Biomedical Informatics: Lecture 2

M. Ceccanti and A. Paoluzzi

Tue, Mar 11, 2014

Outline: Syllabus, GitHub

Documentation Tools

Literate programming

The Biomedical informatics 2014 is a project-oriented course, whose aim is to design and implement a set of coordinated student projects, using the best collaboration (GitHub) and documentation tools and methods (literate programming).

Navigate the links below to get a feeling of:

- Definition (From Wikipedia:) Literate programming

Literate programming

The Biomedical informatics 2014 is a project-oriented course, whose aim is to design and implement a set of coordinated student projects, using the best collaboration (GitHub) and documentation tools and methods (literate programming).

Navigate the links below to get a feeling of:

- Definition (From Wikipedia:) Literate programming
- Resources http://www.literateprogramming.com/

Literate programming

The Biomedical informatics 2014 is a project-oriented course, whose aim is to design and implement a set of coordinated student projects, using the best collaboration (GitHub) and documentation tools and methods (literate programming).

Navigate the links below to get a feeling of:

- Definition (From Wikipedia:) Literate programming
- Resources http://www.literateprogramming.com/

Literate programming

The Biomedical informatics 2014 is a project-oriented course, whose aim is to design and implement a set of coordinated student projects, using the best collaboration (GitHub) and documentation tools and methods (literate programming).

Navigate the links below to get a feeling of:

- Definition (From Wikipedia:) Literate programming
- Resources http://www.literateprogramming.com/

We are going to integrate:

1. nuweb

Literate programming

The Biomedical informatics 2014 is a project-oriented course, whose aim is to design and implement a set of coordinated student projects, using the best collaboration (GitHub) and documentation tools and methods (literate programming).

Navigate the links below to get a feeling of:

- Definition (From Wikipedia:) Literate programming
- Resources http://www.literateprogramming.com/

We are going to integrate:

1. nuweb
2. LaTeX

Literate programming

The Biomedical informatics 2014 is a project-oriented course, whose aim is to design and implement a set of coordinated student projects, using the best collaboration (GitHub) and documentation tools and methods (literate programming).

Navigate the links below to get a feeling of:

- Definition (From Wikipedia:) Literate programming
- Resources http://www.literateprogramming.com/

We are going to integrate:

1. nuweb
2. LaTeX
3. Pandoc

Literate programming

The Biomedical informatics 2014 is a project-oriented course, whose aim is to design and implement a set of coordinated student projects, using the best collaboration (GitHub) and documentation tools and methods (literate programming).

Navigate the links below to get a feeling of:

- Definition (From Wikipedia:) Literate programming
- Resources http://www.literateprogramming.com/

We are going to integrate:

1. nuweb
2. LaTeX
3. Pandoc
4. python and/or javascript and/or C

Latex

LaTeX is a high-quality typesetting system; it includes features designed for the production of technical and scientific documentation.

LaTeX is the de facto standard for the communication and publication of scientific documents.

LaTeX is available as free software.

Markdown

Pandoc understands an extended and slightly revised version of John Gruber's markdown syntax.

This document explains the syntax, noting differences from standard markdown.

Except where noted, these differences can be suppressed by using the `markdown_strict` format instead of markdown.

Pandoc

Pandoc is a universal document converter

Bioinformatic Tools in Python

Python

you have of course already installed your python environment

Otherways:

`www.python.org`

Choose python version 2.7 (NOT python 3.x) for compatibility with most python packages

Biopython

```
$ ipython
Python 2.7.5 (default, Aug 25 2013, 00:04:04)
Type "copyright", "credits" or "license" for more information.

IPython 0.14.dev -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: import Bio

In [2]:
```

General Overview

The Biopython Project is an international association of developers of freely available Python (http://www.python.org) tools for computational molecular biology.

Biopython aims to make it as easy as possible to use Python for bioinformatics by creating high-quality, reusable modules and classes.

(from Biopython Tutorial and Cookbook)

General Overview

The Biopython Project is an international association of developers of freely available Python (http://www.python.org) tools for computational molecular biology.

Biopython aims to make it as easy as possible to use Python for bioinformatics by creating high-quality, reusable modules and classes.

(from Biopython Tutorial and Cookbook)

Assignment

- check if you already have

General Overview

The Biopython Project is an international association of developers of freely available Python (http://www.python.org) tools for computational molecular biology.

Biopython aims to make it as easy as possible to use Python for bioinformatics by creating high-quality, reusable modules and classes.

(from Biopython Tutorial and Cookbook)

Assignment

- check if you already have
- download if needed

Sequence Objects

The central object in bioinformatics is the sequence.

Most of the time when we think about sequences we have in my mind a string of letters like 'AGTACACTGGT'. You can create such Seq object with this sequence as follows - the ">>>" represents the Python prompt followed by what you would type in:

```
>>> from Bio.Seq import Seq
>>> my_seq = Seq("AGTACACTGGT")
>>> my_seq
Seq('AGTACACTGGT', Alphabet())
>>> print(my_seq)
AGTACACTGGT
>>> my_seq.alphabet
Alphabet()
```

Working with sequences

What we have here is a sequence object with a generic alphabet .

