

Methods in
Molecular Biology 1269

Springer Protocols

Ernesto Picardi *Editor*



RNA Bioinformatics

EXTRA
MATERIALS
springerlink.com



Humana Press

METHODS IN MOLECULAR BIOLOGY

Series Editor

John M. Walker

School of Life and Medical Sciences

University of Hertfordshire

Hatfield, Hertfordshire, AL10 9AB, UK

For further volumes:
<http://www.springer.com/series/7651>

RNA Bioinformatics

Edited by

Ernesto Picardi

*Department of Biosciences, Biotechnology and Biopharmaceutics,
University of Bari, Bari, Italy*



Editor

Ernesto Picardi
Department of Biosciences
Biotechnology and Biopharmaceutics
University of Bari
Bari, Italy

Institute of Biomembranes and Bioenergetics
National Research Council
Bari, Italy

National Institute of Biostructures and Biosystems (INBB)
Rome, Italy

ISSN 1064-3745 ISSN 1940-6029 (electronic)
ISBN 978-1-4939-2290-1 ISBN 978-1-4939-2291-8 (eBook)
DOI 10.1007/978-1-4939-2291-8
Springer New York Heidelberg Dordrecht London

Library of Congress Control Number: 2014958476

© Springer Science+Business Media New York 2015

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Printed on acid-free paper

Humana Press is a brand of Springer
Springer is part of Springer Science+Business Media (www.springer.com)

Preface

RNA is a versatile nucleic acid polymer with a structure analogous to single-stranded DNA even though it has a backbone composed of ribose sugar and the organic base thymine (T) is replaced by uracil (U). In contrast to DNA, RNA molecules are less stable and characterized by secondary and tertiary structures, which underline their different function. According to the central dogma of molecular biology, RNA is directly involved in the flux of genetic information from the DNA to the proteins. However, recent developments in molecular biology indicate that RNA molecules have a plethora of functional roles and are indispensable for living organisms and cell homeostasis. Indeed, on the basis of their functions, RNA molecules can be divided into two groups, coding and noncoding. While the coding fraction is represented by messenger RNAs (mRNAs), noncoding RNAs (ncRNAs) include at least two main classes: (1) structural ncRNAs as transfer RNAs (tRNAs), ribosomal RNAs (rRNAs), and small nucleolar RNAs (snoRNAs); (2) regulatory ncRNAs as micro RNAs (miRNAs), piwi-interacting RNAs (piwiRNAs), and long noncoding RNAs (lncRNAs).

A large fraction of eukaryotic transcriptomes consists of ncRNAs that play relevant biological roles as the regulation of gene expression in normal as well as pathological conditions. NcRNA functions are generally mediated by interactions with other RNA molecules or DNA regions or proteins. In all cases, RNA secondary and tertiary structures are basic for a correct function and interaction. However, the characterization of RNA molecules is a challenging task and, thus, during past decades a variety of bioinformatics tools have been developed. Nowadays, RNA bioinformatics represents one of the most active fields of bioinformatics and computational biology.

The interest towards RNA bioinformatics has increased rapidly thanks to the advent of recent high-throughput sequencing technologies that enable the investigation of complete transcriptomes at single nucleotide resolution.

The present book has been conceived with the aim of providing an overview of RNA bioinformatics methodologies, starting from “classical” technologies to predict secondary and tertiary structures, to novel strategies and algorithms based on massive RNA sequencing. Indeed, the content of the book is organized as follows:

- Part I—RNA secondary and tertiary structures. This section includes chapters devoted to main computational algorithms to predict, draw, and edit secondary and tertiary RNA structures or infer RNA-RNA interactions.
- Part II—Analysis of high-throughput RNA sequencing data. The aim of this section is to provide a global overview of current methodologies to handle and analyze large sequencing dataset generated by next-generation sequencing (NGS) technologies. Indeed, the section includes chapters about quality control of RNA sequencing data or the mapping of RNA-Seq reads on complete reference genomes or the gene expression profiling. In addition, methodologies to investigate posttranscriptional events as alternative splicing or RNA editing and entire meta-transcriptomes are also described in detail.

- Part III—Web resources for RNA data analysis. Finally, this section provides chapters about several available web resources to work with RNA data without specific computer requirements (hardware and software) or specialized bioinformatics skills.

Hoping that the book content meets the reader expectations, I would like to acknowledge those who helped make this book possible: all chapter authors for their work and excellent contributions; the Series Editor, John Walker, for his constant support and suggestions; my wife Angela and daughter Adele for their patience and encouragement.

A special thanks is addressed to my mentor Graziano Pesole for his always indispensable paternal suggestions.

Finally, I would to dedicate this effort to my parents since they have always believed in me and to the memory of my first supervisor Carla Quagliariello who introduced me for the first time to the wonderful and fascinating world of RNA bioinformatics.

Bari, Italy

Ernesto Picardi

Contents

Preface	v
Contributors	ix

PART I RNA SECONDARY AND TERTIARY STRUCTURES

1 Free Energy Minimization to Predict RNA Secondary Structures and Computational RNA Design.	3
<i>Alexander Churkin, Lina Weinbrand, and Danny Barash</i>	
2 RNA Secondary Structure Prediction from Multi-aligned Sequences	17
<i>Michiaki Hamada</i>	
3 A Simple Protocol for the Inference of RNA Global Pairwise Alignments	39
<i>Eugenio Mattei, Manuela Helmer-Citterich, and Fabrizio Ferrè</i>	
4 De Novo Secondary Structure Motif Discovery Using RNAProfile	49
<i>Federico Zambelli and Giulio Pavesi</i>	
5 Drawing and Editing the Secondary Structure(s) of RNA	63
<i>Yann Ponty and Fabrice Leclerc</i>	
6 Modeling and Predicting RNA Three-Dimensional Structures.....	101
<i>Jérôme Waldspühl and Vladimir Reinharz</i>	
7 Fast Prediction of RNA–RNA Interaction Using Heuristic Algorithm	123
<i>Soheila Montaseri</i>	

PART II ANALYSIS OF HIGH-THROUGHPUT RNA SEQUENCING DATA

8 Quality Control of RNA-Seq Experiments.	137
<i>Xing Li, Asha Nair, Shengqin Wang, and Liguo Wang</i>	
9 Accurate Mapping of RNA-Seq Data.	147
<i>Kin Fai Au</i>	
10 Quantifying Entire Transcriptomes by Aligned RNA-Seq Data	163
<i>Raffaele A. Calogero and Francesca Zolezzi</i>	
11 Transcriptome Assembly and Alternative Splicing Analysis	173
<i>Paola Bonizzoni, Gianluca Della Vedova, Graziano Pesole, Ernesto Picardi, Yuri Pirola, and Raffaella Rizzi</i>	
12 Detection of Post-Transcriptional RNA Editing Events	189
<i>Ernesto Picardi, Anna Maria D'Erchia, Angela Gallo, and Graziano Pesole</i>	
13 Prediction of miRNA Targets	207
<i>Anastasis Oulas, Nestoras Karathanasis, Annita Louloupi, Georgios A. Pavlopoulos, Panayiota Poirazi, Kriton Kalantidis, and Ioannis Iliopoulos</i>	

14	Using Deep Sequencing Data for Identification of Editing Sites in Mature miRNAs	231
	<i>Shahar Alon and Eli Eisenberg</i>	
15	NGS-Trex: An Automatic Analysis Workflow for RNA-Seq Data	243
	<i>Ilenia Boria, Lara Boatti, Igor Saggese, and Flavio Mignone</i>	
16	e-DNA Meta-Barcoding: From NGS Raw Data to Taxonomic Profiling	257
	<i>Fosso Bruno, Marzano Marinella, and Monica Santamaria</i>	
17	Deciphering Metatranscriptomic Data	279
	<i>Evguenia Kopylova, Laurent Noé, Corinne Da Silva, Jean-Frédéric Berthelot, Adriana Alberti, Jean-Marc Aury, and Hélène Touzet</i>	
18	RIP-Seq Data Analysis to Determine RNA–Protein Associations	293
	<i>Federico Zambelli and Giulio Pavesi</i>	
PART III WEB RESOURCES FOR RNA DATA ANALYSIS		
19	The ViennaRNA Web Services	307
	<i>Andreas R. Gruber, Stephan H. Bernhart, and Ronny Lorenz</i>	
20	Exploring the RNA Editing Potential of RNA-Seq Data by ExpEdit	327
	<i>Mattia D'Antonio, Ernesto Picardi, Tiziana Castrignanò, Anna Maria D'Erchia, and Graziano Pesole</i>	
21	A Guideline for the Annotation of UTR Regulatory Elements in the UTRsite Collection	339
	<i>Matteo Giulietti, Giorgio Grillo, Sabino Liuni, and Graziano Pesole</i>	
22	Rfam: Annotating Families of Non-Coding RNA Sequences	349
	<i>Jennifer Daub, Ruth Y. Eberhardt, John G. Tate, and Sarah W. Burge</i>	
23	ASPicDB: A Database Web Tool for Alternative Splicing Analysis	365
	<i>Mattia D'Antonio, Tiziana Castrignanò, Matteo Pallocca, Anna Maria D'Erchia, Ernesto Picardi, and Graziano Pesole</i>	
24	Analysis of Alternative Splicing Events in Custom Gene Datasets by AStalavista	379
	<i>Sylvain Foissac and Michael Sammeth</i>	
25	Computational Design of Artificial RNA Molecules for Gene Regulation.	393
	<i>Alessandro Laganà, Dario Veneziano, Francesco Russo, Alfredo Pulvirenti, Rosalba Giugno, Carlo Maria Croce, and Alfredo Ferro</i>	
	<i>Index</i>	413

