Requirements for the ribosome modeling software framework

Cedric Landerer

March 4, 2015

1 Input

1. Sequence (coding sequence only)
2. initial expression level per gene
3. observed expression level (optional)

2 Output

1. relative mutation bias per codon
2. relative selection bias per codon
3. estimated expression level per gene

3 General software description and requirements

The software framework has to be able to perform a MCMC based algorithm on a model of choice using coding sequences and expression level as input. The framework will perform a random walk on the defined parameter space by likelihood comparison.

The MCMC class has to provide functions to compare likelihood values and decide the acceptance of a proposed parameter value. The Parameter class is responsible for tracing the values and storing accepted values. The Model class performs the likelihood calculation.

An R interface will be provided to access the framework written in C++ (using the C++ 11 standard).

3.1 Requirements

The software has to be modular, with a consistent interface (specified in section XX) to allow for the exchange of modules. The software must provide the ability to perform heavy tasks in parallel (openmp vs. thread).

4 Targeted User base

We target users with a background in computational biology, capable of programming basic R scripts. A strong computational environment is assumed to perform tasks with a high computational load.

5 Modularity

1. Classes do only perform operations on their own data.
2. Objects do not modify other objects and instances.
3. Objects should not expose member variables but instead use getter and setter functions

6 Interface

6.1 R to C++

The R interface to access will be provided by usage of the R package Rcpp. To expose whole classes it is required that RCPP_MODULES are provided for each C++ class exposed to R

6.2 Between C++ classes

6.2.1 Genome class

The genome class will hold a list of genes.
Public functions provided by the Genome class:

1. **void** readFasta(**char*** filename);
2. **void** writeFasta(**char*** filename);
3. **void** addGene(**const** Gene& gene);
4. Gene& getGene(**int** index);
5. Gene& getGene(std::string id);
6. **int** getGenomeSize();

6.2.2 Model class

The Model class describes the model used in the MCMC. This class will calculate the likelihood for a given set of parameters.
Public functions provided by the Model class:

1. **double** calculateLogLiklihoodPerGene(Gene& gene, **int** geneIndex, ROCParameter& parameter, **bool** proposed);

7 Code

1. line break is required before every new code-block {
2. code blocks will begin in a new line
3. one whitespace is required between operator and variable
4. variables marking a pointer have to start with a lowercase p
5. large data structures should be passed by reference

6. code must not be written twice -¿ convert to function for reuse
7. C++ style array use is preferred over C style array use
8. Each class is written in one header file (.h for function and class definitions) and one code file (.cpp for function implementation).

7.1 Example Code Block

The following code block illustrates the C++ style use of arrays and the specified line break before {.

```
for(unsigned i = start; i < traceLength; i++)
{
  posteriorMean += expressionTrace[i][geneIndex];
}
```

The following code example shows a function definition where the variables *genome* and *parameter* are passed as references

```
double calculateLogLiklihoodPerGene(Gene& gene, int geneIndex,
  ROCParameter& parameter, bool proposed);
```