GC in the frequency domain

Martin Zackrisson

May 26, 2014

1 Abstract

The primary data obtained when sequencing a genome consists of millions of short strings. They are comprised of the letters A, T, C, and G, of which A-T form one pair in DNA double helix and C-G the other. Due to difference in the number of hydrogen bonds in these pairs, they are not equally accessible to the sequencing machines. Further, the relative proportion of A-T versus C-G works as an imperfect signature for each species as well as for the nature of the specific sequence – gene or intergenic region. Today there exist various methods for analyzing C-G frequency (sum of occurancies of either C or G divided by the total) in general or for each position of the primary data strings. This is done to diagnose the quality of the data. However, to the best of my knowledge there are no analyses of the C-G occurancies done in the frequency domain.

The proposed project will read and translate these strings into `numpy` arrays, typically (but not exclusively) representing C or G as 1 and A or T as 0. Here `threading` will be used along with dynamic management of data storage in `numpy` arrays. Potentially *c* or *c++* code will be written to increase the spead of the numeric encoding of the sequencing data. The encoded data will be Fourier transformed and the resulting spectra clustered using relevant methods in `scipy`. A flexible report pipe-line with several complex heatmaps and dendrograms will be produced using `matplotlib`.

The method is expected to yield complimentary views of the data and assist the bioinformatician in deciding how to progress with the sequencing data. However, as the usefulness is uncertain at outset, the main focus of the project is to produce fully documented high quality code in compliance with `numpy`'s documentation standards (`sphinx` used). Further, `unittest` will be employed to secure the functionality of vital parts of the code. This poses a particular challenge as input typically is several gigabyte of data and to some extent need those volumes to be tested correctly. Therefore, setting up of tests will involve generation of very large temporary files with `numpy` and `scipy`. Finally, the codebase will comply with python standards and use `distutils` to ship and *git* as version handling system.

2 Project Plan

- Put project plan and management in useful format (1h)
- Analysis of interfaces needed for the end-user and data structures suitable for internal representation (4h)
- Design of suitable unittests (6h)
- Implementation, input parsing and encoding (4h)
- Implementation, data analysis and reports (10h)
- Verification of code functionality and cleaning up (4h)
- Run on existing genome data (2h)
- Compiling `sphinx` documentation (4h)
- Writing required files LICENSE, INSTALL, CHANGELOG (4h)
- Writing Report (8h)