

BioPerl

Février 2016

Bérénice Batut

✉ berenice.batut@udamail.fr

 PDF



BioPerl

Principe

- Collection de module Perl
- Objectif: Faciliter le developpement de scripts Perl pour des applications bioinformatiques
- Open-source via une organisation [GitHub](#)
- Soutenu par Open Bioinformatics Foundation

Histoire

- 1996 : Début
- 2002
 - Premier Open Bio Hackathon
 - BioPerl 1.0
 - [Article](#)

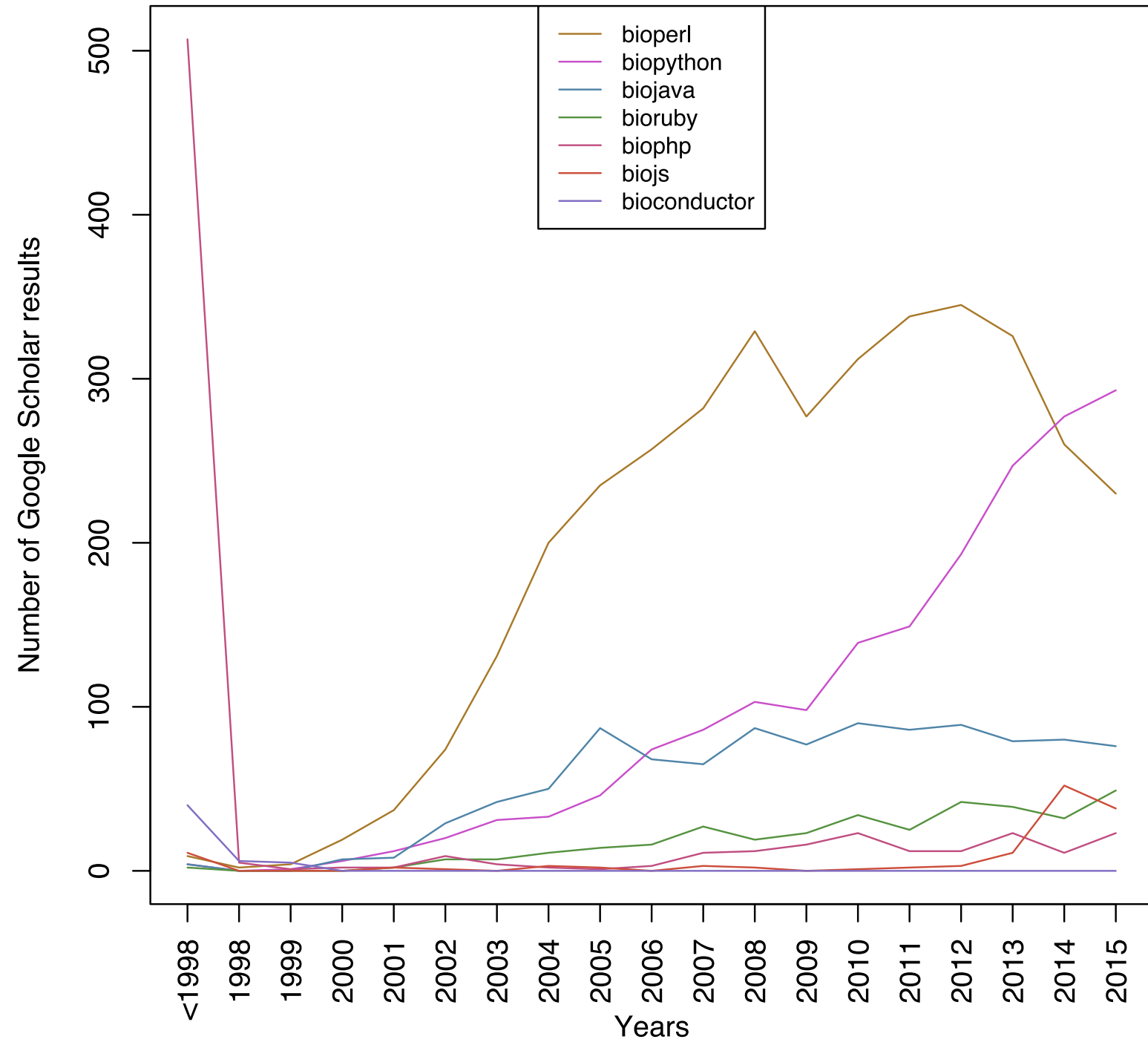
Actuellement

- [GitHub](#)
 - 31 contributeurs
- Dernière release : 1.6.924 en Juillet 2014
- Orienté-objet
- > 40 Modules Perl

Comparaison avec les autres Bio Toolkits

Bio Toolkits

	Release 1.0	Dernière release	Article majeur	Citations
BioPerl	2002	07/2014	2002	1 306
BioPython	2000	10/2015	2009	608
BioJava	2008	07/2015	2008	201
BioRuby	2006	07/2015	-	-
BioPHP	2003	?	-	-
BioJS	2013	09/2014	2013	44
Bioconductor	2001	10/2015		



Installation

Installation sous Linux/Mac OS

```
$ (sudo) cpan -i CPAN
$ cpan
cpan[1]> d /BioPerl/
Reading '/Users/cidam/.cpan/Metadata'
  Database was generated on Thu, 14 Jan 2016 13:53:43 GMT
Distribution      BOZO/Fry-Lib-BioPerl-0.15.tar.gz
Distribution      CDRAUG/Dist-Zilla-PluginBundle-BioPerl-0.20.tar.gz
Distribution      CJFIELDS/BioPerl-1.6.901.tar.gz
Distribution      CJFIELDS/BioPerl-1.6.923.tar.gz
Distribution      CJFIELDS/BioPerl-1.6.924.tar.gz
...
11 items found

cpan[2]> install CJFIELDS/BioPerl-1.6.924.tar.gz
```

Installation sous Windows



DSSL

Manipulation de séquences

Représentation d'une séquence

3 types d'objets pour une séquence

- `Bio::PrimarySeq`
 - Séquence + nom
 - Fichier fasta
