Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Identificação de RNAs não-codificadores por modelos de covariância com prioris Dirichlet adaptadas a grupos de ncRNAs com estruturas secundárias similares (Identifying non-coding RNAs using covariance models with Dirichlet priors specific to groups of ncRNAs of similar secondary structures)

Felipe Almeida Lessa

Brasília
2012

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Identificação de RNAs não-codificadores por modelos de covariância com prioris Dirichlet adaptadas a grupos de ncRNAs com estruturas secundárias similares (Identifying non-coding RNAs using covariance models with Dirichlet priors specific to groups of ncRNAs of similar secondary structures)

Felipe Almeida Lessa

Dissertação apresentada como requisito parcial
para conclusão do Mestrado em Informática

Orientadora
Prof.ª Dr.ª Maria Emília Machado Telles Walter

Coorientadora
Prof.ª Dr.ª Daniele da Silva Baratela Martins Neto

Brasília
2012

Universidade de Brasília — UnB
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Mestrado em Informática

Coordenador: Prof. Dr.ª Mylene Christine Queiroz de Farias

Banca examinadora composta por:

Prof.ª Dr.ª Maria Emília Machado Telles Walter (Orientadora) — CIC/UnB
Prof. Dr. Peter F. Stadler — Bioinformatics Group/University of Leipzig
Prof. Dr. Marcelo de Macedo Brígido — IB/UnB
Prof.ª Dr.ª Tainá Alencar Raiol (suplente) — IB/UnB

CIP — Catalogação Internacional na Publicação

Lessa, Felipe Almeida.

Identificação de RNAs não-codificadores por modelos de covariância com prioris Dirichlet adaptadas a grupos de ncRNAs com estruturas secundárias similares (Identifying non-coding RNAs using covariance models with Dirichlet priors specific to groups of ncRNAs of similar secondary structures) / Felipe Almeida Lessa. Brasília : UnB, 2012.
96 p. : il. ; 29,5 cm.

Dissertação (Mestrado) — Universidade de Brasília, Brasília, 2012.

1. ncRNA, 2. RNA não-codificador, 3. CM, 4. modelo de covariância, 5. Dirichlet, 6. priori, 7. estrutura secundária, 8. Rfam, 9. dendrograma

CDU 004

Endereço: Universidade de Brasília
Campus Universitário Darcy Ribeiro — Asa Norte
CEP 70910-900
Brasília–DF — Brasil

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Identificação de RNAs não-codificadores por modelos de covariância com prioris Dirichlet adaptadas a grupos de ncRNAs com estruturas secundárias similares (Identifying non-coding RNAs using covariance models with Dirichlet priors specific to groups of ncRNAs of similar secondary structures)

Felipe Almeida Lessa

Dissertação apresentada como requisito parcial
para conclusão do Mestrado em Informática

Prof.ª Dr.ª Maria Emília Machado Telles Walter (Orientadora)
CIC/UnB

Prof. Dr. Peter F. Stadler
Bioinformatics Group/University of Leipzig

Prof. Dr. Marcelo de Macedo Brígido
IB/UnB

Prof.ª Dr.ª Tainá Alencar Raiol (suplente)
IB/UnB

Prof. Dr.ª Mylene Christine Queiroz de Farias
Coordenador do Mestrado em Informática

Brasília, 24 de agosto de 2012

Resumo

RNAs não-codificadores (ncRNA) são macromoléculas biológicas que, apesar de não codificarem proteínas, realizam funções regulatórias importante nas células. A ferramenta Infernal é principal referência em buscas de ncRNAs por homologia, e o banco de dados Rfam um dos mais importantes repositórios de ncRNAs. Neste trabalho, mostramos que há *clusters* de famílias de ncRNAs estruturalmente bem similares no Rfam, adaptamos a ferramenta Infernal com novas distribuições a priori específicas a certos grupos de famílias de ncRNAs baseados nesses *clusters* e mostramos que essas prioris específicas a certos grupos podem tornar o Infernal mais sensível e específico.

Palavras-chave: ncRNA, RNA não-codificador, CM, modelo de covariância, Dirichlet, priori, estrutura secundária, Rfam, dendrograma

Abstract

Non-coding RNA (ncRNA) molecules do not code for proteins, but instead play an important regulatory role in cells. Infernal is the main tool for homology searches for ncRNAs, and Rfam is one of the main ncRNA databases. On this work, we show that there are clusters of structurally very similar ncRNA families on Rfam, adapt Infernal with new priors constructed especially for certain groups of ncRNA families (based on these clusters) and show that these group-specific priors may improve Infernal's specificity and sensitivity.

Keywords: ncRNA, non-coding RNA, CM, covariance model, Dirichlet, prior, secondary structure, Rfam, dendrogram

Lista de Figuras

Lista de Tabelas

Sumário

Capítulo 1

Introdução

A visão clássica do Dogma Central da Biologia Molecular é a de que a informação genética flui do DNA – responsável pelo genótipo – para as proteínas – responsáveis pelo fenótipo – utilizando o RNA apenas como intermediário [7]. Contudo, hoje sabe-se que isso não é verdade. Em particular, existem RNAs funcionais que não são utilizados para construir proteínas, denominados RNAs não-codificadores (ncRNAs), mas que ainda assim são extremamente importantes para os mecanismos celulares [8].

Além disso, ao longo dos anos temos aumentado a vazão com a qual somos capazes de sequenciar genomas, culminando com os atuais *high-throughput sequencers*, ou sequenciadores de alto desempenho [9]. O aumento da vazão não significa apenas que podemos ter mais dados, mas também que podemos obtê-los a um custo mais baixo, o que tem ajudado a difundir o uso de sequenciadores de nucleotídeos.

Uma classe de métodos computacionais muito útil, apesar de não muito nova, é a busca por homologia [10]. Dois genes são ditos *homólogos* se possuem um ancestral comum. Em geral genes homólogos possuem sequências similares, e portanto foram desenvolvidas várias ferramentas computacionais que tentam prever homologia calculando medidas de similaridade entre sequências. O maior desafio passa então a ser definir uma medida de similaridade que seja eficiente e próxima o suficiente da homologia real (que pode existir mesmo com sequências bem diferentes).

Uma das ferramentas que popularizou a busca por homologia para proteínas é o BLAST [10]. Essa ferramenta utiliza heurísticas para minimizar o número de edições a serem feitas nas sequências para que as duas sejam idênticas. As edições podem ser inserções, remoções e substituições de bases, sendo que podem ser atribuídos pesos a cada tipo de edição. Apesar do sucesso do BLAST para a busca por proteínas homólogas, utilizá-lo para a busca por homologia de ncRNAs não tem trazido resultados satisfatórios (apesar de funcionarem em alguns poucos casos [11]). Uma causa de sua ineficácia, talvez a principal, é considerar apenas a estrutura primária da sequência, enquanto que a maioria dos ncRNAs depende muito de suas estruturas secundárias para realizar suas funções.

Por isso outros modelos foram desenvolvidos para identificar ncRNAs homólogos (cf. Seção 2.4), como os baseados em modelos de covariância (*covariance models*, CMs, usados neste trabalho) [12, 13], termodinâmica [14, 15], máquinas de vetor de suporte [16, 17, 18] e mapas auto-organizados [19]. Contudo, os métodos para identificar e classificar RNAs

não-codificadores ainda não são sensíveis e específicos o suficiente, além de em geral serem lentos.

A sensibilidade é a medida de quantos ncRNAs de fato são marcados como ncRNAs (i.e., de quantos ncRNAs não são encontrados na busca), definida como a razão entre o número de verdadeiros positivos e a soma dos verdadeiros positivos e falsos negativos (marcados erroneamente como não sendo ncRNAs). Um método com alta sensibilidade não deixa "escapar" nenhum resultado importante. Já a especificidade é uma medida análoga mas que refere-se a quantos resultados marcados como *não* sendo ncRNAs de fato não são ncRNAs. Um método de busca com baixa especificidade eleva os custos de quem usa os resultados pois vários falsos positivos são marcados. Na prática acaba-se tendo que escolher entre encontrar poucos candidatos a ncRNAs (abaixar a sensibilidade e aumentar a especificidade) ou encontrar muitos falsos positivos (aumentar a sensibilidade e diminuir a especificidade), e nenhuma alternativa é satisfatória.

O Infernal [13] é a principal ferramenta utilizada para buscas por homologia de ncRNAs. Dada uma família de ncRNAs, o Infernal constrói um modelo de covariância. Esse CM pode ser usado tanto para realizar buscas por novos ncRNAs da mesma família de ncRNAs quanto para alinhar uma nova sequência ao alinhamento múltiplo de sequências original da família.

O Rfam [20, 21] é um banco de dados de ncRNAs. Seu recurso mais distinto é estar organizado por famílias de ncRNAs, e não apenas sequências. Cada família de ncRNAs possui um alinhamento múltiplo de sequências (muitas vezes provenientes de diferentes espécies) que pertencem à família, sua estrutura secundária e metadados. Os CMs do Infernal são, em geral, construídos a partir de famílias de ncRNAs do Rfam.

Para construir um CM, o Infernal necessita, além da própria família de ncRNAs, de certas informações conhecidas previamente, introduzidas através de uma distribuição a priori [22]. Em suas primeiras versões, era utilizada uma distribuição a priori não-informativa, que possui apenas informações vagas. Contudo, atualmente o Infernal utiliza uma distribuição a priori informativas, a distribuição Dirichlet, pois ela torna os CMs bem mais sensíveis e específicos [23].

Nosso trabalho anterior [6] revelou que prioris Dirichlet com múltiplos componentes não são capazes de significativamente melhorar os resultados das buscas do Infernal. Contudo, isso não quer dizer que a atual priori genérica de um único componente seja o melhor que podemos fazer utilizando a distribuição Dirichlet.

Existem *clusters* de famílias de ncRNAs estruturalmente semelhantes no Rfam (Capítulo 5). Nossa hipótese é de que prioris construídas apenas para certos grupos de famílias de ncRNAs (ao invés de todo o Rfam) poderiam levar a CMs que produzam melhores resultados para as famílias desse grupo.

O problema explorado neste trabalho é aprimorar a sensibilidade e especificidade dos modelos de covariância do Infernal.

O objetivo geral deste trabalho é *tornar as buscas do Infernal com CMs mais sensíveis e específicas através de melhorias nas informações a priori utilizadas.*

Os objetivos específicos são:

1. Analisar o Rfam de forma a obter grupos de famílias que sejam estruturalmente próximas umas das outras.

2. Derivar novas prioris Dirichlet para as probabilidades de transição dos CMs utilizando informações sobre os grupos.

3. Construir um *benchmark* para cada um dos grupos.

4. Verificar, através dos *benchmarks*, o impacto na especificidade e na sensibilidade das pesquisas feitas com o Infernal usando as prioris em relação às pesquisas feitas com a priori genérica que já é embutida no Infernal.

1.1 Organização do texto

No Capítulo 2, descrevemos os RNAs não-codificadores. No Capítulo 3, descrevemos os modelos de covariância. No Capítulo 4, explicamos o que são informações a priori e distribuições Dirichlet. No Capítulo 5, analizamos as famílias Rfam de acordo com suas estruturas secundárias e encontramos *clusters* de famílias estruturalmente próximas. No Capítulo 6, criamos e analizamos prioris específicas a certos grupos (por sua vez compostos de *clusters* que encontramos). Por fim, no Capítulo 7 expomos nossas considerações finais.

Capítulo 2

RNAs não-codificadores

Neste capítulo, descrevemos as características principais de um ncRNA. Na Seção 2.1, discutimos brevemente o Dogma Central da Biologia Molecular. Na Seção 2.2, apresentamos as diferentes estruturas de um ncRNA. Na Seção 2.3, abordamos algumas classificações de ncRNAs. Na Seção 2.4, apresentamos algum métodos e ferramentas relacionados a ncRNAs. Na Seção 2.5, descrevemos alguns bancos de dados de ncRNAs disponíveis publicamente.

2.1 Dogma Central da Biologia Molecular

Antes de introduzirmos os RNAs não-codificadores *per se*, vamos definir de maneira simples "espécie" e "organismo". Não há consenso quanto a exatamente o quê é uma espécie [24], e portanto usaremos uma definição bem aceita atualmente dada por Ernst Mayr em 1942, que diz que "espécies são grupos de populações naturais que estão ou têm o potencial de estar se intercruzando, e que estão reprodutivamente isolados de outros grupos" [25]. Já organismos são "quaisquer estruturas vivas, como plantas, animais, fungos ou bactérias, capazes de crescer e se reproduzir" [26, tradução livre]. Essas definições simples serão suficientes para o nosso trabalho.

O genoma contém as informações hereditárias de um organismo. Na maioria das espécies ele é composto por várias fitas de DNA (ácido desoxirribonucleico). Cada fita de DNA pode ser entendida simplificadamente como uma sequência num alfabeto de quatro elementos: adenina (A), guanina (G), citosina (C) e timina (T). Tais elementos são denominados nucleotídeos ou bases.

Um gene é uma entidade biológica responsável por caracterizar um traço ou uma característica física de um organismo. Os genes são codificados no genoma em segmentos de DNA. As informações de um gene codificadas no DNA são utilizadas para sintetizar proteínas, longas cadeias de aminoácidos. *Grosso modo*, o DNA é transcrito para uma fita de RNA onde a timina é trocada por uracila (U), e então essa fita de RNA é traduzida para uma proteína (Dogma Central da Biologia Molecular). Esse RNA é chamado de RNA mensageiro (mRNA) ou RNA codificador, pois codifica uma proteína.

Contudo, o genoma não é composto apenas de genes. Regiões gênicas do genoma são chamadas de regiões codificadoras de proteínas. Todas as outras regiões intergênicas são chamadas de regiões não-codificadoras.

Algumas seções dessas regiões não-codificadoras são transcritas em RNAs e possuem uma função, mas esses RNAs não são traduzidos em proteínas. Tais RNAs são chamados de *RNAs não-codificadores* (ncRNAs), pois sua função não é codificar a informação de como construir uma proteína. Por muito tempo os ncRNAs, com exceção de alguns importantes e bem conhecidos como o RNA transportador (tRNA), eram chamados coletivamente de "junk DNA" pois acreditava-se que eles não tinham funções celulares importantes [27]. Ao longo dos anos foi-se descobrindo exatamente o oposto, que os ncRNAs são responsáveis pelas mais diversas funções essenciais à manutenção dos mecanismos celulares.

Quando há vários ncRNAs homólogos em diferentes espécies dizemos que eles formam uma *família de ncRNAs*. Esse é um termo recorrente neste trabalho pois cada modelo de covariância representa uma família.

2.2 Estrutura

Toda molécula possui uma organização espacial que descreve a organização dos seus átomos. São criadas várias abstrações para tratar com mais facilidade a estrutura das moléculas de RNAs não-codificadores[1]. Da mais abstrata para a mais concreta, as estruturas são:

Primária Simplesmente a sequência de bases (A, T ou U, C e G) que define a molécula. Essa estrutura é a mais usada porque:

- É a estrutura que os sequenciadores automáticos analisam e dão como resultado. Isso significa que é relativamente fácil obtê-la.
- É fácil armazená-la e tratá-la computacionalmente como uma *string*.

Secundária A estrutura secundária corresponde às ligações entre pares de bases e pode ser representada como uma lista com todas elas. É extremamente importante para o funcionamento dos ncRNAs na célula, contudo não é fácil experimentalmente obter a estrutura secundária (apesar de que computacionalmente já existem ferramentas consagradas como o RNAfold [28]).

Terciária A estrutura espacial tridimensional da molécula, definida através da especificação da posição de cada átomo.

Na estrutura secundária de ncRNAs aparecem vários padrões (*motifs*) recorrentes. A maioria dos ncRNAs pode ser descrita apenas utilizando tais padrões. São eles:

Talos e *loops* Acontece quando duas regiões do mesmo RNA se pareiam (cf. Figura 2.1). A região pareada é chamada de talo e a região da ponta é chamada de *loop*. Este padrão também é chamado de *hairpin* por se parecer com um grampo de cabelo.

Pseudo-nós Acontece quando há dois talos e a metade de um é intercalada com a metade do outro (cf. Figura 2.2). É chamada de pseudo-nó pois se parece com um nó, apesar de a fita de RNA não estar de fato atada num nó.

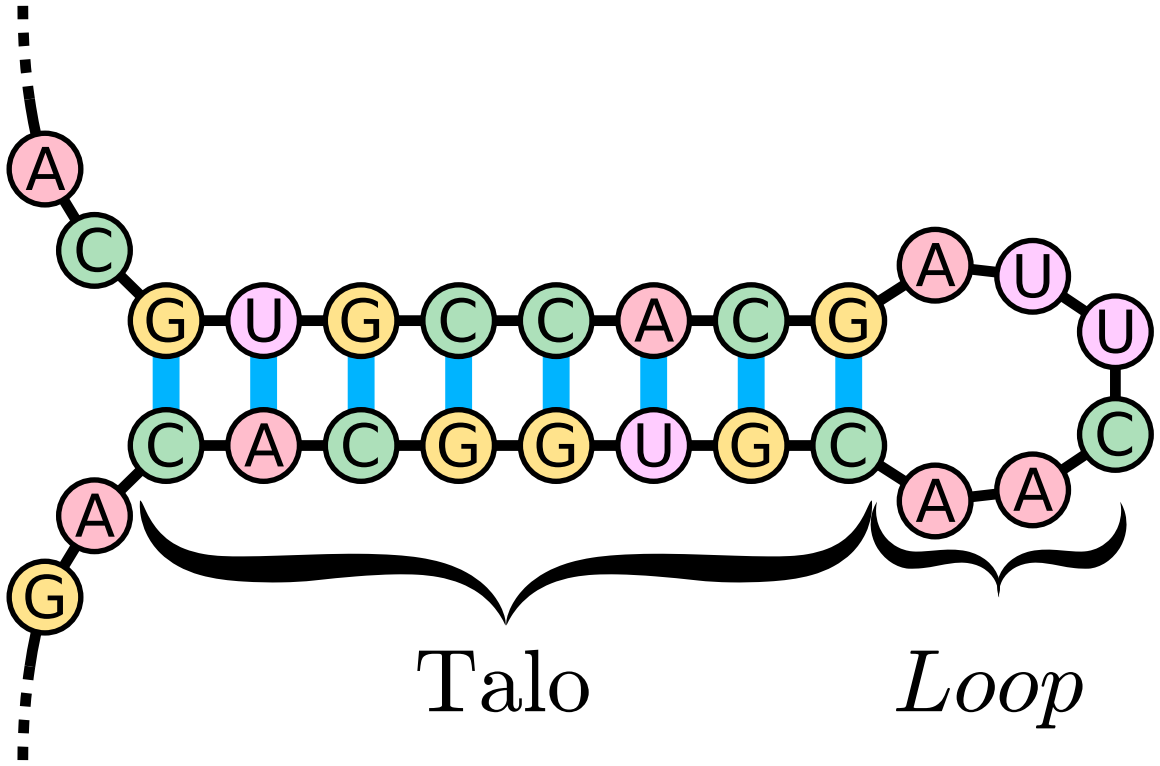


Figura 2.1: Uma estrutura talo-*loop* ou *hairpin* correspondente à estrutura primária ACGUGCCACGAUUCAACGUGGCACAG (Imagem adaptada de [1]).

Como veremos mais adiante, os modelos de covariância não são capazes de representar pseudo-nós (apesar de haverem estudos que buscam modelar pseudo-nós e ao mesmo tempo utilizar CMs, como [29]). Por isso, quanto nos referirmos a uma "estrutura secundária" em geral estaremos desprezando os pseudo-nós.

2.3 Classificação

Podemos classificar ncRNAs de acordo com suas funcionalidades, ou seja, a função que o ncRNA desempenha nos mecanismos celulares [30]. Por exemplo:

snoRNAs Os *small nucleolar RNAs* que tem como função principal guiar modificações químicas de outros ncRNAs.

microRNAs Compostos de poucos nucleotídeos, possuem diversas funções reguladoras e geralmente são compostos de apenas um *hairpin* (cf. Figura 2.1).

tRNAs Como mencionado acima, o RNA transportador, essencial no próprio processo de tradução de mRNAs para proteínas.

rRNAs RNA ribossomal, um componente do ribossomo, que por sua vez também é essencial para o processo de tradução.

[1]Outros compostos biológicos são classificados de maneira semelhante. Um exemplo importante são as proteínas. Contudo não nos preocupamos com eles neste trabalho.

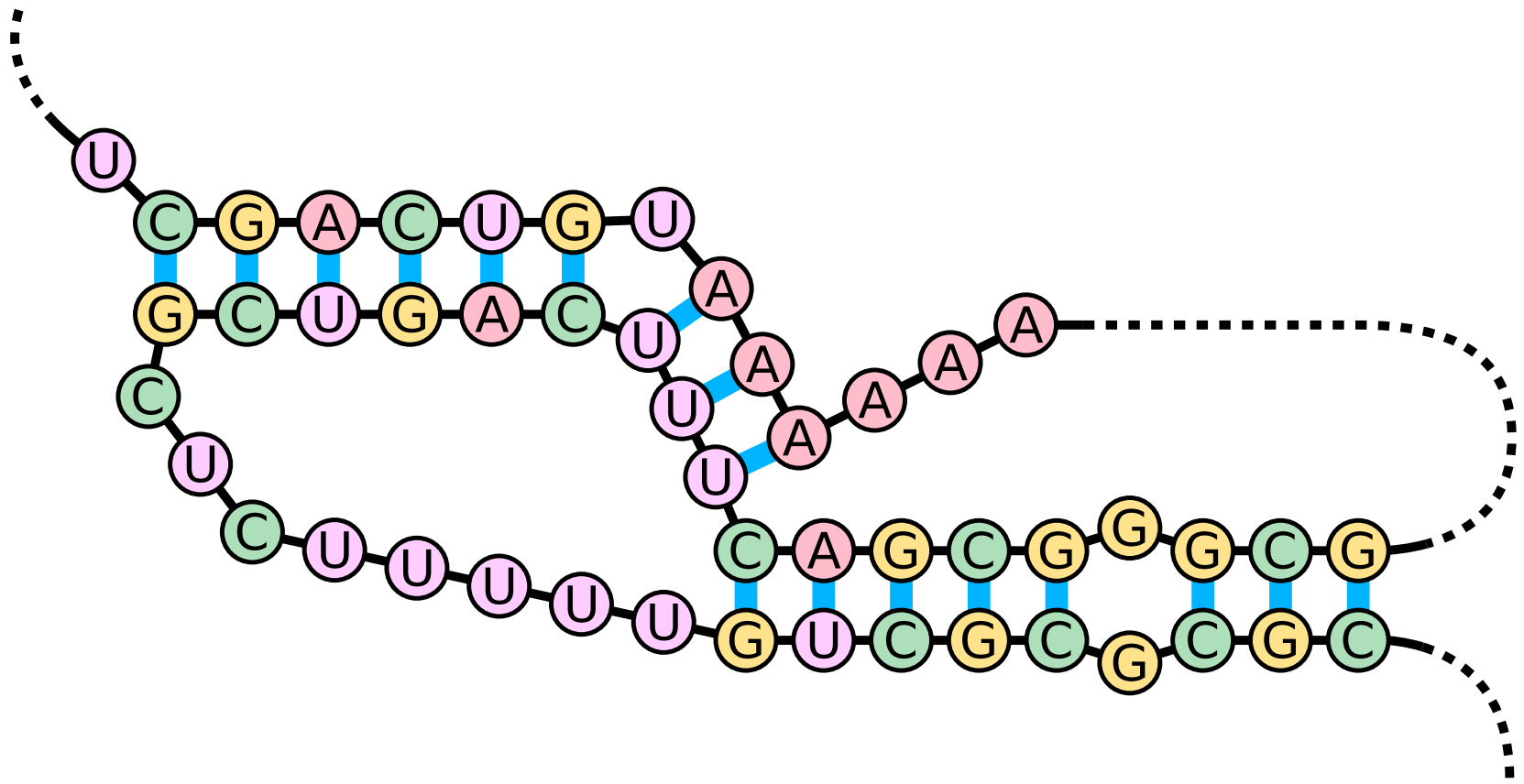


Figura 2.2: Estrutura de pseudo-nó, pois se parece com um nó apesar de a fita de RNA não estar atada. Observação: *não é representado* nos modelos de covariância pois não é possível representar tal estrutura com uma árvore. Imagem retirada de [2].

Muitas famílias de ncRNAs do Rfam têm seu nome derivado de sua função (e.g. a maioria dos microRNAs começa com "mir").

2.4 Métodos e ferramentas

Como dito antes, neste trabalho nos concentramos em métodos e ferramentas voltadas para buscas por ncRNAs. Contudo, problemas semelhantes também já alavancaram o desenvolvimento de outros tipos métodos e ferramentas: predição da estrutura secundária de uma sequência de ncRNA utilizando as leis da Termodinâmica [14, 15]; classificação de ncRNAs[2] com máquinas de vetor de suporte (*support vector machines*, SVM) como o CONC [16], o CPC [17] e o PORTRAIT [18], ou mapas auto-organizados (*self-organizing maps*, SOM) [19].

Utilizando a teoria de processos estocásticos foram já criadas várias ferramentas para buscas por homologia. Para proteínas existem as cadeias escondidas de Markov (*hidden Markov model*, HMM), onde o HMMER [31, 32, 33] em sua recém-lançada terceira versão [31] é um grande expoente e funciona para qualquer família de proteínas. Para ncRNAs existe o modelo de covariância (*covariance model*, CM; note que esse termo pode ter outras conotações em outros contextos, mas neste trabalho sempre nos referimos aos modelos de covariância de Sean Eddy), representado pelo Infernal [13]. O Infernal é um programa de busca por homologia de ncRNAs que condensa as informações de uma família de ncRNAs, representada por um alinhamento múltiplo e sua estrutura secundária, em um modelo de covariância. Um outro exemplo de ferramenta que usa modelos de covariância é o tRNAscan [34], um programa que funciona especificamente para a família tRNA mas que foi manualmente ajustado para ser rápido, sensível e específico para essa família.

Neste trabalho usaremos os CMs como nossa base. De todos os métodos gerais de busca por homologia de ncRNAs que temos conhecimento, eles atualmente são os que possuem melhor sensibilidade e especificidade.

2.5 Bancos de dados

Existem vários bancos de dados de ncRNAs [35]. Entre eles, destacamos:

Rfam [20] Banco de dados de famílias de ncRNAs (e não apenas de sequências isoladas) onde cada família possui um alinhamento múltiplo (de estruturas primárias de ncRNAs daquela família de diversos organismos de diferentes espécies) e a estrutura secundária do consenso do alinhamento múltiplo.

NONCODE [36] Banco de dados com sequências de centenas de diferentes organismos, importando dados de vários outros bancos de dados. Atualmente é o banco de dados mais completo em termos de sequências, contendo 411 mil.

RNAdb [37] Banco de dados de sequências de ncRNAs.

[2]Buscas por homologia são uma forma de classificação. Neste caso, contudo, a classificação não é específica o suficiente para concluir que o ncRNA é de uma determinada família, e portanto fazemos esta distinção de termos.

miRBase [38] Banco de dados especializado em microRNAs.

fRNAdb [39] Em vez de ser um banco de dados como os acima, o fRNAdb é uma união de vários bancos de dados em um só.

Utilizamos apenas o Rfam no nosso trabalho pois é o único banco de dados de ncRNAs organizado por famílias e que possui suas respectivas estruturas secundárias, informações essenciais para este trabalho e para o Infernal.

Capítulo 3

Modelos de covariância

Neste capítulo, estudamos os modelos de covariância (CMs). Na Seção 3.1, explicamos a motivação em usá-los. Na Seção 3.2, abordamos os principais conceitos necessários para entendê-los. Na Seção 3.3, explicamos os principais passos envolvidos na construção e no uso de um CM. Na Seção 3.4, estudamos como o cálculo do peso de cada sequência é feito. Na Seção 3.5, explicamos com mais detalhe a forma como os passos de construção são realizados.

3.1 Motivação

Os modelos de covariância foram desenvolvidos inicialmente em 1994 por Sean Eddy [12] para identificar ncRNAs. A principal ferramenta que os utiliza é o Infernal [13]. Um modelo de covariância representa uma família de ncRNAs. Ele é construído a partir de várias sequências homólogas de ncRNAs, cada uma representando a mesma família mas pertencente a um organismo diferente, todas alinhadas e anotadas com a estrutura secundária do consenso (i.e., a estrutura secundária comum a todas as estruturas primárias do alinhamento múltiplo). Note que a entrada do Infernal consiste dos dados que estão no Rfam.

A principal vantagem de utilizar modelos de covariância, em detrimento a HMMs ou algoritmos de alinhamento de sequências, é a possibilidade de modelar mais informações sobre a família de ncRNAs em questão utilizando sua estrutura secundária. Usar essas informações em geral aumenta as sensibilidade (razão entre os verdadeiros positivos e todos os apontados pelo modelo na busca por homólogos – tanto verdadeiros positivos e falsos negativos) e especificidade (razão entre os verdadeiros negativos e todos os não apontados – verdadeiros negativos e falsos positivos) das buscas por homólogos [12, 23].

O que ainda não foi esclarecido é por que um HMM não consegue representar a estrutura secundária de um ncRNA, apenas sua estrutura primária. *Grosso modo*, um HMM observa sequencialmente as bases e não tem uma "memória" para "lembrar" da estrutura que está sendo procuranda, enquanto que um CM pode olhar para a sequência em vários pontos simultaneamente devido à sua estrutura de árvore. É possível construir um HMM capaz de representar talos fielmente, basta fazer com que cada estado emita até duas bases (digamos, A e B) e, ao final do processo, concatenar as bases A com as bases B em ordem reversa. Contudo, não é possível construir um HMM que represente mais de um talo, em outras palavras, não é possível simular um nó BIF com um HMM.

3.2 Conceitos básicos

Os CMs são compostos de *estados*, *emissões* e *transições* (cf. Figura 3.1). Cada estado pode *emitir* entre zero e duas bases. Partindo de cada estado podem existir zero ou mais transições a outros estados. Existem probabilidades associadas a cada emissão ou transição.

A partir do primeiro estado, nós *transitamos* pelo CM até chegarmos a estados de onde não parta nenhuma transição. Em cada estado fazemos uma escolha quanto a que bases emitir. Esse caminho, chamado árvore de *parse* (*parse tree*), é determinado pela probabilidade das transições que foram feitas e bases que foram emitidas. Concatenando as bases emitidas podemos construir uma sequência de bases. A ideia é que as sequências de bases pertencentes à família da qual se originou o CM tenham maior probabilidade de serem observadas que as outras sequências.

Simplificadamente, o processo de construção de um CM é composto pelos seguintes passos:

1. A partir da estrutura secundária da família, constrói-se algoritmicamente[1] uma árvore-guia (cf. Figuras 3.2, 3.3 e 3.4).

2. A partir da árvore-guia constrói-se algoritmicamente um esqueleto de CM. Todos os estados, emissões e transições já estão presentes no esqueleto de CM, porém não há nenhuma probabilidade de transição.

3. São criadas árvores de *parse* falsas (*fake parse trees*) a partir da árvore-guia e do esqueleto de CM. Essas são árvores de *parse* construídas ignorando-se as probabilidades (visto que as probabilidades ainda não foram calculadas).

4. São calculadas contagens de transições e emissões (o número de vezes em que ocorreram) observadas a partir das árvores de *parse* falsas.

5. Essas contagens são combinadas com informações conhecidas a priori para derivar as probabilidades de transição e emissão e, com isso, completarmos o esqueleto de CM para obter um CM completo.

Observe pelos passos acima que a estrutura do CM é completamente determinada pela estrutura secundária da família. Os dados das sequências e as informações a priori são utilizadas única e exclusivamente para derivar as diversas probabilidades.

Para utilizar um CM, precisamos responder às seguintes perguntas: (i) Qual é a probabilidade do CM emitir esta determinada sequência? (ii) Qual é a probabilidade desta determinada sequência ser observada? Então o resultado de (i) e (ii) são comparados para se calcular uma medida de semelhança entre uma sequência qualquer e a família da qual originou-se o CM.

As informações a priori mencionadas são retratadas por meio de distribuições Dirichlet (cf. Capítulo 4). Para construir uma priori, são sintetizados dados sobre diversas famílias de ncRNAs. Mais especificamente, são geradas contagens (vide processo de construção de um CM) para todas as famílias de um determinado conjunto, e essas contagens são

[1] Neste trabalho consideramos um processo como "algorítmico" quando seu resultado não depende de nenhum processo probabilístico.

utilizadas para estimar uma priori Dirichlet. Até o momento, o Rfam completo tem sido considerado como tal conjunto [23, 6]. Neste trabalho investigamos o uso de outros conjuntos que englobem apenas determinados grupos de famílias do Rfam.

As probabilidades constituem uma das partes mais importantes para classificar o modelo em bom ou ruim. Uma melhora sensível foi observada quando prioris não-informativas (ausência de conhecimento prévio, antes da observação dos dados) foram substituídas por prioris informativas (inserção de conhecimento prévio, antes da observação dos dados) [23]. Por isso é essencial o uso de uma priori adequada.

3.3 Construção, uso e bastidores

Existem vários passos inter-relacionados nos processos de construção e uso dos CMs. Nesta seção estudaremos vários desses passos em detalhe. Sugerimos ao leitor voltar a esta seção sempre que se perder nos detalhes.

Os processos gerais podem ser separados em:

Construção Dada uma família de ncRNAs, representada por um alinhamento múltiplo de sequências e a estrutura secundária de seu consenso, este passo consiste em construir um CM correspondente. Pode ser dividido em duas etapas: uma puramente algorítmica, que constrói um tipo abstrato de dados a partir da estrutura secundária, e outra probabilística, que utiliza as informações do alinhamento múltiplo e outras já conhecidas previamente.

