Bistro-Primer - Tool to design and validate specific PCR primer pairs for phylogenetic analysis

Praful Aggarwal
Marquette University

BISTRO-PRIMER - TOOL TO DESIGN AND VALIDATE SPECIFIC PCR PRIMER PAIRS FOR PHYLOGENETIC ANALYSIS

by

PRAFUL AGGARWAL

A Thesis submitted to the Faculty of the Graduate School,
Marquette University,
in Partial Fulfillment of the Requirements for
the Degree of Master of Science

Milwaukee, Wisconsin

May 2011

ABSTRACT

BISTRO-PRIMER - TOOL TO DESIGN AND VALIDATE

SPECIFIC PCR PRIMER PAIRS

FOR PHYLOGENETIC ANALYSIS

Praful Aggarwal

Marquette University, 2011

Polymerase Chain Reaction is a widely used biological technique which helps in amplifying small quantities of DNA. These amplified DNA copies are then used in several other experiments like DNA sequencing, phylogenetic analysis, etc. PCR primers are short subsequences of nucleotides (basic unit of DNA) that help identify larger regions of the DNA sequence. They help in successfully amplifying the target DNA sequence by identifying complementary regions on the DNA template. Therefore, to successfully perform PCR it is imperative to design good quality primers.

PCR can be used for identifying the phylogenetic classification of an organism. For example, in an anaerobic digester, there is a diverse microbial community involved that works to digest the waste material into carbon dioxide and methane. The methane produced can be used in the future as a renewable fuel. To identify the microbes involved, researchers use PCR that uses primer pair(s) targeting some specific group of microbes, on the 16S rRNA region of their sequences. This way they are more likely to amplify DNA from specific microbes only which are present in the target group. This could help in the phylogenetic classification of unknown microbes.

In this thesis work a PCR primer pair design and validation software tool has been developed. This tool helps in designing primer pairs that amplify a target region in a specific taxonomic rank (e.g. Genus). It uses a novel scoring function to differentiate between the specific and the not-so specific primer pairs. 16S rRNA sequences for four different genera *(Syntrophobacter, Syntrophomonas, Methanosarcina and Streptococcus)* were used to develop and test the tool. To the best of my knowledge, primer pair(s) specific for amplifying *Syntrophobacter or Syntrophomonas* have not yet been published and the results from Bistro-Primer after further validation would be the first specific primer pairs for target amplification of these genera.

ACKNOWLEDGEMENTS

Praful Aggarwal

I am thankful to my committee members for guiding me throughout my thesis. I also want to thank the members of Dr. Maki's lab and Dr. Zitomer's lab for helping me in understanding some of the aspects used in this work. I would also like to thank Prince Peter Mathai for coming up with the requirement of this software. Finally I want to thank my family for standing by me and being a constant source of motivation.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS i

LIST OF TABLES v

LIST OF FIGURES vi

CHAPTER

1 INTRODUCTION 1

1.1 Motivation 1

1.2 Statement of Problem 2

1.3 Summary of Results 2

1.4 Structure of the Thesis 3

2 BACKGROUND 4

2.1 PCR amplification 4

2.1.1 PCR requirements 4

2.1.2 PCR amplification procedure 5

2.2 Primer Design 7

2.2.1 What is a Primer? 8

2.2.2 Primer design problem 8

2.3 Algorithm Used 10

2.3.1 Primer3 11

2.3.2 Primer-BLAST 12

2.3.3 PRIMROSE 13

3 APPROACH 14
3.1 Design Module 14
3.2 Validation Module 17

4 EVALUATION AND RESULTS 20
4.1 Overview of Evaluation Techniques and Datasets 20
4.2 Design Results 21
4.3 Validation of the Design Results 23

5 CONCLUSIONS AND FUTURE WORK 28
5.1 Summary 28
5.2 Conclusion 28
5.3 Future Work 29

BIBLIOGRAPHY 31

APPENDIX

A HOW TO USE BISTRO-PRIMER 35

B BISTRO-PRIMER DATASETS AND RESULTS 37
B.1 TARGETS 37
B.1.1 Syntrophobacter 37
B.1.2 Syntrophomonas 37
B.1.3 Methanosarcina 37
B.1.4 Streptococcus 38
B.2 NON-TARGETS 39
B.2.1 Syntrophobacter 39
B.2.2 Syntrophomonas 40
B.2.3 Methanosarcina 41

B.2.4 Streptococcus 44

B.3 BISTRO-PRIMER RESULTS 45

C BISTRO-PRIMER DESIGN MODULE 58

D BISTRO-PRIMER VALIDATION MODULE 64

LIST OF TABLES

4.1 *Streptococcus* and 16S rRNA specific primer pairs using Bistro-Primer. In this table a subset of the primer pairs designed have been reported with the corresponding target and non-target hits and the score value. 22

4.2 Primer pair evaluation results using emphin silico PCR amlification for *Streptococcus*. This table shows the *in silico* amplification results on the targets and non-targets alongwith the corresponding Bistro-Primer score. 23

4.3 *Syntrophobacter* and 16S rRNA specific primer pairs using Bistro-Primer. This table consists a subset of the final output generated for *Syntrophobacter* specific PCR primer pairs. It depicts high scoring, medium scoring and the low scoring primer pairs. 26

4.4 *Syntrophomonas* and 16S rRNA specific primer pairs using Bistro-Primer. In this table a subset of the designed primer pairs has been shown. These depict the different cases that can be observed i.e. high, intermediate and low scoring primer pairs 26

4.5 *Methanosarcina* and 16S rRNA specific primer pairs using Bistro-Primer. In this table a subset of the primer pairs designed have been reported with the corresponding target and non-target hits and the score value. The non-targets consist of archae and bacterial sequences. 27

B.1 Bistro-Primer results for the genus *Methanosarcina* with 16S rRNA target region. The forward and reverse primers are in 5'-3' direction. The maximum number of targets is 324 and the maximum number of non-targets is 1167. 47

B.2 Bistro-Primer results for *Syntrophobacter* for 16S rRNA target region. The forward and reverse primers are in 5'-3' direction. The maximum number of targets is 87 and the maximum number of non-targets is 198. 51

B.3 Bistro-Primer results for 16S rRNA target in the genus *Syntrophomonas*. The forward and reverse primers are in 5'-3' direction. The maximum number of targets is 93 and the maximum number of non-targets is 198. 54

B.4 Bistro-Primer results for the genus *Streptococcus* with 16S rRNA target region. The forward and reverse primers are in 5'-3' direction. The maximum number of targets is 400 and the maximum number of non-targets is 602. 57

LIST OF FIGURES

2.1 An illustration of the polymerase chain reaction process. It depicts how a single copy of the target region in a DNA molecule is amplified into millions of copies by using the PCR technique. Obtained from National Institutes of Health. National Human Genome Research Institute."Poylmerase Chain reaction - PCR." Retrieved April 12, 2011 from http://www.genome.gov/Glossary/resources/polymerase chain reaction.pdf 5

2.2 The generic primer design problem. The user provides and input template file and the expected output is a single or a set of primer pairs that satisfy the certain parametric conditions. 9

2.3 Example of a potential primer. The red colored sequence does not satisfy certain parameters for a primer and therefore it is ignored. The green colored sequence satisfies the initial conditions and will be carried on for further checking for being used as a primer. 9

2.4 Primer3 workflow. This figure shows a general workflow for Primer3. The user inputs a template sequence and the other parametric values. Primer3 then checks the oligos for all the parameters. The oligos that satisfy all these conditions are listed as potential primers. 12

3.1 Bistro-Primer workflow. The MSA file represents the target multiple sequence alignment file that the user provides to the design module. The user files contain the target and the non-target sequence files that will be used for the validation of the primer pairs. 15

4.1 Bistro-Primer score vs *in silico* PCR targets for genus:*Streptococcus*. In general, it was observed that the higher the score is the higher the number of amplified targets. 24

4.2 Bistro-Primer score vs *in silico* PCR non-targets for genus:*Streptococcus*. In general, it was observed that the higher the score, the lower the number of amplified non-targets. 25

A.1 Snapshot of the Bistro-Primer Design Module 35

A.2 Snapshot of the Bistro-Primer Validation Module 36

CHAPTER 1
INTRODUCTION

The polymerase chain reaction (PCR) is a widely used biological technique to amplify quantities of specific genes from DNA samples. These amplified genes can be then used in a variety of analyses such as identifying new DNA sequences and placing them into an already existing classification system. PCR primers are short sequences of nucleotides (basic unit of DNA) that bind to larger regions of the DNA sequence and aid in the amplification of a gene fragment. The main goal of this thesis work is to develop a software tool that helps researchers design PCR primer pair(s) for classifying the target organisms into phylogenetic taxonomies.

1.1 Motivation

The Water Quality Center at Marquette University is a collaboration between groups from Civil and Environmental Engineering, Biological Sciences and Bioinformatics [1]. One of the major fields of work done by this team involves the study of the anaerobic digestion process as a source of renewable energy. In order to understand this process it is important to understand the microbial communities involved [1, 2]. In certain cases, several unknown microbes are observed alongwith the known microbes. Hence, to be able to identify these microbes inside a digester, PCR has been employed to amplify DNA obtained from these sludge samples.

In certain cases to study the several pathways involved in anaerobic digestion, the research team comes across certain unknown microbial DNA sample. Therefore, for further study, it is important to identify these unknowns. PCR can play a major role in accomplishing this task. Therefore, to run a successful PCR on unknown DNA samples, it is necessary to design certain specific PCR primer pair(s) that could help identify these unknowns. Designing these primer pair(s) in the lab is not the most efficient way. Thus, to have software that could accomplish the task of designing such specific primer pair(s) can be very useful to the biological community.

Over the past two decades a good number of PCR primer design software have been developed [35]. However, as per knowledge none of these does a good job in designing a taxonomic group specific primer pairs. Some of the software try to accomplish this task, but they

fail to validate their results [3, 17]. So, if a software tool that designs PCR primer pairs specific to a gene and a taxonomic rank, then it could help a research team in identifying and classifying unknown DNA samples. Since these primers would be specific to a target taxon, therefore they would amplify only sequences related to the target. As a result, if an unknown sample is amplified using the specific primers, then, that sample can be identified as related to the target group. The need for such a tool and the impact it can have on several research areas motivated me to pursue this problem.

1.2 Statement of Problem

The problem pursued in this work is to develop a software tool that will help research teams to design and validate PCR primer pairs for taxon specific (e.g. genus) target regions . These primer pairs can then be used for classifying the unknown DNA samples that will be amplified alongwith the known targets.

1.3 Summary of Results

- As part of this thesis work, a PCR primer pair design software tool called Bistro-Primer has been developed. This tool designs primer pair(s) specific to a target region in a specific taxonomic rank. It also validate these primers by checking their specificity to a target taxon.
- It uses a scoring function that indicates the specificity and sensitivity of a primer pair to the target taxon. The higher the score, the more specific the primer pair is. Unlike certain other primer design software, the user can provide a DNA template sequence file in FASTA format. [33]
- Four different datasets were used to evaluate Bistro-Primer. These represent the 16S rRNA gene for the following microbial genera: *Syntrophobacter, Syntrophomonas, Methanosarcina and Streptococcus*. The designed primer pairs were tested for specificity by using an *in silico* PCR amplification tool [4, 18]. This tool simulates the PCR technique and thus provided a suitable option for primer validation. Based on the evaluation results from *in*

silico PCR, it can be shown that the scoring function used for validation performs well *in silico* to predict potential primer pairs (refer to Chapter 4).

- At the time of preparing of this thesis there are no known published primer pairs for *Syntrophobacter and Syntrophomonas*. Therefore, once validated the primer pairs suggested in this work would be the first available primer pairs for these genera. This will be a significant contribution to the anaerobic digestion research.

1.4 Structure of the Thesis

The remainder of this thesis is organized as follows. Chapter 2 provides a background on PCR, the primer design problem and the common design algorithm used. Chapter 3 describes the approach used in developing the Bistro-Primer tool. Results and evaluation of the tool are discussed in Chapter 4. And finally, Chapter 5 summarizes the thesis work alongwith the conclusions and future work.

CHAPTER 2

BACKGROUND

2.1 PCR amplification

The polymerase chain reaction (PCR) is a widely used biological technique that amplifies specific genes that may only be available in small quantities [5]. The advent of PCR has changed the outlook of genetic research in the community. Over the years PCR and its different variants have found an application in several fields. For example, amplification of these small DNA quantities has helped research teams involved with the Human Genome Project [6]. It could also help in experiments like identifying new DNA sequences and placing them into an already existing classification system, or in diagnosing a disease etc.

2.1.1 PCR requirements

PCR is a multi-step process and thus the success of the entire process relies on the successful completion of every step. For the successful execution each step of PCR requires some specific elements. These elements include: a DNA template that contains the region that is to be amplified; and a primer pair of two short oligonucleotide sequences called the forward primer and the reverse primer. A forward primer is an oligonucleotide sequence complementary to the DNA target anti-sense strand. The other primer is called a reverse primer and this is another oligonucleotide complementary to the 3' end of the sense strand of the target DNA.

It also requires a DNA polymerase which is an enzyme that catalyzes the polymerization of nucleotides into a strand complementary to a given template strand. Thus for the synthesis of this new strand, the polymerase requires deoxynucleotide triphosphates (dNTPs). These dNTPs are the nitrogen bases that are the building blocks of DNA and include Adenine, Guanine, Cytosine, Thymine commonly known as A,G,C,T respectively. All these along-with buffer solution, certain ions like Magnesium, Potassium etc. help in amplifying small quantities of DNA into millions of copies [32]. Finally, PCR is performed in small test tubes inside a thermal cycler.

2.1.2 PCR amplification procedure

The PCR process can be sub-divided into two major steps: amplifying the target DNA and checking the amplified region for the desired target. Figure 2.1 [7] gives an outline of a sample DNA amplification into millions of copies using PCR.

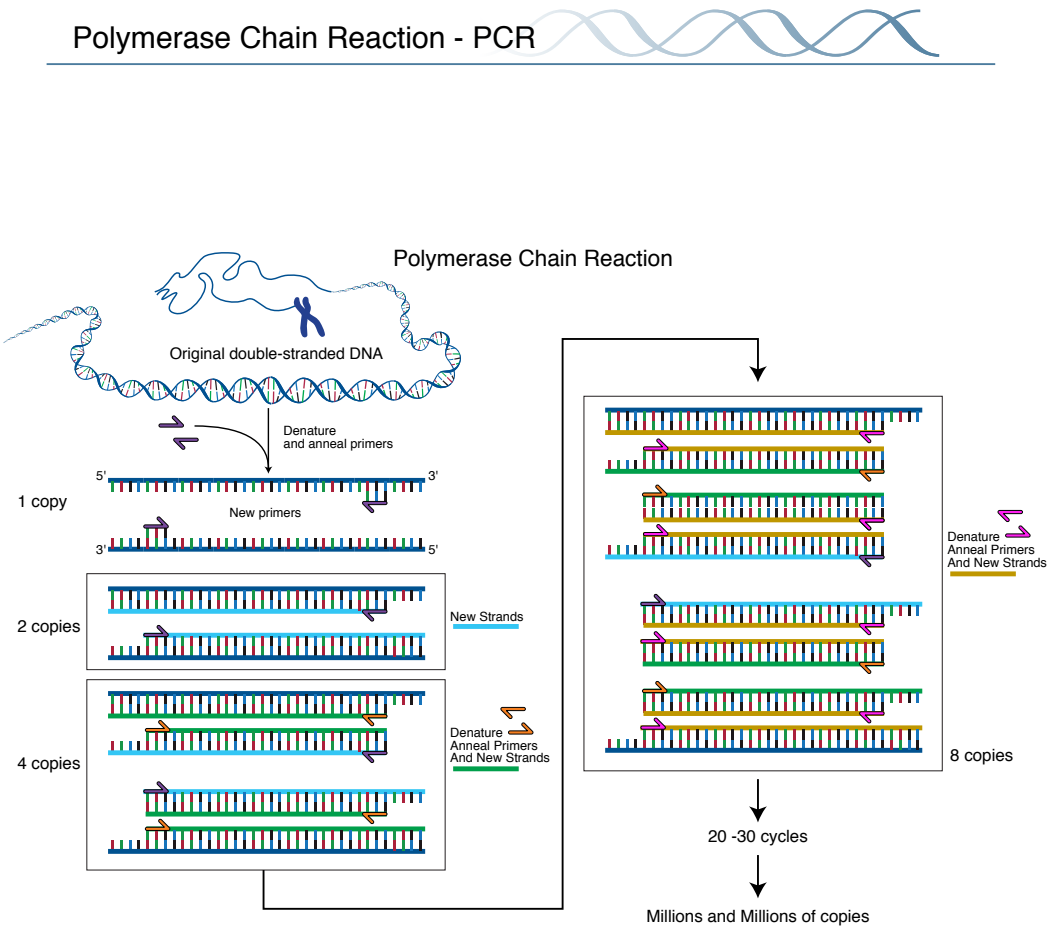

Figure 2.1: An illustration of the polymerase chain reaction process. It depicts how a single copy of the target region in a DNA molecule is amplified into millions of copies by using the PCR technique. Obtained from National Institutes of Health. National Human Genome Research Institute."Poylmerase Chain reaction - PCR." Retrieved April 12, 2011 from http://www.genome.gov/Glossary/resources/polymerase chain reaction.pdf

1. Amplifying the target DNA the process of amplification of DNA involves a cycle of phases that are run several times, thus amplifying the target into thousands and millions of copies. All these phases are heavily dependent on temperature changes and thus need to be carried inside a thermal cycler. This series of internal phases are: denaturation; annealing; and elongation.

Denaturation is the first phase in PCR. In this, the double stranded DNA template is exposed to a temperature of around 94°C. At this temperature, the hydrogen bonds joining the nucleotides on the complementary strands are broken, thus giving single stranded DNA molecule. This breaking up of the double stranded DNA to single stranded is called DNA melting.

Annealing is the next phase in the DNA amplification step. The temperature for this phase depends on the melting temperature of the primers. Primers (oligonucleotides) are present in the tube along-with the single stranded DNA (result of denaturation). These are freely moving and due to their instability, they try to bind (form ionic bonds) to another single stranded sequence, to a form a stable double strand. If a primer is specific to a certain DNA target, it will go and anneal to it making a short but stable connection. DNA polymerase then acts upon this to complete the double strand in the next phase.

Elongation is the phase of the amplification cycle where the DNA polymerase works to form double stranded DNA region. For this, the temperature is changed to 72°C which is appropriate for the working of the polymerase and also helps in breaking any unstable primer-template connections. In this phase, DNA polymerase acts on the product of the annealing phase. It starts elongating the primer bound to the DNA target by adding nucleotides to the 3' end of the primer. These nucleotides are complementary to the corresponding nucleotide on the target strand.

Since there is a forward primer (binding to the anti-sense strand) and a reverse primer (binding to the sense strand) for the same template, in one cycle 2 copies of the target are generated. Thus, with every cycle the number of copies increases exponentially. Generally

30-40 PCR cycles are run for a single target DNA, thus generating millions of copies of the same target.

The time of each phase generally depends on the kind of work being carried out, but even then it remains more or less constant. Also, as these phases are run in cycle, the temperature in the denaturation steps makes sure, that besides DNA melting no other enzymatic process takes place, like extension from the previous cycle (by inactivation of DNA polymerase).

2. Checking the amplified DNA product After the DNA amplification step, it is important to check the amplified region before using it in any other application. Research teams need to make sure that the product represents the expected target. For carrying out this step, they use Gel electrophoresis. They use an agarose-based gel to check for the size of the PCR product. The amplicon is compared with a DNA ladder that consists of DNA molecules of known size. The results of this experiment assist the researchers by providing them with several characteristics of the product, like size, presence of multiple primer binding locations etc., that then help in deciding whether the PCR was successful or not.

Due to its principle, the PCR technique has proved to be a lifeline to the molecular biologists, as it does not restrict their studies due to the availability of small quantities of DNA samples.

2.2 Primer Design

Primer(s) are one of the requirements in PCR. The importance of primer(s) in PCR arises from the way DNA polymerase synthesizes a DNA molecule. It needs some dNTPs to be present on which it can work and synthesize the entire DNA molecule. Thus, a primer provides DNA polymerase with a short sequence of dNTPs to work on. Besides, their role in assisting DNA polymerase in forming the double stranded milecule, primer pairs play a major role in controlling the amplification of the target region. These pairs help in restricting the amplification in the desired target region. Thus it is safe to say that primer(s) play a very important role in PCR based DNA amplification.

2.2.1 What is a Primer?

A primer is a single stranded oligo-nucleotide sequence. Its length varies based on its application, but for regular PCR an 18-24 bp long primer is considered appropriate [26]. PCR uses two primers: forward and reverse. The forward primer is a sequence that is complementary to a region on the anti-sense strand whereas a reverse primer is complementary to some region on the sense strand of a double stranded DNA molecule. Figure 2.1 provides an example of what a primer is in context with a template sequence.

During sequencing of a DNA molecule, sometimes the base composition at a location is not clear. Thus there is a possibility that certain locations may have any of the four nucleotides and their different combinations. Such a condition leads to the presence of a degenerate base at those locations. As a result, besides having the four regular nucleotide bases (A, G, C, T) a primer can also contain degenerate bases [8]. Such primer(s) are called degenerate primer(s). Degenerate primer(s) can be used to target a similar (not identical) gene from different organisms. Also, if a primer is being designed using an amino acid sequence, then also designing degenerate primer(s) is useful as it helps in reducing problems that arise due to degeneracy in the triplet codon(s) where the different codons code for the same amino-acid. An example of a degenerate primer would be `CGCAGGCGGTTWKRTAAGTCTG`, where `W` means that at this position either an `A` or `T` can be observed; `K` represents the presence of either a `G` or `T`; and `R` represents either an `A` or a `G`.

2.2.2 Primer design problem

Due to the role the primer(s) play in carrying out a successful PCR experiment, the designing of primer(s) is of utmost importance and thus needs to be done very carefully. Over the years [9, 25], researchers have laid out certain important parameters that need to be taken into consideration while designing primer(s), a few important ones are: (i) primer melting temperature, (ii) guanine (G) + cytosine (C) content in the primer sequence,(iii) length of the primer, (iv) size of the product that is being amplified, (v) absence of complementarity within a primer (self-complementarity) or complementarity with the other primer (leads to primer-dimers), (vi) temperature at which the primer binds to the given DNA template (annealing temperature), (vii) the nucleotide residue at the 3'end of the primer and also the residues in the 3' clamp (3-4 residues

Problem: Primer Design

Input: DNA or Amino acid sequence

Output: One or more forward and reverse primer pair(s) that satisfy several criteria:

- **Amplifies the target DNA sequence**
- Primer GC content
- Primer melting temperature
- Annealing temperature
- Primer length and product size
- 3'-clamp GC content
- Formation of secondary structure (hairpins)
- Primer-dimer
- Number of occurrences of the primer binding sites

Figure 2.2: The generic primer design problem. The user provides and input template file and the expected output is a single or a set of primer pairs that satisfy the certain parametric conditions.

on the 3' end). There are certain other factors that need to be considered and cannot be left out like, in case of PCR, the difference between the melting temperature of the forward and reverse primers needs to be taken into account, also, the absence of multiple primer binding sites on the same template could also affect the product being amplified. All these parameters help in designing effective primer(s), which in-turn help in successful PCR. In Figure 2.3, the red-colored primer sequences are either short in length or have a low GC content, whereas the green-colored primer has an appropriate GC content and primer length. So, to begin with the red-colored oligo-nucleotides are ignored and the green colored one is further tested as a potential PCR primer.

Figure 2.3: Example of a potential primer. The red colored sequence does not satisfy certain parameters for a primer and therefore it is ignored. The green colored sequence satisfies the initial conditions and will be carried on for further checking for being used as a primer.

2.3 Algorithm Used

Over the past two decades, people have taken up the task of making the PCR primer pair design step more efficient [20]. This has been accomplished by designing primer pairs *in silico* rather than *in vitro*, by developing software that performs this task. As a result, several software implementations are available for designing primer and other oligonucleotides [15]. Software that can design primers for PCR and its several variants like RT-PCR or QPCR or even design primers for a very specific technique like RNAi. Some of the more commonly used primer and oligo design software are: Primer3 [10, 34], Primer- BLAST [3], GeneFisher [11, 28] and PRIMROSE [17].

The several primer design programs follow a similar approach for the most part and differ only slightly in some methods. In general, primer design software requires a DNA (or sometimes protein) sequence from the user. This file generally needs to be in FASTA format. The software might allow the user to input an unaligned or aligned multiple sequence file, but they make use of only one sequence at a time. Then the user sets several parameters like the primer length, the GC content, melting temperature, annealing temperature, product size, melting temperature difference etc. The software finds oligo- nucleotides of the user-defined length. They then, check these subsequences for the several other parameters like the GC content, melting temperature and product size. All the design tools also check for any secondary structures, or complementary within a primer or between a potential primer pair. Another feature taken into account while forming primer pairs, is the melting temperature difference between the forward and reverse primers. The detection of the annealing temperature also is a common feature available in all the design tools. Once all the different parameters have been set, the software provides the user an output with the required number of primer pairs alongwith the values of the several parameters for each of these primers. This helps the user in picking the pairs that suit the more stringent user requirements [20, 26, 29].

