AmpliPy

John Chen

Overview

AmpliPy is a Python implementation of the primer binding and PCR analysis methods of Amplify 4 (https://github.com/wrengels/Amplify4). Given one or more primers and a DNA template, the program conducts a search for possible binding sites for each primer to the template, then tests the binding sites for formation of amplicons. AmpliPy does not implement the primer Tm or dimerization calculations features from Amplify 4.

The calculation of primer binding follows the primability and stability measures stated in the Amplify 4 help menu. AmpliPy's calculations tend to be a bit higher than Amplify 4's numbers, possibly due to differences in rounding between the two methods.

Usage

Installation

AmpliPy requires an installation of Python 3.6 or higher. No other installation steps are required.

Data preparation

AmpliPy takes two input files, one specifying the primers to be analyzed and another that specifies DNA template for the PCR. The DNA template is simply a plain text file with just DNA (no headers or annotations). The primer file follows the format of Amplify 4, with primer sequences in the first column and the primer names in the second column. See the example template and primer files for more information.

Command line

To use AmpliPy from the command line, simply run the 'amplipy.py' file using Python and supply the template file and the primer file as the first two arguments. Then, specify the primers to analyze by specifying a list (separated by spaces) of primer names (-n) or primer list positions (-p, numbering starts from 1).

For example, these two commands are equivalent.

```
python   amplipy.py   example_template.txt   example_primers.txt   -p   1   3
python   amplipy.py   example_template.txt   example_primers.txt   -n   TEM1-fwd  B1
```

By default, the template DNA is treated as linear. To specify a circular template, use the '-c' option.

```
python   amplipy.py    example_template.txt    example_primers.txt   -p   1   3   -c
```

To send the output to a file instead of the command line, specify a file with the '-o' option.

```
Python   amplipy.py    example_template.txt    example_primers.txt   -p   1   3   -c   -o
test_result.txt
```

Running from another Python script

To conduct the analysis from another script, import the ‘MakePrimer’ and ‘PCR’ functions from AmpliPy. See ‘import_example.py’ for a full example.

```
from amplipy import MakePrimer, PCR, PrimerSearch


f_primer = MakePrimer("TTCAAATATGTATCCGCTCATGAGACAAT","TEM1-fwd")

r_primer = MakePrimer("CCTTTTGCTGGCCTTTTGCTCC","B1")

fwd, rev = PrimerSearch(f_primer, template, circular=True, silent=True)

pdt = PCR([f_primer, r_primer],template, circular=True, silent=True)
```

MakePrimer() takes a DNA sequence and optionally a label to create a primer object. Lists of one or more primer objects can then be fed to the PCR() function as the first argument, followed by the template DNA and a boolean indicating whether the template is circular.

The output of PCR() is a list of potential PCR products, each represented by a dictionary that includes their sequence (under the key ‘seq’), the amplicon quality (quality), the description of the PCR strength (q_desc) and the primer binding sites that generated the product (f_site and r_site).

Each primer binding site is also a dictionary containing information on the binding site, including the position of the 3’ end (pos), the primer binding stats (primability, stability and quality), the direction of binding relative to the template (dir) and the information on the binding context which can be printed for visualization (binding).

To search for primer binding sites without testing PCR products, the user can make direct use of the PrimerSearch() function by providing the primer object, the template and a boolean on the circularity of the template as arguments. The outputs are the primer binding sites in the fwd and rev direction, in the same output format as stated above.

By default, PrimerSearch() and PCR() will still print their output. To suppress this behavior, set the optional ‘silent’ argument to True.