Uso Dado um CM e uma sequência de bases (chamada sequência-alvo), este passo consiste em realizar uma busca por homólogos na sequência-alvo ou alinhar a sequência-alvo ao CM para torná-la parte do alinhamento múltiplo da família utilizada para construir o CM.

Bastidores Este processo é o único que não é realizado frequentemente e consiste em consolidar as informações a priori utilizadas na construção de um CM.

Duas ou mais sequências formam um alinhamento múltiplo quando todas possuem o mesmo comprimento. Para isso *gaps* (espaços) são inseridos em locais apropriados. Um *gap* numa determinada posição de uma sequência indica que ela não possui a base correspondente que as outras sequências possuem. Com isso o comprimento de cada sequência passa a ser o número de bases somado ao número de *gaps*.

Os *gaps* não são inseridos arbitrariamente. Os *gaps* são relacionados à evolução das espécies e existem vários algoritmos que buscam os alinhamentos múltiplos menos intrusivos. Não abordaremos tal problema aqui, e apenas assumiremos que um alinhamento múltiplo está disponível.

Nas subseções subsequentes, os processos gerais descritos acima serão mais detalhados. Na Figura 3.6, os passos dos três processos são exibidos.

3.3.1 Construção

No Infernal, a construção dos CMs é gerenciada pelo programa `cmbuild`. Como entrada, é fornecido um arquivo contendo a representação de uma família de ncRNAs:

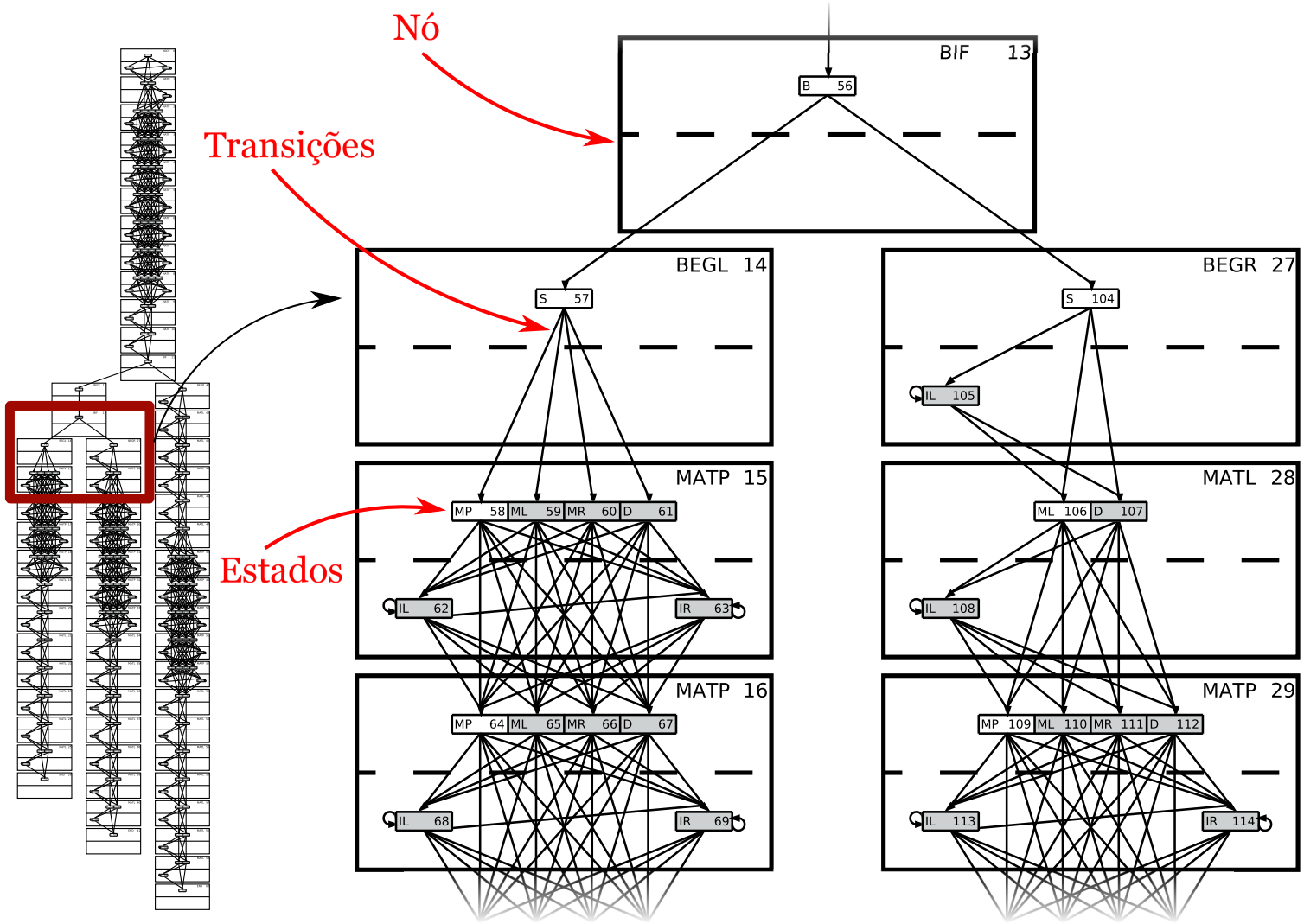


Figura 3.1: Diagrama em formato de árvore de um CM de uma família de tRNAs. À esquerda mostramos o CM completo e à direita uma ampliação de parte dele. Indicamos em vermelho o que são os nós, estados e transições (emissões não estão representadas).

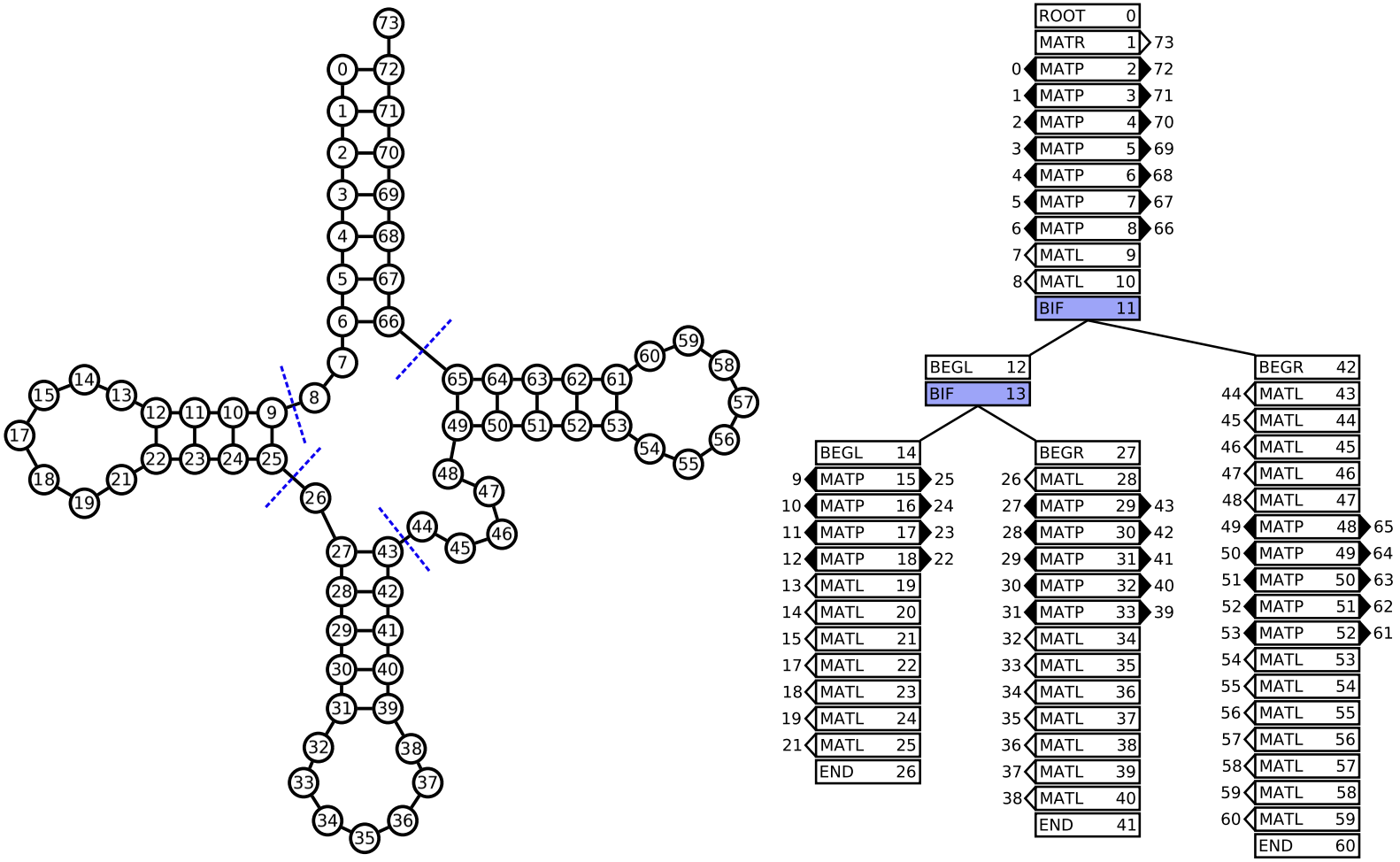


Figura 3.2: À esquerda, diagrama da estrutura secundária de uma família de tRNAs. À direita, a árvore-guia correspondente. Linhas tracejadas azuis mostram que pontos de bifurcação foram escolhidos.

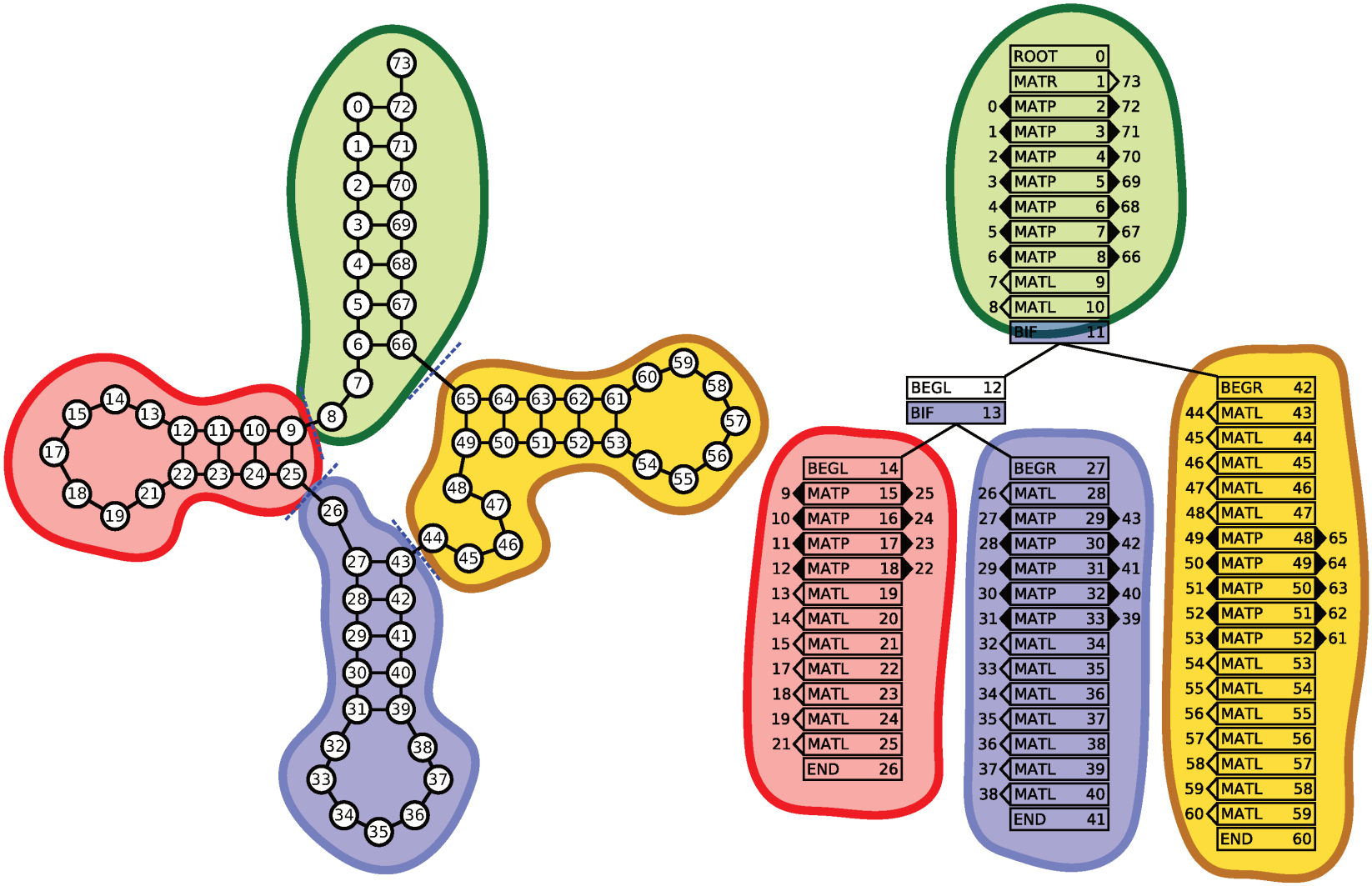


Figura 3.3: Sobreposição de balões mostrando como talos são representados na árvore-guia e como bifurcações correspondem a concatenações de talos.

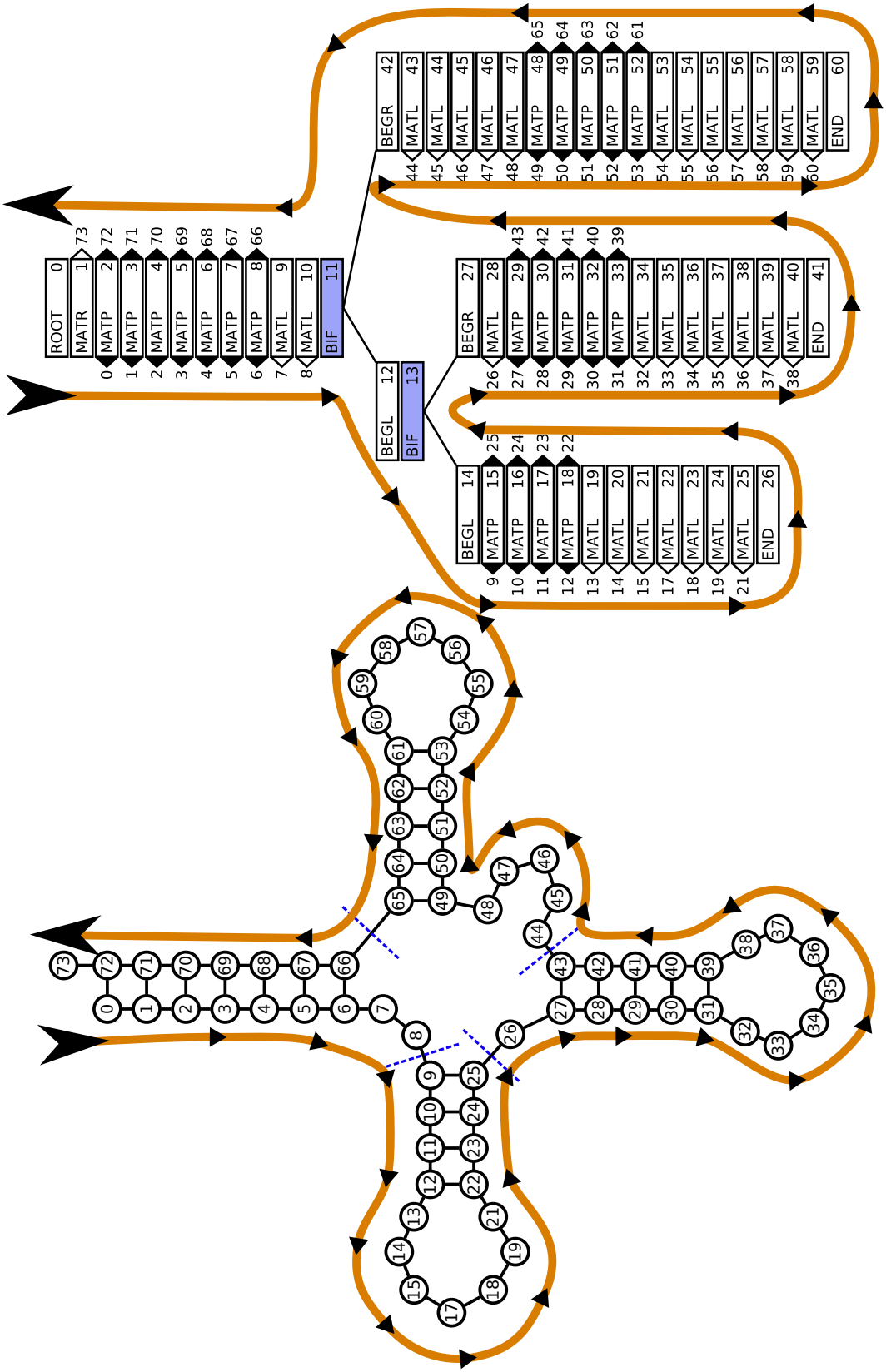


Figura 3.4: Curva mostrando como a árvore-guia é capaz de representar a estrutura primária através.

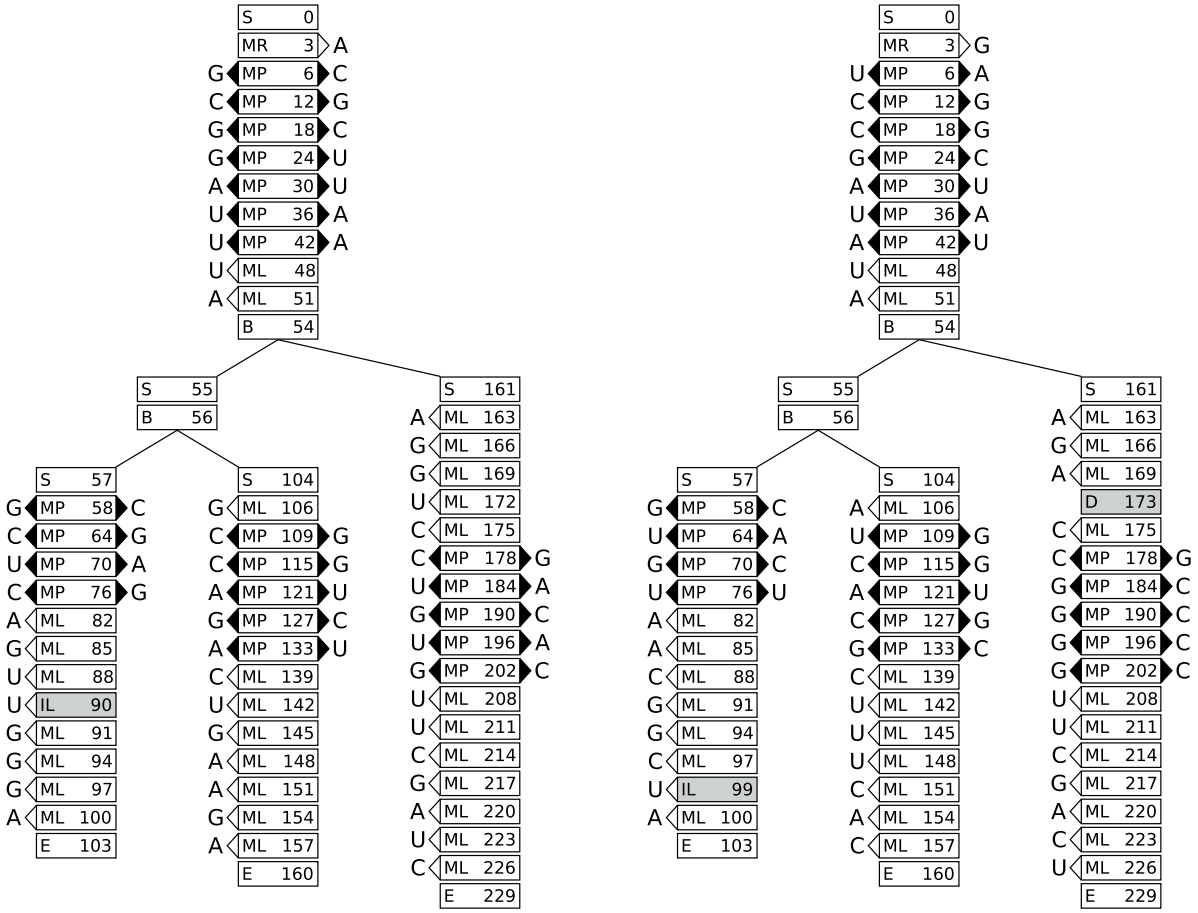


Figura 3.5: Dois exemplos de árvores de *parse*. Em particular, estas são árvores de *parse* falsas. Observe que há duas inserções (estado 90 à esquerda e estado 99 à direita) e uma remoção (estado 173 à direita) em relação ao consenso.

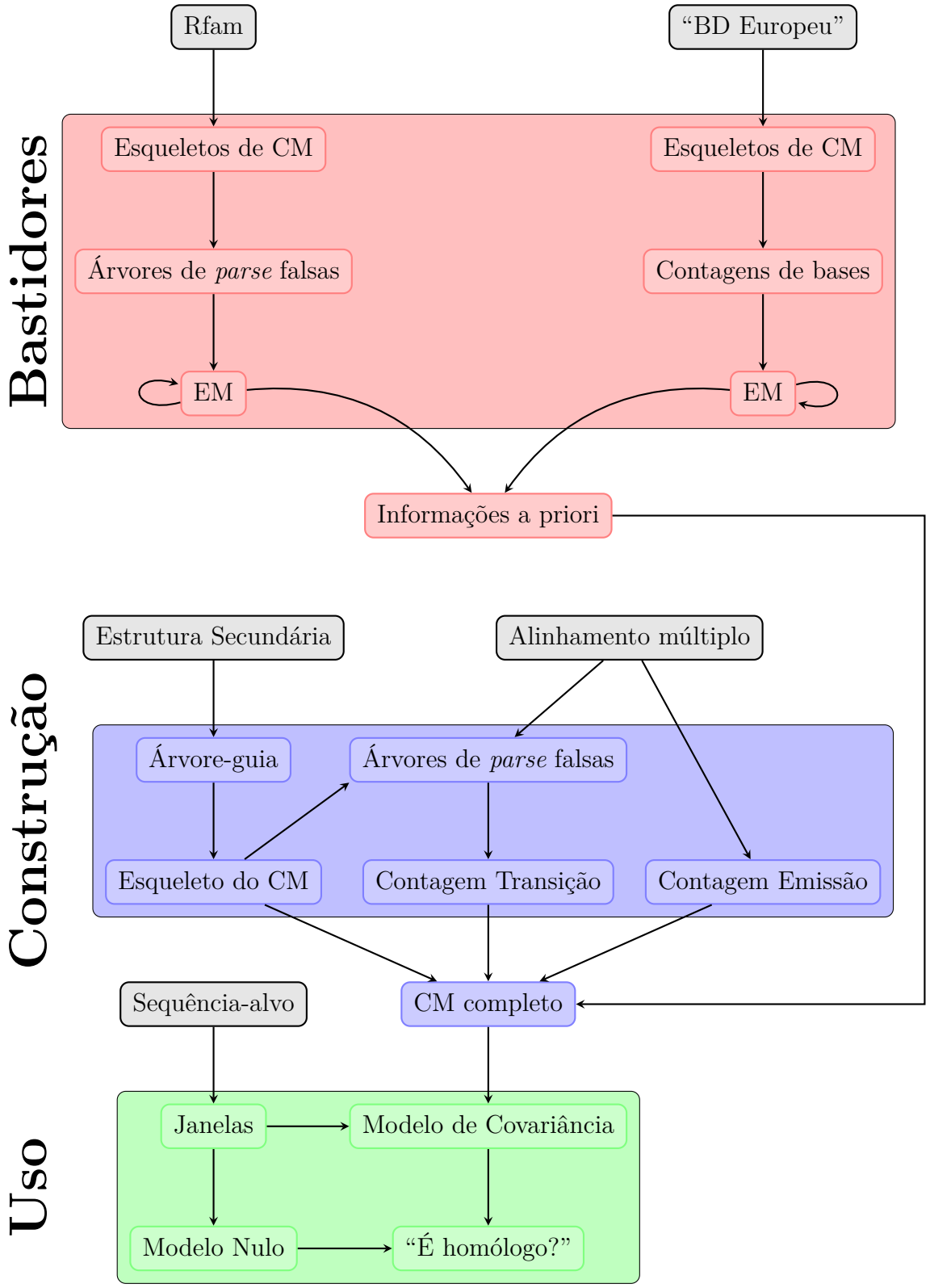


Figura 3.6: Visão geral da construção e do uso dos modelos de covariância. Dentro dos retângulos estão as estruturas mantidas em memória (ao menos conceitualmente). Fora dos retângulos estão as estruturas salvas em disco. Por "BD Europeu" nos referimos ao *European Ribosomal RNA Database* [3, 4].

um alinhamento múltiplo e a estrutura secundária do consenso no formato Stockholm. O Stockholm possui extensão `.sto` e foi concebido para uso no Infernal, no Rfam, no Pfam (banco de dados de proteínas) e outras ferramentas. Além dos dados necessários acima, um arquivo Stockholm também pode conter dezenas de tipos de metadados, sobre o alinhamento múltiplo (como os autores, referências a outros bancos de dados e palavras-chaves) ou sobre as sequências em si (como nome do organismo ou fenótipo).

O resultado esperado deste processo geral é um modelo de covariância pronto para o processo de Uso. A construção é separada nas seguintes etapas:

Estrutural Em primeiro lugar o esqueleto de CM é construído:

1. A estrutura secundária é capturada numa estrutura de dados adequada. São representadas as posições de início e fim de cada parte da estrutura secundária.
2. A partir desta estrutura de dados os pontos onde serão inseridas bifurcações são escolhidos (cf. Figuras 3.2, 3.3 e 3.4). Depois dessa escolha é criada a **árvore-guia** (cada elemento da árvore-guia é chamado de **nó** ou *node*, como nas árvores como estruturas de dados). Alguns nós podem ser omitidos se eles não fizerem parte do consenso, i.e., se mais de 50% das sequências do alinhamento contiverem um *gap* na posição correspondente.
3. A partir da árvore-guia o **esqueleto do CM** é montado. Por esqueleto nos referimos a todos os **estados** (ou **state**, um elemento do CM, assim como em um HMM) corretos e que não serão alterados e todas as possíveis transições entre eles. O que deve estar presente num CM e que não está num esqueleto de CM são as probabilidades de emissão e de transição.

Cálculo dos pesos Para cada sequência do alinhamento múltiplo é calculado um peso. Esse peso leva em conta a similaridade entre as sequências. É dado um peso menor às sequências que têm muitas outras semelhantes a elas e um peso maior àquelas que têm poucas outras semelhantes. Com isso o viés do modelo é reduzido. Todas as contagens feitas posteriormente são multiplicadas pelo peso da sequência de onde foram coletadas (e.g. se uma árvore de *parse* falsa construída a partir de uma sequência com peso 0.9 passa duas vezes por uma determinada transição, então a contagem considerada é de 1.8, e não 2.0 como seria se não houvessem pesos).

Probabilística (emissão) As probabilidades de emissão são construídas a partir do esqueleto do CM, da seguinte forma:

1. Para cada coluna do alinhamento múltiplo, uma **contagem de bases** é feita (e.g. "cinco adeninas e três uracilas observadas na coluna 42").
2. Para cada estado, as contagens das colunas correspondentes são usadas. Combinando essas contagens com as informações a priori, calculamos **probabilidades de emissão** que serão associadas ao estado.

Probabilística (transição) Os mesmos passos "observar e calcular" são usados para a transição. Contudo, observar as transições entre estados não é tão óbvio, visto que os estados não fazem parte do alinhamento múltiplo. Assim, para obter as probabilidades de transição:

1. Para cada sequência do alinhamento múltiplo é algoritmicamente gerada uma **árvore de** *parse* **falsa** (*fake parse tree*, cf. Figura 3.5). Intuitivamente, as árvores de *parse* são a projeção de uma sequência de bases no CM, o caminho de estados com maior probabilidade de ser seguido para que tal sequência seja observada. Contudo, uma árvore de *parse* falsa não representa o caminho de maior probabilidade, e sim o caminho que segue mais fielmente a árvore-guia.
2. As **contagens de transição** são obtidas a partir das árvores de *parse* falsas. Para cada tipo de transição é criado um conjunto de vetores, inicialmente com valores nulos. As árvores de *parse* falsas são percorridas e a cada transição somamos 1 à sua posição no vetor de seu tipo de transição. Não é incomum certas transições nunca serem observadas.
3. Para cada transição, as **probabilidades de transição** são calculadas levando em consideração as contagens observadas e as informações conhecidas a priori.

Todas as etapas acima são determinísticas. Por exemplo, se houverem duas escolhas de pontos de bifurcação igualmente bem balanceadas (e.g. ao unir três ramos a, b e c de mesmo tamanho podemos ter como resultado tanto $((a, b), c)$ ou $(a, (b, c))$), há sempre uma regra que determina qual deles será escolhido. Isso faz com que a geração do CM seja de fato uma função, implementada da mesma maneira tanto no Infernal quanto no Regene.

3.3.2 Uso

Existem dois usos primários para um modelo de covariância. Ambos são intimamente relacionados, porém possuem objetivos distintos:

Alinhamento O CM pode ser usado para alinhar qualquer sequência ao seu alinhamento múltiplo original. Para isso:

1. Construímos a árvore de *parse* (não-falsa) para a sequência a ser alinhada.
2. Construímos as árvores de *parse* falsas para as sequências originais do alinhamento múltiplo.
3. Simultaneamente convertemos de volta todas as árvores de *parse* construídas acima[2] para o formato de um alinhamento múltiplo. Isso é possível porque cada árvore de *parse* representa suas próprias estruturas primária e secundária, contudo todas as estruturas secundárias estão atreladas entre si pelos estados do CM que as originou. Realizar este passo com árvores de *parse* construídas por CMs distintos não seria possível.

Busca por homólogos O uso mais comum dos CMs, porém, é para realizar busca por ncRNAs homólogos. Essa busca é feita dentro de uma sequência-alvo que pode ter um comprimento arbitrariamente maior que o do o comprimento médio do CM; como o CM é um modelo probabilístico, podemos calcular a esperança do comprimento das sequências observadas. Para realizar tal busca:

[2]Uma árvore de *parse* é falsa ou não dependendo do método usado para construi-la. Depois de construídas, elas são indistinguíveis e podem até mesmo coincidir para uma determinada sequência.

1. Atravessamos o arquivo utilizando uma **janela** de tamanho apropriado. Uma janela é uma subsequência da sequência-alvo. Atravessar com uma janela de tamanho k e em passos de tamanho s significa considerar, em ordem, as subsequências de índices $[1, k], [s + 1, k + s], \ldots, [si + 1, k + si], \ldots$ até o menor i tal que $k + si >= n$, onde n é o tamanho da sequência-alvo.
2. Para cada janela, calculamos a probabilidade do CM gerar a sequência da janela (i.e., de que observemos tal sequência).
3. Para a mesma janela estimamos a probabilidade daquela sequência ter sido gerada aleatoriamente. Para isso utilizamos outro modelo probabilístico, o **modelo nulo**.
4. Utilizamos a razão entre a probabilidade de observar a sequência no CM e no modelo nulo para decidir se devemos considerar ou não a janela como um possível homólogo.

Opcionalmente poderíamos ter realizado antes dos passos 2 e 3 acima dois outros passos idênticos utilizando um HMM que procura não rejeitar aquilo que o CM aceitaria. Tal passo é chamado de um **filtro** e tem como um intuito descartar janelas que não fazem parte da família com uma alta probabilidade. O tempo de execução de um HMM é quadrático em relação ao comprimento da família, enquanto que o de um CM é cúbico.

3.3.3 Bastidores

Neste processo obtemos as prioris usadas na construção dos CMs. Este processo é transparente para os usuários de modelos de covariância, sejam eles usuários do Infernal ou apenas do Rfam.

As informações a priori procuram modelar a experiência de quem trabalha com ncRNAs. Por exemplo, imagine que, por algum motivo, os ncRNAs tivessem sempre um número par de talos. Neste passo nós deveríamos expressar essa informação. Isso poderia evitar que nossos modelos tentariam tivessem um número ímpar de talos. Para mais detalhes, veja a Seção 4.1.

Em sua concepção, o Infernal tinha um modelo bem simples de informações a priori chamado de **não-informativo** [23]. Atualmente o Infernal utiliza densidades Dirichlet para as informações sobre as transições e misturas Dirichlet para as emissões [23]. Isso significa que dezenas de parâmetros devem ser estimados.

Tais parâmetros não são intuitivos, no sentido de que não podemos determiná-los qualitativamente e de que eles nunca foram calculados e usados em outros contextos. Portanto nós mesmos devemos derivá-los de alguma forma. Os passos que são seguidos atualmente para o cálculo das densidades Dirichlet das transições são [23]:

1. As sequências-semente[3] do Rfam são obtidas para servirem como dados crus. Na nossa analogia anterior, o Rfam tem o papel da experiência do especialista.

[3]Cada família de ncRNAs cadastrada no Rfam possui um grupo de sequências chamadas de sementes ou *seeds*. Tais sequências foram analisadas manualmente e acredita-se que os dados são de alta qualidade.

2. Para cada família, é gerado um esqueleto de CM (não podemos gerar nada além de um esqueleto de CM pois um CM completo precisa das próprias prioris Dirichlet que estamos calculando). Para cada sequência-semente da família, é gerada uma árvore de *parse* falsa.

3. As contagens de transições observadas são calculadas a partir das árvores de *parse* falsas criadas acima. Nessas contagens também são levados em conta os pesos das sequências.

