Simple Sequence Physical Complexity Calculator

Robert Bakarić
rbakaric@exaltum.eu
07.07.2016
PhyComplex-0.01

Abstract

This program implements a measure for the complexity of sequences ("strings"), rooted in automata theory called physical complexity proposed by [1]. This complexity is estimated for an ensembles of sequences expressed as the difference between the maximal entropy of the ensemble and its actual entropy given the particular environment within which the sequence is to be interpreted. Thus the measure calculates the amount of information about the environment that is encoded in the sequence, and is conditional on the environment.

The measure should not be confused with the Kolmogorov complexity since physical complexity only deals either with the intrinsic regularity or irregularity of a sequence in this case. Program is a simple perl library utilizing the muscle software [2].

Contents

1 Installation

The simplest way to compile this program is to:

1. Unpack the PhyComplex package (`PhyComplex-XXX.tar.gz`):

   ```
   tar -xvzf PhyComplex-XXX.tar.gz
   ```

2. Change the current directory to `PhyComplex-XXX`:

   ```
   cd PhyComplex-XXX/
   ```

3. Build the program for your system :

   ```
   perl Makefile.PL
   ```

4. Compile the program:

   ```
   make
   ```

5. Test:

   ```
   make test
   ```

6. Install the program:

   ```
   (sudo) make install
   ```

2 Input files

PhyComplex takes a simple fasta file and computes the physical complexity of a query sequence defined through -q option. An example of the input file can be found in `./PhyComplex-xxx/demo` and it should look like this: `Test.fa`:

```
>CADANIAT00000001 cds:known chromosome:ASM1142v1:IV:1268:6235:1 ...
ATGCATTGCTCGTCTCTTGAATTCCTACTGGCGGACCGGTACGGCTCTGGCGATCCCTCA
TATTCCTACTGGCGGATAGGCACCGATTATCAGGCTCTTGGTAAGGCCATCCGCAATCAT
CTGGTTGAGCTTAGATTATTGCTTGATATTTTCAGAGGTATCCTGCAGAGCTTTGCTGGT
CAGATATATAATAACCCTGTTTGCAGCCTGAATATAGTCTGTATTTCTGGCTATTTTGCA
AGGCTGTTTAGTTTATCTTTGAAGTATTTAGCTGCCTCTCTGGCTGCTGGTATTGATAGG
TGTATTAAGCAAGGAATTAAGATACATGCCTTTGCCTTTGTAAACAGATACTATATATTT
AGTACTATATACTTCAATGGCTGTATAGGCCTGAGTAATTCAAGGAGATGTCAGCTTAGT
AATATCTATATCTCTACTGGTAGTATAGAATACCTTGACCTTAGTGACTCGGCCAAGGCC
TGCGCTGTCCTGAAGGCGGTGAGCCACCTACAAGACTTCCTTGCAACAACAATCCTTCTT
TCTCATTTCTTCTTTAGCGATTCCTTCTTGGCTAAAGGTCTTGCAATATTTGATAGTGAT
CTATTTATCCCTGTTCAGACTAGTTGTAAAAGCTATATTAGTCAAGTATCTGTTTTTGTT
GGGTTTGCTTTGGCTATAAATATTAGCCAGGAGATTGCAGTGTTTAATTGCCAATTCTTC
AAAGCTTTGAAATCTGCCTCTTGA
>CADANIAT00000002 cds:known chromosome:ASM1142v1:IV:8987:9589:1 ...
ATGCTGCTATGGGGTTTGTCTCGGGCCAGCAGTCTACTGGCGCGGTGGGTTAACCACAAT
AGTTACCTTGAGAAAATAACTGCTGGCCCAGCAATCCTGGATCCCCCTTGGGGCAAAGCA
GGCAGGCAGCAGGTATCCCTGCATGGGTTGATTAAGGACTGCTGGAGACTTGGTGGTCAC
AGTAGCGGGGGCGCAGGTGTCGCTCCTCAATTGGCTTTACCAGCGATGGCAGTTGGGAAA
AGTGGTTGTGGAGCAGAAATATGGTCCAAAGACGCAAAGCTTCGAACTTACAAACTCTGC
TACTATGCAGCCATGTCTTTGATTAGGGAAGAAAAATGA
>CADANIAT00000003 cds:known chromosome:ASM1142v1:IV:12345:13757:-1 ...
ATGACTATCATGCATCATACGCCGCCAAAGGAATTGAATAATATAAACCAGATCCAAGAT
TGTAGGGGGCTCGAGGAGGTCTCTCAAGTTATGGGACCCTGGTTTCAGGTAAAAGGTGTT
AAGCATAGAACAAACCCCAAGCTACTGGGCCTCACTGATTTAAAACTGACCAAGTACTGC
TGCCTAACCGGGGCTGGGCTGGCAGGGAACTAG
...
```

