Final Project Narrative Report
Rohan Ray
AS.410.712 Advanced Practical Computer Concepts for Bioinformatics

Background

Molecular cloning is a foundational technique in molecular biology for replicating sections of DNA, but in order to do so, a compatible restriction enzyme must be selected. This process is usually done manually and can be fragmented and error-prone. Tools like NEB Cutter and REBASE are the current standard for accomplishing this, but integrating all restriction enzyme databases into a single, interactive tool is still a hole in this process that remains to be filled. My project aimed to build this tool: a Smart Restriction Enzyme Selector (SRES) that streamlines this step. By allowing users to input a DNA sequence and select cloning preferences, the tool provides a one-click solution to grabbing compatible enzyme pairs that are suitable for their ligation and cutting preferences, ensuring that they not only accurately cut the DNA at the ends, but also work with the user's buffer requirements.

Building the Database

The core of this project relied on building an SQL version of the REBASE enzyme database. The Restriction Enzyme Database is one of the most holistic collections of information about restriction enzymes, including methylases, the microorganisms from which they have been isolated, recognition sequences, cleavage sites, methylation specificity, and the commercial availability of the enzymes (*https://rebase.neb.com/rebase/rebase.html*). This data was pulled from the REBASE website in a .txt file with a custom tagged format (<1> to <8>), with each of these tags having different information, like enzyme name, recognition sequence, commercial sources, and reference IDs. To parse this file, I wrote a Python script to extract these fields and used an SQL command to insert them into a database on the class SQL server.

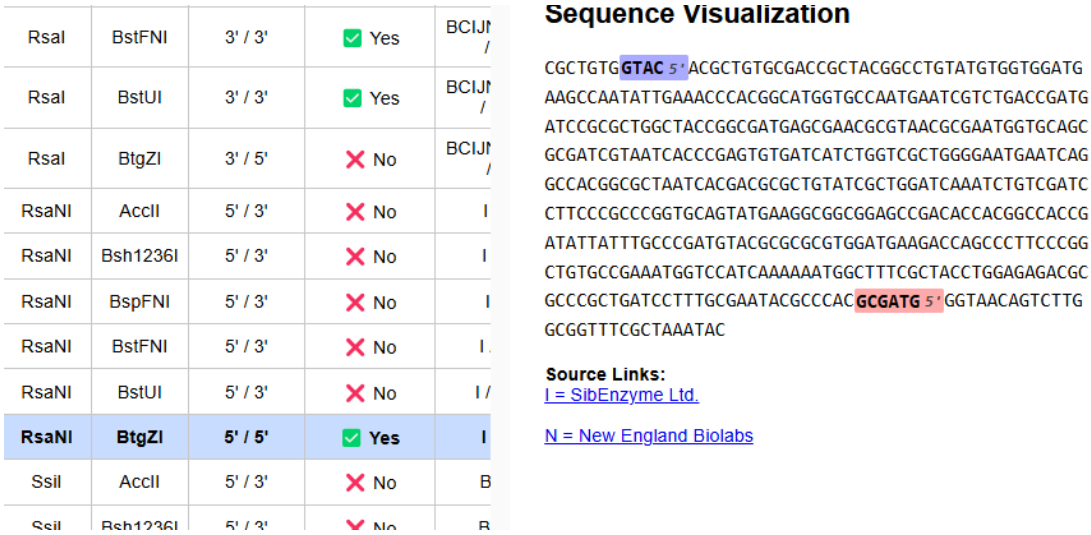
```
<1>M.Aap248I                                          enzyme name
<2>MboI                                               prototype enzyme
<3>Aggregatibacter aphrophilus FDAARGOS_248           microorganism
<4>B. Goldberg                                        source
<5>GATC                                               recognition sequence
<6>2(6)                                               methylation site
<7>                                                   commercial availability
<8>418                                                references

<1>M.Aap5906II
<2>MboI
<3>Aggregatibacter aphrophilus ATCC 33389 NCTC5906
<4>P. Informatics
<5>GATC
<6>2(6)
<7>
<8>1488
```

