Project Report for Course Comp 598 Fall 2016

Inferring RNA ancestors

Student:	Faizy Ahsan
Instructor:	Prof. Jerome Waldispuhl

School of Computer Science
Center for Bioinformatics
McGill University, Montreal

December 2016

Abstract

The goal of this project is to develop methods for inferring RNA sequences of ancestors from the given extant species RNA sequences. We use Vienna RNA package to predict and analyze RNA secondary structures available from Rfam database. We predict ancestral sequences using modified version of Sankoffs algorithm. We force each ancestral sequence to preserve the base pair dependency of consensus structure. Finally, we identify and analyze several RNA families based on the stability to their consensus structure.

1 Introduction

Ribonucleic Acids (RNAs) are nucleic acids that play an essential role in various biological processes like coding, decoding, regulation and expression of genes. RNAs are synthesized from Deoxyribonucleic acid (DNA) strands through a process called transcription. RNAs are classified into different families based on their evolution from a common ancestor. In this project we develop methods to infer RNA sequences for RNA families. We present the problem statement in §2 and explain the methods in §3. The data set is described in §4. The results are presented in §5 and we conclude with §6.

2 Problem Definition

We are provided with a set of aligned RNA sequences from an RNA family and a phylogenetic tree, T, relating these sequences. We have to predict the RNA sequences of each ancestral node in T.

3 Methods

We read the aligned sequences, L, and the secondary structure of consensus sequence, S, from the Stockholm file of an RNA family. We use Sankoff algorithm [4] to predict ancestral sequences from L. The penalty for 'gap' in the sequence is included in the cost matrix, which looks like,

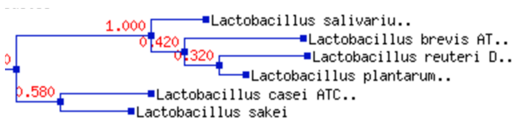
	A	C	G	U	gap
A	0	2	1	2	2
C	2	0	2	1	2
G	1	2	0	2	2
U	2	1	2	0	2
gap	2	2	2	2	0

We force the ancestral sequences to have same base pair dependencies as in S. We identify all the base pair positions (i,j) in S. If i is A or C, j becomes U or G. If i is U, j becomes A or G with probability 0.5 and if i is G, j becomes C or U with probability 0.5. Similarly, if i and j are gaps, (i,j) becomes one of the six valid base pairs with probability (1/6). And if only i is gap, we repeat the process with (j,i). The secondary structures are analyzed using the ViennaRNA package [3].

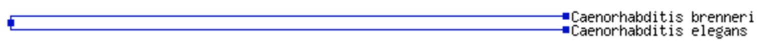
4 Data

We use the Stockholm files available from Rfam database [2]. We analyze the following four RNA families,

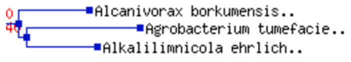
- 23S-methyl (RF01065)


1.000
0.420
0.320
0.580
Lactobacillus salivariu..
Lactobacillus brevis AT..
Lactobacillus reuteri D..
Lactobacillus plantarum..
Lactobacillus casei ATC..
Lactobacillus sakei

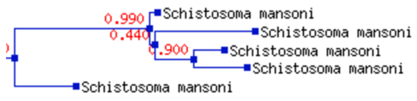
- Nematode snoRNA ceN27 (RF01624)



- mini-ykkC RNA motif (RF01068)


Alcanivorax borkumensis..
Agrobacterium tumefacie..
Alkalilimnicola ehrlich..

- Hammerhead ribozyme (type I) (RF00163)



5 Results

In this section, we show results with the eight objectives mentioned in the project description [1]. We develop a python file 'rnaAncestor.py' to do the analyses with the secondary structures that are described below.

5.1 Objective 1: Parser for Stockholm files

We implement the parser in the method **stockholmParser(filename, tree)**. It takes Stockholm file name and its type as input filename and tree. The input variable 'tree' should be set false when the method is called for parsing the Stockholm file. The method would return the alignments as a dictionary with species name and its sequence as key, value pairs. The method returns the consensus structure too. The following code snippet shows the working of this method with RNA family RF01065,

```
[alignment, consensus] = stockholmParser( 'RF01065.txt', False )
your file:  RF01065.txt
consensus structure:
:<.<<<<<<<<<<<<<<<<<...._________......>>>>>>>>>>
>>.>>>>>>----..------<<<<<<<<<<<<<<--<<<_____
>>>>>>>>>>>>>>>>>:::::
Alignments:
CP000423.1/889561-889673
UU-AUUUUUAACUUUUAUCAGGGCUAGCUUUUGGGUAAUUGAUAA
AAGUUU-GCGAUAAAUUACUUCACUAACGUAUCGCUUCGCUGAGU
GCG-CUCCGAAGCAAUGCGUUUUUUU
CR936503.1/496018-496118
UU-UUCCCUAACUUUUAUCAG---AAUACUUUU------UGAUAAA
AGCUA-GUGAUGAAAU--GACACUAACGUAGUUUCUCGCCGUGAA
UAAC--CGAGGGACUACGUUUUUUU
AL935263.2/1998551-1998453
CG-UUUGGUAGUUAACAUCG----ACAUGU---------CGUUGGUG
ACUA-CCGAGUUGUAC-UUCAUUAACGUAGCGUUUCGCCGUGCCCA
-CACCGAAAUACUACGUUUUUUU
CP000705.1/649443-649549
UCGUUUGAUAAUUGACAUCGAAUAUAAGUGUUU------CGUUGGUA
AUUA-UCAAGGAGUAC-UUCAUUAAAGCAGCGUUUCGCCGUGCCCA-
CACCGAGACCCUGCUUUUUUUA
```

```
CP000233.1/1171100-1171001
CG-GUUGUUAGUAAUUAACG----GAACUGU--------CGUUAGUUG
CUAUAUAAUUGUAC--UUCAUUAACGUAGCUGUUCGCCGUGCCCA-C
ACCGAAUGGCUACGCUUUUUU
CP000416.1/1433094-1432993
CG-UUUGGCGGUCGAUAUCA----GCGUUUAAC------UGUUAGCGG
CAG-ACAAGUAGCAC-UUCACUAAAGUAGUUCCUCGCCGUGCCCA-C
ACCGAGGAGCUGCUUUUUUUG
```

