PARS

PfAm bRowSer

python package for Pfam databases usage

Basic Decription

Pfam **automatisation** **utility**

The python package for browsing, downloading and convenient usage of files deposed in **pfam** and **rfam** databases

Easy **translation** of databases accessions numbers

Convenient for **automatisation**, writing scripts executed on the **server**, downloading **large datasets** for analysis

Usable for beginners as well as for more advanced

Objectives

Pfam page scraping

Finding Biopython tools compatible with downloaded data

Wrapping HMMER tools : hmmsearch, hmmpress and hmmscan

The available tools

There is neither **biopython** module for pfam database nor any other professional program.

(like Entrez for NCBI and any other popular and developed tool for our purpose)

The programing aproach

object-oriented programming

easy installing by **pip**

imports just **a few modules**

the long-term goal is to make our package a module of **Biopython**

Technologies

GitHub

python™ Package Index

biopython

HMMER

Sphinx

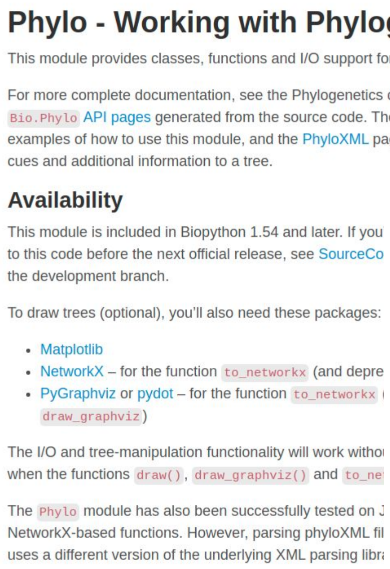
beautifulsoup4

The code

PARS + Biopython

Compatibile with biopython formats:

- Rfam/Pfam sequences
  - Biopython SeqIO format
- Phylogenetic tree
  - Biopython Phylo format



Phylo - Working with Phylogenetic Trees

This module provides classes, functions and I/O support for working with phylogenetic trees.

For more complete documentation, see the Phylogenetics chapter of the Biopython Tutorial and the `Bio.Phylo` API pages generated from the source code. The Phylo cookbook page has more examples of how to use this module, and the PhyloXML page describes how to attach graphical cues and additional information to a tree.

Availability

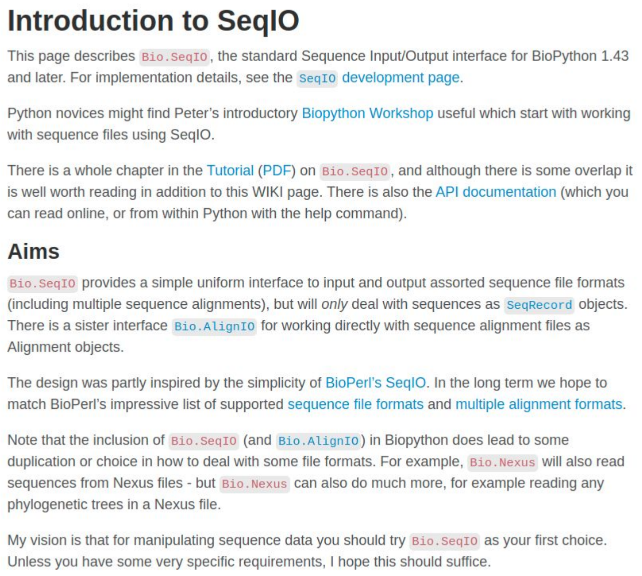
This module is included in Biopython 1.54 and later. If you to this code before the next official release, see SourceCo the development branch.

To draw trees (optional), you'll also need these packages:

- Matplotlib
- NetworkX – for the function `to_networkx` (and depre
- PyGraphviz or pydot – for the function `to_networkx` `draw_graphviz`)

The I/O and tree-manipulation functionality will work witho when the functions `draw()`, `draw_graphviz()` and `to_ne`

The `Phylo` module has also been successfully tested on J NetworkX-based functions. However, parsing phyloXML fil uses a different version of the underlying XML parsing libr



Introduction to SeqIO

This page describes `Bio.SeqIO`, the standard Sequence Input/Output interface for BioPython 1.43 and later. For implementation details, see the `SeqIO` development page.

Python novices might find Peter's introductory Biopython Workshop useful which start with working with sequence files using SeqIO.

There is a whole chapter in the Tutorial (PDF) on `Bio.SeqIO`, and although there is some overlap it is well worth reading in addition to this WIKI page. There is also the API documentation (which you can read online, or from within Python with the help command).