- `Bio::SeqFeatureI`
 - Caractéristique sur une séquence (séquence, localisation et annotation)
 - Entrée simple d'une table de caractéristique EMBL/GenBank/DDBJ
- `Bio::Seq`
 - 1 séquence et une collection de caractéristiques
 - Entrée simple d'une table EMBL/GenBank/DDBJ

Classe Bio::Seq

Classe Bio::Seq

```
$ perldoc Bio::Seq
```

```
NAME
```

```
    Bio::Seq - Sequence object, with features
```

```
...
```

```
DESCRIPTION
```

```
    A Seq object is a sequence with sequence features placed on it.  
    The Seq object contains a PrimarySeq object for the actual sequence  
    and also implements its interface.
```

```
...
```


Créer d'un objet `Bio::Seq`

```
use Bio::Seq;  
  
my $seqobj = Bio::Seq->new(  
    -seq => "ACTGTGTGTCC",  
    -id => "Chlorella sorokiniana",  
    -accession_number => "CAA41635"  
);
```

Méthodes (1)

Méthodes renvoyant des chaines de caractères et acceptant parfois des chaine de caractères pour modifier des propriétés

```
$seqobj->seq();           # string of sequence
$seqobj->subseq(5,10);      # part of the sequence as a string
$seqobj->accession_number(); # when there, the accession number
$seqobj->alphabet();        # one of 'dna','rna',or 'protein'
$seqobj->version()          # when there, the version
$seqobj->keywords();        # when there, the Keywords line
$seqobj->length()           # length
$seqobj->desc();            # description
$seqobj->primary_id();      # a unique id for this sequence regardless
# of its display_id or accession number
$seqobj->display_id();      # the human readable id of the sequence
```

Méthodes (2)