Contributors

- ADRIANA ALBERTI • *Genoscope—National Sequencing Center, Evry, France*
- SHAHAR ALON • *Department of Neurobiology, George S. Wise Faculty of Life Sciences, Tel-Aviv University, Tel-Aviv, Israel; Sagol School of Neuroscience, Tel-Aviv University, Tel-Aviv, Israel*
- KIN FAI AU • *Department of Internal Medicine, University of Iowa, Iowa City, IA, USA; Department of Biostatistics, University of Iowa, Iowa City, IA, USA*
- JEAN-MARC AURY • *Genoscope—National Sequencing Center, Evry, France*
- DANNY BARASH • *Department of Computer Science, Ben-Gurion University, Be'er-Sheva, Israel*
- STEPHAN H. BERNHART • *Department of Bioinformatics, University of Leipzig, Leipzig, Germany*
- JEAN-FRÉDÉRIC BERTHELOT • *Inria Lille Nord-Europe, Villeneuve d'Ascq, France*
- LARA BOATTI • *Department of Sciences and Technological Innovation (DiSIT), University of Piemonte Orientale "A. Avogadro", Alessandria, Italy*
- PAOLA BONIZZONI • *Department of Informatics, Systems and Communication (DISCo), University of Milano-Bicocca, Milano, Italy*
- ILENIA BORIA • *Department of Chemistry, University of Milano, Milano, Italy*
- FOSCO BRUNO • *Department of Biosciences, Biotechnology and Biopharmaceutics, University of Bari, Bari, Italy*
- SARAH W. BURGE • *Wellcome Trust Sanger Institute, Cambridgeshire, UK; European Molecular Biology Laboratory, European Bioinformatics Institute, Cambridgeshire, UK*
- RAFFAELE A. CALOGERO • *Department of Molecular Biotechnology and Health Sciences, University of Torino, Torino, Italy*
- TIZIANA CASTRIGNANÒ • *Consorzio Interuniversitario CINECA, Rome, Italy*
- ALEXANDER CHURKIN • *Department of Computer Science, Ben-Gurion University, Be'er-Sheva, Israel*
- CARLO MARIA CROCE • *Department of Molecular Virology, Immunology and Medical Genetics, Comprehensive Cancer Center, The Ohio State University, Columbus, OH, USA*
- MATTIA D'ANTONIO • *Consorzio Interuniversitario CINECA, Rome, Italy*
- ANNA MARIA D'ERCHIA • *Department of Biosciences, Biotechnology and Biopharmaceutics, University of Bari, Bari, Italy; Institute of Biomembranes and Bioenergetics, National Research Council, Bari, Italy*
- JENNIFER DAUB • *Wellcome Trust Sanger Institute, Cambridgeshire, UK; European Molecular Biology Laboratory, European Bioinformatics Institute, Cambridgeshire, UK*
- RUTH Y. EBERHARDT • *Wellcome Trust Sanger Institute, Cambridgeshire, UK; European Molecular Biology Laboratory, European Bioinformatics Institute, Cambridgeshire, UK*
- ELI EISENBERG • *Raymond and Beverly Sackler School of Physics and Astronomy, Tel-Aviv University, Tel-Aviv, Israel; Sagol School of Neuroscience, Tel-Aviv University, Tel-Aviv, Israel*
- FABRIZIO FERRÈ • *Center for Molecular Bioinformatics (CBM), Department of Biology, University of Rome Tor Vergata, Rome, Italy*

- ALFREDO FERRO • *Department of Clinical and Molecular Biomedicine, University of Catania, Catania, Italy*
- SYLVAIN FOISSAC • *UMR1388 GenPhySE, French National Institute for Agricultural Research (INRA), Castanet Tolosan, France*
- ANGELA GALLO • *RNA Editing Laboratory, Oncohaematology Department, IRCCS Ospedale Pediatrico “Bambino Gesù”, Rome, Italy*
- ROSALBA GIUGNO • *Department of Clinical and Molecular Biomedicine, University of Catania, Catania, Italy*
- MATTEO GIULIETTI • *Institute of Biomembranes and Bioenergetics, National Research Council, Bari, Italy*
- GIORGIO GRILLO • *Institute of Biomedical Technologies, National Research Council, Bari, Italy*
- ANDREAS R. GRUBER • *Biozentrum, University of Basel, Basel, Switzerland; Swiss Institute of Bioinformatics, Basel, Switzerland*
- MICHIAKI HAMADA • *Faculty of Science and Engineering, Waseda University, Tokyo, Japan; Computational Biology Research Center, National Institute of Advanced Industrial Science and Technology (AIST), Tokyo, Japan*
- MANUELA HELMER-CITTERICH • *Center for Molecular Bioinformatics (CBM), Department of Biology, University of Rome Tor Vergata, Rome, Italy*
- IOANNIS ILIOPoulos • *Division of Basic Sciences, University of Crete Medical School, Heraklion, Greece*
- KRITON KALANTIDIS • *Institute of Molecular Biology and Biotechnology, FORTH, Heraklion, Greece; Department of Biology, University of Crete, Heraklion, Greece*
- NESTORAS KARATHANASIS • *Institute of Molecular Biology and Biotechnology, FORTH, Heraklion, Greece; Department of Biology, University of Crete, Heraklion, Greece*
- EVGUENIA KOPYLOVA • *Laboratoire d’Informatique Fondamentale de Lille (LIFL), UMR CNRS 8022, Villeneuve d’Ascq Cédex, France; Inria Lille Nord-Europe, Villeneuve d’Ascq Cédex, France*
- ALESSANDRO LAGANÀ • *Department of Molecular Virology, Immunology and Medical Genetics, Comprehensive Cancer Center, The Ohio State University, Columbus, OH, USA*
- FABRICE LECLERC • *Institut de Génétique et Microbiologie (IGM) UMR 8621, Université Paris Sud, Orsay, France*
- XING LI • *Division of Biomedical Statistics and Informatics, Department of Health Sciences Research, Mayo Clinic, Minneapolis, MN, USA*
- SABINO LIUNI • *Institute of Biomedical Technologies, National Research Council, Bari, Italy*
- RONNY LORENZ • *Institute for Theoretical Chemistry, University of Vienna, Vienna, Austria*
- ANNITA LOULOUPI • *Biomedical Science-Medical Biology, University of Amsterdam, Amsterdam, The Netherlands*
- MARZANO MARINELLA • *Institute of Biomembranes and Bioenergetics, CNR, Bari, Italy*
- EUGENIO MATTEI • *Center for Molecular Bioinformatics (CBM), Department of Biology, University of Rome Tor Vergata, Rome, Italy*
- FLAVIO MIGNONE • *Department of Sciences and Technological Innovation (DiSIT), University of Piemonte Orientale “A. Avogadro”, Alessandria, Italy*
- SOHEILA MONTASERI • *Department of Mathematics, Statistics and Computer Science, University of Tehran, Tehran, Iran*
- ASHA NAIR • *Division of Biomedical Statistics and Informatics, Department of Health Sciences Research, Mayo Clinic, Minneapolis, MN, USA*

- LAURENT NOÉ • *Laboratoire d’Informatique Fondamentale de Lille (LIFL), UMR CNRS 8022, Villeneuve d’Ascq Cédex, France; Inria Lille Nord-Europe, Villeneuve d’Ascq Cédex, France*
- ANASTASIS OULAS • *Institute of Marine Biology, Biotechnology and Aquaculture, HCMR, Heraklion, Greece*
- MATTEO PALLOCCA • *Translational Oncogenomics Unit, Italian National Cancer Institute “Regina Elena”, Rome, Italy*
- GIULIO PAVESI • *Dipartimento di Bioscienze, Università di Milano, Milano, Italy*
- GEORGIOS A. PAVLOPOULOS • *Division of Basic Sciences, University of Crete Medical School, Heraklion, Greece*
- GRAZIANO PESOLE • *Department of Biosciences, Biotechnology and Biopharmaceutics, University of Bari, Bari, Italy; Institute of Biomembranes and Bioenergetics, National Research Council, Bari, Italy; National Institute of Biostructures and Biosystems (INBB), Rome, Italy*
- ERNESTO PICARDI • *Department of Biosciences, Biotechnology and Biopharmaceutics, University of Bari, Bari, Italy; Institute of Biomembranes and Bioenergetics, National Research Council, Bari, Italy; National Institute of Biostructures and Biosystems (INBB), Rome, Italy*
- YURI PIROLA • *Department of Informatics, Systems and Communication (DISCo), University of Milano-Bicocca, Milano, Italy*
- PANAYIOTA POIRAZI • *Institute of Molecular Biology and Biotechnology, FORTH, Heraklion, Greece*
- YANN PONTY • *Laboratoire d’Informatique de ‘X (LIX) UMR 7161, CNRS and INRIA AMIB, Ecole Polytechnique, Palaiseau, France*
- ALFREDO PULVIRENTI • *Department of Clinical and Molecular Biomedicine, University of Catania, Catania, Italy*
- VLADIMIR REINHARZ • *School of Computer Science, McGill University, Montreal, QC, Canada*
- RAFFAELLA RIZZI • *Department of Informatics, Systems and Communication (DISCo), University of Milano-Bicocca, Milano, Italy*
- FRANCESCO RUSSO • *Department of Clinical and Molecular Biomedicine, University of Catania, Catania, Italy; Laboratory for Integrative System Medicine (LISM), Institute of Informatics and Telematics (IIT) and Institute of Clinical Physiology (IFC), National Research Council (CNR), Pisa, Italy*
- IGOR SAGGESE • *Department of Sciences and Technological Innovation (DiSIT), University of Piemonte Orientale “A. Avogadro”, Alessandria, Italy*
- MICHAEL SAMMETH • *Bioinformatics, National Laboratory of Cientific Computing (LNCC), Rio de Janeiro, Brazil*
- MONICA SANTAMARIA • *Institute of Biomembranes and Bioenergetics, CNR, Bari, Italy*
- CORINNE DA SILVA • *Genoscope—National Sequencing Center, Evry, France*
- JOHN G. TATE • *Wellcome Trust Sanger Institute, Cambridgeshire, UK; European Molecular Biology Laboratory, European Bioinformatics Institute, Cambridgeshire, UK*
- HÉLÈNE TOUZET • *Laboratoire d’Informatique Fondamentale de Lille (LIFL), UMR CNRS 8022, Villeneuve d’Ascq Cédex, France; Inria Lille Nord-Europe, Villeneuve d’Ascq Cédex, France*
- GIANLUCA DELLA VEDOVA • *Department of Informatics, Systems and Communication (DISCo), University of Milano-Bicocca, Milano, Italy*

