Searching Genomes for Noncoding RNA

Based on the following papers:

1. Zhang S, B Hass, E Eskin, V Bafna. "Searching genomes for noncoding RNA using FastR", IEEE/ACM Transactions on Computational Biology and Bioinformatics, 2(4) October-December 2005.
2. Zhang S, I Borovok,, Y Aharonowitz, R Sharan, V Bafna. "A sequence-based filtering method for noncoding RNA identification and its application to searching for riboswitch elements", Bioinformatics, 22(14): e557-e565, 2006.

Outline

1. Background

DNA is transcribed to messenger RNA (mRNA) and then translated to proteins. The human genome is composed of roughly three billion bases of DNA. However, there are only twenty-two thousand genes that code for proteins. Consequently, genes only make up 1.5% of the genome.

Some genes encode RNA molecules that do not code for proteins, but are transcribed into RNA molecules involved in cellular regulatory processes. Sometimes these sequences are called *RNA genes* or noncoding RNAs. There is an "Expanded universe" of noncoding RNA, the importance scientists are appreciating more with every day. Noncoding RNAs play a wide variety of cellular roles including the following:

- rRNA – ribosomal RNA (structure/function of ribosomes)
- tRNA – transfer RNA (translation)
- snRNA – small-nuclear RNA (RNA splicing, telomere maintenance)
- snoRNA – small-nucleolar (chemical modification of rRNA)
- miRNA – microRNA (translational regulation)
- gRNA – guideRNA (mRNA editing)
- tmRNA – tRNA/mRNA combination molecule (degradation of defective proteins)
- riboswitches (translational and transcriptional regulation - *figure 0*).
- ribozymes (autocatalytic RNA)
- RNAi – RNA interference (gene regulation by double-stranded RNA)

RNA is a hot topic and of fundamental biological importance. Noncoding RNAs are an essential part of transcription, translation, alternative-splicing, and gene regulation. Just last week, professors Andrew Fire (at Stanford Medical School) and Craig Mello received the Nobel Prize for their discovery of RNA interference.

Since RNA is usually single-stranded (as opposed to DNA) its bases often bind with each other, causing the RNA polymer to fold upon itself into a specific conformation. The description of which bases form bonds with each other is called the secondary structure of an RNA molecule. The secondary structure of noncoding RNA molecules is frequently more essential to its function than its sequence. As a result secondary structure is an important part of many tools involved in noncoding RNA discovery. This is because the secondary structure of orthologous noncoding RNA sequences is evolutionarily conserved.

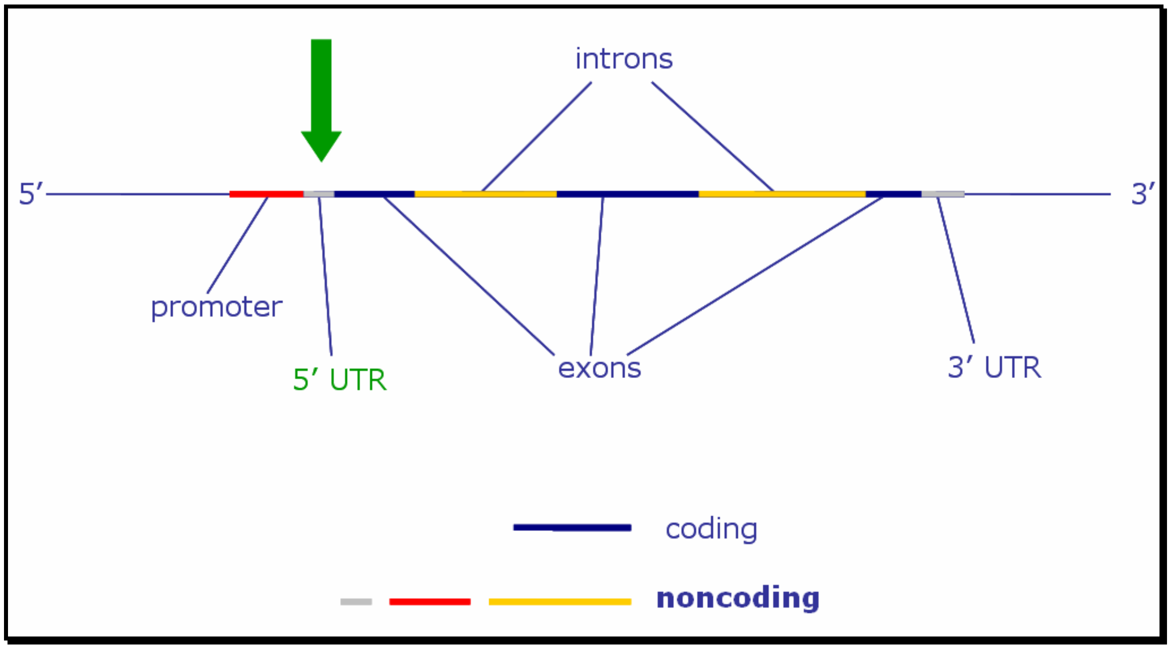
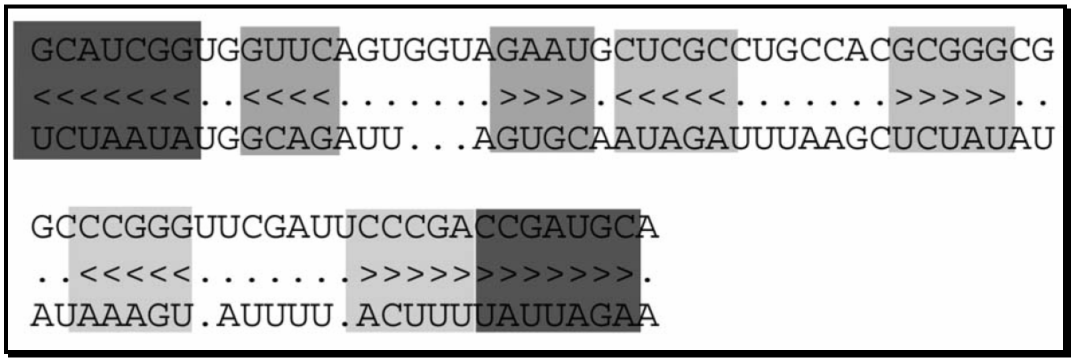


Figure 0: Riboswitches - a **riboswitch** is a part of an mRNA molecule that can directly bind a small target molecule, and whose binding of the target affects the gene's activity. Thus, an mRNA that contains a riboswitch is directly involved in regulating its own activity, depending on the presence or absence of its target molecule. Riboswitches are usually found in the 5' UTR (Untranslated Region) of genes.

2. NonCoding RNA Prediction

Since genes make up only a small percentage of the gene prediction has become a standard problem in computational biology. Basic gene prediction programs look for signals such as start and stop codons. These are three-base sequences that tell the translation machinery to start and stop. They also look for statistically-significant sequence conservation across related genomes.

