Currently, there are three main pieces with the goal of assembling full (or nearly full) viruses from viruses found during virome analysis.

These were designed by Cale as an Excel tool and a C++ package- now executable.

Beckmann- Go back and tell the problem

Problem: When Trinity, Velvet or other assembly tools assemble the short reads into contigs, we never get full viruses out of them. It is always produced as a series of contigs that when examined closely, have significant overlap.

Currently- researchers need to pull all contigs with the same annotation out of the dataset and then assemble their viruses by hand using the overlap as a guide.

Once assembled, this new genome is then used against the small reads to validate the data and determine the mutations and SNPs present in the sequence- percentage change.

Why does the assembly program fail? Currently unknown, but in all the times I have been doing this, some self assembly has always occurred when the full or nearly full contig is desired.

Then, validation against the SNPs to ensure that the assembly was correct and valid- in addition to determining the sequence changes that are present in the reads that represent other viral populations.

Cale’s programs- assemble based on annotation into a complete virus that can then be validated, eliminating the need to do a self-assembly and skipping that step.

Notes: A dataset to test the programs more thoroughly needs to be constructed to determine the problem and flag data strings that cause the problem- so that it can be seen how it functions

Currently, the assembly only works if there are no mismatches. Do we want to account for mismatches in the datasets?

Comment: Most of the time after Assembly, there are very few mismatches- the assembly is taking this into consideration somehow, however, it is constructing separate contigs for the same virus. Unclear why that is happening. Until we know why that happens, it may not useful for us to take into account mismatches until later, when we need to know that information based on the quasi-species problem.

Program is currently in C++, it may be more useful in Python to be incorporated into a database of bioinformatics tools available.

Only a few hundred lines, so it could be converted without too many problems, but it should be done know if we want to add any functionality to it.

Action Steps-

1. Aasma gives James data to look over- KM creates folder for everyone
2. James converts to python- with Cale
3. Set up a test dataset
4. Set up a meeting for the first week of May to go over progress

Target Journal: Journal of Virological Methods for now

Paper drafted with test cases by the end of the Summer so it can be worked on for submission.

KM- Work on introduction, add in the problem and why this is necessary for our datasets.