Even though all the software almost follow the same approach in primer design and selection, very rarely do two software output the same primer pairs. The main reason behind this is that, different software could use different techniques to detect the different parameter values for the primers. For example, some tools use the following formula for detecting the melting temperature for every primer: Tm = 2°C * (A+T) + 4°C *(G+C) [37] . However, a lot of tools

implement a nearest neighbor based thermodynamic approach to carry out the same task, as this is believed to be a more accurate prediction [19, 36]. The design tools could also differ in the selection of the residues in the 3' clamp and at the 3' end. Primer3, one of the most commonly used primer design tools s discussed in the following paragraphs.

2.3.1 Primer3

Primer3 is a freely available software that helps a user design PCR primer/pair(s) and/or hybridization probes. It can be used either online or locally on a machine. Over the years, Primer3 has become very popular among researchers who need some kind of oligonucleotide sequences (primer/probe) for their work. The ease of use and the vast selection of options is what makes Primer3 special. Primer3 designs primer(s) and/or hybridization probes by identifying shorter subsequences in a single input template sequence, which satisfy the several parameters set by the user. A user can design as many number of primer pair(s) or probe(s) as they want. They can also prioritize the parameters involved in designing the primers, based on their needs. Primer3 accepts a single sequence in FASTA format. It then checks that sequence for the user-defined parameter values. The output is a list of primer pairs with their feature values. Figure 2.4 gives the basic workflow of the Primer3 tool.

A good number of primer design software use Primer3 as the core design software and make certain modifications to the input and output layer and also add some extra features. For example, Primer-BLAST uses Primer3 to design the PCR primer pairs. On top of this, it uses NCBI's BLAST [16] to allow the user to check the resultant primer pairs against Genbank for checking the specificity of the primers to a certain level. Another example of software built on Primer3 is Primaclade [27]. In Primaclade, a user can input a file of sequences in the FASTA format. This tool takes one sequence at a time and supplies it to Primer3 and gets the results back. It does this for every sequence in the file. At the end it removes any duplicates and lists out the good primer pairs. Hence it is safe to say that the use of these design software helps the user in selecting good primer pairs more efficiently and more effectively. But, to make this process successful the users need to use their judgement on selecting the right tool to use and also to provide the right parameters for their specific use.

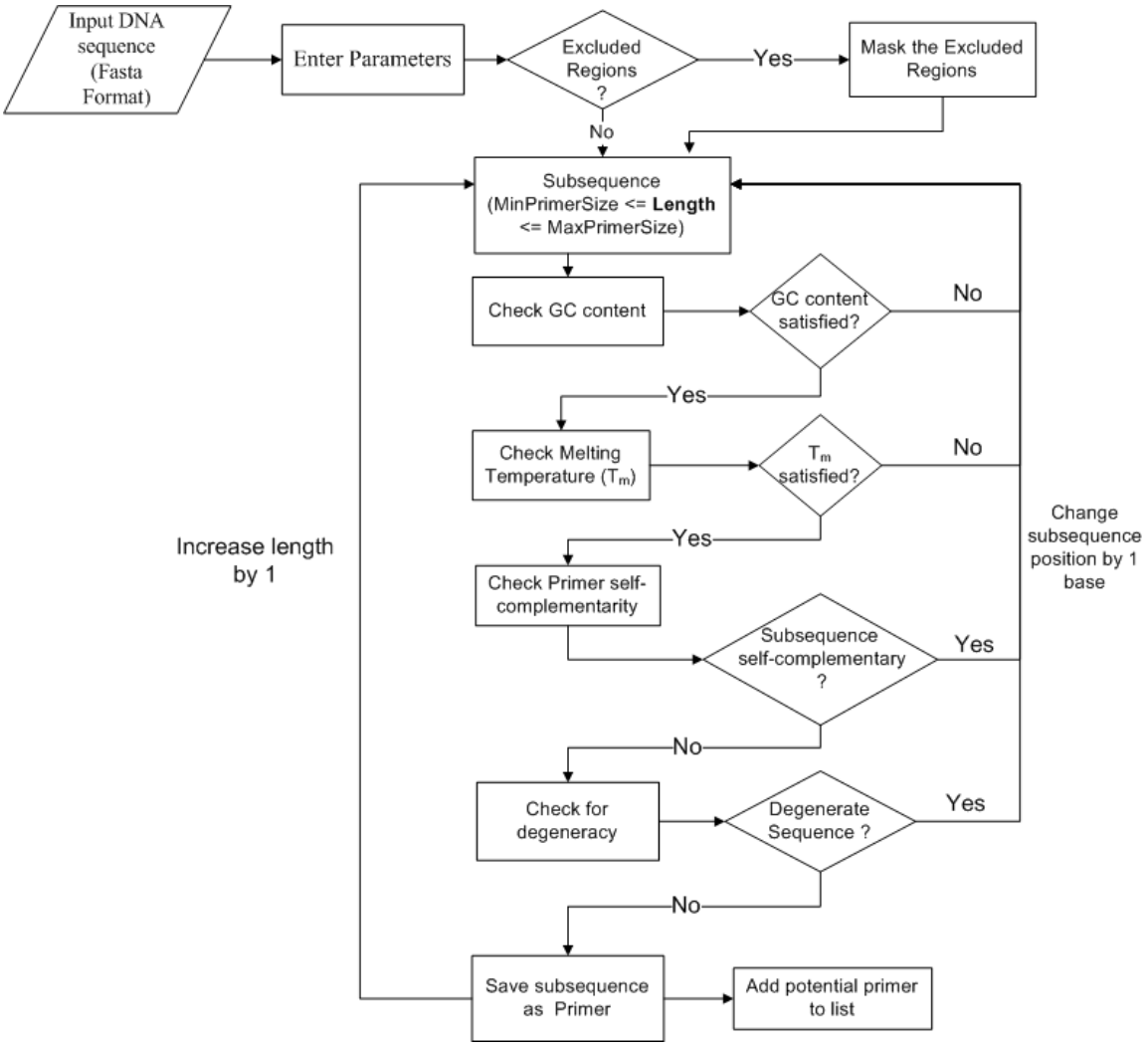

Figure 2.4: Primer3 workflow. This figure shows a general workflow for Primer3. The user inputs a template sequence and the other parametric values. Primer3 then checks the oligos for all the parameters. The oligos that satisfy all these conditions are listed as potential primers.

2.3.2 Primer-BLAST

Primer-BLAST is a primer design software hosted on the National Center for Biotechnology Information (NCBI) server. It generates specific PCR primers for the input template sequence. It uses the Primer3 software to design the primers and then these primers are searched against a user-selected database using NCBI's BLAST search. The blast results avoid primer pairs that may lead to the amplification of any sequence other than the template.

Although Primer-BLAST generates primers that are specific to the template sequence it does not provide the user with any information about the specificity of the primer pairs. The user

therefore has no idea about how specific each primer pair is to the target sequence. Primer-BLAST does not do a good job in defining the non-targets and also in clarifying the process of ignoring the potential non-target amplification by the primer pairs.

2.3.3 PRIMROSE

PRIMROSE is a primer design software that can be used to identify 16S rRNA probes and PCR primers. It is designed to use data from the Ribosomal Database Project (RDP). This tool was developed in 2002 and comes with the RDP release 8.1. The current version of RDP (release 10) is very different from the 8.1 release, therefore the use of this version would not give the user accurate results. Therefore the user will have to provide their own target and non-target database.

Based on the input, the tool then designs oligonucleotides that would be specific to the user defined target database. For designing these oligos the program asks the user for a non-target threshold value. This value represents the maximum number of non-targets to be identified by an oligonucleotide. Any primer pair that exceeds this threshold is ignored by the program. However, for a user to provide a rough non-target threshold is not very flexible as the user would have to keep changing the threshold value for obtaining specific oligos. Also, the output from PRIMROSE is a single 5'-3' (sense) oligonucleotide sequence and it does not provide a primer pair that could be used in PCR. It provides the option of checking the same sense sequence as the anti-sense sequence which is not the same as having a PCR primer pair for amplification.

Both Primer-BLAST and PRIMROSE come close to achieving the goal of designing taxon specific PCR primers. However, they have some major limitations which have been addressed in Bistro-Primer. The next chapter discusses the approach taken by Bistro-Primer for designing taxon specific PCR primer pairs.

CHAPTER 3

APPROACH

The software developed for this thesis work is called Bistro-Primer. It is a PCR primer pair design tool that assists a user in designing primers required for phylogenetic analysis i.e. it designs primer pairs that are specific to a particular taxonomic rank. At a high level, Bistro-Primer consists of two modules: the primer design module; and the primer pair scoring and validation module (see Figure 3.1). The primer design module consists of a standard PCR primer design program like Primer3, GeneFisher etc., that follow the general primer design technique described in Sections 2.2 and 2.3. The validation module is the major contribution of this thesis work. This module uses a scoring function that differentiates between the specific and the generic primer pairs. Both the modules have been discussed in more detail in the remainder of this chapter.

3.1 Design Module

The design module of Bistro-Primer as the name suggests is the component that helps in designing the PCR primer pair(s). Most of the available primer design software design general primer pair(s) for PCR and some of them may be used to design gene specific primer pair(s). Bistro-Primer's design module on the other hand could be used to design either general, gene specific, phylogenetic group or gene and phylogenetic group specific forward and reverse primer pair(s) for carrying out the polymerase chain reaction.

For the development of the design module it was important to understand the underlying principle of PCR and primer design. It was also necessary to understand the basic and advanced requirements for primer design. The primer design problem was discussed earlier in Section 2.2.

The design module is a Python script called `dt_primer_design.py`. The basic outline of the working of this script is as follows. It takes input in the form of a DNA multiple sequence alignment file in the FASTA format. It also asks the user to set several important requirements for PCR primer pair design. From these aligned sequences, the software generates a consensus sequence based on certain parameters. This consensus sequence is then used along-with the several user-defined parameters to design oligo-nucleotides that represent potential forward primers.

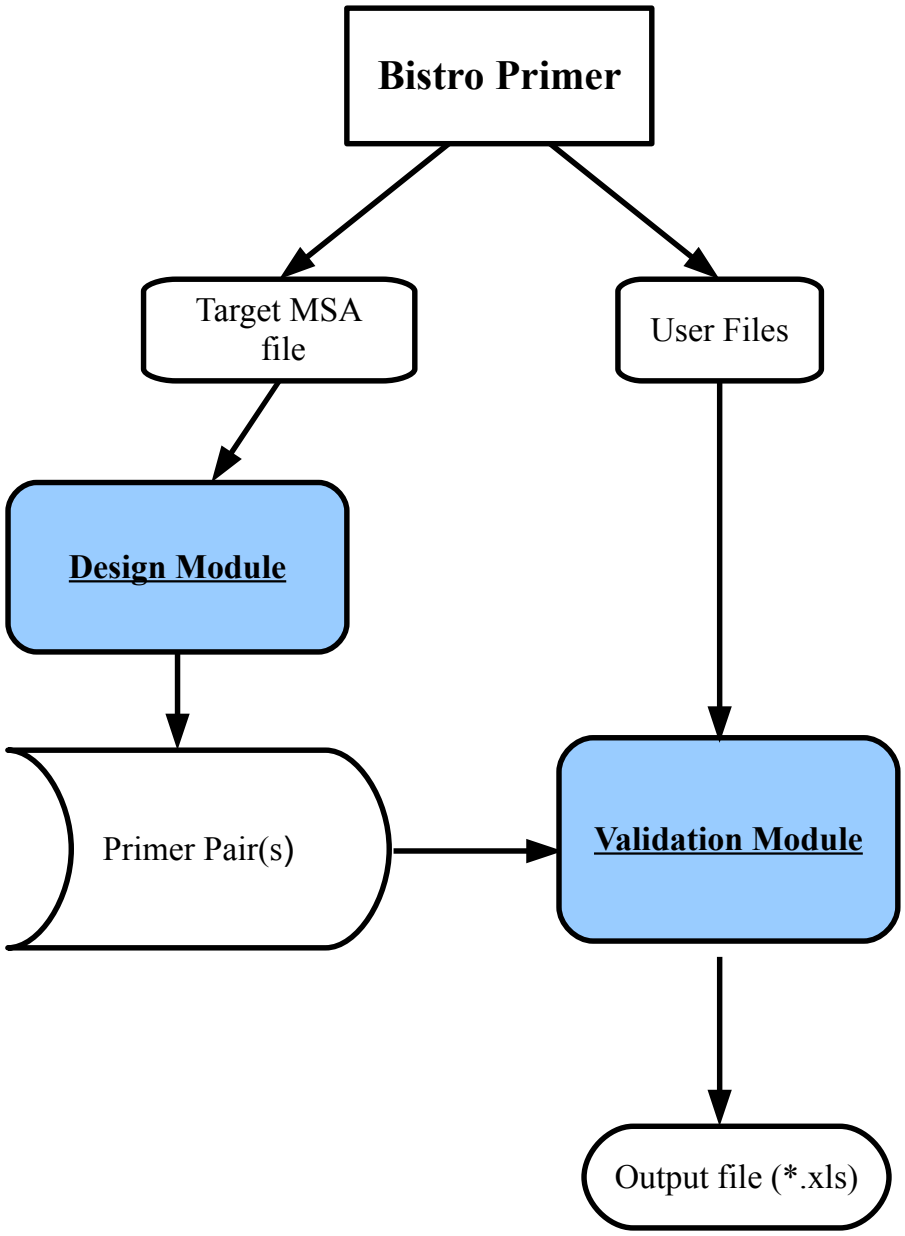

Figure 3.1: Bistro-Primer workflow. The MSA file represents the target multiple sequence alignment file that the user provides to the design module. The user files contain the target and the non-target sequence files that will be used for the validation of the primer pairs.

Similarly, the reverse primers are formed using the reverse complement of the consensus sequence and then the forward and reverse primers are paired based on a couple of parameters. The following paragraphs discuss the designing technique in more detail.

Firstly, the module reads the input multiple sequence alignment file [21, 31]. The sequences in this file are used to generate a consensus sequence. It asks the user to input a maximum threshold value [28]. This value is used in generating the consensus sequence. This value represents the maximum percentage of a particular nucleotide residue at a position. If the percentage of that residue is higher than this value, this residue is assumed to be conserved at this position in the consensus sequence. This value varies between 50–100%. Thus, the software determines the percentage of each residue at every position of the alignment and based on the maximum threshold determines the residue at a position in the consensus sequences. The software also takes into account the possible degeneracy at a position in the consensus sequence [17]. The degeneracy at a position is determined in the same way as the non-degenerate residues. The consensus sequence generated is written to a file and provided as an output to the user.

The consensus sequence generated is now used as the template DNA sequence. So, the next step is the selection of oligonucleotides from this template sequence.This is based on several required parameters set by the user and also certain default checks like secondary structure, primer self complementarity etc. These parameters are necessary for designing good PCR primers. The design module prompts the user to set these parameters one by one. Once set, the module takes into account each of these parameters. The first parameter to be checked is the length of the oligonucleotide. The software breaks the consensus sequence into subsequences of the user-defined length range. These oligos are then checked for the GC content, another user-defined value. If this value is satisfied, the software checks for the melting temperature of the oligo. The melting temperature is checked using `Tm_staluc()` function provided in Biopython. This function uses the nearest neighbor thermodynamic approach to calculate the melting temperature of a DNA oligonucleotide. After satisfying this condition, the software checks for the presence of self-complementarity in a potential primer and also checks for the presence of any secondary structures like hairpin loops [12]. Self-complementarity is checked by identifying substrings in the primer whose reverse complement is also present in the same sequence. This check is followed by a check to identify the presence of a cytosine (C) or guanine (G) at the 3’ end of the primer. Once

all these checks have been conducted on the consensus sequence, they are repeated on the reverse complement of the consensus sequence to identify the potential reverse primers.

The final step in primer pair design is that of pairing the forward and reverse primers into primer pair(s). To accomplish this task, the software takes into account the product size (PCR amplicon size) for a potential primer pair. It makes sure that the reverse primer lies after the forward primer and also that the primers don't overlap. It then checks for the presence or absence of primer dimers by identifying the common substrings which have their reverse complement in the other sequence, i.e. the software checks if a forward primer has a substring whose reverse complement is present in the reverse primer. It then checks the difference in the melting temperatures of the forward and reverse primers. The user can set the value of this difference and thus a potential primer pair's melting temperature difference will lie within the user-defined limit. In the design module, the user can limit the number of primer pair(s) returned by the software by setting a value for the same. After successfully performing all these analysis, the module provides the user with 3 different files: the consensus sequence file, the forward primer and the reverse primer files. Although the forward and reverse primers are in separate files, they are arranged as pairs i.e. the forward primer and the reverse primer of a pair have the same location in their respective files.

3.2 Validation Module

The feature that separates Bistro-Primer from most of the available PCR primer pair design software is the validation of the primer pair(s) for specificity towards a particular gene and a phylogenetic group. To accomplish this task, a validation module was designed and developed. This module has also been written using Python [13, 38] and Biopython [14, 22]. It is called `dt_primer_validation.py`. The way this module works is that, it finds out the number of target (good hits) and the number of non-target (bad hits) for each primer pair and tries to rank the primer pairs based on a score generated by the good and bad hits.

The user needs to provide the target and the non-target sequence file to the validation module. These files must be in FASTA format. The target sequence files contains the sequences that belong to the target taxonomic group. On the other hand, the non-target sequence file can have

any sequences that are either related or unrelated to the target taxon. Alongwith these files, the validation module also requires the output files from the design module which includes, the consensus sequence file, the forward and the reverse primer files. The consensus sequence is used to identify the product size of the amplicon. The forward and reverse primer files consist of the corresponding forward and reverse primers designed.

Firstly, this module accepts all the required files from the user. It then creates a reverse complement of the target and the non-target sequence files. The next step is to search for the presence of a forward primer hit in the target and the non-target sequence files and then to search for the reverse primer hit in the same reverse complement files of the target and the non-target sequences. According to theory, PCR amplification of a double stranded target can only take place if and only if both the forward as well as the reverse primers are present for the product. If only one of the primers is present, then the amplified product would most likely be single stranded and would not show show up as an amplified product on the agarose gel. Therefore, to amplify a target region both the forward and reverse primers for that region must be present. As a result of this observation, it was important to find the target and non-target hits for each primer pair. Thus, the next step is to find the number of common (paired) target and non-target hits for each primer pair.

The final and a major step in this validation module is to compute a score that represents a ranking system to differentiate between the good and the not so good primer pairs for a given gene and phylogenetic group amplification. A scoring function has been designed that uses the paired target and non-target hits. This score helps in contrasting the effect of the targets and non-targets on the primers. The main idea behind this score is to be able to screen out good primer pairs based on their specificity and sensitivity towards a target taxon. According to this score, both the paired target hits and the paired non-target hits have a similar weight attached to them. Thus every non-target takes something away from the target hits. The total number of target hits in the denominator makes sure that having more non-target hits than target hits for a primer pair does not make that primer pair specific to the target phylogenetic group,

$$Score = \frac{\text{Paired Target Hits} - \text{Paired Non-Target Hits}}{\text{Total Target Hits}}.$$

The final output from this module is an MS Excel file. The file contains the forward and reverse primer pair sequences in the 5'–3' direction. This file also contains the product size (PCR amplicon size), the forward and reverse primer target and non-target hits followed by the paired target and paired non-target hits. The final column of the output is the score calculated for each primer pair using the scoring function. The product size is identified using the consensus sequence and therefore in some cases could be off by a few bases. The user can sort this file based on whatever parameter they choose, for example, sorting the file with a decreasing score gives some idea about the better PCR primer pair(s) based on the user's targets and non-targets. The score is to give an indication to the user suggesting the best possible PCR primer pair(s) for a certain experiment, but the final decision of selecting the primer pair lies with the user.

Since there are many PCR primer pair design software available, one of the features of Bistro-Primer was to develop a software that would provide the user with an opportunity to make use of any of the available software and not be restricted to just using Bistro-Primer. This is where the two-module approach comes in handy to the user. The user can use either of these modules according to their needs, for example, they can design and validate the primer pair(s) using Bistro-Primer or they may design the primer(s) using Primer3 and use the validation module of Bistro-Primer to check them against some target and non-target sequences.

However, if using another design software a user needs to keep a few things in mind. To use the validation module, they will still have to provide the software all the files needed by the validation module including the consensus sequence file, the forward and reverse primers and the target and non-target sequence files.

CHAPTER 4
EVALUATION AND RESULTS

In order to successfully develop software, it is important to validate the software. Thus, it was important to evaluate Bistro-Primer to support the results obtained. Hence, in this chapter, the evaluation techniques used to assess Bistro-Primer are discussed. The datasets used in the evaluation process are also discussed in this chapter.

4.1 Overview of Evaluation Techniques and Datasets

Bistro-Primer designs and validates PCR primer pair(s) that can be used in phylogenetic analysis studies using a scoring mechanism. For Bistro-Primer, four prokaryotic datasets were used. The first one consisted of 16S rRNA sequences from the genus Syntrophobacter. The other three datasets were also 16S rRNA sequences for the genera, Syntrophomonas, Methanosarcina and Streptococcus. The Syntrophobacter datset was used during the development of Bistro-Primer. All these datasets were obtained from the Ribosomal Database Project (RDP), Release 10 (latest update) [23, 24, 39]. The GenBank accession IDs for all the target and non-target sequences have been listed in Appendix B.

The Syntrophobacter dataset consists of 87 sequences. The Syntrophomonas dataset contains 93 sequences, Methanosarcina contains 324 sequences and Streptococcus contains 400 sequences. All these sequences are good quality sequences, that represent both type and non-type strains, that are either uncultured or are isolates. These sequences are 1200 or more bases long. The multiple alignment files of these sequences were downloaded from RDP in the FASTA format. The major reason for selecting these genera (except Streptococcus) was the fact that these are few of the major genera worked on in the Anaerobic Digestion team at Marquette University.

These sequence datasets were used to design gene (16S rRNA) and taxonomic rank specific PCR primer pair(s) by Bistro-Primer. As discussed earlier, the major contribution of Bistro-Primer was the validation module that makes use of a scoring function to select these best possible primer pair candidates. Therefore, it was imperative to demonstrate the effect of the scoring function in screening out the best primer pair candidates for a target taxonomic rank.

Testing hundreds of primer pair candidates in wet-lab is both inefficient and costly. Therefore, an *in silco* PCR amplification tool has been used that helps obtain expected theoretical PCR results. This tool consists of around 1300 prokaryotic genomes and requires a primer pair to test for possible amplification in a template genome using the primer pair. There are other *in silico* PCR tools available on the world wide web [30].

However, to further validate the results, in vitro PCR amplification was also performed using the primer pair with the highest score from each of the four datasets. The following Sections discuss the results in more details.

4.2 Design Results

Bistro-Primer designs PCR primer pair(s) based on the several user-defined parameters. The final output is provided to the user in the form of an MS-Excel file. This file contains several columns that represent certain parameters and rows that represent the several different primer pair(s). For their convenience, the user can sort the file based on any of the available parameters. It is advised to sort it on the basis of the Score for each primer pair, as that gives an indication of the better primer pair(s).

The first column in the output file contains the forward primer sequence. The second column has the reverse primer sequence. Both the forward and reverse primers are in 5'-3' direction. The next column has the target amplicon size (from the consensus). Then comes the number of target hits identified by the forward primer and then is the number of targets hit by the reverse primer. The following two columns are the number of forward and reverse non-target hits. The next column contains the number of targets that can be amplified by the corresponding primer pair. The second last column in the file represents the number of non-targets (based on the non-target sequence file) that will be amplified by the primer pair. Finally comes the *Score* of the primer pair.

A good gene and taxon specific primer pair is expected to amplify a large number of target regions from the different organisms in a given taxonomic rank, and zero or a very small number of non-targets. Since, the score calculated by Bistro-Primer makes use of this concept, thus, the best possible primer pair candidates should have the highest score. The Bistro-Primer results support

Forward Primer (5'-3')	Reverse Primer (5'-3')	Paired Target Hits (max 400)	Paired Non-Target Hits (max 602)	Score
AGAAGGTTTTCGGATCGTAAAG	TCCATATATCTACGCATTTCACC	368	0	0.92
GAAACTCAAAGGAATTGACG	CCTTCCTCCGGTTTATTAC	367	3	0.91
GATTAGATACCCTGGTAGTCCACG	CGAGCTGACGACAACCATG	352	65	0.72
TGGTCTGTAACTGACGCTG	CCATTGTAGCACGTGTGTAG	293	18	0.69
CGCAGGCGGTTWKRTAAGTCTG	GTGCTTAATGCGTTAGCTSCG	130	0	0.33
AAACTCAAAGGAATTGACGGG	TGTAGCACGTGTGTAGCC	383	281	0.26

Table 4.1: *Streptococcus* and 16S rRNA specific primer pairs using Bistro-Primer. In this table a subset of the primer pairs designed have been reported with the corresponding target and non-target hits and the score value.

the above statement, as the primer pair(s) with the highest score are the ones with a high number of target amplifications and a low number of non-target amplifications. These results also substantiate the concept that a primer pair that identifies a high number of targets and also a high number of non-targets has a comparatively lower score. Finally, a primer pair with low target hits and low/high non- target hits also has a lower score.