3 Program options

I order to see program options type:

```
perl ./bin/PhyComplex.pl -h
```

Expected output:

```
  _____  _             _____                          _
 |  __ \| |           / ____|                        | |
 | |__) | |__   _   _| |     ___  _ __ ___  _ __ | | _____  __
 |  ___/| '_ \| | | | |    / _ \| '_ ` _ \| '_ \| |/ _ \ \/ /
 | |    | | | | |_| | |___| (_) | | | | | | |_) | |  __/>  <
 |_|    |_| |_|\__, |\_____\___/|_| |_| |_| .__/|_|\___/_/\_\
                __/ |                     | |
               |___/                      |_|
                                                        v0.01
Usage:

-i input fasta file
-h prints this
-q query sequence id [def: 0]
-v verbos
```

4 Functions and Modules

Complexity::Physical module:

`new` : Constructor. Creates a new Complexity::Physical object.

(Ex.:my $PSobject = Complexity::Physical→new();)

`PhyComplexCompute` : Where $array is an array reference to a 2D matrix consisting of the aligned strings. Alignment is required or be global

Example: $array

```
[['-','A','B','C','H','-','I','I','-'],
 ['-','A','C','C','S','-','-','I','-'],
 ['-','-','C','C','A','-','O','I','-']]
```

queryid refers to the index position of the alignment in @array. For example 0 refers to `['-','A','B','C','H','-','I','I','-']` setting it to be the string for which the complexity is to be calculated.

Function returns the complexity value for a given query sequence.

(Ex.: my $ps = $PSobject→PhyComplexCompute(queryid => $query, aligned => $array);)

`_ComputePosEntropy` : This is a private function that computes the conditional entropy for a given location in a query sequence. Function requires list of all characters occupying a given location and their occurrence counts. Moreover, the information about the number of characters present at a given site needs to be provided through alphabet option (key).

(Ex.:$H_cond = $self→_ComputePosEntropy(characters => %hash, alphabet => $postot);)

_Log : Private function designed for computing a $log_y(x)$. First argument is the value and the second is the base.

(Ex.:$log = $self→_Log($x,$y)

5 Example

A minimal example demonstrating the usage of BWT.pl demo program:

```
perl ./bin/PhyComplex.pl -i demo/Test.fa
```

```
CADANIAT00000001 cds:known ... 391.25794853831
```

5.1 Physical.pm

Adding the `use Complexity::Physical;` module file to your program will allow you to include all the functions described in section 4. A minimal example:

```
use Complexity::Physical;

my $PSobject = Complexity::Physical->new();

$array = [['-','A','B','C','H','-','I','I','-'],
          ['-','A','C','C','S','-','-','I','-'],
          ['-','-','C','C','A','-','O','I','-']];

$query = 0;

# -----        Compute Physical Complexity        ----- #
my $ps = $PSobject->PhyComplexCompute(queryid => $query, aligned => $array);

# -----               Print result                ----- #
print "Physicyl complexity of sequence 0 is: $ps\n"; ## Expected result:  3
```

6 Acknowledgement

1. Adami, C. and Cerf, N.J. (2000) Physical complexity of symbolic sequences, Physica D 137 62-69.

2. Edgar, RC (2004). "MUSCLE: multiple sequence alignment with high accuracy and high throughput". Nucleic Acids Research 32 (5): 179297

7 Future work

Upon request!