Prediction of RNA genes is more challenging because noncoding RNA signals in the genome are not as strong as the signals for protein coding genes. This is because sequence conservation is frequently statistically insignificant. Often times, only a small percentage of residues in an RNA regulatory molecule must be conserved to maintain its function.



```
GCAUCGGUGGUUCAGUGGUAGAAUGCUCGCCUGCCACGCGGGCG
<<<<<<<..<<<<.......>>>>.<<<<<.......>>>>>..
UCUAAUAUGGCAGAUU...AGUGCAAUAGAUUUAAGCUCUAUAU

GCCCGGGUUCGAUUCCCGACCGAUGCA
..<<<<<.......>>>>>>>>>>>>.
AUAAAGU.AUUUU.ACUUUUAUUAGAA
```

Figure 1: Alignment of two tRNA sequences from *Drosophila melanogaster*

On the other hand, the secondary structure (*figure 2*) of noncoding RNA molecules often usually highly conserved, providing another tool for finding co-variation across genomes. However, structure is also frequently inadequate for detecting noncoding RNAs. If we just scan for structure we will find random sequences that will fold into ways that suggest they are functional.

These noncoding RNAs can also be found in a wide range of places. Some noncoding RNAs are whole transcribed units such as rRNA, whereas others such as riboswitches are in the UTRs of genes. They may be in intergenic regions, introns, and while they are rarely in coding-regions they may be on the reverse strand of coding regions. Thus, this huge search space demands fast algorithms.

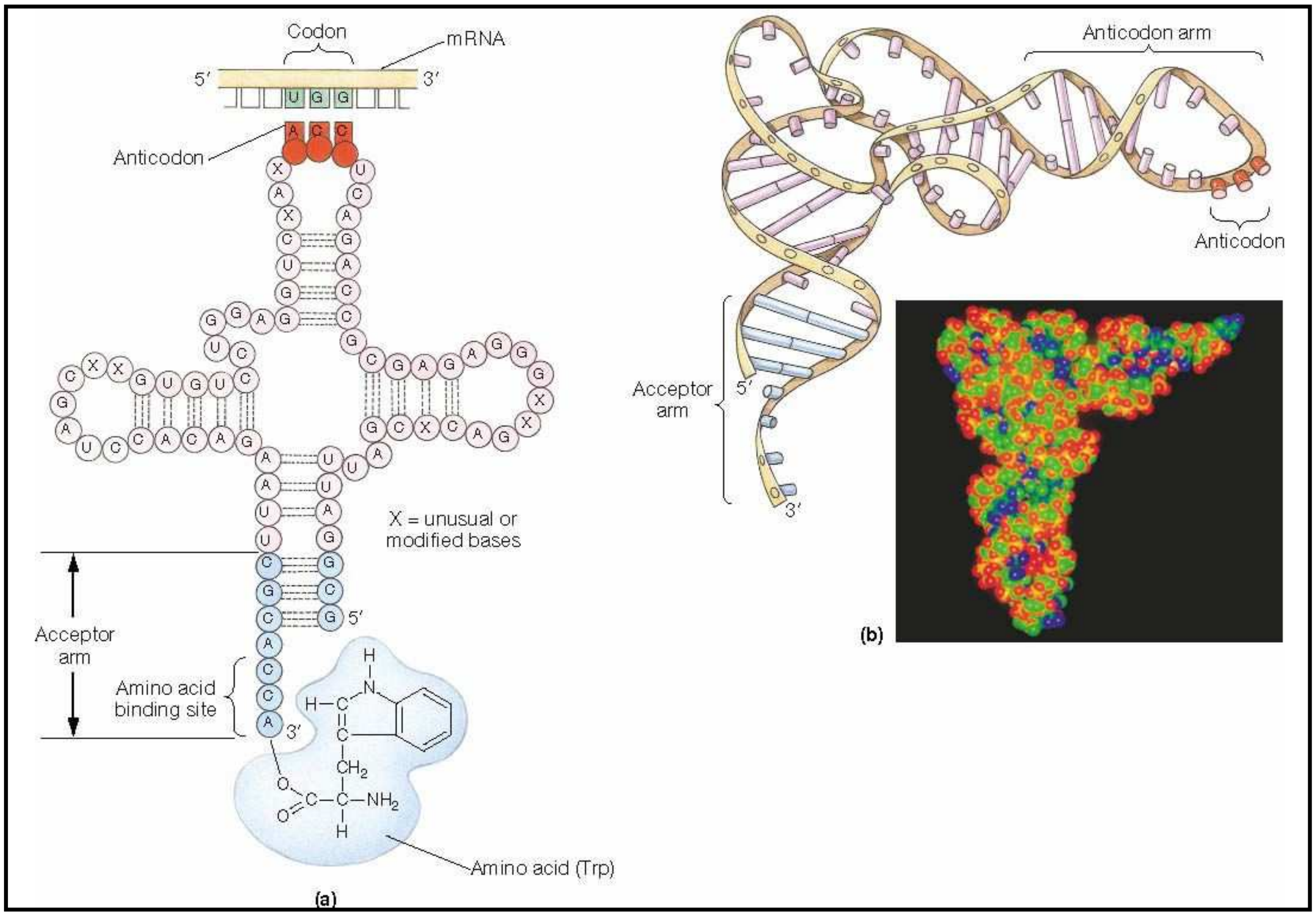
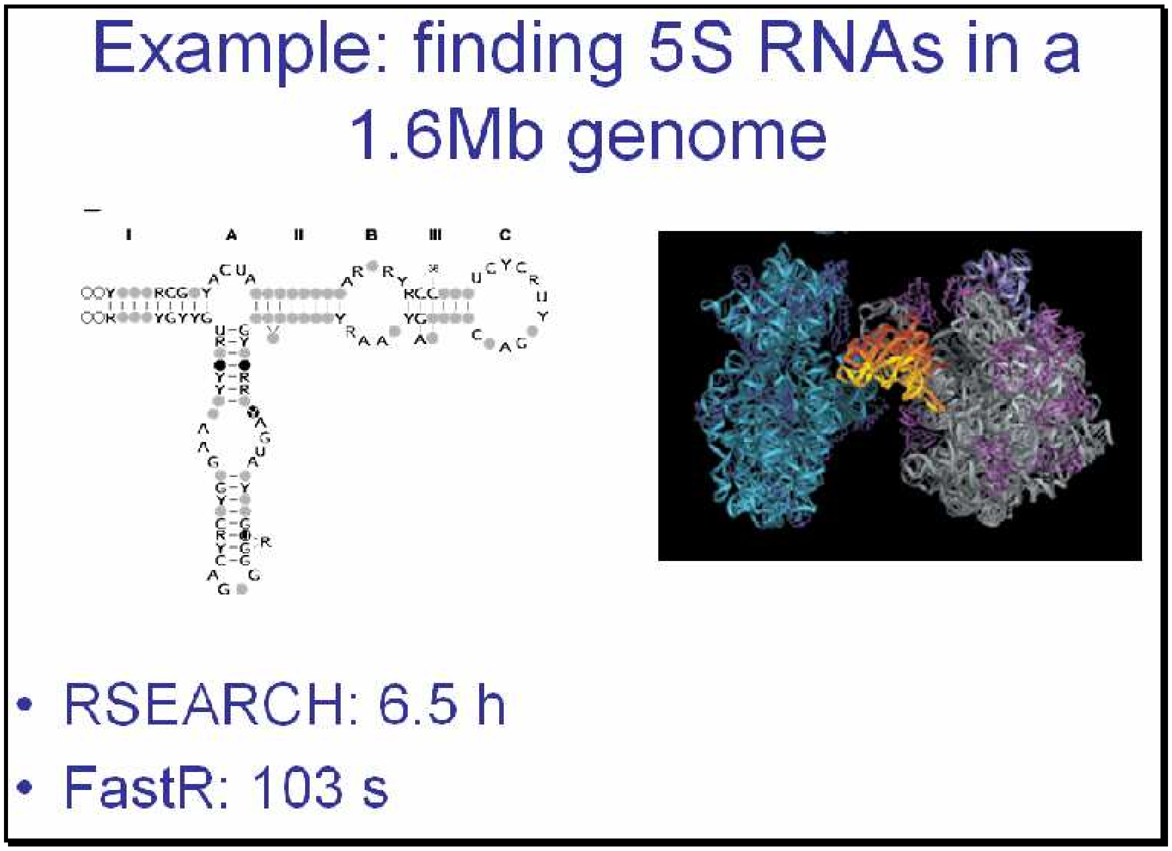