In addition to having an alphabet, the Seq object differs from the Python string in the methods it supports. You can't do this with a plain string:

```
>>> my_seq
Seq('AGTACACTGGT', Alphabet())
>>> my_seq.complement()
Seq('TCATGTGACCA', Alphabet())
>>> my_seq.reverse_complement()
Seq('ACCAGTGTACT', Alphabet())
```

(from Biopython Tutorial and Cookbook)

Sequence annotation objects

Immediately "above" the Seq class is the Sequence Record or `SeqRecord` `class`, defined in the `Bio.SeqRecord` module. This class allows higher level features such as identifiers and features (as `SeqFeature` objects) to be associated with the sequence, and is used throughout the sequence input/output interface `Bio.SeqIO`

If you are only going to be working with simple data like FASTA files, you can probably skip this chapter for now. If on the other hand you are going to be using richly annotated sequence data, say from GenBank or EMBL files, this information is quite important.

(from Biopython Tutorial and Cookbook)

The SeqRecord object

.seq The sequence itself, typically a Seq object.

(from Biopython Tutorial and Cookbook)

The SeqRecord object

.seq The sequence itself, typically a Seq object.

.id The primary ID used to identify the sequence – a string.

(from Biopython Tutorial and Cookbook)

The SeqRecord object

.seq The sequence itself, typically a Seq object.

.id The primary ID used to identify the sequence – a string.

.name A "common" name/id for the sequence – a string. It could also be a clone name.

(from Biopython Tutorial and Cookbook)

The SeqRecord object

.seq	The sequence itself, typically a Seq object.
.id	The primary ID used to identify the sequence – a string.
.name	A "common" name/id for the sequence – a string. It could also be a clone name.
.description	A human readable description or expressive name for the sequence – a string.

(from Biopython Tutorial and Cookbook)

The SeqRecord object

.seq The sequence itself, typically a Seq object.

.id The primary ID used to identify the sequence – a string.

.name A "common" name/id for the sequence – a string. It could also be a clone name.

.description A human readable description or expressive name for the sequence – a string.

.letter_annotations Holds per-letter-annotations using a (restricted) dictionary of additional information about the letters in the sequence. The keys are the name of the information, and the information is contained in the value as a Python sequence (i.e. a list, tuple or string) with the same length as the sequence itself. This is often used for quality scores or secondary structure information

(from Biopython Tutorial and Cookbook)

The SeqRecord object

.seq	The sequence itself, typically a Seq object.
.id	The primary ID used to identify the sequence – a string.
.name	A "common" name/id for the sequence – a string. It could also be a clone name.
.description	A human readable description or expressive name for the sequence – a string.
.letter_annotations	Holds per-letter-annotations using a (restricted) dictionary of additional information about the letters in the sequence. The keys are the name of the information, and the information is contained in the value as a Python sequence (i.e. a list, tuple or string) with the same length as the sequence itself. This is often used for quality scores or secondary structure information
.annotations	A dictionary of additional information about the sequence. The keys are the name of the information, and the information is contained in the value. This allows the addition of more "unstructured" information to the sequence.

(from Biopython Tutorial and Cookbook)

The SeqRecord object

.seq	The sequence itself, typically a Seq object.
.id	The primary ID used to identify the sequence – a string.
.name	A "common" name/id for the sequence – a string. It could also be a clone name.
.description	A human readable description or expressive name for the sequence – a string.
.letter_annotations	Holds per-letter-annotations using a (restricted) dictionary of additional information about the letters in the sequence. The keys are the name of the information, and the information is contained in the value as a Python sequence (i.e. a list, tuple or string) with the same length as the sequence itself. This is often used for quality scores or secondary structure information
.annotations	A dictionary of additional information about the sequence. The keys are the name of the information, and the information is contained in the value. This allows the addition of more "unstructured" information to the sequence.
.features	A list of `SeqFeature` objects with more structured information about the features on a sequence (e.g. position of genes on a genome, or domains on a protein sequence).

(from Biopython Tutorial and Cookbook)

The SeqRecord object

.seq	The sequence itself, typically a Seq object.
.id	The primary ID used to identify the sequence – a string.
.name	A "common" name/id for the sequence – a string. It could also be a clone name.
.description	A human readable description or expressive name for the sequence – a string.
.letter_annotations	Holds per-letter-annotations using a (restricted) dictionary of additional information about the letters in the sequence. The keys are the name of the information, and the information is contained in the value as a Python sequence (i.e. a list, tuple or string) with the same length as the sequence itself. This is often used for quality scores or secondary structure information
.annotations	A dictionary of additional information about the sequence. The keys are the name of the information, and the information is contained in the value. This allows the addition of more "unstructured" information to the sequence.
.features	A list of `SeqFeature` objects with more structured information about the features on a sequence (e.g. position of genes on a genome, or domains on a protein sequence).
.dbxrefs	A list of database cross-references as strings.

(from Biopython Tutorial and Cookbook)

Sequence Input/Output

The `Bio.SeqIO` moduleaims to provide a simple interface for working with assorted sequence file formats in a uniform way.

```
>>> from Bio import SeqIO
>>> help(SeqIO)
```

(from Biopython Tutorial and Cookbook)

References

Reference

1. Biopython Tutorial and Cookbook