Primer Design Pipeline

User Manual

Updated January 2017

Pathogen and Microbiome Institute

Josh Gutman | jng86@nau.edu

Table of Contents

Installation

1. Navigate to the directory where you want the program.

2. Clone the repository using Git.

   Using HTTPS:
   ```
   git clone https://github.com/JoshGutman/Primer_Design_Pipeline.git
   ```

   Using SSH:
   ```
   git clone git@github.com:JoshGutman/Primer_Design_Pipeline.git
   ```

3. Alternatively, if you are working on NAU's Monsoon, the program can be found in

   ```
   /scratch/jng86/primer_design_pipeline/Primer_Design_Pipeline
   ```

Dependencies

Dependency	**Monsoon Module**
Python 3.5 or later	`module load anaconda/3.latest`
BioPython	`module load anaconda/3.latest`
BLASTN	`module load blast+/2.2.29`
MUSCLE	`module load muscle/3.8.31`
Primer3	Included in the repository

Configuration

Target Multifasta

The target multifasta is a multifasta file (.fasta) that contains all of the genes that you want primers for. The format should be the following:

```
>Gene1
ACGTACGTACGT
>Gene2
GCTAGCTAGCTAGCTAGCTA
>Gene3
GTCAGCTAGCTAGCATGTCAGCTAGCTGTCGA
```

And so on for each gene you want. For more information, see https://en.wikipedia.org/wiki/FASTA_format

Reference Directory

The reference directory is a directory that contains all of the .fasta files that you want to look for primers in, as well as the outgroup .fasta files. Do not include any other file types other than .fasta here.

Primer3 Configuration File

The primer3 configuration file is a plain text file that contains metadata for primer3. A template can be found in

```
Primer_Design_Pipeline/tools/primer3_template.txt
```

This template includes default values for all information necessary for primer3 to run, meaning that you don't need to change anything to use it. Some of the important information that you may or may not want to change includes:

```
PRIMER_OPT_SIZE
PRIMER_MIN_SIZE
PRIMER_MAX_SIZE
PRIMER_MIN_TM
PRIMER_OPT_TM
PRIMER_MAX_TM
```

Much of the information will be filled out automatically by the program, including:

```
SEQUENCE_ID
SEQUENCE_TEMPLATE
PRIMER_TM_FORMULA
PRIMER_SALT_MONOVALENT
PRIMER_INTERNAL_SALT_MONOVALENT
PRIMER_SALT_DIVALENT
PRIMER_INTERNAL_SALT_DIVALENT
PRIMER_DNTP_CONC
PRIMER_INTERNAL_DNTP_CONC
PRIMER_SALT_CORRECTIONS
PRIMER_DNA_CONC
PRIMER_INTERNAL_DNA_CONC
```

For more information, see
http://primer3.sourceforge.net/primer3_manual.htm#globalTags

Genome Target List

The genome target list is a list of all the genome names that you want to look in for primers in. Primer Design Pipeline allows you to look for primers in both desired genomes and undesired genomes. The reference directory will include the desired genomes and the undesired genomes (known as “outgroups”). To help the program differentiate between the two, you must include a file with all the names of the desired genome names. The utility program
`Primer_Design_Pipeline/tools/make_target_list.py`
automates this process. It is discussed later in the Tools section.

Ordering Sheets

Primer Design Pipeline can generate order sheets with most of the information required to order primers. Using these order sheets, it can also check for conflicts in an already-established PCR multiplex. The order sheets are required to be in a certain format. Deviations from this format will cause errors. The sheets can be generated using the utility program
`Primer_Design_Pipeline/tools/generate_order_sheet.py`
It is discussed later in the Tools section.

Running Primer Design Pipeline

Flags

Required:

Short	**Long**	**Description**
-t	--target	Path to target multifasta
-d	--directory	Path to reference directory
-c	--config	Path to primer3 config file
-g	--genome	Path to genome target list

Optional:

Short	**Long**	**Description**	**Default**
-r	--reference	Path to amplicon reference assembly	All fastas in reference directory
-l	--lower	Lower bound of desired amplicon size	150
-u	--upper	Upper bound of desired amplicon size	250
-i	--ignore	Threshold percentage to consider SNP a degeneracy	98.0
-ol	--oligo_conc	Oligo concentration (µM) for temperature calculations	0.25
-na	--na_conc	Na+ concentration (mM) for temperature calculations	50.00
-mg	--mg_conc	Mg++ concentration (mM) for temperature calculations	0.00
-m	--multiplex	Path to multiplex ordering sheet (.csv file)	None
-k	--keep	Keep all temporary files for debugging purposes	False

Setup

1. Load dependencies:

```
module load anaconda/3.latest
module load blast+/2.2.29
module load muscle/3.8.31
```

2. Create Reference Directory
   a. Make a new directory:
      mkdir myproject

   b. Copy or link all desired .fasta files into the reference directory (do not include outgroups in this step)

   c. If you do not have a target list, run the generate_target_list.py tool in Primer_Design_Pipeline/tools/generate_target_list.py:
      python generate_target_list.py -d /path/to/reference/directory

   d. Copy or link all outgroups into the reference directory

3. Create a new directory for that will contain the target multifasta, primer3 config file, and the output of Primer Design Pipeline. It is recommended, although not necessary to create this new directory inside the reference directory.