DARIO VENEZIANO • *Department of Molecular Virology, Immunology and Medical Genetics, Comprehensive Cancer Center, The Ohio State University, Columbus, OH, USA;*

Department of Clinical and Molecular Biomedicine, University of Catania, Catania, Italy

JÉRÔME WALDISPÜHL • *School of Computer Science, McGill University, Montreal, QC, Canada*

SHENGQIN WANG • *School of Biological Science and Medical Engineering,*

Southeast University, Nanjing, Jiangsu, PR, China

LIGUO WANG • *Division of Biomedical Statistics and Informatics, Department of Health Sciences Research, Mayo Clinic, Minneapolis, MN, USA*

LINA WEINBRAND • *Department of Computer Science, Ben-Gurion University, Beer-Sheva, Israel*

FEDERICO ZAMBELLI • *Istituto di Biomembrane e Bioenergetica, Consiglio Nazionale delle Ricerche - CNR, Bari, Italy*

FRANCESCA ZOLEZZI • *Singapore Immunology Network (SIgN), Agency of Science, Technology and Research (A*STAR), Biopolis, Singapore, Singapore*

Part I

RNA Secondary and Tertiary Structures

Chapter 1

Free Energy Minimization to Predict RNA Secondary Structures and Computational RNA Design

Alexander Churkin, Lina Weinbrand, and Danny Barash

Abstract

Determining the RNA secondary structure from sequence data by computational predictions is a long-standing problem. Its solution has been approached in two distinctive ways. If a multiple sequence alignment of a collection of homologous sequences is available, the comparative method uses phylogeny to determine conserved base pairs that are more likely to form as a result of billions of years of evolution than by chance. In the case of single sequences, recursive algorithms that compute free energy structures by using empirically derived energy parameters have been developed. This latter approach of RNA folding prediction by energy minimization is widely used to predict RNA secondary structure from sequence. For a significant number of RNA molecules, the secondary structure of the RNA molecule is indicative of its function and its computational prediction by minimizing its free energy is important for its functional analysis. A general method for free energy minimization to predict RNA secondary structures is dynamic programming, although other optimization methods have been developed as well along with empirically derived energy parameters. In this chapter, we introduce and illustrate by examples the approach of free energy minimization to predict RNA secondary structures.

Key words RNA folding prediction, Free energy minimization methods, RNA free energy parameters, RNA bioinformatics, RNA secondary structure prediction

1 Introduction

The RNA molecule, once considered as an intermediate step between DNA and protein, has become a central subject of research in recent years. Although the functional role of RNAs is often related to their 3-D structure, the RNA secondary structure is experimentally accessible and contains much information to shed light on the relationship between structure and function. In general, RNA folding is thought to be hierarchical in nature [1, 2] where a stable secondary structure forms first and subsequently there is a refinement to the tertiary fold. Thus, predicting RNA secondary structures from sequence is an important problem in computational biology. The secondary structure of an RNA molecule is a representation of the pattern, given an initial RNA

sequence, of complementary base-pairings that are formed between the nucleic acids. The sequence, represented as a string of four letters, is a single strand consisting of the nucleotides A, C, G, and U, which are generally assumed to pair to form a secondary structure with minimum free energy. Therefore, aside from the comparative method in case a multiple sequence alignment of homologous sequences is available, if a practitioner would like to predict secondary structure from an RNA sequence that has no homologs or that belongs to a poorly annotated family in a comprehensive RNA database such as Rfam [3], then free energy minimization should be the method of choice.

The folding prediction problem of the secondary structure from a single RNA sequence has been an area of active research since the late 1970s. Dynamic programming methods for this problem were initially developed in [4–6]. Energy minimization methods by dynamic programming using auxiliary information [7, 8] have led to Zuker’s mfold prediction server [9] and the Vienna RNA package [10, 11]. The predictive accuracy of these packages was improved by incorporating expanded energy rules [12], derived from an independent set of experiments, into the folding prediction algorithm. It should be noted that other methods aside from dynamic programming have been employed in the context of free energy minimization, mainly stochastic ones such as in MPGAfold [13], but the ones that are mostly used are variants of dynamic programming. For example, RNAfold in the Vienna RNA package is slightly different from mfold in using the partition function algorithm of McCaskill [14], but both are based on dynamic programming and their results are in most cases very similar. An important aspect of energy minimization software from the practical standpoint is the ability to predict suboptimal solutions in addition to the optimal one [15, 16]. The “mfold style” suboptimal solutions are computed along with a filtering step that ensures that suboptimal solutions differ to avoid redundancy [15]. A different way of calculating the suboptimal solutions was carried out in [16] for the Vienna RNA package. It calculates all suboptimal solutions within an energy range above the minimum free energy without a pre-prescribed filtering step. It is then possible to apply a post-prescribed filtering step, such as with RNASHapes [17]. We will illustrate in the examples the merits of both approaches, namely the “mfold style” and the “Vienna style” ways of calculating the suboptimal solutions. Finally, there are many advanced applications that utilize free energy minimization, such as in a genome-scale structure-based clustering of RNAs called LocARNA [18] and de novo prediction of structured RNAs from genomic sequences [19], to name but a few. Basic tasks that accompany structure prediction by energy minimization are available in the software packages for the new mfold package called UNAFold [20] and the most recent version of the Vienna RNA package [21], both being actively used and regularly maintained. Herein, we provide a

few examples that illustrate the use of free energy minimization to predict RNA structures and utilize the programs included in the software prediction packages.

2 Materials

2.1 Biological Sequences

We obtained the RNA sequences for our examples from the study published by You et al. [22] and Krol et al. [23]. This is by no means inclusive as there are many different ways to obtain biological sequences, e.g., from a sequencing experiment to investigate a certain organism or from a known database such as Rfam [3], to name but a few. Let us briefly describe these biological sequences and their significance. The first sequence used for illustration, the 5BSL3.2 HCV taken from [22], is a functionally important stem-loop structure for virus replication that appears in the coding region of the hepatitis C virus RNA-dependent RNA polymerase, NS5B. The 5BSL3.2 is about 50 bases in length and is part of a larger predicted cruciform structure (5BSL3). It was confirmed experimentally that the 5BSL3.2 consists of an 8-bp lower helix, a 6-bp upper helix, a 12-base terminal loop, and an 8-bp internal loop. Mutagenesis experiments were performed to investigate the relationship between its structure and function. In addition, the size of the sequence is amenable to energy minimization prediction methods (*see Note 1*). Thus, the sequence of the 5BSL3.2 HCV provides a good test bed for energy minimization prediction methods since the structure of both the wildtype sequence and several of its mutants was already investigated by structure probing experiments. The second sequence used for illustration, an miRNA precursor taken from [23], is a hairpin loop structure of about 70 bases in length. It is important as a participant in gene regulation and its structure was verified experimentally, thus providing another good test bed for energy minimization prediction methods.

For convenience, the sequence of the 5BSL3.2 HCV is shown below:

AGCGGGGGAGACAUUAUCACAGCCUGUCGUGGCC
GACCCCGC

Its two-point mutants that were experimentally tested in [22] are clearly specified in the text herein and the relevant figure captions.

2.2 Computational Hardware

The most widely used energy minimization software for predicting RNA secondary structures from sequence can be accessed using web servers, e.g., [9, 11], requiring no special computational resources. With the exception of some specific large-scale tasks that may require whole genome scans, or specific cases of a software that may require a massively parallel computational platform [13], almost all tasks that utilize energy minimization prediction are approached with a standard PC. If one prefers to download a

software package instead of using a webserver directly or for more involved tasks, the most straightforward hardware platforms to use are Linux and Unix-based computers, although Windows and Mac OS X and possibly other operating systems are in general supported as well.

2.3 Computational Software

When downloading the most widely used energy minimization software packages such as UNAFold/mfold or the Vienna RNA package, some common requirements are to have a Perl interface such as Perl 5.6.1 or greater and an application programming interface (API) such as OpenMP and OpenGL. Specifications can be found on the instructions page that is available for each individual package.

3 Methods

Free energy minimization to predict RNA secondary structures is an advanced procedure from the algorithmic side, which has been gradually developed and improved over the years, but conceptually simple from the practical side. The user submits a sequence as input and can either choose to accompany the sequence with default parameters or to change the value of the parameters according to the problem at hand. The most basic free energy minimization problem is to predict structure from sequence, but there are several other related problems that use the basic problem repeatedly and are therefore relying on free energy minimization. Thus, we chose to illustrate free energy minimization by representative examples. First, the basic problem of predicting structure from sequence will be exemplified. Second, a related problem of predicting which sequence mutations will cause a dramatic change in structure will be shown. Finally, the third example will illustrate the problem of sequence design, which is the inverse problem (sequence from structure) relative to the direct problem (structure from sequence). It uses the direct problem iteratively, after starting from an initial guess for the sequence and mutating the sequence in each step in order to optimize a certain objective function.

The basic structure for the user to follow is common in all cases:

1. Access the relevant webserver and/or download the program.
2. Insert an input sequence and change parameter values (default values are given).
3. Run the program and observe the output predicted structure, mostly given in graphical forms (e.g., RNA secondary structure drawings, dot plots) along with textual information.