Some of the example results from the Streptococcus dataset are shown in Table 4.1. In the table the first two primer pairs have a high count of target hits and a very low count of non-target hits and thus have a very high score. The next two pairs are moderate scored pairs where they either have a high target and moderate non-target count, or a moderate target and low non-target count. The final two pairs represent primer pairs with a very low score that is attributable to either a low target and non-target count or a high target and non-target count.

To the best of my knowledge no specific PCR primer pairs for *Syntrophobacter and Syntrophomonas* have been reported at the time of this thesis preparation, therefore, after further validation of Bistro-Primer, the best performing primer pairs can be reported as specific PCR primer pairs for these genera.

These results support the scoring function as a good way to select better primer pair(s) that can be used for gene and taxon specific PCR amplification. The next Section further validates the scoring function by checking these primer pair(s) against an *in silico* PCR amplification tool.

4.3 Validation of the Design Results

Although the design results concur with certain expectations from a gene and taxon specific primer pair, yet it is important to check these results for a better corroboration. This task was accomplished by checking the primer pair(s) using the *in silico* amplification tool discussed in Section 4.1. The primer pairs were checked for the amplification of the target gene and also the target taxon. The primer pairs were also checked against the non-target genomes for any amplification. Table 4.2 shows some of the *in silico* PCR results for 16S rRNA gene and the genus Streptococcus.

Forward Primer (5'-3')	**Reverse Primer (5'-3')**	**in silico PCR on target genomes (Max 49)**	**in silico PCR on non-target genomes (Max 1259)**	**Score**
AGAAGGTTTTCGGATCGTAAAG	TCCATATATCTACGCATTTCACC	49	0	0.92
GAAACTCAAAGGAATTGACG	CCTTCCTCCGGTTTATTAC	49	0	0.91
GATTAGATACCCTGGTAGTCCACG	CGAGCTGACGACAACCATG	49	195	0.72
TGGTCTGTAACTGACGCTG	CCATTGTAGCACGTGTGTAG	32	62	0.69
CGCAGGCGGTTWKRTAAGTCTG	GTGCTTAATGCGTTAGCTSCG	13	0	0.33
AAACTCAAAGGAATTGACGGG	TGTAGCACGTGTGTAGCC	49	387	0.26

Table 4.2: Primer pair evaluation results using emphin silico PCR amlification for *Streptococcus*. This table shows the *in silico* amplification results on the targets and non-targets alongwith the corresponding Bistro-Primer score.

The same primer pairs from Table 4.2 have been reported in Table 4.1. Based on these observations, it can be said that there is a consistency in the Bistro-Primer and the *in silico* PCR amplification results. The better performing Bistro-Primer based primer pairs perform similarly in

the *in silico* amplification tool and the not-so good primer pairs do not perform well even in the amplification tool. This consistency supports the scoring function and the validation technique used in Bistro-Primer.

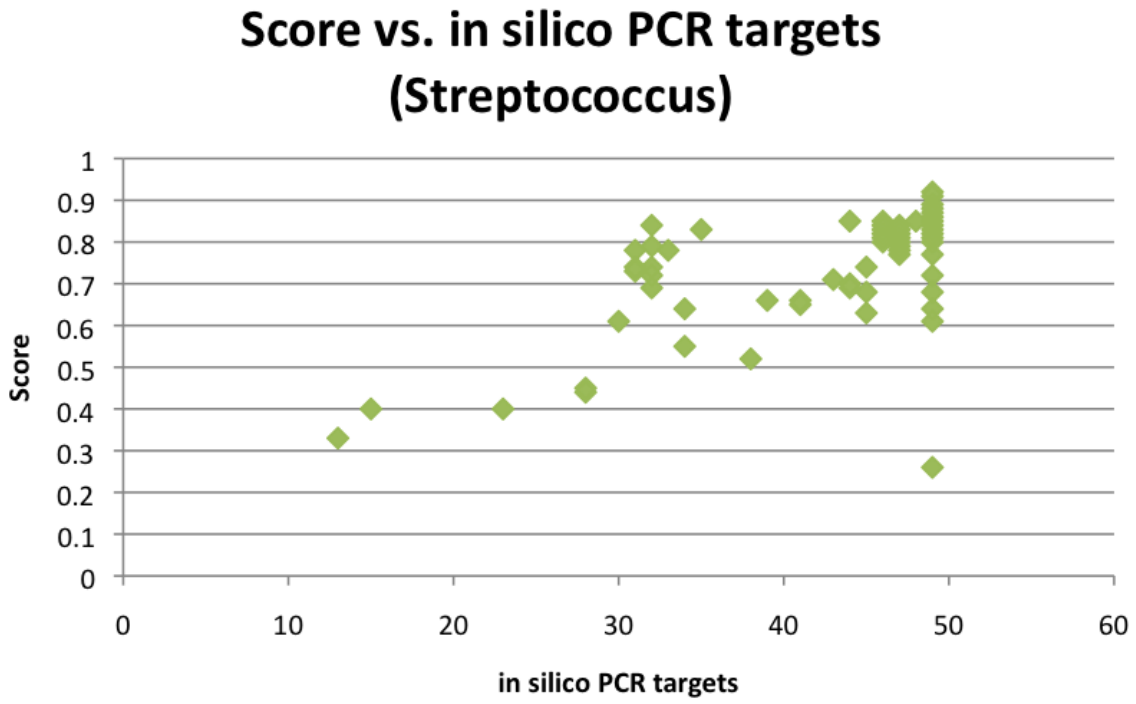

Figure 4.1: Bistro-Primer score vs *in silico* PCR targets for genus:*Streptococcus*. In general, it was observed that the higher the score is the higher the number of amplified targets.

To further study the results from *in silico* PCR, a Score vs *in silico* PCR target hits (Figure 4.1) and Score vs *in silico* PCR non-target hits (Figure 4.2) scatter plots were used. According to Figure 4.1, it can be observed that for a higher score, the number of target genomes amplified is higher as compared to the low scoring primer pairs. This holds up the results obtained from Bistro-Primer. The reason for observing lower target amplifying primer pairs having a higher score is that the number of non-targets being amplified is also much smaller, thus leading to a relatively higher score.

On the other hand, in Figure 4.2 it can be seen that majority of the high scoring primer pairs have a very small to moderate number of non-target amplifications. Whereas, most of the medium to low scoring primer pairs amplify larger number of non-targets. Some primer pairs do not amplify any of the non-targets and still have a lower score, and such a scenario can be due to a small number of target genome amplification.

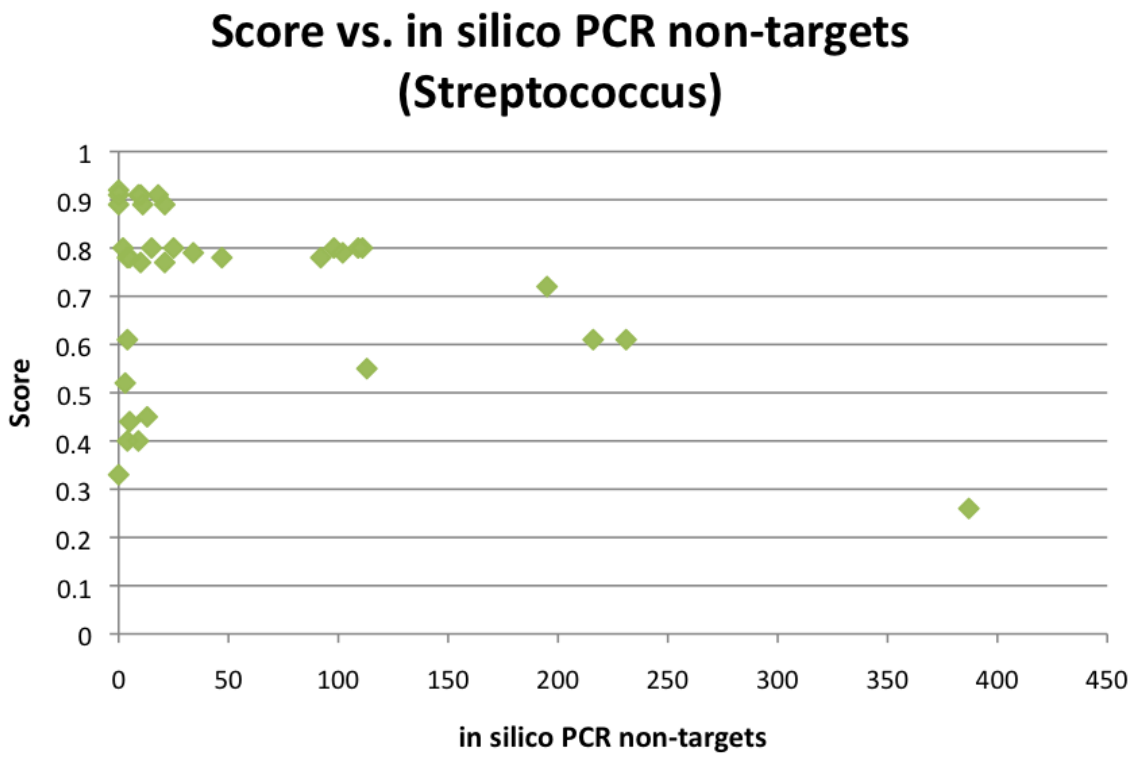

Figure 4.2: Bistro-Primer score vs *in silico* PCR non-targets for genus:*Streptococcus*. In general, it was observed that the higher the score, the lower the number of amplified non-targets.

The following tables include some of the Bistro-Primer results for the other three datasets *(Syntrophobacter, Syntrophomonas and Methanosarcina)*. These tables (Table 4.3, Table 4.4, Table 4.5) depict subsets of results obtained from Bistro-Primer. Certain conditions like a high target and high non-target count lead to a lower specificity score. This is an expected result, as a specific primer pair should not amplify a large number of non-targets and this actually leads to a lower specificity of the primer.

All these primer pairs were tested with the *in silico* PCR amlification tool for target amplification. However, only certain primer pairs from *Syntrophobacter* were tested for non-target amplification as well. The results observed in all the cases followed the same pattern as predicted by Bistro-Primer. The complete set of Bistro-Primer results for all the datasets have been listed in Appendix B.3

Forward Primer (5'-3')	Reverse Primer (5'-3')	Paired Target Hits (max 87)	Paired Non-Target Hits (max 198)	Score
TGAWGAAGGCCTTCGGGTCG	CCCGTCAATTCCTTTGAGTTTTAG	81	5	0.87
AGGTGTAGCGGGTACTCATTC	CGTATTCACCGCGGCATG	72	0	0.83
AAAGCCCTGTCAGGTGGG	ACGTCATCCCCACCTTCC	63	3	0.69
AAGCGTGGGGAGCAAACAGG	RGGCARGGGTTGCGCTCGTTG	73	17	0.64
GAGTAACGCGTAGGYAACCTACCC	ATGAGGACTTGACGTCATCCC	35	0	0.4
CAGCAGCCGCGGTAATAC	ATGAGGACTTGACGTCATCCC	75	45	0.34

Table 4.3: *Syntrophobacter* and 16S rRNA specific primer pairs using Bistro-Primer. This table consists a subset of the final output generated for *Syntrophobacter* specific PCR primer pairs. It depicts high scoring, medium scoring and the low scoring primer pairs.

Forward Primer (5'-3')	Reverse Primer (5'-3')	Paired Target Hits (max 93)	Paired Non-Target Hits (max 198)	Score
GTGTCGTGAGATGTTGGGTTAAG	GCCCAGRTCATAAAGGGCATGATG	85	0	0.91
ACTGGGACTGAGACACGG	CATAAAGGGCATGATGATTTGACG	87	4	0.89
AACTACGTGCCAGCAGCCG	TAGCAAGGGTTGCGCTCGTTG	62	0	0.67
AGGAAYACCAGTGGCGAAGGC	CGAATTAAACCACATGCTCCACC	84	27	0.61
CCCAGACTCCTACGGGAGGCAG	CACGACACGAGCTGACGACAACC	82	55	0.29
GGATAACAGTTGGAAACG	TCTCTGTACCATCCATTG	21	0	0.23

Table 4.4: *Syntrophomonas* and 16S rRNA specific primer pairs using Bistro-Primer. In this table a subset of the designed primer pairs has been shown. These depict the different cases that can be observed i.e. high, intermediate and low scoring primer pairs

Forward Primer (5'-3')	Reverse Primer (5'-3')	Paired Target Hits (max 324)	Paired Non-Target Hits (max 1167)	Score
TACCAGAACGGGTTCGACGGTG	CCACCCGTTGTTGTGCTCCC	302	0	0.93
TACCCGGGTAGTCCCAGC	TTAAGTTTCAGCCTTGCGGC	300	7	0.90
CTCGCCCTCGTGAAGCTGG	GGTGTGTGCAAGGAGCAGGG	276	25	0.77
GGTGGAGCCTGCGGTTTAATTGG	GGGTGGTTTGACGGGCGG	273	73	0.62
AGCCWACGACGGGTACGGG	AATAATCACGATCACCACTCGGG	164	0	0.51
CTGCGGCCTATCAGGTAGTAG	CTCAGAATCCATCTCCGGGC	309	224	0.26

Table 4.5: *Methanosarcina* and 16S rRNA specific primer pairs using Bistro-Primer. In this table a subset of the primer pairs designed have been reported with the corresponding target and non-target hits and the score value. The non-targets consist of archae and bacterial sequences.

CHAPTER 5
CONCLUSIONS AND FUTURE WORK

5.1 Summary

For this thesis, a command line tool named Bistro-Primer, that designs PCR primer pair(s) that can be used for phylogenetic analysis has been developed. The tool consists of a design module and a validation module. The design module was developed using a generic algorithmic approach described in Section 2.3. For designing the primer pairs the user sets the several parameter values to be used. Based on these values the design module provides the user with a list of forward and reverse primers (both are in the 5'-3' direction).

The validation module is the major contribution of this thesis. A scoring function has been introduced as part of this module. This score helps determine the designed primer pairs that are better suited for amplifying regions that belong to the same taxonomic rank (e.g. genus). This can in-turn help research teams to identify and classify unknown samples to a particular taxonomic schema. The validation module works by accepting sequences from the target taxon and also non-target taxa. It then checks for each primer pair in every sequence and provides this information to the scoring function. The output is an excel sheet that contains a list of PCR primer pairs along-with the necessary parametric information of each primer. This file also contains the score for each primer pair. Thus, the user can select primer pairs based on the features they find important.

5.2 Conclusion

The evaluation of the tool was performed using four datasets collected from the Ribosomal Database Project, Release 10. Three *(Syntrophomonas, Methanosarcina and Streptococcus)* out of these four datasets were test sets and one *(Syntrophobacter)* was a development dataset. All these datasets were prokaryotic sequences representing the 16S rRNA gene sequences. The primers pairs generated by Bistro-Primer were checked for their specificity to a certain target gene(16S rRNA) and taxon were evaluated using an *in silico* PCR amplification tool.

Based on the evaluation results, it was observed that the scoring function performs well in determining the specific and non-specific primer pairs. The primer pairs with a higher score are observed to amplify more targets than non-targets. On the other hand, the low scoring primer pairs seem to amplify less target and more/less non-targets or a large number of targets and also non-targets. Thus, based on the evaluation tests performed, it can be concluded that Bistro-Primer does well in designing PCR primer pairs that amplify organisms that belong to the target taxonomic rank. As a result, Bistro-Primer can be used in performing phylogenetic analysis based PCR.

The high scoring primer pairs are proposed for *Syntrophobacter* and *Syntrophomonas* target amplification. These genera do not have any known published primer pairs at the time of preparing this thesis. Thus, if validated, these would be the first primer pairs for these genera. This would be a significant progress in the anaerobic digestion research.

5.3 Future Work

The following suggestions will help improve the tool's specificity and also can help in expanding its scope:

1. The scoring function introduced in Bistro-Primer is based on the number of target and non-target hits for each primer pair. However, one of the observations while evaluating Bistro-Primer was that a large proportion of the non-targets were in some way related to the targets, for example, the non-target hits were classified under the same higher taxonomic rank (e.g. family) as the target hits. Thus, an improvement to the scoring function could be to use certain weighted component that takes the above-mentioned condition into account.

2. Another approach for the scoring function could be to use percentage values instead of actual counts. This could help in further correlating the paired target and paired non-target hits.

3. The current approach, allows a Bistro-Primer user to validate primer pairs designed using any available design software. But an improvement could be to seamlessly integrate an external primer design software (like Primer3) in Bistro- Primer and provide the user with an option to use the external software instead of the design module of Bistro-Primer.

4. Performing more exhaustive laboratory tests will help in further validating this tool and in-turn attract more users.

5. The tool currently designs only PCR primer pairs. A useful addition will be to design specific molecular probes that could be used in the different hybridization studies to identify specific regions amongst a particular taxonomic unit.

BIBLIOGRAPHY

[1] Water Quality Center at Marquette University: http://www.marquette.edu/engineering/civil_environmental/centers_water.shtml.

[2] Zitomer Lab Group: http://www.eng.mu.edu/Zitomer_Lab_Group/index.html.

[3] Primer-BLAST: http://www.ncbi.nlm.nih.gov/tools/primer-blast/.

[4] in silico PCR amplification tool: http://insilico.ehu.es/.

[5] Polymerase Chain Reaction: http://en.wikipedia.org/wiki/Polymerase_chain_reaction.

[6] Human Genome Project Information: http://www.ornl.gov/sci/techresources/Human_Genome/home.shtml.

[7] National Institutes of Health and National Human Genome Research Institute. "Polymerase Chain Reaction - PCR." Retrieved April 12, 2011, from http://www.genome.gov/Glossary/resources/polymerase_chain_reaction.pdf.

[8] IUPAC code: http://www.bioinformatics.org/sms2/iupac.html.

[9] Primer Design: http://bioweb.uwlax.edu/genweb/molecular/seq_anal/primer_design/primer_design.htm.

[10] Primer3 Input: http://frodo.wi.mit.edu/primer3/.

[11] GeneFisher: http://bibiserv.techfak.uni-bielefeld.de/genefisher2/.

[12] Longest Common Substring: http://en.wikibooks.org/wiki/Algorithm_implementation/Strings/Longest_common_substring.

[13] Python: http://www.python.org.

[14] Biopython: http://www.biopython.org.

[15] ABD-ELSALAM, K. A. Bioinformatic tools and guideline for pcr primer design. *African Journal of Biotechnology 2*, 5 (2003), 91–95.

[16] ALTSCHUL, S. F., GISH, W., MILLER, W., MYERS, E. W., AND LIPMAN, D. J. Basic local alignment search tool. *J. Mol. Biol. 215* (1990), 403–410.

[17] ASHELFORD, K. E. E. A. Primrose: a computer program for generating and estimating phylogenetic range of 16s rrna oligonucleotide probes and primers in conjunction with the rdp-ii database. *Nucleic Acids Research 30* (2002), 3481–3489.

[18] BIKANDI, J., SAN MILLÁN, R., REMENTERIA, A., AND GARAIZAR, J. In silico analysis of complete bacterial genomes: Pcr, aflp-pcr, and endonuclease restriction. *Bioinformatics 20*, 5 (March 2004), 798–9.

[19] BRESLAUER, K. J., RONALD, F., BLOCKER, H., AND MARKY, L. A. Predicting dna duplex stability from the base sequence. *Proc. Natl. Acad. Sci. 83* (1986), 3746–3750.

[20] BURPO, F. J. A critical review of pcr primer design algorithms and cross-hybridization case study. *Biochemistry 218* (2001).

[21] CHENNA, R., SUGAWARA, H., KOIKE, T., LOPEZ, R., GIBSON, T. J., HIGGINS, D. G., AND THOMPSON, J. D. Multiple sequence alignment with the clustal series of programs. *Nucleic Acids Res. 31*, 13 (2003), 3497–3500.

[22] COCK, P. J. A., ANTAO, T., CHANG, J. T., CHAPMAN, B. A., COX, C. J., DALKE, A., FRIEDBERG, I., HAMELRYCK, T., KAUFF, F., WILCZYNSKI, B., AND DE HOON, M. J. L. Biopython: freely available python tools for computational molecular biology and bioinformatics. *Bioinformatics 25*, 11 (2009), 1422–1423.

[23] COLE, J. R., CHAI, B., FARRIS, R. J., WANG, Q., KULAM-SYED-MOHIDEEN, A. S., MCGARRELL, D. M., BANDELLA, A. M., CARDENAS, E., GARRITY, G. M., AND TIEDJE, J. M. The ribosomal database project (rdp-ii): introducing myrdp space and quality controlled public data. *Nucleic Acids Res. 35 (Database issue)* (2007), D169–D172.

[24] COLE, J. R., WANG, Q., CARDENAS, E., FISH, J., CHAI, B., FARRIS, R. J., KULAM-SYED-MOHIDEEN, A. S., MCGARRELL, D. M., MARSH, T., GARRITY, G. M., AND TIEDJE, J. M. The ribosomal database project: improved alignments and new tools for rrna analysis. *Nucleic Acids Res. 37 (Database issue)* (2009), D141–D145.

[25] DIEFFENBACH, C. W., LOWE, T. M., AND DVEKSLER, G. S. General concepts for pcr primer design. *Genome Res. 3* (1993), S30–S37.

[26] DIEFFENBACH, C. W., LOWE, T. M. J., AND DVEKSLER, G. S. *PCR PRIMER A LABORATORY MANUAL*. Cold Spring Harbor Laboratory Press, 1995.

[27] GADBERRY, M. D., MALCOMBER, S. T., DOUST, A. N., AND KELLOGG, E. A. Primaclade - a flexible tool to find primers across multiple species. *Bioinformatics 21* (2005), 1263–1264.

[28] GIEGERICH, R., MEYER, F., AND SCHLEIERMACHER, C. Genefisher - software support for detection of postulated genes. *Proc Int Conf IntellSyst Mol Biol 4* (1996), 68–77.

[29] HYNDMAN, D. L., AND MITSUHASHI, M. *PCR PROTOCOLS*, second ed., vol. 226. Humana Press, 2003.

[30] KENT, W. J., SUGNET, C. W., FUREY, T. S., ROSKIN, K. M., PRINGLE, T. H., ZAHLER, A. M., AND HAUSSLER, D. The human genome browser at ucsc. *Genome Res. 12*, 6 (June 2002), 996–1006.

[31] LARKIN, M. A., BLACKSHIELDS, G., BROWN, N. P., CHENNA, R., MCGETTIGAN, P. A., MCWILLIAM, H., VALENTIN, F., WALLACE, I. M., WILM, A., LOPEZ, R., THOMPSON, J. D., GIBSON, T. J., AND HIGGINS, D. G. Clustalw and clustalx version 2. *Bioinformatics 23*, 21 (2007), 2947–2948.

[32] MULLIS, K. B. The unusual origin of the polymerase chain reaction. *Scientific American 262* (1990), 56–63.

[33] PEARSON, W. R., AND LIPMAN, D. J. Improved tools for biological sequence comparison. *Proceedings of the National Academy of Sciences of the United States of America 85*, 8 (1988), 2444–8.

[34] ROZEN, S., AND SKALETSKY, H. J. Primer3 in the www for general users and for biologist programmers. *In: Krawetz S, Misener S (eds) Bioinformatics Methods and Protocols: Methods in Molecular Biology. Humana Press, Totowa, NJ* (2000), 365–386.

[35] RYCHLIK, W., AND RHOADS, R. E. A computer program for choosing optimal oligonucleotides for filter hybridization, sequencing and in vitro amplification of dna. *Nucleic Acids Res. 17*, 21 (1989), 8543–8551.

[36] RYCHLIK, W., SPENCER, W. J., AND RHOADS, R. E. Optimization of annealing temperature for dna amplification in vitro. *Nucleic Acids Res. 18* (1990), 6409–6412.

[37] SUGGS, S. V., HIROSE, T., MYAKE, E. H., KAWASHIMA, M. J., JOHNSON, K. I., AND WALLACE, R. B. Using purified genes. *ICN-UCLA Symp. Mol. Cell. Biol. 23* (1981), 683–693.

[38] VAN ROSSUM, G., AND DE BOER, J. Interactively testing remote servers using the python programming language. *CWI Quaterly 4*, 4 (December 1991), 283–303.

[39] WANG, Q., GARRITY, G. M., TIEDJE, J. M., AND COLE, J. R. Naive bayesian classifier for rapid assignment of rrna sequences into the new bacterial taxonomy. *Appl Environ Microbiol 73*, 16 (2007), 5261–5267.

APPENDIX A
HOW TO USE BISTRO-PRIMER

Bistro-Primer is a command line tool. Once you download the executable files, you can copy the files in a directory. For running Bistro-Primer, the user needs a target Multiple Sequence Alignment file, a target sequence file (ungapped) and a non-target sequence file.