4. Tais contagens são usadas como dados observados em um estimador de parâmetros para as distribuições desejadas. Uma distribuição para cada tipo de transição é criada. O algoritmo EM [40], que é iterativo, geralmente é utilizado neste passo.

Passos semelhantes são seguidos para calcular as misturas Dirichlet para as informações a priori sobre as probabilidades de emissão.

O Infernal coleta todas as informações a priori acima em um único arquivo `.pri` que usa um formato interno. O programa `cmbuild` pode então usar tal arquivo em vez do padrão embutido em seu código. Utilizamos esse recurso para realizar comparações entre diferentes distribuições Dirichlet. Com isso, garantimos que a única alteração feita entre cada teste é a priori usada, pois o código que constrói os modelos e realiza as buscas é o mesmo, o do Infernal.

Nosso trabalho concentra-se nas probabilidades de transição.

3.4 Calculando o peso das sequências

Num alinhamento múltiplo, cada sequência provém de um organismo diferente. É possível que haja, por exemplo, várias sequências provenientes de organismos semelhantes e apenas uma sequência proveniente de um organismo mais distante filogeneticamente. Um modelo construído a partir desse alinhamento múltiplo teria um viés forte para as sequências que possuam mais representantes.

Para evitar esse viés, calculamos um peso para cada sequência. Os pesos são normalizados de forma que sua soma seja igual ao número de sequências do alinhamento múltiplo. Intuitivamente, é dado um peso menor que 1.0 às sequências muito representadas e um peso maior que 1.0 às sequências pouco representadas.

Utilizamos o algoritmo de Gerstein/Sonnhammer/Chothia [41] para calcular esses pesos, o mesmo utilizado por padrão pelo Infernal, que funciona da seguinte maneira:

1. Para cada par de sequências calculamos a similaridade entre elas utilizando o percentual de identidade. O percentual de identidade entre sequências alinhadas A e B é

$$s(A,B) = \frac{\left[\sum_{i=1}^{N} \delta_{A_i,B_i}\right] - g}{N - g},$$

onde N é o comprimento das sequências, A_i e B_i são o conteúdo da i-ésima coluna das sequências A e B, respectivamente, g é o número de colunas onde ambas as

sequências possuem *gaps*[4] e $\delta_{i,j}$ é o delta de Kronecker:

$$\delta_{x,y} = \begin{cases} 1, & x = y \\ 0, & x \neq y \end{cases}.$$

Note que $0 \leq s(A, B) \leq 1$ pois a soma tem valor entre g (pois as colunas onde ambas as sequências têm *gaps* são iguais) e N (pois há N termos onde cada um é 0 ou 1).

2. Construímos um dendrograma com todas as sequências utilizando o algoritmo *unweighted pair group method with arithmetic mean*, ou simplesmente UPGMA. Ele é aglomerativo, isto é, vai agrupando as sequências até que haja somente um grupo. Existem vários outros algoritmos aglomerativos que se diferem apenas na forma como a similaridade entre dendrogramas[5] é definida, porém o UPGMA é adequado quando é assumido que há uma taxa constante de evolução entre todas as espécies.

   Dado um conjunto S de sequências:

   (a) Criamos o conjunto $D = \{\text{Leaf}(x)/x \in S\}$ onde, para cada sequência dada, há um dendrograma unitário que a contém.

   (b) Enquanto houver mais de um elemento em D, i.e., $|D| > 1$, obtemos os dois dendrogramas com maior similaridade

   $$\arg\max_{\mathcal{A} \neq \mathcal{B}} \overline{s}(\mathcal{A}, \mathcal{B}),$$

   onde a similaridade entre dendrogramas $\overline{s}$ é definida como

   $$\overline{s}(\mathcal{A}, \mathcal{B}) = \frac{1}{|\mathcal{A}| \cdot |\mathcal{B}|} \sum_{x \in \mathcal{A}} \sum_{y \in \mathcal{B}} s(x, y).$$

   Substituímos então D por

   $$D' = (D/\{\mathcal{A}, \mathcal{B}\}) \cup \{\text{Branch}(\mathcal{A}, \mathcal{B})\}$$

   e repetimos este passo.

   Ao final desses passos existe apenas um elemento em D, o dendrograma completo.

3. Percorremos o dendrograma completo das folhas em direção à raiz (i.e. de baixo para cima), anotando o peso de cada sub-dendrograma w_D (não confundir com o

[4]Em um alinhamento de duas sequências temos sempre que $g = 0$. Mas se o alinhamento entre as duas sequências for retirado de um alinhamento múltiplo, é possível observer $g > 0$ (e.g., pegue as duas últimas sequências do alinhamento composto por "ABC", "-BC" e "-B-").

[5]Em geral diz-se "similaridade entre *clusters*", onde um *cluster* é simplesmente um conjunto de sequências, mas aqui estamos interessados no dendrograma completo. Preferimos então utilizar sempre o termo "dendrograma".

peso das sequências, que é nosso resultado final desejado):

$$w_D(x) = \begin{cases} 0, & x = \text{Leaf}(\cdot) \\ w_D(\mathcal{A}) + w_D(\mathcal{B}) + p(\mathcal{A}) + p(\mathcal{B}) - 2p(x), & x = \text{Branch}(\mathcal{A}, \mathcal{B}) \end{cases},$$

onde p é, intuitivamente, a "posição" do dendrograma se ele fosse desenhado, definido por

$$p(x) = \begin{cases} 1, & x = \text{Leaf}(\cdot) \\ \overline{s}(\mathcal{A}, \mathcal{B}), & x = \text{Branch}(\mathcal{A}, \mathcal{B}) \end{cases}.$$

Portanto o valor $p(\mathcal{A}) - p(x)$ é, intuitivamente, o comprimento do traço ligando a ramificação x ao seu filho $\mathcal{A}$.

4. Para cada sequência (que está numa folha), calculamos seu peso não-normalizado $w_S(x)$:

   (a) Atribuimos $\omega \leftarrow p(P_1)$ e $i \leftarrow 1$, onde P_1 é o pai de Leaf(x).

   (b) Encontramos P_{i+1}, o pai de P_i, atribuimos

   $$\omega \leftarrow \omega \cdot \left[1 + \frac{p(P_i) - p(P_{i+1})}{w_D(P_{i+1})}\right]$$
   $$i \leftarrow i + 1$$

   e repetimos este passo.

   (c) Se P_i é a raiz, então $w_S(x) = \omega$.

5. Por fim calculamos o peso normalizado $w(x)$ de cada sequência:

$$w(x) = \frac{|S|}{\sum_{y \in S} w_S(y)} \cdot w_S(x).$$

Note que se houver alguma ramificação com filhos $\mathcal{A}$ e $\mathcal{B}$ tal que $\overline{s}(\mathcal{A}, \mathcal{B}) = 1$, então a todos os filhos dessa ramificação deve ser atribuído o mesmo peso.

As complexidades de tempo e espaço dos passos acima dependem de escolhas particulares de estruturas de dados. Em nossas bibliotecas temos:

1. O cálculo das similaridades leva tempo $O(Ln^2)$, onde n é o número de sequências e L é o comprimento delas. Durante os cálculos o uso de memória é $O(1)$, mas o resultado necessita de memória $O(n^2)$ pois é uma matriz com as similaridades. Essa é uma implementação direta da definição.

2. O algoritmo UPGMA leva tempo $O(n^3)$, pois são necessários $n-1$ passos para construir o dendrograma, e em cada passo percorremos toda a matriz de similaridades em busca do par de dendrogramas a serem unidos, o que leva $O(n^2)$. O consumo de memória é $O(n^2)$, porque a entrada – uma matriz de similaridades – tem esse tamanho. É possível construir soluções mais engenhosas que levem menos tempo, mas para nossos propósitos esta é suficiente.

3. O passo de anotar os pesos leva tempo $O(n)$ e é muito simples de ser feito.

4. Já o passo de calcular os pesos leva tempo $O(n^2)$ no pior caso, porém $O(n \log n)$ em média. Para cada folha, percorremos os nós até a raiz, portanto se o dendrograma for razoavelmente balanceado então o maior caminho entre uma folha e raiz terá $O(\log n)$ nós.

5. Por fim, $O(n)$ para normalizar o resultado.

O algoritmo mais lento em geral é o UPGMA que leva tempo $O(n^3)$. No total é gasto tempo $O(Ln^2 + n^3)$ e memória $O(n^2)$.

3.5 Detalhes de implementação

Já vimos na Seção 3.3.1 os passos necessários para construir um esqueleto de CM. Nesta seção, detalharemos cada um desses passos e incluiremos referências aos trechos de código Haskell relevante. Os trechos de códigos não precisam ser lidos, pois nenhuma informação importante será perdida.

3.5.1 Estrutura secundária

O texto representando a estrutura secundária é lido para uma estrutura de dados bem simples. Nós construimos um conjunto de talos, onde cada talo é um par de partes de talo (a parte da esquerda e a da direita). Todas as colunas que não estiverem presentes em nenhum talo desse conjunto é um *singlet*. Cada parte de talo informa as posições em que começa e termina, o número de bases (i.e., final − início + 1) − *gaps*) e possui uma lista ordenada de colunas de *gaps*.

A representação no código segue exatamente a descrição acima:

```
data StemPart = StemPart Offset Offset Offset [Offset]
type Stem     = StemPart :+: StemPart
type Stems    = Set Stem
```

Os pseudo-nós são removidos utilizando um algoritmo bem simples que verifica todas as interseções entre dois talos e remove um deles (o que atravesse o maior número de outros talos) quando eles estão formando um pseudo-nó. Esse passo é repetido até que não haja nenhum pseudo-nó. A lista de interseções pode ser construída em tempo $O(n^2)$ e encontrar o talo a remover leva tempo $O(n)$, portanto o algoritmo leva tempo $O(n^2)$.

Convertemos o conjunto de talos para uma representação em forma de uma floresta de talos. Nós construimos nossa floresta de forma que (a) nós irmãos sejam talos paralelos e estejam em ordem crescente e (b) nós filhos sejam talos aninhados. Note que só é possível criar essa representação se não houverem pseudo-nós. Como a estrutura de conjuntos `Set` mantém todos os talos em ordem numa árvore binária de pesquisa, é fácil construir um algoritmo de tempo $O(n \log n)$:

1. Se o conjunto S de talos for vazio, retorne a floresta vazia.

2. Tome o talo t mais à esquerda do conjunto S de talos. Seja $S' = S/\{t\}$ o conjunto dos talos restantes.

3. Particione o conjunto $S/\{t\}$ em dois conjuntos S_l e S_r onde S_l possui todos os talos que estão à direita da posição mais à direita de t e $S_r = S'/S_l$. Este passo pode ser executado em tempo logarítmo.

4. Execute recursivamente o algoritmo nos conjuntos S_l e S_r. Como resposta são obtidas as florestas de talos F_l e F_r.

5. Construa a árvore T com t na raiz e F_l como filhos.

6. Retorne a floresta $F = \{T\} \cup F_r$.

É fácil ver que os passos acima de fato calculam uma árvore de talos como desejamos, e também é fácil ver como que não poderíamos separar em S_l e S_r na presença de pseudo-nós. Esse algoritmo é implementado na função `stemsToTree`.

3.5.2 Pontos de bifurcação

Um ponto de bifurcação é uma coluna onde deverá ser colocado um nó BIF. Em geral, tentamos escolher os pontos de bifurcação de forma a deixar os ramos da árvore-guia com um número balanceado de nós. Esse comportamento é o mesmo do Infernal.

Note que se um talo está aninhado em outro então não há necessidade de fazer uma bifurcação, basta colocar os nós do talo de fora e depois os nós do talo de dentro. A bifurcação só é necessária para separar talos paralelos. Nesse sentido a bifurcação é equivalente a uma concatenação, já que com apenas MATL, MATR e MATP nós podemos descrever qualquer tipo de aninhamento de talos não-paralelos.

Para decidir onde os pontos de bifurcação serão colocados, a função `findBifPoints` percorre a floresta de talos dividindo sempre o mais no meio possível. O seu resultado é uma outra árvore, dessa vez com cada nó tendo nenhum, um ou dois filhos, assim como a árvore-guia que procuramos. A árvore de bifurcações tem nenhum filho quando há apenas *singlets*, tem um filho quando há um talo e o filho representa todos os talos que estão aninhados nele, ou tem dois filhos quando há um ponto de bifurcação, i.e., dois ou mais um talos paralelos que precisam ser concatenados com uma bifurcação.

A estrutura de dados usada no código também inclui, no caso de um ponto de bifurcação, informações sobre o talo mais à esquerda e sobre o ponto de bifurcação. Essas informações facilitam o desenho do algoritmo da próxima subseção.

```
data BifPoints = Nested !Stem BifPoints
               | BifPoint { bpStart :: !Offset
                          , bpEnd   :: !Offset
                          , bpBifP  :: !Offset
                          , bpLeft  :: BifPoints
                          , bpRight :: BifPoints }
               | NoStems
```

Note que não há talos (`Stem`) em `BifPoint`, apenas em `Nested`.

3.5.3 Árvore-guia

Construir a árvore guia a partir da árvore de bifurcações é simples. Essa estrutura intermediária existe apenas para separar a tarefa de achar os pontos de bifurcação da

tarefa de construir um nó por coluna em duas funções distintas fáceis de entender. Essa divisão é útil principalmente porque apesar de construir a árvore-guia ser simples, há varios casos a serem considerados.

Para construir os nós, o algoritmo chama-se recursivamente para cada nó da árvore de bifurcações, sempre se lembrando das posições iniciais e finais que devem ser preenchidas naquele passo (e.g. na primeira chamada todas as posições devem ser preenchidas, quando um ponto de bifurcação é encontrado a chamada recursiva da esquerda deve preencher a parte da esquerda, enquanto a chamada recursiva de direita deve preencher o que sobrar, etc.). Para cada talo são inseridos MATPs, enquanto que entre todas as partes são inseridos MATLs e MATRs (lembrando que representamos apenas os talos, deixando os *singlets* implícitos).

Porém devem ser levadas em conta as informações de colunas consideradas inserções. Então, por exemplo, se em um ponto devia ser inserido um MATP mas a coluna da esquerda, da direita, ou ambas são inserções, então nenhum nó é inserido. É possível que ramos inteiros da árvore desapareçam! Apesar de improvável na prática, no final "simplificamos" as árvores apagando bifurcações que não tenham nós em ambos os filhos.

3.5.4 Árvores de *parse* falsas

O processo de construção de uma árvore de *parse* falsa é semelhante ao de construção da árvore-guia pois ambos essencialmente apenas percorrem uma estrutura enquanto criam outra, porém ambos têm que tratar diversos casos.

Para construir uma árvore de *parse* falsa basta percorrer a árvore-guia tomando nota de:

1. Colunas sem *gaps* que estão representadas por MATL, MATR ou MATP, e portanto são do consenso.
2. Colunas sem *gaps* não-representadas, portanto inserções (em relação ao consenso).
3. Colunas com *gaps* representadas na árvore-guia, portanto remoções (em relação ao consenso).
4. Colunas com *gaps* não-represetandas, portanto parte do consenso também.

Além disso, deve ser tomado algum cuidado quando uma das colunas de um MATP é *gap* e a outra não.

Assim que descobrimos quais colunas são parte do consenso, basta escolher os nós correspondentes do CM.

3.5.5 Modelo de covariância

Já o próprio CM é representado como um simples vetor de estados. Dentro de cada estado armazenamos as informações sobre suas emissões e transições. Emissões são representadas como vetores de probabilidade cujas chaves são números inteiros (e.g. de 1 a 4) ou pares de números inteiros, dependendo se o estado é MP ou não. Transições são representadas como vetores de probabilidade cujas chaves são índices de estados.

Como em Haskell há tipos algébricos, podemos especificar precisamente quais estados têm que tipos de informação associada:

Tabela 3.1: Correspondência entre nós e estados.

Nó	*Split set*	*Insert set*
MATP	MP, ML, MR, D	IL, IR
MATL	ML, D	IL
MATR	MR, D	IR
BEGL	S	
BEGR	S	IL
ROOT	S	IL, IR
BIF	B	
END	E	

```
data State = MP !PairEmissions   !Transitions
           | ML !SingleEmissions !Transitions
           | MR !SingleEmissions !Transitions
           | IL !SingleEmissions !Transitions
           | IR !SingleEmissions !Transitions
           | S  !Transitions
           | D  !Transitions
           | B  !StateNum !StateNum
           | E
```

Para construir o CM a partir da árvore-guia, basta seguir a lista de quais estados estão presentes em quais nós. Para cada tipo de nó de uma árvore-guia, sempre são criados os mesmos tipos de estado no CM correspondente como na Tabela 3.1. Em cada linha, a coluna "Nó" indica o nome do tipo de nó da árvore-guia, e as colunas *"Split set"* e *"Insert set"* representam os estados criados no CM a partir de cada tipo de nó.

Capítulo 4

Informações conhecidas a priori

Neste capítulo, descreveremos a motivação e o uso das distribuições Dirichlet para codificar informações conhecidas a priori. Na Seção 4.1, apresentamos os conceitos básicos. Na Seção 4.2, definimos a distribuição Dirichlet. Na Seção 4.3, estudamos a estimação dos parâmetros de uma distribuição Dirichlet. Na Seção 4.4, apresentamos o algoritmo de otimização `CG_DESCENT`. Na Seção 4.5, descrevemos com mais detalhes a interação da distribuição a priori com os CMs.

4.1 Conceitos básicos

Conforme estudado no Capítulo 3, de cada estado de um CM partem várias transições[1] para outros estados. Cada possível transição tem um *estado de origem* (de onde parte a transição) e um *estado de destino* (que pode ser o mesmo estado de origem ou um outro estado).

Vamos assumir que a partir de um dado estado de origem partam m possíveis transições. A probabilidade da i-ésima transição é denotada por $0 \leq p_i \leq 1$ tal que $\sum_{i=1}^{m} p_i = 1$. Para construir um CM completo a partir de um esqueleto de CM, precisamos conhecer o vetor aleatório $\vec{p} = (p_1, \ldots, p_m)$ de todos os possíveis estados de origem.

Vamos assumir que há r famílias de ncRNA do Rfam (e.g. em sua versão 10.1, o Rfam possui $r_{10.1} = 1973$). A partir de cada família, podemos obter um esqueleto de CM e várias árvores de *parse* falsas (uma por sequência do alinhamento múltiplo da família). Fixado um certo grupo de transição (cf. Seção 4.5), podemos observar nas árvores de *parse* falsas de uma família todas as transições do grupo fixado.

Seja $\vec{N} = (N_1, \ldots, N_m)$ um *vetor aleatório de contagens de transições*, i.e., N_i é o número de vezes que a i-ésima transição ocorreu. A partir de cada uma das r famílias de ncRNAs, podemos obter uma amostra de $\vec{N}$ da maneira descrita acima (note que há uma observação por família, e não por árvore de *parse* falsa). Assumimos então que $\vec{N}$ segue uma distribuição multinomial com parâmetro $\vec{p} = (p_1, \ldots, p_m)$.

Para estimar $\vec{p}$, utilizaremos amostras observadas de $\vec{N}$ e informações conhecidas a priori atráves de inferência Bayesiana. O princípio desta teoria é de combinar informações conhecidas por meio da distribuição a priori, com a informação obtida da amostra (dados

[1] O foco deste trabalho será apenas nas probabilidades de transições. Contudo, o tratamento das probabilidades de emissão é análogo.

observados), via função de verossimilhança, para construir a distribuição posteriori que sintetiza todas as informações.

Assumimos, portanto, que $\vec{p}$ segue uma distribuição Dirichlet com parâmetros $\vec{\alpha} = (\alpha_1, \ldots, \alpha_m)$ (cf. Definição 1).

Dados então $\vec{\alpha}$ e $\vec{N}$, podemos estimar $\vec{p}$ a partir da posteriori. Assumindo a distribuição a priori como sendo uma Dirichlet, e sendo esta uma distribuição conjugada, segue que a distribuição a posteriori também é uma Dirichlet com parâmetros $\vec{\alpha} + \vec{N}$. Daí, temos a seguinte estrutura:

$$\begin{aligned} \vec{p} &\sim \text{Dirichlet}(\vec{\alpha}) && \text{(priori)} \\ \vec{N} \mid \vec{p} &\sim \text{Multinomial}(\vec{p}) && \text{(verossimilhança)} \\ \vec{p} \mid \vec{N} &\sim \text{Dirichlet}(\vec{\alpha} + \vec{N}) && \text{(posteriori)} \end{aligned}$$

Neste trabalho, quando usamos o termo "uma priori" estamos nos referindo na verdade a um conjunto de parâmetros de diferentes distribuições Dirichlet, uma por grupo de transição. Nosso interesse então é estimar $\vec{\alpha}$, os parâmetros da distribuição a priori, de forma a obtermos valores $\vec{p}$ que melhorem a especificidade e a sensibilidade dos CMs.

A distribuição Dirichlet é utilizada neste trabalho, portanto, por dois motivos: em primeiro lugar porque o vetor aleatório envolvido é constituído por probabilidades, e em segundo lugar por ser a Dirichlet uma distribuição conjugada.

4.2 A distribuição Dirichlet

Sjölander et al. [42] utilizaram distribuições Dirichlet para estimar distribuições de aminoácidos e descreveram como utilizá-las em qualquer contexto. Descreveremos neste trabalho apenas os principais pontos, e direcionamos o leitor ao trabalho de Sjölander et al. para mais detalhes.

Definição 1. A *função de densidade da distribuição Dirichlet* é uma função do vetor de probabilidades $\vec{p}$ com parâmetros $\vec{\alpha} = (\alpha_1, \ldots, \alpha_m)$ com $\alpha_i > 0$ dada por

$$\rho(\vec{p}) = \frac{1}{Z} \prod_{i=1}^{m} p_i^{\alpha_i - 1},$$

onde Z é a constante que faz f_ρ integrar para um.

Dada uma densidade Dirichlet ρ, a esperança de p_i é

$$E(p_i) = \frac{\alpha_i}{\alpha_0},$$

onde $\alpha_0 = \sum_{i=1}^{m} \alpha_i$, e o segundo momento é dado por

$$E(p_i p_j) = \begin{cases} \dfrac{\alpha_i(\alpha_i + 1)}{\alpha_0(\alpha_0 + 1)}, & \text{se } i = j \\ \dfrac{\alpha_j \alpha_i}{\alpha_0(\alpha_0 + 1)}, & \text{se } i \neq j \end{cases}.$$

A probabilidade estimada $\hat{p}_i$ é a estimativa média a posteriori dada por

$$\hat{p}_i = E(p_i \mid \vec{N} = \vec{n}) = \frac{n_i + \alpha_i}{n_0 + \alpha_0}$$

onde $\vec{n} = (n_1, \ldots, n_m)$ é um valor observado de $\vec{N}$ e $n_0 = \sum_{i=1}^{m} n_i$.

4.2.1 Misturas Dirichlet

Ao invés de uma densidade Dirichlet, podemos utilizar como distribuição a priori uma mistura Dirichlet.

Definição 2. Dizemos que a distribuição de p é uma mistura finita de distribuições Dirichlet, se a função densidade de p é dada por

$$\rho(p) = q_1\rho_1(p) + q_2\rho_2(p) + \cdots + q_l\rho_l(p),$$

onde l é o número de componentes da mistura, ρ_i é a i-ésima densidade Dirichlet componente da mistura, com parâmetro $\vec{\alpha}_i = (\alpha_{i,1}, \ldots, \alpha_{i,m})$, e q_i é o i-ésimo peso positivo onde $\sum_{i=1}^{n} q_i = 1$. Denotamos por

$$\vec{\theta} = (\vec{\alpha}_1, \ldots, \vec{\alpha}_l, q_1, \ldots, q_l)$$

o vetor com todos os parâmetros de ρ.

Note que se $l = 1$, então a mistura tem apenas um componente de peso unitário e é equivalente a uma densidade Dirichlet.

No caso de uma mistura, o valor esperado de p_i é dado por

$$E(p_i) = \sum_{j=1}^{l} q_j \frac{\alpha_{j,i}}{\alpha_{j,0}},$$

onde $\alpha_{j,0} = \sum_{i=1}^{m} \alpha_{j,i}$. Nesse caso, a probabilidade estimada $\hat{p}_i$ é dada por

$$\hat{p}_i = \sum_{j=1}^{l} P(\vec{\alpha}_j \mid \vec{N} = \vec{n}) \frac{\alpha_{j,i} + n_i}{\alpha_{j,0} + n_0},$$

onde $P(\vec{\alpha}_j \mid \vec{N} = \vec{n}) \neq q_j$ apesar de o outro fator ser igual à estimativa média posterior do j-ésimo componente. Na verdade,

$$P(\vec{\alpha}_j \mid \vec{N} = \vec{n}) = \frac{q_j P(\vec{N} = \vec{n} \mid \vec{\alpha}_j, n_0)}{P(\vec{N} = \vec{n} \mid \vec{\theta}, n_0)}$$

onde

$$P(\vec{N} = \vec{n} \mid \vec{\theta}, n_0) = \sum_{i=1}^{l} q_i P(\vec{N} = \vec{n} \mid \vec{\alpha}_i, n_0)$$

$$P(\vec{N} = \vec{n} \mid \vec{\alpha}_j, n_0) = \frac{\Gamma(n_0 + 1)\Gamma(\vec{\alpha}_{j,0})}{\Gamma(n_0 + \vec{\alpha_{j,0}})} \prod_{i=1}^{m} \frac{\Gamma(n_i + \alpha_{j,i})}{\Gamma(n_i + 1)\Gamma(\alpha_{j,i})}.$$

A principal vantagem em utilizar uma mistura Dirichlet com $l > 1$ é ser capaz de modelar dados mais heterogêneos. Como nosso trabalho anterior revelou que o ganho em utilizar misturas Dirichlet com $r = 1372$ não é significativo [6], neste trabalho focamos exclusivamente em densidades Dirichlet. Apesar disso, nas próximas seções consideraremos o caso geral de uma mistura com $l \geq 1$ componentes.

4.3 Estimação

Introduzimos conhecimento a priori sobre $\vec{p}$ através da distribuição ρ, portanto é necessário obter informação sobre $\vec{\theta}$. Usamos o estimador de máxima verossimilhança de $\vec{\theta}$ para isso.

São observados os vetores de contagem $\vec{n}_1, \ldots, \vec{n}_r$, onde $\vec{n}_i = (n_{i,1}, \ldots, n_{i,m})$. Neste trabalho, esses r vetores são obtidos a partir do banco de dados Rfam como explicado anteriormente. Assumimos que o seguinte processo foi utilizado para obter independentemente tais vetores [42]:

1. Um componente j da mistura é escolhido aleatoriamente de acordo com os pesos q_j.
2. Um vetor de parâmetros $\vec{p}$ é escolhido independentemente de acordo com $P(\vec{p} \mid \vec{\alpha}_j)$.
3. Um vetor de contagens $\vec{n}_i$ é gerado de acordo com a distribuição multinomial de parâmetro $\vec{p}$.

Desejamos então encontrar $\vec{\theta}$ que maximiza a probabilidade de ocorrência desses vetores $\vec{n}_i$. Como cada vetor foi observado independentemente, a probabilidade de observar todos é o produto da probabilidade de observar cada um. Portanto procuramos o modelo que maximiza $\prod_{t=1}^{r} P(\vec{N} = \vec{n}_t \mid \vec{\theta}, n_{t,0})$, onde $n_{t,0} = \sum_{i=1}^{m} n_{t,i}$. Equivalentemente, queremos $\vec{\theta}$ que minimiza a *função objetiva*

$$f(\vec{\theta}) = -\sum_{t=1}^{m} \log P(\vec{n}_t \mid \vec{\theta}, n_{t,0})$$

Para a minimização utilizamos o algoritmo *expectation maximization* (EM) [40]. O EM é um método iterativo para encontrar estimativas de parâmetros em modelos estatísticos através da maximização da verossimilhança.

Os parâmetros $\vec{\alpha}_j$ e os parâmetros q_j são estimados separadamente em um processo de dois estágios que é iterado até que um mínimo local seja encontrado ou o número máximo de iterações seja atingido:

1. Os parâmetros $\vec{\alpha}_j$ são estimados mantendo-se os parâmetros q_j fixos. Várias iterações são realizadas até os valores se estabilizarem.

2. Os parâmetros q_j são estimados mantendo-se os parâmetros $\vec{\alpha}_j$ fixos, e então volta-se para o passo anterior. Não é necessário usar iterações neste passo pois há uma forma fechada para a melhor otimização. Note que este passo é desnecessário se houver apenas um componente.

4.3.1 Estimação dos parâmetros $\vec{\alpha}_j$

Como os valores $\alpha_{j,i}$ são estritamente positivos, fazemos uma reparametrização tomando $\alpha_{j,i} = e^{w_{j,i}}$, onde $w_{j,i}$ é qualquer número real. Assim, a derivada parcial da função objetiva com respeito a $w_{j,i}$ é

$$\frac{\partial f(\vec{\theta})}{\partial w_{j,i}} = -\alpha_{j,i} \sum_{t=1}^{r} P(\vec{\alpha}_j \mid \vec{N} = \vec{n}_t, \vec{\theta}) \left[\begin{array}{c} \Psi(\alpha_{j,0}) - \Psi(n_{t,0} + \alpha_{j,0}) + \\ +\Psi(n_{t,i} + \alpha_{j,i}) - \Psi(\alpha_{j,i}) \end{array} \right]$$

onde $\Psi(\cdot) = \frac{\Gamma'(\cdot)}{\Gamma(\cdot)}$ é a função digama. Através do gradiente da função objetiva o `CG_DESCENT` é usado para realizar este passo da estimação.

4.3.2 Estimação dos parâmetros q_j

Ao contrário dos parâmetros $\alpha_{j,i}$, podemos analiticamente minimizar a função objetiva em relação aos parâmetros q_j, bastando calcular

$$q_i' = \frac{1}{m} \sum_{t=1}^{r} P(\vec{\alpha}_i \mid \vec{n}_t, \vec{\theta})$$

onde os q_i' são os novos valores de q_i após este passo.

4.4 Método não-linear do gradiente conjugado

Minimizar os parâmetros $\vec{\alpha}_j$ analiticamente é difícil dada a complexidade do gradiente da função objetiva (cf. Seção 4.3.1). Utilizamos então técnicas iterativas para buscar seu valor mínimo.

Existe uma grande quantidade de técnicas diferentes de otimização. Sjölander et al. sugerem um método de gradiente descendente [42], enquanto Nawrocki utilizou um método não-linear de gradiente conjugado com a fórmula de Polak-Ribière [23].

Os vários métodos não-lineares de gradiente conjugado derivam do (único) método (linear) de gradiente conjugado, ou simplesmente CG. O CG é o método mais popular para resolver grandes sistemas lineares de equações pois pode trabalhar facilmente com matrizes esparsas [43]. Ele é iterativo e encontra a solução exata de um sistema linear de n equações em exatamente n passos.

Para minimizar funções não-lineares, o CG pode ser adaptado de várias maneiras, como usando a fórmula de Fletcher-Reeves ou usando a fórmula de Polak-Ribière [44, 43].

Usamos o método não-linear de gradiente conjugado desenvolvido por Hager e Zhang chamado `CG_DESCENT` [44]. Esse método relativamente novo foi escolhido por dois motivos: primeiro, porque ele tem um bom desempenho em termos de números de iterações necessárias para encontrar o mínimo, o que é superior a outros métodos não-lineares de gradiente conjugado; segundo, porque os autores disponibilizam a biblioteca `CG_DESCENT` 3.0, uma implementação robusta em linguagem C e bem testada do método que criaram, com uma licença livre.

4.5 Detalhes de implementação

No início deste capítulo definimos $\vec{p}$ e $\vec{N}$ utilizando CMs como motivação. Contudo, tudo que foi discutido até o momento neste capítulo pode ser igualmente aplicado a qualquer outro problema. Nossa biblioteca `statistics-dirichlet` [45] implementa o processo de estimação acima e pode ser usada fora do contexto dos CMs. Há, porém, vários detalhes que serão discutidos nesta seção sobre como aplicar este conhecimento aos CMs.

4.5.1 Grupos de transição

Em primeiro lugar devemos salientar que não existe *uma distribuição Dirichlet* para as probabilidades de transição, e sim um conjunto de *74 distribuições Dirichlet*, uma para cada grupo de transição. Neste trabalho, porém, abusamos da linguagem e nos referimos ao conjunto de todas as distribuições Dirichlet simplesmente como "a priori Dirichlet".

Cada transição num CM tem um *estado de origem* e um *estado de destino*. Além disso, esses estados foram criados a partir de um nó na árvore-guia, e portanto dizemos que cada transição num CM também tem um *nó de origem* e um *nó de destino*. Esses nós podem ser o mesmo (por exemplo, de um estado ML para um estado IL dentro de um nó MATL) ou podem ser dois nós consecutivos na árvore-guia (por exemplo, de um estado D de um nó MATL para um estado MP de um nó MATP). A Figura 3.1 (p. 13) ilustra ambos os casos. Nós dizemos que o *nó sucessor* é o nó que sucede o nó de origem, ou seja, todas as transições partindo de um certo nó podem apenas ou continuar no mesmo nó, ou ir para o nó sucessor.

Cada *grupo de transição* representa todas as transições possíveis *quando fixamos o nó de origem, o estado de origem e o nó sucessor*. Por exemplo, um possível grupo de transição é o "BEGR/S → MATL" e as transições presentes nesse grupo são, em ordem:

1. "BEGR/S → BEGR/IL" (do *insert set* de BEGR);
2. "BEGR/S → MATL/ML" (do *split set* de MATL);
3. "BEGR/S → MATL/D" (do *split set* de MATL).

