Benchmarking viral genome assembly tools

Type of project: Bachelor End Project, Master End Project, Literature Review

Student background: CS, Nanobiology, LST

Daily supervisor: Jasmijn Baaijens, Pattern Recognition and Bioinformatics (j.a.baaijens@tudelft.nl)

Background
When a virus enters the human body and infects a host cell, it will start to make many copies of its genome. However, viral genome replication is highly error prone, so many of these copies will be slightly different from the original genome. These genomic changes due to replication errors are referred to as *mutations*, and the ensemble of closely-related virus strains within the host organism is called a *viral quasispecies*. Identifying each of the virus strains present in the patient is important when designing a suitable treatment plan.

Viral particles can be extracted from the patient (for example using a nose swab or a blood sample) and genome sequencing then enables reading small fragments (*sequencing reads*) of these viral genomes. Then the computational task is to reconstruct the viral quasispecies from the sequencing reads, a process known as *viral quasispecies assembly*, but this is extremely challenging: it is as if a collection of different editions of the same book has been put through a shredder and you need to reconstruct each of the individual editions of the book.

In this project we will evaluate existing approaches to viral quasispecies assembly by building representative benchmarking data. What we learn from this will then help us in further method development and data analysis.

Project components
Note that these are suggestions, the scope and focus of the project will be adapted to suit your expertise and interests, as well as your program requirements.

- Design and build simulated benchmarking datasets that resemble real sequencing data from viral infections
- Analyze real sequencing data to be used for benchmarking
- Use these benchmarking datasets to explore the limits of existing approaches
- Link method performance to algorithmic design; what do we learn from this for future approaches?