Méthodes renvoyant des nouveaux objet Bio::Seq

```
$seqobj->trunc(5,10) # truncation from 5 to 10 as new object  
$seqobj->revcom      # reverse complements sequence  
$seqobj->translate    # translation of the sequence
```

Méthodes (3)

Méthode pour déterminer si une chaîne de caractère peut être acceptée par la méthode `seq()`

```
$seqobj->validate_seq($string)
```

Manipulation de séquences

```
$seq = $seqobj->seq();  
$length = $seqobj->length();  
$subseq = $seqobj->subseq($length/2, $length);  
$new_seq = $seq.$subseq;  
if($seqobj->validate_seq($new_seq)){  
    $seqobj->seq($new_seq);  
}  
print $seqobj->seq(), " ", $seqobj->length()\n";
```

Traduction

```
$translated_obj = $seqobj;  
if( $seqobj->alphabet() == 'dna'){  
    $translated_obj = $seqobj->translate();  
}  
print $translated_obj->seq(),"\n";
```

Récupération de statistiques sur une séquence

Classe Bio::Tools::SeqStats

```
$ perldoc Bio::Tools::SeqStats
```

NAME

Bio::Tools::SeqStats - Object holding statistics
for one particular sequence

...

DESCRIPTION

Bio::Tools::SeqStats is a lightweight object for the calculation of simple statistical and numerical properties of a sequence. By "lightweight" I mean that only "primary" sequences are handled by the object. The calling script needs to create the appropriate primary sequence to be passed to SeqStats if statistics on a sequence feature are required. Similarly if a codon count is desired for a frame-shifted sequence and/or a negative strand sequence, the calling script needs to create that sequence and pass it to the SeqStats object.

Méthodes

- `count_monomers`
 - Comptage du nombre de chaque type de monomère
- `get_mol_wt`
 - Calcul du poids moléculaire
- `count_codons`
 - Comptage du nombre de chaque type de codons
- `hydropathicity`
 - Calcul l'hydropathicité moyenne de Kyte-Doolittle

QCM

```
sub f{  
  my ($x,$y) = @_;  
  $$x = 20;  
  $z = $y+10;  
  return (\$z);  
}  
my $v = 10;  
my $w = 20;  
my $x = f(\$v,$w);  
my $y = $$x + $v;
```

Que contient \$y?

☐ 30

☐ 40

☐ 50

☐ 60

Manipulation de fichiers de séquences

Classe Bio::SeqIO

```
$ perldoc Bio::SeqIO
```

NAME

Bio::SeqIO - Handler for SeqIO Formats

...

DESCRIPTION

Bio::SeqIO is a handler module for the formats in the SeqIO set (eg Bio::SeqIO::fasta). It is the officially sanctioned way of getting the format objects, which most people should use.

The Bio::SeqIO system can be thought of like biological file handles. They are attached to filehandles with smart formatting rules (eg, genbank format, or EMBL format, or binary trace file format) and can either read or write sequence objects (Bio::Seq objects, or more correctly, Bio::SeqI implementing objects, of which Bio::Seq is one).

Constructeur

- Paramètres possibles
 - `-file`
 - `-string`
 - `-format`: fasta, nexus, fastq, quality, excel, raw, tab, ...
 - `-alphabet`: dna, rna ou protein
- Création d'un objet `Bio::SeqIO`: ouverture d'un flux sur le fichier ou la chaîne de caractères

Méthodes

- `next_seq`
 - Lecture du prochain objet "séquence" dans le flux
 - Renvoi d'un objet `Bio::Seq` ou rien si aucune séquence disponible
- `write_seq`
 - Ecriture d'un objet `Bio::Seq` dans le flux
- `format, alphabet, ...`