Note que a primeira transição da lista acima tem como *nó de destino* o mesmo *nó de origem* BEGR, porém seu *nó sucessor* ainda é MATL. Para a segunda e terceira transições, seus *nós de origem* são MATL, assim como seus *nós sucessores*. Devido a esse fato de que transições para estados do *insert set* mantém-se no mesmo nó (por definição), optamos por usar o termo "nó sucessor" e evitar confundir o significado do grupo de transição.

Apesar de existirem oito estados possíveis, veja que no grupo de transição "BEGR/S → MATL" existem apenas três possibilidades. O número de transições possíveis varia de grupo para grupo, sendo de duas (e.g. "MATP/IR → END" tem como estados de destino possíveis IR – voltando para o mesmo estado – e E do próximo nó) a seis transições possíveis (e.g. "MATP/ML → MATP" tem como estados de destino possíveis IL, IR, MP, ML, MR e D).

O número mágico de 74 grupos de transição surge quando consideramos que combinações de nó/estado/nó podem surgir em CMs com base na forma como eles são construídos [46, p. 73][2]. Nem todo nó gera todos os possíveis tipos de estados, e portanto não há $8 \times 8 = 64$ possíveis combinações de nó/estado de origem, e sim apenas 20 (cf. Tabela 3.1, p. 28). Além disso, os nós END, BIF, BEGL, BEGR e ROOT são especiais: ou não há transições a partir de um desses nós (END), ou não há transições para ele (ROOT), ou seu nó predecessor já é pré-fixado (BEGL e BEGR) ou a partir dele não existem transições no sentido probabilístico (BIF); portanto não há $20 \times 8 = 160$, e sim 74 possibilidades.

4.5.2 Agrupamentos

Basicamente, para cada grupo de transição há uma mistura Dirichlet associada. Essa mistura é usada para calcular as estimativas a posteriori das transições que pertencem a esse grupo.

Na prática, certos grupos de transição são pouco observados e por isso é feito um *agrupamento* de grupos de transição. Existir um agrupamento entre grupos de transição $G_1, \ldots, G_k$ significa que ao invés de estimarmos uma mistura Dirichlet para cada grupo de transição, estimamos uma única mistura Dirichlet para todos eles. Os vetores de contagem usados para essa estimação são todos os vetores de contagem observados para qualquer um dos grupos de transição. A mistura Dirichlet obtida é então replicada para todos os grupos de transição. Esses grupos de transição podem ser agrupados juntos apenas se eles forem compatíveis dois a dois.

Definição 3. Dois grupos de transição G_i e G_j são ditos *compatíveis* se, e somente se:

- Ambos tiverem o mesmo número n de possíveis transições.
- Para todo $1 \leq k \leq n$, o k-ésimo estado de destino da transição G_i é similar ao k-ésimo estado de destino da transição G_j.

Definição 4. Dois estados e_1 e e_2 são ditos *similares* se alguma das seguintes condições ocorrer:

- Se $e_1 = e_2$, ou seja, eles são o mesmo estado.
- Se $e_1 = \text{IL}$ e $e_2 = \text{IR}$ ou vice-versa.
- Se $e_1 = \text{B}$ e $e_2 = \text{E}$ ou vice-versa.

Note que esta similaridade é uma relação de equivalência.

Não conhecemos uma caracterização sobre que tipos de agrupamento são possíveis. Porém essa definição é intuitiva e o agrupamento usado por Nawrocki a satisfaz.

[2]Note que há um erro no artigo de 2007 de Nawrocki [23], onde é mencionado que há 73 e não 74 grupos de transição. Nawrocki esqueceu-se do grupo de transição "BEGL/S → BIF".

4.5.3 Vetores de contagem

A última questão a ser abordada é a forma como obtemos os vetores de contagem para realizar a estimação. Na Seção 3.3.3 já foram dadas informações sobre este processo porém sem detalhes sobre as misturas Dirichlet.

Relembrando, o número r de vetores de contagens é o número de famílias presentes no Rfam usado como entrada, e.g. $r_{10.1} = 1973$ para o Rfam 10.1, $r_{9.1} = 1372$ para o Rfam 9.1 ou $r_{6.1} = 379$ para o Rfam 6.1. Além disso, existem 74 grupos de transição. Então para cada grupo de transição existem r vetores de contagem, um por família. Para cada família, existem 74 vetores de contagem, um por grupo de transição, cada um usado em uma priori diferente.

O vetor de contagem de um grupo de transição G da i-ésima família é obtido da seguinte maneira:

1. Os pesos de cada sequência-semente são calculados (cf. Seção 3.4).

2. A árvore-guia, o esqueleto de CM e uma árvores de *parse* por sequência-semente são construídos (cf. Seção 3.5).

3. Para cada árvore de *parse* falsa, calculamos um vetor de contagem da seguinte maneira:

   (a) Para cada transição que se encaixa no grupo de transição G presente na árvore de *parse* falsa, é incrementado em 1 o valor da contagem correspondente ao estado de destino tomado nessa transição em particular.

   (b) Os valores finais são multiplicados pelo peso da sequência-semente de onde esta árvore de *parse* falsa foi criada.

4. Os vetores de contagem de todas as árvores de *parse* falsas são somados para se obter um único vetor de contagem. Este vetor é o vetor de contagem do grupo de transição G para esta i-ésima família.

Todos os passos até a construção das árvores de *parse* falsas podem ser compartilhados para todos os grupos de transição, sem haver necessidade de recalculá-los.

Chapter 5

Clustering Rfam's ncRNA families

In this chapter, we analyze Rfam's ncRNA families with respect to their structural similarity. In Section 5.1, we present our published article. In Section 5.2, we give a more detailed description of `rfam-dendrogram`, the tool we created for this work.

5.1 Main content

Our work has been published on Genes 2012. Please see the article included in Annex I.

5.2 Implementation details

We built a tool named `rfam-dendrogram`. While developing it, we used many libraries that I previously developed, such as `hierarchical-clustering` [47] and `gsc-weighting` [48]. However, we also developed two new libraries: `biostockholm` [49], which parses and pretty prints Stockholm files (based on `Regene` [6] but updated, faster and more general), and `hierarchical-clustering-diagrams` [50], which draws dendrograms of any kind of leaves.

The `rfam-dendrogram` tool has many modes of operation. Some of them did not end up being used on the final results presented in Annex I but were used while developing this work. The modes of operation are:

`distances` Takes as input the Rfam database and outputs a distance matrix of all of its families. Some metadata about families is also preserved to be used on later steps. Calls RNAdistance in order to calculate the distance between each pair of families.

`dendrogram` Takes as input `distances`'s output and outpus a dendrogram (in an internal format). The user may choose between single linkage, complete linkage and UPGMA. It's also possible to restrict the input to the families that match a user-specified regular expression.

`draw` Takes as input `dendrogram`'s output and outputs an image representing the dendrogram. Has many useful and interesting options (and a few boring ones), such as:

`--gsc` Uses the Gerstein-Sonnhammer-Cothia algorithm [51] to calculate a weight for each family. Each family's width is then multiplied by its weight.

`--color` Allows you to color groups of families that match a given regular expression.

`--clans` Mark Rfam's clans on the image. Its main use is knowing which clans were well-preserved and which were not.

`--make-clusters` One of the most complex options. Highlights all subdendrograms that contain a given family (which is a leaf). Also computes statistics about which families of a user-specified (via a regular expression) group is included on each subdendrogram.

Besides drawing an image, `draw` may also output a Newick file. Despite being extremely limited, it may be imported in many standalone dendrogram viewers, such as TreeViewX [52].

`classify` Tries to find taxonomic information about Rfam's families on EBI's database [53]. Due to some technical difficulties in accessing EBI's data, this mode usually is able to correctly find out this information only for less than 70% of Rfam's families. Takes as input both Rfam's data and a list of taxons (by default, phyla).

`matrix` Takes as input `classify`'s output and outputs a graphical representation of a matrix of families versus taxons.

`distribution` Takes as input `distances`'s output and outputs a plot of the kernel density estimation of the pairwise family distance distribution. Allows the user to restrict the input using a regular expression as well.

`outliers` Takes as input `dendrogram`'s output and outputs an ordered list of Rfam's families. Families that are farther away from all others (i.e., outliers) are closer to the top.

`clanstats` Takes as input both `dendrogram`'s output and a clan list. Computes α, β and γ for all clans (cf. Annex I). The output may be either a plot or the raw data.

All images may be saved on either PNG, PS, PDF or SVG file formats.

No intermediate file format is documented; all of them are considered strictly internal to `rfam-dendrogram`. The serialization of all intermediate files is done automatically. In order to document these files we would need to start manually saving/loading all intermediate files while not enjoying any advantage in the near future. These file formats may be changed if need arises, though.

Those different modes of operation could be all done in a single step by `rfam-dendrogram`. However, it is a lot more convenient to separate them. For example, you do not need to recalculate all family distances just to change a color in the resulting image.

Besides the source code, `rfam-dendrogram`'s package also has many shell scripts that reproduce *all data* used on Annex I just by calling `rfam-dendrogram`. We hope that makes it as easy as possible to follow our steps.

Chapter 6

Group-specific priors

On this chapter we create two groups of ncRNA families. For each of them we (a) derive a prior, (b) construct a benchmark, and (b) test it against Infernal's default prior. On Section 6.1, we describe how these groups, priors and benchmarks were created. On Section 6.2, we present our results.

6.1 Data

As explained on Chapter 5, we found six disjoint clusters of structurally similar ncRNA families on Rfam (cf. Table 6.1). Three clusters (SNORD1, SNORD2 and SNORA) were had mostly snoRNAs, two clusters (miRNA1 and miRNA2) has mostly microRNAs and finally a last cluster (CRISPR) has mostly CRISPRs. All of them also have some other ncRNA families that are not of the types described but still are structurally similar to rest.

Given that we need as many ncRNA families as possible in order to derive good priors, we created two groups using five of the six clusters:

snoRNA Composed of clusters SNORD1, SNORD2 and SNORA. Contains 578 ncRNA families (cf. Table 6.2).

microRNA Composed of clusters miRNA1 and miRNA2. Contains 517 ncRNA families (cf. Table 6.2)

These two groups of ncRNA families represent 55.5% of all Rfam 10.1 families.

Our prior work showed that priors with more than one component (using all Rfam families) do not give significantly better results than priors with a single component [6]. Given that we have a smaller number of ncRNA families on our groups (578 and 517 vs. 1371 at the time) and that our groups have a smaller variability in secondary structure (since they're structurally similar), we focus on priors with a single component.

In order to derive priors for our groups, we adapted the Regene tool [6] and added the ability to derive priors of subsets of Rfam's data (instead of using the whole Rfam). One prior was created for each group.

As it's difficult to assess whether a given prior is better than another one (for the same reasons that it's difficult to create good priors in the first place), we resorted to benchmarks in order to find out if our priors gave better results than Infernal's default

prior. Infernal's source code already includes a benchmark dubbed "rmark-1" but it contains only a handful snoRNAs and microRNAs, rendering it useless for our purposes. However, since version 1.1rc1, Infernal's source code also includes a tool able to create new benchmarks. Using it we created an snoRNA benchmark and a microRNA benchmark.

On Tables 6.4 and 6.5 we present which families were selected for the snoRNA and the microRNA benchmarks, respectively. For each family we also show some details about its sequences (i.e., primary structures):

- The "Avg. % id." columns show the average percent identity among sequence of the training set alignment.
- The "Length" columns show the ncRNA families' multiple alignment length in columns.
- The "Count" columns show the number of sequences in the ncRNA family.
- The "Fragments" columns show the number of sequences that were excluded because they were considered fragments (i.e., their length was less than 70% of the average length).
- The "Training" columns show the number of sequences that were selected to be used as training sequences. These sequences are the ones used to create the test CMs.
- The "Test" columns show the number of sequences that were selected to be used as test sequences. These sequences were embedded on a 10 Mbp pseudo-chromosome.

There are families on Tables 6.4 and 6.5 that are not snoRNAs or microRNAs, respectively. They were included because they were part of the clusters used to make these groups, which in turn is a consequence of being structurally very similar to other snoRNAs or microRNAs.

The benchmark consists in building and calibrating a CM for each test family and for each prior (either one of our priors or Infernal's default prior). Searches are then conducted using the CMs on the pseudo-chromosome. Given that we know where the test sequences are on the pseudo-chromosome, we know which results these searches should give.

6.2 Priors and benchmarks

We were able to successfully derive most Dirichlet priors. For the snoRNA group, we were able to successfully derive 66 of all 74 Dirichlet priors. For the microRNA group, we were able to successfully derive 64 of all 74 Dirichlet priors. Some of these priors weren't successfully derived because either there wasn't any data for their type of transition or because `CG_DESCENT` diverged. These unsuccessfully derived priors are then assumed to be a Dirichlet density with parameters $\vec{\alpha} = (1/m, 1/m, \ldots, 1/m)$ where m is the number of possible transitions. The types of transition that were not successfully derived are listed on Table 6.6.

On Figures 6.1 and 6.2 we present the results of our benchmarks. They are presented both as a MER value and as a ROC curve.

Table 6.1: Clusters of snoRNAs, miRNAs, and CRISPRs.

Cluster	Number of ncRNA families included (family statistics)
SNORD1	334 (94.9% are SNORDs, contains 74.4% of all SNORDs)
SNORD2	86 (81.4% are SNORDs, contains 16.4% of all SNORDs)
SNORA	158 (81.0% are SNORAs, contains 57.1% of all SNORAs)
miRNA1	45 (86.6% are plant microRNAs, contains 43.6% of all of them)
miRNA2	472 (85.6% are animal microRNAs, contains 91.6% of all of them)
CRISPR	100 (59.0% are CRISPRs, contains 90.8% of all CRISPRs)

The MER value is the minimum error rate, the minimum sum of false positives and true negatives at any score threshold. MER values are non-negative and the best possible MER is zero.

The ROC curve shows the trade-off between specificity and sensibility. The horizontal axis is the number of errors per query, which means that on the left we have a high specificity and on the right we have a low specificity. The vertical axis is the proportion of positives that are found, which means that on the bottom we have a low sensibility and on the top we have a high sensibility. We also show as dotted curves the bootstrapped confidence intervals of the ROC curves.

Taking into account only the MER values and the ROC curves, we could say that both of our priors were slightly better than Infernal's default prior. However, these results are not statistically significant: on both benchmarks, both ROC curves land on each others confidence interval. Also, note that the confidence interval is rather large for all ROC curves. Our gut feeling is that there were too few test sequences on our benchmarks.

While we're confident that group-specific priors can lead to better results than a generic priors for some families, we still need more tests in order to be sure.

Table 6.2: List of ncRNA families included on group "snoRNA" (578 families).

23S-methyl, ACA64, Afu_190, Afu_191, Afu_198, Afu_199, Afu_254, Afu_264, Afu_294, Afu_298, Afu_300, Afu_304, Afu_335, Afu_455, Afu_513, Afu_514, Alpha_RBS, Bacillus-plasmid, CAESAR, CC1840, CC3552, CDKN2B-AS, DdR1, DdR10, DdR11, DdR12, DdR13, DdR14, DdR15, DdR16, DdR17, DdR18, DdR2, DdR4, DdR5, DdR6, DdR7, DdR8, GlmY_tke1, HIV_GSL3, HLE, HOTAIR_3, HSR-omega_2, Hammerhead_1, Hammerhead_3, Lnt, MALAT1, MIR478, NrrF, OxyS, PRINS, PhotoRC-II, PreQ1, PrfA, Pxr, RMST_4, RUF2, RatA, RsaA, RsaE, S15, SAH_riboswitch, SAM_V, SAM_alpha, SCARNA11, SCARNA14, SCARNA15, SCARNA18, SCARNA20, SCARNA21, SCARNA23, SCARNA3, SCARNA4, SCARNA8, SMK_box_riboswitch, SNORA1, SNORA11, SNORA13, SNORA14, SNORA15, SNORA17, SNORA18, SNORA19, SNORA20, SNORA21, SNORA22, SNORA23, SNORA24, SNORA25, SNORA28, SNORA29, SNORA3, SNORA30, SNORA31, SNORA32, SNORA33, SNORA35, SNORA36, SNORA38, SNORA4, SNORA40, SNORA41, SNORA42, SNORA43, SNORA44, SNORA46, SNORA47, SNORA48, SNORA49, SNORA5, SNORA50, SNORA51, SNORA52, SNORA54, SNORA55, SNORA56, SNORA57, SNORA58, SNORA61, SNORA62, SNORA63, SNORA64, SNORA65, SNORA66, SNORA67, SNORA68, SNORA69, SNORA7, SNORA70, SNORA71, SNORA72, SNORA76, SNORA77, SNORA79, SNORA8, SNORA84, SNORA9, SNORD100, SNORD101, SNORD102, SNORD103, SNORD105, SNORD107, SNORD108, SNORD109A, SNORD11, SNORD110, SNORD111, SNORD112, SNORD113, SNORD115, SNORD116, SNORD11B, SNORD12, SNORD121A, SNORD123, SNORD124, SNORD125, SNORD126, SNORD127, SNORD14, SNORD15, SNORD18, SNORD19, SNORD19B, SNORD20, SNORD21, SNORD22, SNORD23, SNORD24, SNORD25, SNORD26, SNORD27, SNORD28, SNORD29, SNORD30, SNORD31, SNORD33, SNORD34, SNORD35, SNORD36, SNORD37, SNORD38, SNORD39, SNORD41, SNORD42, SNORD43, SNORD44, SNORD45, SNORD46, SNORD47, SNORD48, SNORD49, SNORD5, SNORD50, SNORD51, SNORD52, SNORD53_SNORD92, SNORD56, SNORD57, SNORD58, SNORD59, SNORD60, SNORD61, SNORD62, SNORD63, SNORD65, SNORD66, SNORD69, SNORD70, SNORD72, SNORD73, SNORD74, SNORD75, SNORD78, SNORD79, SNORD81, SNORD82, SNORD83, SNORD87, SNORD88, SNORD91, SNORD93, SNORD94, SNORD95, SNORD96, SNORD98, SNORD99, SNORND104, S_pombe_snR10, S_pombe_snR3, S_pombe_snR33, S_pombe_snR35, S_pombe_snR42, S_pombe_snR46, S_pombe_snR5, S_pombe_snR90, S_pombe_snR92, S_pombe_snR93, Spot_42, Termite-leu, U54, U6atac, VrrA, bxd_3, bxd_4, c-di-GMP-II, ceN100, ceN101, ceN102, ceN103, ceN104, ceN105, ceN106, ceN108, ceN109, ceN110, ceN111, ceN113, ceN114, ceN125, ceN126, ceN22, ceN27, ceN28, ceN30, ceN33, ceN36-1, ceN38, ceN39, ceN40, ceN41, ceN42, ceN43, ceN44, ceN45, ceN46, ceN47, ceN48, ceN49, ceN51, ceN53, ceN54, ceN58, ceN61, ceN63, ceN65, ceN67, ceN68, ceN69, ceN70, ceN80, ceN81, ceN82, ceN84, ceN86, ceN88, ceN89, ceN92, ceN93, frnS, greA, msr, plasmodium_snoR14, plasmodium_snoR16, plasmodium_snoR17, plasmodium_snoR20, plasmodium_snoR21, plasmodium_snoR24, plasmodium_snoR26, plasmodium_snoR27, plasmodium_snoR28, plasmodium_snoR30, rimP, rli27, rli49, rli61, sR1, sR10, sR11, sR12, sR13, sR14, sR15, sR16, sR17, sR18, sR19, sR2, sR20, sR21, sR22, sR23, sR24, sR28, sR3, sR30, sR32, sR33, sR34, sR35, sR36, sR38, sR39, sR4, sR40, sR41, sR42, sR43, sR44, sR45, sR47, sR48, sR49, sR5, sR51, sR52, sR53, sR55, sR58, sR60, sR7, sR8, sR9, sn1185, sn1502, sn2317, sn2343, sn2417, sn2429, sn2524, sn2841, sn2903, sn2991, sn3060, sn3071, sn668, snR13, snR161, snR189, snR3, snR33, snR39, snR39B, snR40, snR41, snR47, snR49, snR5, snR50, snR51, snR52, snR56, snR58, snR62, snR65, snR67, snR68, snR73, snR75, snR76, snR77, snR78, snR79, snR80, snR81, snR85, snR87, snR9, snoCD11, snoF1_F2, snoJ26, snoJ33, snoMBII-202, snoMe18S-Gm1358, snoMe18S-Um1356, snoMe28S-Am2589, snoMe28S-Am2634, snoMe28S-Am982, snoMe28S-Cm2645, snoMe28S-Cm3227, snoMe28S-Cm788, snoMe28S-G3255, snoMe28S-Gm1083, snoMe28S-Gm3113, snoMe28S-U3344, snoPyro_CD, snoR01, snoR03, snoR07, snoR09, snoR1, snoR10, snoR101, snoR104, snoR11, snoR111, snoR113, snoR114, snoR116, snoR117, snoR118, snoR12, snoR121, snoR126, snoR127, snoR128, snoR13, snoR130, snoR134, snoR137, snoR14, snoR16, snoR160, snoR17, snoR18, snoR19, snoR20, snoR20a, snoR21, snoR22, snoR23, snoR24, snoR25, snoR26, snoR27, snoR28, snoR29, snoR30, snoR31, snoR31_Z110_Z27, snoR32_R81, snoR35, snoR38, snoR4, snoR41, snoR43, snoR44_J54, snoR4a, snoR53Y, snoR60, snoR639, snoR64, snoR64a, snoR66, snoR69Y, snoR71, snoR72, snoR72Y, snoR74, snoR77, snoR77Y, snoR79, snoR80, snoR8a, snoR9_plant, snoTBR5, snoTBR7, snoU105B, snoU109, snoU13, snoU18, snoU30, snoU31b, snoU36a, snoU40, snoU43, snoU43C, snoU49, snoU54, snoU6-47, snoU6-53, snoU6-77, snoU61, snoU82P, snoU83, snoU83B, snoU83C, snoU83D, snoZ101, snoZ102_R77, snoZ103, snoZ105, snoZ107_R87, snoZ118, snoZ119, snoZ122, snoZ13_snr52, snoZ152, snoZ155, snoZ157, snoZ159, snoZ161_228, snoZ162, snoZ163, snoZ165, snoZ168, snoZ169, snoZ17, snoZ173, snoZ175, snoZ182, snoZ185, snoZ188, snoZ196, snoZ199, snoZ206, snoZ221_snoR21b, snoZ223, snoZ247, snoZ256, snoZ266, snoZ267, snoZ278, snoZ30, snoZ30a, snoZ40, snoZ43, snoZ5, snoZ6, snoZ7, snopsi18S-1854, snopsi18S-841, snopsi28S-1192, snopsi28S-2876, snopsi28S-3316, snopsi28S-3327, snosnR48, snosnR54, snosnR55, snosnR57, snosnR60_Z15, snosnR61, snosnR64, snosnR66, snosnR69, snosnR71, v-snoRNA-1.

Table 6.3: List of ncRNA families included on group "microRNA" (517 families).

Actino-pnp, Bacteria_small_SRP, Corona_package, Deinococcus_Y_RNA, F6, GABA3, Gurken, HBV, HHBV_epsilon, K_chan_RES, L21_leader, Lacto-usp, Lambda_thermo, MIR1023, MIR1027, MIR1122, MIR1222, MIR1428, MIR1444, MIR1446, MIR158, MIR162_2, MIR168, MIR169_5, MIR171_1, MIR171_2, MIR1846, MIR2118, MIR2587, MIR390, MIR394, MIR397, MIR398, MIR403, MIR405, MIR408, MIR439, MIR444, MIR473, MIR474, MIR475, MIR476, MIR477, MIR480, MIR529, MIR530, MIR535, MIR812, MIR815, MIR828, MIR845_1, PYLIS_1, Parecho_CRE, RNA-OUT, Retro_dr1, Rota_CRE, SECIS_2, SNORD86, SNORD90, TB10Cs2H1, TB10Cs5H2, TB10Cs5H3, TB11Cs4H3, TB9Cs1H2, Termite-flg, Tombus_IRE, Vault, Xist_exon4, Y_RNA, bantam, ceN23-1, ceN74-2, ciona-mir-92, g2, hvt-mir-H, lactis-plasmid, let-7, lin-4, lsy-6, mir-1, mir-10, mir-101, mir-103, mir-105, mir-11, mir-1178, mir-1180, mir-1183, mir-12, mir-1207, mir-1208, mir-122, mir-1224, mir-1226, mir-1227, mir-1237, mir-1249, mir-1251, mir-1253, mir-1255, mir-126, mir-1265, mir-127, mir-1275, mir-128, mir-1280, mir-1287, mir-129, mir-1296, mir-130, mir-1307, mir-133, mir-134, mir-135, mir-136, mir-137, mir-138, mir-1388, mir-139, mir-14, mir-140, mir-142, mir-143, mir-144, mir-145, mir-146, mir-147, mir-1473, mir-148, mir-149, mir-150, mir-153, mir-154, mir-155, mir-156, mir-16, mir-160, mir-17, mir-172, mir-181, mir-182, mir-1827, mir-1829, mir-183, mir-184, mir-185, mir-186, mir-187, mir-188, mir-19, mir-190, mir-191, mir-1912, mir-192, mir-193, mir-194, mir-196, mir-197, mir-198, mir-199, mir-2, mir-200, mir-202, mir-2024, mir-203, mir-204, mir-205, mir-207, mir-208, mir-21, mir-210, mir-214, mir-216, mir-217, mir-218, mir-219, mir-22, mir-221, mir-223, mir-2238, mir-224, mir-2241, mir-228, mir-23, mir-230, mir-231, mir-232, mir-233, mir-234, mir-235, mir-239, mir-24, mir-240, mir-241, mir-242, mir-244, mir-245, mir-246, mir-248, mir-249, mir-25, mir-250, mir-251, mir-2518, mir-253, mir-254, mir-255, mir-259, mir-26, mir-263, mir-268, mir-27, mir-274, mir-275, mir-276, mir-277, mir-2774, mir-2778, mir-278, mir-279, mir-28, mir-280, mir-2807, mir-281, mir-282, mir-283, mir-284, mir-286, mir-287, mir-288, mir-289, mir-29, mir-290, mir-296, mir-298, mir-299, mir-3, mir-30, mir-3017, mir-302, mir-304, mir-305, mir-306, mir-308, mir-314, mir-315, mir-316, mir-317, mir-3179, mir-318, mir-3180, mir-32, mir-320, mir-322, mir-324, mir-326, mir-328, mir-33, mir-330, mir-331, mir-335, mir-337, mir-338, mir-339, mir-34, mir-340, mir-342, mir-344, mir-345, mir-346, mir-350, mir-351, mir-353, mir-354, mir-355, mir-357, mir-358, mir-359, mir-36, mir-360, mir-361, mir-363, mir-365, mir-367, mir-370, mir-374, mir-375, mir-378, mir-383, mir-384, mir-392, mir-395, mir-399, mir-412, mir-42, mir-422, mir-423, mir-425, mir-43, mir-431, mir-432, mir-433, mir-434, mir-44, mir-448, mir-449, mir-450, mir-451, mir-452, mir-454, mir-455, mir-456, mir-458, mir-46, mir-460, mir-463, mir-471, mir-48, mir-484, mir-486, mir-488, mir-489, mir-49, mir-490, mir-491, mir-492, mir-497, mir-498, mir-499, mir-5, mir-50, mir-500, mir-503, mir-504, mir-505, mir-506, mir-52, mir-540, mir-541, mir-542, mir-544, mir-548, mir-549, mir-55, mir-550, mir-551, mir-552, mir-553, mir-556, mir-557, mir-558, mir-562, mir-563, mir-567, mir-569, mir-572, mir-573, mir-574, mir-575, mir-576, mir-577, mir-578, mir-58, mir-580, mir-581, mir-582, mir-583, mir-584, mir-586, mir-589, mir-590, mir-592, mir-593, mir-597, mir-598, mir-599, mir-6, mir-60, mir-600, mir-604, mir-605, mir-607, mir-609, mir-61, mir-612, mir-615, mir-616, mir-618, mir-62, mir-621, mir-624, mir-625, mir-626, mir-628, mir-63, mir-631, mir-632, mir-633, mir-636, mir-639, mir-64, mir-640, mir-642, mir-643, mir-644, mir-648, mir-649, mir-650, mir-651, mir-652, mir-653, mir-654, mir-657, mir-661, mir-662, mir-663, mir-665, mir-668, mir-67, mir-671, mir-672, mir-673, mir-674, mir-675, mir-676, mir-684, mir-689, mir-692, mir-7, mir-70, mir-708, mir-71, mir-711, mir-720, mir-73, mir-74, mir-744, mir-75, mir-760, mir-761, mir-764, mir-765, mir-767, mir-77, mir-770, mir-786, mir-787, mir-789, mir-790, mir-791, mir-8, mir-80, mir-802, mir-81, mir-83, mir-84, mir-85, mir-86, mir-87, mir-872, mir-874, mir-875, mir-876, mir-877, mir-879, mir-883, mir-885, mir-887, mir-891, mir-9, mir-90, mir-92, mir-920, mir-922, mir-927, mir-929, mir-932, mir-934, mir-936, mir-937, mir-938, mir-939, mir-940, mir-941, mir-942, mir-944, mir-96, mir-969, mir-981, mir-983, mir-987, mir-988, mir-995, mir-996, mir-999, mir-BART1, mir-BART12, mir-BART15, mir-BART17, mir-BART2, mir-BART20, mir-BART3, mir-BART5, mir-BART7, mir-BHRF1-1, mir-BHRF1-2, mir-BHRF1-3, mir-M7, mir-TAR, mir-iab-4, mraW, nuoG, potC, rli31, rli51, rox1, rox2.

Table 6.4: List of ncRNA families included on snoRNA's benchmark.

Family name	Avg. % id.	Length	Count	Fragments	Training	Test
Hammerhead_3	68%	85	84	0	82	1
SNORD25	80%	90	8	0	5	1
SNORD96	70%	101	9	0	5	2
SNORD15	60%	165	15	3	13	2
S15	65%	136	79	0	68	8
Alpha_RBS	69%	135	39	0	34	3
msr	76%	83	10	0	7	3
SNORA13	69%	164	48	0	45	3
SNORA35	67%	147	31	0	26	1
SNORD88	76%	113	35	0	34	1
U6atac	71%	153	61	0	60	1
sR4	66%	66	11	0	7	1
sR28	80%	55	6	0	5	1
sR13	73%	58	6	0	5	1
snoR72Y	66%	101	10	0	9	1
snR5	80%	204	11	0	10	1
sR5	72%	60	10	0	5	2
sR8	69%	54	7	0	5	1
Afu_254	60%	165	6	0	5	1
Termite-leu	94%	78	20	0	14	2
rimP	70%	151	50	0	49	1
c-di-GMP-II	55%	113	54	0	51	3
Total	—	—	610	3	544	41

Figure 6.1: ROC curves and MER values for the snoRNA benchmark. The "snoRNA-default-prior" curve represents Infernal's default prior while the "snoRNA-our-prior" curve represents our prior made using data from the snoRNA group.

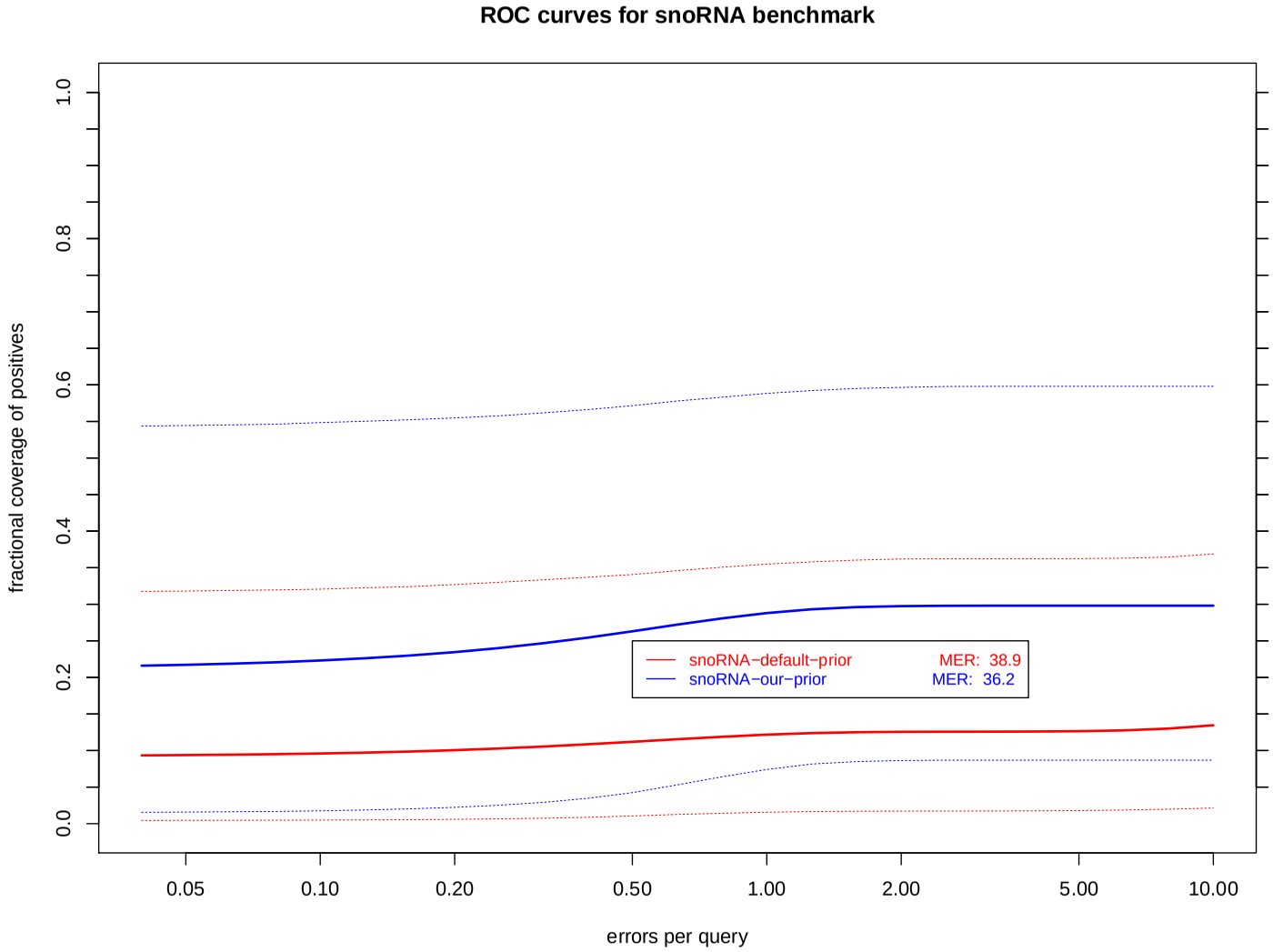


Figure 6.2: ROC curves and MER values for the microRNA benchmark. The "miRNA-default-prior" curve represents Infernal's default prior while the "miRNA-our-prior" curve represents our prior made using data from the microRNA group.