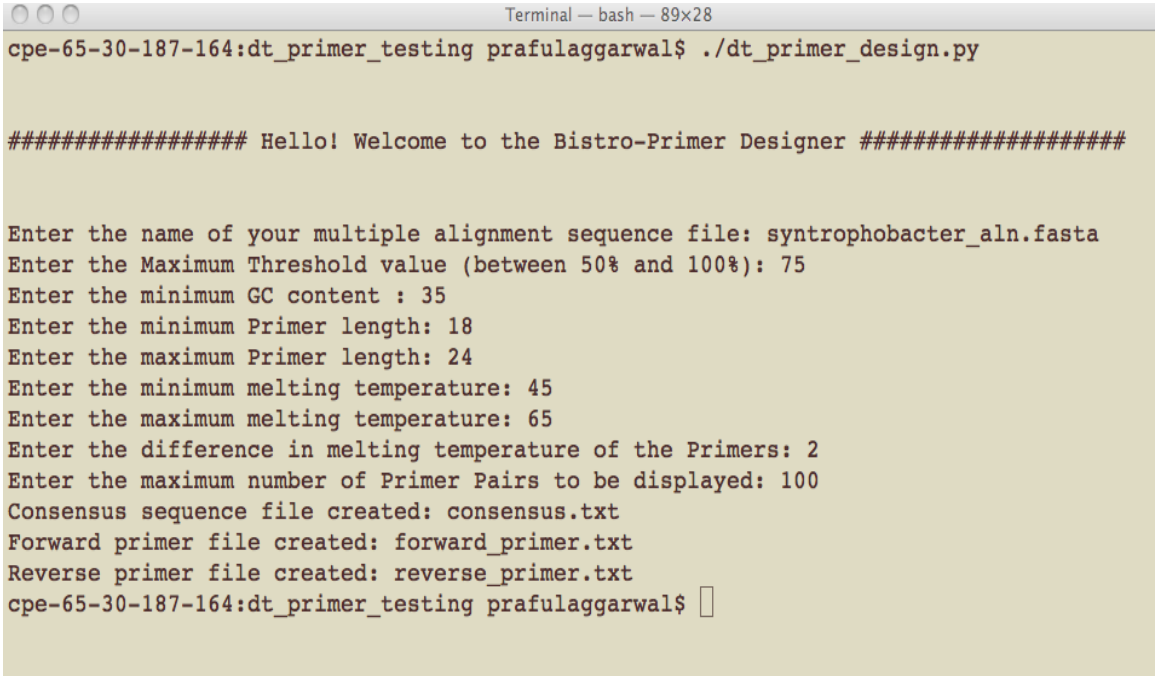

Figure A.1: Snapshot of the Bistro-Primer Design Module

Navigate to the directory where the Bistro-Primer executables are located. First, run the design module: `dt_primer_design.py` file using the following command: `./dt_primer_design.py`. It generates a forward and reverse primer file along with the consensus sequence file. Next run the validation module: `dt_primer_validation.py`, using the following command: `./dt_primer_validation.py`. This module will generate the final output file in `(*.xls)` format. The user can open this file using MS Excel and view the results.

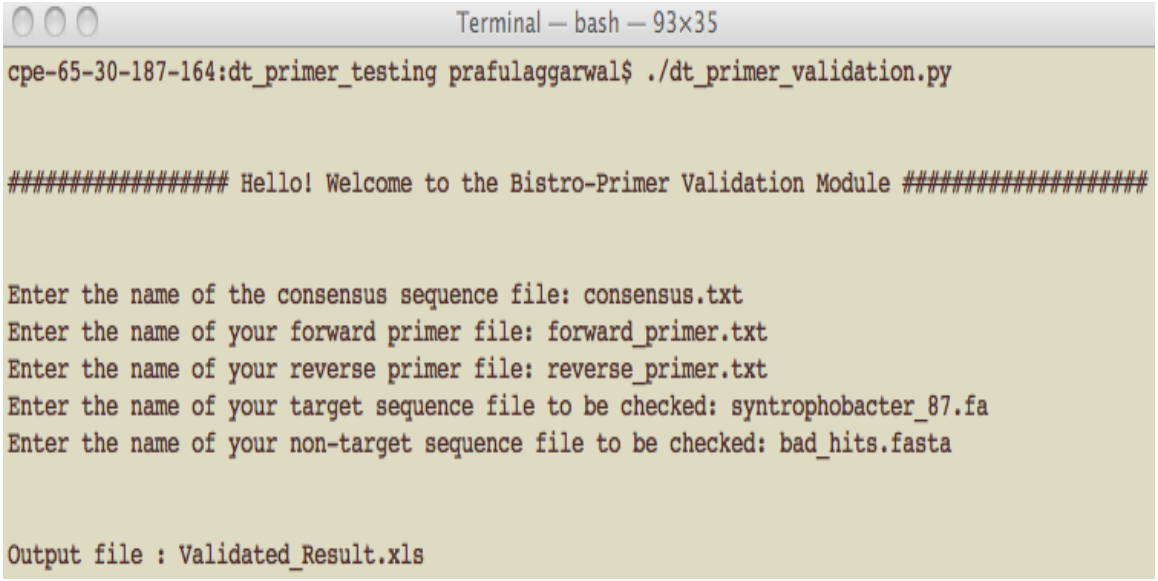

Figure A.2: Snapshot of the Bistro-Primer Validation Module

APPENDIX B
BISTRO-PRIMER DATASETS AND RESULTS

The Genbank accession IDs of the target and non-target datasets used for the development and testing of Bistro-Primer are listed in this appendix.

B.1 TARGETS

B.1.1 Syntrophobacter

X70905, X82875, X82874, AJ009502, X94911, AF395427, AJ306805, AF482435, AF482439, AF524857, AY780561, AJ617831, AY426441, AY426461, AY426465, AY426467, AY651787, AB232801, AB232820, AY775505, AB238765, DQ404834, DQ415751, AM176858, AM176868, DQ133940, AB252680, CP000478, CP000478, EF515491, EF515501, EF515568, EF515593, EF515613, EF515617, EF515665, EF515670, EF515676, AB262718, EF688192, EF688226, EF688265, EU542431, EU542530, EU888828, FJ189533, AB447621, AB447624, AB447625, AB447628, AB447629, AB447637, AB447640, AB447644, AB447647, AB447691, AB447703, AB447723, FJ535519, FJ638562, FJ638582, FJ769512, FJ799157, FJ484661, FJ971723, GQ402673, GQ402689, GQ402720, AM490743, FN646432, FN646434, FN646439, FN646446, FN646448, FN646453, FN646454, FN646460, FN646461, GU180163, GU202953, GQ261306, GU208388, GU208398, AB539932, CP000478, CP000478, GU389854

B.1.2 Syntrophomonas

AB098336, AF050585, AF275925, AF349760, AY290767, AB021306, AF022248, AF022249, AY540494, AF482440, AF482442, AF482445, AF482448, AF529116, M26491, M26492, AY553938, AY426456, AY536889, AY643536, DQ080135, DQ080180, DQ086234, AB186860, AB186861, DQ112186, DQ112187, AB192059, AB244314, DQ288691, AB234271, AB234272, AB248623, AB248632, AB248633, AB248636, DQ449033, DQ449034, AB234008, DQ459209, DQ459211, DQ459212, DQ459214, DQ666175, DQ666176, AB252683, AB274039, AB274040, CP000448, CP000448, CP000448, DQ898277, AB294300, AB294308, DQ339705, DQ339710, DQ339714, EF515497, EF515712, EF559161, DQ984660, DQ984666, EF644505, EF644507, AM947535, AM947547, EU498379, EU887777, EU887781, EU887790, EU887793, EU887794, AB447752, FJ799130, FJ799160, FJ825442, FJ825457, GQ340223, FJ469361, FN563242, GU139310, FN646462, GQ203634, GQ203635, GQ203649, HM308556, AB539934, AB539940, AB539941, HM041938, CP000448, CP000448, CP000448

B.1.3 Methanosarcina

AB065295, AB065296, AY692059, AF020341, AF262036, AF411467, AF411469, AF432127, AY196682, AY196685, AY225109, AY260430, AY260431, AY260432, X69874, AY641448, AF028691, AF028692, AF519802, AJ002476, AJ012094, AJ012095, AJ012742, AJ238648, M59136, M59137, M59138, M59140, M59144, U20150, U20151, U20153, U89773, AY570657, AY570658, AY570674, AY570679, AY570681, AY667271, DQ068083, DQ068084, DQ068085, AY663809, AB084244, AB092915, AB114309, CP000099, CP000099, CP000099, DQ058823, AB237738, AB244741, AE008384, AE008384, AE008384, AB248616,

AB248617, AE010299, AE010299, AE010299, DQ478747, AB266896, AB266912, DQ250385, EF175709, EF175716, AB288240, AB288241, AB288243, AB288244, AB288247, AB288250, AB288255, AB288257, AB288259, AB288261, EF376987, AB288283, AB294254, DQ339720, DQ339721, EF452664, EU155896, EU155897, EU155898, EU155899, EU155943, EU155944, DQ987528, EU369604, EU369605, EU369610, EU369614, EU369617, EU369618, EU369623, EU369625, AB288264, AB300208, EU910623, EU857627, EU857628, U81776, U81777, FJ685724, FJ685733, FJ685738, AB494248, AB494252, FJ982667, FJ982669, FJ982701, FJ982703, FJ982707, FJ982708, FJ982709, FJ982711, FJ982737, FJ982740, FJ982743, AB509221, GQ328818, GQ328819, AB495356, EU420698, EU420708, AB529909, AB529910, AB529911, AB529912, AB529913, AB529914, AB529915, AB529916, AB541582, AB541583, AB541584, AB541585, AB541586, AB541587, AB541588, AB541590, AB541592, AB541593, AB541594, AB541595, AB541596, AB541597, AB541598, AB541599, AB541600, AB541601, AB541602, AB541603, AB541604, AB541605, AB541606, AB541607, AB541608, AB541609, AB541610, AB541611, AB541612, AB541613, AB541614, AB541615, AB541616, AB541620, AB541621, AB541622, AB541623, AB541624, AB541625, AB541626, AB541627, AB541628, AB541629, AB541630, AB541631, AB541632, AB541633, AB541634, AB541635, AB541660, AB541661, AB541662, AB541663, AB541664, AB541665, AB541666, AB541667, AB541668, AB541669, AB541670, AB541671, AB541672, AB541673, AB541705, AB541706, AB541707, AB541708, AB541709, AB541710, AB541711, AB541712, AB541713, AB541714, AB541715, AB541716, AB541717, AB541718, AB541734, AB541735, AB541736, AB541737, AB541738, AB541739, AB541740, AB541741, AB541742, AB541743, AB541744, AB541745, AB541746, AB541747, AB541748, AB541795, AB541796, AB541797, AB541798, AB541799, AB541800, AB541801, AB541802, AB541803, AB541804, AB541805, AB541806, AB541822, AB541823, AB541824, AB541825, AB541826, AB541827, AB541828, AB541829, AB541830, AB541831, AB541832, AB541833, AB541834, AB541835, AB541836, AB541857, AB541858, AB541859, AB541860, AB541861, AB541862, AB541863, AB541864, AB541865, AB541866, AB541867, AB541868, AB541869, AB541871, AB541872, AB541873, AB541874, AB541875, AB541876, AB541877, AB541878, AB541879, AB541880, AB541881, AB541882, AB541883, AB541884, AB541885, AB541886, AB541887, AB541888, AB541889, AB541890, AB541891, AB541892, AB541893, AB541894, AB541927, AB541928, AB541929, AB541930, AB541931, AB541932, AB541933, AB541935, AB541936, AB541937, AB541938, AB541939, AB541940, AB541941, AB541942, AB541943, AB541944, AB541945, AB541946, AB541947, AB541948, AB541949, AB541950, AB541951, AB541952, AB541953, AB288262, AB598272, FR733661, FR733698, HQ591417, HQ678046, HQ678096, HQ678098

B.1.4 Streptococcus

Z94011, AJ243966, AF139599, AJ297216, AB071337, AB071345, AJ413205, AB008926, AF433167, AJ131965, AB071350, AB023576, AF139602, AB071347, Y18026, X68418, AJ295848, AJ295853, AB071340, Y10869, Y09007, AB071343, X89967, AB071338, AB071344, AB071341, AB071348, Y07601, Z94012, AJ297215, AB071349, AB038371, AJ413204, AF139600, AJ301607, Y17358, AB071336, X94337, AJ297218, AJ427479, AB023573, AB023574, AJ319643, AJ297214, AB023575, AJ297217, AB071346, AB071342, AF139601, AJ413203, AJ243965, AJ314609, AF139603, AJ319644, Y10867, AB071339, AJ305257, AJ307888, X78826, X78825, X81023, AJ420200, AJ420196, AJ420197, AJ420199, AJ420198, AJ420201, X58317, X58302, X59032, X53653, X58312, X59028, X58305, X58320, AJ314611, X53652, X58310, X58321, X59030, AJ314610, X58311, X58319, X59061, X59031, X58307, X58314, X58316, X59029, AJ409287,

X58318, X58301, X58303, X58304, X58315, X58308, X58309, AB102730, AY324631, AY324632, AJ583201, AJ583202, AY442818, AY442813, AB096755, AB104845, AB104840, AB104843, AB104844, AB104841, AB104839, AB104842, AB104848, AB104850, AB104846, AB104849, AB104847, AB112407, AE006615, AF202012, AF202013, AY278609, AY278629, AY278630, AY278631, AY278632, AY278633, AY278634, AY278635, AY648569, AB174791, AB174792, AF157108, AF290487, AF349918, AF349920, AF349932, AF371504, AF371505, AF371506, AF371507, AF371508, AF371509, AF371510, AF371511, AF385545, AF385550, AF385574, AF408257, AF408258, AF408260, AF408263, AF432131, AF432132, AF432134, AF432135, AF432136, AF432137, AF432139, AF481230, AY005042, AY005043, AY005044, AY005046, AY167955, AY167959, AY256519, U87828, U87829, U87830, AY584476, AY584477, AY584478, AY584479, AY324610, AY324611, AY324612, AY324613, AY327522, AY327523, AY691525, AY691526, AY691527, AY691528, AY691529, AY691530, AY691531, AY691532, AY691533, AY691534, AY691535, AY691536, AY691537, AY691541, AY691542, AB002479, AB002480, AB002481, AB002482, AB002483, AB002484, AB002485, AB002486, AB002487, AB002488, AB002489, AB002490, AB002491, AB002492, AB002493, AB002494, AB002495, AB002496, AB002497, AB002498, AB002499, AB002500, AB002501, AB002502, AB002503, AB002504, AB002505, AB002506, AB002507, AB002508, AB002509, AB002510, AB002511, AB002512, AB002513, AB002514, AB002515, AB002516, AB002517, AB002518, AB002519, AB002520, AB002521, AB002522, AB002523, AB002524, AB002525, AB002526, AB002527, AB006119, AB006120, AB008314, AB008315, AE008537, AE009954, AE009954, AE010074, AF003928, AF003929, AF003930, AF003931, AF003932, AF003933, AF009475, AF009476, AF009477, AF009478, AF009479, AF009480, AF009481, AF009482, AF009483, AF009484, AF009485, AF009486, AF009487, AF009488, AF009489, AF009490, AF009491, AF009492, AF009493, AF009494, AF009495, AF009496, AF009497, AF009498, AF009499, AF009500, AF009501, AF009502, AF009503, AF009504, AF009505, AF009506, AF009507, AF009508, AF009509, AF014814, AF014815, AF014816, AF014817, AF014818, AF014819, AF014820, AF015927, AF015928, AF124350, AF135453, AF145239, AF145240, AF145243, AF145244, AF145245, AF145246, AF176100, AF176101, AF176102, AF176103, AF176104, AF176105, AF176106, AF176107, AF176108, AF177729, AF184974, AF201898, AF201899, AF202263, AF221604, AF227836, AF235052, AF284578, AF284579, AF298197, AF298198, AF298199, AF316591, AF316592, AF316593, AF316594, AF316595, AF316596, AF323911, AF335572, AF335573, AF336367, AF357559, AF357560, AF396920, AF396921, AF396922, AF429762, AF429763, AF429764, AF429765, AF429766, AF432856, AF439398, AF459431, AF459432, AF459433, AF479579, AF479580, AY098489, AY099095, AY121359, AY121360, AY121361, AY121362, AY138231, AY138233, AY173079, AY188347, AY188348, AY188349, AY188350, AY188351, AY188352, AY188353, AY188354, AY207051, AY207062, AY207064, AY216449, AY232832, AY232833, AY273147, AY277937, AY277938, AY277939, AY277940, AY277941, AY281076, AY281077, AY281078, AY281079, AY281080, AY281081, AY281082, AY281083, AY281084, AY281085, AY281086, AY281087

B.2 NON-TARGETS

B.2.1 Syntrophobacter

EF092240, AB303221, GU118971, AJ519665, AJ583203, U41563, EU803405, FN554392, GQ342374, HM339559, HM141886, AJ292577, DQ395918, DQ395761, AF498724, AY234728, AM162421, AM162405, AM887761, EU403966, FJ870384, AJ133631, AB004579, AB060161, AB059666, AB059679, AY140236, AY007662, DQ351931, AB269752, GQ240231,

AB024598, AF433165, DQ196466, EU284591, EU571146, AB116125, AB196471, EU876857, AY923143, EU427463, DQ113697, AY879308, HM251881, HM336484, AF213055, AB089842, AF460984, AY140239, AY911446, DQ533684, EF095720, FJ890913, FJ154517, EU419199, X89071, AB022035, L37424, DQ922995, EF522947, EU246229, AF282252, AF282253, FJ469298, DQ417202, CP001034, AJ417075, AY960571, DQ015078, EU504461, EU505269, EU506715, EU506742, X83946, AB050107, AB050108, AB050111, AJ416906, AY895186, AY895192, AY895197, AY895203, AY974823, AY975284, AY975390, AF493693, AY345549, AJ746140, DQ640009, EU800553, FJ593908, FJ849204, FJ849269, AY167327, EU052265, FJ202169, FJ202276, FJ202494, FJ202801, AJ229236, AF029039, AM157648, EU573107, EU626629, GU136582, GU454979, X78419, DQ814515, DQ817024, FJ456794, AF361018, AF248959, AJ289193, AJ295330, U43570, X89561, AY326627, AJ347027, DQ147278, DQ147287, AB360448, EF159861, AM936101, AM936303, AM936687, AM936719, AB038407, EF422408, Y10773, S83623, AB239484, FJ711181, FJ960443, FJ390117, FJ390125, FJ390131, HM288971, AJ234049, AJ234041, X81062, AJ234037, AJ234040, AF385505, AM084228, EU534527, GQ100969, GQ113810, AJ316570, AY039806, AJ241002, AY211662, DQ395003, DQ811833, CP001087, CP001087, FJ628304, X95180, AJ012591, AF449228, AB237692, AJ866934, EF442993, FJ712577, AF382396, CP000251, EU331409, GU731320, AB016470, EF999354, AY340830, DQ404769, GU339469, AF230531, M26635, AM086646, GU339475, X85132, X85131, CP000252, AM933651, GQ472447, FN429788, GU993263, AF170417, AF385080, FN356278, HM041921, X83274, AB195925, EF442978, AF170420, EU156147, AB212873, EU399662, FJ538126, GQ844327, GU208245, FJ462073

B.2.2 Syntrophomonas

EF092240, AB303221, GU118971, AJ519665, AJ583203, U41563, EU803405, FN554392, GQ342374, HM339559, HM141886, AJ292577, DQ395918, DQ395761, AF498724, AY234728, AM162421, AM162405, AM887761, EU403966, FJ870384, AJ133631, AB004579, AB060161, AB059666, AB059679, AY140236, AY007662, DQ351931, AB269752, GQ240231, AB024598, AF433165, DQ196466, EU284591, EU571146, AB116125, AB196471, EU876857, AY923143, EU427463, DQ113697, AY879308, HM251881, HM336484, AF213055, AB089842, AF460984, AY140239, AY911446, DQ533684, EF095720, FJ890913, FJ154517, EU419199, X89071, AB022035, L37424, DQ922995, EF522947, EU246229, AF282252, AF282253, FJ469298, DQ417202, CP001034, AJ417075, AY960571, DQ015078, EU504461, EU505269, EU506715, EU506742, X83946, AB050107, AB050108, AB050111, AJ416906, AY895186, AY895192, AY895197, AY895203, AY974823, AY975284, AY975390, AF493693, AY345549, AJ746140, DQ640009, EU800553, FJ593908, FJ849204, FJ849269, AY167327, EU052265, FJ202169, FJ202276, FJ202494, FJ202801, AJ229236, AF029039, AM157648, EU573107, EU626629, GU136582, GU454979, X78419, DQ814515, DQ817024, FJ456794, AF361018, AF248959, AJ289193, AJ295330, U43570, X89561, AY326627, AJ347027, DQ147278, DQ147287, AB360448, EF159861, AM936101, AM936303, AM936687, AM936719, AB038407, EF422408, Y10773, S83623, AB239484, FJ711181, FJ960443, FJ390117, FJ390125, FJ390131, HM288971, AJ234049, AJ234041, X81062, AJ234037, AJ234040, AF385505, AM084228, EU534527, GQ100969, GQ113810, AJ316570, AY039806, AJ241002, AY211662, DQ395003, DQ811833, CP001087, CP001087, FJ628304, X95180, AJ012591, AF449228, AB237692, AJ866934, EF442993, FJ712577, AF382396, CP000251, EU331409, GU731320, AB016470, EF999354, AY340830, DQ404769, GU339469, AF230531, M26635, AM086646, GU339475, X85132, X85131, CP000252, AM933651, GQ472447, FN429788, GU993263,

AF170417, AF385080, FN356278, HM041921, X83274, AB195925, EF442978, AF170420, EU156147, AB212873, EU399662, FJ538126, GQ844327, GU208245, FJ462073

B.2.3 Methanosarcina

AF191225, D85038, AY882744, DQ833821, DQ833822, DQ833823, DQ833831, DQ833832, DQ833833, DQ833844, DQ833865, DQ833866, DQ833873, DQ833882, DQ833883, DQ833884, DQ833885, DQ833886, DQ833920, DQ924700, DQ924792, DQ924793, DQ924794, DQ924795, DQ924796, EF057391, EF156537, EF156565, EF156566, EF156567, AY350586, AB087499, AY882700, DQ333311, DQ833934, DQ833935, DQ833936, DQ833937, DQ833955, DQ833956, DQ833957, DQ833958, DQ833959, DQ833961, DQ833962, DQ833963, DQ833984, DQ834014, EF057392, EF156482, EF156483, EF156484, EF156538, EF156540, FJ797313, FJ797316, FJ797323, FJ797327, FJ797334, FJ797337, AB104858, AB020530, AY196660, AY196661, X68711, X68712, X68713, X68714, X68715, X68716, X68717, X68718, X68719, X68720, X99046, X99047, X99048, Z37156, AF095262, AF095264, X15364, AB084240, AB084241, AY526511, AY526512, AY526513, AY526518, AE000666, AE000666, DQ657903, DQ657904, DQ683581, EF100758, DQ867043, DQ867048, DQ867050, EF198051, DQ649328, FJ418154, EU807735, FN547955, AB523785, AB539925, AB539926, AB539928, AB539929, AB539930, AB539931, HM041913, HM041914, CP001710, CP001710, AF050620, AJ308972, AB196288, AJ578125, CR626856, CR626857, CR626858, AB077217, AB236067, AB236112, AB236115, AB236118, AB236119, AB236120, AB243806, AM114193, AM114193, AM114193, EU155959, EU155960, EU155961, EU155962, EU155963, EU155964, EU155965, EU155966, EU155967, EU155968, EU155969, EU155970, EU155971, EU155972, EU155973, FJ685725, AP011532, AP011532, GU363061, AY692060, AY196683, AJ133792, AJ576220, M60880, AY570675, AY667273, AB092917, AB175345, AB175351, AB232795, AB232796, AB232798, AB232799, AB233300, AB233304, AB236073, AB236085, AB236087, AB236096, AB236101, AB244307, AB244743, AB248618, AB248619, CP000254, CP000254, CP000254, CP000254, AB266913, DQ841215, EU155983, EU432166, EU591660, EU591666, EU591673, EU591675, EU888808, EU888809, EU888810, EU888814, EU910621, FJ164110, AB447794, AB447800, AB447808, AB447812, AB447815, AB447827, AB447854, AB447867, AB447868, AB447869, AB447870, AB494239, AB494240, AB494243, AB517986, AB517987, GU135463, EU420713, HM187501, HM187507, HQ678044, HQ678059, HQ678095, M59932, AB301476, DQ925859, EU606020, AE009439, AY519654, X99570, AF411292, AY099164, AY099166, AY099167, AY099168, AY099169, AY099175, AY559125, D45214, L19921, Z54172, Z70246, Z70247, AJ225071, AJ419868, U20163, BA000001, AJ248283, AB193962, D87344, DQ167233, AB235311, AJ585956, AJ585957, AJ585959, EU682399, AE009950, FJ862775, FJ862776, FJ862777, FJ862778, FJ862779, FJ949575, AB603518, EF092240, EF629834, AB303221, EU622297, FJ202144, FJ202764, FJ203188, GU118971, AM997427, AY326582, AY289398, AF507710, AY921764, AJ863217, AM085462, DQ404639, EF019375, EF612358, EF492905, EF492913, EU131941, EU131942, EU131993, EU132093, EU132094, EU132338, EU132344, EU132439, EU669602, EU589288, EU979072, FJ478546, GQ214110, HM270156, HM186390, HM444873, HM444966, D83359, AF015929, D83353, D83354, D83355, D83356, D83357, D83358, D83360, D83361, D83362, D83363, D83364, D83365, D83366, D83367, D83368, D83369, D83370, D83372, D83373, D83374, D83371, AM157422, AM157429, AM157443, EU071501, EU418446, AY061974, EF092457, EU260047, EU260048, EU260049, GU584133, AB078038, AB078073, AB078076, AB189382, DQ836305, EF218994, DQ446174, EF123560, EU375076, EU491220, EU491708, EF687498, EU328009, EU328095, AM990845, FJ202064, FJ202244, FJ203289, FJ203328, FJ203387, AB433335,