Figure 2: 2D and 3D structure. The image on the left is an example of the 2D structure of a tRNA molecule. The two images on the right are depictions of the 3D structure of a tRNA molecule.

3. Searching Genomes for Noncoding RNA Using FastR

If sequence conservation and structural conservation are both inadequate for noncodingRNA finding then perhaps a combination approach may be best. This is the approach taken by Zhang et al. in their gene predictor FastR, which looks for structure in evolutionary conserved sequences. The specific goal of the application of their program in their paper *Searching Genomes for Noncoding RNA Using FastR* is to find new instances of a given noncoding RNA family in new genomes.

Programs with the same type of functionality as FastR that already exist include CMSearch, RSEARCH and ERPIN. However, FastR searches genomes much faster – hence its name. As an example of the speed-up Zhang et al. performed a search of 5S RNAs in a 1.6Mb genome (relatively small compared to human 3Gb genome) using FastR and RSEARCH. RSEARCH completed the scan in 6.5 hours, whereas FastR took 103 seconds (*figure 3*). FastR is able to achieve this amazing speed-up by applying a database filter as part of a pre-processing step before aligning any genomic sequences to the query sequence.



Example: finding 5S RNAs in a 1.6Mb genome

- RSEARCH: 6.5 h
- FastR: 103 s

Figure 3: Example benchmarking. 5S RNA is a ribosomal RNA molecule. Part of its 2D structure is shown on the left. Its 3D structure is one of the molecules in the picture of the ribosome on the right. In this comparison RSEARCH took 6.5 hours to search a 1.6Mb genome for 5S RNAs, whereas FastR only took 103 seconds.

4. FastR's Genome Filtering Method

A database filter is a computational procedure that takes a database as input and outputs a subset of that database (*figure 4*). The purpose of the database filter is to reduce a search space. A good has the following attributes:

- The object being searched for remains in the database after filtering (sensitivity).

- The filtered database is significantly smaller than the original database.
- The filtering operation is fast (efficiency).

A filter is useless if it filters out the item being searched for. Also, a filter that removes an insignificant amount of the database may not have any impact on the search time, and may even take longer than the speed increase. Also, if the filtering operation takes more time than the search, the operation is useless. An efficient filter is a one that quickly removes large amounts of the original database and a sensitive filter is one that does not incorrectly remove instances of the searched-for item.

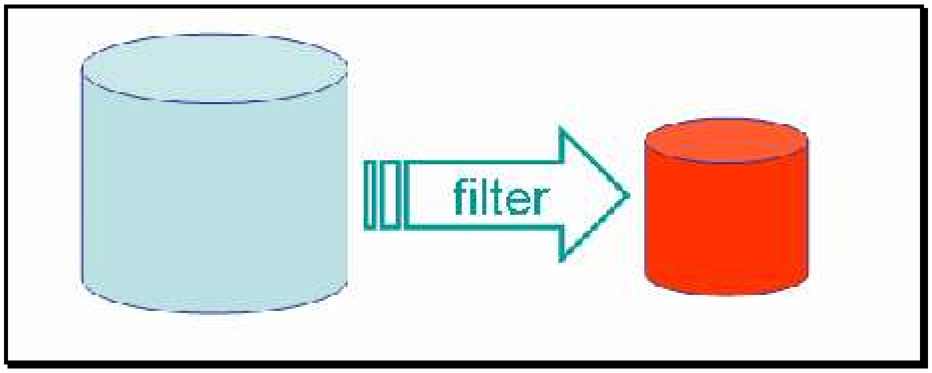


Figure 4: A good database filter is one that significantly reduces the search space

FastR's solution is to filter a genome (the database) using both sequence and structural features. The basic structural feature the filter uses is called a (k, w) stack

DEFINE: (k, w) stack: A pair of substrings of at least length k that are at most w bases apart (*figure 5*).

If we use a (7,70)-stack filter, we eliminate 90% of the DB from consideration. This stack is common to members of the tRNA family, so it is an appropriate example of a filter. How much of the genome FastR can filter depends on the query RNA family.

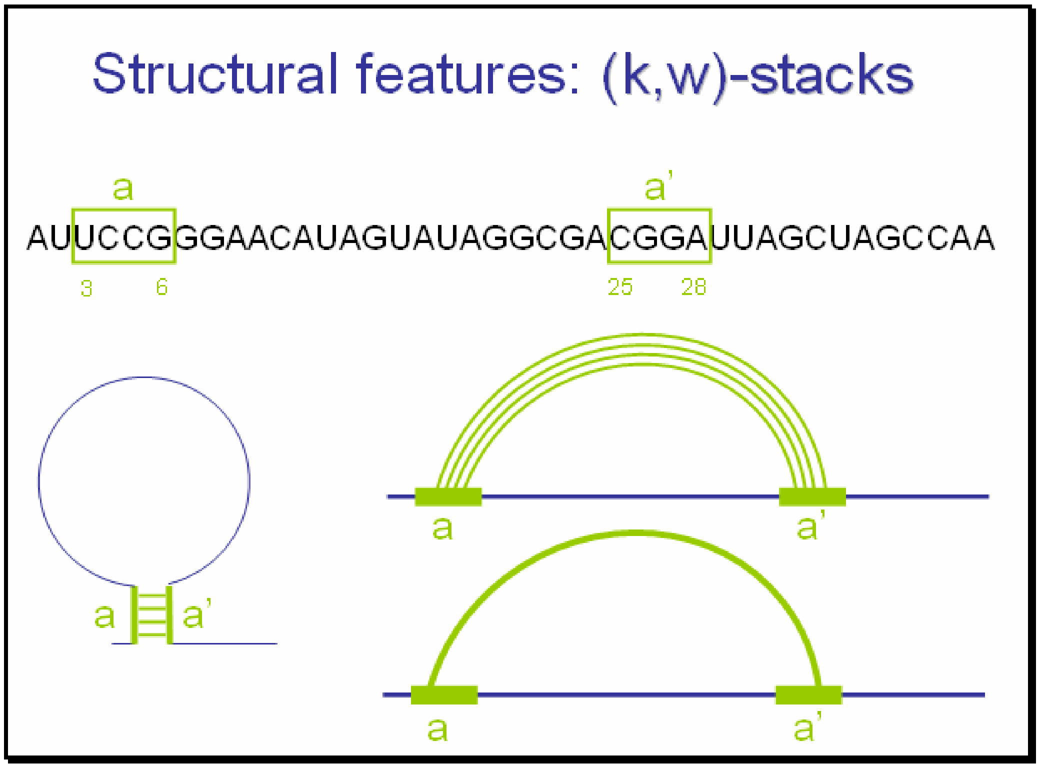

Structural features: (k,w)-stacks
a
a'
AUUCCGGGAACAUAGUAUAGGCGACGGAUUAGCUAGCCAA
3
6
25
28
a
a'
a
a'
a
a'

