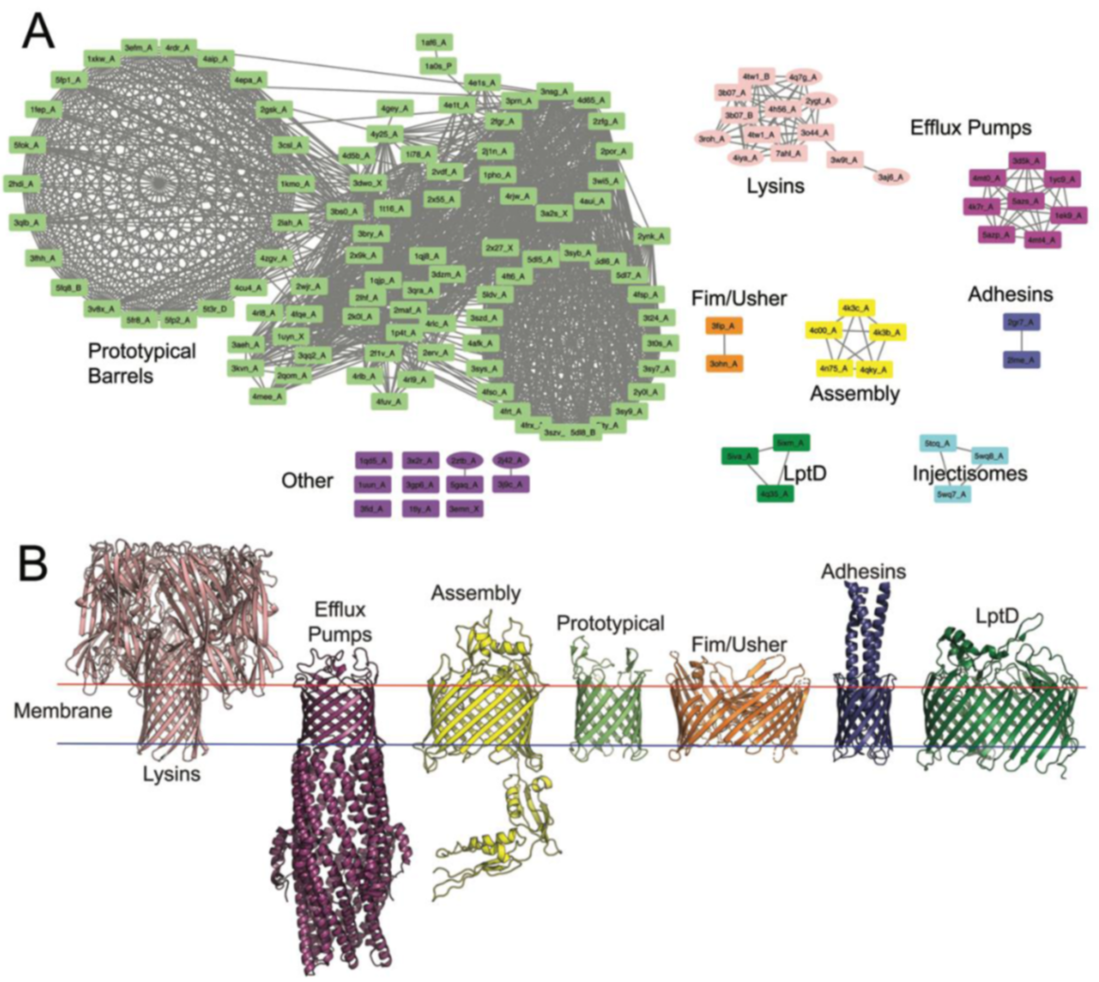
Untitled

Jaden Anderson

May 1st, 2018

Introduction

Gram negative bacteria have a unique arrangement of membranes that help them persist in their enviornment: an outer and inner membrane. On these membranes exist a variety of proteins and structures that have a range of functions that the bacteria utilize[@Remmert2010]. Outer membrane proteins are used to interact with their extracellular enviornment that have a diverse array of functions like import and export of substances, cell adhesion, proteolysis, membrane protein insertion, and more. Even though proteins on the outer membrane have a variety of functions, surprisingly the secondary structure of these proteins (i.e. the structure of what the proteins look like) are very similar[@Franklin, @article]. While these secondary structures are highly similar, their sequence composition (or similarity) is highly variable which makes studying their evolution very difficult. An evolutionary analysis based on HMM profiles was carried out and distinguished roughly 8 unique groups of outer membrane beta barrels (OMBBs) [@Franklin].



The folding patterns of the OMBBs is controlled by the assembly group of proteins which consist of a small group of proteins as seen from Figure 1. The most notable assembly complex in gram negative bacteria is the Beta barrel assembly complex or BamA. It is responsible for folding the majority of OMBBs into the outer membrane however its evolutionary relationship with the other OMBBs its responsible for is not well understood. Given it's important role in the folding process of OMBBs it is surprising that BamA or the assembly proteins have little to no connectedness through HMM profiles which can be used as a measure of evolutionary relationships. In this study, I will perform a sequence based analysis of BamA and the current 130 structurally characterized OMBBs. My hypothesis is that BamA sequences in different species will be highly connected or overlap with critically important beta barrels from that same species of BamA, which might give insight onto the folding preferences of the beta barrel assembly complex.

Methods

Basic Local Alignment Search Tool or BLAST

BLASTing protein sequences is a fairly standard practive when trying to determine relationships between different proteins. The BLASTing algorithm that I used is PSIBlast which is based off the NCBI blastp algorithm. Normal BLAST (or blastp) is a sequence similarity search method which identifies like protein sequences based off given thresholds like homology or sequence similiarty. PSIBLAST is an iterative tool that builds protein sequence profiles from blastp results in order to refine search space and produce higher scoring alignments. I PSIblasted BamA along with all 130 structurally characterized OMBBs in order to generate multiple sequence alignments. In order to make sure I was pulling similar sequences, I used a false positive rate or evalue of 10^-50.

Determining Overlapping Species

Once I've obtained the PSIBLASTed sequences I will determine the number of overlapping sequences of BamA with a client protein (one that BamA is responsible for folding). To do this, I will exploit the format of FASTA files and use regular expression to grab unique species identifiers. Once I have grabbed those species from BamA, I will compare that list to each of the species that resulted from the PSIBLAST alignments for each client protein. This process can be found in the python chunk get_overlapping_sp of this RMD. In order for this script to be sucessful, I must use the reformat_MSA python chunk to reformat the alignments generated from the PSIBLASTed results. This is done to make the parsing quicker.

Testing Formats

In order to ensure that the data that I've grabbed from the NCBI webserver from PSIBLAST is in the proper formats, I will be loading the Rpackage that I wrote for the last project in order to use the format and protein unit tests on my sequences. I will not need to use the same_size unit test, as this analysis does not care about the size of the proteins in the multiple sequence alignment.

```python
import re
import os
def reformat_dir(filename):
    with open(filename, "r") as f, open(filename +"_reformated.txt", "w+") as w:
        seq = ""
        begin_line = ""
        count = 0
        seq_counts = []
        for line in f:
            if ">" in line:
               begin_line = line
               w.write(begin_line)
               w.write(seq + "\n")
               if(len(seq) != 0):
                   seq_counts.append(len(seq))
                   seq = ""
            else:
                seq += line.strip()
    return()
```