5.2 Objective 2: Implement Sankoff Algorithm

We implement Sankoff algorithm with the method **sankoff(parent, leaves, cost_matrix, alignment)**. It takes four inputs,

- parent: dictionary with ancestor and its children nodes as key value pairs
- leaves: list of all extant RNA sequences
- cost_matrix: list of lists as the penalty matrix for the mutations
- alignment: dictionary with extant species name and its RNA sequence as key value pairs.

This method computes the ancestral sequences and returns a dictionary with all node name and the node's sequence of the phylogenetic tree as key value pairs.

5.3 Objective 3: Expand Sankoff Algorithm

We modify the cost_matrix by including the gap penalties and pass it to the **sankoff(.)** method.

5.4 Objective 4: Calculate Ancestral RNA sequences

We calculate the ancestral RNA sequences using the method **sankoff(.)**. It returns the ancestral species name and its sequence as a key, value pair.

5.5 Objective 5: Secondary structure and Distance

We compute the secondary structure with the method **computeStruc(sequence)**. It takes a sequence and returns the secondary structure computed by RNAfold.

The distance between the ancestral sequence and the consensus structure is computed by the method **bp_distance(fold, target)**. It takes the secondary structure of the ancestral sequence as fold and the consensus secondary structure as target. It returns the distance computed by RNAdistance.

5.6 Objective 6: Base Pairing dependencies

We compute the ancestral RNA sequences using the method **sankoff(.)**. Then, for each ancestral sequence we call the method **extendedSankoffwithBpDependency(sequence, target)**. The input 'sequence' is an ancestral sequence and 'target' is the consensus secondary structure. The method returns the sequence modified by the method described in §3 in order to have the consensus structure base pair dependencies.

5.7 Objective 7: Repeat experiments

We calculate the secondary structure and compute the distance from the secondary structures of the sequences returned from the method **extendedSankoffwithBpDependency(.)**. The mean of distances calculated without and with base pairing dependencies are shown in the columns meanDist1 and meanDist2 of table 1.

5.8 Objective 8: Experiments with more families

We experimented with the four RNA families mentioned in §4. The results are shown in table 1. Column Length shows the length of sequences in each family. The number of extant species are shown in 'No. of Sequences' column. The column 'GC_content' shows the GC content in the extant species. The columns 'meanDist1' and 'meanDist2' shows the mean distances of ancestral sequences that are calculated without and with base pairing dependencies respectively. The column 'mean ancestor MFE' shows the mean of minimum free energy as computed by RNAfold with '-p' option on ancestral sequences with base pairing dependency.

Table 1: Results with Four RNA families

Family	Length	No. of Sequences	GC_content	meanDist1	meanDist2	mean ancestor MFE
RF01065	116	6	257	48.4	16	0.058
RF01624	80	2	66	90	90	0.12
RF01068	55	3	108	15	8	0.14
RF00163	124	5	127	38	27	0.19

It should be noted that the base pairing enforcement on the ancestral sequences are performed with a probability i.e. an invalid base pair is replaced with a valid base pair using a random draw. Therefore, the values in columns 'meanDist1', 'meanDist2' and 'mean ancestor MFE' could vary with each run of the program 'rnaAncestor.py'. However, the values shown in these columns are a good estimate over several run times.

6 Conclusion

We fulfill all the objectives that were asked in the project description [1]. We observe that the family RF01065 is most stable with the consensus structure. The minimum free energy is also dependent on other factors like length, no. of sequences and the GC content. Often, the minimum free energy positively correlates with the difference of meanDist1 and meanDist2. However, it is not true if few extant species are considered with small length sequences and small number of sequences as in the case of family RF01624.

We developed a robust python program to predict and analyze the secondary structures of ancestral RNA sequences. Experiments on more RNA families with our program should conclude with the relationship of the length, number and GC content of sequences to the stability of RNA ancestral sequences on the consensus structure.

References

[1] Project description. `http://www.cs.mcgill.ca/~jeromew/docs/comp598/projects/RNA_ancestors.pdf`. [Online; accessed 15-December-2016].

[2] Sam Griffiths-Jones, Alex Bateman, Mhairi Marshall, Ajay Khanna, and Sean R Eddy. Rfam: an rna family database. *Nucleic acids research*, 31(1):439–441, 2003.

[3] Ronny Lorenz, Stephan H Bernhart, Christian Hoener Zu Siederdissen, Hakim Tafer, Christoph Flamm, Peter F Stadler, and Ivo L Hofacker. Viennarna package 2.0. *Algorithms for Molecular Biology*, 6(1):1, 2011.

[4] David Sankoff. Simultaneous solution of the rna folding, alignment and protosequence problems. *SIAM Journal on Applied Mathematics*, 45(5):810–825, 1985.