# 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 PCR products are simply a list of possibilities given a set of primer binding sites. AmpliPy does not conduct actual PCR simulations.

The calculation of primer binding follows the primability and stability measures stated in the Amplify 4 help menu. In simple terms, the method checks how well the 3' end of a primer matches any potential binding sites in the template. 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 list positions (-p, numbering starts from 1).

For example, the following code tests using the 1st and 3rd primers in the primer file.

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

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.

```python
from amplipy import Primer, Template, PCR_Manager, Visuals

# define the template
template_seq = "TTAAGGAGCAAAAGGCCAGCAAAAGGCCAGGAACCGTAAAAAGGCCGCGTTGCTGGCGTTTTTCC
ATAGGCTCCGCCCCCCTGACGAGCATCACAAAAATCGACGCTCAAGTCAGAGGTGGCGAAACCCGACAGGACTATAAAGAT
ACCAGGCGTTTCCCCCTGGAAGCTCCCTCGTGCGCTCTCCTGTTCCGACCCTGCCGCTTACCGGATACCTGTCCGCCTTTC
TCCCTTCGGGAAGCGTGGCGCTTTCTCATAGCTCACGCTGTAGGTATCTCAGTTCGGTGTAGGTCGTTCGCTCCAAGCTGG
GCTGTGTGCACGAACCCCCCGTTCAGCCCGACCGCTGCGCCTTATCCGGTAACTATCGTCTTGANNCCAACCCGGTAAGAC
ACGACTTATCGCCACTGGCAGCAGCCACTGGTAACAGGATTAGCAGAGCGAGGTATGTAGGCGGTGCTACAGAGTTCTTGA
AGTGGTGGCCTAACTACGGCTACACTAGAAGGACAGTATTTGGTATCTGCGCTCTGCTNNAGCCAGTTACCTTCGGAAAAA
GAGTTGGTAGCTCTTGATCCGGCAAACAAACCACCGCTGGTAGCGGTGGTTTTTTTGTTTGCAAGCAGCAGATTACGCGCA
GAAAAAAAGGATCTCAAGAAGATCCTTTGATCTTTTCTACGGGGTCTGACGCTCAGTGGAACGAAAACTCACGTTAAGGGA
TTTTGGTCATGAGACTAGTTCGGCCTATTGGTTAAAAAATGAGCTGATTTAACAAAAATTTTAACAAAATTCACGCCCCGC
CCTGCCACTCATCGCAGTACTGTTGTAATTCATTAAGCATTCTGCCGACATGGAAGCCATCACAAACGGCATGATGAACCT
GAATCGCCAGCGGCATCAGCACCTTGTCGCCTTGCGTATAATATTTGCCCATAGTGAAAACGGGGGCGAAGAAGTTGTCCA
TATTGGCCACGTTTAAATCAAAACTGGTGAAACTCACCCAGGGATTGGCTGAGACGAAAAACATATTCTCAATAAACCCTT
TAGGGAAATAGGCCAGGTTTTCACCGTAACACGCCACATCTTGCGAATATATGTGTAGAAACTGCCGGAAATCGTCGTGGT
ATTCACTCCAGAGCGATGAAAACGTTTCAGTTTGCTCATGGAAAACGGTGTAACAAGGGTGAACACTATCCCATATCACCA
GCTCACCGTCTTTCATTGCCATACGTAATTCCGGATGAGCATTCATCAGGCGGGCAAGAATGTGAATAAAGGCCGGATAAA
ACTTGTGCTTATTTTTCTTTACGGTCTTTAAAAAGGCCGTAATATCCAGCTGAACGGTCTGGTTATAGGTACATTGAGCAA
CTGACTGAAATGCCTCAAAATGTTCTTTACGATGCCATTGGGATATATCAACGGTGGTATATCCAGTGATTTTTTTCTCCA
TGCGAAACGATCCTCATCCTGTCTCTTGATCAGAGCTTGATCCCCTGCGCCATCAGATCCTTGGCGGCGAGAAAGCCATCC
AGTTTACTTTGCAGGGCTTCCCAACCTTACCAGAGGGCGCCCCAGCTGGCAATTCCGGTGACGTCAGGTGGCACTTTTCGG
GGAAATGTGCGCGGAACCCCTATTTGTTTATTTTTCTAAATACATTCAAATATGTATCCGCTCATGAGACAATAACCCTGA
TAAATGCTTCAATAATATTGAAAAAGGAAGCCCATGGGATTCAAACTTTTGAGTAAGTTATTGGTCTATTTGACCGCGTCT
ATCATGGCTATTGCGAGCCCGCTCGCTTTTTCCGTAGATTCTAGCGGAGAATATCCGACAGTCAGCGAAATTCCGGTCGGG
GAGGTCCGGCTTTACCAGATTGCCGATGGTGTTTGGTCGCATATCGCAACGCAGTCGTTTGATGGCGCAGTCTACCCGTCC
AATGGTCTCATTGTCCGTGATGGTGATGAGTTGCTTTTGATTGATACAGCGTGGGGTGCGAAAAACACAGCGGCACTTCTC
GCGGAGATTGAGAAGCAAATTGGACTTCCTGTAACGCGTGCAGTCTCCACGCACTTTCATGACGACCGCGTCGGCGGCGTT
GATGTCCTTCGGGCGGCTGGGGTGGCAACGTACGCATCACCGTCGACACGCCGGCTAGCCGAGGTAGAGGGGAACGAGATT