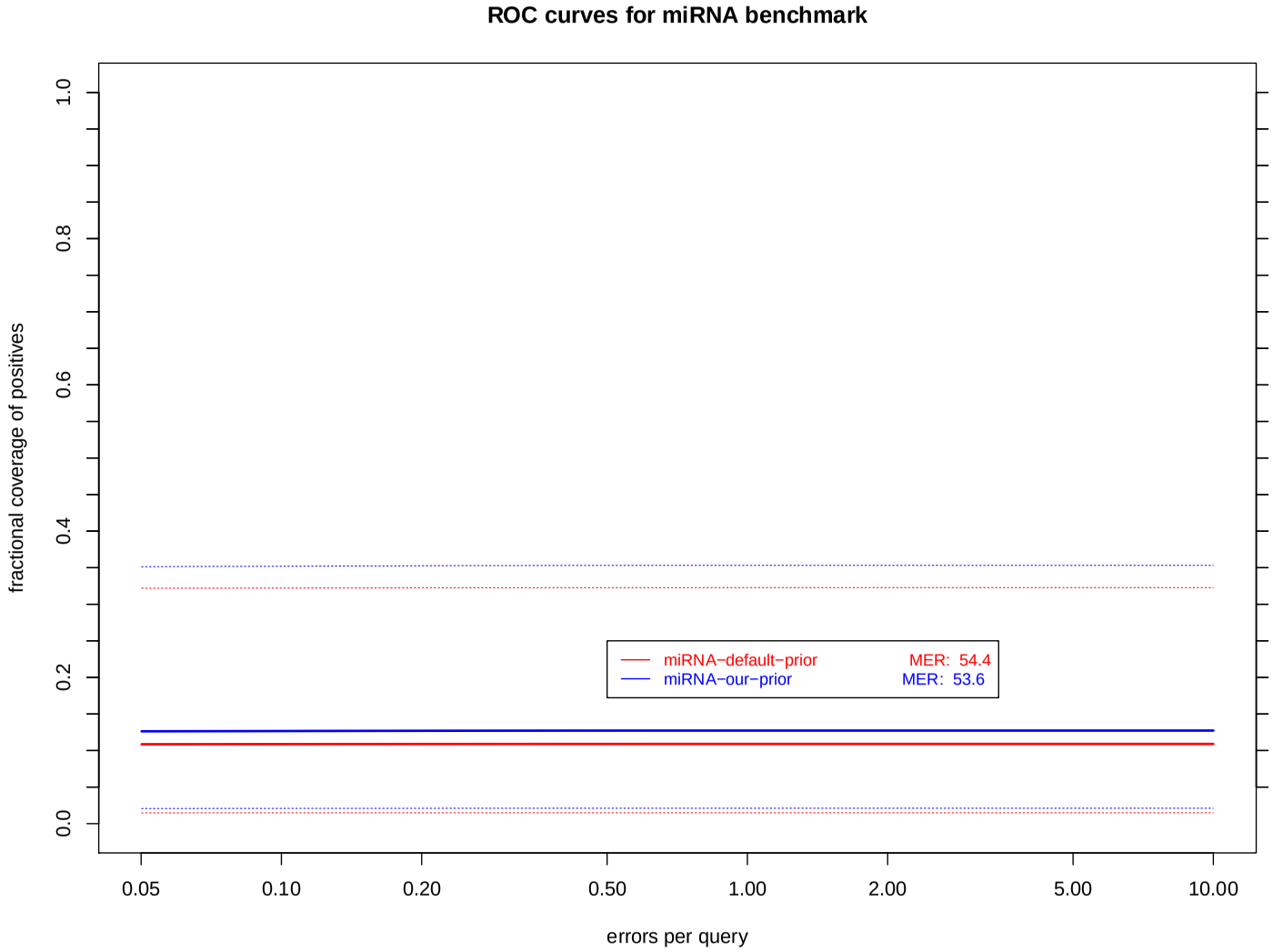


Table 6.5: List of ncRNA families included on microRNA's benchmark.

Family name	Avg. % id.	Length	Count	Fragments	Training	Test
Vault	59%	164	74	1	72	2
mir-10	71%	79	36	0	34	2
Bacteria_small_SRP	51%	122	238	33	209	13
mir-172	62%	126	11	0	7	2
L21_leader	58%	141	38	0	28	3
mir-216	71%	94	36	0	20	2
mir-210	76%	99	26	0	13	1
mir-36	56%	113	38	0	30	6
mir-239	63%	108	9	0	5	3
mir-87	85%	103	11	0	8	1
mir-81	67%	104	9	0	8	1
mir-283	65%	102	13	0	12	1
mir-340	61%	109	43	1	41	2
MIR473	60%	112	6	0	5	1
MIR474	59%	131	11	0	10	1
mir-548	78%	93	54	0	53	1
Lacto-usp	76%	92	6	0	5	1
mraW	62%	124	46	0	16	16
MIR2118	72%	190	7	0	6	1
Total	—	—	712	35	582	60

Table 6.6: List of types of transition that were not successfully derived.

snoRNA priors	microRNA priors
MATP_ML → BIF	MATP_ML → BIF
MATP_ML → END	MATP_ML → END
MATP_MR → BIF	MATP_MR → BIF
MATP_MR → END	MATP_MR → END
MATP_D → BIF	MATP_D → BIF
MATP_D → END	MATP_D → END
MATL_ML → MATL	MATP_IL → BIF
BEGR_S → MATL	MATP_IL → END
	BEGL_S → BIF
	ROOT_IL → BIF

Chapter 7

Conclusion

In this work built a new tool, `rfam-dendrogram`, able to do many analysis on Rfam's data based on its families' secondary structure. Using `rfam-dendrogram`, we found six clusters of structurally similar ncRNA families (SNORD1, SNORD2, SNORA, miRNA1, miRNA2, CRISPR), together covering 60.5% of all Rfam families. From these clusters we created two groups, one made up mostly of microRNAs (miRNA1, miRNA2) and the other of snoRNAs (SNORD1, SNORD2, SNORA).

Group-specific priors, constructed using data from each of these groups, instead of the whole Rfam, were then derived. A benchmark for each group was also created and the priors were tested against Infernal's default prior. Although our new priors had slightly better results, all test results were statistically inconclusive.

As future work, we would like to:

- Create a better benchmark and get more relevant data about the group-specific priors. For example, in order to test the microRNA prior, we may: create a pseudo-chromosome by embedding miRBase data into a random sequence of non-microRNAs; build and calibrate CMs for all microRNA families; and try to find miRBase's microRNAs using the calibrated CMs.
- Release `rfam-dendrogram`, which is in a pretty good shape, and `regene`, which still needs some work.
- Evaluate priors that are specific to some group of organisms (instead of a grupo of ncRNA families), which means that we would filter the sequences from the families (instead of filtering the families themselves). Interesting groups that may be explored include bacterias (gram-positive, gram-negative, both), animals, plants.

Referências

[1] WIKIPEDIA. *File:Stem-loop.svg — Wikipedia, The Free Encyclopedia*. 2006. Disponível em: <http://en.wikipedia.org/wiki/File:Stem-loop.svg>. vi, 6

[2] WIKIPEDIA. *File:Pseudoknot.svg — Wikipedia, The Free Encyclopedia*. 2006. Disponível em: <http://en.wikipedia.org/wiki/File:Pseudoknot.svg>. vi, 7

[3] WUYTS, J. et al. The European large subunit ribosomal RNA database. *Nucleic Acids Research*, Oxford University Press, v. 29, p. 175–177, 2001. Disponível em: <http://nar.oxfordjournals.org/cgi/content/short/29/1/175>. vi, 18

[4] WUYTS, J. et al. The European database on small subunit ribosomal RNA. *Nucleic Acids Research*, Oxford University Press, v. 30, p. 183–185, 2001. Disponível em: <http://nar.oxfordjournals.org/cgi/content/short/29/1/175>. vi, 18

[5] LESSA, F. A. et al. Clustering Rfam 10.1: Clans, Families, and Classes. *Genes*, v. 3, n. 3, p. 378–390, jul. 2012. ISSN 2073-4425. Disponível em: <http://www.mdpi.com/2073-4425/3/3/378/>. ix, 54

[6] LESSA, F. A. et al. Regene: Automatic Construction of a Multiple Component Dirichlet Mixture Priors Covariance Model to Identify Non-coding RNA. In: *Bioinformatics Research and Applications*. Springer, 2011. p. 380–391. Disponível em: <http://www.springerlink.com/index/G3020371520157N5.pdf>. ix, 2, 12, 32, 37, 39, 75

[7] CRICK, F. Central dogma of molecular biology. *Nature*, Nature Publishing Group, v. 227, n. 5258, p. 561–563, Agosto 1970. Disponível em: <http://dx.doi.org/10.1038/227561a0>. 1

[8] GRIFFITHS-JONES, S. Annotating Noncoding RNA Genes. *Annu. Rev. Genomics Hum. Genet*, v. 8, p. 279–298, 2007. 1

[9] LEY, T. J. et al. A pilot study of high-throughput, sequence-based mutational profiling of primary human acute myeloid leukemia cell genomes. *Proceedings of the National Academy of Sciences of the United States of America*, National Academy of Sciences, v. 100, n. 24, p. 14275–14280, Novembro 2003. Disponível em: <http://dx.doi.org/10.1073/pnas.2335924100>. 1

[10] ALTSCHUL, S. F. et al. Basic local alignment search tool. *Journal of molecular biology*, v. 215, n. 3, p. 403–410, Outubro 1990. ISSN 0022-2836. Disponível em: <http://dx.doi.org/10.1006/jmbi.1990.9999>. 1

[11] MOUNT, S. M. et al. Spliceosomal Small Nuclear RNA Genes in Eleven Insect Genomes. *RNA*, v. 13, p. 5–14, 2007. 1

[12] EDDY, S. R.; DURBIN, R. RNA sequence analysis using covariance models. *Nucleic acids research*, Cambridge, Inglaterra, v. 22, n. 11, p. 2079–2088, Junho 1994. ISSN 0305-1048. Disponível em: <http://view.ncbi.nlm.nih.gov/pubmed/8029015>. 1, 10

[13] NAWROCKI, E. P.; KOLBE, D. L.; EDDY, S. R. Infernal 1.0: Inference of RNA alignments. *Bioinformatics*, v. 25, p. 1335–1337, 2009. Disponível em: <http://infernal.janelia.org/>. 1, 2, 8, 10

[14] ZUKER, M.; MATHEWS, D. H.; TURNER, D. H. Algorithms and thermodynamics for RNA secondary structure prediction: A pratical guide. In: *RNA Biochemistry and Biotechnology*. NATO Advanced Study Institute: Kluwer Academic Publishers, 1999. 1, 8

[15] HOFACKER, I. L.; FEKETE, M.; STADLER, P. F. Secondary structure prediction for aligned RNA sequences. *Journal of Molecular Biology*, Wien, Áustria, v. 319, n. 5, p. 1059–1066, Junho 2002. ISSN 0022-2836. Disponível em: <http://dx.doi.org/10.1016/S0022-2836(02)00308-X>. 1, 8

[16] LIU, J.; GOUGH, J.; ROST, B. Distinguishing protein-coding from non-coding RNAs through Support Vector Machines. *PLoS Genet.*, v. 2, n. 4, p. e29–e36, Apr 2006. Disponível em: <Available at http://www.ncbi.nlm.nih.gov/pubmed/16683024>. 1, 8

[17] KONG, L. et al. CPC: assess the protein-coding potential of transcripts using sequence features and support vector machine. *Nucleic Acids Res.*, v. 35, p. 345–349, 2007. 1, 8

[18] ARRIAL, R. T.; TOGAWA, R. C.; BRIGIDO, M. M. Screening non-coding RNAs in transcriptomes from neglected species using PORTRAIT: case study of the pathogenic fungus Paracoccidioides brasiliensis. *BMC Bioinformatics*, v. 10, p. 239, 2009. ISSN 1471-2105. Disponível em: <http://dx.doi.org/10.1186/1471-2105-10-239>. 1, 8

[19] SILVA, T. C. et al. SOM-PORTRAIT: Identifying Non-coding RNAs Using Self-Organizing Maps. In: *4th Brazilian Symposium on Bioinformatics - BSB 2009*. [S.l.]: Springer, 2009. (Lecture Notes in Computer Science, v. 5676), p. 73–85. 1, 8

[20] GARDNER, P. P. et al. Rfam: Wikipedia, clans and the "decimal" release. *Nucleic Acids Research*, Oxford University Press, v. 39, n. Database issue, p. D141–D145, 2011. Disponível em: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3013711/>. 2, 8

[21] GARDNER, P. P. et al. Rfam: updates to the RNA families database. *Nucl. Acids Res.*, v. 37, n. suppl_1, p. D136–140, January 2009. ISSN 1362-4962. Disponível em: <http://dx.doi.org/10.1093/nar/gkn766>. 2

[22] LEE, P. M. *Bayesian Statistics: An Introduction (Arnold Publication)*. Wiley, 2009. 352 p. Disponível em: <http://www.amazon.com/Bayesian-Statistics-Introduction-Arnold-Publication/dp/0340814055>. 2

[23] NAWROCKI, E. P.; EDDY, S. R. Query-Dependent Banding (QDB) for Faster RNA Similarity Searches. *PLoS Computational Biology*, v. 3, n. 3, p. e56, March 2007. Disponível em: <http://dx.doi.org/10.1371/journal.pcbi.0030056>. 2, 10, 12, 21, 33, 35

[24] QUEIROZ, K. Ernst Mayr and the modern concept of species. *Proceedings of the National Academy of Sciences of the United States of America*, Washington, EUA, v. 102, p. 6.600–6.607, Maio 2005. ISSN 0027-8424. Disponível em: <http://dx.doi.org/10.1073/pnas.0502030102>. 4

[25] MAYR, E. *Systematics and the Origin of Species, from the Viewpoint of a Zoologist*. Harvard University Press, 1942. ISBN 0674862503. Disponível em: <http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0674862503>. 4

[26] CHAMBERS 21st Century Dictionary. [s.n.]. Edição on-line. Disponível em: <http://www.chambersharrap.co.uk/chambers/features/chref/chref.py/main?query=organism&title=21st>. Acesso em: 22 ago. 2010. 4

[27] SETUBAL, C.; MEIDANIS, J. *Introduction to Computational Molecular Biology*. Boston, EUA: PWS Publishing, 1997. 320 p. Capa dura. ISBN 0534952623. 5

[28] ZUKER, M.; STIEGLER, P. Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. *Nucleic Acids Research*, Oxford University Press, v. 9, n. 1, p. 133–148, jan. 1981. ISSN 1362-4962. Disponível em: <http://dx.doi.org/10.1093/nar/9.1.133>. 5

[29] HUANG, Z. et al. Fast and accurate search for non-coding RNA pseudoknot structures in genomes. *Bioinformatics*, Department of Computer Science, University of Georgia, Athens, GA 30602., v. 24, n. 20, p. 2281–2287, Outubro 2008. ISSN 1460-2059. Disponível em: <http://dx.doi.org/10.1093/bioinformatics/btn393>. 6

[30] STORZ, G. An expanding universe of noncoding RNAs. *Science*, American Association for the Advancement of Science, v. 296, n. 5571, p. 1260–1263, 2002. Disponível em: <http://www.ncbi.nlm.nih.gov/pubmed/12016301>. 6

[31] EDDY, S. R. Accelerated Profile HMM Searches. *PLoS computational biology*, Public Library of Science, v. 7, n. 10, p. e1002195, out. 2011. ISSN 1553-7358. Disponível em: <http://dx.plos.org/10.1371/journal.pcbi.1002195>. 8

[32] DURBIN, R. et al. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge, Inglaterra: Cambridge University Press, 1998. 8

[33] EDDY, S. R. Profile hidden Markov models. *Bioinformatics*, Washington, EUA, v. 14, n. 9, p. 755–763, Outubro 1998. ISSN 1367-4803. Disponível em: <http://dx.doi.org/10.1093/bioinformatics/14.9.755>. 8

[34] LOWE, T. M.; EDDY, S. R. tRNAscan-SE: a program for improved detection of transfer RNA genes in genomic sequence. *Nucleic acids research*, v. 25, n. 5, p. 955–64, mar. 1997. ISSN 0305-1048. Disponível em: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC146525/>. 8

[35] SOLDà, G. et al. An Ariadne's thread to the identification and annotation of noncoding RNAs in eukaryotes. *Briefings in bioinformatics*, v. 10, n. 5, p. 475–89, set. 2009. ISSN 1477-4054. Disponível em: <http://www.ncbi.nlm.nih.gov/pubmed/19383843>. 8

[36] BU, D. et al. NONCODE v3.0: integrative annotation of long noncoding RNAs. *Nucleic acids research*, v. 40, n. Database issue, p. D210–5, jan. 2012. ISSN 1362-4962. Disponível em: <http://nar.oxfordjournals.org/cgi/content/abstract/gkr1175v1>. 8

[37] PANG, K. C. et al. RNAdb–a comprehensive mammalian noncoding RNA database. *Nucleic acids research*, v. 33, n. Database issue, p. D125–30, jan. 2005. ISSN 1362-4962. Disponível em: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC540043/>. 8

[38] KOZOMARA, A.; GRIFFITHS-JONES, S. miRBase: integrating microRNA annotation and deep-sequencing data. *Nucleic acids research*, v. 39, n. Database issue, p. D152–7, jan. 2011. ISSN 1362-4962. Disponível em: <http://nar.oxfordjournals.org/cgi/content/abstract/39/suppl_1/D152>. 9

[39] MITUYAMA, T. et al. The Functional RNA Database 3.0: databases to support mining and annotation of functional RNAs. *Nucleic acids research*, v. 37, n. Database issue, p. D89–92, jan. 2009. ISSN 1362-4962. Disponível em: <http://nar.oxfordjournals.org/cgi/content/abstract/37/suppl_1/D89>. 9

[40] DEMPSTER, A. P.; LAIRD, N. M.; RUBIN, D. B. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society Series B Methodological*, JSTOR, v. 39, n. 1, p. 1–38, 1977. ISSN 00359246. Disponível em: <http://www.jstor.org/stable/2984875>. 22, 32

[41] GERSTEIN, M.; SONNHAMMER, E. L. L.; CHOTHIA, C. Volume changes in protein evolution. *Journal of Molecular Biology*, v. 236, n. 4, p. 1067–1078, Março 1994. ISSN 0022-2836. Disponível em: <http://www.sciencedirect.com/science/article/B6WK7-4C5PW9J-D/2/19e0472db9e7521d8dbe0fdcae64f1ab>. 22

[42] SJÖLANDER, K. et al. Dirichlet mixtures: a method for improved detection of weak but significant protein sequence homology. *Computer Applications in the Biosciences*, Santa Cruz, EUA, v. 12, n. 4, p. 327–345, Agosto 1996. ISSN 0266-7061. Disponível em: <http://view.ncbi.nlm.nih.gov/pubmed/8902360>. 30, 32, 33

[43] SHEWCHUK, J. R. *An introduction to the conjugate gradient method without the agonizing pain*. [S.l.], 1994. Disponível em: <http://www.cs.cmu.edu/~quake-papers/painless-conjugate-gradient.pdf>. 33

[44] HAGER, W. W.; ZHANG, H. A new conjugate gradient method with guaranteed descent and an efficient line search. *SIAM J. on Optimization*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, v. 16, n. 1, p. 170–192, 2005. ISSN 1052-6234. Disponível em: <http://dx.doi.org/10.1137/030601880>. 33, 34

[45] LESSA, F. HackageDB: statistics-dirichlet. 2012. Disponível em: <http://hackage.haskell.org/package/statistics-dirichlet>. 34

[46] THE Infernal's User Guide. Available at `http://infernal.janelia.org/`. 35

[47] LESSA, F. *HackageDB: hierarchical-clustering*. 2012. Disponível em: <http://hackage.haskell.org/package/hierarchical-clustering>. 37

[48] LESSA, F. *HackageDB: gsc-weighting*. 2012. Disponível em: <http://hackage.haskell.org/package/gsc-weighting>. 37

[49] LESSA, F. *HackageDB: biostockholm*. 2012. Disponível em: <http://hackage.haskell.org/package/biostockholm>. 37

[50] LESSA, F. *HackageDB: hierarchical-clustering-diagrams*. 2012. Disponível em: <http://hackage.haskell.org/package/hierarchical-clustering-diagrams>. 37

[51] GERSTEIN, M.; SONNHAMMER, E. L.; CHOTHIA, C. Volume changes in protein evolution. *Journal of molecular biology*, v. 236, n. 4, p. 1067–78, mar. 1994. ISSN 0022-2836. Disponível em: <http://www.ncbi.nlm.nih.gov/pubmed/8120887>. 38

[52] PAGE, R. *TreeView X*. 2012. Disponível em: <http://darwin.zoology.gla.ac.uk/ rpage/treeviewx/>. 38

[53] LEINONEN, R. et al. The European Nucleotide Archive. *Nucleic Acids Research*, Oxford University Press, v. 39, n. Database issue, p. D28–D31, 2011. Disponível em: <http://www.ncbi.nlm.nih.gov/pubmed/20972220>. 38

Anexo I

Article published on Genes 2012 [5]

Genes **2012**, *3*, 378-390; doi:10.3390/genes3030378

OPEN ACCESS

genes

ISSN 2073-4425

www.mdpi.com/journal/genes

Article

Clustering Rfam 10.1: Clans, Families, and Classes

Felipe A. Lessa [1], Tainá Raiol [2], Marcelo M. Brigido [2], Daniele S. B. Martins Neto [3], Maria Emília M. T. Walter [1,*] and Peter F. Stadler [4,5,6,7,8,9]

[1] Department of Computer Science, Institute of Exact Sciences, University of Brasília, Brasília 70910-900, Brazil; E-Mail: felipe.lessa@gmail.com

[2] Department of Cellular Biology, Institute of Biology, University of Brasília, Brasília 70910-900, Brazil; E-Mails: tainaraiol@gmail.com (T.R.); brigido@unb.br (M.M.B.)

[3] Department of Mathematics, University of Brasília, Brasília 70910-900, Brazil; E-Mail: daniele@mat.unb.br

[4] Bioinformatics Group, Department of Computer Science, and Interdisciplinary Center for Bioinformatics, University of Leipzig, Härtelstraße 16-18, D-04107 Leipzig, Germany; E-Mail: studla@bioinf.uni-leipzig.de

[5] Max Planck Institute for Mathematics in the Sciences, Inselstraße 22, Leipzig D-04103, Germany

[6] Fraunhofer Institut für Zelltherapie und Immunologie–IZI Perlickstraße 1, D-04103 Leipzig, Germany

[7] Institute for Theoretical Chemistry , University of Vienna, Währingerstraße 17, Wien A-1090, Austria

[8] Center for non-coding RNA in Technology and Health, University of Copenhagen, Grønnegårdsvej 3, DK-1870 Frederiksberg C, Denmark

[9] Santa Fe Institute, 1399 Hyde Park Rd., Santa Fe, NM 87501, USA

***** Author to whom correspondence should be addressed; E-Mail: mariaemilia@unb.br; Tel.: +55-61-3107-7358; Fax: +55-61-3107-7359.

Received: 5 May 2012, in revised form: 4 June 2012 / Accepted: 15 June 2012 / Published: 5 July 2012

Abstract: The `Rfam` database contains information about non-coding RNAs emphasizing their secondary structures and organizing them into families of homologous RNA genes or functional RNA elements. Recently, a higher order organization of `Rfam` in terms of the so-called clans was proposed along with its "decimal release". In this proposition, some of the families have been assigned to clans based on experimental and computational data in order to find related families. In the present work we investigate an alternative classification for the RNA families based on tree edit distance. The resulting clustering recovers some of the `Rfam` clans. The majority of clans, however, are not recovered by the structural clustering. Instead, they get dispersed into larger clusters, which correspond roughly to

well-described RNA classes such as snoRNAs, miRNAs, and CRISPRs. In conclusion, a structure-based clustering can contribute to the elucidation of the relationships among the Rfam families beyond the realm of clans and classes.

Keywords: Rfam; non-coding RNA; secondary structure; clans; clusters

1. Introduction

The Rfam database systematically collects sequences, alignments, consensus secondary structures, covariance models (CMs) and the corresponding annotation for RNAs with evolutionarily conserved secondary structures [1,2]. The database is constructed using a small manually curated "seed alignment" for each RNA family which is then expanded by a large-scale search for homologs in nucleotide databases. Rfam-families consist of homologous sequences that can be reasonably well aligned and that share some common function.

Rfam release 10.0 [3] introduced the concept of a *clan* as a means for describing explicit relationships between Rfam families for which homology is recognizable but sequence similarities are too faint for good alignments (such as archaeal RNase P, nuclear RNase P, and the two bacterial RNase P types a and b) or which are classified into different Rfam families because of diverse functions (for example RNAse MRP RNA and the four RNAse P RNA families). Rfam clans thus correspond to the definition of RNA families as used e.g., in [4].

At an even higher level of aggregation, *RNA classes* comprise families that share characteristic sequence and/or structure features, without necessarily being evolutionarily related. The best known examples of RNA classes are animal microRNAs, both classes of small nucleolar RNAs (the box C/D snoRNAs and the box H/ACA snoRNAs) and transfer RNAs. At the level of RNA classes, it is not required for class members to be related by common descent.

There is strong support for the hypothesis that all tRNAs are homologs deriving from a single clover-leaf-structured ancestor [5,6]. Additional RNA families such as mascRNA [7], menRNAs [8], or BC1 [9] are also descendants of tRNAs and hence belong to the tRNA clan. MicroRNA families, on the other hand, frequently arise *de novo* [10–12]. In fact, presence/absence patterns of microRNA families have turned out to be a valuable and nearly homoplasy-free phylogenetic marker [13]. It has been argued that novel microRNA families can easily arise in transcribed regions, considering that stem-loop structures resembling microRNA precursors frequently occur in random RNA sequences. This mechanism is most easily seen in the expansion of microRNA clusters by hairpins that are unrelated to more ancient cluster's components [14]. Topics of innovation and expansion of microRNA families are reviewed e.g., in [15,16]. In regard to snoRNAs it is not clear to what extent families are ancestrally related. While distant homologies can be established in some cases, e.g., U87 and U88 [17], there is also some evidence for the lineage-specific innovation of snoRNA families, e.g., in birds [18, Figure S8].

Clustering of RNAs based on their sequences and/or structural characteristics is probably the simplest approach for identifying families, clans, or classes, see e.g., [19–21]. Here we are only concerned with

higher levels of aggregation beyond the level of Rfam families; thus we focus on structural similarity. A wide variety of different algorithmic schemes have been proposed to quantify (dis)similarities among known secondary structures. Since secondary structures have canonical representations in the form of ordered trees, tree alignments (RNAforester [22]) and tree editing [23,24] are the most natural and elegant means of comparison. For the specific purpose of clustering, however, it is a fundamental shortcoming of alignments that the cost functions violate the triangle inequality and hence do not form a metric on the set of labeled ordered trees [22]. Hence we employ here a tree-edit distance.

The use of direct structure comparison becomes quite limited in practical applications, because secondary structures of individual sequences are unknown in most cases. Computational prediction of secondary structures for individual sequences, on the other hand, is not sufficiently accurate. This limitation may be overcome, or at least alleviated, however, by using comparative information, see e.g., [25]. Successful applications of RNA clustering [19,21] typically use combined sequence and structure alignments based on the Sankoff algorithm [26] to combine thermodynamic rules with conservation information. For each Rfam family, a high-quality manual sequence alignment together with a matching consensus secondary structure model is available. These consensus structures can be readily used for structure-based clustering.

In this contribution we explore to what extent the manually annotated clans of related Rfam families are detectable by unsupervised clustering, whether RNA classes such as microRNA and the two snoRNA classes are recognizable, and whether there are good candidates for clans or classes of Rfam families that have remained so far undetected.

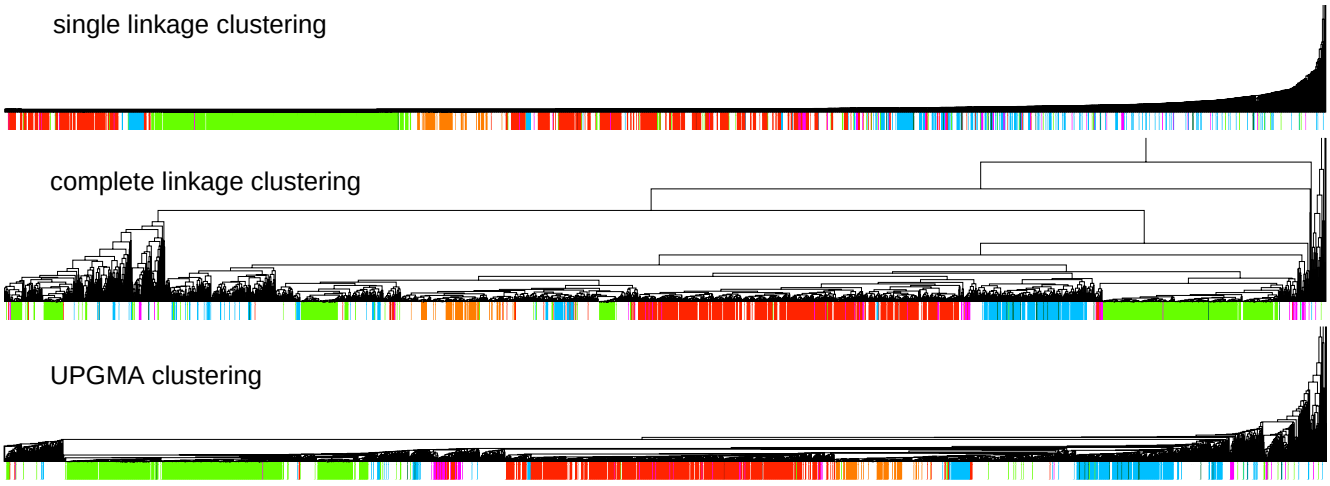
2. Results and Discussion

2.1. Clusters and RNA Classes

Starting from the matrix of tree edit distances between the 1973 families collected in Rfam 10.1 we computed hierarchical clusterings using UPGMA [27], single linkage [28], and complete linkage [29] methods. Results for each computation are represented as ultrametric trees (Figure 1). High-resolution versions of all diagrams are available at http://www.biomol.unb.br/rfam/.

Firstly the three hierarchies were compared to each other. By doing this, it was observed that the single linkage hierarchy is clearly different from the other two, considering that it has a caterpillar-like shape with only a few discernible clusters. Figure 1 shows a strong tendency of the microRNA, snoRNA and, to a lesser extent, the CRISPR families to cluster together. All three clustering methods also show a pronounced, although not total, separation among animal, plant, and viral microRNAs. In contrast to viral microRNAs, differences between plant and animal microRNAs have been well recorded in the literature, e.g., reviewed in [30]. Most viral microRNA families in Rfam are from Herpesviridae (16 of 20 viral microRNA families), which clustered with animal microRNA families, as expected. Although viral microRNAs are wide-spread [31,32], they are often poorly conserved and hence have not been included as an Rfam family.

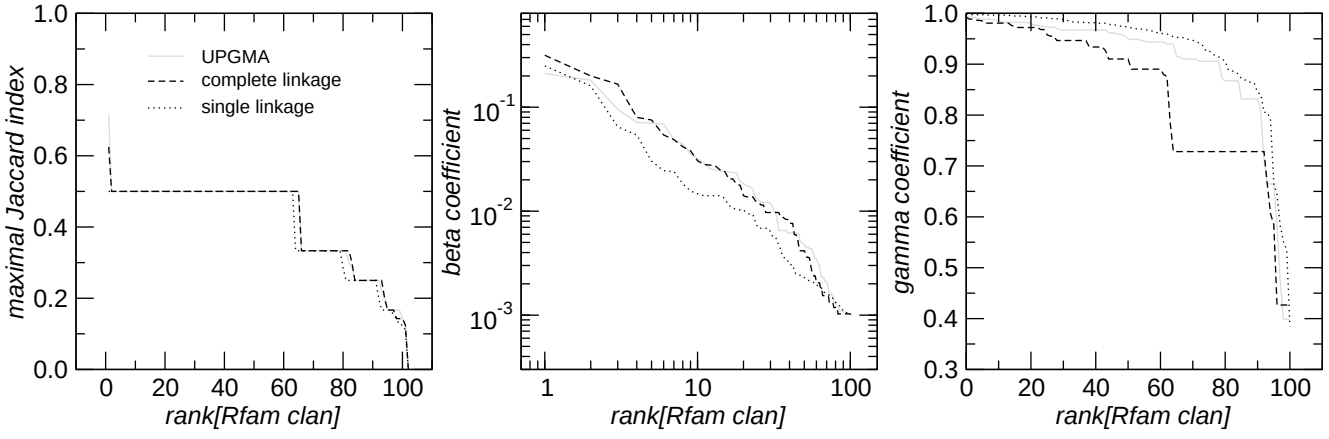
Figure 1. Dendrograms of the consensus structures of all `Rfam 10.1` families computed with three different hierarchical clustering methods. Large important classes of ncRNAs are highlighted. Reddish colors denote three classes of microRNAs animal (scarlet), plant (fuchsia), and viral (brown). Box C/D snoRNAs are represented by bright green, while light blue indicates box H/ACA snoRNAs. Prokaryotic CRISPR families are shown in orange.