Ecrire de séquences dans un fichier

```
use Bio::SeqIO;
use Bio::Seq;
my $seqio_obj = Bio::SeqIO->new(-file => '>sequence.fasta',
    -format => 'fasta' );
my $seqobj = Bio::Seq->new(
    -seq => "ACTGTGTGTCC",
    -id => "Chlorella sorokiniana"
);
$seqio_obj->write_seq($seqobj);
my $seqobj = Bio::Seq->new(
    -seq => "ACTGTGTGTCCTGTGTCC",
    -id => "Modified Chlorella sorokiniana"
);
$seqio_obj->write_seq($seqobj);
```

Lecture des séquences d'un fichier

```
use Bio::SeqIO;  
  
$seqio_obj = Bio::SeqIO->new(-file => "sequence.fasta",  
                             -format => "fasta" );  
  
while ($seq_obj = $seqio_obj->next_seq){  
    print $seq_obj->seq, "\n";  
}
```


QCM

```
sub f{  
  my ($x,$y) = @_;  
  $$x = 20;  
  $z = $y+10;  
  return (\$z);  
}  
my $v = 10;  
my $w = 20;  
my $x = f(\$v,$w);  
my $y = $$x + $v;
```

Que contient \$y?

☐ 30

☐ 40

☐ 50

☐ 60

Accès aux bases de données

Récupération d'une séquence dans une base de données

Bases de données accessibles

Base de données	Module
GenBank	Bio::DB::GenBank
SwissProt	Bio::DB::SwissProt
GenPept	Bio::DB::GenPept
EMBL	Bio::DB::EMBL
SeqHound	Bio::DB::SeqHound
Entrez Gene	Bio::DB::EntrezGene
RefSeq	Bio::DB::RefSeq

Classe `Bio::DB::GenBank`

```
$ perldoc Bio::DB::GenBank
```

```
NAME
```

```
    Bio::DB::GenBank - Database object interface to GenBank
```

```
...
```

```
DESCRIPTION
```

```
    Allows the dynamic retrieval of Bio::Seq sequence objects from the  
    GenBank database at NCBI, via an Entrez query
```

```
...
```

Constructeur

```
use Bio::DB::GenBank;  
$db_obj = Bio::DB::GenBank->new;
```

Méthodes

- `get_Seq_by_id($unique_id)`
- `get_Seq_by_acc($accession_number)`
- `get_Seq_by_version($versioned_accession_number)`
- `get_Seq_by_gi($genbank_gi_number)`

⚠ Utiliser le bon identifiant pour la bonne méthode

Récupération d'une séquence dans une base de données

```
use Bio::DB::GenBank;
use Bio::Seq;

$db_obj = Bio::DB::GenBank->new;

$seq_obj = $db_obj->get_Seq_by_id(2);

print $seq_obj->display_id(),"\n";
```


Récupération de plusieurs
séquences avec des requêtes
plus complexes

Bases de données et modules pour les requêtes

Base de données	Module
GenBank	Bio::DB::Query::GenBank
SwissProt	Bio::DB::Query::SwissProt
GenPept	Bio::DB::Query::GenPept
EMBL	Bio::DB::Query::EMBL
SeqHound	Bio::DB::Query::SeqHound
Entrez Gene	Bio::DB::Query::EntrezGene
RefSeq	Bio::DB::Query::RefSeq

Classe `Bio::DB::Query::GenBank`

```
$ perldoc Bio::DB::Query::GenBank
```

NAME

`Bio::DB::Query::GenBank` - Build a GenBank Entrez Query

...

DESCRIPTION

This class encapsulates NCBI Entrez queries. It can be used to store a list of GI numbers, to translate an Entrez query expression into a list of GI numbers, or to count the number of terms that would be returned by a query. Once created, the query object can be passed to a `Bio::DB::GenBank` object in order to retrieve the entries corresponding to the query.

...