Figure 5: A (k, w) stack is a pair of substrings of at least length k that are at most w bases apart

FastR's filter allows for three more complex (k, w) stacks:

- Nested (k, w, l) stacks – two (k, w) stacks where one is inside the other.
- Parallel (k, w, l) stacks - two (k, w) stacks that are next to each other.
- Multiloop (k, w, l) stacks – parallel stacks nested in another stack

For each of these stacks the variable "l" refers to the distance between the stacks in question. These different stacks are illustrated in *figure 6-8.*

The filtering algorithm involves a two step process. First FastR builds a hash table of the position of all kmers (sequences of length *k*) in the database, where the keys are kmers and the values are the indices where instances of the kmer begin. It then identifies all of the (k, w) stacks. To do this it iterates over each kmer in the hash table and sees if its reverse compliment exists. If the reverse compliment does exist in the hash table, it sees if any of the kmer/reverse compliment pairs are within w distance apart. FastR then computes complex stacking using *dynamic programming* methods.

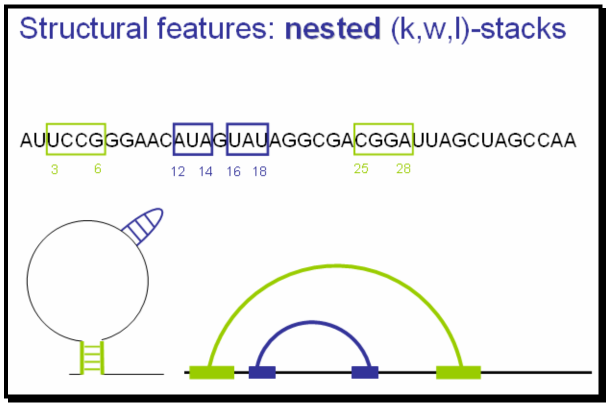

Structural features: nested (k,w,l)-stacks
AUUCCGGGAACAUAGUAUAGGCGACGGAUUAGCUAGCCAA
3 6 12 14 16 18 25 28

Figure 6: Nested Stacks are simple stacks where one is within the other.

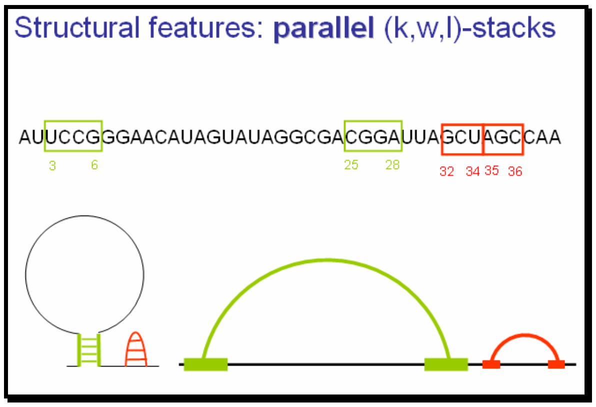

Structural features: parallel (k,w,l)-stacks
AUUCCGGGAACAUAGUAUAGGCGACGGAUUAGCUAGCCAA
3 6 25 28 32 34 35 36

Figure 7: Parallel Stacks are simple stacks that occur next to each other.

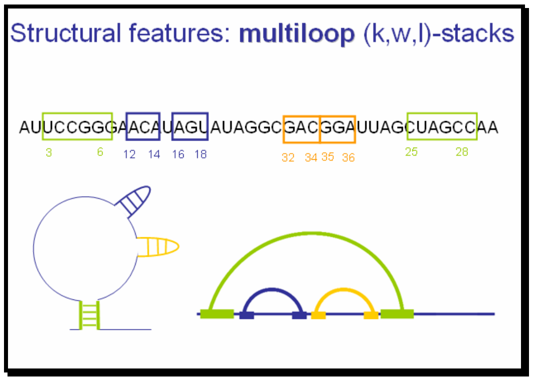

Structural features: multiloop (k,w,l)-stacks
AUUCCGGGAACAUAGUAUAGGCGACGGAUUAGCUAGCCAA
3 6 12 14 16 18 32 34 35 36 25 28

Figure 8: Multiloop Stacks are parallel stacks nested in another stack

5. FastR's RNA Alignment Method

There are three possible ways in which a query RNA sequence can be aligned to genomic sequences taking into account sequence and secondary structure:

- Query sequence to genomic sequence
- Query structure to genomic structure
- Query structure to genomic sequence

FastR uses the third approach. It represents the secondary structure of the query sequence as a binary tree and then searches for it in the genomic sequence. The tree's structure is defined by three rules. Starting on opposite ends of the sequence, where *i* is the rightmost term under consideration and *j* is the leftmost term under consideration:

- If *i* and *j* are paired create a black node with one child. Increment *i*, decrement *j*, and move to the child.
- When *j* is unpaired create a white node with one child. Decrement *j* and move to the child.
- When *j* is paired, but not to *i,* create a white node with two children and recurse upon both children. For the left child *i=i* and *j=k*, where k is between *i* and *j*. For the right child *i=k*+1 and *j=j*.

These rules are illustrated in *figure 9-11.* An example binary tree for a secondary structure is shown in *figure 12.*