AB443430, FJ205242, GQ259312, GU118565, GU118573, AM997917, AB540003, HQ397470, X54287, Z22730, AY015427, DQ883811, DQ883812, DQ923134, EF116933, EF116934, EF198330, GQ480939, GQ480940, GU479394, HM807296, HM807297, HM807303, AJ871305, AJ871306, AY731374, AB167239, AB193261, EU109511, EU289506, EU289511, FJ380134, FJ380135, FJ542906, GQ009187, HM270088, HM318948, FR682689, EU710748, AJ320223, AJ309733, AF393377, M83548, AE000657, AE000657, AE000657, AE000657, AJ132734, AJ132735, AJ132736, AJ132733, AJ001049, AJ431256, AJ507320, AF068784, AF068785, AF068787, AF068793, AF068799, AF068800, AF068808, AY268936, AY268938, AY268939, AY704389, DQ413022, DQ413023, AJ969464, AJ969466, AM268865, EF644681, AY605161, AF050593, AF050594, AF255600, AF419685, AY340822, AY297961, AY297962, AF507893, AY862535, AB252429, DQ329836, DQ329839, DQ329840, DQ329842, DQ329850, DQ329852, DQ329853, EF029852, EF515591, EF515667, AM712331, EU156145, EU266842, EU266849, EU266853, EU522662, EU644109, EU638713, EU981282, AB428365, FJ535510, FJ638586, FJ638604, FJ769502, FJ799125, AM490691, FJ375457, GU472722, GU390767, GQ203631, GU120616, HM041956, AF364564, AF364575, AF177275, AF083616, AF098330, EU683885, AB506677, AB506678, AF400484, U68460, AY140910, AY140911, EU326493, AF448723, FJ976094, FJ976095, AJ290825, AJ299413, Y08102, AJ290831, AY693833, Y10641, Y10642, Y10644, Y10646, AY394783, AY394784, AY394785, M58468, DQ383297, DQ383299, DQ383300, DQ383302, DQ383307, DQ383308, DQ383311, DQ383314, DQ383318, EF560700, EF560701, FJ484464, FJ485087, FJ485088, FJ485095, FJ485097, FJ485113, FJ485117, FJ485128, FJ485132, FJ485139, FJ485141, FJ485586, FJ902350, FJ902355, FJ716277, FJ716280, FJ716297, FJ716298, FJ716337, FJ716348, AE006470, AE006470, X86447, AB079642, AF039293, AY922003, EU133958, EU133998, EU134177, CP000875, CP000875, CP000875, CP000875, CP000875, GQ396860, HM341183, CP000875, CP000875, CP000875, CP000875, CP000875, X81319, AY820248, DQ666683, DQ991965, GQ922842, GQ922843, CP002432, CP002432, CP002432, AJ515881, AJ515882, AB086060, AB189456, U75602, AJ874309, AJ874313, DQ867052, EU407777, FN356326, AP011529, AP011529, AP011529, AP011529, X95744, AJ299402, AY570637, DQ079637, DQ079648, DQ991966, AJ430586, AY672508, AB107956, AB175519, AM268866, EU555123, CP002361, CP002361, X69194, L39875, AY861803, EU240006, EU240007, EU635938, EU635939, EU924243, CP001146, CP001146, CP001251, CP001251, FJ638602, FJ626840, HM004592, HM004611, CP001251, CP001146, CP001146, CP001251, AJ307981, AJ307982, AJ307980, EF061956, CP002281, CP002281, CP002281, CP002281, CP002281, CP002281, CP002282, CP002282, Y16799, Y16800, X84049, X54275, AY770718, DQ677014, DQ811895, EU052253, EU236316, FJ264772, FJ664817, FJ717186, FJ717187, FJ717188, M58678, EF608534, CP001739, CP001739, CP001739, CP001739, CP001739, CP001739, CP001739, CP001739, AJ441225, AY390428, AY390429, DQ889896, EU050935, EU050936, EU245555, EU617881, FJ197625, FJ202097, FJ202118, FJ202142, FJ202184, FJ202217, FJ202317, FJ202349, FJ202408, FJ202455, FJ202507, FJ202540, FJ202543, FJ202640, FJ202652, FJ202688, FJ202698, FJ202812, FJ202888, FJ202901, FJ202993, AB470952, GQ259325, FJ628253, GQ348819, AM997408, AM997901, GU230466, EU420746, HM799094, HQ673373, AJ231195, DQ814305, DQ814371, DQ814645, DQ814661, DQ814734, DQ814799, DQ815010, DQ815100, DQ815160, DQ815199, DQ815238, X64372, HM780068, CP002353, CP002353, CP002353, AY682384, AM162574, AJ633979, DQ917805, DQ917810, DQ917818, DQ486505, EU268103, EU287303, EU287310, EU287361, EU287385, EU407192, EF092165, EF092170, EF092185, EF092190, EU925871, EU919796, FJ205254, FJ746157, FJ847942, EU682495, GQ348566, GQ349441, GQ350589, GU117970, GU119022, AM997828, AM997838, GU289640, AJ421425, AF235130, AB167073, DQ015776, DQ015827, DQ015837, DQ015854, DQ521548, EF157203,

EF988634, EU143343, EU375040, EU375042, EU375044, FJ825822, FJ826117, EU740416, EU740417, EU740418, GU452538, GU452539, HM137558, HM137559, Z21632, M88719, AY714984, AY996806, DQ446116, DQ446117, DQ676343, EF520615, EF648066, EU386041, EU386042, EU491310, FJ203481, FJ710672, GQ354919, GU118906, FJ820401, GU591504, AF402980, DQ329719, EF515509, EF515634, EF454237, AB364473, GQ249604, FJ461889, FJ461890, FJ461891, FJ461892, FJ461893, FJ461894, FJ461895, FJ461896, FJ461897, FJ461898, FJ461899, FJ461900, FJ461901, FJ461902, FJ461903, FJ461904, FJ461905, L08066, HQ616114, AF129869, AY642589, AY642583, DQ431898, EU219938, EU245639, AM947514, EF999972, FJ674735, X96725, AF027096, AF332514, AF334601, AF418169, EU721792, FJ638609, FJ469306, FJ469345, FJ469353, AB539937, HM041951, AB534057, AY648568, L10658, L10659, AF509468, EF444748, EU245202, EU245215, EU245347, EU588727, EU721761, FJ469321, GQ203636, GQ203638, GQ203639, HM003101, HM037999, DQ097276, DQ486482, AB286015, AB286016, AB286017, AB286018, AB286019, AB297922, AY702164, AF289243, U21491, EU249955, EU249975, DQ851108, DQ851108, GU061657, GU062005, HQ671971, HQ672204, X70810, X70810, X70810, X70810, X12890, V00159, V00159, AJ294725, AJ294725, AY193169, AY193171, AF050611, AF423188, AF424767, AF424772, AJ312015, AF229774, AF229777, AY692053, AY692054, AY692055, AY692056, AY692057, AY692058, AY693812, AB071701, AY251025, X16932, X51423, AJ009508, AJ009509, AJ133791, AJ576211, AJ576221, AJ576227, AJ576230, AJ576240, M59141, M59146, AY570656, AY570662, AY570685, AY586394, AY667272, AY817738, AY426474, AY426475, AY426477, AY426479, AY970347, AJ937876, AB077211, AB084242, AB092914, AB175352, AB175353, AB175354, AB232791, AB232797, AB233292, AB233295, AB233299, AB236076, AB236094, AB244305, AB244306, AB244744, AY835417, AB248604, AB248605, AB248606, AB248607, AB248608, AB248611, AB248612, AB248613, AB248614, AB248615, DQ478742, AB266890, AB266891, AB266892, AB266894, AB266904, AB266919, CP000477, CP000477, DQ841239, DQ841240, DQ867049, DQ522924, EF198034, EF198035, EF198049, EF198050, EF198052, AB294257, DQ339718, DQ339719, EU155900, EU155901, EU155902, EU155903, EU155904, EU155905, EU155906, EU155907, EU155908, EU155909, EU155910, EU155911, EU155912, EU155913, EU155914, EU155915, EU155916, EU155917, EU155945, EU155946, EU155948, EU155949, EU155950, EU155951, EU155952, EU155953, EU155954, EU155955, EU155956, AB329663, AB329664, AM745179, AM745249, AM746093, AB434763, AB434765, AB434767, EU155947, EU580025, EU580026, EU580027, EU580028, EU580029, EU580030, EU580031, EU580033, EU580034, EU580035, EU580036, EU580037, EU580038, EU580039, EU580040, EU580041, EU580042, EU580043, EU580044, EU580045, EU591661, EU591663, EU591668, EU591670, EU591671, EU591674, EU662669, EU662681, EU662697, EU888804, EU888805, EU888806, EU888811, EU888812, EU888815, EU910619, EU910626, FJ164111, FJ167429, FJ167430, FJ167431, FJ167433, FJ167437, EU857625, EU857629, EU857630, EU857632, AB447761, AB447762, AB447764, AB447765, AB447768, AB447769, AB447770, AB447772, AB447775, AB447776, AB447777, AB447778, AB447779, AB447780, AB447781, AB447782, AB447784, AB447785, AB447786, AB447788, AB447789, AB447790, AB447792, AB447793, AB447797, AB447798, AB447802, AB447803, AB447804, AB447806, AB447807, AB447809, AB447810, AB447813, AB447814, AB447816, AB447817, AB447818, AB447820, AB447822, AB447824, AB447825, AB447826, AB447828, AB447833, AB447834, AB447836, AB447837, AB447839, AB447840, AB447841, AB447842, AB447843, AB447844, AB447846, AB447847, AB447849, AB447850, AB447851, AB447853, AB447855, AB447856, AB447857, AB447858, AB447859, AB447860, AB447861, AB447864, AB447865, AB447866, AB447871, AB447873, AB447874, AB447875, AB447878, AB447879, EU721745, EU721747, EU721751, EU721755, AB479392, AB479394, AB479409, AB479410, FJ638501, FJ638502,

FJ638505, FJ638506, FJ638507, FJ638509, FJ638510, FJ638512, FJ638513, FJ705108, FJ705109, FJ705113, FJ705115, FJ705116, FJ705125, FJ705126, FJ705128, FJ705129, FJ712370, FJ712391, FJ712397, AB494241, AB494254, FJ971742, FJ971745, FJ971746, FJ973573, AM229252, FN429785, AB550818, AB550819, AB550820, GU135459, GU135460, GU135461, GU591524, AB539923, AB539924, HM041906, GU388805, GU389068, GU389112, HQ588687, HQ592613, HQ592618, HQ592619, HQ592620, HQ592624, HQ592625, HQ677943, HQ677988, HQ678099

B.2.4 Streptococcus

EF092240, EF629834, AB303221, EU622297, FJ202144, FJ202764, FJ203188, GU118971, AM997427, AY326582, AY289398, AF507710, AY921764, AJ863217, AM085462, DQ404639, EF019375, EF612358, EF492905, EF492913, EU131941, EU131942, EU131993, EU132093, EU132094, EU132338, EU132344, EU132439, EU669602, EU589288, EU979072, FJ478546, GQ214110, HM270156, HM186390, HM444873, HM444966, D83359, AF015929, D83353, D83354, D83355, D83356, D83357, D83358, D83360, D83361, D83362, D83363, D83364, D83365, D83366, D83367, D83368, D83369, D83370, D83372, D83373, D83374, D83371, AM157422, AM157429, AM157443, EU071501, EU418446, AY061974, EF092457, EU260047, EU260048, EU260049, GU584133, AB078038, AB078073, AB078076, AB189382, DQ836305, EF218994, DQ446174, EF123560, EU375076, EU491220, EU491708, EF687498, EU328009, EU328095, AM990845, FJ202064, FJ202244, FJ203289, FJ203328, FJ203387, AB433335, AB443430, FJ205242, GQ259312, GU118565, GU118573, AM997917, AB540003, HQ397470, X54287, Z22730, AY015427, DQ883811, DQ883812, DQ923134, EF116933, EF116934, EF198330, GQ480939, GQ480940, GU479394, HM807296, HM807297, HM807303, AJ871305, AJ871306, AY731374, AB167239, AB193261, EU109511, EU289506, EU289511, FJ380134, FJ380135, FJ542906, GQ009187, HM270088, HM318948, FR682689, EU710748, AJ320223, AJ309733, AF393377, M83548, AE000657, AE000657, AE000657, AE000657, AJ132734, AJ132735, AJ132736, AJ132733, AJ001049, AJ431256, AJ507320, AF068784, AF068785, AF068787, AF068793, AF068799, AF068800, AF068808, AY268936, AY268938, AY268939, AY704389, DQ413022, DQ413023, AJ969464, AJ969466, AM268865, EF644681, AY605161, AF050593, AF050594, AF255600, AF419685, AY340822, AY297961, AY297962, AF507893, AY862535, AB252429, DQ329836, DQ329839, DQ329840, DQ329842, DQ329850, DQ329852, DQ329853, EF029852, EF515591, EF515667, AM712331, EU156145, EU266842, EU266849, EU266853, EU522662, EU644109, EU638713, EU981282, AB428365, FJ535510, FJ638586, FJ638604, FJ769502, FJ799125, AM490691, FJ375457, GU472722, GU390767, GQ203631, GU120616, HM041956, AF364564, AF364575, AF177275, AF083616, AF098330, EU683885, AB506677, AB506678, AF400484, U68460, AY140910, AY140911, EU326493, AF448723, FJ976094, FJ976095, AJ290825, AJ299413, Y08102, AJ290831, AY693833, Y10641, Y10642, Y10644, Y10646, AY394783, AY394784, AY394785, M58468, DQ383297, DQ383299, DQ383300, DQ383302, DQ383307, DQ383308, DQ383311, DQ383314, DQ383318, EF560700, EF560701, FJ484464, FJ485087, FJ485088, FJ485095, FJ485097, FJ485113, FJ485117, FJ485128, FJ485132, FJ485139, FJ485141, FJ485586, FJ902350, FJ902355, FJ716277, FJ716280, FJ716297, FJ716298, FJ716337, FJ716348, AE006470, AE006470, X86447, AB079642, AF039293, AY922003, EU133958, EU133998, EU134177, CP000875, CP000875, CP000875, CP000875, CP000875, GQ396860, HM341183, CP000875, CP000875, CP000875, CP000875, CP000875, X81319, AY820248, DQ666683, DQ991965, GQ922842, GQ922843, CP002432, CP002432, CP002432, AJ515881, AJ515882, AB086060, AB189456, U75602, AJ874309, AJ874313, DQ867052, EU407777, FN356326, AP011529, AP011529,

AP011529, AP011529, X95744, AJ299402, AY570637, DQ079637, DQ079648, DQ991966, AJ430586, AY672508, AB107956, AB175519, AM268866, EU555123, CP002361, CP002361, X69194, L39875, AY861803, EU240006, EU240007, EU635938, EU635939, EU924243, CP001146, CP001146, CP001251, CP001251, FJ638602, FJ626840, HM004592, HM004611, CP001251, CP001146, CP001146, CP001251, AJ307981, AJ307982, AJ307980, EF061956, CP002281, CP002281, CP002281, CP002281, CP002281, CP002281, CP002282, CP002282, Y16799, Y16800, X84049, X54275, AY770718, DQ677014, DQ811895, EU052253, EU236316, FJ264772, FJ664817, FJ717186, FJ717187, FJ717188, M58678, EF608534, CP001739, CP001739, CP001739, CP001739, CP001739, CP001739, CP001739, CP001739, AJ441225, AY390428, AY390429, DQ889896, EU050935, EU050936, EU245555, EU617881, FJ197625, FJ202097, FJ202118, FJ202142, FJ202184, FJ202217, FJ202317, FJ202349, FJ202408, FJ202455, FJ202507, FJ202540, FJ202543, FJ202640, FJ202652, FJ202688, FJ202698, FJ202812, FJ202888, FJ202901, FJ202993, AB470952, GQ259325, FJ628253, GQ348819, AM997408, AM997901, GU230466, EU420746, HM799094, HQ673373, AJ231195, DQ814305, DQ814371, DQ814645, DQ814661, DQ814734, DQ814799, DQ815010, DQ815100, DQ815160, DQ815199, DQ815238, X64372, HM780068, CP002353, CP002353, CP002353, AY682384, AM162574, AJ633979, DQ917805, DQ917810, DQ917818, DQ486505, EU268103, EU287303, EU287310, EU287361, EU287385, EU407192, EF092165, EF092170, EF092185, EF092190, EU925871, EU919796, FJ205254, FJ746157, FJ847942, EU682495, GQ348566, GQ349441, GQ350589, GU117970, GU119022, AM997828, AM997838, GU289640, AJ421425, AF235130, AB167073, DQ015776, DQ015827, DQ015837, DQ015854, DQ521548, EF157203, EF988634, EU143343, EU375040, EU375042, EU375044, FJ825822, FJ826117, EU740416, EU740417, EU740418, GU452538, GU452539, HM137558, HM137559, Z21632, M88719, AY714984, AY996806, DQ446116, DQ446117, DQ676343, EF520615, EF648066, EU386041, EU386042, EU491310, FJ203481, FJ710672, GQ354919, GU118906, FJ820401, GU591504, AF402980, DQ329719, EF515509, EF515634, EF454237, AB364473, GQ249604, FJ461889, FJ461890, FJ461891, FJ461892, FJ461893, FJ461894, FJ461895, FJ461896, FJ461897, FJ461898, FJ461899, FJ461900, FJ461901, FJ461902, FJ461903, FJ461904, FJ461905, L08066, HQ616114, AF129869, AY642589, AY642583, DQ431898, EU219938, EU245639, AM947514, EF999972, FJ674735, X96725, AF027096, AF332514, AF334601, AF418169, EU721792, FJ638609, FJ469306, FJ469345, FJ469353, AB539937, HM041951, AB534057, AY648568, L10658, L10659, AF509468, EF444748, EU245202, EU245215, EU245347, EU588727, EU721761, FJ469321, GQ203636, GQ203638, GQ203639, HM003101, HM037999, DQ097276, DQ486482, AB286015, AB286016, AB286017, AB286018, AB286019, AB297922, AY702164, AF289243, U21491, EU249955, EU249975, DQ851108, DQ851108, GU061657, GU062005, HQ671971, HQ672204, X70810, X70810, X70810, X70810, X12890, V00159, V00159, AJ294725, AJ294725, AY193169, AY193171

B.3 BISTRO-PRIMER RESULTS

Tables B.1–B.4 contain the complete results generated by Bistro-Primer for the 4 data sets evaluated.

Forward	Reverse	Product size	Forward targets	Reverse targets	Forward non targets	Reverse non targets	Paired targets	Paired non targets	Score
TACCAGAACGGGTTCGACGGTG	CCACCCGTTGTTGTGCTCCC	214	309	314	0	1	302	0	0.93
TAACACCGGCGGCCCGAG	GCGACGGCCATGCACCTC	533	311	311	33	29	300	0	0.93
CGTCTTACCAGAACGGGTTCGACG	GTGTAGCCCGGGARATTCGGGGC	500	307	315	0	27	299	0	0.92
TTATTGGGTCTAAAGGGTCC	CTCGTTGCCTGACTTAAC	544	313	307	0	443	297	0	0.92
GCCCGGAGATGGATTCTGAGAC	ACGGGTCTCGCTCGTTGC	760	313	306	395	0	297	0	0.92
TACCCGGGTAGTCCCAGCC	TCACCGCGCTATATTGAAACGC	584	313	303	0	0	294	0	0.91
TACCCGGGTAGTCCCAGC	TTAAGTTTCAGCCTTGCGGC	126	315	306	95	416	300	7	0.9
TCTTACCAGAACGGGTTCGAC	ATTCCTTTAAGTTTCAGCCTTGCG	191	308	304	0	479	291	0	0.9
TGTCAGGCATGGCGCGACCGTG	CAGTGGGCACGGGTCTCGCTCG	292	312	301	0	0	291	0	0.9
TGGTGATCGTGATTATTGG	TCCCATYCATTGTAGCCCG	703	298	311	0	0	290	0	0.9
GGCGTCTTACCAGAACGGGTTC	ACGGGTCTCGCTCGTTGC	385	304	306	0	0	290	0	0.9
AGTGGTGATCGTGATTATTGGG	TTGTCCCATYCATTGTAGCCCG	708	298	310	0	0	289	0	0.89
CCCCGAATYTCCCGGGCTACACGC	CTCACTCGGGTGGTTTGACGGGC	216	316	294	27	54	289	0	0.89
CAAGAGCCCGGAGATGGATTCTG	CTTCCCTGCGGCACCAGAC	521	314	295	388	0	289	0	0.89
ATGGCGCGACCGTGTCTGG	CGGCCATGCACCTCCTCTCAG	221	303	308	0	159	288	0	0.89
TGGGTCTAAAGGGTCCGTAGCCGG	TCACGGCTTCCCTGCGGCAC	312	307	300	0	0	285	0	0.88
TCGTACTGTGAAGCATCCTG	ATYCATTGTAGCCCGCGTGTAG	180	309	312	350	29	299	15	0.88
GGGTGTAATGTACCTACTAGCC	ACCTCTTACCTCTCCCGG	372	291	316	27	1	284	0	0.88
AACACGTGGATAACCTGCCCTTG	AACCCGTTCTGGTAAGACGCC	584	300	304	25	0	284	0	0.88
AACTTTACAATGCGGGAAACCGTG	GCCGTACTTCCCAGGTGGC	489	304	306	18	276	287	3	0.88
GATGCTCGCTAGGTGTCAGG	TTCACGAGGGCGAGTTACAG	518	300	291	1	23	283	0	0.87
CAAGGATGGGTCTGCGGCCTATC	RTCAGATTTCCCGGAGGACTGACC	345	296	305	32	0	282	0	0.87
AACGATGCTCGCTAGGTGTCAGG	GCTTCACAGTACGAACTGGCGAC	267	300	305	1	0	282	0	0.87
TAAAGGGTCCGTAGCCGGTTTGG	ACACCTAGCGAGCATCGTTTACGG	263	300	301	0	37	278	0	0.86
AGGCGTCTTACCAGAACGGGTTCG	TCACTCGGGTGGTTTGACGGGC	696	303	294	0	55	277	0	0.85
CGGGCYACGGTAGGTCAGTATGC	CGGATTCCAGCTTCACGAGGG	159	309	288	2	11	277	0	0.85
GCCCAAGGATGGGTCTGCGGC	CGGGCCGCCGGTGTTACCG	279	298	310	64	33	288	12	0.85
GTGRGACCACCTGTGGCGAAGGC	ACTACGGATTCCAGCTTCACGAGG	641	305	288	0	11	274	0	0.85
GCCTGAATCGCTGAGAGGAGG	AGTTACAGCCCTCGATCCGAAC	295	289	300	0	23	271	0	0.84
AGACYTTGCCTGAATCGC	CRGGGKAGGGACCCATTGTC	248	293	286	0	0	265	0	0.82
GGTGTTCGCCTAAGCCATG	ACAAGATTTCACTCCTACCCCTG	583	273	313	0	0	264	0	0.81
ATAACCTGCCCTTGGGWCCGG	CATTGTCCCATYCATTGTAGCCCG	1081	271	309	37	0	263	0	0.81
TGTTCGCCTAAGCCATGC	ACCTACCGTRGCCCGCAC	1081	273	313	0	0	262	0	0.81
TGTTCGCCTAAGCCATGCG	TCGTCCCTCACCGTCGAAC	645	272	308	0	0	260	0	0.8
TGGATAACCTGCCCTTGGGWCC	GGGGCATACTGACCTACCG	1041	268	314	30	371	260	0	0.8

GAGCCTGCGGTTTAATTGG	CGAGTTACAGCCCTCGATC	376	298	300	138	23	277	18	0.8
ACTGCTATCGGTGTTCGCCTAAG	CATYCATTGTAGCCCGCGTGTAGC	1138	270	309	0	6	259	0	0.8
TGCCCAAGGATGGGTCTGCGG	AGGTGGCTCGCTTCACGGCTTC	622	274	307	51	287	261	5	0.79
AGGAATTGGCGGGGGAGCACAAC	CATGCTGGTAACAGTGGGCACGGG	215	309	262	1	0	254	0	0.78
CTCGCCCTCGTGAAGCTGG	GGTGTGTGCAAGGAGCAGGG	81	289	307	28	453	276	25	0.77
GGATAACCTGCCCTTGGGWCC	TGAGTCCAATTAAACCGCAGG	800	273	297	37	22	251	0	0.77
TGCCAGACTTGGAACCGGGAGAGG	AGGCTCCACCCGTTGTTGTGCTC	312	257	310	0	1	247	0	0.76
GGATAACCTGCCCTTGGGWCCGG	TTTCAGCCTTGCGGCCGTAC	740	271	305	37	480	253	35	0.67
GGTGGAGCCTGCGGTTTAATTGG	GGGTGGTTTGACGGGCGG	473	297	295	91	343	273	73	0.62
GCGTACTGCTCAGTAACACGTG	ATAGGCCGCAGACCCATCC	140	282	303	117	287	264	71	0.6
GGATAACGCATATVTGCTGGAATG	CCGCCAATTCCTTTAAGTTTC	705	197	310	0	533	192	0	0.59
GTGTTCGCCTAAGCCATGCG	CCCAAGGGCAGGTTATCCACG	71	272	301	0	26	190	0	0.59
CWACGACGGGTACGGGTTG	GGTGTCCCCTTATCACGG	124	172	296	288	0	168	0	0.52
AGCCWACGACGGGTACGGG	AATAATCACGATCACCACTCGGG	244	171	296	292	0	164	0	0.51
CTACTAGCCWACGACGGGTACGG	TAATCCGGTTCGTGCCCCC	474	169	312	290	3	162	1	0.5
CCTACTAGCCWACGACGGGTACG	TAATCCGGTTCGTGCCCCC	475	169	312	290	3	162	1	0.5
CTGCGGCCTATCAGGTAGTAG	CTCAGAATCCATCTCCGGGC	93	314	315	230	397	309	224	0.26
AAGGATGGGTCTGCGGCCTATC	TTTAAGTTTCAGCCTTGCGGCCG	648	298	303	243	399	281	219	0.19
CAAGAGCCCGGAGATGGATTCTG	CCCAGGTGGCTCGCTTCAC	542	314	304	388	290	296	273	0.07

Table B.1: Bistro-Primer results for the genus *Methanosarcina* with 16S rRNA target region. The forward and reverse primers are in 5'-3' direction. The maximum number of targets is 324 and the maximum number of non-targets is 1167.