CCCACGCACTCTCTTGAAGGACTTTCATCGAGCGGGGACGCAGTGCGCTTCGGTCCAGTAGAACTCTTCTATCCTGGTGCT
GCGCATTCGACCGACAACTTAATTGTGTACGTCCCGTCTGCGAGTGTGCTCTATGGTGGTTGTGCGATTTATGAGTTGTCA
CGCACGTCTGCGGGGAACGTGGCCGATGCCGATCTGGCTGAATGGCCCACCTCCATTGAGCGGATTCAACAACACTACCCG
GAAGCACAGTTCGTCATTCCGGGGCACGGCCTGCCGGGCGGTCTTGACTTGCTCAAGCACACAACGAATGTTGTAAAAGCG
CACACAAATCGCTCAGTCGTTGAGTAACTCGAGAAGCTTGTCACCAAGTTTACTCATATATACTTTAGATTGATTTAAAAC
TTCATTTTTAATTTAAAAGGATCTAGGTGAAGATCCTTTTTC"
template = Template(template_seq, is_circular=True)

# define primers
primer1 = Primer("TTCAAATATGTATCCGCTCATGAGACAAT", "TEM1-fwd")
primer2 = Primer("CCTTTTGCTGGCCTTTTGCTCC", "B1")
```

```


# to check just for binding sites
primer1_binding_sites = PCR_Manager.test_binding_on_template(primer1, template)
primer2_binding_sites = PCR_Manager.test_binding_on_template(primer2, template)

# show the parameters available in the data output
print(primer1_binding_sites)
print(primer1_binding_sites['fwd'][0]) # each primer is in a BindingResult datacl
ass
# use the Visuals class to print formatted binding sites
Visuals.print_primer_sites(primer1_binding_sites)

# to check for potential products when given a list of all binding sites
pdt_list = PCR_Manager.predict_products([primer1_binding_sites, primer2_binding_s
ites])
for pdt in pdt_list:
    print(pdt.visualization())


# To conduct the whole process and print the results in one command
PCR_Manager.PCR( [primer1, primer2], template, show_DNA=True, show_pdt=True )
```

*Primer*() takes a DNA sequence and optionally a label to create a primer object. Similarly, *Template*() takes a sequence and creates a template, with an optional '*is_circular*' parameter; templates are treated as linear by default.

Lists of one or more *Primer* objects can then be fed to the *PCR_Manager. test_binding_on_template*() function as the first argument, followed by the template.

The output of *PCR_Manager. test_binding_on_template*() is a dictionary of potential forward (key = 'fwd') and reverse (key = 'rev') primer binding sites, each represented by a '*BindingResult*' object. The binding site dictionary can be visualized using the *Visuals.print_primer_sites*() function. Each *BindingResult* contains information, including the position of the 3' end (pos), the primer binding stats (primability, stability and quality), whether the direction of binding is reversed relative to the template (reverse).

A list of binding site dictionaries can be supplied to the *PCR_Manager.predict_products*() function to check for potential PCR products. The sites need to share the same template. Each product is represented by a *PCR_Product* object that includes their sequence (under the key 'seq') and the primer binding sites that generated the product (f_site and r_site).

Use the *PCR_Manager.PCR*() function to simultaneously conduct a search of potential primer binding sites on a template and list possible products. The output is directly printed to the terminal.