2.2. Clusters and `Rfam` Clans

`Rfam 10.1` defines 102 clans, the majority of which comprises only two `Rfam` families. Figure 2 shows that they have not been well recovered by the clustering of the consensus structure. In fact, the coefficient β which measures a clan's tightness within the dendrogram shows a power-law like behavior, indicating that only a few clans are tightly clustered while most clans spread out over large areas of the dendrogram.

Figure 2. Distribution of α (maximal Jaccard index), β, and γ for all of the 102 `Rfam` clans. These data show that most clans do not appear tightly clustered w.r.t. any of the three methods. The clans shown in the x-axis, together with α, β, and γ are listed in the supplementary material.



This is not entirely unexpected for several distinct reasons. Firstly, many clans consist of structurally related microRNA families. As all animal microRNA precursors share a very similar stem-loop structure,

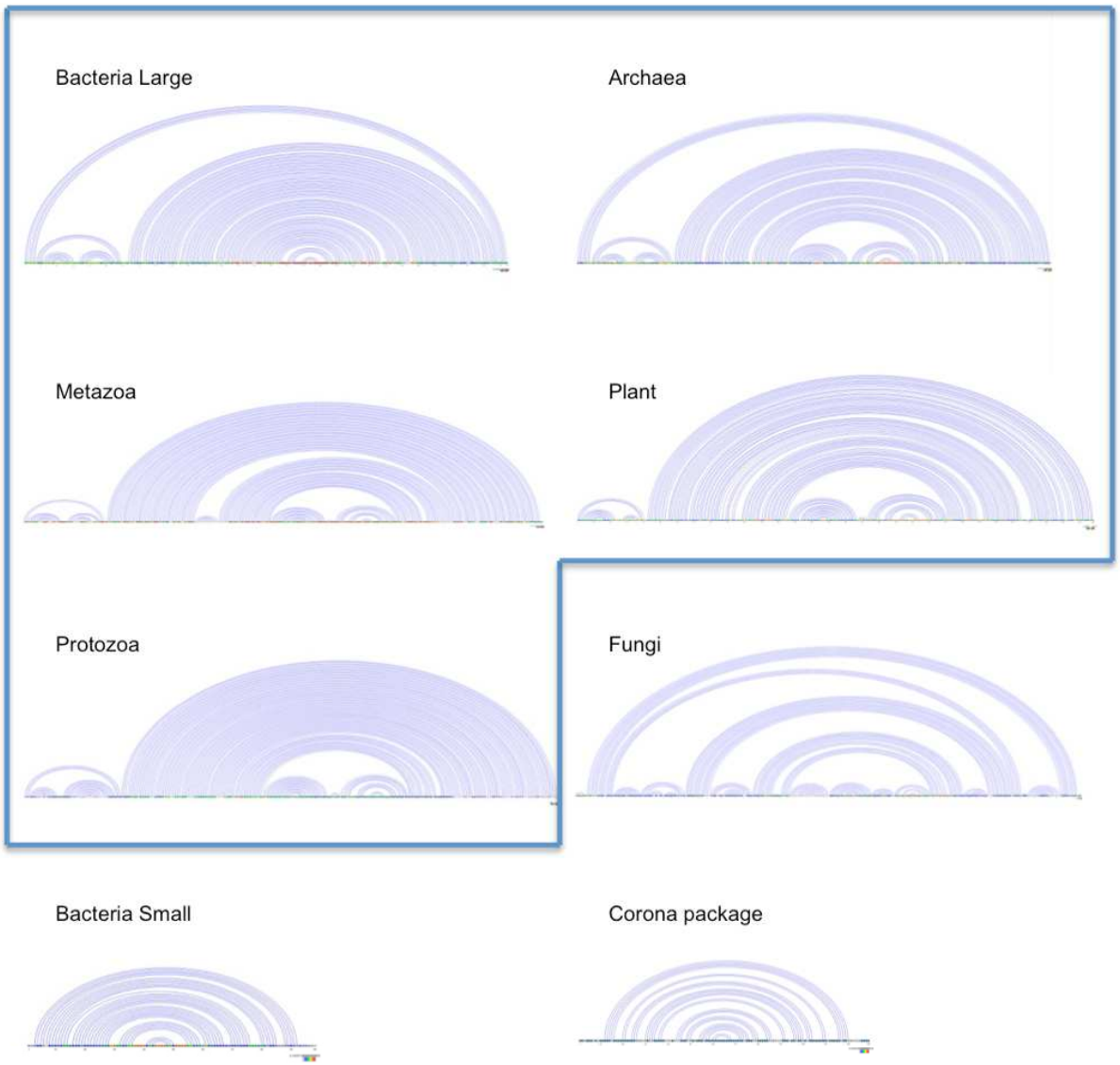
similarities inside a clan are not stronger than those shared with other microRNA families. These clans are thus dispersed throughout the microRNA clusters in Figure 1, leading to small values of α and β. Secondly, this intra-clan similarity clusters most of the animal microRNAs in a single tree branch. In contrast, some other clans combine functionally related families that are clearly structurally distant in their supposed homologous structure: the tRNA clan (CL0001) for instance contains four families of bacterial tmRNAs along with the tRNA and the tRNA-Sec families.

Compared to a tRNA, a tmRNA has a big loop insertion that reflects its role as an mRNA-like repair template for ribosomal protein synthesis. Thus tmRNAs and tRNAs are separated by very large tree-edit distances. The four tmRNA families (RF00023, RF01849, RF01850, RF01851) also feature major differences. The most common one-piece type and distinct families of the two-piece tmRNAs can be respectively found in α-proteobacteria, β-proteobacteria, and cyanobacteria. In addition, some tmRNAs with a permuted organization have also been described [33,34].

Despite the usually poor representation of `Rfam` clans in the structure-based clustering, there are a few clans that at least partially agree with the clustering data. The SRP clan (CL00003) is the one that was most coherently recovered as measured by the maximal Jaccard index $\alpha \approx 0.71$. It consists of 5 phylogenetically defined subgroups of signal recognition particle RNAs, whose homology is well established [35]: Metazoa SRP (RF00017), Bacteria large SRP (RF01854), Plant SRP (RF01855), Archaea SRP (RF01857) and Protozoa SRP (RF01856) (Figure 3). In addition to these families, which cluster together in our analysis, the SRP clan also contains Bacteria small SRP (RF00169) and Fungi SRP (RF01502), which were located far apart from the clustered SRP families in the UPGMA tree. The bacterial small SRP RNA (4.5S RNA) family (RF00169) is fully functional in mycoplasma and gram-negative bacteria, and harbors the conserved helices 5 and 8, which can be found in all kingdoms [35]. The reduced size of such RNAs prevented them from clustering with other SRP families. In our analysis, they clustered with RF00182, an unrelated viral element without any obvious functional or phylogenetic association. The fungal SRP RNA components (RF01502), on the other hand, possessed a highly conserved structure when compared to the other clan member. It contains several extra helices, however, that explain the large values of the tree distance in comparison to the other clan members. Thus, it appeared separated in a distinct tree branch. This case exemplifies that clustering with global distance functions alone cannot cope with those cases where there are dramatic structural differences between homologous RNAs.

Therefore, tree edit distance does not necessarily respect phylogenetic relationships. Nevertheless, homologous structures are frequently located in close proximity in the tree. Ribonuclease P, an ubiquitous ribozyme required for tRNA processing [36,37], for instance, is represented by four `Rfam` families in the clan (CL0002). Although they are located within the same subtree, they cluster with functionally and evolutionarily unrelated Rfam families such as fungi_U3 (RF01846), U1_yeast (RF00488), and the internal ribosomal entry sites of hepatitis A virus, IRES_HepA (RF00228). Interestingly, the alpha_tmRNA is also found in the RNAse P sub-tree, very close to RNAseP_nuc (RF00009) and RUF21 (RF01825), an yeast ncRNA with unknown functions. On the other hand, the prototypic tmRNA (RF00023) clusters together with the RNAse_MRP (RF00030), a distant homolog of RNAse P [38,39]. Therefore, one can speculate that RNAse P and tmRNA might share a common evolutionary history.

Figure 3. Linear representations of the secondary structures of the SRP clan members. The seven depicted SRP clan members display a conserved stem loop structure. The five of them that appear as a cluster in the UPGMA tree are delimited by a blue boundary. Fungal SRP family contains extra loops, while the Small Bacterial SRP families contain only a conserved stem loop domain, thus they have been both excluded from the cluster. An UPGMA neighbor of the Small Bacterial SRP family, an unrelated virus derived RFAM family (Corona package), is shown for comparison.



It is worth to mention that functional similarity, on the other hand, does necessarily imply structural similarity. The IRES structures, frequently found in RNA virus and several cellular genes [40], are fine examples, although there are neither common designs, nor common signatures, or even common origins. Therefore, it is not surprising that they are found dispersed all over the cluster tree, gathering diverse functionally unrelated families.

The large plateau at $\alpha = 0.5$ in Figure 2 mostly consists of size two clans, which are not recovered as cherries in the clustering tree. The CRISPR-2 clan is among the few larger clans that cluster together with most of its families. Using the dispersion coefficient β as a measure, clans CRISPR-2 and CRISPR-1

are well clustered. In addition a few microRNA and snoRNA clans with only two members received relatively large β values. A visual inspection of the UPGMA tree (Figure 1) reveals that differences in structural complexity can be directly inferred from the tree: larger and more complex RNAs, in particular rRNAs, appear isolated in this tree's leftmost. In the other extreme, simple structural elements such as microRNA's precursor hairpins and snoRNAs' simple structures were clearly clustered together.

2.3. Significant Structure-Based Clusters

It is natural to ask whether the hierarchical clustering trees can be used to identify significant structure-based clusters, and whether the families with strong similarities share biological functions. The `Rfam` clans have been constructed manually and consist of groups of families sharing common ancestors too divergent to be aligned, or also presenting good alignments but distinct functions so that they could not be included in the same `Rfam` family [3]. Some of these clans clearly share a common evolutionary ancestry, considering that similarities in their biological functions were experimentally verified. Hierarchical clustering, on the other hand, suggests how `Rfam` families that are not contained in clans could be related to each other and how clans may be organized at even higher levels of aggregation. It thus provides a more inclusive annotation.

Generally speaking, clan families were not tightly clustered together by our structure-based distance trees. In particular, larger, more complex secondary structures appeared widely separated from smaller, simpler ones (Figure 4(a)). Figure 4(b) shows in more detail the right hand side of Figure 4(a). As expected, C/D box snoRNAs (SNORD), H/ACA box snoRNAs (SNORA), miRNAs, and CRISPRs were clustered tightly together. Our data suggest then six clusters: SNORD1, SNORD2, SNORA, miRNA1, miRNA2 and CRISPR. The families included in these clusters are listed in Table 1. Such clusters possess very simple and clear secondary structure relationships and aggregate a large fraction of the families in the `Rfam` database.

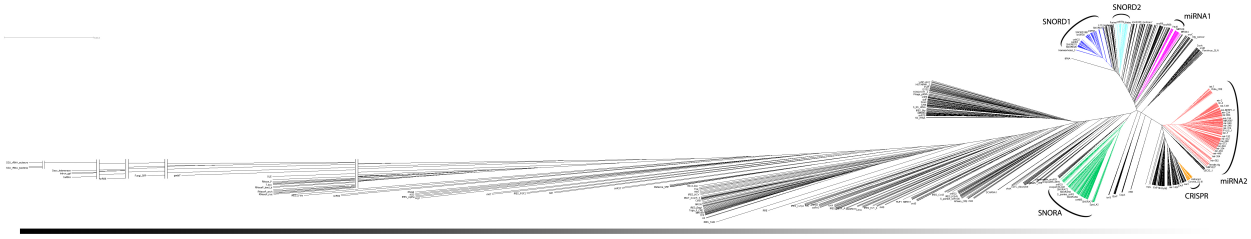
3. Experimental Section

Stockholm formatted alignments were retrieved as well as some metadata (such as accession numbers and short descriptions) for each of the 1,973 families of the `Rfam` database version 10.1 [3]. Then, the consensus structure was extracted from each alignment. Both `Rfam` and `Vienna RNA Package` [41] use string representations for secondary structures in which each base pair is denoted by a matching pair of parentheses and each unpaired bases by dots. This string has a natural interpretation as an ordered forest: dots denote leaf nodes and pairs of parentheses, interior nodes.

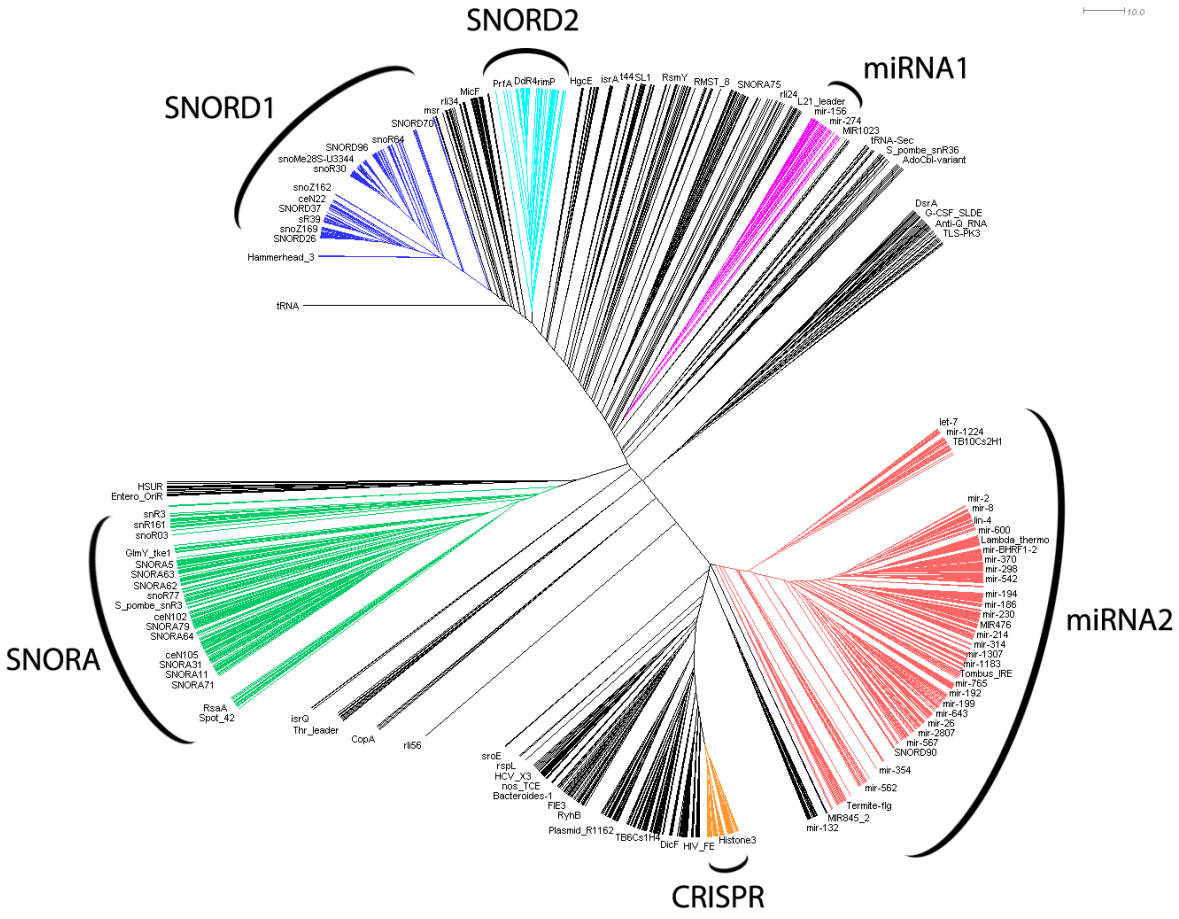
In this study, a full tree editing distance was used, in which every unpaired base and base pair were represented as nodes in the tree representation of the secondary structure. Canonical edit operations for forests were done by insertions and deletions of nodes. Upon node's deletion, its children became parent-node's children. Interior nodes represented base pairs and hence two nucleotides, while leaf nodes referred to single nucleotides, therefore in/del costs were set to 2 for interior nodes and 1 for leaves. Edit distance $d(x, y)$ of two ordered forests x and y can in general be computed efficiently by means of a dynamic programming algorithm [42]. Here we used an implementation available in the `RNAdistance` program, a component of the `Vienna RNA Package` (version 1.8.4). One disadvantage of using this

distance measure is that two secondary structures of very distinct lengths always become very distant, even if they possess similar motifs. In addition, one of the most time consuming steps was running `RNAdistance`, since almost 2 million structure comparisons had to be completed. Nevertheless, this procedure took only a few hours.

Figure 4. Circular view of the UPGMA structural distance-based tree. (**a**) Circular view of the complete UPGMA dendrogram. One can see below the tree a gradient indicating more complex structures on the left and simpler ones on the right. Vertical bars represent shortening of branch length; (**b**) Closer view of clustered snoRNAs (SNORD1 (dark blue), SNORD 2 (light blue) and SNORA (green)), miRNAs (miRNA1 (pink) and miRNA2 (red)) and CRISPR (orange).



(**a**)



(**b**)

Table 1. Clusters of snoRNAs, miRNAs, and CRISPRs.

Cluster	Number of `Rfam` families included	Percentage of `Rfam` families of the expected ncRNA	Clans (name and identification) with all families included in *Cluster*
SNORD1	334	94.9%	SNORD52 (CL00063), U54 (CL00008), SNORD26 (CL00050), SNORD44 (CL00060), SNORD58 (CL00064), SNORD101 (CL00074), SNORD105 (CL00075), SNORND104 (CL00077) SNORD61 (CL00067), SNORD39 (CL00057), SNORD18 (CL00047), SNORD34 (CL00055), SNORD96 (CL00072), SNORD110 (CL00076), SNORD30 (CL00052), SNORD19 (CL00048), SNORD100 (CL00073)
SNORD2	86	81.4%	SNORD15 (CL00045)
SNORA	158	81.0%	SNORA7 (CL00025), SNORA28 (CL00033), SNORA44 (CL00036), SNORA17 (CL00029), SNORA35 (CL00034), SNORA5 (CL00024), SCARNA4 (CL00019)
miRNA1	45	86.6%	MIR171 (CL00099)
miRNA2	472	85.6%	mir-34 (CL00087), mir-216 (CL00094), mir-279 (CL00095), mir-36 (CL00088), mir-81 (CL00091), mir-182 (CL00093), mir-3 (CL00084), mir-50 (CL00089), mir-BART (CL00097), mir-137 (CL00092), mir-73 (CL00090)
CRISPR	100	59.0%	CRISPR-1 (CL00014), CRISPR-2 (CL00015)

Various agglomerative clustering methods differ only in their definition of the distance measure D between clusters. In each step, the two closest clusters, p and q, are united to a single cluster $p \cup q$. The distance of $p \cup q$ to all clusters c is then obtained recursively starting from $D(\{x\}, \{y\}) = d(x, y)$ for clusters consisting of individual points. The form of the recursion determines the particular clustering method [43]. For UPGMA $D(c, p \cup q) = |p|/(|p|+|q|)D(c, p) + |q|/(|p|+|q|)D(c, p)$, for single linkage $D(c, p \cup q) = \min[D(c, p), D(c, q)]$, and for complete linkage $D(c, p \cup q) = \max[D(c, p), D(c, q)]$. The resulting hierarchy of clusters is conveniently represented as a dendrogram T, in which leafs are the individual points. Each cluster c is uniquely identified by a node c' as the set of leafs of subtree rooted at c'. Dendrograms are drawn with a custom-made tool that allows to highlight sets of leaves using regular expressions that match against the extracted `Rfam` metadata.

Distance-based clusters are compared to `Rfam` clans and some other groupings, such as microRNAs or snoRNAs, using three quantitative measures. Denote by q an externally defined group, and let c be a cluster of T. Then we define

$$\alpha(q) = \max_c \frac{|q \cap c|}{|q \cup c|} \qquad \beta(q) = \frac{|q|}{\min_{c:q \subset c} |c|} \qquad \gamma(q) = 1 - \frac{\text{height}(q)}{\min_{c:q \subset c} \text{height}(c)}\,. \tag{1}$$

where height(q) is the height of a given subtree q as calculated by the cluster distance $D(\cdot, \cdot)$, *i.e.*, zero for a leaf q, or $D(l, r)$ for a subtree q having children l and r. The maximal Jaccard index $\alpha(q)$ compromises between coverage and contamination. $\beta(q)$ measures how dispersed q is in T by computing the fraction

of members of q which compose the smallest cluster $c \in T$ that entirely contains q. This measure is quite sensitive to individual outliers. On the other hand, $\gamma(q)$, which is also a measure of dispersion, takes the dendrogram cluster heights avoiding to assign bad scores to groups of very similar families. If the pre-defined group q appears as a cluster in T, $\alpha(q) = \beta(q) = 1$.

The proposed clusters were constructed under careful examination of the most representative `Rfam` families of a particular ncRNA in the UPGMA dendrogram, *i.e.*, a cluster was formed containing the largest number of relatively close `Rfam` families in the tree. The parsing of the `Rfam`, the hierarchical clustering, and the drawing of the resulting dendrograms were implemented in `Haskell`, the source code of which is available at http://hackage.haskell.org/package/ under the open source 3-clause BSD license. As mentioned before, the most time-consuming step was the computation of the distance matrix with `RNAdistance`, while all the other computations took only a few minutes on a notebook.

4. Conclusions

We have developed a tool that calculates and draws automatically dendrograms showing the relationship among all the `Rfam`'s ncRNA families. Such dendrograms have demonstrated to be able to confirm already expected relationships, such as snoRNAs and microRNAs, and also to be able to expose unknown ones, such as those discussed in Section 2.2.

The computational clustering reported here includes all of the 1,973 `Rfam` families, compared to only 306 families that were manually annotated as members of `Rfam` clans. Our analysis suggests that the automatic and manual methods should be combined to comprehensively reveal the structural and evolutionary relationships of the entire content of the `Rfam` database.

As future work, the dendrograms could be further analyzed, considering that it is suspected that there may be more information to be extracted than what we have already covered with this work. We are interested in studying putative ncRNAs' classes that can be defined based on the information we obtained from the dendrograms. For instance, to measure the clusters' consistency derived from those trees, it would be useful to compute the probability of finding the correct cluster for a given sequence. This could be used to know if a particular clustering could be used to predict a snoRNA or a miRNA class based on the secondary structure. It would also be interesting to see if there is a biological basis for the appearance of two clusters of box C/D snoRNAs and microRNAs, respectively, in Figure 4(b). Finally, there is also room for testing different distance measures, including ones that are less affected by length differences or measures that explicitly take into account conserved sequence elements.

Acknowledgments

This work was supported in part by CAPES (to F.A.L. and T.R.), by the *Deutsche Forschungsgemeinschaft* (grant STA 850/7-1 within SPP-1258 to P.F.S.), and by CNPq and FINEP 01.08.0166.00 (to M.E.M.T.W).

References

1. Griffiths-Jones, S.; Bateman, A.; Marshall, M.; Khanna, A.; Eddy, S.R. Rfam: An RNA family database. *Nucleic Acids Res.* **2003**, *31*, 439–441.
2. Griffiths-Jones, S.; Moxon, S.; Marshall, M.; Khanna, A.; Eddy, S.R.; Bateman, A. Rfam: Annotating non-coding RNAs in complete genomes. *Nucleic Acids Res.* **2005**, *33*, 121–124.
3. Gardner, P.P.; Daub, J.; Tate, J.; Moore, B.L.; Osuch, I.H.; Griffiths-Jones, S.; Finn, R.D.; Nawrocki, E.P.; Kolbe, D.L.; Eddy, S.R.; Bateman, A. Rfam: Wikipedia, clans and the "decimal" release. *Nucleic Acids Res.* **2011**, *39*, D141–D145.
4. The Athanasius F. Bompfünewerer RNA Consortium:.; Backofen, R.; Flamm, C.; Fried, C.; Fritzsch, G.; Hackermüller, J.; Hertel, J.; Hofacker, I.L.; Missal, K.; Mosig, Axel Prohaska, S.J.; Rose, D.; *et al*. RNAs everywhere: Genome-wide annotation of structured RNAs. *J. Exp. Zool. B: Mol. Dev. Evol.* **2007**, *308B*, 1–25.
5. Eigen, M.; Lindemann, B.F.; Tietze, M.; Winkler-Oswatitsch, R.; Dress, A.W.M.; von Haeseler, A. How old is the genetic code? Statistical geometry of tRNA provides an answer. *Science* **1989**, *244*, 673–679.
6. Rodin, A.S.; Szathmáry, E.; Rodin, S.N. One ancestor for two codes viewed from the perspective of two complementary modes of tRNA aminoacylation. *Biol. Direct* **2009**, *4*, doi:10.1186/1745-6150-4-4.
7. Wilusz, J.E.; Freier, S.M.; L., S.D. 3' End Processing of a Long Nuclear-Retained Noncoding RNA Yields a tRNA-like Cytoplasmic RNA. *Cell* **2008**, *135*, 919–932.
8. Sunwoo, H.; Dinger, M.E.; Wilusz, J.E.; Amaral, P.P.; Mattick, J.S.; Spector, D.L. MEN ϵ/β nuclear-retained non-coding RNAs are up-regulated upon muscle differentiation and are essential components of paraspeckles. *Genome Res.* **2009**, *19*, 347–359.
9. Rozhdestvensky, T.S.; Kopylov, A.M.; Brosius, J.; Hüttenhofer, A. Neuronal BC1 RNA structure: Evolutionary conversion of a tRNA(Ala) domain into an extended stem-loop structure. *RNA* **2001**, *7*, 722–730.
10. Hertel, J.; Lindemeyer, M.; Missal, K.; Fried, C.; Tanzer, A.; Flamm, C.; Hofacker, I.L.; Stadler, P.F. The students of bioinformatics computer labs 2004 and 2005. The expansion of the metazoan microRNA repertoire. *BMC Genomics* **2006**, *7*, 1–15.
11. Sempere, L.F.; Cole, C.N.; McPeek, M.A.; Peterson, K.J. The phylogenetic distribution of metazoan microRNAs: Insights into evolutionary complexity and constraint. *J. Exp. Zool. B. Mol. Dev. Evol.* **2006**, *306*, 575–588.
12. Niwa, R.; Slack, F.J. The evolution of animal microRNA function. *Curr. Opin. Genet. Dev.* **2007**, *17*, 145–150.
13. Heimberg, A.M.; Cowper-Sal·lari, R.; Sémon, M.; Donoghue, P.C.; Peterson, K.J. MicroRNAs reveal the interrelationships of hagfish, lampreys, and gnathostomes and the nature of the ancestral vertebrate. *Proc. Natl. Acad. Sci. USA* **2010**, *107*, 19379–19383.
14. Tanzer, A.; Stadler, P.F. Molecular evolution of a microRNA cluster. *J. Mol. Biol.* **2004**, *339*, 327–335.

15. Chen, K.; Rajewsky, N. The evolution of gene regulation by transcription factors and microRNAs. *Nat. Rev. Genet.* **2007**, *8*, 93–103.
16. Berezikov, E. Evolution of microRNA diversity and regulation in animals. *Nat. Rev. Genetics* **2011**, *12*, 846–860.
17. Marz, M.; Gruber, A.R.; Höner zu Siederdissen, C.; Amman, F.; Badelt, S.; Bartschat, S.; Bernhart, S.H.; Beyer, W.; Kehr, S.; Lorenz, R.; *et al*. Animal snoRNAs and scaRNAs with Exceptional Structures. *RNA Biol.* **2011**, *8*, 938–946.
18. Dalloul, R.A.; Long, J.A.; Zimin, A.V.; Aslam, L.; Beal, K.; Blomberg, L.A.; Bouffard, P.; Burt, D.W.; Crasta, O.; Crooijmans, R.P.M.A.C.; *et al*. Multi-platform next-generation sequencing of the domestic turkey (*Meleagris gallopavo*): Genome assembly and analysis. *PLoS Biol.* **2010**, *8*, doi:10.1371/journal.pbio.1000475.
19. Will, S.; Missal, K.; Hofacker, I.L.; Stadler, P.F.; Backofen, R. Inferring non-coding RNA families and classes by means of genome-scale structure-based clustering. *PLoS Comp. Biol.* **2007**, *3*, doi:10.1371/journal.pcbi.0030065.
20. Liu, Q.; Olman, V.; Liu, H.; Ye, X.; Qiu, S.; Xu, Y. RNACluster: An integrated tool for RNA secondary structure comparison and clustering. *J. Comput. Chem.* **2008**, *29*, 1517–1526.
21. Torarinsson, E.; Havgaard, J.H.; Gorodkin, J. Multiple structural alignment and clustering of RNA sequences. *Bioinformatics* **2007**, *23*, 926–932.
22. Höchsmann, M.; Töller, T.; Giegerich, R.; Kurtz, S. Local similarity in RNA secondary structures. *Proc. IEEE Comput. Soc. Bioinform. Conf.* **2003**, *2*, 159–168.
23. Shapiro, B.A.; Zhang, K.Z. Comparing multiple RNA secondary structures using tree comparisons. *Comput. Appl. Biosci.* **1990**, *6*, 309–318.
24. Jiang, T.; Lin, G.; Ma, B.; Zhang, K. A general edit distance between RNA structures. *J. Comput. Biol.* **2002**, *9*, 371–388.
25. Hofacker, I.L.; Fekete, M.; Stadler, P.F. Secondary structure prediction for aligned RNA sequences. *J. Mol. Biol.* **2002**, *319*, 1059–1066.
26. Sankoff, D. Simultaneous solution of the RNA folding, alignment, and proto-sequence problems. *SIAM J. Appl. Math.* **1985**, *45*, 810–825.
27. Sokal, R.R.; Michener, C.D. A statistical method for evaluating systematic relationships. *Univ. Kansas Sci. Bull.* **1958**, *28*, 1409–1438.
28. Sibson, R. SLINK: An optimally efficient algorithm for the single-link cluster method. *Comput. J. (BCS)* **1973**, *16*, 30–34.
29. Sorensen, T. A method of establishing groups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on Danish commons. *Biologiske Skrifter* **1948**, *5*, 1–34.
30. Axtell, M.J.; Westholm, J.O.; Lai, E.C. Vive la différence: Biogenesis and evolution of microRNAs in plants and animals. *Genome Biol.* **2011**, *12*, doi:10.1186/gb-2011-12-4-221.
31. Li, S.C.; Shiau, C.K.; Lin, W.C. Vir-Mir db: Prediction of viral microRNA candidate hairpins. *Nucleic Acids Res.* **2008**, *36*, D184–D189.
32. Cullen, B.R. Viruses and microRNAs: RISCy interactions with serious consequences. *Genes Dev.* **2011**, *25*, 1881–1894.

33. Sharkady, S.M.; Williams, K.P. A third lineage with two-piece tmRNA. *Nucleic Acids Res.* **2004**, *32*, 4531–4538.
34. Mao, C.; Bhardwaj, K.; Sharkady, S.M.; Fish, R.I.; Driscoll, T.; Wower, J.; Zwieb, C.; Sobral, B.W.; Williams, K.P. Variations on the tmRNA gene. *RNA Biol.* **2009**, *6*, 355–361.
35. Rosenblad, M.A.; Larsen, N.; Samuelsson, T.; Zwieb, C. Kinship in the SRP RNA family. *RNA Biol.* **2009**, *6*, 508–516.
36. Piccinelli, P.; Rosenblad, M.A.; Samuelsson, T. Identification and analysis of ribonuclease P and MRP RNA in a broad range of eukaryotes. *Nucleic Acids Res.* **2005**, *33*, 4485–4495.
37. Walker, S.C.; Engelke, D.R. Ribonuclease P: The evolution of an ancient RNA enzyme. *Crit. Rev. Biochem. Mol. Biol.* **2006**, *41*, 77–102.
38. Schmitt, M.E.; Bennett, J.L.; Dairaghi, D.J.; Clayton, D.A. Secondary structure of RNase MRP RNA as predicted by phylogenetic comparison. *FASEB J.* **1993**, *7*, 208–213.
39. Woodhams, M.D.; Stadler, P.F.; Penny, D.; Collins, L.J. RNAse MRP and the RNA processing cascade in the eukaryotic ancestor. *BMC Evol. Biol.* **2007**, *7*, doi:10.1186/1471-2148-7-S1-S13.
40. Pisarev, A.V.; Shirokikh, N.E.; Hellen, C.U. Translation initiation by factor-independent binding of eukaryotic ribosomes to internal ribosomal entry sites. *C R Biologie* **2005**, *328*, 589–605.
41. Hofacker, I.L.; Fontana, W.; Stadler, P.F.; Bonhoeffer, L.S.; Tacker, M.; Schuster, P. Fast folding and comparison of RNA secondary structures. *Monatsh. Chem.* **1994**, *125*, 167–188.
42. Zhang, K.; Shasha, D. Simple fast algorithms for the editing distance between trees and related problems. *SIAM J. Comput.* **1989**, *18*, 1245–1262.
43. Lance, G.N.; Williams, W.T. A general theory of classifactory sorting strategies I. Hierarchical systems. *Comp. J.* **1967**, *9*, 373–380.