4. Create/move the target multifasta , primer3 config file, **and the target list** to the new directory.

5. Make sure that the only thing in the reference directory is .fasta files.

Running

Run Primer Design Pipeline by using Python to call `primer_design_pipeline.py` in the directory `Primer_Design_Pipeline/`

Here is an example run:

```
python primer_design_pipeline.py -t /path/to/target_multifasta.fasta -d
/path/to/reference_directory -c /path/to/primer3_config.txt -g
/path/to/target_list.txt
```

Using the test data in `Primer_Design_Pipeline/test_data`:

```
python primer_design_pipeline.py -t test_data/design/multifasta.fasta -d
test_data/ -c test_data/design/primer3_config.txt -g
/test_data/design/target_list.txt
```

Running Primer Design Pipeline as a SLURM job:

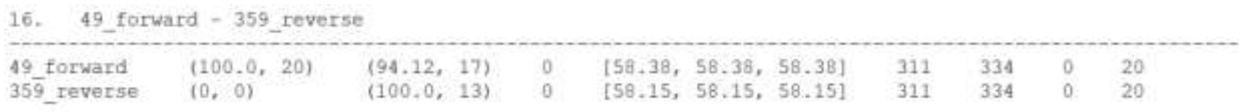
```
#!/bin/bash
#SBATCH --job-name=primer_design_pipeline
#SBATCH -c 4
#SBATCH --time=3:00:00
#SBATCH --mem=50000

python primer_design_pipeline.py -t test_data/design/multifasta.fasta -d
test_data/ -c test_data/design/primer3_config.txt -g
/test_data/design/target_list.txt
```

Output

Primer Design Pipeline will output several files, in addition to creating a few directories.

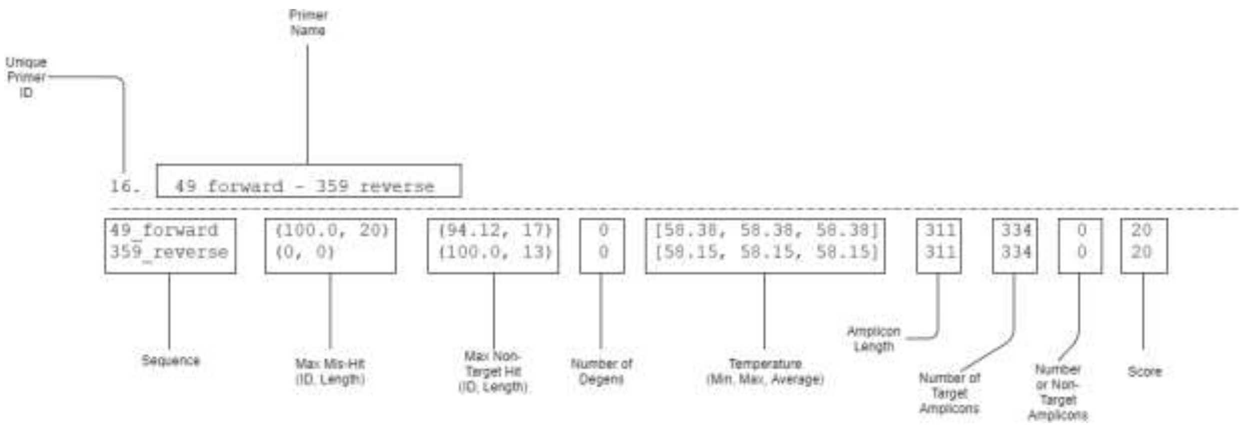
best_primers.txt

The most important output file is best_primers.txt, which will be located in the directory you ran primer_design_pipeline.py from. It will contain the three best primers for each genome in the target multifasta. Each primer contains metadata and information regarding its quality:



```
16.   49_forward - 359_reverse
----------------------------------------------------------------------------------------------------
49_forward      (100.0, 20)     (94.12, 17)     0     [58.38, 58.38, 58.38]     311     334     0     20
359_reverse     (0, 0)          (100.0, 13)     0     [58.15, 58.15, 58.15]     311     334     0     20
```

Here is the same output labeled:



```
16.   49 forward - 359 reverse
----------------------------------------------------------------------------------------------------
49_forward     (100.0, 20)   (94.12, 17)    0   [58.38, 58.38, 58.38]   311   334   0   20
359_reverse    (0, 0)        (100.0, 13)    0   [58.15, 58.15, 58.15]   311   334   0   20
```

Primer Name
Unique Primer ID
Sequence
Max Mis-Hit (ID, Length)
Max Non-Target Hit (ID, Length)
Number of Degens
Temperature (Min, Max, Average)
Amplicon Length
Number of Target Amplicons
Number or Non-Target Amplicons
Score

A mis-hit is when a primer has hit twice in the same genome. A non-target hit is when a primer hits one of the outgroups. They both have two fields: ID and length. The length is the number of bases the mis-hit is. The ID is the percentage accuracy of the hit. I.e. a mis-hit with an ID of 100.0 and a length of 20 has all 20 bases hit. A mis-hit with an ID of 94.12 and a length of 17 has 16 out of 17 bases hit.