Forward	Reverse	Product size	Forward targets	Reverse targets	Forward non targets	Reverse non targets	Paired targets	Paired non targets	Score
TGAWGAAGGCCTTCGGGTCG	CCCGTCAATTCCTTTGAGTTTTAG	521	83	82	15	53	81	5	0.87
CTGCTGTGCCGYAGCTAACGCG	ATGAGGACTTGACGTCATCCCCAC	358	80	82	3	45	77	2	0.86
TCTGATGTGAAAGCCCYGGGC	ATGCTGATCCGCGATTACTAG	768	83	82	3	62	78	3	0.86
AGCAGTGAGGAATTTTGCGC	CCGGGAATTCCCCCTTCC	332	84	79	13	4	75	3	0.83
AGGTGTAGCGGGTACTCATTC	CGTATTCACCGCGGCATG	553	76	81	0	45	72	0	0.83
TGTAGCGGGTACTCATTCCTGCTG	CATCTCACGACACGAGCTGACG	256	76	85	3	155	74	3	0.82
TGGTTTAATTCGACGCAAC	AMCTTCAYGGAGTCGAGTTGCAG	386	81	83	50	28	78	8	0.8
ATGATCAGCCACACTGGCACTGG	CTGACGACAGCCATGCAGCAC	769	76	80	3	74	70	2	0.78
TGGGTGAWGAAGGCCTTCGGGTC	CCTCCGTATTACCGCGGCTG	138	82	80	9	62	75	7	0.78
GTAGCGGGTACTCATTCCTGCTG	ACAGCCATGCAGCACCTGTC	233	76	79	3	40	71	3	0.78
AGTCTGATGTGAAAGCCCYGGGC	RKCCGGGGATGTCAAGCCC	410	83	73	3	3	71	3	0.78
GGCCTTCGGGTCGTAAAGCCCTG	TCACCGCGGCATGCTGATCCG	961	79	81	10	55	76	8	0.78
AGTCCACGCTGTAAACGATG	CAGGCGGAKCACTTAACGCG	85	82	73	21	3	70	3	0.77
CTGTGCCGYAGCTAACGCGTTAAG	GCTCGTTGCGGGACTTAACC	258	74	82	3	153	70	3	0.77
AGCCGCGGTAATACGGAGGGTGC	GCTCCCCACGCTTTCGCGTCTC	257	79	83	37	17	77	10	0.77
GGGCCTGCGTCCTATCAG	TTACGACCCGAAGGCCTTCWTC	202	71	83	0	22	67	0	0.77
GGCGTAAAGCGCGTGYAG	AGGACTTGACGTCATCCC	637	77	82	13	52	73	7	0.76
AGCGTTATTCGGAATTACTGGG	CGTTGCGTCGAATTAAACCAC	425	75	81	8	50	69	3	0.76
GACGCGAAAGCGTGGGGAG	ATGCAGCACCTGTCTCCCGG	299	83	72	17	4	68	2	0.76
TGTAGCGGGTACTCATTCCTGC	AGGTTCTTCGCGTTGCGTC	151	76	79	3	31	68	3	0.75
AGGCCTTCGGGTCGTAAAGC	CTRCACGCGCTTTACGCCCAG	171	83	74	28	9	70	5	0.75
TGGAGAGGAAGGGGGAATTCC	ATGTCAAGCCCAGGTAAGGTTC	338	76	72	0	20	65	0	0.75
AAGGCCTTCGGGTCGTAAAGCC	ACCCTCCGTATTACCGCGGC	131	80	81	10	38	74	9	0.75
AWGAAGGCCTTCGGGTCGTAAAGC	CTRCACGCGCTTTACGCCCAG	175	83	74	19	9	70	5	0.75
TAGCGGGTACTCATTCCTGCTG	AGGTAAGGTTCTTCGCGTTGC	154	76	79	3	63	68	3	0.75
AAGGCCTTCGGGTCGTAAAGCC	CACCCTCCGTATTACCGCGGC	132	80	81	10	37	74	9	0.75
WGAAGGCCTTCGGGTCGTAAAG	CGCGCTTTACGCCCAGTAATTC	169	83	74	19	6	70	6	0.74
TTCGGAATTACTGGGCGTAAAGCG	CCCACGCTTTCGCGTCTCAG	222	73	82	6	21	69	6	0.72
AGGAATACCAGTGGCGAAG	YGGAGTCGAGTTGCAGAC	619	71	84	17	51	67	4	0.72
TGCCGYAGCTAACGCGTTAAGTG	CAACATCTCACGACACGAGCTG	235	71	84	7	148	68	5	0.72
TGCGCAATGGSMGCAAKSCTGACG	GCGTTGCGTCGAATTAAAC	606	67	81	7	51	65	3	0.71
CGACGACGGGTAGCTGGTC	TACAGCGTGGACTACCAGGGTATC	539	67	80	10	29	65	3	0.71
GTCGTAAAGCCCTGTCAG	CTCCCGATCTCTACGAATTTC	290	65	86	7	2	63	2	0.7
GACGACGGGTAGCTGGTCTG	AAAGGCCATGAGGACTTGACGTC	937	67	83	10	37	66	5	0.7
WACTGACGCTGAGACGCGAAAGC	GGGTTGCGCTCGTTGCGGG	369	79	83	19	154	75	14	0.7

GGGAGGAATACCAGTGGCGAAGGC	CGACAGCCATGCAGCACCTGTC	358	68	79	3	39	62	2	0.69
AAAGCCCTGTCAGGTGGG	ACGTCATCCCCACCTTCC	767	65	83	3	126	63	3	0.69
AGGATGATCAGCCACACTGGC	CTTCCCACCTGACAGGGCTTTAC	151	84	62	4	3	61	1	0.69
GCCAGACTCCTACGGGAGGC	CATCTCACGACACGAGCTGACG	755	82	85	30	155	80	21	0.68
TCACAGTTCGGATYGGAG	TYACCRACCAYACCTTGGTACG	190	83	62	3	2	59	0	0.68
GCAGCCGCGGTAATACGG	CTCCRATCCGAACTGTGAACGGC	793	80	68	72	3	62	3	0.68
AGGCCTTCGGGTCGTAAAGCC	CCTGGGCATAAAGGCCATGAGGAC	809	80	61	10	7	59	1	0.67
GGATGGGCCTGCGTCCTATCAGC	TTCCTCACTGCTGCCTCCCGTAGG	139	63	82	0	59	58	0	0.67
GATGAGCACTAGGTGTAGCGG	ARGGCARGGGTTGCGCTCGTTG	306	65	78	0	18	58	0	0.67
ATTAGATACCCTGGTAGTCCAC	TTAGTCTTGCGACCGTAATC	122	83	61	145	3	61	3	0.67
GCGGTAATACGGAGGGTGCRAGCG	ACCGCGGCATGCTGATCCG	847	78	82	35	55	74	16	0.67
GTCCCGCAACGAGCGCAACC	RACTTTCGTGGTGTGACGGGCG	327	82	70	156	12	66	9	0.66
TGAGACGCGAAAGCGTGGGG	AAGCCCAGGTAAGGTTCTTCGCG	236	83	72	18	29	69	12	0.66
GCTGGTCTGAGAGGATGATCAGCC	CAGAAGGGCGCCTTCGCC	454	83	64	24	4	61	4	0.66
GATTAGATACCCTGGTAGTCCAC	CRACCAYACCTTGGTACGCTGC	696	83	62	145	4	60	4	0.64
CAGAGGAAGCACCGGCTAACTC	CAGGTAAGGTTCTTCGCGTTGC	496	72	78	9	62	65	9	0.64
AGGAATACCAGTGGCGAAGGC	GCAGCACCTGTCTCCCGG	345	68	73	17	4	56	0	0.64
AGCGTTATTCGGAATTACTGGG	TGGCAACTAARGGCARGGGTTGC	586	75	64	8	13	59	3	0.64
RGGATGGGCCTGCGTCCTATCAG	CGACAGCCATGCAGCACCTGTC	843	63	79	0	39	56	0	0.64
TGGTCTGAGAGGATGATCAGC	TTTGTACCGCCCATTGTAGTAC	964	84	60	24	4	58	2	0.64
TGAGTACTGGAGAGGAAGGGG	ACACGAGCTGACGACAGC	425	60	83	0	86	56	0	0.64
CTTCGGGTCGTAAAGCCCTG	ACTTTCGTGGTGTGACGGG	1004	79	70	10	12	62	6	0.64
AAGCGTGGGGAGCAAACAGG	RGGCARGGGTTGCGCTCGTTG	356	82	78	72	22	73	17	0.64
CCGCGTGGGTGAWGAAGGC	GGGCGGTGTGTACAAGGC	1007	79	75	15	129	67	12	0.63
GGGAGCAAACAGGATTAGATACCC	GATGTCAAGCCCAGGTAAGGTTC	225	81	72	79	16	68	13	0.63
CTGCATGGCTGTCGTCAGC	AGCCCACGCACTTCTGGTAC	394	80	66	74	6	61	6	0.63
GAAGCACCGGCTAACTCCGTGC	CTTGCGACCGTAATCCCCAGGC	407	82	60	42	3	58	3	0.63
TGTAAACGATGAGCACTAGGTG	RKCCGGGGATGTCAAGCC	195	63	76	0	3	55	0	0.63
CCTTCGGGTCGTAAAGCCC	TTCCCCCTTCCTCTCCAGTAC	260	80	59	10	0	55	0	0.63
TTCCTGCTGTGCCGYAGCTAACG	TTAGCCCACGCACTTCTGGTAC	603	77	64	3	6	57	3	0.62
GACGGGTAGCTGGTCTGAGAG	GTACCGCCCATTGTAGTACGTG	971	82	61	24	4	58	4	0.62
CTACACACGTACTACAATG	ACCRACCAYACCTTGGTACG	260	77	62	24	2	54	0	0.62
CAACGCGAAGAACCTTAC	ACCAYACCTTGGTACGCTG	513	79	62	120	4	58	4	0.62
GCGCAACCCYTGCCYTTAGTTGCC	CGTATTCACCGCGGCATGC	273	65	81	13	44	60	7	0.61
YGGCCTACCAAGGCGACGACGGG	GTATTACCGCGGCTGCTGGCACGG	274	62	80	5	71	57	4	0.61
GATGACGTCAAGTCCTCATG	GATCCGCGATTACTAGCG	173	81	85	53	56	80	27	0.61
AAGCACCGGCTAACTCCGTGCC	TCACCGCGGCATGCTGATCCG	878	82	81	54	55	78	26	0.6
CAACCCYTGCCYTTAGTTGCCAKC	TCCAMCTTCAYGGAGTCGAGTTG	230	64	84	13	28	62	10	0.6
TGTTGGGTTAAGTCCCGCAACGAG	ACTTTCGTGGTGTGACGGGCG	337	81	70	152	12	64	12	0.6

GAGCGCAACCCYTGCCYTTAGTTG	ACTGTGAACGGCTTTTTGGGRTTG	197	80	60	14	3	54	2	0.6
GGGGAGCAAACAGGATTAGATAC	CCTTTGTACCGCCCATTGTAG	483	81	67	79	10	61	9	0.6
TAAAGCCCTGTCAGGTGGG	ATGCAGCACCTGTCTCCC	631	65	75	3	4	53	2	0.59
ACTCAAAGGAATTGACGGGGG	AGCCCTGGGCATAAAGGC	316	82	62	169	9	60	9	0.59
ACGACGGGTAGCTGGTCTG	TGGCAACTAARGGCARGGGTTGCG	855	73	64	14	13	54	3	0.59
GCGTGCYTAACACATGCAAGTCG	CAGGAATGAGTACCCGCTACAC	822	63	76	72	3	54	3	0.59
GCCTGCGTCCTATCAGCTRGTTG	CRACTTTCGTGGTGTGACGGGC	1193	58	70	3	12	53	3	0.57
AAKSCTGACGCAGCAACGCCGCG	GCCTRCACGCGCTTTACGCCCAG	206	68	74	21	9	56	6	0.57
RCGAAAGYKSTGCTAATACCGG	CGATTACTAGCGATTCCAMC	1214	56	84	3	54	53	3	0.57
AGAGGAAGCACCGGCTAACTCC	CCACGCACTTCTGGTACAGCCRAC	951	72	68	12	7	56	6	0.57
ACGCTGGCGGCGTGCYTAACAC	AGCCATGCAGCACCTGTCTCCC	1042	61	75	61	4	52	3	0.56
TGGCGGCGTGCYTAACACATGC	AGGAGTCTGGCCCGTGTTCC	318	66	71	65	11	57	9	0.55
GATGGGCCTGCGTCCTATCAGC	ACTTTCGTGGTGTGACGGGCG	1196	63	70	0	12	48	0	0.55
ACACACGTACTACAATGGGCGG	TGTGTACAAGGCCCGGGAAC	174	61	77	4	122	52	4	0.55
TGAGCACTAGGTGTAGCGG	TTTGTACCGCCCATTGTAGTAC	434	66	60	4	4	48	0	0.55
ACTGGAGAGGAAGGGGGAATTCCC	TGCAGCACCTGTCTCCCGG	400	59	72	0	4	48	0	0.55
YWCRRGGATGGGCCTGCGTCC	ACCAGGGTATCTAATCCTGTTTG	582	50	81	0	125	46	0	0.53
ACTGGAGAGGAAGGGGGAATTCC	TCAGTWWCCGTCCAGAAGGGCGC	96	59	52	0	4	44	0	0.51
CTGGAACACGGGCCAGAC	CAGGTTAAGCCCRGGGCTTTCAC	306	71	53	11	2	45	2	0.49
AGTACTGGAGAGGAAGGGGG	TYCCCTTTGTACCGCCCATTGTAG	604	60	58	0	3	41	0	0.47
AGCAGCCGCGGTAATACGGAGGG	GGGCGGTGTGTACAAGGCCCG	886	80	75	40	128	69	28	0.47
RRGGATGGGCCTGCGTCCTATCAG	RGTTAGCCCACGCACTTCTGGTAC	1228	60	63	0	6	40	0	0.46
TGGACGGWWACTGACGCTG	GTGAACGGCTTTTTGGGRTTG	560	61	61	11	3	42	3	0.45
TAACACATGCAAGTCGVACG	CCTTCCTCTCCAGTACTC	635	57	60	54	0	39	0	0.45
TAACTCCGTGCCAGCAGCC	TGCACTTCCCAGGTTAAGCCCRGG	128	82	44	78	2	41	2	0.45
TGTGGTTTAATTCGACGCAACGCG	CCCAACATCTCACGACACGAGCTG	142	81	84	50	148	79	40	0.45
CGGTAATACGGAGGGTGCRAGCG	TCTCCCGGTCCCCCGAAGGRG	519	78	42	35	1	40	1	0.45
GAGAGGAAGGGGGAATTC	CGGTCCCCCGAAGGRGAAMWC	380	79	41	4	0	38	0	0.44
TGGGCGTAAAGCGCGTGYAGGC	CTGTCTCCCGGTCCCCCGAAGGRG	485	76	42	13	1	39	1	0.44
TGGACGGWWACTGACGCTGAGAC	GAACGGCTTTTTGGGRTTGGCTC	558	60	61	10	3	41	3	0.44
GTRGGGTAAYGGCCTACCAAGGCG	GGATGTCAAGCCCAGGTAAGG	746	53	74	18	13	45	8	0.43
CCYGGGCTTAACCTGGGAAGTGC	TTAGTCTTGCGACCGTAATCCCC	297	44	61	2	3	39	2	0.43
CATGCAAGTCGVACGAGAAAGSC	TGAACGGCTTTTTGGGRTTG	1257	56	61	6	3	36	0	0.41
GAGTAACGCGTAGGYAACCTACCC	ATGAGGACTTGACGTCATCCC	1109	38	82	0	51	35	0	0.4
WCRRGGATGGGCCTGCGTC	RACCAYACCTTGGTACGCTG	1260	50	62	0	4	34	0	0.39
YYKKWCTTGAGTACTGGAGAGG	AGACTCCRATCCGAACTG	671	34	84	0	46	33	0	0.38
TGTAGCGGGTACTCATTCCTGC	GCBRKCCGGGGATGTCAAGCCCAG	178	76	37	3	2	34	2	0.37
CAGCAGCCGCGGTAATAC	ATGAGGACTTGACGTCATCCC	688	80	82	158	51	75	45	0.34
GDKTGACGGTACCACCAGAGGAAG	RATCCGAACTGTGAACGGC	833	30	69	1	3	27	1	0.3

GGAGGAATACCAGTGGCGAAGGCG	CTCCCCACGCTTTCGCGTCTCAG	69	68	82	17	17	1	0	0.01
CGAAGGCGCCCTTCTGGAC	TTGCTCCCCACGCTTTCGC	56	65	84	4	51	1	0	0.01
AGTGGCGAAGGCGCCCTTCTG	CACGCTTTCGCGTCTCAGCGTC	53	64	80	3	21	1	0	0.01

Table B.2: Bistro-Primer results for *Syntrophobacter* for 16S rRNA target region. The forward and reverse primers are in 5'-3' direction. The maximum number of targets is 87 and the maximum number of non-targets is 198.

Forward	Reverse	Product size	Forward targets	Reverse targets	Forward non targets	Reverse non targets	Paired targets	Paired non targets	Score
GTGTCGTGAGATGTTGGGTTAAG	GCCCAGRTCATAAAGGGCATGATG	152	91	87	141	0	85	0	0.91
AGCAGTGGGGAATATTGCGC	TCGCGTTGCATCGAATTAAACC	622	89	89	2	17	85	0	0.91
TTAGATACCCTGGTAGTCCAC	GGCGGGATACTTATTGCG	97	88	89	145	0	84	0	0.9
ACTGGGACTGAGACACGG	CATAAAGGGCATGATGATTTGACG	900	90	88	63	4	87	4	0.89
AGGAAYACCAGTGGCGAAGG	TAGCCCAGRTCATAAAGGGCATG	513	88	87	48	0	82	0	0.88
GGTGRACGGCCACACTGGG	TCTCACGACACGAGCTGAC	780	89	92	8	155	88	8	0.86
CATGTGGTTTAATTCGATGCAAC	TGATTTGACGTCATCCCCAC	253	89	82	13	58	80	0	0.86
TAAAGAGCACGTAGGCGG	ACRGCGTGGACTACCAGGG	243	82	85	0	107	80	0	0.86
AGGCAGCAGTGGGGAATATTGCGC	GCTTTCGCACCTCAGCGTCAGGG	422	89	82	2	0	80	0	0.86
GGGCGAGCGTTGTCCGGAATTAC	YCGCCTTCGCCACTGGTRTTCCTC	193	87	88	4	35	83	3	0.86
AAACAGGATTAGATACCCTGG	ATAAAGGGCATGATGATTTGAC	434	89	88	126	4	84	4	0.86
ACTGGGCGTAAAGAGCAC	ACACGAGCTGACGACAAC	512	82	90	0	69	79	0	0.85
TAACTACGTGCCAGCAGC	TAAAGGGCATGATGATTTGACG	705	85	88	60	4	80	1	0.85
AAACCCTGACGCAGCGACGCC	GCACCTCAGCGTCAGGGTCAGTCC	384	87	79	2	0	78	0	0.84
TAGGGGGCGAGCGTTGTCC	AACCACATGCTCCACCGCTTG	419	86	89	8	84	82	4	0.84
GTCGCAAGGCTGAAACTC	CGATTACTAGCGATTCCGAC	457	85	86	21	13	78	0	0.84
GAATTACTGGGCGTAAAGAGCACG	TTGTAGCACGTGTGTAGCCCAG	681	82	88	0	31	78	0	0.84
GACTCCTACGGGAGGCAGC	TCATCCCCACCTTCCTCCBGCTTG	854	89	84	163	4	81	4	0.83
WTCTTGAGGGCAGGAGAGG	GAGCTGACGACAACCATGC	422	81	89	0	94	77	0	0.83
GACACGGCCCAGACTCCTACGG	GCCCAGTAATTCCGGACAACGCTC	244	85	86	63	4	80	4	0.82
ACGGCCCAGACTCCTACGGGAGG	ACGCTTTCGCACCTCAGCGTCAGG	444	86	83	78	0	76	0	0.82
AGGAGGAAYACCAGTGGC	TACCAGGGTATCTAATCCTG	94	83	90	7	162	81	6	0.81
TGAAATGCGTAGAAATCAG	BGCTTGTCACVGGCAGTC	481	81	86	0	0	74	0	0.8
TGACCCTGACGCTGAGGTGCG	TGCGACCGTACTCCCCAGGC	157	79	83	0	48	73	0	0.78
CTGACCCTGACGCTGAGGTGC	ACCGTACTCCCCAGGCGGG	154	79	83	0	4	73	0	0.78
AGCCCCGGCTAACTACGTG	GCTACCCACGCTTTCGCAC	281	85	80	15	0	73	0	0.78
GCCCAGACTCCTACGGGAGGC	GCCTACGTGCTCTTTACGCCCAG	255	86	79	78	0	72	0	0.77
CGCAATGGGGGAAACCCTGAC	CGCACCTCAGCGTCAGGG	396	77	84	21	0	72	0	0.77
TGGGCGTAAAGAGCACGTAGGC	TCGCACCTCAGCGTCAGGG	202	79	84	0	0	71	0	0.76
TGCGCAATGGGGGAAACCC	CTGGCACGTAGTTAGCCGGG	154	77	86	22	14	71	0	0.76
TACTAGGTGTRGGAGGTATCGACC	CAATCCGAACTGAGAATGGC	485	88	76	0	4	71	0	0.76
TTGCGCAATGGGGGAAACCC	CAACGCTCGCCCCCTACG	187	77	86	22	9	71	0	0.76
GTTGTCCGGAATTACTGGGC	TTCGCGATTACTAGCGATTCC	807	88	73	14	8	70	0	0.75
CGCCGCGTGAGCGAWGAAG	GCTACCCACGCTTTCGCAC	382	81	80	18	0	70	0	0.75
TGCGTAGAAATCAGGAGG	AGTCCCATTAGAGTGCTCAKC	462	83	77	0	0	69	0	0.74