3.1 Predicting RNA Secondary Structures

We start by describing the basic procedure of predicting RNA secondary structures by energy minimization given an initial sequence. The format for the input sequence is similar in almost all software packages that have been developed to solve this problem. The user can either select to manually insert a sequence consisting of the

letters A,C,G,U or use the FASTA format. For example, in the mfold webserver [9], the user starts by filling in the RNA Folding Form, inserting the 5BSL3.2 HCV sequence that was given towards the end of Subheading 2, including its name. Default parameters can remain with the values given in the various fields unless there is a clear justification to insert different values. Upon pressing “Fold RNA” found at the end of the RNA Folding Form, mfold performs the free energy minimization prediction and outputs several ways to display the results. Selecting “Structure 1” with displaying it in one of the different formats available, such as pdf, will result in Fig. 1. Going back to the RNA Folding Form and increasing the percent of suboptimality from the default of 5–30, repeating the same procedure above, will result in four different structures instead of one. Opting for several suboptimal solutions instead of a single optimal solution can be important (see Note 2). The default parameter is usually enough when using the “mfold style” suboptimal solutions [15], but in our case we increase the percent of suboptimality because we are interested in checking if one of the weak suboptimals has the potential to become dominant when introducing a change to the sequence. All structures can now be viewed using an energy dot plot by selecting all four structures in the “Compare selected foldings” field and pressing “Do the

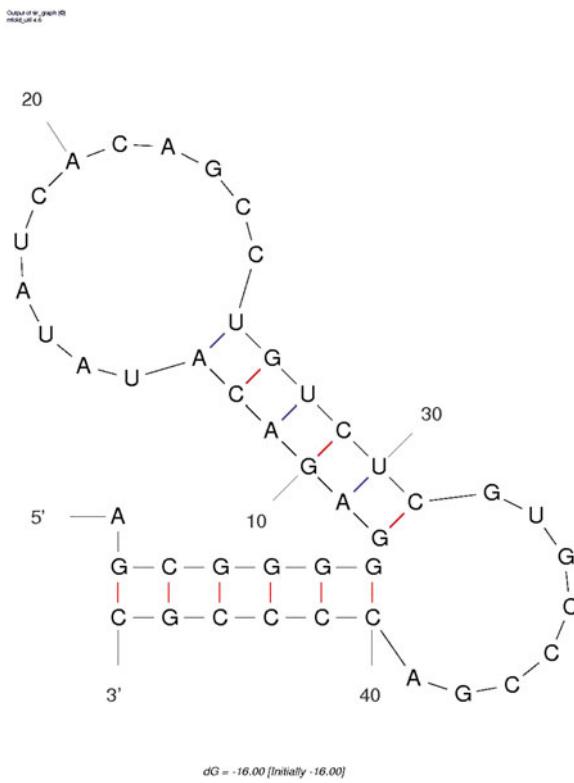


Fig. 1 Free energy minimization for the secondary structure prediction of wild-type 5BSL3.2 HCV. UNAFold version 3.8 with mfold utils version 4.6 was used to predict and generate the secondary structure drawing

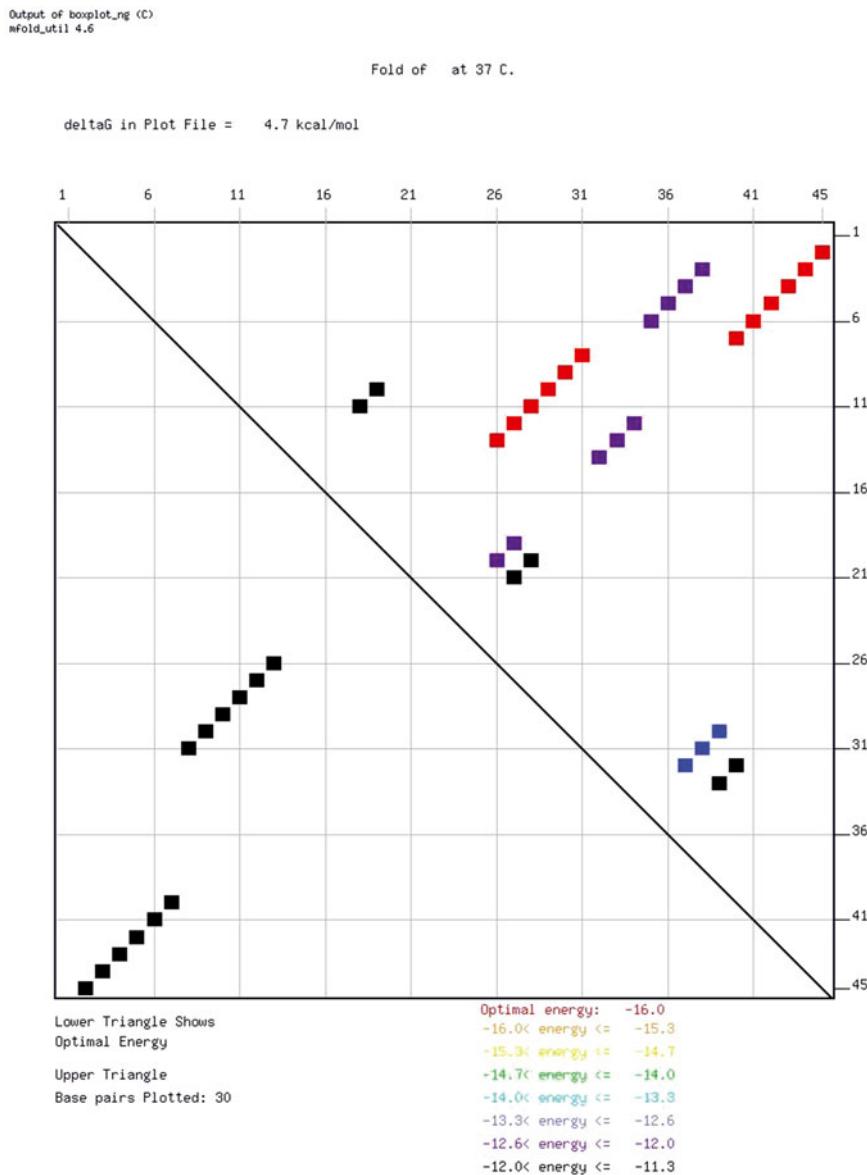
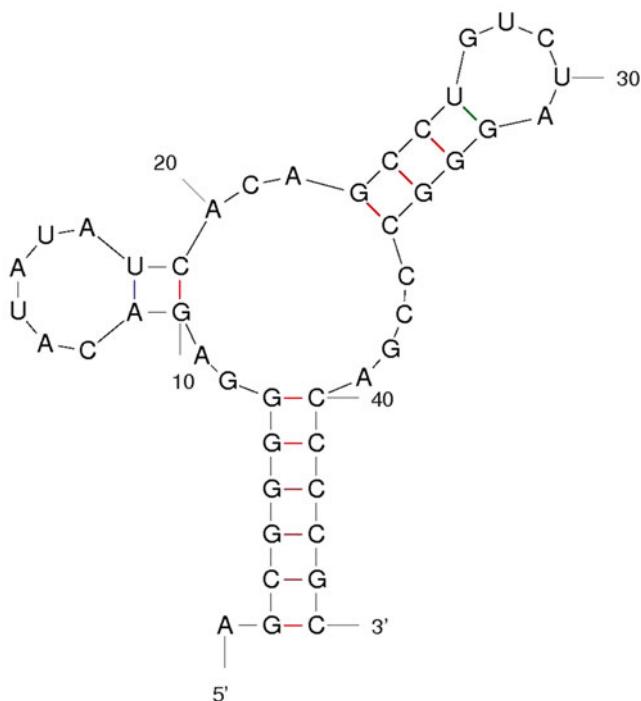


Fig. 2 Energy dot plot for the secondary structure prediction of wildtype 5BSL3.2 HCV. Using mfold util version 4.6, the percent of suboptimality was increased to 30 to capture the fourth suboptimal solution, for an illustrative reason that will become clear after generating and examining Fig. 3

comparison". This will result in Fig. 2, where in the upper triangular of the dot plot all four structures can be viewed while the lower triangular contains only the optimal structure. Going back to the RNA Folding Form and slightly modifying the sequence by inserting a two-point mutation C31A-U33G, repeating the same procedure as above with default parameters (percent of suboptimality is now back to 5) will result in Fig. 3. Notice that the stable optimal structure for the mutant resembles the fourth suboptimal structure for the wildtype predicted structure. Such an observation makes sense because often times the mutant structure of a point mutation



$dG = -13.45$ [Initially -17.10]

Fig. 3 Free energy minimization for the secondary structure prediction of a specific two-point mutant 5BSL3.2 HCV (C31A-U33G). UNAFold version 3.8 with mfold utils version 4.6 was used to predict and generate the secondary structure drawing. One can observe the similarity in shape between the structure obtained and the fourth suboptimal solution displayed in the dot plot of Fig. 2

that is in the near-neighborhood of the wildtype will originate from a suboptimal solution of the wildtype that becomes the optimal solution upon introducing the mutation.