List of clusters

Cluster SNORD1: 334 families, of which 94.9% are SNORDs, contains 74.4% of all SNORDs.

Families: Hammerhead_3, SNORD56, CAESAR, SNORD26, SNORD41, SNORD93, SNORD66, snoR10, sn2429, ceN89, snoR60, SNORD107, sn3071, SNORD42, SNORD82, SNORND104, snoU49, snoMe28S-Cm788, snoU43C, SNORD50, sn2317, plasmodium_snoR24, snoR28, plasmodium_snoR30, plasmodium_snoR17, snoZ175, plasmodium_snoR28, SNORD31, SNORD102, SNORD109A, snoZ17, SNORD99, SNORD112, SNORD52, SNORD75, ceN53, ceN108, sn2841, SNORD108, ceN109, ceN111, ceN70, ceN30, ceN44, snoU54, ceN61, SNORD51, SNORD115, snoJ33, snoZ155, HIV_GSL3, SNORD72, SNORD105, snoR66, SNORD95, snoR79, sn3060, snoZ169, SNORD123, snoZ267, SNORD126, snoZ40, SNORD98, sn1185, snoR25, SNORD69, sn668, snoPyro_CD, ceN114, ceN65, SNORD44, ceN63, SNORD48, sR4, ceN54, sR24, sR58, sR45, sR60, sR20, sR53, sR22, sR40, sR32, sR47, sR13, sR38, sR36, sR48, sR39, sR16, sR14, snoU82P, sR52, sR7, sR23, sR2, sR33, sR44, sR28, sR51, sR35, sR43, sR30, sR18, sR17, sR49, sR41, sR10, sR8, sR34, SNORD37, SNORD78, DdR17, sR9, snoR101, sR42, sR15, sR11, v-snoRNA-1, DdR14, sR19, sR1, sR5, sR21, sR12, sR55, DdR1, SAM_V, snoR43, SNORD38, snoR4a, ceN22, ceN27, PRINS, SNORD83, snoR72, SCARNA18, snoMe28S-G3255, snoCD11, Lnt, SAM_alpha, PreQ1, snoZ162, Bacillus-plasmid, snoR30, SNORD36, snoR117, snoZ6, sn2903, SNORD27, snoR12, snoMe28S-Gm3113, SNORD29, snoZ165, snoU83, snoR09, SNORD61, snoU83D, snoR07, snoMe28S-Am982, snoU83C, snoR121, snoR24, SNORD57, SNORD11, snoR20a, snoU31b, snoU30, snR50, SNORD21, snoMe28S-Gm1083, SNORD35, snoR77Y, snoZ196, snoR27, snoZ173, snoR26, snoR31_Z110_Z27, snoR160, U54, snoZ163, SNORD111, snoR11, snoU83B, sn2991, snR39B, snoR16, snoR8a, SNORD25, SNORD101, sn1502, SNORD73, SNORD39, SNORD28, SNORD62, snoMe28S-Cm3227, snoR29, ceN28, snoZ168, SNORD20, SNORD45, SNORD47, snoR22, SNORD110, sn2343, SNORD74, snoZ7, ceN106, SNORD59, snoR23, snoMe28S-Cm2645, snoMe18S-Gm1358, snoR01, snoMe28S-U3344, snoZ256, snoR18, snoZ223, snoZ122, SNORD79, SNORD87, SNORD103, SNORD100, SNORD81, snoR21, SNORD127, SNORD34, snoZ185, snoZ182, SNORD11B, SNORD12, SNORD30, DdR11, SNORD65, sR3, DdR12, snoZ105, snoU18, DdR2, SNORD49, SNORD5, Afu_199, snoR4, DdR13, DdR16, snoU105B, snoR35, DdR10, DdR5, plasmodium_snoR20, SNORD19B, SNORD19, snoTBR7, SNORD96, snoR53Y, snoZ103, ceN69, snoMBII-202, SNORD113, SNORD43, snoR114, snoR20, snoR14, snoR13, snoR44_J54, snoR32_R81, snoMe28S-Am2634, snoR41, SNORD46, SNORD24, SNORD18, snoU61, snosnR57, snosnR71, snoU36a, snoMe18S-Um1356, snoR128, snR62, SNORD58, SNORD63, snoR116, snoZ159, snoR72Y, snoR31, snoR19, snoTBR5, snR78, snoZ30a, SNORD53_SNORD92, snosnR55, snR58, Afu_335, DdR7, Afu_298, Afu_455, MIR478, snR52, Afu_300, snoU43, snR77, snoR64a, DdR6, Afu_294, DdR8, snoZ161_228, snoZ43, SNORD33, snoR38, SNORD88, snoZ266, snoR71, SNORD60, snoR64, snoZ101, snoZ102_R77, SNORD91, SNORD124, SNORD125, snoZ199, snoZ278, snoU6-53, SNORD116, snoZ30, snoZ118, HOTAIR_3, SNORD70, SNORD121A, c-di-GMP-II, rli27, HSR-omega_2, RatA, msr, snoZ157, snoZ119.

Cluster SNORD2: 86 families, of which 81.4% are SNORDs, contains 16.4% of all SNORDs.

Families: SNORD14, PrfA, S15, Alpha_RBS, snoZ221_snoR21b, snoR9_plant, plasmodium_snoR14, ceN113, snoZ5, sn2417, snR13, plasmodium_snoR16, snosnR60_Z15, snoZ107_R87, snoZ206, snoZ152, snoZ188, snR76, snosnR64, snR56, Afu_304, DdR15, snoU13, Afu_191, snR79, snR51, snoU40, snR41, ceN103, snoR113, snoR130, ceN40, snosnR48, ceN33, snosnR69, snoR1, snoR17, sn2524, snoR126, snosnR66, snoR69Y, SNORD23, snosnR54, Afu_514, Afu_513, DdR4, snR73, snosnR61, snR40, Afu_264, snR47, Afu_190, snR65, snoR118, snoMe28S-Am2589, snR39, snR87, bxd_4, MALAT1, snoZ247, snoJ26, bxd_3, PhotoRC-II, RMST_4, Hammerhead_1, rimP, SNORD15, Afu_198, ceN47, SNORD22, SNORD94, snoR127, plasmodium_snoR21, snoU6-47, snR67, snR75, snR68, snoZ13_snr52, plasmodium_snoR26, snoU6-77, Afu_254, CDKN2B-AS, SAH_riboswitch, U6atac, SNORA29, SMK_box_riboswitch.

Cluster SNORA: 158 families, of which 81.0% are SNORAs, contains 57.1% of all SNORAs.

Families: Spot_42, OxyS, CC1840, CC3552, VrrA, RsaE, greA, HLE, snoR111, frnS, RsaA, rli49, SNORA71, SNORA40, SNORA4, SNORA68, SCARNA3, ceN92, SNORA25, snoF1_F2, S_pombe_snR93, SNORA42, SNORA49, S_pombe_snR92, SCARNA20, ceN49, ceN68, ceN84, SNORA11, SCARNA8, SNORA24, ceN88, ceN125, SNORA65, ceN41, SNORA36, SNORA51, SNORA56, ceN43, ceN86, SNORA31, SNORA3, ceN100, ceN80, ceN93, ceN81, ceN58, SNORA20, ceN38, ceN126, ceN48, ceN67, ceN101, ceN39, ceN105, SNORA30, ceN45, SNORA76, ceN42, ceN51, DdR18, SCARNA23, 23S-methyl, rli61, Pxr, SNORA64, SNORA7, SNORA48, SNORA84, ceN82, SNORA69, SNORA55, SNORA15, SCARNA4, SNORA79, SNORA58, SNORA35, SNORA77, ceN110, SCARNA15, SNORA47, snopsi28S-3327, snoR134, snopsi28S-1192, snoR74, snoR137, ceN102, snopsi28S-3316, SNORA67, SCARNA21, SNORA13, SNORA54, SNORA14, SNORA46, S_pombe_snR3, RUF2, SNORA28, snoR80, SNORA32, ACA64, ceN46, snoR104, S_pombe_snR33, snoR77, SCARNA11, SCARNA14, SNORA8, SNORA17, SNORA62, SNORA50, SNORA43, snoR639, SNORA1, SNORA19, SNORA33, snoU109, ceN104, snR85, SNORA63, SNORA21, SNORA41, ceN36-1, S_pombe_snR90, snopsi18S-1854, SNORA52, SNORA38, S_pombe_snR5, SNORA72, SNORA9, SNORA5, SNORA66, SNORA44, SNORA70, SNORA57, SNORA61, S_pombe_snR46, GlmY_tke1, plasmodium_snoR27, SNORA22, SNORA18, Termite-leu, snopsi18S-841, snoR03, snR49, SNORA23, snR33, snR80, snR161, S_pombe_snR10, snR9, S_pombe_snR35, snR5, S_pombe_snR42, snR81, snR189, snR3, snopsi28S-2876, NrrF.

Cluster miRNA1: 45 families, of which 53.3% are plant miRNAs, contains 43.6% of all plant miRNAs.

Families: Vault, MIR480, Y_RNA, L21_leader, MIR394, MIR390, MIR398, MIR408, mir-172, MIR171_1, MIR168, MIR812, mir-156, MIR477, MIR162_2, MIR171_2, mir-11, MIR473, MIR828, MIR529, MIR403, mir-160, mir-28, MIR397, mir-399, mir-395, mir-689, K_chan_RES, rli51, MIR475, mir-277, mir-544, mir-3017, mir-286, mir-584, mir-274, MIR474, MIR1122, mir-649, Deinococcus_Y_RNA, MIR169_5, MIR444, MIR2118, MIR530, MIR1023.

Cluster miRNA2: 472 families, of which 85.6% are animal miRNAs, contains 91.6% of all animal miRNAs.

Families: let-7, mir-284, mir-306, mir-324, mir-218, mir-449, mir-995, mir-996, mir-684, mir-24, mir-302, mir-71, mir-67, ceN74-2, rox2, mir-1224, mir-433, RNA-OUT, mir-290, mir-16, mir-33, mir-235, Rota_CRE, HHBV_epsilon, mir-877, mir-133, mir-23, mir-216, mir-146, mir-147, mir-245, potC, TB10Cs2H1, TB11Cs4H3, TB10Cs5H3, TB10Cs5H2, TB9Cs1H2, mir-384, mir-1275, mir-2, mir-103, mir-425, mir-2241, mir-17, mir-122, mir-203, mir-9, mir-221, mir-460, mir-96, mir-208, mir-181, mir-10, mir-30, F6, mir-101, mir-155, mir-308, mir-486, mir-14, mir-8, mir-46, mir-130, mir-3, mir-927, mir-92, mir-231, mir-1, mir-138, mir-19, mir-887, lin-4, mir-242, mir-874, mir-29, mir-790, mir-134, mir-361, mir-135, mir-196, mir-128, mir-499, mir-148, mir-791, mir-872, mir-920, mir-263, mir-275, mir-232, mir-600, mir-500, mir-TAR, mir-378, rox1, HBV, mir-484, mir-62, mir-572, mir-1827, GABA3, SECIS_2, Lambda_thermo, mir-BART1, mir-1829, mir-761, mir-557, mir-BART2, mir-139, Gurken, mir-BHRF1-1, mir-BHRF1-3, g2, mir-383, mir-760, mir-673, mir-1255, mir-BHRF1-2, mir-318, mir-392, mir-458, mir-5, mir-12, mir-BART20, mir-BART3, lsy-6, mir-21, mir-367, mir-675, mir-1473, mir-32, mir-299, mir-1180, mir-374, mir-668, mir-671, mir-503, mir-60, mir-1226, mir-250, mir-879, mir-875, mir-1912, mir-488, mir-281, mir-370, mir-105, mir-451, mir-883, mir-885, mir-455, mir-224, ciona-mir-92, mir-185, mir-574, mir-136, mir-197, mir-506, mir-491, mir-592, mir-61, mir-207, MIR815, mir-938, mir-149, mir-BART7, mir-941, mir-298, mir-422, mir-1287, mir-939, mir-1227, SNORD86, mir-BART15, mir-631, mir-322, mir-604, mir-708, mir-504, mir-200, mir-1280, mir-542, mir-662, mir-289, mir-52, mir-150, mir-304, mir-940, mir-331, mir-583, mir-563, mir-194, mir-541, mir-190, mir-305, mir-BART5, mir-34, mir-317, mir-153, mir-145, MIR845_1, mir-315, mir-787, mir-330, mir-676, mir-551, mir-569, mir-365, mir-489, mir-259, mir-351, mir-186, mir-581, mir-609, mir-42, mir-279, mir-50, mir-244, mir-43, MIR2587, mir-22, mir-423, mir-786, mir-191, MIR1444, mir-326, mir-230, mir-942, mir-934, mir-625, mir-944, mir-3179, mir-605, mir-1265, mir-578, mir-86, mir-253, mir-296, mir-505, mir-316, mir-452, mir-876, mir-359, mir-580, mir-548, MIR439, mir-80, mir-789, mir-90, mir-228, MIR476, mir-586, mir-576, mir-642, mir-577, mir-624, mir-651, mir-932, mir-616, mir-339, mir-335, mir-802, mir-618, mir-357, mir-246, mir-248, mir-251, mir-214, mir-552, mir-599, mir-77, mir-81, mir-75, mir-49, mir-628, mir-BART12, mir-558, mir-621, mir-340, mir-314, mir-1307, mir-597, Actino-pnp, mir-650, mir-154, mir-922, mir-1183, mir-615, mir-648, PYLIS_1, mir-640, mir-936, mir-663, MIR1428, mir-593, Tombus_IRE, mir-1207, mir-1208, mir-25, mir-937, mir-198, mir-7, mir-744, mir-632, mir-288, mir-589, mir-612, mir-765, mir-770, MIR1446, mir-63, mir-607, mir-1237, mir-767, mir-1253, mir-498, mir-192, mir-654, mir-183, mir-193, mir-239, mir-929, mir-345, mir-711, mir-320, mir-633, hvt-mir-H, mir-199, mir-450, mir-144, mir-432, mir-202, mir-556, mir-1388, mir-337, mir-549, mir-74, mir-672, mir-233, mir-434, mir-988, mir-342, mir-350, mir-360, mir-142, mir-85, mir-653, mir-83, mir-137, MIR158, MIR1846, mir-643, mir-2024, mir-BART17, mir-764, mir-987, mir-1249, MIR535, mir-268, mir-665, mir-2778, mir-652, mir-983, mir-573, mir-129, bantam, mir-210, mir-346, mir-328, mir-590, mir-582, mir-26, mir-219, mir-182, mir-338, mir-27, mir-553, mir-2774, mir-1251, mir-375, mir-463, mir-891, mir-126, mir-M7, mir-iab-4, mir-540, mir-471, mir-363, mir-692, mir-204, mir-240, mir-355, mir-73, mir-344, mir-550, mir-2807, mir-223, mir-456, mir-217, mir-287, mir-644, mir-140, mir-55, mir-234, mir-674, mir-1296, mir-254, mir-3180, mir-143, mir-70, mir-127, mir-412, mir-280, mir-639, mir-431, mir-497, mir-567, mir-575, mir-6, mir-44, mir-283, mir-276, mir-278, mir-241, mir-184, mir-282, mir-58, mir-188, mir-36, mir-64, nuoG, mir-981, mir-87, mir-358, SNORD90, mir-1178, mir-2238, mir-448, mir-48, mir-969, mir-354, mir-636, mir-492, Xist_exon4, mir-562, Parecho_CRE, rli31, mir-720, mir-454, Bacteria_small_SRP, Corona_package, Retro_dr1, mir-657, Lacto-usp, mir-626, mir-661, Termite-flg, lactis-plasmid, mir-205, mir-187, mir-598, mir-84, mir-490, mir-353, mir-255, ceN23-1, mir-2518, MIR405, mir-249, mir-999, MIR1027, MIR1222, mraW.

Cluster CRIPSR: 100 families, of which 59.0% are CRISPRs, contains 90.8% of all CRISPRs.

Families: Histone3, CRISPR-DR14, CRISPR-DR42, CRISPR-DR43, CRISPR-DR38, CRISPR-DR55, CRISPR-DR60, HOTAIR_1, cHP, sR6, RF_site1, RF_site3, CRISPR-DR32, CRISPR-DR47, CRISPR-DR62, TLS-PK6, CRISPR-DR7, CRISPR-DR64, CRISPR-DR10, CRISPR-DR5, CRISPR-DR46, CRISPR-DR33, CRISPR-DR36, CRISPR-DR30, CRISPR-DR63, CRISPR-DR18, CRISPR-DR13, CRISPR-DR24, CRISPR-DR52, CRISPR-DR19, CRISPR-DR53, CRISPR-DR59, CRISPR-DR48, RF_site8, CRISPR-DR49, RF_site9, bxd_5, REN-SRE, PK-BYV, CRISPR-DR22, eiav_FSE, fiv_FSE, CRISPR-DR58, TMV_UPD-PK3, PK1-TEV_CVMV, JUMPstart, SBWMV1_UPD-PKe, blv_FSE, BMV3_UPD-PK1, CRISPR-DR45, UPD-PKc, SBWMV2_UPD-PKl, SBWMV1_UPD-PKb, SBRMV1_UPD-PKf, SBWMV2_UPD-PKb, SBRMV1_UPD-PKd, SBWMV2_UPD-PKk, PSLVbeta_UPD-PK2, TMV_UPD-PK2, BMV3_UPD-PK3, TMV_UPD-PK1, Corona_SL-III, CRISPR-DR11, CRISPR-DR9, CRISPR-DR29, CRISPR-DR20, CRISPR-DR16, CRISPR-DR66, CRISPR-DR21, CRISPR-DR57, CRISPR-DR8, SECIS_4, CRISPR-DR2, CRISPR-DR51, CRISPR-DR3, CRISPR-DR39, CRISPR-DR44, CRISPR-DR26, CRISPR-DR37, CRISPR-DR54, UPSK, SBWMV1_UPD-PKh, UPD-PK2, UPD-PKg, UPD-PKib, CRISPR-DR4, HIV-1_SL3, HIV-1_SL4, CRISPR-DR17, CRISPR-DR25, CRISPR-DR28, TCV_Pr, CRISPR-DR35, CRISPR-DR6, CRISPR-DR27, CRISPR-DR23, CRISPR-DR61, CRISPR-DR15, CRISPR-DR56, CRISPR-DR65.

Alpha, beta and gamma values for all clans

ClanAC	ClanID	alpha	beta	gamma
CL00003	SRP	0,71428571	0,00356234	0,58360825
CL00015	CRISPR-2	0,5	0,21052632	0,99272328
CL00019	SCARNA4	0,5	0,18181818	0,97074271
CL00048	SNORD19	0,5	0,0952381	0,99365403
CL00024	SNORA5	0,5	0,07142857	0,96057036
CL00089	mir-50	0,5	0,06896552	0,98191669
CL00099	MIR171	0,5	0,04444444	0,95002992
CL00090	mir-73	0,5	0,02898551	0,97720389
CL00097	mir-BART	0,5	0,025	0,98263434
CL00007	U4	0,5	0,02439024	0,91554699
CL00092	mir-137	0,5	0,02409639	0,97694778
CL00006	U2	0,5	0,02352941	0,91341612
CL00073	SNORD100	0,5	0,02325581	0,99249899
CL00052	SNORD30	0,5	0,01851852	0,99158521
CL00029	SNORA17	0,5	0,01612903	0,95853922
CL00084	mir-3	0,5	0,01351351	0,97939706
CL00025	SNORA7	0,5	0,01282051	0,94878695
CL00047	SNORD18	0,5	0,01212121	0,98895276
CL00055	SNORD34	0,5	0,01212121	0,98895276
CL00072	SNORD96	0,5	0,01212121	0,98895276
CL00076	SNORD110	0,5	0,01212121	0,98895276
CL00008	U54	0,5	0,00615385	0,98240714
CL00050	SNORD26	0,5	0,00615385	0,98240714
CL00060	SNORD44	0,5	0,00615385	0,98240714
CL00064	SNORD58	0,5	0,00615385	0,98240714
CL00074	SNORD101	0,5	0,00615385	0,98240714
CL00075	SNORD105	0,5	0,00615385	0,98240714
CL00077	SNORND104	0,5	0,00615385	0,98240714
CL00088	mir-36	0,5	0,00503778	0,97282105
CL00091	mir-81	0,5	0,00503778	0,97282105
CL00087	mir-34	0,5	0,00451467	0,96715421
CL00094	mir-216	0,5	0,00451467	0,96715421
CL00095	mir-279	0,5	0,00451467	0,96715421
CL00010	Hammerhead	0,5	0,00434783	0,96735847
CL00049	SNORD25	0,5	0,00434783	0,96735847
CL00079	snR68	0,5	0,00434783	0,96735847
CL00081	snoU13	0,5	0,00397614	0,95562708
CL00020	SL	0,5	0,00325733	0,94515921
CL00085	mir-15	0,5	0,00243902	0,9617419
CL00016	FinP-traJ	0,5	0,00231214	0,94689386
CL00100	U3	0,5	0,0020377	0,66457625
CL00046	SNORD16	0,5	0,00133511	0,94367721
CL00070	SNORD77	0,5	0,00133511	0,94367721
CL00080	snoR53	0,5	0,00133511	0,94367721
CL00086	mir-28	0,5	0,00133511	0,94367721
CL00018	SCARNA3	0,5	0,00119976	0,93967039
CL00030	SNORA20	0,5	0,00119976	0,93967039
CL00011Glm		0,5	0,00115741	0,90999479
CL00026	SNORA8	0,5	0,00115741	0,90999479
CL00009	U6	0,5	0,00109709	0,90576014
CL00012	SAM	0,5	0,00109709	0,90576014
CL00039	SNORA56	0,5	0,00109709	0,90576014
CL00044	SNORD12	0,5	0,00109709	0,90576014
CL00059	SNORD43	0,5	0,00109709	0,90576014

Alpha, beta and gamma values for all clans

CL00043	SNORA74	0,5	0,00107411	0,87194524
CL00022	SNORA3	0,5	0,00107009	0,86744392
CL00023	SNORA4	0,5	0,00107009	0,86744392
CL00017	IRES1	0,5	0,00103896	0,83190149
CL00061	SNORD46	0,5	0,00103896	0,83190149
CL00082	snoU85	0,5	0,00103896	0,83190149
CL00098	MIR169	0,5	0,00103896	0,83190149
CL00101	Cobalamin	0,5	0,00103896	0,83190149
CL00041	SNORA64	0,5	0,0010352	0,80766785
CL00078	snR30-U17	0,5	0,00102775	0,71320381
CL00013	7SK	0,5	0,00101885	0,66457625
CL00037	SNORA48	0,5	0,00101729	0,47470574
CL00036	SNORA44	0,33333333	0,02564103	0,94878695
CL00033	SNORA28	0,33333333	0,01923077	0,94878695
CL00067	SNORD61	0,33333333	0,01734104	0,98586023
CL00056	SNORD35	0,33333333	0,00652174	0,96735847
CL00058	SNORD41	0,33333333	0,00652174	0,96735847
CL00065	SNORD59	0,33333333	0,00652174	0,96735847
CL00068	SNORD62	0,33333333	0,00652174	0,96735847
CL00071	SNORD88	0,33333333	0,00652174	0,96735847
CL00002	RNaseP	0,33333333	0,00304569	0,39855097
CL00083	mir-2	0,33333333	0,00200267	0,94367721
CL00028	SNORA13	0,33333333	0,00173611	0,90999479
CL00042	SNORA65	0,33333333	0,00164564	0,90576014
CL00062	SNORD49	0,33333333	0,00164564	0,90576014
CL00031	SNORA21	0,33333333	0,00160514	0,86744392
CL00005	U1	0,33333333	0,00152827	0,66457625
CL00004	Telomerase	0,33333333	0,00152284	0,39855097
CL00014	CRISPR-1	0,3	0,07	0,9871821
CL00096	mir-290	0,28571429	0,00487805	0,9617419
CL00045	SNORD15	0,25	0,04878049	0,97382537
CL00034	SNORA35	0,25	0,03225806	0,95853922
CL00063	SNORD52	0,25	0,01197605	0,97472972
CL00093	mir-182	0,25	0,01007557	0,97282105
CL00053	SNORD31	0,25	0,00869565	0,96735847
CL00027	SNORA9	0,25	0,00231481	0,90999479
CL00035	SNORA36	0,25	0,00231481	0,90999479
CL00032	SNORA27	0,25	0,00214018	0,86744392
CL00040	SNORA62	0,25	0,00214018	0,86744392
CL00038	SNORA52	0,22222222	0,00479904	0,93967039
CL00066	SNORD60	0,2	0,01086957	0,96735847
CL00057	SNORD39	0,16666667	0,03529412	0,98681884
CL00051	SNORD29	0,16666667	0,02391304	0,96735847
CL00069	SNORD74	0,16666667	0,00600801	0,94367721
CL00021	SNORA2	0,16666667	0,00347222	0,90999479
CL00001	tRNA	0,16666667	0,00304569	0,39855097
CL00102	group-II-D1D4	0,14285714	0,00363636	0,83190149
CL00054	SNORD33	0,125	0,0173913	0,96735847

ClanAC	ClanN	Clan members
CL00003	7	[RF00017,RF00169,RF01854,RF01855,RF01857,RF01502,RF01856]
CL00015	4	[RF01318,RF01320,RF01376,RF01377]
CL00019	2	[RF00426,RF00423]
CL00048	2	[RF00569,RF01183]
CL00024	2	[RF00392,RF01240]
CL00089	2	[RF00672,RF00824]
CL00099	2	[RF00643,RF00692]
CL00090	2	[RF00830,RF00831]
CL00097	2	[RF00866,RF00363]
CL00007	2	[RF00015,RF00618]
CL00092	2	[RF00694,RF00859]
CL00006	2	[RF00004,RF00007]
CL00073	2	[RF00609,RF00046]
CL00052	2	[RF00088,RF01283]
CL00029	2	[RF00560,RF00416]
CL00084	2	[RF00716,RF00818]
CL00025	2	[RF00409,RF01246]
CL00047	2	[RF00093,RF01159]
CL00055	2	[RF00147,RF01205]
CL00072	2	[RF00055,RF01299]
CL00076	2	[RF00610,RF01280]
CL00008	2	[RF01277,RF00206]
CL00050	2	[RF00136,RF00087]
CL00060	2	[RF00287,RF00359]
CL00064	2	[RF00151,RF00608]
CL00074	2	[RF00186,RF00339]
CL00075	2	[RF00584,RF01173]
CL00077	2	[RF00289,RF01199]
CL00088	2	[RF00685,RF00794]
CL00091	2	[RF00727,RF00728]
CL00087	2	[RF00711,RF00456]
CL00094	2	[RF00654,RF00747]
CL00095	2	[RF00754,RF00948]
CL00010	2	[RF00163,RF00008]
CL00049	2	[RF01188,RF00054]

CL00079	2	[RF01287,RF01235]
CL00081	2	[RF01210,RF00304]
CL00020	2	[RF00198,RF00199]
CL00085	2	[RF00254,RF00455]
CL00016	2	[RF00243,RF00107]
CL00100	4	[RF00012,RF01846,RF01847,RF01848]
CL00046	2	[RF00138,RF01216]
CL00070	2	[RF00309,RF00591]
CL00080	2	[RF01279,RF00338]
CL00086	2	[RF00655,RF00917]
CL00018	2	[RF00422,RF00565]
CL00030	2	[RF00429,RF00401]
CL00011Glm	2	[RF00083,RF00128]
CL00026	2	[RF00393,RF01257]
CL00009	2	[RF00026,RF00619]
CL00012	2	[RF00162,RF00634]
CL00039	2	[RF00417,RF01248]
CL00044	2	[RF00581,RF01249]
CL00059	2	[RF00221,RF01238]
CL00043	2	[RF00090,RF01263]
CL00022	2	[RF00334,RF01260]
CL00023	2	[RF00394,RF01264]
CL00017	2	[RF00061,RF00209]
CL00061	2	[RF00218,RF01259]
CL00082	2	[RF01296,RF01294]
CL00098	2	[RF00645,RF00865]
CL00101	2	[RF00174,RF01482]
CL00041	2	[RF00264,RF01267]
CL00078	2	[RF00045,RF01271]
CL00013	2	[RF00100,RF01052]
CL00037	2	[RF00554,RF01272]
CL00036	4	[RF00405,RF00418,RF01237,RF01245]
CL00033	3	[RF00400,RF00545,RF01269]
CL00067	3	[RF00270,RF01170,RF01200]
CL00056	3	[RF00211,RF01207,RF00328]
CL00058	3	[RF00274,RF00588,RF01214]

CL00065	3	[RF00273,RF00473,RF00160]
CL00068	3	[RF00153,RF00205,RF01218]
CL00071	3	[RF00604,RF01424,RF01209]
CL00002	6	[RF00010,RF00009,RF00011,RF00030,RF00373,RF01577]
CL00083	3	[RF00047,RF00143,RF00813]
CL00028	3	[RF00396,RF01255,RF01438]
CL00042	3	[RF00302,RF01292,RF01254]
CL00062	3	[RF00277,RF00337,RF01300]
CL00031	3	[RF00412,RF01258,RF01437]
CL00005	3	[RF00003,RF00548,RF00488]
CL00004	3	[RF00024,RF01050,RF00025]
CL00014	7	[RF01315,RF01317,RF01327,RF01338,RF01352,RF01379,RF01328]
CL00096	4	[RF01413,RF00639,RF00665,RF00668]
CL00045	4	[RF00067,RF01185,RF01226,RF01223]
CL00034	4	[RF00598,RF00407,RF00430,RF00566]
CL00063	4	[RF00325,RF00276,RF00333,RF01176]
CL00093	4	[RF00663,RF00706,RF00702,RF00843]
CL00053	4	[RF00089,RF00266,RF01281,RF01177]
CL00027	4	[RF00411,RF01256,RF01243,RF01436]
CL00035	4	[RF00340,RF01242,RF01262,RF01439]
CL00032	4	[RF00568,RF00443,RF01265,RF01440]
CL00040	4	[RF00091,RF01261,RF01251,RF01434]
CL00038	8	[RF00425,RF00155,RF00419,RF01224,RF01239,RF01252,RF00307,RF01435]
CL00066	5	[RF00271,RF00345,RF00527,RF00471,RF01194]
CL00057	6	[RF00571,RF00157,RF01178,RF00358,RF01181,RF00268]
CL00051	11	[RF00049,RF00476,RF01203,RF00212,RF00070,RF00592,RF00135,RF01302,RF00479,RF01198,RF00475]
CL00069	9	[RF00181,RF01169,RF00284,RF00530,RF00152,RF00357,RF01278,RF00570,RF00509]
CL00021	6	[RF00410,RF00190,RF00544,RF01253,RF01250,RF01441]
CL00001	6	[RF00005,RF00023,RF01849,RF01852,RF01851,RF01850]
CL00102	7	[RF02001,RF01998,RF02012,RF02005,RF02004,RF01999,RF02003]
CL00054	8	[RF00133,RF00134,RF00532,RF00535,RF00280,RF00472,RF01201,RF01197]

Anexo II

Article published on ISBRA 2011 [6]

Regene: Automatic Construction of a Multiple Component Dirichlet Mixture Priors Covariance Model to Identify Non-coding RNA

Felipe Lessa[1], Daniele Martins Neto[2], Kátia Guimarães[3], Marcelo Brigido[4], and Maria Emilia Walter[1]

[1] Department of Computer Science – University of Brasilia
[2] Department of Mathematics – University of Brasilia
[3] Center of Informatics – Federal University of Pernambuco
[4] Institute of Biology – University of Brasilia

Abstract Non-coding RNA (ncRNA) molecules do not code for proteins, but play important regulatory roles in cellular machinery. Recently, different computational methods have been proposed to identify and classify ncRNAs. In this work, we propose a covariance model with multiple Dirichlet mixture priors to identify ncRNAs. We introduce a tool, named Regene, to derive these priors automatically from known ncRNAs families included in Rfam. Results from experiments with 14 families improved sensitivity and specificity with respect to single component priors.

1 Introduction

The classic central dogma of molecular biology says that genetic information flows from DNA to proteins with different types of RNAs as intermediates, the DNA storing information of the genotype, and the proteins producing phenotypes. This orthodox view of the central dogma suggests that RNA is an auxiliary molecule involved in all stages of protein synthesis and gene expression. But recent research [7] have shown that some types of RNA may indeed regulate other genes and control gene expression and phenotype by themselves. Many other functions of RNAs are already known, and new functions are continuously being discovered. Roughly speaking, RNAs can be divided into two classes, mRNAs – which are translated into proteins, and non-coding RNAs (ncRNAs) – which play several important roles besides protein coding in the cellular machinery.

Supporting the biologists findings to distinguish mRNAs from ncRNAs, recently, many computational methods based on different theories and models have been proposed. It is remarkable that methods that successfully identified mRNAs, such as BLAST, in general fail when used to identify ncRNAs, although they work in some cases [13]. Examples of these models use thermodynamics [19,10], Support Vector Machine (SVM) like CONC [12], CPC [11] and PORTRAIT [2], or Self-Organizing Maps [17].

Particularly, for the problem of RNA similarity search, Eddy et al. [6] proposed a covariance model (CM) and implemented it in the tool Infernal (from *INFErence of RNA aLignment*) [15]. Infernal is used by the Rfam [8] database, which is continously growing, but contains 1,372 RNA families in its current version 9.1. Nawrocki and Eddy [14] obtained better sensitivity and specificity results including informative Dirichlet priors through the use of Bayesian inference. The principle of this theory is to combine prior information with information obtained from the sample (likelihood) to construct the posterior distribution, which synthesizes all the information. So, this method adds, in a transparent way, prior knowledge to the data. Priors play a key role in this case, since they allow to include subjective information and knowledge about the analyzed problem. Informative prior expresses specific information about a variable, and non-informative prior refers to vague and general information about this variable.