Fig 1. Example formatting of allenz.txt file sourced from REBASE

One of the largest complications during this process was handling the cut sites on the recognition sequences. For a bit of scientific context on why this was an issue, cleavage sites can be blunt (cutting both DNA strands at the same position) or result in overhangs (one strand is cut slightly upstream or downstream of the other, producing sticky 5’ or 3’ ends). In REBASE notation, cleavage positions are indicated either using a caret symbol (^) to show the exact cut location, or with offset notation like (5/10), where the numbers indicate how many bases away from the recognition site the top and bottom strands are cut. These notations have implications on how effectively the restriction enzymes bind to the DNA and if the cloning process is compatible as a whole. Back to the parsing script, two separate cases were built in to handle each notation, using regular expressions to parse out the offset notation and a simple search to handle the caret notations. Building this information into the SQL database that our tool will pull from reduces the amount of computational strain that would otherwise be present on the html/cgi side of the project, enabling faster queries. I also opted to build two separate datasets, one for the enzymes only with commercial sources listed, essentially a way to buy them, (smaller dataset), and the other had the complete dataset for more comprehensive searches in case users want to go out and source the enzymes themselves.

To verify the accuracy of the enzyme matching logic, several enzyme pairs returned by the tool were manually compared against known ligation-compatible pairs from NEB's online resources. For example, EcoRI and HindIII, both known to generate distinct 5′ overhangs, were correctly flagged as incompatible. Likewise, RsaNI and BtgZI—both generating compatible 5′ overhangs—were returned as valid ligation candidates, validating our overhang-matching logic.



RsaI	BstFNI	3' / 3'	✅ Yes	BCIJ… /
RsaI	BstUI	3' / 3'	✅ Yes	BCIJ… /
RsaI	BtgZI	3' / 5'	❌ No	BCIJ… /
RsaNI	AccII	5' / 3'	❌ No	I
RsaNI	Bsh1236I	5' / 3'	❌ No	I
RsaNI	BspFNI	5' / 3'	❌ No	I
RsaNI	BstFNI	5' / 3'	❌ No	I
RsaNI	BstUI	5' / 3'	❌ No	I /
RsaNI	**BtgZI**	**5' / 5'**	✅ **Yes**	**I**
SsiI	AccII	5' / 3'	❌ No	B
SsiI	Bsh1236I	5' / 3'	❌ No	B

Sequence Visualization

CGCTGTG**GTAC** 5' ACGCTGTGCGACCGCTACGGCCTGTATGTGGTGGATG
AAGCCAATATTGAAACCCACGGCATGGTGCCAATGAATCGTCTGACCGATG
ATCCGCGCTGGCTACCGGCGATGAGCGAACGCGTAACGCGAATGGTGCAGC
GCGATCGTAATCACCCGAGTGTGATCATCTGGTCGCTGGGGAATGAATCAG
GCCACGGCGCTAATCACGACGCGCTGTATCGCTGGATCAAATCTGTCGATC
CTTCCCGCCCGGTGCAGTATGAAGGCGGCGGAGCCGACACCACGGCCACCG
ATATTATTTGCCCGATGTACGCGCGCGTGGATGAAGACCAGCCCTTCCCGG
CTGTGCCGAAATGGTCCATCAAAAAATGGCTTTCGCTACCTGGAGAGACGC
GCCCGCTGATCCTTTGCGAATACGCCCAC**GCGATG** 5' GGTAACAGTCTTG
GCGGTTTCGCTAAATAC

Source Links:
I = SibEnzyme Ltd.

N = New England Biolabs

Fig 2. Validation testing with Escherichia coli partial lacZ gene, strain CECT 428 DNA sequence.

Backend and Database Logic

The Python and CGI script serves as the backend, using mysql.connector to query the SQL database populated with the parsed REBASE data. Upon receiving user input of a DNA sequence and their associated preferences, it first strips whitespace and checks for valid nucleotide characters. The script then retrieves a list of candidate enzymes from the SQL database and scans the DNA sequence

for acceptable restriction sites located within a defined number of base pairs from the sequence ends. Enzymes with internal cut sites get excluded through this process. The filter_enzymes function applies the core logic of scanning the input sequences to find enzymes that cut only near the 5' and 3' ends within the user-specified window size (such as 50 bp). Recognition sites are matched using regular expressions, and if a recognition site appears internally in the sequence (outside the window sizes at the ends), that enzyme gets discarded.

Once valid 5' and 3' enzymes are identified, the script compares every possible pair to evaluate ligation compatibility. This decision is based on whether their overhang types match (if both are blunt, or both produce 5' overhangs of the same type). Incompatible overhangs are still visible to users, but it is clearly indicated that they don't work.