3.2 Predicting RNA Deleterious Mutations

The first example above served to illustrate the most basic use of free energy minimization to predict RNA structures. Having also observed in this example the relationship between suboptimal solutions of the wildtype and their possible appearance as an optimal solution of a near-neighbor mutant, we now focus on a second example that utilizes this concept to efficiently predict deleterious mutations in the sequence that may cause a dramatic change in structure. As detailed in a review on this topic [24], there are several ways to approach this problem [25, 26]. The method described in [25] relies on Vienna's RNAfold [10] for the direct problem of RNA structure prediction, Vienna's RNAsubopt [16] for calculating all suboptimal solutions, and the concept described above of a suboptimal solution becoming an optimal one. To experiment, the user can now access the RNAmute webserver [27] and insert the same example sequence of the 5BSL3.2 HCV that was used as input

in the first two figures. Changing the number of mutations to 2 in the appropriate field and pressing “Submit Form” will result in a table of mutations that are ranked according to the distance of their free energy minimization predicted structure relative to the wildtype predicted structure. One of the few two-point mutations appearing distinctively with the largest distance is the one shown in Fig. 4, where the labeled changes appearing in red, corresponding to the example depicted in Fig. 3. This well illustrates the fact that components from free energy minimization to predict RNA secondary structures, such as RNAfold and RNAsubopt from the Vienna RNA package, may be used as the backbone of other related problems such as RNA deleterious mutation prediction (*see Note 3*).

3.3 Computational RNA Design

The final example is taken from the field of RNA design. The inverse RNA folding problem for designing sequences that fold into a given RNA secondary structure was introduced in [10]. The approach to solve it by stochastic optimization relies on the solution of the direct problem. Initially, a seed sequence is chosen, after which a local search strategy (or global, as in [28]) is used to mutate the seed and apply repeatedly the direct problem of RNA

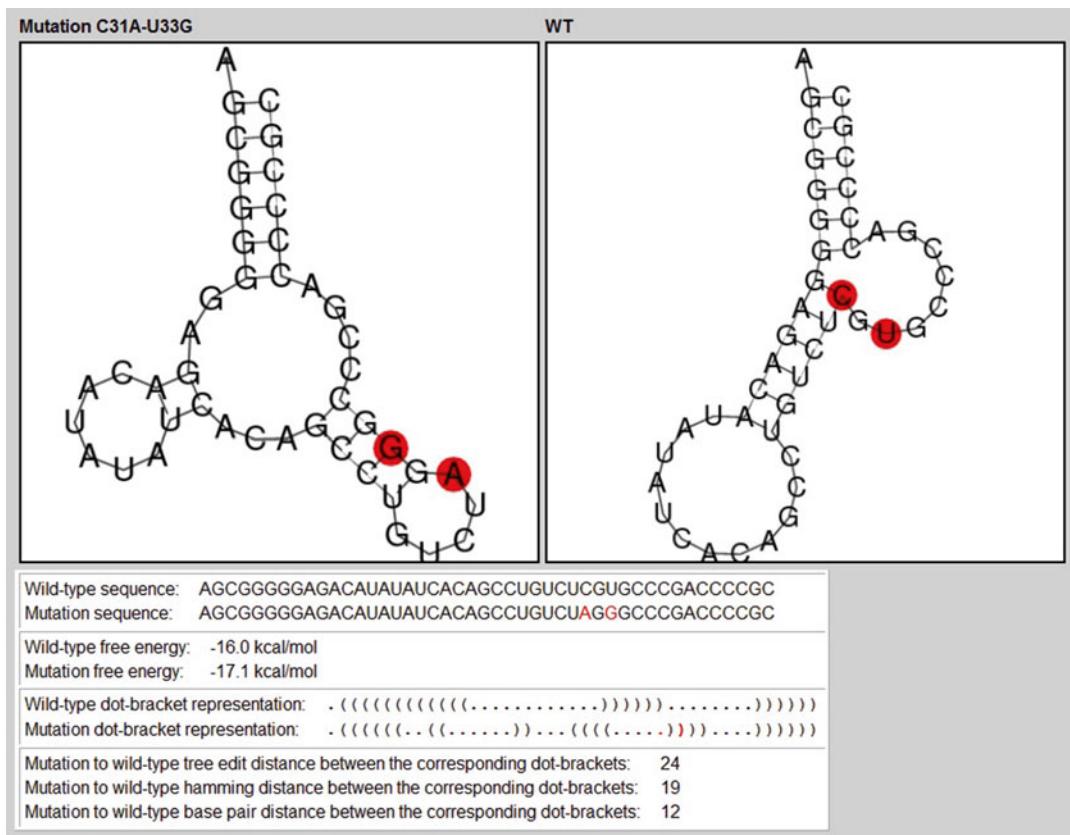


Fig. 4 Predicted rearranging mutation in the 5BSL3.2 wildtype. The RNAmute webserver was used, relying on RNAfold, RNAsubopt, and other programs from the Vienna RNA package. The output two-point mutant C31A-U33G happens to coincide with the experimental result that was illustrated in Fig. 3

folding prediction by energy minimization. Recently, an extension to the problem was developed that allows designing sequences that fold into a prescribed shape, leaving some flexibility in the secondary structure of RNA motifs that do not necessarily possess a known functional role. The shape of the RNA can be represented as a tree-graph [29] and its relation to RNA secondary structures can then be clarified by examining Fig. 5. The program that implements

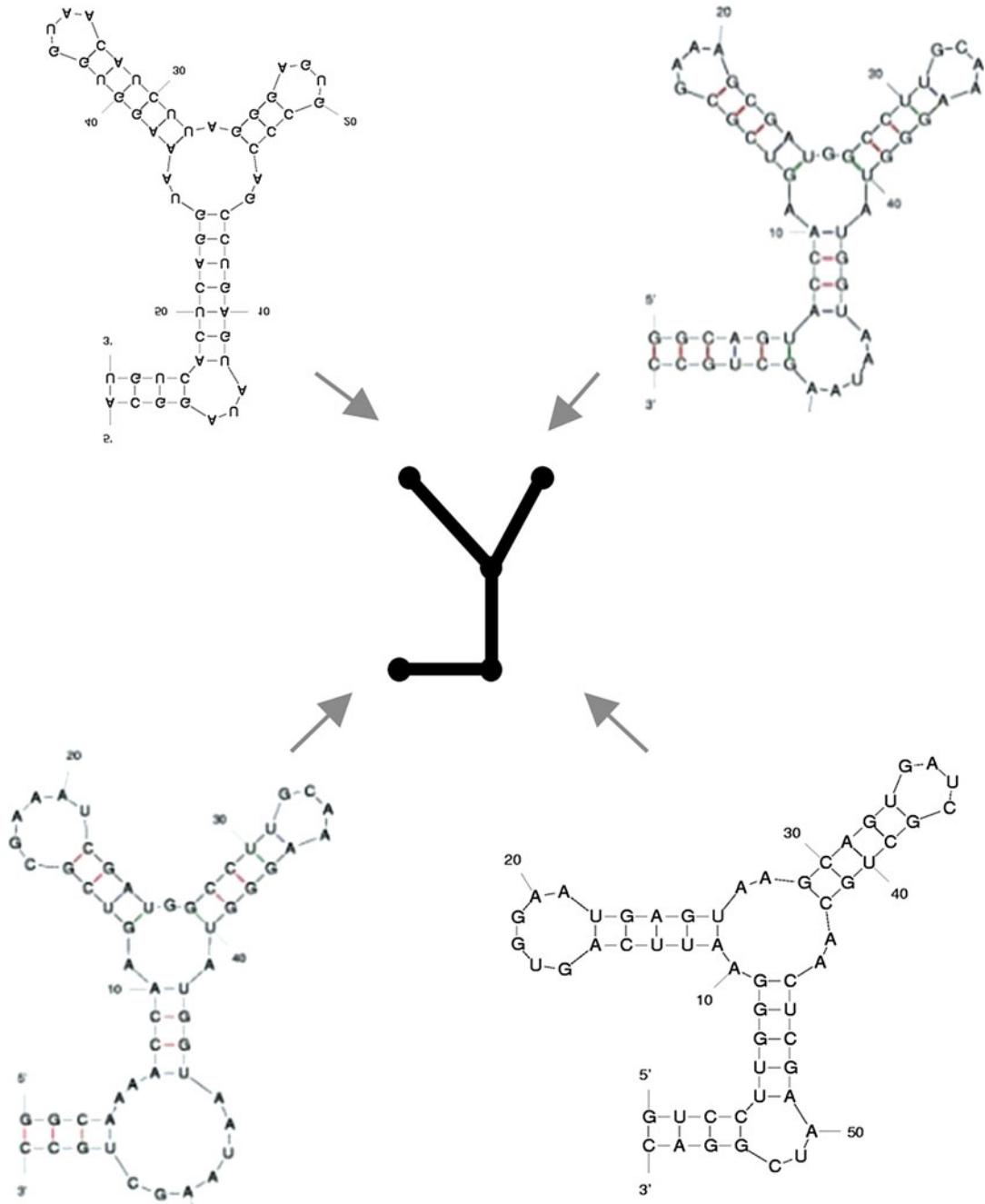


Fig. 5 A tree-graph illustration for a coarse grain representation [29] of RNA secondary structures

this type of sequence design, put forth in [30], relies on programs from the Vienna RNA package such as RNAfold, RNAinverse, and RNAdistance [10]. This time, the user downloads a program called RNAfbinv [31] (RNA fragment-based design), runs the application, and in the input screen inserts a secondary structure as input in dot-bracket representation. Upon pressing “Fragment”, a new window opens where the user can choose a fragment constraint using a combo-box to select a desired motif that should be preserved. Then, in the initial input screen, upon pressing “Process” a subsequent window is displayed where certain physical parameters appear with default numerical values. Upon pressing “Result”, the program computes several designed sequences ranked by their base-pair distance from the input structure. Examples for such a solution and the various stages in the GUI that demonstrate the user experience are available in Figs. 6, 7, and 8. Figure 7 shows an example of the design results of an artificial structure while in Fig. 8, a design result of an miRNA precursor (in compliance with Note 4) taken from [23] is shown.