Aims

`Bio.SeqIO` provides a simple uniform interface to input and output assorted sequence file formats (including multiple sequence alignments), but will *only* deal with sequences as `SeqRecord` objects. There is a sister interface `Bio.AlignIO` for working directly with sequence alignment files as Alignment objects.

The design was partly inspired by the simplicity of BioPerl's SeqIO. In the long term we hope to match BioPerl's impressive list of supported sequence file formats and multiple alignment formats.

Note that the inclusion of `Bio.SeqIO` (and `Bio.AlignIO`) in Biopython does lead to some duplication or choice in how to deal with some file formats. For example, `Bio.Nexus` will also read sequences from Nexus files - but `Bio.Nexus` can also do much more, for example reading any phylogenetic trees in a Nexus file.

My vision is that for manipulating sequence data you should try `Bio.SeqIO` as your first choice. Unless you have some very specific requirements, I hope this should suffice.

Pfam family class

One class to get all information

Methods for tree, alignments and domains architectures downloading

Can be replaced by individual function

Use most default formats

```
Globin = PfamFamily(‘PF00042’)


Output:
{'db': 'pfam',
 'access': 'PF00042',
 'short_name': 'Globin',
 'type': 'Domain',
 'seed_len': 73,
 'full_len': 10097,
 'avarage_len': 99.6,
 'avarage_id': 21.0,
 'avarage_coverage': 37.14,
 'changestatus': 'Changed',
 'description': 'Globin',
 'go_ref': ['GO:0020037'],
 'so_ref': ['SO:0000417'],
 'pubmed_ref': ['3656444', '6292840', '2448639', '9108146'],
 'pdb_ref': ['3G4W',
  '3TM9',
  '3MOU',
    '1G09',...]}
```

Rfam family class

```
Riboswitch=RfamFamily('RF01739')


Output:
  {'db': 'rfam',
   'access': 'RF01739',
   'short_name': 'glnA',
   'type': 'family',
   'go_ref': ['GO:0070406'],
   'so_ref': ['SO:0000035'],
   'pubmed_ref': ['18787703', '20230605', '21282981'],
   'pdb_ref': ['5DDR', '5DDQ', '5DDP']}
```

xfam_to module

- Download references to PDB, GO, SO or PubMed without creating whole family object
- Works for pfam and rfam entries

```
globin_pubmed =
pfam_to_pubmed('PF00042')


Output:
['3656444', '6292840',
'2448639', '9108146']


riboswitch_pdb =
rfam_to_pdb('RF01739')


Output:
['5DDR', '5DDQ', '5DDP']
```

hmm_download module

convenient downloading of hmm profiles

```
import hmm_download
list = hmm_download.load_data("data/pfam-seq.csv")
families = hmm_download.get_names(list)
hmm_download.download_hmm(families, "hmm_folder")

family = ["PF00042", "PF00002"]
hmm_download.download_hmm(family, "hmm_folder")
```

hmm_file module

implements class for parsing hmm files

hammer_to_object module

implements function for parsing file to an object of the HMMERProfileFileBuilder class

```
import hammer_to_object
hmmObj = hammer_to_object.file_to_object("hmm_folder/PF00042.hmm")
print(hmmObj.get_length())
Output:
111
#modify
m = {'A': 2.61238, 'C': 4.6652, 'D': …
i = {'A': 2.68618, 'C': 4.42225, 'D': ...
t = {('m', 'm'): 0.36202, ('m', 'i'): 5.18438, ...

hmmObj.add_position(m, i, t)
print(hmmObj.get_length())
Output:
112
#generate file
hmmObj.file_format("hmm_folder/test.hmm")
```

hmm_command module

wrapped tools from HAMMER: hmmpress, hmmscan and hmmsearch

hmm_autosearch module

perform hmm search all vs all (all hmm vs all fasta)

```
import hmmer_command
hmmer_command.hmmsearch(o="out/hem.hmmsearchout",
hmm_file="hmm_folder/PF00042.hmm",
fasta_file="fasta_folder/example_hem.fasta")


import hmm_autosearch
hmm_autosearch.automatic_search("hmm_folder",
"fasta_folder", "outsearch", "search")
```

results

The python3 **package** for **easy browsing** and **downloading data**: family sequences, alignments, trees, hmm profiles and family descriptions.

Compatible with **biopython modules** and with wrapped **HMMER tools**: hmmsearch, hmmpress and hmmscan.

Convenient for detailed analysis of **protein family**

Conclusions and future

- more advanced search
- write modules for other families databases
- display images e.g RNA structure, domains architecture
- improve code to be a part of Biopython