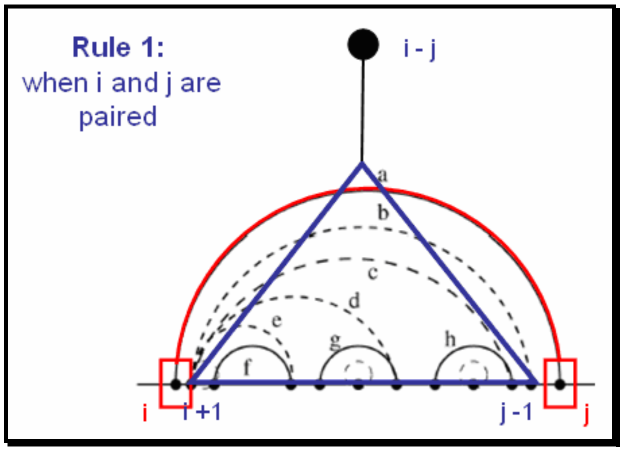


Figure 9: Rule 1 of the binary tree construction

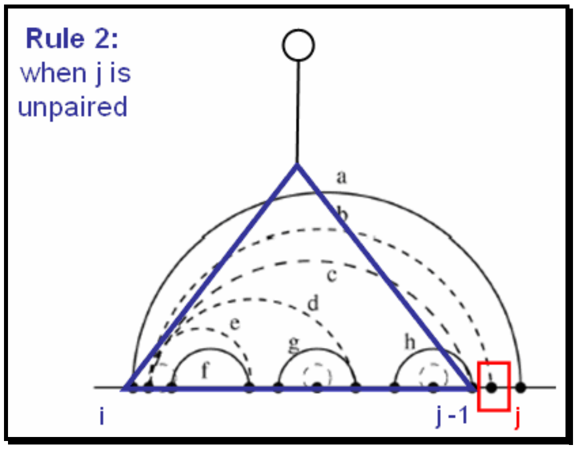


Figure 10: Rule 2 of the binary tree construction

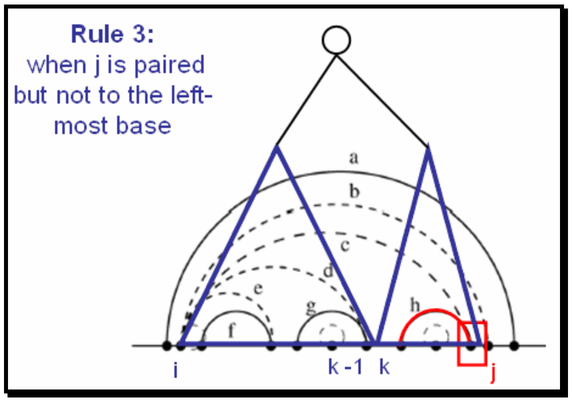


Figure 11: Rule 3 of the binary tree construction

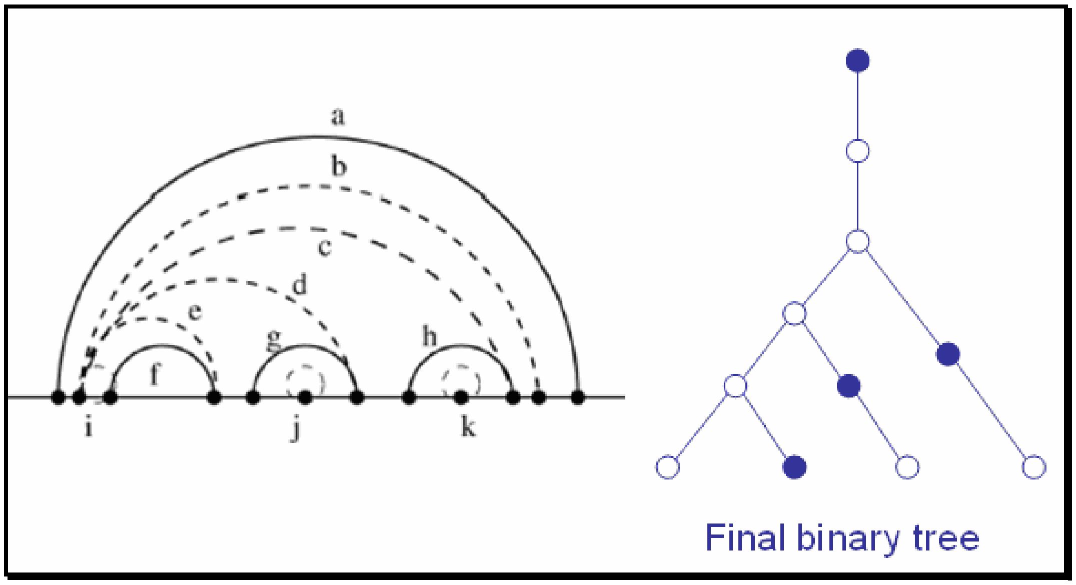


Figure 12: A sample binary tree

The pseudocode for the binary tree construction is shown below:

```
procedure Binarize(i,j)   (* Binarize the interval $(i, j)$. *)
if $(i = j)$
    return (create_node(i,j,dotted,Nil)); (* A dotted node with 0 child. *)
if $(i, j) \in S$
    v = Binarize(i+1,j-1);
    return (create_node(i,j,solid,v)); (* A solid node with 1 child $v$. *)
if $(k, j) \in S$ for some $i < k < j$
    vl = Binarize(1,k-1);
    vr = Binarize(k,j);
    return (create_node(i,j,dotted,vl,vr)); (*A dotted node with 2 children, $vl$ and $vr$. *)
if $(i < j)$
    v = Binarize(i,j-1);
    return (create_node(i,j,dotted,v)); (* A dotted node with 1 child $v$. *)
end if
```

Given this data structure the optimal alignment is performed using *dynamic programming* on a three dimensional matrix in which the axes are *i, j,* and v, where *i* is the starting point of a genomic sequence, *j* is the end point of a genomic sequence and *v* is the first node of the binary tree that represents the query structure. A mathematical definition of this procedure is shown below:

```
procedure alignRNA
(*S is the set of base-pairs in RNA structure of $s$. $S'$ is the augmented set. *)
for all intervals $(i, j)$, $1 \le i < j \le n$, all nodes $v \in S'$
    if $v \in S$
        $A[i,j,v] = \max \begin{cases} A[i+1, j-1, \text{child}(v)] + \delta(t[i], t[j], s[l_v], s[r_v]), \\ A[i, j-1, v] + \gamma('-', t[j]), \\ A[i+1, j, v] + \gamma('-', t[i]), \\ A[i+1, j, \text{child}[v]] + \gamma(s[l_v], t[i]) + \gamma(s[r_v], '-'), \\ A[i, j-1, \text{child}[v]] + \gamma(s[l_v], '-') + \gamma(s[r_v], t[j]), \\ A[i, j, \text{child}[v]] + \gamma(s[l_v], '-') + \gamma(s[r_v], '-'), \end{cases}$
    else if $v \in S' - S$, and $v$ has one child
        $A[i,j,v] = \max \begin{cases} A[i, j-1, \text{child}[v]] + \gamma(s[r_v], t[j]), \\ A[i, j, \text{child}[v]] + \gamma(s[r_v], '-'), \\ A[i, j-1, v] + \gamma('-', t[j]), \\ A[i+1, j, v] + \gamma('-', t[i]), \end{cases}$
    else if $v \in S' - S$, and $v$ has two children
        $A[i,j,v] = \max_{i \le k \le j}\{A[i, k-1, \text{left\_child}[v]] + A[k, j, \text{right\_child}[v]]\}$
    end if
end for
```

The *alignRNA* procedure is simplified in *figure 13-15*.