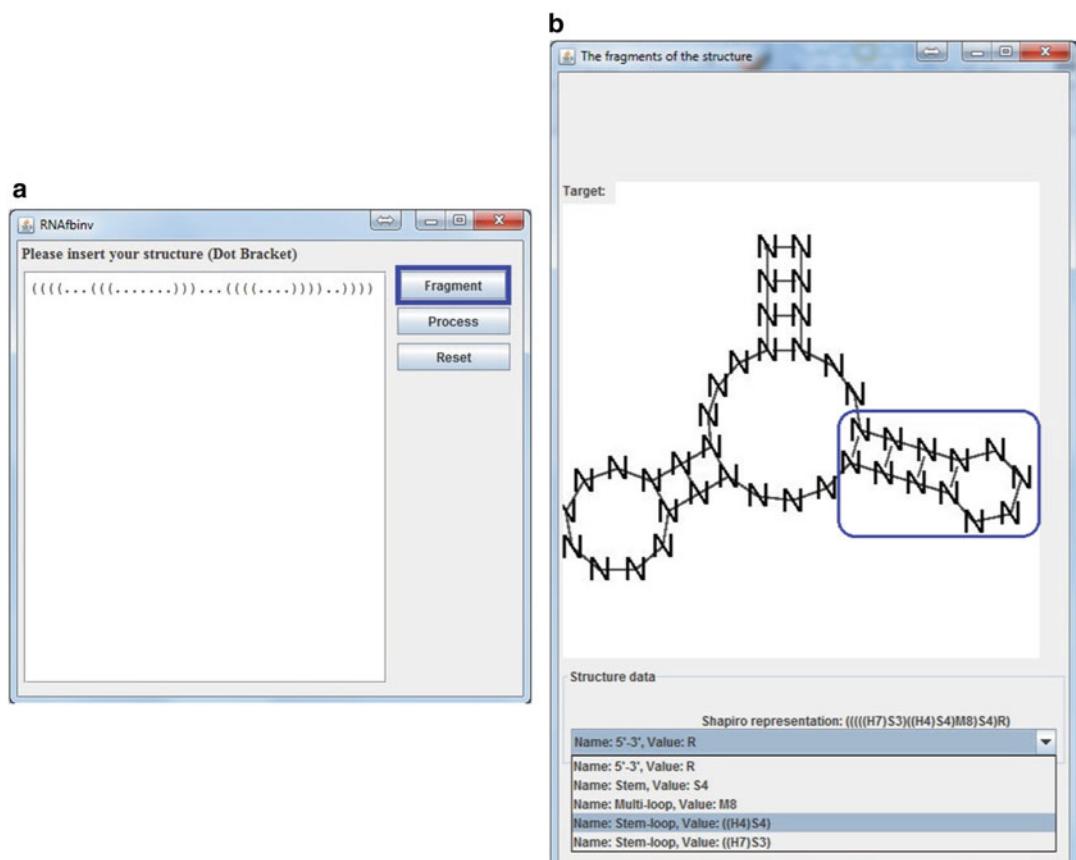


Fig. 6 Sequence design with a motif constraint by using repetitively free-energy minimization of RNA secondary structures. (a) Initial input screen in the RNAfbinv application for designing sequences. (b) Motif selection screen in the RNAfbinv application for designing sequences

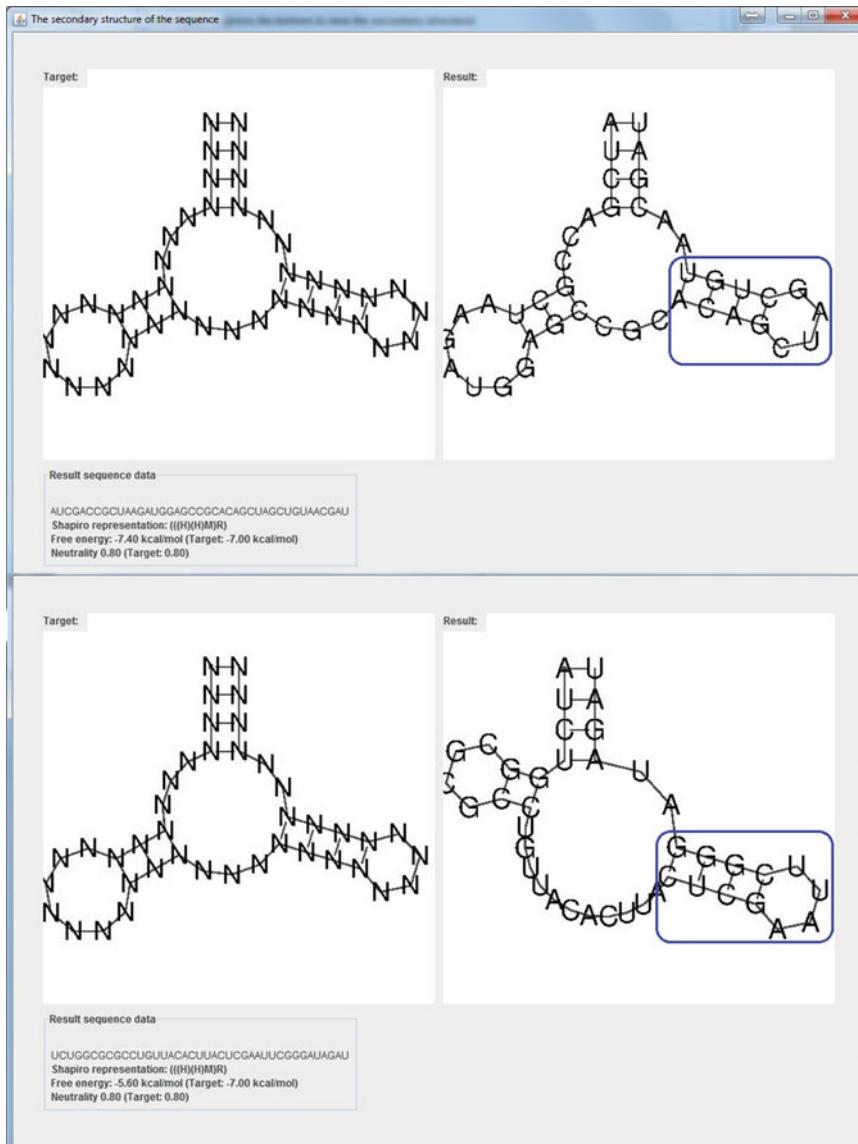


Fig. 7 Sequence design of an artificial example by energy minimization. Best result and 20th result obtained with RNAfbinv according to base-pair distance from the wildtype structure are displayed

In the future, the uses for such free energy minimization applications that involve computational RNA design are expected to grow considerably, with the increase of experimental evidence regarding the functional importance of certain RNA motifs. Also, these applications can be used to detect known noncoding RNAs in new locations when the query RNA pattern is suspected to possess more conservation in structure and less conservation in sequence. Results of the flexible computational RNA design procedure are sequences that can be searched for efficiently by using sequence-based methods.

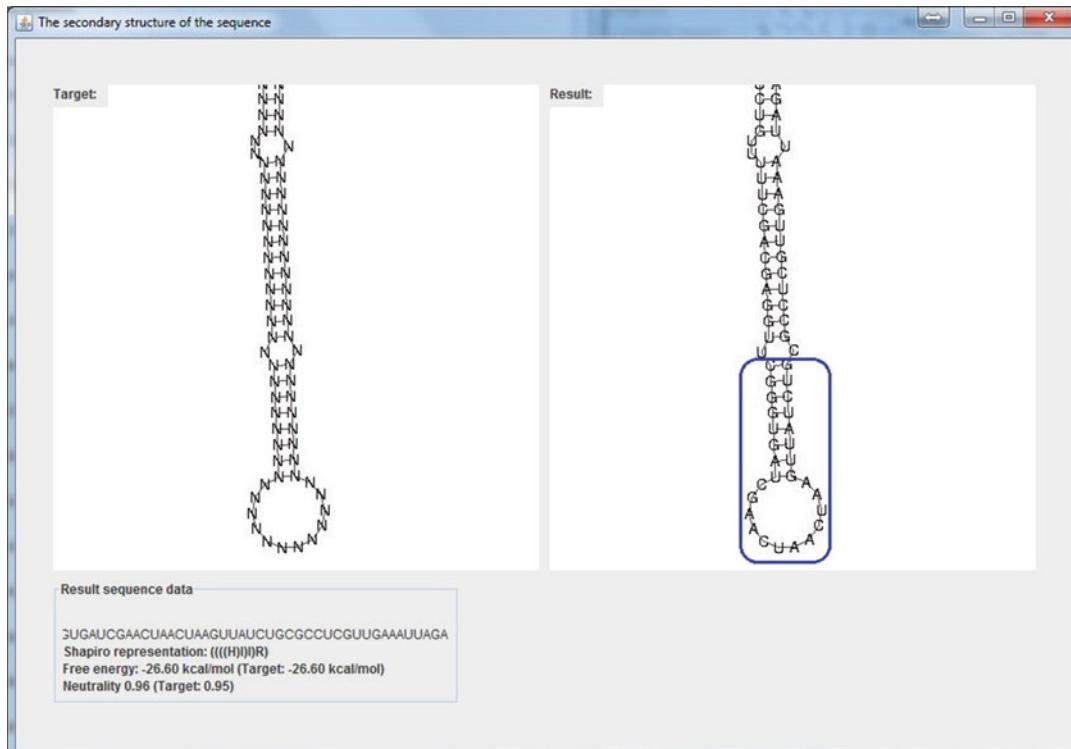


Fig. 8 Sequence design of an miRNA precursor by energy minimization. Best result obtained with RNAfbinv according to the base-pair distance from the wildtype structure is shown with the depicted motif constraint

4 Notes

1. All free energy minimization methods rely on thermodynamic parameters that are empirically derived for small combinations of nucleotides. Therefore, the upper range estimate for the sequence length that these programs are useful for is ~150 nt. Predictions made with longer sequences tend to become inaccurate and possess artifacts. If the user needs to make predictions for longer sequences, it should be done locally by a moving window approach, for example with Vienna's RNALfold [21]. This way, energy minimization predictions are performed inside a window whose size is no longer than 150 nt.
2. An important breakthrough in the field of RNA folding prediction by energy minimization was the ability to calculate meaningful suboptimal solutions. This was first devised in ref. 15, known as the “mfold style” approach that performs an automatic filtering to the suboptimal solutions. It is still being widely used at present and it differs from the “Vienna style” approach that calculates all suboptimal solutions within an

energy range above the minimum free energy without a pre-prescribed filtering step [16]. Whether one should favor one approach over the other depends on the application, and there are also alternative approaches to calculate suboptimal solutions [17]. In any case, it is recommended to take advantage of the ability to calculate suboptimal solutions in free energy minimization to predict secondary structures.