Frontend and Visualization

The web frontend, written in HTML and Javascript, presents users with the clean input form (index.html), supporting customizable parameters like preferred end type (sticky, blunt, any), enzyme source scope (commerical only or all), and the maximum window size (distance from sequence ends within which the enzymes can cut). These results are rendered in results.html using a Jinja2 template, showing enzyme pairs in a table along with overhang compatibility, commercial sources, and if buffer conditions are met. When users click on a row, JavaScript dynamically highlights the recognition sites of the selected enzymes in the input sequence, showing hte overhang annotations.

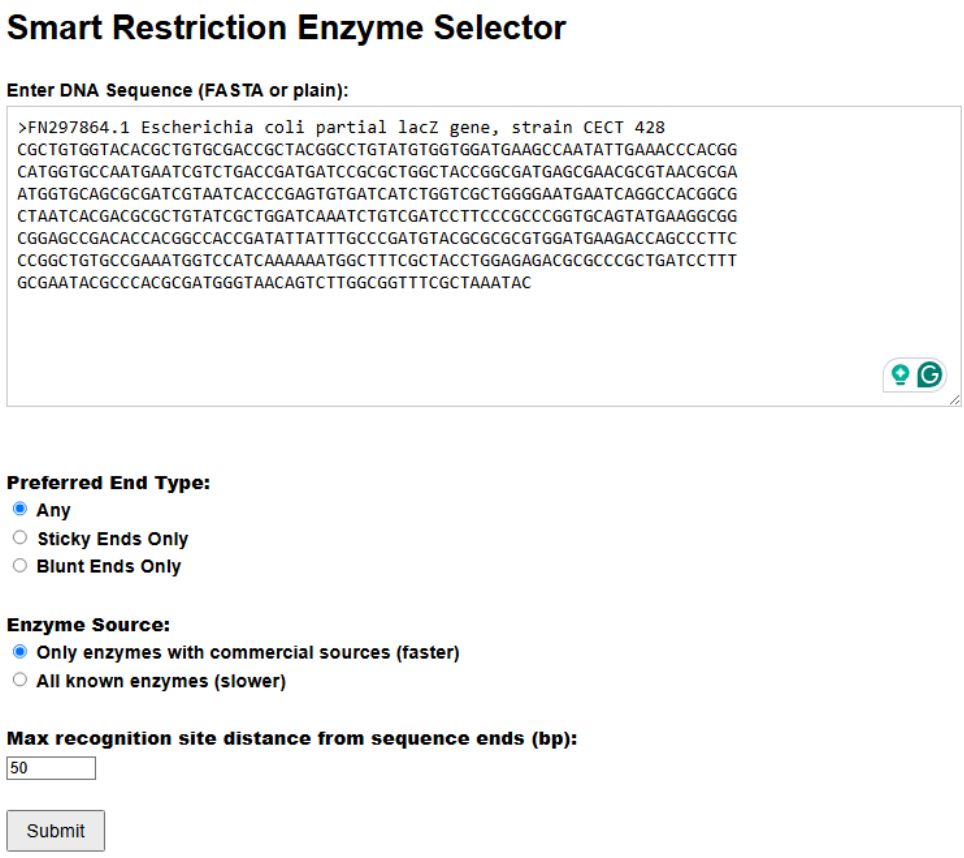
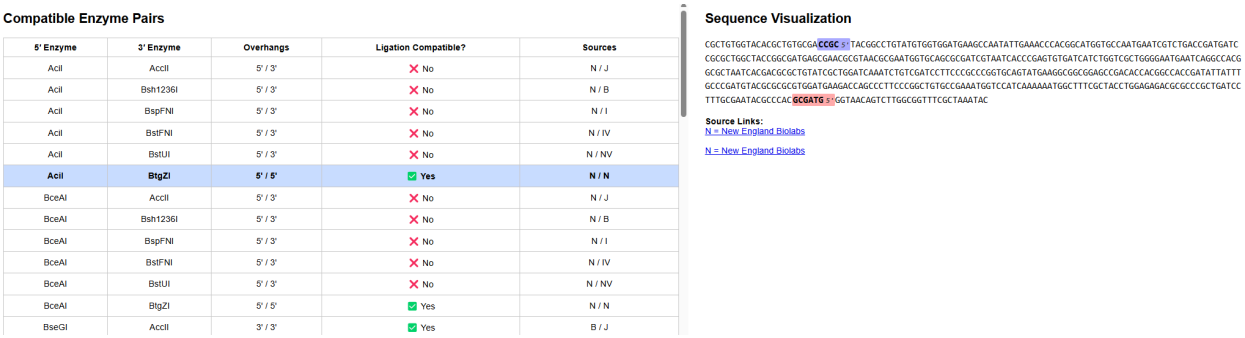


Fig 3. Index.html page where users can input their desired DNA sequence, and any specific preferences for their search.