Subdirectories

For each genome in the target multifasta, a new subdirectory will be created in the directory you ran primer_design_pipeline.py from. These subdirectories will include three important pieces of information: candidate_primers.txt, primer_conflicts.txt, and primers.blast.out.

1. candidate_primers.txt has all the primers found for that genome, rather than just the best three in best_primers.txt.
2. primer_conflicts.txt has a list of the primers of that genome which may conflict with each other at increasing lengths.
3. primers.blast.out is the output of BLASTing all of the primers against themselves.

The primer_conflicts.txt and primers.blast.out files will help determine if a certain primer will work well in PCR.

combos.pickle

combos.pickle is a binary file that contains all information related to all primers. It is used to automatically generate order sheets using Primer_Design_Pipeline/tools/generate_order_sheet.py. This tool is discussed in the Tools section.

Tools

generate_order_sheet.py

Ordering sheets can automatically be generated using generate_order_sheet.py. The order sheet will contain most of the information needed to order each specified primer. However, there is some information that requires manual filling out.

It can be run using the following:

```
python generate_order_sheet.py -o [output .csv file name] -p
[comma-delimited primer IDs]
```

Here is a more concrete example:

```
python generate_order_sheet.py -o myOrderSheet.csv -p
1,4,6,10,15
```

If you aren't running generate_order_sheet.py from the same directory that your combos.pickle is located, you have to manually specify the location of combos.pickle using the -i flag:

```
python generate_order_sheet.py -o myOrderSheet.csv -p
1,4,6,10,15 -i /path/to/combos.pickle
```

If you want to add more primers to an order sheet you previously generated, you can simply specify the path of your pre-existing order sheet with the -o flag, **as long as you didn't change the format of the pre-existing order sheet in any way.**

make_target_list.py

A target list can automatically be generated with make_target_list.py. It is a fairly simple script that just looks in the directory that you give it, finds all .fasta files, strips the .fasta extension, and writes the genome name to a text file. As such, you need to make sure that the targets are the only thing in the directory when you run it (meaning, make sure you run it before you put the outgroups into the directory).

It can be run with the following:

```
python make_target_list.py -d /path/to/reference_directory
```

database_amplicons.py

Primer Design Pipeline can look for amplicons in every gene/genome in an entire database relatively quickly. database_amplicons.py uses SLURM job arrays to submit several SLURM jobs in parallel.

Flag	**Type**	**Description**
-f	string	Forward primer sequence
-r	string	Reverse primer sequence
-d	string	Path to outermost database directory
-a	int	Desired maximum amplicon size (Default=no limit)
-t	bool (presence)	Output information about which specific genes had 0 amplicons, 1 amplicon, or >1 amplicon. Set this flag to True only if the -d flag is pointing to a specific species within a database instead of the entire database. Default=False
-e	bool (presence)	Execute the job. Highly recommended to set this to True.
-p	int	If you have a combos.pickle file in the directory you are running this script from, you can pass -p [unique primer ID] instead of passing -f and -r. The program will automatically fetch the forward and reverse sequences from combos.pickle. The unique primer IDs can be found in best_primers.txt or candidate_primers.txt. They are the numbers preceding each primer name.

Example runs:

```
python database_amplicons.py -f ACGTACGT -r GCTAGCTA -d
/common/contrib/databases/genbank_bacteria/ -a 500 -e

python database_amplicons.py -f ACGTACGT -r GCTAGCTA -d
/common/contrib/databases/genbank_bacteria/Escherichia_coli/ -a 500 -e
-t

python database_amplicons.py -p 12 -d
/common/contrib/databases/genbank_bacteria/Escherichia_coli/ -a 500 -e
-t
```

Notes:

- Unfortunately, you can only run it one primer at a time.
- If you run it with -e (which you should), it will submit a SLURM job array under your account -- probably 10 to 100 jobs depending on how big the database is, and will take around 30 minutes to an hour.
- In the directory you run it from, it will create a few files and directories: database_amplicon_job.sh, database_amplicon_results_job.sh, and tmp/. Feel free to delete these after the jobs have finished.
- **The results will be in a file called database_amplicon_results.txt where ever you ran the script from.**
- It will create a bunch of separate SLURM out files.

primer3_template.txt

primer3_template.txt is a template for the primer3 config file (which is required for the -c flag). It is ready to use, meaning that the user can use it without changing anything. However, it may be important to look at some of the attributes, like the desired amplicon size and temperature.