3. As observed in the illustrations for Example 2 and Example 3, one can solve more specific problems such as RNA deleterious mutation prediction and computational RNA design using components taken from the latest versions of the most widely used RNA software prediction packages [20, 21].
4. RNA folding [1, 2] is an involved process that depends on many factors in the environment, such as ion concentration and small molecules that can bind to the RNA and influence the folding in ways that are impossible to predict. These limitations should be taken into account and one needs to be careful from making predictions about the folding of RNAs that are known to reside in such environments. Nevertheless, there are many cases of significant interest where folding prediction by energy minimization is indispensable as an approach to analyze RNA structure and its relation to function.

Acknowledgments

The authors would like to thank Idan Gabdank and Assaf Avihoo for their assistance in this study. This work was partially supported by the Kreitman Foundation at Ben-Gurion University.

References

1. Brion P, Westhof E (1997) Hierarchy and dynamics of RNA folding. *Annu Rev Biophys Biomol Struct* 26:113–137
2. Tinoco I, Bustamante C (1999) How RNA folds. *J Mol Biol* 293:271–281
3. Griffiths-Jones S, Bateman A, Marshall M, Khanna A, Eddy SR (2003) Rfam: an RNA family database. *Nucleic Acids Res* 31: 439–441
4. Nussinov R, Pieczenik G, Grigg JR, Kleitman DJ (1978) Algorithms for loop matchings. *SIAM J Appl Math* 35:68–82
5. Waterman MS, Smith TF (1978) RNA secondary structure: a complete mathematical analysis. *Math Biosci* 42:257–266
6. Nussinov R, Jacobson AB (1980) Fast algorithm for predicting the secondary structure of single stranded RNA. *Proc Natl Acad Sci U S A* 77(11):6309–6313
7. Zuker M, Stiegler P (1981) Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. *Nucleic Acids Res* 9:133–148
8. Zuker M, Sankoff D (1984) RNA secondary structures and their prediction. *Bull Math Biol* 46:591–621
9. Zuker M (2003) Mfold web server for nucleic acid folding and hybridization prediction. *Nucleic Acids Res* 31:3406–3415
10. Hofacker IL, Fontana W, Stadler PF, Bonhoeffer LS, Tacker M, Schuster P (1994) Fast folding and comparison of RNA secondary structures. *Monatsh Chem* 125:167–188
11. Hofacker IL (2003) Vienna RNA secondary structure server. *Nucleic Acids Res* 31: 3429–3431
12. Mathews DH, Sabina J, Zuker M, Turner DH (1999) Expanded sequence dependence of

- thermodynamic parameters improves prediction of RNA secondary structure. *J Mol Biol* 288:911–940
- 13. Shapiro BA, Wu J-C, Bengali D, Potts MJ (2001) The massively parallel genetic algorithm for RNA folding: MIMD implementation and population variation. *Bioinformatics* 17:137–148
 - 14. McCaskill JS (1990) The equilibrium partition function and base pair binding probabilities for RNA secondary structure. *Biopolymers* 29: 1105–1119
 - 15. Zuker M (1989) On finding all suboptimal foldings of an RNA molecule. *Science* 244: 48–52
 - 16. Wuchty S, Fontana W, Hofacker IL, Schuster P (1999) Complete suboptimal folding of RNA and the stability of secondary structures. *Biopolymers* 49:145–165
 - 17. Steffen B, Voss B, Rehmsmeier M, Reeder J, Giegerich R (2006) RNAshapes: an integrated RNA analysis package based on abstract shapes. *Bioinformatics* 22(4):500–503
 - 18. Will S, Reiche K, Hofacker IL, Stadler PF, Backofen R (2007) Inferring non-coding RNA families and classes by means of genome-scale structure-based clustering. *PLoS Comput Biol* 3(4):e65
 - 19. Gorodkin J, Hofacker IL, Torarinsson E, Yao Z, Havgaard JH, Ruzzo WL (2010) De novo prediction of structures RNAs from genomic sequences. *Trends Biotechnol* 28(1):9–20
 - 20. Markham NR, Zuker M (2008) Software for nucleic acid folding and hybridization. *Methods Mol Biol* 453:3–31
 - 21. Lorenz R, Lorenz R, Bernhart SH, Höner zu Siederdissen C, Siederdissen C, Tafer H, Flamm C, Stadler PF, Hofacker IL (2011) ViennaRNA package 2.0. algorithms. *Mol Biol* 6:26
 - 22. You S, Stump DD, Branch AD, Rice CM (2004) A cis-acting replication element in the sequence encoding the NS5B RNA-dependent RNA polymerase is required for hepatitis c virus RNA replication. *J Virol* 78(3): 1352–1366
 - 23. Krol J, Sobczak K, Wilczynska U, Drath M, Janiska A, Kaczynska D, Krzyzosiak WJ (2004) Structural features of microRNA (miRNA) precursors and their relevance to miRNA biogenesis and small interfering RNA/short hairpin RNA design. *J Biol Chem* 279: 42230–42239
 - 24. Barash D, Churkin A (2011) Mutational analysis in RNAs: comparing programs for RN deleterious mutation prediction. *Brief Bioinform* 12(2):104–114
 - 25. Churkin A, Barash D (2008) An efficient method for the prediction of deleterious multiple-point mutations in the secondary structure of RNAs using suboptimal folding solutions. *BMC Bioinformatics* 9:222
 - 26. Waldspühel J, Devadas S, Berger B, Clote P (2008) Efficient algorithms for probing the RNA mutational landscape. *PLoS Comput Biol* 4:e1000124
 - 27. Churkin A, Gabdank I, Barash D (2011) The RNAmute web server for the mutational analysis of RNA secondary structures. *Nucleic Acids Res* 39:W92–W99
 - 28. Levin A, Lis M, Ponty Y, O'Donnell CW, Devadas S, Berger B, Waldspühel J (2012) A global sampling approach to designing and reengineering RNA secondary structures. *Nucleic Acids Res* 40(20):10041–10052
 - 29. Shapiro BA (1988) An algorithm for comparing RNA secondary structures. *Comput Appl Biosci* 4:387–393
 - 30. Avihoo A, Churkin A, Barash D (2011) RNAExinv: an extended inverse RNA folding from shape and physical attributes to sequences. *BMC Bioinformatics* 12(319):24
 - 31. Weinbrand L, Avihoo A, Barash D (2013) RNAfbinv: an interactive Java application for fragment-based design of RNA sequences. *Bioinformatics* 29(22):2938–2940

Chapter 2

RNA Secondary Structure Prediction from Multi-Aligned Sequences

Michiaki Hamada

Abstract

It has been well accepted that the RNA secondary structures of most functional non-coding RNAs (ncRNAs) are closely related to their functions and are conserved during evolution. Hence, prediction of conserved secondary structures from evolutionarily related sequences is one important task in RNA bioinformatics; the methods are useful not only to further functional analyses of ncRNAs but also to improve the accuracy of secondary structure predictions and to find novel functional RNAs from the genome. In this review, I focus on common secondary structure prediction from a given *aligned* RNA sequence, in which one secondary structure whose length is equal to that of the input alignment is predicted. I systematically review and classify existing tools and algorithms for the problem, by utilizing the information employed in the tools and by adopting a unified viewpoint based on maximum expected gain (MEG) estimators. I believe that this classification will allow a deeper understanding of each tool and provide users with useful information for selecting tools for common secondary structure predictions.

Key words Common/consensus secondary structures, Comparative methods, Multiple sequence alignment, Covariation, Mutual information, Phylogenetic tree, Energy model, Probabilistic model, Maximum expected gain (MEG) estimators

1 Introduction

Functional non-coding RNAs (ncRNAs) play essential roles in various biological processes, such as transcription and translation regulation [11, 45, 49]. Not only nucleotide sequences but also secondary structures are closely related to the functions of ncRNAs, so secondary structures are conserved during evolution. Hence, the prediction of these conserved secondary structures (called “common secondary structure prediction” throughout this review) from evolutionarily related RNA sequences is among the most important tasks in RNA bioinformatics, because it provides useful information for further functional analysis of the targeted RNAs [4, 6, 28, 41, 46]. It should be emphasized that common secondary structure predictions are also useful to improve the accuracy of

secondary structure prediction [13, 48] or to find functional RNAs from the genome [15, 66].

Approaches to common secondary structure prediction are divided into two categories with respect to the input: (1) *unaligned* RNA sequences and (2) *aligned* RNA sequences. Common secondary structure predictions from *unaligned* RNA sequences can be solved by utilizing the Sankoff algorithm [54]. However, it is known that the Sankoff algorithm has huge computational costs: $O(L^{3n})$ and $O(L^{2n})$ for time and space, respectively, where L is the length of RNA sequences and n is the number of input sequences. For this reason, the second approach, whose input is *aligned* RNA sequences, is often used in actual analyses. The problem focused on this review is formulated as follows.

Problem 1 (RNA Common Secondary Structure Prediction of Aligned Sequences)

Given an input multiple sequence alignment A, predict a secondary structure y whose length is equal to the length of the input alignment. The secondary structure y is called the common (or consensus) secondary structure of the multiple alignment A.