Constructeur

- Paramètres possibles
 - `-db : protein, nucleotide, ...`
 - `-query`
 - `-mindate`
 - `-maxdate`
 - `-reldate`
 - `-datatype`
 - `-ids`
 - `-maxids`
- Création d'un objet `Bio::DB::Query::GenBank :`
ouverture d'un flux sur des objets `Bio::Seq`

Méthodes

- `count`
 - Renvoi du nombre de résultats de la requête
- `ids`
 - Renvoi/Modifie la liste des identifiants des résultats

Récupération de plusieurs séquences

```
use Bio::DB::GenBank;
use Bio::DB::Query::GenBank;

$query = "Arabidopsis[ORGN] AND topoisomerase[TITL] and 0:3000[SLEN]";
$query_obj = Bio::DB::Query::GenBank->new(-db => 'nucleotide',
    -query => $query );

$gb_obj = Bio::DB::GenBank->new;

$stream_obj = $gb_obj->get_Stream_by_query($query_obj);

while ($seq_obj = $stream_obj->next_seq) {
    print $seq_obj->display_id, "\t", $seq_obj->length, "\n";
}
```

Parser des rapports de recherche

Classe Bio::SearchIO

```
$ perldoc Bio::SearchIO
```

NAME

Bio::SearchIO - Driver **for** parsing Sequence Database Searches (BLAST, FASTA, ...)

...

DESCRIPTION

This **is** a driver **for** instantiating a parser **for** report files **from** sequence database searches. This object serves **as** a wrapper **for** the format parsers **in** Bio::SearchIO::* - you should **not** need to ever use those format parsers directly. (For people used to the SeqIO system we are deliberately using the same pattern).

...

Constructeur

- Paramètres possibles
 - -file
 - -format
 - -output_format
 - -inclusion_threshold
 - -signif
 - -check_all_hits
 - -min_query_len
 - -best
- Création d'un objet `Bio::Search` avec un flux sur le fichier

Formats

Name	Format
blast	BLAST (WUBLAST, NCBI BLAST, bl2seq)
fasta	FASTA -m9 and -m0
blasttable	BLAST -m9 or -m8 output (both NCBI and WUBLAST tabular)
megablast	MEGABLAST
blastxml	NCBI BLAST XML
...	...

Méthodes

- `next_result`
- `write_result`
- `write_report`
- `result_count`
- `best_hit_only`
- `check_all_hits`

Représentation des données dans **Bio::Search**

- Bio::Search
 - Bio::Search::Result
 - Bio::Search::Hit
 - Bio::Search::HSP (high-scoring segment pair)

Méthodes de `Bio::Search::Result`

- `algorithm`
- `query_name`
- `query_accession`
- `query_length`
- `query_description`
- `database_name`
- `available_statistics`
- `available_parameters`
- `num_hits`
- `hits`

Méthodes de `Bio::Search::Hit`

- `name`
- `length`
- `accession`
- `description`
- `algorithm`
- `raw_score`
- `significance`
- `hsps`
- `num_hsps`
- `locus`
- `accession_number`

Méthodes de Bio::Search::HSP (1)

- algorithm
- evaluate
- expect
- frac_identical
- frac_conserved
- gaps
- query_string
- hit_string
- length('total'/'hit'/'query')
- num_conserved
- num_identical

Méthodes de `Bio::Search::HSP` (1)

- `rank`
- `seq_inds('hit'/'query', 'identical'/'conserved'/'conserved-notidentical')`
- `score`
- `range('hit'/'query')`
- `percent_identity`
- `strand('hit'/'query')`
- `start('hit'/'query')`
- `end('hit'/'query')`
- `matches('hit'/'query')`
- `get_aln`

Parcours d'un fichier issu d'une requête Blast

```
use Bio::SearchIO;

my $in = new Bio::SearchIO(
    -format => "blast",
    -file => "report.bls");
while(my $result = $in->next_result){
    while(my $hit = $result->next_hit){
        while(my $hsp = $hit->next_hsp){
            print "Query=", $result->query_name,
                " Hit=", $hit->name,
                " Length=", $hsp->length('total'),
                " Percent_id=", $hsp->percent_identity,
                "\n";
        }
    }
}
```

Références

- [BioPerl GitHub Page](#)
- [Wiki BioPerl](#)