TGCGCAATGGGGGAAACCC	GCTGCTGGCACGTAGTTAGCC	158	77	84	22	60	69	0	0.74
GGTTTAATTCGATGCAAC	RSTCACYGGCTTCGGGTG	491	89	70	19	0	68	0	0.73
AGGAAYACCAGTGGCGAAGGCG	TTCGCGATTACTAGCGATTCCGAC	645	88	72	48	1	68	0	0.73
ACGCCGCGTGAGCGAWGAAGG	CTGGCACGTAGTTAGCCGGGG	125	81	86	14	14	74	6	0.73
ACCCTGACGCAGCGACGC	GCGTTAGCTGCGGCACRGAAGGG	487	89	71	7	2	70	2	0.73
ATGGGGGAAACCCTGACGCAGC	TCGCACCTCAGCGTCAGGGTC	393	77	79	21	0	68	0	0.73
TGCGCAATGGGGGAAACCC	GGCTGCTGGCACGTAGTTAGC	159	77	83	22	60	68	0	0.73
GATGTTGGGTTAAGTCCCGC	TTCGCGATTACTAGCGATTCC	277	90	73	139	8	72	5	0.72
CTTTATGAYCTGGGCTACAC	GTTRRSTCACYGGCTTCGGGTG	240	86	69	0	0	65	0	0.7
TAATGGGACTGCCBGTGACAAG	GTACCATCCATTGTAGCACG	100	83	72	0	4	62	0	0.67
GVAAGCCCCGGCTAACTACGTG	ATGATGATTTGACGTCATCCCCAC	709	76	81	14	33	65	3	0.67
AACGCAATAAGTATCCCGCCTGG	GGCTTCGGGTGTTGCCRRCTTTCG	572	88	65	0	3	62	0	0.67
AACTACGTGCCAGCAGCCG	TAGCAAGGGTTGCGCTCGTTG	611	84	66	87	0	62	0	0.67
GAAAGCGTGGGTAGCAAACAGG	GCCRRCTTTCGTGGTGTGACGGG	659	81	71	19	11	63	2	0.66
TGGGTTAAGTCCCGCAACGAGCG	ACAAGGCCCGGGAACGCATTCAC	308	90	63	151	0	61	0	0.66
CGCGTCTGATTAGCTAGTTGGTG	CTGACGACAACCATGCACCAC	833	65	89	0	60	61	0	0.66
TATTGCGCAATGGGGGAAACCC	CAAGGGTTGCGCTCGTTGC	752	77	70	8	77	65	5	0.65
ACCCTGGTAGTCCACGCYG	GAACTGAGAATGGCTTTTTGAG	509	85	61	105	0	60	0	0.65
VAAGCCCCGGCTAACTACGTGCC	AGCTGACGACAACCATGCACCACC	573	76	88	15	60	72	12	0.65
TCTGGACTGACCCTGACGC	GCCRRCTTTCGTGGTGTGACGGG	686	73	71	0	11	59	0	0.63
GAGACACGGCCCAGACTCC	GGCTTCGGGTGTTGCCRRCTTTCG	1113	85	65	69	3	60	2	0.62
TKTCTGGACTGACCCTGACGCTG	CTTATTGCGTTAGCTGCGGCACRG	140	73	72	0	0	58	0	0.62
AGGAAYACCAGTGGCGAAGGC	CGAATTAAACCACATGCTCCACC	252	88	88	48	79	84	27	0.61
RKWTCTTGAGGGCAGGAGAGG	GATTCGCGATTACTAGCGATTC	712	73	73	0	8	57	0	0.61
CTAACTACGTGCCAGCAG	GAACTGAGAATGGCTTTTTGAG	796	85	61	60	0	56	0	0.6
CAAGGCRACGATCAGTAGCCGG	CACYGGCTTCGGGTGTTGCC	1169	73	66	6	2	53	0	0.57
TGGAAACGRCTGCTAATACCGC	CGACRTGCTGATTCGCGATTAC	1213	75	56	0	5	51	0	0.55
TCTGATTAGCTAGTTGGTGGGG	TGGCACGTAGTTAGCCGG	282	56	86	1	33	51	0	0.55
GTTAAGTCCCGCAACGAGCG	CGCATTCACCGCGACRTGCTG	291	90	51	152	0	51	0	0.55
AATGGGACTGCCBGTGACAAGC	ATTCACCGCGACRTGCTGATTC	226	83	55	0	5	49	0	0.53
CTAACGCAATAAGTATCCCGCC	ATTCACCGCGACRTGCTGATTC	513	79	55	0	5	48	0	0.52
CTTATRGATGRGCCCGCGTCTG	CBGCTTGTCACVGGCAGTCCC	953	51	86	0	0	48	0	0.52
GTGGGGATGACGTCAAATCATC	RTGCTGATTCGCGATTACTAGC	183	82	57	33	5	47	0	0.51
GGGTAAMGGCCTACCAAGGC	AGCAACTAATAGCAAGGGTTG	871	58	65	17	0	47	0	0.51
AGCTAGTTGGTGGGGTAAMGGC	TCGCGTTGCATCGAATTAAAC	729	47	89	11	17	45	0	0.48
TTAGCTAGTTGGTGGGGTAAMGGC	CCCCTACGTCTTACCGCG	300	45	91	9	10	45	0	0.48
ACGTCAAATCATCATGCC	TTTTTGAGATTCGCTCCAC	99	89	45	56	0	44	0	0.47
GTGTRGGAGGTATCGACCCCTTC	TGCTCTCTGTACCATCCATTGTAG	427	88	48	0	0	44	0	0.47
AAMGGCCTACCAAGGCRACGATC	TAAAGGGCATGATGATTTGACG	951	44	88	1	4	43	0	0.46

GGAGGAAYACCAGTGGCGAAGG	AACATCTCACGACACGAGCTG	376	88	90	48	155	85	43	0.45
AGGAAYACCAGTGGCGAAGG	AACATCTCACGACACGAGC	374	88	90	48	155	85	43	0.45
TGTTGGGTTAAGTCCCGC	TGCAGACTVCAATCCGAACTGAG	236	90	78	152	38	75	34	0.44
GHGGGGGATAACAGTTGGAAAC	CTCTTTACGCCCAGTAATTC	438	40	84	0	3	40	0	0.43
GCAACCCTTGCTATTAGTTGC	GCTGCTCTCTGTACCATCC	152	65	49	0	0	40	0	0.43
TAACAGTTGGAAACGRCTGC	TCTGTACCATCCATTGTAG	1104	50	72	2	0	39	0	0.42
CACTGGGACTGAGACACGGC	CGGTGTGTACAAGGCCCGG	1087	90	77	63	124	75	37	0.41
AGCTAGTTGGTGGGGTAAMGG	TAGCAACTAATAGCAAGGGTTG	884	47	65	11	0	38	0	0.41
ACGAGCGCAACCCTTGCTATTAG	ACCGCGACRTGCTGATTCGCG	271	65	55	0	5	37	0	0.4
ACGGTACCWTAMSAGVAAGCCCC	GTTTACRGCGTGGACTAC	336	37	86	0	126	36	0	0.39
AGCTAGTTGGTGGGGTAAMGG	TTCGCGATTACTAGCGATTC	1111	47	73	11	8	36	0	0.39
TAACTACGTGCCAGCAGCCG	TGACGACAACCATGCACCACC	559	84	89	59	60	80	45	0.38
AGHGGGGGATAACAGTTGGAAACG	GTCACVGGCAGTCCCATTAGAGTG	1029	39	84	0	0	35	0	0.38
GATTAGATACCCTGGTAGTCCACG	CGAGCTGACGACAACCATG	286	88	89	123	94	84	50	0.37
ATTAGCTAGTTGGTGGGGTAAMG	ACCATCCATTGTAGCACG	1004	45	72	9	4	34	0	0.37
GGGTAAMGGCCTACCAAGGCRACG	TTGCCRRCTTTCGTGGTGTGACG	1168	43	70	13	3	31	0	0.33
GTACCWTAMSAGVAAGCCCCGG	AGGGTCAGTCCAGAMAGYC	267	38	73	0	0	30	0	0.32
CGGGGTTGCAYWTGAAACTG	CGGGTGTTGCCRRCTTTCG	806	37	69	0	3	30	0	0.32
TAACTACGTGCCAGCAGCCG	GCTTTTTGAGATTCGCTCCACCTC	783	84	33	59	0	30	0	0.32
GAGACACGGCCCAGACTC	TCGTTGCGGGACTTAACCC	781	85	90	69	153	83	54	0.31
CACGGCCCAGACTCCTACGG	GCTCGTTGCGGGACTTAACCC	779	85	90	64	153	83	54	0.31
GCVGGAGGAAGGTGGGGATGACG	YTCGCTGCTCTCTGTACCATCC	92	85	30	9	0	28	0	0.3
CCCAGACTCCTACGGGAGGCAG	CACGACACGAGCTGACGACAACC	746	86	89	78	69	82	55	0.29
ACAGTTGGAAACGRCTGC	GCAACCCCGBRGTTGAGCYSC	485	72	30	3	0	27	0	0.29
CTAGGTGTRGGAGGTATCGACC	CGGYYTCGCTGCTCTCTGTAC	440	88	25	0	0	25	0	0.27
CTGACCCTGACGCTGAGGTG	YYTCGCTGCTCTCTGTACCATCC	519	79	30	0	0	25	0	0.27
TGAGCGAWGAAGGCCTTAGGGTYG	TTTGAGATTCGCTCCACCTCRCG	882	67	32	0	0	22	0	0.24
GGATAACAGTTGGAAACG	TCTCTGTACCATCCATTG	1109	40	56	2	0	21	0	0.23
GCRACGATCAGTAGCCGG	RTGCAACCCCGBRGTTGAGCYSC	361	75	27	6	0	21	0	0.23
CYVCGGGGTTGCAYWTGAAACTGG	CGCGATTACTAGCGATTC	732	21	89	0	61	20	0	0.22
CRCTTATRGATGRGCCCGCGTCTG	TGCAACCCCGBRGTTGAGCYSC	417	51	30	0	0	17	0	0.18
VYRHNTTAGTGGCGGACGGGTGAG	TTGCCRRCTTTCGTGGTGTGACG	1335	27	70	0	3	15	0	0.16

Table B.3: Bistro-Primer results for 16S rRNA target in the genus *Syntrophomonas*. The forward and reverse primers are in 5'-3' direction. The maximum number of targets is 93 and the maximum number of non-targets is 198.

Forward	Reverse	Product size	Forward targets	Reverse targets	Forward non targets	Reverse non targets	Paired targets	Paired non targets	Score
AGAAGGTTTTCGGATCGTAAAG	TCCATATATCTACGCATTTCACC	272	377	389	0	6	368	0	0.92
GTTTTCGGATCGTAAAGCTCTG	GTTTCAACCTTGCGGTCG	456	381	378	0	19	365	0	0.91
GAAACTCAAAGGAATTGACG	CCTTCCTCCGGTTTATTAC	257	388	375	377	3	367	3	0.91
AGTGAAGAAGGTTTTCGGATC	ATAAGGGGCATGATGATTTGAC	750	376	390	6	39	370	6	0.91
GAAGGTTTTCGGATCGTAAAG	AGTTTCAACCTTGCGGTC	461	377	378	9	19	363	0	0.91
TGGAAACGATAGCTAATACCGC	GCAGTCTCGCTAGAGTGC	929	368	383	0	10	357	0	0.89
CCAGACTCCTACGGGAGGCAGC	GCTGGCACGTAGTTAGCCGTCC	161	367	380	347	32	356	0	0.89
TGCATGGTTGTCGTCAGCTCG	CCTCCGGTTTATTACCGGCAGTC	120	367	371	92	3	359	3	0.89
GTGGAGCATGTGGTTTAATTCG	GTCATAAGGGGCATGATGATTTG	256	366	386	266	6	360	5	0.89
CTGAGACACGGCCCAGACTC	CCCAGGTCATAAGGGGCATGATG	843	368	370	153	5	356	2	0.89
TTGGAAACGATAGCTAATACCGC	GCTCGTTGCGGGACTTAAC	883	366	382	0	443	354	0	0.89
ACGGGAGGCAGCAGTAGGG	ACGTGTGTAGCCCAGGTCATAAGG	831	373	370	65	5	358	5	0.88
GCGTTGTCCGGATTTATTG	AACATCTCACGACACGAG	516	365	389	33	389	358	5	0.88
CACGGCCCAGACTCCTAC	GGGGCATGATGATTTGACGTC	825	370	392	157	40	368	15	0.88
GGCGAAAGCGGCTCTCTGG	ATTCACCGCGGCGTGCTG	624	384	368	5	6	357	5	0.88
CGACCGCAAGGTTGAAACTC	CGTGCTGATCCGCGATTAC	453	378	377	19	6	357	5	0.88
GGTTTTCGGATCGTAAAGCTC	CCCAATAAATCCGGACAACG	125	381	361	0	32	352	0	0.88
GTAGATATATGGAGGAACACC	CCAGGGTATCTAATCCTGTTYG	101	389	364	6	355	357	6	0.88
TGGAAACGATAGCTAATACCG	ACGATCCGAAAACCTTCTTC	255	368	380	3	6	351	0	0.88
AGCAACGCCGCGTGAGTGAAG	CTCACGACACGAGCTGACGACAAC	638	376	374	12	87	360	10	0.88
TGCCGGTAATAAACCGGAGG	GATCCGCGATTACTAGCGATTC	201	371	369	3	117	351	3	0.87
CATGTGGTTTAATTCGAAGC	TTATTACCGGCAGTCTCG	203	366	367	98	3	351	3	0.87
ATTGGAAACGATAGCTAATACCGC	CACGAGCTGACGACAACC	851	363	368	0	94	348	0	0.87
GACTCCTACGGGAGGCAGCAGTAG	TTCACCGCGGCGTGCTGATC	981	371	365	63	6	346	3	0.86
CTCCTACGGGAGGCAGCAGTAGGG	TCACCGCGGCGTGCTGATCC	978	371	366	65	6	347	5	0.86
TTGTCCGGATTTATTGGGC	TCAATTCCTTTGAGTTTCAACC	363	361	376	5	57	346	5	0.85
AGGAACACCGGTGGCGAAAG	TGAGTTTCAACCTTGCGGTCGTAC	193	363	376	3	18	342	2	0.85
CTACACACGTGCTACAATGGYTGG	TTGCAGCCTACAATCCGAAC	97	364	353	8	14	340	0	0.85
AACGATGAGTGCTAGGTGTTRG	ACCARCCATTGTAGCACGTG	404	369	366	0	8	339	0	0.85
TCCATGTGTAGCGGTGAAATGCG	CGGTGTGTACAAGGCCCGG	695	389	349	28	272	343	4	0.85
GTAATACGTAGGTCCCGAGCG	CAGTCTCGCTAGAGTGCCC	601	352	382	28	10	339	0	0.85
CGAACGGGTGAGTAACGC	CTTCGAATTAAACCACATGCTCC	806	354	367	36	98	344	6	0.85
ACGGGAGGCAGCAGTAGG	ATGATTTGACGTCATCCCCACC	801	374	388	65	38	370	32	0.85
CGTAAACGATGAGTGCTAGGTG	CACGAGCTGACGACAACC	242	349	368	0	94	337	0	0.84
GACTGAGACACGGCCCAGACTCC	AGAGCCGCTTTCGCCACCG	390	368	366	152	2	337	0	0.84

GACCGAGCAACGCCGCGTGAG	TATTCACCGCGGCGTGCTGATCCG	932	371	364	8	6	342	5	0.84
TGAGTAACGCGTAGGTAAC	ACTCGTTGTACCARCCATTGTAG	1068	365	364	0	0	335	0	0.84
AAGAAGGTTTTCGGATCGTAAAGC	CTAGCACTCATCGTTTACGGC	390	376	347	0	11	335	0	0.84
GACGGGGGCCCGCACAAG	GGCGTGCTGATCCGCGATTACTAG	426	359	365	377	6	340	6	0.84
TCCTACGGGAGGCAGCAGTAGGG	GCGGTCGTACTCCCCAGGCG	522	372	372	65	18	351	18	0.83
TGAGAGGGTGATCGGCCAC	CCTCCGGTTTATTACCGGCAG	825	348	371	88	3	335	3	0.83
ACCTTACCAGGTCTTGACATC	GACTTCATGTAGGCGAGTTG	339	377	341	1	7	332	0	0.83
ACTGAGACACGGCCCAGAC	TCGCCACCGGTGTTCCTC	379	368	364	153	35	335	3	0.83
TGACGGGGGCCCGCACAAG	ATTCACCGCGGCGTGCTGATCC	436	359	364	353	6	336	6	0.83
CACGGCCCAGACTCCTACGGG	CGCTTTCGCCACCGGTGTTCC	377	368	361	157	3	331	1	0.83
CGAGCGTTGTCCGGATTTATTGG	GGCGTGGACTACCAGGGTATC	266	348	362	2	190	332	2	0.83
CTGGGGAGTACGACCGCAAGGTTG	GCTGACGACAACCATGCACCACC	174	365	368	18	55	346	17	0.82
GGACGGCTAACTACGTGCCAG	ATGTAGGCGAGTTGCAGCCTAC	800	380	343	32	14	331	2	0.82
TAGATATATGGAGGAACACCGG	ACTACCAGGGTATCTAATCCTG	104	363	365	5	356	334	5	0.82
GTGCCAGCAGCCGCGGTAATAC	GACGACAACCATGCACCACCTGTC	529	386	365	401	29	357	29	0.82
CCGGATTTATTGGGCGTAAAGC	TGATGATTTGACGTCATCCCCAC	620	335	385	6	39	332	5	0.82
TGGAGCATGTGGTTTAATTCG	ATAAGGGGCATGATGATTTGAC	252	366	390	266	39	364	37	0.82
GGGACGGCTAACTACGTGCC	TCCGACTTCATGTAGGCGAGTTG	810	378	341	31	7	328	2	0.82
ACACCGGTGGCGAAAGCG	CCTGGTAAGGTTCTTCGCGTTGC	264	364	353	8	32	328	2	0.82
ACTCCGCCTGGGGAGTACGACCG	AGGCCCGGGAACGTATTCACCGC	495	370	349	18	141	330	4	0.82
TATTGGGCGTAAAGCGAGC	CATATATCTACGCATTTCACCGC	148	335	389	5	5	329	4	0.81
ATATATGGAGGAACACCGG	AGGGTATCTAATCCTGTTYGCTC	95	364	340	5	254	328	5	0.81
GTGAGTAACGCGTAGGTAAC	TTACAAACTCTCGTGGTGTG	1239	364	348	0	27	322	0	0.81
TAAAGCGAGCGCAGGCGG	AGAGCCGCTTTCGCCACC	167	329	366	3	2	321	0	0.8
RAACAGGATTAGATACCCTGGTAG	TGATGATTTGACGTCATCCCC	396	364	385	354	39	358	38	0.8
ACGTCAAATCATCATGCCCC	ACTCTCGTGGTGTGACGG	230	392	349	40	74	346	27	0.8
TAACTACGTGCCAGCAGCCG	TGACGACAACCATGCACCACC	537	379	368	223	55	356	37	0.8
TAACTGACGCTGAGGCTCGAAAGC	ATCTCACGACACGAGCTGACGAC	315	336	377	3	412	321	3	0.8
ACGCCGCGTGAGTGAAGAAG	TTTCGAGCCTCAGCGTCAGTTAC	343	376	336	11	3	321	3	0.8
TTCCATGTGTAGCGGTGAAATGCG	GCACTCATCGTTTACGGCGTGG	150	389	349	28	34	344	27	0.79
CTCGAAAGCGTGGGGAGCRAACAG	CTTCCTCCGGTTTATTACCGGCAG	391	333	371	5	3	319	3	0.79
CGGATTTATTGGGCGTAAAGCG	AAGACCTGGTAAGGTTCTTCGC	427	336	363	6	5	318	5	0.78
TGCAGAAGGGGAGAGTGGAATTC	CCGCGATTACTAGCGATTCCG	671	334	371	0	42	313	0	0.78
CCAGACTCCTACGGGAGGC	ARCCATTGTAGCACGTGTGTAGCC	853	368	364	404	36	338	25	0.78
AGAAGGGGAGAGTGGAATTC	YAACACCTAGCACTCATCG	176	337	370	0	0	310	0	0.78
AGTAGGGAATCTTCGGCAATG	ACTTCGGGTGTTACAAACTCTC	1023	374	317	9	4	310	3	0.77
GCGAACGGGTGAGTAACGC	TAGCGATTCCGACTTCATGTAGGC	1167	351	336	36	7	311	4	0.77
GGCTCTCTGGTCTGTAACTGAC	ATGATTTGACGTCATCCCCACC	441	306	388	2	38	299	2	0.74
CTGCGTTGTATTAGCTAGTTG	GAAGATTCCCTACTGCTGC	136	301	372	0	33	297	0	0.74

AGTGAAGAAGGTTTTCGGATCG	TGCTTAATGCGTTAGCTSCGGC	432	376	304	6	0	296	0	0.74
CCGSAGCTAACGCATTAAGC	AGTTGCAGCCTACAATCC	446	308	353	0	14	294	0	0.74
TGCCGSAGCTAACGCATTAAGC	GATGTCAAGACCTGGTAAGGTTC	145	307	377	0	0	291	0	0.73
CTCTCTGGTCTGTAACTGACG	CAGTCTCGCTAGAGTGCC	397	300	383	2	10	289	0	0.72
GATTAGATACCCTGGTAGTCCACG	CGAGCTGACGACAACCATG	265	365	367	254	94	352	65	0.72
GCGTTGTCCGGATTTATTG	GTCAGTTACAGACCAGAGAG	206	365	308	33	2	287	0	0.72
CAGTAGGGAATCTTCGGCAATGG	CAGCGTCAGTTACAGACCAGAG	375	374	300	9	2	286	0	0.72
AATTCGAAGCAACGCGAAGAACC	CAATAGGGGTTGCGCTCGTTG	150	356	303	54	21	290	5	0.71
GTAAAGCTCTGTTGTWAGAGAAG	ACTACCAGGGTATCTAATCC	348	303	366	0	357	281	0	0.7
TGGTCTGTAACTGACGCTG	CCATTGTAGCACGTGTGTAG	474	300	389	18	307	293	18	0.69
AGTTGCGAACGGGTGAGTAACGC	AGTCCCAGTGTGGCCGATCACC	206	292	349	0	24	274	0	0.69
GAGTTGCGAACGGGTGAGTAACG	TCCCCACGCTTTCGAGCCTC	632	289	327	0	5	273	0	0.68
CACTGGGACTGAGACACGGC	CGGTGTGTACAAGGCCCGG	1030	384	349	110	272	338	67	0.68
TTGGAAACGATAGCTAATAC	CYCACCAACTAGCTAATACAAC	86	366	270	3	5	264	0	0.66
VVTAGCGGGGGATAACTATTGG	CGCCCAATAAATCCGGAC	387	279	361	0	5	262	0	0.66
CGCGTAGGTAACCTRCCTVVTAG	TAGCTAATACAACGCAGG	113	267	368	0	0	259	0	0.65
CTAGTTGGTGRGGTAAMGGC	TSCSYCCATTGCCGAAGATTC	135	269	351	73	17	257	0	0.64
AGCAGCCGCGGTAATACGTAGG	TGTGACGGGCGGTGTGTACAAG	861	384	340	154	310	331	74	0.64
TAGCTAGTTGGTGRGGTAAMG	GGTCAGRSTTSCSYCCATTGCCG	147	268	347	53	32	257	1	0.64
CCAGACTCCTACGGGAGG	KMGGGATGTCAAGACCTGGTAAG	631	368	267	404	1	252	1	0.63
GTTGCGAACGGGTGAGTAAC	TGCTTAATGCGTTAGCTSCGGC	720	292	304	0	0	244	0	0.61
GAGACACGGCCCAGACTC	TCGTTGCGGGACTTAACCC	731	368	382	153	444	358	114	0.61
CACGGCCCAGACTCCTACGG	GCTCGTTGCGGGACTTAACCC	729	370	382	157	443	359	115	0.61
ATTAGCTAGTTGGTGRGGTAAMGG	GTCAATTCCTTTGAGTTTC	644	268	388	53	377	266	46	0.55
AAGGCRACGATACATAGC	TGTACCARCCATTGTAGC	920	232	365	1	8	207	0	0.52
AMGGCTCACCAAGGCRACGATAC	GATTTGACGTCATCCCCAC	880	186	388	3	69	181	3	0.45
AMGGCTCACCAAGGCRACGATAC	GATTCCGACTTCATGTAGGC	1022	186	338	3	7	174	0	0.44
ACCAAGGCRACGATACATAGCCG	AGCGTCAGTTACAGACCAGAG	459	230	300	1	2	162	1	0.4
ACTGTHDAACTTGAGTGCAGAAG	AAGACCTGGTAAGGTTCTTC	342	182	363	0	5	161	0	0.4
GCAGGCGGTTWKRTAAGTCTGAAG	AATGCGTTAGCTSCGGCAC	281	180	308	0	1	132	0	0.33
CGCAGGCGGTTWKRTAAGTCTG	GTGCTTAATGCGTTAGCTSCG	288	180	305	0	11	130	0	0.33
AAACTCAAAGGAATTGACGGG	TGTAGCACGTGTGTAGCC	312	391	390	530	326	383	281	0.26

Table B.4: Bistro-Primer results for the genus *Streptococcus* with 16S rRNA target region. The forward and reverse primers are in 5'-3' direction. The maximum number of targets is 400 and the maximum number of non-targets is 602.