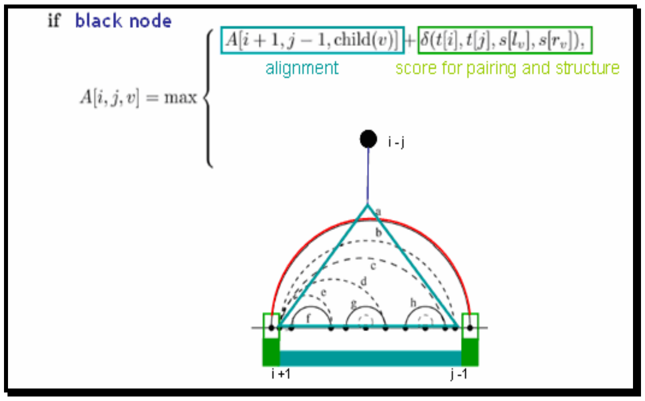

if black node
$A[i,j,v] = \max \{ \boxed{A[i+1, j-1, \text{child}(v)]} + \boxed{\delta(t[i], t[j], s[l_v], s[r_v]),}$
alignment
score for pairing and structure
i - j
a
b
c
d
e
f
g
h
i +1
j -1

Figure 13: Rule for a black node

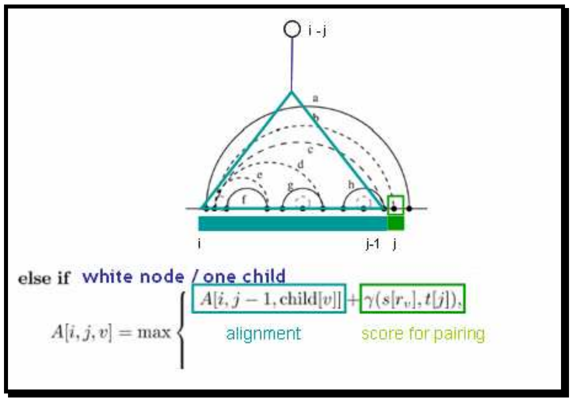

i - j
a
b
c
d
e
f
g
h
i
j-1
j
else if white node / one child
$A[i,j,v] = \max \{ \boxed{A[i, j-1, \text{child}[v]]} + \boxed{\gamma(s[r_v], t[j]),}$
alignment
score for pairing

Figure 14: Rule for a white node with one child

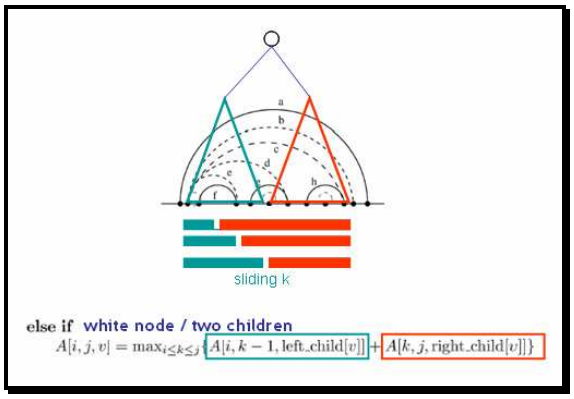
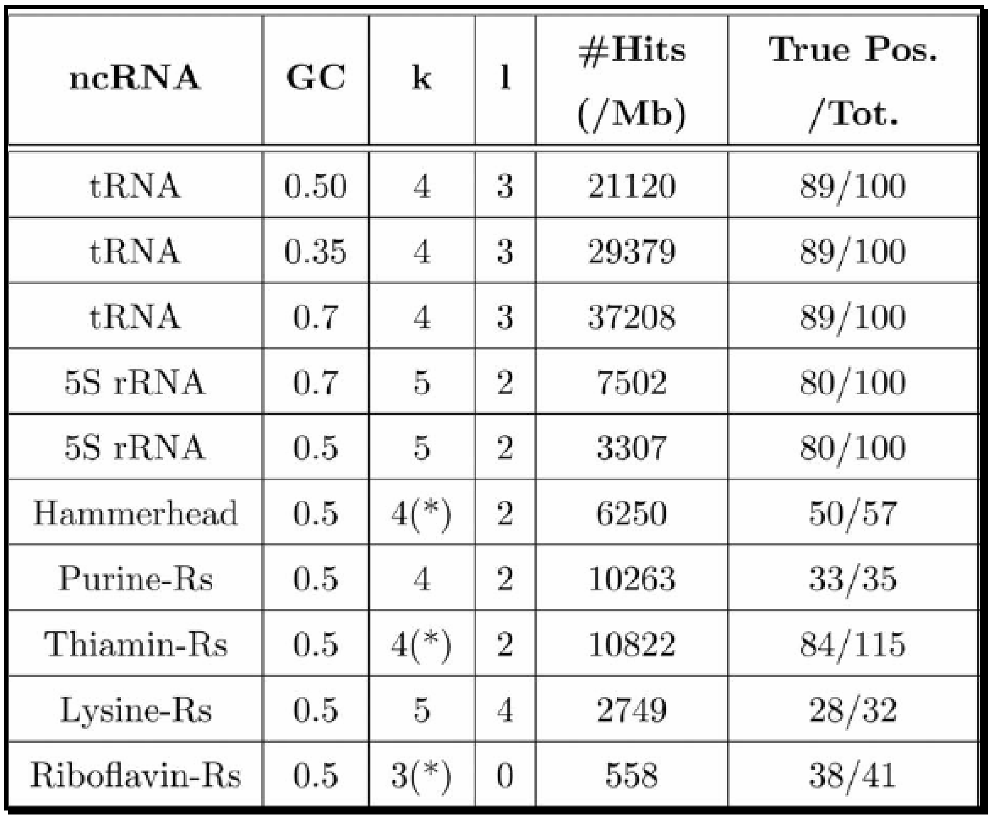

a
b
c
d
e
f
g
h
sliding k
else if white node / two children
$A[i,j,v] = \max_{i \le k \le j} \{ \boxed{A[i, k-1, \text{left\_child}[v]]} + \boxed{A[k, j, \text{right\_child}[v]]\}}$

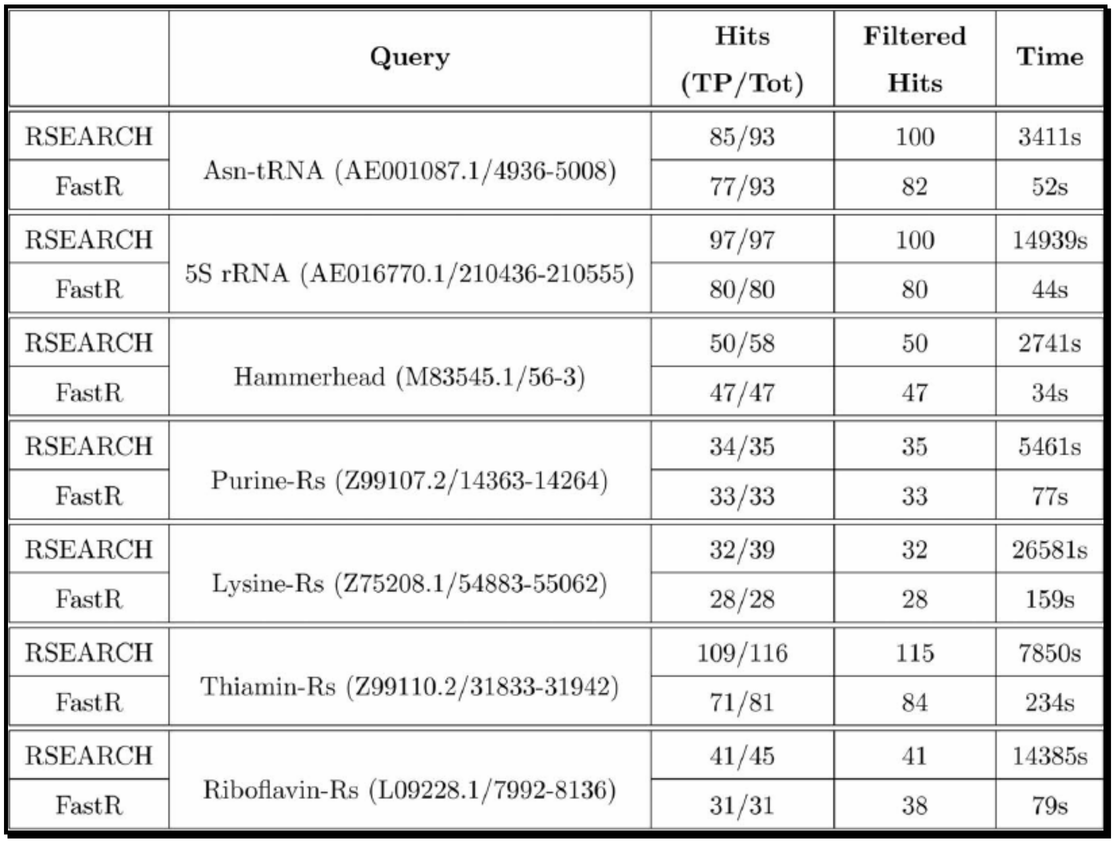
Figure 15: Rule for a white node with two children