The objectives of this work are to create a new tool to automatically derive priors and to evaluate the use of Dirichlet mixture priors with more than one component in CMs. The main justification is to make more precise the identification of this CM transition distribution within empirical data. The tool *Regene* (from the character *Regene* Regette of the japanese anime Gundam 00 and *gene*) to derive the Dirichlet mixture priors from Rfam is built, as well as a module to automatically draw the CM and the parse tree diagrams.

In Section 2, we describe CMs, particularly the CM proposed by Eddy and Durbin [6] that was used in the tests. In Section 3, we first show how the Dirichlet mixture priors are estimated using the Expectation-Maximization (EM) method, and then describe the Conjugate Gradient Descent method, used to minimize the objective function. In Section 4, we present our method and show some implementation details. In Section 5, we discuss experiments with 14 Rfam families and compare the obtained results with those of Nawrocki and Eddy [14]. Finally, in Section 6, we conclude and suggest future work.

2 Covariance Models

The covariance model (CM) proposed by Eddy and Durbin [6] represents a specific family of non-coding RNAs. These CMs are a subset of the stochastic context-free grammars (SCFGs), called a "profile SCFG" [5]. The main types of production used in these CMs are $P \rightarrow aXb$ for base pairs in a stem, $L \rightarrow aX$ and $R \rightarrow Xb$ for single stranded bases, and $B \rightarrow SS$ for bifurcations that are used to separate a loop with multiple stems. Each non-terminal is called a "state", and terminals (i.e. bases of the sequences) are called "emissions", noting that this terminology has been borrowed from hidden Markov models (HMMs) [4].

These grammars are "profiles" because states are mechanically constructed from the secondary structure consensus. A "guide tree" is then built, where the nodes represent single stranded bases, base pairs, and loops with multiple stems.

For example, in Figure 1 we present the consensus secondary structure of a fictitious family of tRNAs and the generated data structures. In this figure, `MATP` nodes represent both sides of a stem, and two `BIF` nodes are used to rep-

resent the internal loop. See Eddy [5] for a detailed explanation about the CM building process. We note that the guide tree diagrams shown in this figure were automatically created by our Regene tool (`regene-diagrams` module).

We now explain how the model is used. Each state may emit zero, one or two bases and indicate a set of allowed transitions. We generate RNA sequences from the model by "walking" through the states, collecting its emissions. Each "walk" is called a *parse tree*.

Parse trees may be used to align a new sequence to an existing CM. Although there are possibly many parse trees that emit a given sequence, each parse tree has an associated probability, which allows us to find the tree presenting the highest probability. However, in this paper we are more interested in a particular type of a parse tree, named *fake parse tree* [1]. It is constructed by walking over consensus-representing states, not including probability values, and going to insertion or deletion states only when that column of the sequence is not in the consensus. In Figure 2 we show the diagrams of fake parse trees constructed from the CM shown in Figure 1.

In order to calculate the probabilities of a CM, a fake parse tree is created for each one of the sequences in the multiple alignment. The transitions and emissions generated in each state of all parse trees are counted, and these counts are then converted into probabilites using the posterior Dirichlet mixture probabilites as described in Sections 3 and 4.

Fake parse trees are used to obtain the Dirichlet mixture priors in a first step. For every family of the corresponding set used for training, a CM and its fake parse trees are constructed. Instead of observing transition counts of individual states, transitions of a particular type are counted [14]. Each type defines the guide tree node, the origin state of a transition, and its destination node. For example, transitions of type "`MATL/D` $\rightarrow$ `MATR`" may go to `IL`, `MR` or `D` state type. Each family count is then used as a training vector for the corresponding Dirichlet mixture.

3 Dirichlet Mixture Priors

To identify ncRNAs using CMs, it is essential to estimate the transition parameters. In Infernal [15], transition parameters are mean posterior estimates, combining observed counts from an input that is a RNA multiple alignment with an informative Dirichlet prior. Nawrocki and Eddy [14] use this prior through a Dirichlet mixture with a single component, based on the fact that they accurately predict target subsequence lengths, mainly when there are few query alignments sequences. The use of informative priors is more efficient for a small number of observations. Considering this, and with the purpose of facilitating the identification of transitions within empirical data, we use as informative prior a mixture of Dirichlet densities with more than one component. Sjölander et al. [18] has used such a tool for estimating amino acids distributions. In this work, we estimate the transition probabilities of a CM in the process of identifying ncRNAs.

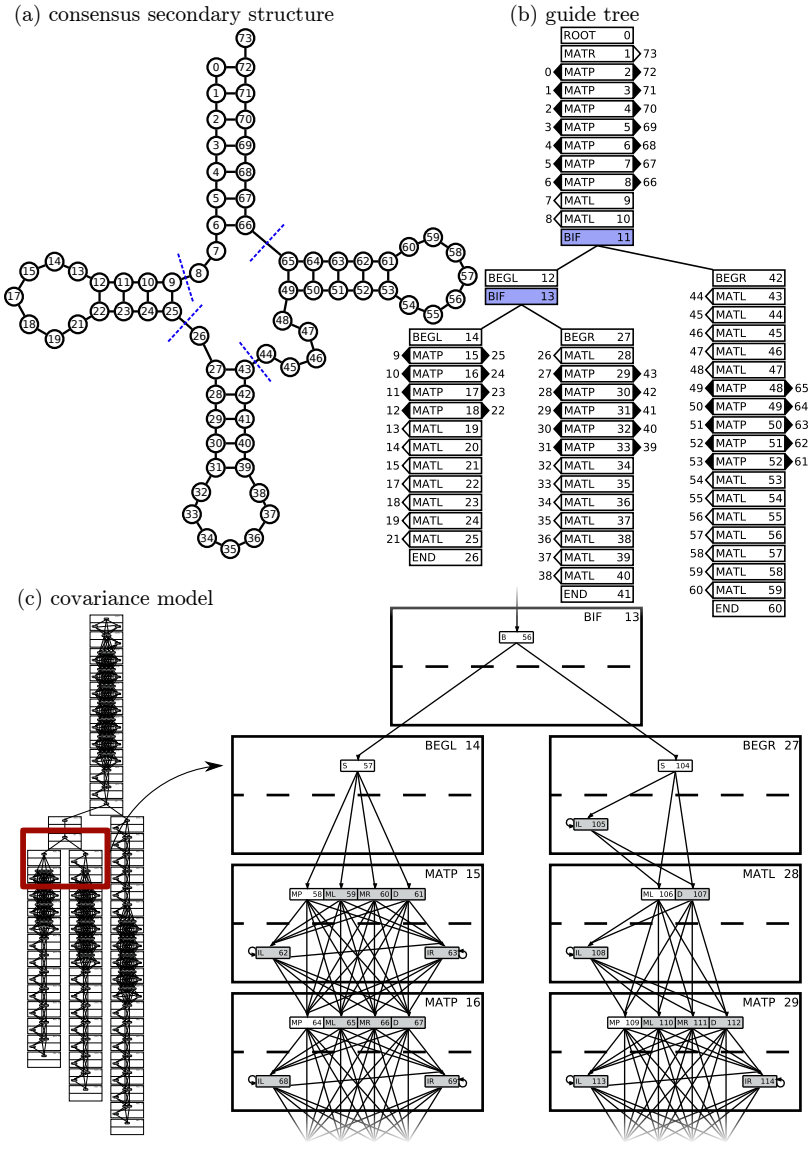


Figure 1. Diagrams of a fictitious tRNA family extracted from the `trna.5.sto` file of Infernal's tutorial, noting that (b) and (c) were automatically drawn by Regene. (a) The consensus secondary structure. Dashed lines show bifurcations created by `BIF` nodes to separate stems. (b) The corresponding guide tree. (c) The CM created from this guide tree.

(a) Multiple sequence alignment

```
# STOCKHOLM 1.0
tRNA1         GCGGAUUUAGCUCAGUUGGG.AGAGCGCCAGACUGAAGAUCUGGAGGUCCUGUGUUCGAUCCACAGAAUUCGCA
tRNA2         UCCGAUAUAGUGUAAC.GGCUAUCACAUCACGCUUUCACCGUGGAGA.CCGGGGUUCGACUCCCCGUAUCGGAG
tRNA3         UCCGUGAUAGUUUAAU.GGUCAGAAUGGGCGCUUGUCGCGUGCCAGA.UCGGGGUUCAAUUCCCCGUCGCGGAG
tRNA4         GCUCGUAUGGCGCAGU.GGU.AGCGCAGCAGAUUGCAAAUCUGUUGGUCCUUAGUUCGAUCCUGAGUGCGAGCU
tRNA5         GGGCACAUGGCGCAGUUGGU.AGCGCGCUUCCCUUGCAAGGAAGAGGUCAUCGGUUCGAUUCCGGUUGCGUCCA
#=GC SS_cons <<<<<<<..<<<<.........>>>>.<<<<<.......>>>>>.....<<<<<.......>>>>>>>>>>>>.
//
```

(b) Parse trees

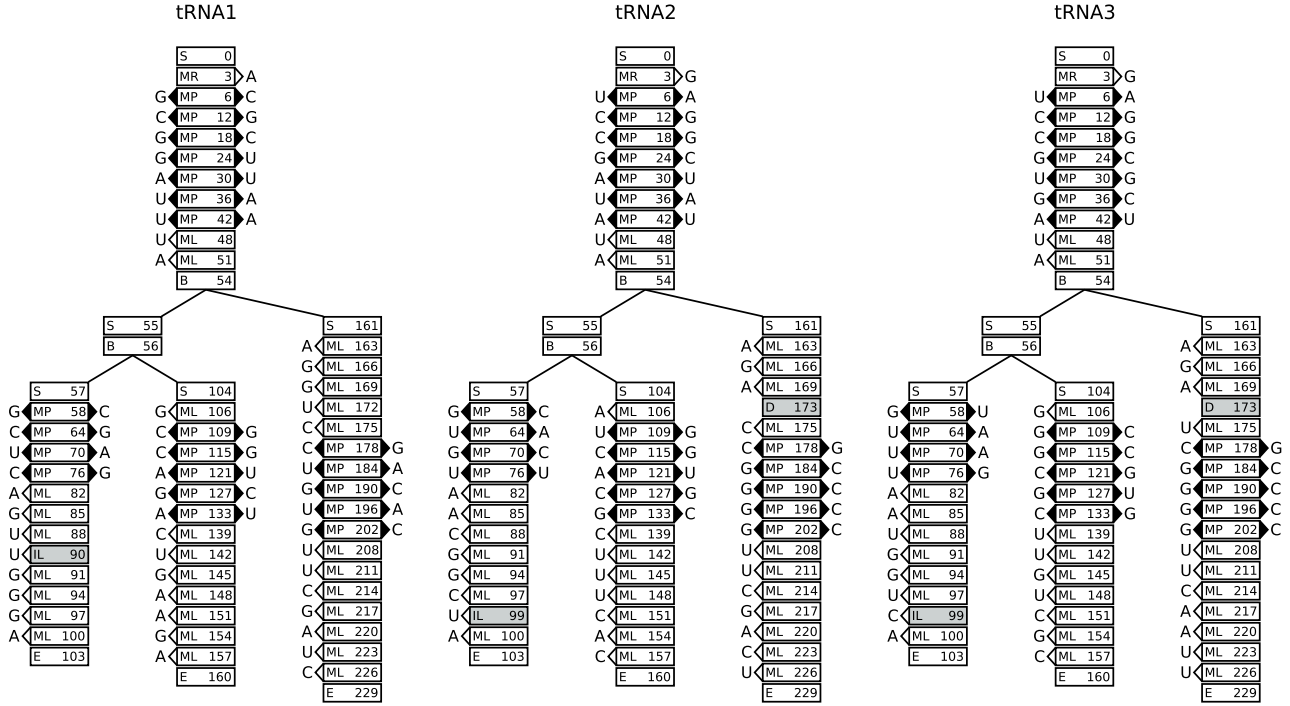


Figure 2. (a) Multiple sequence alignment of the CM shown in Figure 1. (b) Diagrams of the fake parse trees for the three sequences, tRNA1, tRNA2 and tRNA3, created by Regene (`regene-diagrams` module).

Following we present the model and the structure used to obtain the transition estimates of a CM, and after that the `CG_DESCENT` method to minimize the objective function.

3.1 Obtaining the Mixture Prior

Let $\mathbf{N} = (N_1, \ldots, N_m)$ be a random vector of transition counts, where N_i is a random number of times that each i^{th} transition occurs. Assume that $\mathbf{N}$ is multinomial distributed with a parameter $\mathbf{p}$. In this case, $\mathbf{p} = (p_1, \ldots, p_m)$ is the vector of transition probabilities to be estimated, where p_i is the i^{th} transition probability, and m is the fixed number of transitions.

The Bayesian approach is used, with some prior knowledge of $\mathbf{p}$, to estimate the transition parameters $p_1, \ldots, p_m$. In this paper, we consider that the prior density of $\mathbf{p}$, denoted by ρ, is the following Dirichlet mixture density:

$$\rho = q_1\rho_1 + \cdots + q_l\rho_l\,, \tag{1}$$

where each ρ_j is a single Dirichlet density with parameter $\boldsymbol{\alpha}_j = (\alpha_{j1}, \ldots, \alpha_{jm})$, with $\alpha_{ji} > 0$, $i = 1, \ldots, m$, $q_j > 0$, and $\sum_{j=1}^{l} q_j = 1$. The ρ_j densities are called *mixture components*, and the q_j values are called *mixture coefficients*. The entire set of parameters defining a prior in the case of a mixture, which are obtained from the Rfam database, is $\Theta = (\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_l, q_1, \ldots, q_l)$.

The $\widehat{p}_i$ estimated probability of the i^{th} transition is the mean posterior estimate, given by:

$$\widehat{p}_i = E(p_i \mid \mathbf{N} = \mathbf{n}) = \sum_{j=1}^{l} P\left(\boldsymbol{\alpha}_j \mid \mathbf{N} = \mathbf{n}\right) \frac{\alpha_{ji} + n_i}{\alpha_{j0} + n_0}\,, \tag{2}$$

where $n_0 = \sum_{i=1}^{m} n_i$, $\alpha_{j0} = \sum_{i=1}^{m} \alpha_{ji}$, and $\mathbf{n} = (n_1, \ldots, n_m)$ is the data observed from variable $\mathbf{N}$. Probability $P\left(\boldsymbol{\alpha}_j \mid \mathbf{N} = \mathbf{n}\right)$, calculated from the Bayes rule, with $P\left(\mathbf{N} = \mathbf{n} \mid \Theta, n_0\right) = \sum_{k=1}^{l} q_k P\left(\mathbf{N} = \mathbf{n} \mid \boldsymbol{\alpha}_k, n_0\right)$, is:

$$P\left(\boldsymbol{\alpha}_j \mid \mathbf{N} = \mathbf{n}\right) = \frac{q_j P\left(\mathbf{N} = \mathbf{n} \mid \boldsymbol{\alpha}_j, n_0\right)}{P\left(\mathbf{N} = \mathbf{n} \mid \Theta, n_0\right)}\,. \tag{3}$$

From the model, and taking $\Gamma\left(.\right)$ as the Gamma function, we have:

$$P(\mathbf{N} = \mathbf{n} \mid \boldsymbol{\alpha}_j, n_0) = \frac{\Gamma(n_0 + 1)\Gamma\left(\boldsymbol{\alpha}_{j0}\right)}{\Gamma(n_0 + \boldsymbol{\alpha}_{j0})} \prod_{i=1}^{m} \frac{\Gamma(n_i + \alpha_{ji})}{\Gamma(n_i + 1)\Gamma(\alpha_{ji})}\,. \tag{4}$$

Note that, in the case of a single-component density ($l = 1$), we have $\Theta = \boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_m)$ and the estimate for the i^{th} transition probability is:

$$\widehat{p}_i = E(p_i \mid \mathbf{N} = \mathbf{n}) = \frac{\alpha_i + n_i}{\alpha_0 + n_0}\,. \tag{5}$$

In this context, to introduce prior knowledge about $\mathbf{p}$ through the density ρ, it is necessary to inform about Θ. Therefore, we have used the maximum

likelihood estimator for Θ, observing the count vectors of the ncRNA families of the Rfam database.

Given a set of r fake parse trees, which are constructed from r Rfam families, we get a count vector of transitions for each family, based on the multiple alignment obtained for each family. The result is a set of count vectors $\mathbf{n_1} = (n_{11}, \ldots, n_{1m}), \ldots, \mathbf{n_r} = (n_{r1}, \ldots, n_{rm})$, where n_{ji} is the number of times that the i^{th} transition occurs in the j^{th} fake parse tree.

Our objective is to estimate, using maximum likelihood, the parameters of mixture priors obtained from the set of count vectors. Therefore, we want to find Θ that maximizes $\prod_{t=1}^{r} P(\mathbf{N} = \mathbf{n}_t \mid \Theta, n_{t0})$, where $n_{t0} = \sum_{i=1}^{m} n_{ti}$, or equivalently, Θ that minimizes the following *objective function*:

$$f(\Theta) = -\sum_{t=1}^{r} \log P(\mathbf{N} = \mathbf{n}_t \mid \Theta, n_{t0}) \; . \tag{6}$$

Next, we present the procedure to estimate the parameters of the mixture priors, using the EM (expectation-maximization) algorithm, which has been shown to be efficient to find the maximum likelihood estimate in such cases (see Duda and Hart [3]).

In a mixture density, there are two sets of parameters, $\boldsymbol{\alpha}_j$ and q_j, $j = 1, \ldots, l$, that are jointly estimated in an iterative two steps process. First one is estimated, keeping the other fixed, and then the process is reverted. Next, we present two procedures to separately estimate each set of parameters. For more details, see Sjölander et al. [18].

Estimating the $\boldsymbol{\alpha}$ Parameters The α_{ji} parameters are strictly positive and we use the `CG_DESCENT` method (see Section 3.2) to estimate them via EM. In this case, a reparametrization is needed, which takes $\alpha_{ji} = e^{w_{ji}}$, where w_{ji} is a real number with no restrictions. Thus, the partial derivate of the objective function with respect to w_{ji}, taking $\Psi(.) = \frac{\Gamma'(.)}{\Gamma(.)}$ as the digamma function, is:

$$\frac{\partial f(\Theta)}{\partial w_{ji}} = -\sum_{t=1}^{r} \frac{\partial \log P(\mathbf{N} = \mathbf{n}_t \mid \Theta, n_{t0})}{\partial \alpha_{ji}} \alpha_{ji} \tag{7}$$

$$= -\sum_{t=1}^{r} \alpha_{ji} \left[\Psi(\alpha_{j0}) - \Psi(n_{t0} + \alpha_{j0}) + \Psi(n_{ti} + \alpha_{ji}) - \Psi(\alpha_{ji}) \right] . \tag{8}$$

Estimating the q Parameters To estimate the mixture coefficients q_j, $j = 1, \ldots, l$, which must be non-negative and sum to 1, a reparametrization is also needed. Let $q_j = \frac{Q_j}{Q_0}$, where Q_j is strictly positive and $Q_0 = \sum_j Q_j$. The partial derivate of the objective funtion with respect to Q_j is:

$$\frac{\partial f(\Theta)}{\partial Q_j} = -\sum_{t=1}^{r} \frac{\partial \log P(\mathbf{N} = \mathbf{n}_t \mid \Theta, n_{t0})}{\partial Q_j} = \frac{m}{Q_0} - \frac{\sum_{t=1}^{r} P(\boldsymbol{\alpha}_j \mid \mathbf{N} = \mathbf{n}_t)}{Q_j} \; . \tag{9}$$

It follows that the maximum likelihood estimate of Q_j is:

$$\widehat{Q}_j = \frac{Q_0}{r} \sum_{t=1}^{r} P\left(\boldsymbol{\alpha}_j \mid \mathbf{N} = \mathbf{n}_t\right) , \tag{10}$$

and hence the estimate for q_j is:

$$\widehat{q}_j = \frac{Q_j}{Q_0} = \frac{1}{r} \sum_{t=1}^{r} P\left(\boldsymbol{\alpha}_j \mid \mathbf{N} = \mathbf{n}_t\right) . \tag{11}$$

3.2 Conjugate Gradient Method

EM method requires minimization of the objective function. Since the objective function is not simple, analytic methods are not good choices to compute its minimum. Instead, we compute its gradient and iteratively optimize it.

There are many different ways of optimizing a nonlinear function. Sjölander et al. [18] suggested a gradient descent method, while Nawrocki used a nonlinear conjugate gradient method with the Polak-Ribière formula [14]. We used the nonlinear conjugate gradient method proposed by Hager and Zhang [9], called `CG_DESCENT`. This relatively new method was chosen for two reasons. First, it performs well on a set of different nonlinear functions. The other advantage is that the `CG_DESCENT` 3.0 library, written by the authors, is robust, fast and well-tested.

4 Multiple Component Dirichlet Mixture Priors

We first present our method and then important implementation details.

4.1 Regene Method

We created the new Regene tool that combines the techniques described in the previous sections into a complete program that maps a set of ncRNA families into a Dirichlet mixture. Particularly, we used the EM method for Dirichlet mixtures and the `CG_DESCENT` method (Section 3), and counted the transitions of CMs built from those ncRNA families (Section 2). Note that we did not estimate the emission priors, but the transition priors. We used the same emission priors from Nawrocki and Eddy [14].

We first inform the objective function and its gradient as a black box to the `CG_DESCENT` method, that implements the EM method. EM method was used with our countings in a similar way. Since each counting has a meaning, we used these countings as training sequences for the EM method. We did not need to code any knowledge about CMs in our EM method, nor change our counting method to estimate the Dirichlet mixtures. In a nutshell, we separately implemented each module, and combined them through a pipeline (Figure 3).

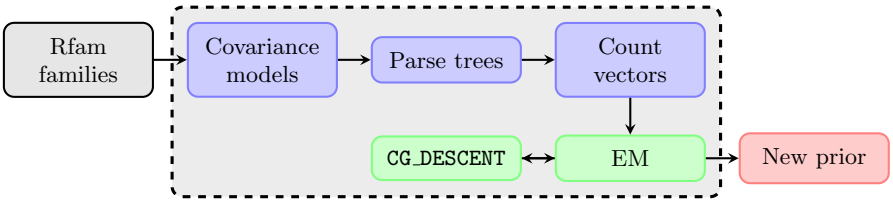


Figure 3. The Regene tool pipeline to derive a multiple component Dirichlet mixture prior. The new prior is used to construct CMs.

4.2 Regene Implementation Details

The pipeline of Figure 3 was developed in 5,900 source lines of code in the Haskell language. Bindings were created to use the `CG_DESCENT` 3.0 library with Haskell functions, which were packed with the name `nonlinear-optimization`.

The EM method was implemented into a generic library that may be used for any kind of Dirichlet density or mixture. It is now hardcoded to use the `CG_DESCENT` method, although another method could be easily used as well. The library is packed into the `statistics-dirichlet` library.

Counting the transitions is somewhat more involved, as there are many details to be considered. Infernal has a non documented option for printing the counts of a given family, as clarified by personal communication with E. Nawrocki. We reimplemented whatever needed into the `regene` library, from parsers of the Stockholm file format (used to store the ncRNA families' data) to CM data structures. Using the library, we created the `regene-priori` program that glues everything together.

As a nice result of our reimplementation of the data structures, we also created the `regene-diagrams` program. It may be used to create diagrams of guide trees, CMs (in a flat or in a graph-like view) and fake parse trees. Besides having an easy-to-use graphical user interface, it also has a console interface that may be used to automatically create diagrams from other programs, such as a website. Our code is available in the Hackage's public repository under the free GNU Public License (GPL) [16].

Unfortunately, the objective function did not behave very well in all cases. Sometimes, `CG_DESCENT` would find solutions with high α value, especially when dealing with a high number of components. High α values imply that data from the specific family would not be used. Our implementation allows to discard values above a given threshold. In our experiments, we used threshold 10^4.

5 Results and Discussion

To measure the sensitivity and specificity of our priors, we used the same `cmark-1` benchmark used by Nawrocki and Eddy [14], which is included in Infernal. From the seed alignments of Rfam 7.0, they selected 51 families, from which we used 14 families in our tests. These families were chosen due to their different sequence

Table 1. Minimum error rate (MER) calculated from our tests. Column "#QSs" represent the number of query sequences, column "N" is the MER using Nawrocki's prior. Columns "A1", "A2", "A3", "A4", "A5" and "A10" show the MER using our priors constructed from Rfam 6.1 data (same data as Nawrocki's) with 1, 2, 3, 4, 5 and 10 components, respectively. Columns "B1" to "B10" show the MER using priors constructed from Rfam 9.1 data with 1 to 10 components.

Rfam 7.0 family			Using Rfam 6.1						Using Rfam 9.1					
ID	Name	#QSs	N	A1	A2	A3	A4	A5	A10	B1	B2	B3	B4	B5	B10
RF00004	U2	76	0	0	0	0	0	0	0	0	0	0	0	0	0
RF00009	RNaseP_nuc	26	19	19	19	19	19	19	19	19	19	19	19	19	19
RF00011	RNaseP_bact_b	30	0	0	0	0	0	0	0	0	0	0	0	0	0
RF00017	SRP_euk_arch	28	7	6	6	6	6	6	6	9	9	8	9	9	9
RF00023	tmRNA	19	12	11	13	11	11	11	11	11	11	11	11	11	11
RF00029	Intron_gpII	7	2	1	1	1	1	1	1	2	1	2	2	2	1
RF00030	RNase_MRP	18	3	3	3	3	3	3	3	3	3	3	3	3	3
RF00031	SECIS	11	16	17	17	17	16	17	17	19	19	19	19	19	19
RF00037	IRE	36	1	1	1	1	1	1	1	1	1	1	1	1	1
RF00168	Lysine	33	0	0	0	0	0	0	0	0	0	0	0	0	0
RF00174	Cobalamin	87	0	0	0	0	0	0	0	0	0	0	0	0	0
RF00177	SSU_rRNA_5	145	3	4	4	4	4	4	4	4	4	4	4	0	4
RF00234	glmS	8	0	0	0	0	0	0	0	0	0	0	0	0	0
RF00448	IRES_EBNA	7	1	1	1	1	1	1	1	1	1	1	1	1	1
Summed across all families			64	63	65	63	62	63	63	69	68	68	69	65	68
Summary MER statistics			78	76	79	78	76	74	76	79	78	78	79	77	79

lengths and distinct biological functions. They also used some sequences to create CMs for the benchmark, while others were used as test sequences. Those test sequences were inserted into a 1 Mb pseudo-genome with independent and identically distributed background distribution of bases. The test consists in the use of the CMs to find those test sequences. Results were reported as minimum error rate (MER), the sum of the number of false positives and false negatives for the best found threshold. We did exactly the same steps to realize our tests.

But while Nawrocki and Eddy [14] used Infernal v0.72 bit scores, we used Infernal v1.0 E-values, and then we had to "calibrate" the CMs. Therefore, our results were slightly different, although using the same data. E-values were chosen since they present the best criterion regarding to statistical significance [1]. Besides, biologists commonly use E-values when analyzing the Infernal results.

The results for Nawrocki's prior and twelve priors created by `regene-priori` are presented in Table 1. In the line "Summary MER statistics", note that, among the priors constructed with Rfam 6.1 data, we reach the optimal point with 5 components (column "A5", presenting the lowest "Summary MER Statistics"), which is the best prior data that we tested. Priors constructed using Rfam 9.1 data did not behave very well in the tests, with the best one having 5 components (column "B5"). Perhaps that could be explained by the fact that the tests used the Rfam 7.0 families.

Figure 4 shows two ROC curves for the most representative priors, as indicated in the "Summary MER Statistics" line of Table 1. ROC curves show how the sensitivity increases while the specificity decreases. The right part of the graphics show that high specificity leads to low sensitivity, while the left part shows that most sequences can be found with low specificity. Figure 4(a) shows that multiple component mixtures improve sensitivity and specificity relative to the single component mixture. Note that the mixture with 10 components (A10) does not improve the mixture with 5 components (A5), so using more than 5 components is not necessary. Besides, our 5 component prior slightly improved Nawrocki's single component prior (N). However, Figure 4(b) shows that our single component prior (A1) is worse when compared to the one obtained by Nawrocki, although both priors were constructed from the same Rfam 6.1 data. This result could be explained by the fact that the estimation methods are different. We believe that using more than one component in the method used by Nawrocki and Eddy [14] would improve their results.

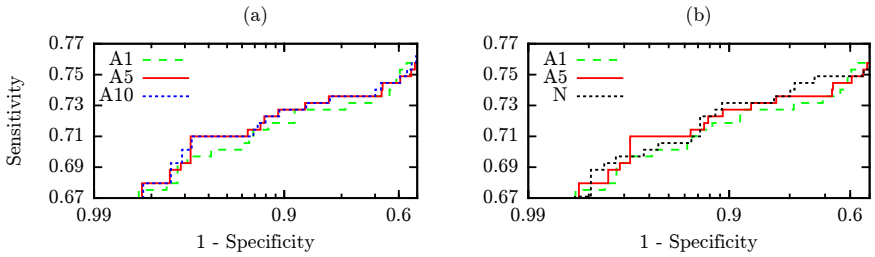


Figure 4. ROC curves for representatives priors. (a) Comparisons of our priors A1, A5 and A10. (b) Comparisons of our A1 and A5 priors with Nawrocki's (N).

6 Conclusions and Future Work

In this work we created a new tool to automatically derive priors and evaluated the use of Dirichlet mixture priors with more than one component in CMs. We built a tool named *Regene* to automatically derive the priors from Rfam data, as well as a module to automatically draw guide trees, Eddy's CMs (in a flat or in a graph-like view) and fake parse trees. Experiments were done with 14 Rfam families, comparing 13 different priors – with 1, 2, 3, 4, 5 and 10 components using Rfam 6.1 and Rfam 9.1 data, and with Nawrocki's prior with 1 component. The use of multiple components improved sensitivity and specificity.

We plan to run experiments using Rfam 9.1 instead of Rfam 7.0, and also to extend the experiments to more families. We will also look into other estimation methods such as Monte-Carlo EM, which could improve the results.

Acknowledgements

FL, DMN and MEW thank Brazilian sponsoring agency FINEP, and KG, MB and MEW thank Brazilian sponsoring agency CNPq for financial support.

References

1. The Infernal's user guide. Available at `http://infernal.janelia.org/`.
2. R. Arrial, R. Togawa, and M. Brigido. Screening non-coding RNAs in transcriptomes from neglected species using PORTRAIT: case study of the pathogenic fungus Paracoccidioides brasiliensis. *BMC Bioinformatics*, 10:239, 2009.
3. R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis.* Wiley, 1973.
4. S. R. Eddy. Profile hidden Markov models. *Bioinformatics*, 14(9):755–763, 1998.
5. S. R. Eddy. A memory-efficient dynamic programming algorithm for optimal alignment of a sequence to an RNA secondary structure. *BMC Bioinformatics*, 3:18+, 2002.
6. S. R. Eddy and R. Durbin. RNA sequence analysis using covariance models. *Nucleic acids research*, 22(11):2079–2088, 1994.
7. S. Griffiths-Jones. Annotating Noncoding RNA Genes. *Annu. Rev. Genomics Hum. Genet*, 8:279–298, 2007.
8. S. Griffiths-Jones, S. Moxon, M. Marshall, A. Khanna1, S. R. Eddy, and A. Bateman. Rfam: annotating non-coding RNAs in complete genomes. *Nucleic Acids Research.*, 33:D121–D124, 2005. Rfam database available in: http://www.sanger.ac.uk/Software/Rfam/.
9. W. W. Hager and H. Zhang. A new conjugate gradient method with guaranteed descent and an efficient line search. *SIAM J. on Optimization*, 16(1):170–192, 2005.
10. I. L. Hofacker, M. Fekete, and P. F. Stadler. Secondary Structure Prediction for Aligned RNA Sequences. *Journal of Molecular Biology*, 319(5):1059–1066, 2002.
11. L. Kong, Y. Zhang, Z-Q Ye, X-O Liu, S-O Zhao, L. Wei, and G. Gao. CPC: assess the protein-coding potential of transcripts using sequence features and support vector machine. *Nucleic Acids Res.*, 35:345–349, 2007.
12. J. Liu, J. Gough, and B. Rost. Distinguishing protein-coding from non-coding RNAs through Support Vector Machines. *PLoS Genet.*, 2(4):e29–e36, Apr 2006.
13. S. M. Mount, V. Gotea, C. F. Lin, K. Hernandez, and W. Makalowski. Spliceosomal Small Nuclear RNA Genes in Eleven Insect Genomes. *RNA*, 13:5–14, 2007.
14. E. P. Nawrocki and S. R. Eddy. Query-Dependent Banding (QDB) for Faster RNA Similarity Searches. *PLoS Computational Biology*, 3(3):e56, March 2007.
15. E. P. Nawrocki, D. L. Kolbe, and S. R Eddy. Infernal 1.0: Inference of RNA alignments. *Bioinformatics*, 25:1335–1337, 2009.
16. Regene. http://regene.exatas.unb.br.
17. T. C. Silva and et al. SOM-PORTRAIT: Identifying Non-coding RNAs Using Self-Organizing Maps. In *4th Brazilian Symposium on Bioinformatics - BSB 2009*, volume 5676 of *Lecture Notes in Computer Science*, pages 73–85. Springer, 2009.
18. Sjölander and et al. Dirichlet mixtures: a method for improved detection of weak but significant protein sequence homology. *Computer Applications in the Biosciences*, 12(4):327–345, 1996.
19. M. Zucker, D. H. Matthews, and D. H. Turner. *Algorithms and thermodynamics for RNA secondary structure prediction: A practical guide. In RNA Biochemistry and Biotechnology, NATO ASI Series.* Kluwer Academic, 1999. pp. 11-43.