APPENDIX C
BISTRO-PRIMER DESIGN MODULE

```
#Bistro-Primer design module
#Author - Praful Aggarwal

#!/usr/bin/env python
################################################################################
from Bio.Seq import Seq
from Bio.SeqRecord import SeqRecord
from Bio import SeqIO
from Bio import AlignIO
from Bio.SeqUtils.MeltingTemp import *
from Bio.SeqUtils import *
from Bio.Alphabet import IUPAC
from numpy import*
from operator import itemgetter

################################################################################

print "\n"
print "################### Hello! Welcome to the Bistro-Primer Designer
    ####################"

print '\n'


## User defined parameters and target MSA file

filename = raw_input("Enter the name of your multiple alignment sequence file: "
    )
try:
    handle = open(filename, "r")
except IOError:
    print ' file does not exist'
    exit()
max_threshold = float(raw_input("Enter the Maximum Threshold value (between 50%
    and 100%): "))
GC_content = float(raw_input("Enter the minimum GC content : "))
min_length = int(raw_input("Enter the minimum Primer length: "))
max_length = int(raw_input("Enter the maximum Primer length: "))
melt_temp_min = float(raw_input("Enter the minimum melting temperature: "))
melt_temp_max = float(raw_input("Enter the maximum melting temperature: "))
melt_temp_diff = float(raw_input("Enter the difference in melting temperature of
     the Primers: "))
number_of_primer_pairs = int(raw_input("Enter the maximum number of Primer Pairs
     to be displayed: "))

a=[]
count=0
count_G=0
n=0
j=0
i=0
T=[]
G=[]
```

```
C=[]
A=[]
Gap=[]
consensus=[]

## dictionary used in detecting primer-dimers and self-complementarity

COMP_BASE = { "T": "A", "A": "T", "G": "C", "C": "G","Y": "R","R": "Y","W": "W",
    "S": "S","K": "M","M": "K","B": "V","V": "B","D": "H","H": "D","N": "N" }

for seq_record in SeqIO.parse(handle, "fasta"):
    a.append(seq_record.seq)
    count=count+1
    if(n <= len(a[count-1])):
       n = len(a[count-1])
strr =[]
k=0
for k in range(count):
    strr.append(a[k])

for j in range(n):
    countT=0;
    countG=0;
    countC=0;
    countA=0;
    countR=0;
    countM=0;
    countW=0;
    countY=0;
    countS=0;
    countK=0;
    countV=0;
    countD=0;
    countB=0;
    countH=0;
    countGap=0;
    for i in range(count):
        if strr[i][j] == 'T':
           countT = countT+1
        elif strr[i][j] == 'G':
           countG = countG+1
        elif strr[i][j] == 'C':
           countC = countC+1
        elif strr[i][j] == 'A':
           countA = countA+1
        elif strr[i][j] == '-':
           countGap = countGap+1
        else:
           pass

    T.append(countT)
    G.append(countG)
    C.append(countC)
    A.append(countA)
    Gap.append(countGap)

Mthreshold = max_threshold*float(count)/100

## create a gapped consensus sequence
```

```
for j in range(n):
    if count==T[j] or T[j]>=Mthreshold:
       consensus.append('T')
    elif count==A[j] or A[j]>=Mthreshold:
       consensus.append('A')
    elif count==G[j] or G[j]>=Mthreshold:
       consensus.append('G')
    elif count==C[j] or C[j]>=Mthreshold:
       consensus.append('C')
    elif count==A[j]+G[j] or A[j]+G[j]>=Mthreshold:
       consensus.append('R')
    elif count==A[j]+C[j] or A[j]+C[j]>=Mthreshold:
       consensus.append('M')
    elif count==A[j]+T[j] or A[j]+T[j]>=Mthreshold:
       consensus.append('W')
    elif count==C[j]+T[j] or C[j]+T[j]>=Mthreshold:
       consensus.append('Y')
    elif count==C[j]+G[j] or C[j]+G[j]>=Mthreshold:
       consensus.append('S')
    elif count==G[j]+T[j] or T[j]+G[j]>=Mthreshold:
       consensus.append('K')
    elif count==A[j]+G[j]+C[j] or A[j]+G[j]+C[j]>=Mthreshold:
       consensus.append('V')
    elif count==A[j]+G[j]+T[j] or A[j]+G[j]+T[j]>=Mthreshold:
       consensus.append('D')
    elif count==C[j]+G[j]+T[j] or C[j]+G[j]+T[j]>=Mthreshold:
       consensus.append('B')
    elif count==A[j]+C[j]+T[j] or A[j]+C[j]+T[j]>=Mthreshold:
       consensus.append('H')
    elif count==A[j]+T[j]+G[j]+C[j] or A[j]+G[j]+C[j]+T[j]>=Mthreshold:
       consensus.append('N')
    else:
       consensus.append('-')

gapped_consensus_list = ''.join(consensus)

ungapped_consensus = ''.join(gapped_consensus_list.split('-'))
ungapped_consensus_seq = Seq(ungapped_consensus, IUPAC.unambiguous_dna)
ungapped_consensus_rev_compl_bp = ungapped_consensus_seq.reverse_complement()
ungapped_consensus_rev_compl = str(ungapped_consensus_rev_compl_bp)

## open a file for writing the consensus sequence

handle_consensus = open("consensus.txt","w")
handle_consensus.write(ungapped_consensus)
handle_consensus.close()

CC=[]
CC_rev=[]
GG=[]
GG_rev=[]
E=[]
Rev=[]
GC=[]
GC_rev=[]
GC_good_oligos=[]
GC_bad_oligos=[]
GC_good_oligos_rev=[]
```

```
GC_bad_oligos_rev=[]
window=18

## function to identify primer-dimers
def Primer_dimer(F, Re):
     m = len(F); n = len(Re)
     D = [[0] * (n+1) for i in xrange(m+1)]
     dimer = set()
     p_dimer = 0
     for i in xrange(m):
         for j in xrange(n):
             if F[i] == COMP_BASE[Re[j]]:
                 v = D[i][j] + 1
                 D[i+1][j+1] = v
                 if v > p_dimer:
                     p_dimer = v
                     dimer = set()
                 if v == p_dimer:
                     if type(F)==list:
                         dimer.add( tuple(F[i-v+1:i+1]) )
                     else:
                         dimer.add( F[i-v+1:i+1] )
     return list(dimer)

## generating oligo-nucleotides from the ungapped consensus sequence

for k in range(min_length,max_length+1):
    for i in range(len(ungapped_consensus)-k+1):
        E.append(ungapped_consensus[i:i+k])
        Rev.append(ungapped_consensus_rev_compl[i:i+k])

GC_value=[]
GC_value_rev=[]
T_value=[]
T_value_rev=[]
T_m_value=[]
T_m_value_rev=[]

## calculate GC content and melting temperature for forward oligos and separate
    good oligos from bad oligos(based on the parameters)
for i in range(len(E)):
    CountC=0
    CountG=0
    for j in range(len(E[i])):
        if E[i][j]=='G':
           CountG=CountG+1
        elif E[i][j]=='C':
           CountC=CountC+1;
        else:
           pass
    CC.append(CountC)
    GG.append(CountG)
    GC.append((CC[i]+GG[i])*100/len(E[i]))
    T_value.append(Tm_staluc(E[i]))
    if GC[i]>=GC_content and T_value[i]>=melt_temp_min and T_value[i]<=
        melt_temp_max and len(Primer_dimer(E[i][0:len(E[i])/2],E[i][len(E[i])/2:
        len(E[i])]))<=1 and (E[i][len(E[i])-1]=='C' or E[i][len(E[i])-1]=='G'):
       GC_good_oligos.append(E[i])
       GC_value.append(GC[i])
```

```
            T_m_value.append(T_value[i])
        else:
            GC_bad_oligos.append(E[i])

    ## calculate GC content and melting temperature for reverse oligos and separate
        good oligos from bad oligos(based on the parameters)
    for i in range(len(Rev)):
        CountC=0
        CountG=0
        for j in range(len(Rev[i])):
            if Rev[i][j]=='G':
                CountG=CountG+1
            elif Rev[i][j]=='C':
                CountC=CountC+1;
            else:
                pass
        CC_rev.append(CountC)
        GG_rev.append(CountG)
        GC_rev.append((CC_rev[i]+GG_rev[i])*100/len(Rev[i]))
        T_value_rev.append(Tm_staluc(Rev[i]))
        if GC_rev[i]>=GC_content and T_value_rev[i]>=melt_temp_min and T_value_rev[i
            ]<=melt_temp_max and len(Primer_dimer(Rev[i][0:len(Rev[i])/2],Rev[i][len
            (Rev[i])/2:len(Rev[i])]))<=1 and (Rev[i][len(Rev[i])-1]=='C' or Rev[i][
            len(Rev[i])-1]=='G'):
            GC_good_oligos_rev.append(Rev[i])
            GC_value_rev.append(GC_rev[i])
            T_m_value_rev.append(T_value_rev[i])
        else:
            GC_bad_oligos_rev.append(Rev[i])

    ## generate a primer pair based on the difference in the melting temperature
    Primer_Pair=[]
    for i in range(len(GC_good_oligos)):
        for j in range(len(GC_good_oligos_rev)):
            melting_temp_diff = T_m_value[i] - T_m_value_rev[j]
            if abs(melting_temp_diff)<= melt_temp_diff and abs(melting_temp_diff)
                >0.5 and len(Primer_dimer(GC_good_oligos[i], GC_good_oligos_rev[j]))
                <=1:
                Primer_Pair.append([GC_good_oligos[i],GC_good_oligos_rev[j], abs(
                    melting_temp_diff)])
    count_pp=0

    dict_primer_pair = {}
    for i in Primer_Pair: dict_primer_pair[i[2]]=i


    Primer_Pair_sorted = sorted(dict_primer_pair.values(), key=itemgetter(2))
    primer_pair_sorted_forward=[]
    primer_pair_sorted_reverse=[]

    for i in range(len(Primer_Pair_sorted)):
        for j in range(0,1):
            primer_pair_sorted_forward.append(Primer_Pair_sorted[i][j])

    for i in range(len(Primer_Pair_sorted)):
        for j in range(1,2):
            primer_pair_sorted_reverse.append(Primer_Pair_sorted[i][j])

    ## create forwrad and reverse primer files
```

```
file_pair_f = file("forward_primer.txt","w")
file_pair_r = file("reverse_primer.txt","w")
for i in range(2*number_of_primer_pairs):
    primer_string_forward = str(primer_pair_sorted_forward[i])
    primer_string_reverse = str(primer_pair_sorted_reverse[i])
    file_pair_f.write(primer_string_forward+'\n')
    file_pair_r.write(primer_string_reverse+'\n')

file_pair_f.close()
file_pair_f.close()
handle.close()
print "\n"
print "Consensus sequence file created: consensus.txt"
print "Forward primer file created: forward_primer.txt"
print "Reverse primer file created: reverse_primer.txt"
```

APPENDIX D
BISTRO-PRIMER VALIDATION MODULE

```
# Bistro-Primer validation module
# Author - Praful Aggarwal

#!/usr/bin/env python
################################################################################
from Bio.Seq import Seq
from Bio.SeqRecord import SeqRecord
from Bio import SeqIO
from Bio import AlignIO
from Bio.SeqUtils.MeltingTemp import *
from Bio.SeqUtils import *
from Bio.Alphabet import IUPAC
from numpy import*
from operator import itemgetter
import re
from collections import defaultdict
import os

################################################################################

print "\n"
print "################### Hello! Welcome to the Bistro-Primer Validation Module
    ####################"

print '\n'

## Get the names of all the necessary user files

filename_con = raw_input("Enter the name of the consensus sequence file: ")
try:
    handle = open(filename_con, "r")
except IOError:
    print ' file does not exist'
    exit()

filename_forward = raw_input("Enter the name of your forward primer file: ")
try:
    handle = open(filename_forward, "r")
except IOError:
    print ' file does not exist'
    exit()

filename_reverse = raw_input("Enter the name of your reverse primer file: ")
try:
    handle = open(filename_reverse, "r")
except IOError:
    print ' file does not exist'
    exit()

filename_forward_good = raw_input("Enter the name of your target sequence file
    to be checked: ")
try:
    handle = open(filename_forward_good, "r")
```

```
except IOError:
    print ' file does not exist'
    exit()

filename_forward_bad = raw_input("Enter the name of your non-target sequence
    file to be checked: ")
try:
    handle = open(filename_forward_bad, "r")
except IOError:
    print ' file does not exist'
    exit()

def make_rc_record(record):
    """Returns a new SeqRecord with the reverse complement sequence."""
    return SeqRecord(seq = record.seq.reverse_complement(), \
                     id = "rc_" + record.id, \
                     description = "reverse complement")

## Open all the necessary user files

handle_consensus = open(filename_con, "r")
consensus = handle_consensus.read()

handle_forward = open(filename_forward, "r")
handle_reverse = open(filename_reverse, "r")
handle_forward_good = open(filename_forward_good, "r")
records_good = map(make_rc_record, SeqIO.parse(handle_forward_good, "fasta"))
handle_forward_good.close()
handle_forward_good_rd = file(filename_forward_good).readlines()
data_forward_good_rd = ''.join(handle_forward_good_rd[1:])
data_forward_good_rd = data_forward_good_rd.replace('\n','')

## create reverse target sequence files

handle_reverse_good = open("rev_comp_good", "w")
SeqIO.write(records_good, handle_reverse_good, "fasta")
handle_reverse_good.close()
handle_reverse_good_rd = file("rev_comp_good").readlines()
data_reverse_good = ''.join(handle_reverse_good_rd[1:])
data_reverse_good = data_reverse_good.replace('\n','')

handle_forward_bad = open(filename_forward_bad, "r")
records_bad = map(make_rc_record, SeqIO.parse(handle_forward_bad, "fasta"))
handle_forward_bad.close()
handle_forward_bad_rd = file(filename_forward_bad).readlines()
data_forward_bad_rd = ''.join(handle_forward_bad_rd[1:])
data_forward_bad_rd = data_forward_bad_rd.replace('\n','')

## create reverse non-target sequence file

handle_reverse_bad = open("rev_comp_bad", "w")
SeqIO.write(records_bad, handle_reverse_bad, "fasta")
handle_reverse_bad.close()
handle_reverse_bad_rd = file("rev_comp_bad").readlines()
data_reverse_bad = ''.join(handle_reverse_bad_rd[1:])
data_reverse_bad = data_reverse_bad.replace('\n','')

Count_forward_good=[]
Count_reverse_good=[]
```

```
Count_forward_bad=[]
Count_reverse_bad=[]
count_forward_good=[]
count_reverse_good=[]
count_forward_bad=[]
count_reverse_bad=[]

pair_id_seq_id = []
pair_id_seq_id_bad = []
syn_dict = SeqIO.index(filename_forward_good, "fasta")
syn_dict_bad = SeqIO.index(filename_forward_bad, "fasta")
seq_forw=[]
seq_forw_bad =[]
seq_rev_comp=[]
seq_rev_comp_bad=[]
key = syn_dict.keys()
key1 = syn_dict_bad.keys()
dict_id_seq = defaultdict(list)
dict_id_seq_bad = defaultdict(list)

target_length = float(len(syn_dict))

## Function that creates regular expressions for the degenerate sequences for
    searching

def degen_to_re(seq):
    reseq = seq.replace("R", "[AG]")
    reseq = reseq.replace("Y","[CTU]")
    reseq = reseq.replace("S","[GC]")
    reseq = reseq.replace("W","[ATU]")
    reseq = reseq.replace("K","[GTU]")
    reseq = reseq.replace("M","[AC]")
    reseq = reseq.replace("B","[CGTU]")
    reseq = reseq.replace("D","[AGTU]")
    reseq = reseq.replace("H","[ACTU]")
    reseq = reseq.replace("V","[GCA]")
    reseq = reseq.replace("N","[GCTAU]")
    return reseq
matches_tf =[]
matches_tr =[]
matches_ntf =[]
matches_ntr =[]
set = ['R','Y','S','W','K','M','B','D','H','V','N']

file_forward = open(filename_forward, 'r')
file_reverse = open(filename_reverse, 'r')
forward=[]
reverse=[]
while 1:
    line1 = file_forward.readline()
    if not line1: break
    line2 = file_reverse.readline()
    if not line2: break
    forward.append(line1.rstrip())
    reverse.append(line2.rstrip())

degen_forward = []
degen_reverse = []
reverse_reverse = []
```

```
reverse_seq = []
product = []
degen_reverse_rc = []
degen_reverse_seq =[]


for i in range(len(forward)):
    degen_forward.append(degen_to_re(forward[i]))

for i in range(len(reverse)):
    degen_reverse.append(degen_to_re(reverse[i]))
    reverse_seq.append(Seq(reverse[i]))
    reverse_reverse.append(str(reverse_seq[i].reverse_complement()))

## Get the product size for the primer pairs

for i in range(len(degen_forward)):
    product.append(consensus[consensus.find(forward[i]):consensus.find(
        reverse_reverse[i])+len(reverse_reverse[i])])

product_len = []
for i in range(len(product)):
    product_len.append(len(product[i]))

## Search for forward primer targets

for i in range(len(forward)):
    matches_tf.append(re.findall(degen_forward[i].lower(),data_forward_good_rd))
    Count_forward_good.append([forward[i],len(matches_tf[i])])
    count_forward_good.append(len(matches_tf[i]))

## Search for forward primer non-targets

for i in range(len(forward)):
    matches_ntf.append(re.findall(degen_forward[i].lower(),data_forward_bad_rd))
    Count_forward_bad.append([forward[i],len(matches_ntf[i])])
    count_forward_bad.append(len(matches_ntf[i]))

## Search for reverse primer targets

for i in range(len(reverse)):
    matches_tr.append(re.findall(degen_reverse[i].lower(),data_reverse_good))
    Count_reverse_good.append([reverse[i],len(matches_tr[i])])
    count_reverse_good.append(len(matches_tr[i]))

## Search for reverse primer non-targets

for i in range(len(reverse)):
    matches_ntr.append(re.findall(degen_reverse[i].lower(),data_reverse_bad))
    Count_reverse_bad.append([reverse[i],len(matches_ntr[i])])
    count_reverse_bad.append(len(matches_ntr[i]))

## Search for paired target hits

for i in range(len(syn_dict)):
     seq_record = syn_dict[key[i]]
     seq_forw.append(str(seq_record.seq))
     seq_rev_comp.append(str(seq_record.seq.reverse_complement()))
     for j in range(len(forward)):
```

```
            forw_hit = re.findall(degen_forward[j].lower(), seq_forw[i])
            rev_hit = re.findall(degen_reverse[j].lower(), seq_rev_comp[i])
            if forw_hit and rev_hit:
               for l in range(len(set)):
                   if set[l] not in forward[j] and set[l] not in reverse[j] and set
                       [l+1] not in forward[j] and set[l+1] not in reverse[j] and
                       set[l+2] not in forward[j] and set[l+2] not in reverse[j]
                       and set[l+3] not in forward[j] and set[l+3] not in reverse[j
                       ] and set[l+4] not in forward[j] and set[l+4] not in reverse
                       [j] and set[l+5] not in forward[j] and set[l+5] not in
                       reverse[j] and set[l+6] not in forward[j] and set[l+6] not
                       in reverse[j] and set[l+7] not in forward[j] and set[l+7]
                       not in reverse[j] and set[l+8] not in forward[j] and set[l
                       +8] not in reverse[j] and set[l+9] not in forward[j] and set
                       [l+9] not in reverse[j] and set[l+10] not in forward[j] and
                       set[l+10] not in reverse[j]:
                      product = seq_forw[i][seq_forw[i].find(str(forward[j]).lower
                          ()):seq_forw[i].find(str(reverse_reverse[j]).lower())+len
                          (reverse_reverse[j])]
                      break
                   else:
                      product = consensus[consensus.find(forward[j]):consensus.find
                          (reverse_reverse[j])+len(reverse_reverse[j])]
                      break
               if len(product) > 2*(len(forward[j])+len(reverse[j])):
                  pair_id = str(j)+'_'+str(j)
                  pair_id_seq_id.append((pair_id, key[i]))


## Search for paired non-target hits

for i in range(len(syn_dict_bad)):
     seq_record_bad = syn_dict_bad[key1[i]]
     seq_forw_bad.append(str(seq_record_bad.seq))
     seq_rev_comp_bad.append(str(seq_record_bad.seq.reverse_complement()))
     for j in range(len(forward)):
         #for k in range(len(rev)):
          forw_hit_bad = re.findall(degen_forward[j].lower(), seq_forw_bad[i])
          rev_hit_bad = re.findall(degen_reverse[j].lower(), seq_rev_comp_bad[i
              ])
          if forw_hit_bad and rev_hit_bad:
             for l in range(len(set)):
                 if set[l] not in forward[j] and set[l] not in reverse[j] and
                     set[l+1] not in forward[j] and set[l+1] not in reverse[j]
                     and set[l+2] not in forward[j] and set[l+2] not in reverse[
                     j] and set[l+3] not in forward[j] and set[l+3] not in
                     reverse[j] and set[l+4] not in forward[j] and set[l+4] not
                     in reverse[j] and set[l+5] not in forward[j] and set[l+5]
                     not in reverse[j] and set[l+6] not in forward[j] and set[l
                     +6] not in reverse[j] and set[l+7] not in forward[j] and
                     set[l+7] not in reverse[j] and set[l+8] not in forward[j]
                     and set[l+8] not in reverse[j] and set[l+9] not in forward[
                     j] and set[l+9] not in reverse[j] and set[l+10] not in
                     forward[j] and set[l+10] not in reverse[j]:
                    product_bad = seq_forw_bad[i][seq_forw_bad[i].find(str(
                        forward[j]).lower()):seq_forw_bad[i].find(str(
                        reverse_reverse[j]).lower())+len(reverse_reverse[j])]
                    break
                 else:
```

```
                    product_bad = consensus[consensus.find(forward[j]):consensus
                        .find(reverse_reverse[j])+len(reverse_reverse[j])]
                    break
            if len(product_bad) > 2*(len(forward[j])+len(reverse[j])) :
                pair_id_bad = str(j)+'_'+str(j)
                pair_id_seq_id_bad.append((pair_id_bad, key1[i]))

for k,v in pair_id_seq_id:
    dict_id_seq[k].append(v)
key2 = dict_id_seq.keys()
sort_dict_id_key =[]
sort_dict_id_val=[]
sort_dict_id_bad_key=[]
sort_dict_id_bad_val=[]

for k,v in pair_id_seq_id_bad:
    dict_id_seq_bad[k].append(v)
key3 = dict_id_seq_bad.keys()
for k in sorted(dict_id_seq):
    sort_dict_id_key.append(k)
    sort_dict_id_val.append(dict_id_seq[k])

for k in sorted(dict_id_seq_bad):
    sort_dict_id_bad_key.append(k)
    sort_dict_id_bad_val.append(dict_id_seq_bad[k])


forw_index=[]
rev_index=[]

for i in range(len(sort_dict_id_key)):
    f = sort_dict_id_key[i].split('_')
    for j in range(len(f)):
        forw_index.append(f[j])
        rev_index.append(f[j+1])
        break


file_forward.close()
file_reverse.close()

##  Write the output file

handle_dict_pid_excel = file("Validated_Result.xls","w")
handle_dict_pid_excel.write('Forward'+'\t'+'Reverse'+'\t'+'Product Size'+'\t'+'
    Forward_Target Hits'+'\t'+'Reverse_Target Hits'+'\t'+'Forward_Non_Target
    Hits'+'\t'+'Reverse_Non_Target Hits'+'\t'+'Paired_Target_Hits'+'\t'+'
    Paired_Non_Target Hits'+'\t'+'Score'+'\n')
handle_dict_pid_excel.close()

handle_dict_pid_excel = file("Validated_Result.xls","a")
for k in range(len(dict_id_seq)):
    for l in range(len(dict_id_seq_bad)):
        if str(sort_dict_id_key[k]) == str(sort_dict_id_bad_key[l]):
            ind = int(forw_index[k])
            score = (len(sort_dict_id_val[k]) - len(sort_dict_id_bad_val[l]))/
                target_length
            handle_dict_pid_excel.write(forward[ind]+'\t'+reverse[ind]+'\t'+str(
                product_len[ind])+'\t'+str(count_forward_good[ind])+'\t'+str(
```

```
                count_reverse_good[ind])+'\t'+str(count_forward_bad[ind])+'\t'+
                str(count_reverse_bad[ind])+'\t'+str(len(sort_dict_id_val[k]))+'\
                t'+str(len(sort_dict_id_bad_val[l]))+'\t'+str(score)+'\n')
            break
    else:
        ind = int(forw_index[k])
        score = len(sort_dict_id_val[k])/target_length
        handle_dict_pid_excel.write(forward[ind]+'\t'+reverse[ind]+'\t'+str(
            product_len[ind])+'\t'+str(count_forward_good[ind])+'\t'+str(
            count_reverse_good[ind])+'\t'+str(count_forward_bad[ind])+'\t'+str(
            count_reverse_bad[ind])+'\t'+str(len(sort_dict_id_val[k]))+'\t'+'0'+'
            \t'+str(score)+'\n')


handle_dict_pid_excel.close()

print "\n"
print "Output file : Validated_Result.xls"

## Remove the reverse_complement_files

remove_file1 = "rev_comp_good"
remove_file2 = "rev_comp_bad"
os.remove(remove_file1)
os.remove(remove_file2)
```