Validation of FastR's Performance

A simple procedure was used to test FastR's performance and benchmark it against RSEARCH. Known instances of noncoding RNAs(tRNA, 5S rRNA, Ribozymes, riboswitches) were inserted in a random sequence (1MB) then they applied each algorithm. They found that FastR is somewhat less sensitive than RSEARCH but significantly faster. The results are shown in the following tables. The GC column equals the percent GC pairings in RNA family being tested.



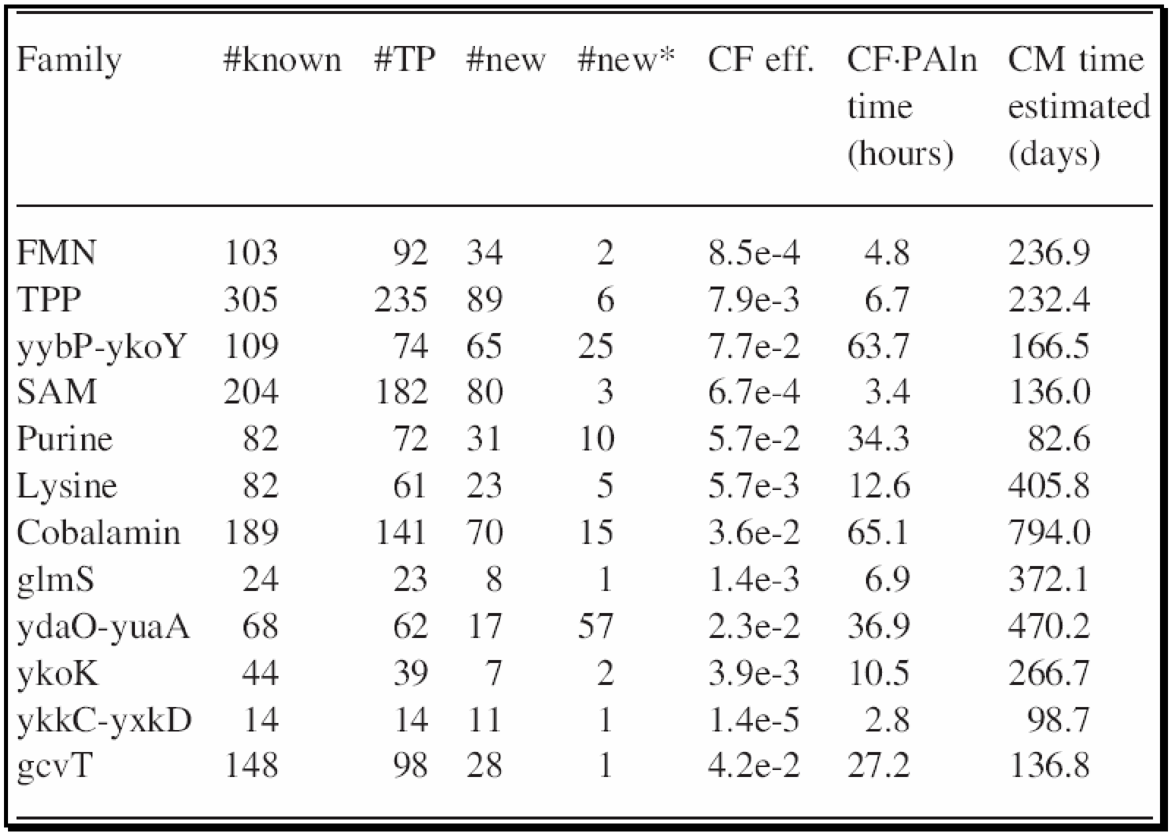
ncRNA	GC	k	l	#Hits (/Mb)	True Pos. /Tot.
tRNA	0.50	4	3	21120	89/100
tRNA	0.35	4	3	29379	89/100
tRNA	0.7	4	3	37208	89/100
5S rRNA	0.7	5	2	7502	80/100
5S rRNA	0.5	5	2	3307	80/100
Hammerhead	0.5	4(*)	2	6250	50/57
Purine-Rs	0.5	4	2	10263	33/35
Thiamin-Rs	0.5	4(*)	2	10822	84/115
Lysine-Rs	0.5	5	4	2749	28/32
Riboflavin-Rs	0.5	3(*)	0	558	38/41



	Query	Hits (TP/Tot)	Filtered Hits	Time
RSEARCH	Asn-tRNA (AE001087.1/4936-5008)	85/93	100	3411s
FastR		77/93	82	52s
RSEARCH	5S rRNA (AE016770.1/210436-210555)	97/97	100	14939s
FastR		80/80	80	44s
RSEARCH	Hammerhead (M83545.1/56-3)	50/58	50	2741s
FastR		47/47	47	34s
RSEARCH	Purine-Rs (Z99107.2/14363-14264)	34/35	35	5461s
FastR		33/33	33	77s
RSEARCH	Lysine-Rs (Z75208.1/54883-55062)	32/39	32	26581s
FastR		28/28	28	159s
RSEARCH	Thiamin-Rs (Z99110.2/31833-31942)	109/116	115	7850s
FastR		71/81	84	234s
RSEARCH	Riboflavin-Rs (L09228.1/7992-8136)	41/45	41	14385s
FastR		31/31	38	79s