Compatible Enzyme Pairs

5' Enzyme	3' Enzyme	Overhangs	Ligation Compatible?	Sources
AclI	AccII	5' / 3'	✗ No	N / J
AclI	Bsh1236I	5' / 3'	✗ No	N / B
AclI	BspFNI	5' / 3'	✗ No	N / I
AclI	BstFNI	5' / 3'	✗ No	N / IV
AclI	BstUI	5' / 3'	✗ No	N / NV
AclI	**BtgZI**	**5' / 5'**	✓ **Yes**	**N / N**
BceAI	AccII	5' / 3'	✗ No	N / J
BceAI	Bsh1236I	5' / 3'	✗ No	N / B
BceAI	BspFNI	5' / 3'	✗ No	N / I
BceAI	BstFNI	5' / 3'	✗ No	N / IV
BceAI	BstUI	5' / 3'	✗ No	N / NV
BceAI	BtgZI	5' / 5'	✓ Yes	N / N
BseGI	AccII	3' / 3'	✓ Yes	B / J

Sequence Visualization

CGCTGTGGTACACGCTGTGCGACCGC 5'TACGGCCTGTATGTGGTGGATGAAGCCAATATTGAAACCCACGGCATGGTGCCAATGAATCGTCTGACCGATGATC
CGCGCTGGCTACCGGCGATGAGCGAACGCGTAACGCGAATGGTGCAGCGCGATCGTAATCACCCGAGTGTGATCATCTGGTCGCTGGGGAATGAATCAGGCCACG
GCGCTAATCACGACGCGCTGTATCGCTGGATCAAATCTGTCGATCCTTCCCGCCCGGTGCAGTATGAAGGCGGCGGAGCCGACACCACGGCCACCGATATTATTT
GCCCGATGTACGCGCGCGTGGATGAAGACCAGCCCTTCCCGGCTGTGCCGAAATGGTCCATCAAAAAATGGCTTTCGCTACCTGGAGAGACGCGCCCGCTGATCC
TTTGCGAATACGCCCAC GCGATG 5'GGTAACAGTCTTGGCGGTTTCGCTAAATAC

Source Links:
N = New England Biolabs

N = New England Biolabs

Fig 4. Results page (left column is the compatible enzyme pairs table, right is the sequence visualization with highlighted recognition sequences) [Legend and tooltips are also shown on the left side, not seen in this image]

Challenges and Resolutions

Parsing REBASE's text format presented some initial difficulties, especially in consistently extracting recognition sequences and understanding the overhang information. Another challenge was performance, as naively checking all possible enzyme pairs quickly stalled out the html file many times during testing. To resolve this, I opted for pre-filtering the enzyme candidates and limiting the pairwise checks to those only near sequence ends. Frontend bugs were also an issue, especially with the JavaScript code involving the visualization aspect of the recognition sequence. Originally, the script would store the exact index positions of the recognition sequence from the original DNA sequence as integers alongside each pair of enzymes, but in order to do this for hundreds of entries, the processing time took far too long and eventually became untenable. To fix this, I decided to dynamically search for the recognition sequence with regular expression filtering as an on-click feature whenever a row gets clicked. This ended up resolving the issue quite well, and also allowed me to experiment a bit with the aesthetics of the visualization aspect.

Future Work and Conclusions

With a bit more time, I'd definitely want to add a step-by-step guide to the cloning process to make this as much of a "one-stop shop" as possible for my users. This would involve not only giving them links to where they can purchase the restriction enzymes, but also the full experimental protocol for actually conducting the cloning process with their newly selected enzymes. On a smaller scale however, I think batch processing multiple sequences at once could be helpful, and adding more of the metadata for each enzyme from other databases could be useful as well.

In all, I felt that this project was a great foundation builder that allowed me to integrate the various components of creating a bioinformatics tool that we have slowly learned all semester, from HTML templates to CGI scripts and SQL databases. I'd be curious to see if researchers actually find my tool effective and what improvements they would suggest or what features they'd like to see added.

https://github.com/roray02/rray16_712final