use Bio::Tools::Sigcleave;
# create a Seq object, for example:
 $seqobj = Bio::Seq->new(-seq =>
   "AALLHHHHHHGGGGPPRTTTTTVVVVVVVVVVVVV
   VVVV");
 $sigcleave_object = new Bio::Tools::Sigcleave
     ( -seq         => $seqobj,
       -threshold    => 3.5,
       -description  => 'test sigcleave protein seq',);
$formatted_output = $sigcleave_object->pretty_print;
```