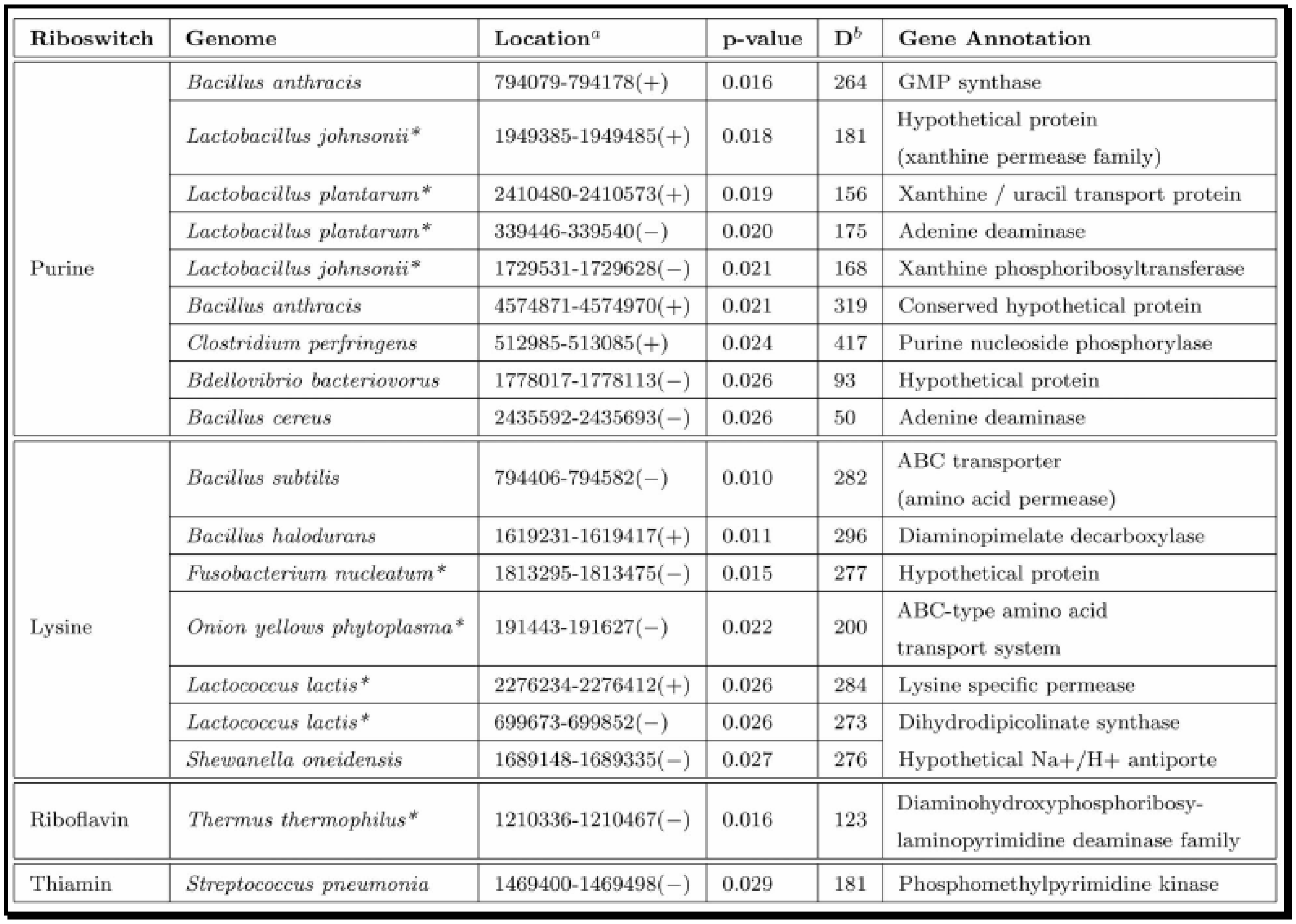
6. Application of FastR for Discovery of Novel Instances of Known Families of Riboswitches

In their paper *Searching genomes for noncoding RNA using FastR,* Zhang et al then looked for new instances of known riboswitch families. The table below summarizes their results, where *#known* is the number of known riboswitches in the all bacterial and archaeal genomes, *#TP* is the number of predicted known hits, *#new* is the number of new predictions in these genomes, and *#new** is the number of new predictions in these genomes that had previously been annotated as ncRNA in Rfam.



Family	#known	#TP	#new	#new*	CF eff.	CF·PAln time (hours)	CM time estimated (days)
FMN	103	92	34	2	8.5e-4	4.8	236.9
TPP	305	235	89	6	7.9e-3	6.7	232.4
yybP-ykoY	109	74	65	25	7.7e-2	63.7	166.5
SAM	204	182	80	3	6.7e-4	3.4	136.0
Purine	82	72	31	10	5.7e-2	34.3	82.6
Lysine	82	61	23	5	5.7e-3	12.6	405.8
Cobalamin	189	141	70	15	3.6e-2	65.1	794.0
glmS	24	23	8	1	1.4e-3	6.9	372.1
ydaO-yuaA	68	62	17	57	2.3e-2	36.9	470.2
ykoK	44	39	7	2	3.9e-3	10.5	266.7
ykkC-yxkD	14	14	11	1	1.4e-5	2.8	98.7
gcvT	148	98	28	1	4.2e-2	27.2	136.8

The next table describes what the authors describe as the eighteen most promising candidates from the 468 putative riboswitches discovered by FastR



Riboswitch	Genome	Location[a]	p-value	D[b]	Gene Annotation
Purine	*Bacillus anthracis*	794079-794178(+)	0.016	264	GMP synthase
	*Lactobacillus johnsonii**	1949385-1949485(+)	0.018	181	Hypothetical protein (xanthine permease family)
	*Lactobacillus plantarum**	2410480-2410573(+)	0.019	156	Xanthine / uracil transport protein
	*Lactobacillus plantarum**	339446-339540(−)	0.020	175	Adenine deaminase
	*Lactobacillus johnsonii**	1729531-1729628(−)	0.021	168	Xanthine phosphoribosyltransferase
	Bacillus anthracis	4574871-4574970(+)	0.021	319	Conserved hypothetical protein
	Clostridium perfringens	512985-513085(+)	0.024	417	Purine nucleoside phosphorylase
	Bdellovibrio bacteriovorus	1778017-1778113(−)	0.026	93	Hypothetical protein
	Bacillus cereus	2435592-2435693(−)	0.026	50	Adenine deaminase
Lysine	*Bacillus subtilis*	794406-794582(−)	0.010	282	ABC transporter (amino acid permease)
	Bacillus halodurans	1619231-1619417(+)	0.011	296	Diaminopimelate decarboxylase
	*Fusobacterium nucleatum**	1813295-1813475(−)	0.015	277	Hypothetical protein
	*Onion yellows phytoplasma**	191443-191627(−)	0.022	200	ABC-type amino acid transport system
	*Lactococcus lactis**	2276234-2276412(+)	0.026	284	Lysine specific permease
	*Lactococcus lactis**	699673-699852(−)	0.026	273	Dihydrodipicolinate synthase
	Shewanella oneidensis	1689148-1689335(−)	0.027	276	Hypothetical Na+/H+ antiporte
Riboflavin	*Thermus thermophilus**	1210336-1210467(−)	0.016	123	Diaminohydroxyphosphoribosy-laminopyrimidine deaminase family
Thiamin	*Streptococcus pneumonia*	1469400-1469498(−)	0.029	181	Phosphomethylpyrimidine kinase