Ideas for Biopython 2.0

Release

Patrick Kunzmann

Jun 17, 2017

Biopython is a wonderful package, providing a variety of tools that make the life of computational biologists a lot easier. Unfortunately *Biopython* is also very crowded: There is a mass of subpackages with inconsistent naming schemes, some of them not really updated for 10 years. The documentation styles reach from *rst* over *numpydoc* to standard python doc.

I want to propose a community effort to 'tidy up' *Biopython*: Removing deprecated subpackages, harmonizing naming and documentation schemes, sorting modules, updating for modern scientific Python. The following sections will contain the proposed changes. Since those drastic changes would make the new version completely incompatible with existing *Biopython* versions, the new version is *Biopython 2.0*. Of course, everything is open for discussion. The development could happen in a dedicated *Biopython* branch or alternatively in a new repository. The development and release of version 2.0 should be parallel to version 1.x, since 1.x should be supported for some further years for compatibilty reasons.

I know, this endeavour might be a lot of work, but if enough of us Biopythoneers work on this, we might finish it in a few months. So, now let's begin with the suggested changes.

0.1 Deprecation of Python 2.x support

Since Python 3 is now released for almost 10 years, I propose to take the opportunity of not caring for backwards compatibility and deprecate the Python 2.x support.

0.2 numpy, scipy and matplotlib as dependencies

numpy enables convenient and fast data handling in Python and is one of the reasons why Python is so popular in scientific computing. *scipy* adds a lot of functionality to *numpy* and *matplotlib* is a popular tool for plotting. Since all 3 packages are nearly a must-have in scientific usage of Python (e.g. bundled in *Anaconda*), I would suggest that these packages are required for using *Biopython 2.0*.

0.3 Usage of numpydoc and Sphinx

numpydoc is used by *numpy* for its documentation (surprise!). It is based on *reStructuresText* and can be interpreted by *Sphinx* via the *numpy* extension. Since *Biopython* lacks a uniform documentation format, I suggest the usage of *numpydoc*. Since the style is well defined, we do not have to invent a documentation style ourselves. Using *Sphinx* for creating our documentation, we also have the opportunity to put some other ressources into the documentation (extended examples, tutorials, etc.). We could also build the entire *Biopython* website this way, so everything is in one place.

0.4 Lower case package and module naming

Since we do not have to care for compatibility, we can now name all packages and modules uniformly in lower case.

0.5 Removal of outdated subpackages

Biopython contains a lot of subpackages, that are not really maintained anymore. I suggest we try to look for new 'owners' (via the mailing list) for these packages, but otherwise remove them in *Biopython 2.0*.

0.6 Module placement and imports

For clarity reason, the top-level package (bio), should not contain any modules, but only supackages. These packages can contain modules or subpackages (folders) by themselves. For convenient import of classes and functions `__init__.py` imports all attributes in the modules in this package:

```
from .module1 import *
from .module2 import *
```

Attributes of modules that are considered package private are not imported. Also modules of subpackages of this package are not imported. As *Biopython* user you now can import all attributes in the following ways:

```
from bio.package1 import attribute1, attribute2
import bio.package2 as pck2
from bio.package3.subpackage import attribute3
```

This way *attribute1*, *attribute2* and *attribute3* must not be in `__init__.py`, but can be in any module in the same package. `__init__.py` itself only contains the imports and not any attributes, for clarity reasons.

0.7 Package organisation

On order to clarifiy the *Biopython* structure, the following section contains a proposal for the new project structure:

```
doc/
tests/
scripts/
bio/
        sequence/
                align/
                motifs/
                restriction/
                seq.py
                ...
        structure/
        files/
                pdb/
                pdbx/
                fasta/
                genbank/
                file.py
                ...
        database/
                pdb/
                expasy/
                entrez/
                ...
        phylo/
        app/
                blast/
                clustal/
                app.py
                local.py
                online.py
                container.py
                ...
```

```
        sql/
        misc/
        __init__.py
setup.py
...
```

0.7.1 doc

This contains all documentation parts that are not included in the *.py files. If we want to build a website entirely based on *Sphinx*, we could also put the website content here.

0.7.2 tests

As until now, this folder contains unit tests for *Biopython 2.0*. But like the rest of the project, this should get a clearer structure. Maybe this should get the same folder structure as the *Bio* package where applicable. Additional folders could contain the test files (pdb, fasta, etc.).

0.7.3 sequence

This subpackage contains everything concerning sequences, including the *Seq* class itself. Since motifs, alignments and restriction are some kind of special applications, those are available in subpackages.

A possible change could also be to exchange the internal string containing the sequence for a Python byte array, since one byte is sufficient to represent letters in every *Alphabet*. Alternatively we could also use *numpy ndarray* with byte type. This could be more efficient in some applications, but will also cause more work in the conversion of the current *Seq* class.

0.7.4 structure

This subpackage holds everything regarding molecular 3D structures. Mostly it has the same function as the old *Bio.PDB* package, without the read/write part. Since *numpy* is planned to be a requirement for *Biopython 2.0*, I suggest to switch the data model over to my *numpy* array based implementation (padix-key/biopython), which enables more efficient data handling.

0.7.5 files

This subpackage contains read/write functionality for all kinds of biological file formats. For each file format or group of file formats there is a separate subpackage, since a user often won't handle a lot of different file types. The file formats are put into a separate package (rather than beeing in structure or sequence), because some file formats contain structural and sequence information and some file types provide information not represented at all by a *Biopython 2.0* subpackage.

The read and write functionality for a specific file type is combined in one class inheriting from the abstract base class *File* in `file.py`. This approach has the advantage, that the user can read the file, change partially the content and write it back into the file. Example:

```
pdb_file = PDBFile()
pdb_file = load("file.pdb")
atom_array = pdb_file.get_structure()
atom_array = translate(atom_array, (1,2,3))
```

```
pdb_file.set_structure(atom_array)
pdb_file.save("file2.pdb")
```

In this case the structural information of the PDB file has changed, but the rest of the information (header, secondary structure information, etc.) is preserved.

0.7.6 database

This subpackage contains functionality for interaction with online databases, while each database has its own subpackage.

0.7.7 phylo

This subpackage has the content of the current *Bio.Phylo*.

0.7.8 app

This subpackage contains interfaces for usage of external software (e.g. for alignments). Each interface for external tools inherits from one of the folowing abstract base classes, which in turn inherit from *Application*:

OnlineApp: The *Application* is using an online tool, e.g. NCBI BLAST.

LocalApp: The *Application* interacts via command line with preinstalled software.

ContainerApp: The *Application* interacts via command line with a *Docker* container. This approach might be useful for software where installation can be complex.

0.7.9 sql

This subpackage has the content of the current *BioSQL*.

0.7.10 misc

This subpackage contains all subpackages, which do not fit in the upper categories.