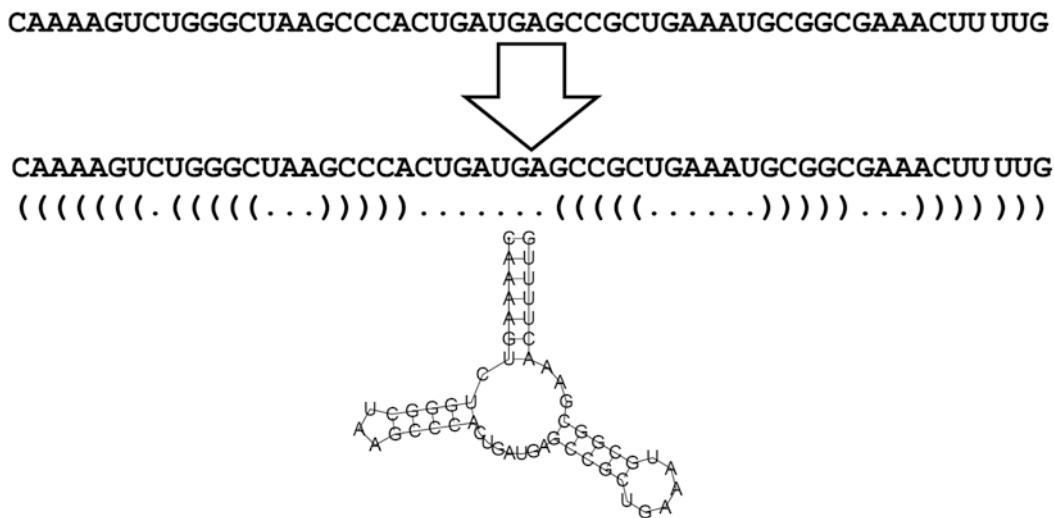
In contrast to conventional RNA secondary structure prediction (see Fig. 1a), the input of common secondary structure prediction is a multiple sequence alignment of RNA sequences (see Fig. 1b) and the output is a secondary structure with the same length as the alignment. In general, the predicted common RNA secondary structure is expected to represent a secondary structure that commonly appears in the input alignments or is conserved during evolution.

To develop tools (or algorithms) for solving this problem, it should be made clear what a *better* common secondary structure is. However, the evaluation method for predicted common RNA secondary structures is *not* trivial. A predicted common secondary structure depends on not only RNA sequences in the alignment but also multiple alignments of input sequences, and in general, no reference (correct) common secondary structures are available (see Note 1). A predicted common secondary structure is therefore evaluated, based on reference RNA secondary structures for each *individual* RNA sequence in the alignment as follows.

Evaluation Procedure 1 (For Problem 1)

Given reference secondary structures for each RNA sequence, evaluate the predicted common secondary structure y as follows. First, map y onto each RNA sequence in the alignment A (see the column “Mapped structure with gaps” in Fig. 2, for example). Second, remove all gaps in each sequence and the corresponding base-pairs in the mapped secondary structure in order to maintain the consistency of the secondary structures (see the column “Mapped structure without gaps” in Fig. 2, for example). Third, calculate the quantities TP,

a Secondary structure prediction



b Common secondary structure prediction

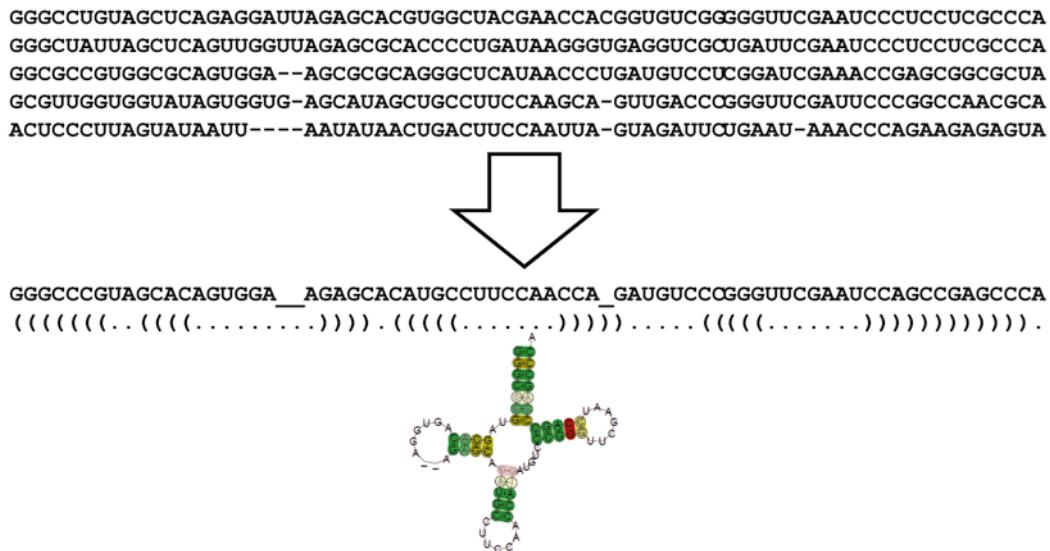


Fig. 1 (a) Conventional RNA secondary structure prediction, in which the input is an individual RNA sequence and the output is an RNA secondary structure of the sequence. (b) Common (or consensus) RNA secondary structure prediction in which the input is a multiple sequence alignment of RNA sequences and the output is an RNA secondary structure whose length is equal to the length of the alignment. The secondary structure is called the common (or consensus) secondary structure

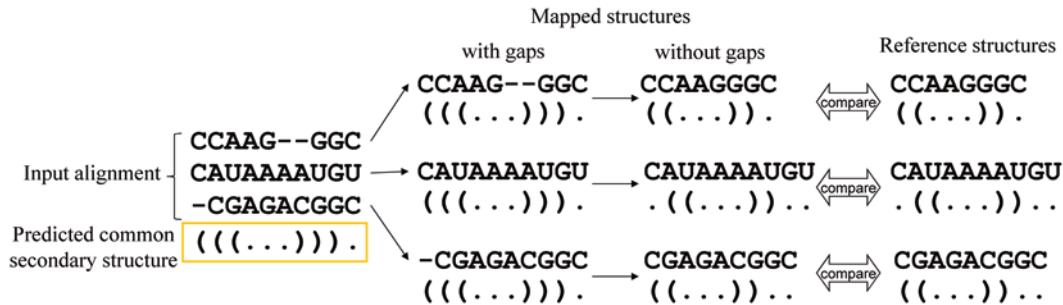


Fig. 2 An evaluation procedure for the predicted common RNA secondary structure of an input alignment, in which the reference secondary structure of each RNA sequence in the alignment is given. This procedure is based on the idea that a common secondary structure should reflect as many of the secondary structures of each RNA sequence in the input alignment as possible. Mapped RNA secondary structures without gaps are computed by getting rid of base-pairs that correspond to gaps. Note that it is difficult to compare a predicted common secondary structure with a *reference* common secondary structure, because the reference common RNA secondary structure for an arbitrary input alignment is not available in general. In most studies of common secondary structure prediction, evaluation is conducted by using this procedure or a variant. See [23] for a more detailed discussion of evaluation procedures for Problem 1

TN, FP, and FN for each mapped secondary structure $y^{(map)}$ with respect to the reference secondary structure (TP, TN, FP, and FN are the respective numbers of true positive, true-negative, false-positive and false-negative base-pairs of a predicted secondary structure with respect to the reference structures). Finally, calculate the evaluation measures sensitivity (SEN), positive predictive value (PPV), and Matthew correlation coefficient (MCC) for the sum of TP, TN, FP, and FN over all the RNA sequences in the alignment A.

Figure 2 shows an illustrative example of Evaluation Procedure 1. Note that there exist a few variants of this procedure (e.g., [5]).

In this review, I aim to classify the existing tools (or algorithms) for Problem 1; These tools are summarized in Table 1, which includes all the tools for common secondary structure prediction as of 17th June 2013. To achieve this aim, I describe the information that is often utilized in common secondary structure predictions, and classify tools from a unified viewpoint based on maximum expected gain (MEG) estimators. I also explain the relations between the MEG estimators and Evaluation Procedure 1.

This review is organized as follows. In Subheading 2, I summarize the information that is commonly utilized when designing algorithms for common secondary structure prediction. In Subheading 3, several concepts to be utilized in the classification of tools are presented, and the currently available tools are classified within this framework.

Table 1

List of tools for common secondary structure prediction from *aligned* sequences

Tool	References	Description
(Without pseudoknot)		
CentroidAlifold	[19, 23]	Achieved superior performance in a recent benchmark (CompaRNA) [48]. Using an MEG estimator with the γ -centroid-type gain function (Subheading 3.3.2) in combination with a mixture probability distribution, including several types of information (Subheading 3.2.4)
ConStruct	[37, 72]	A semi-automatic, graphical tool based on mutual information (Subheading 2.2)
KNetFold	[6]	Computes a consensus RNA secondary structure from an RNA sequence alignment based on machine learning (Bayesian networks)
McCaskill-MEA	[32]	Adopting majority rule with the McCaskill energy model [40], leading to an algorithm that is robust to alignment errors
PETfold	[59]	Considers both phylogenetic information and thermodynamic stability by extending Pfold, in combination with an MEA estimation
Pfold	[34, 35]	Uses phylogenetic tree information with simple SCFG
PhyloRNAalifold	[14]	Incorporates the number of co-varying mutations on the phylogenetic tree of the aligned sequences into the covariance scoring of RNAalifold
PPfold	[63]	A multi-threaded implementation of the Pfold algorithm, which is extended to evolutionary analysis with a flexible probabilistic model for incorporating auxiliary data, such as data from structure probing experiments
RNAalifold	[5, 26, 27]	Considers both thermodynamic stability and co-variation in combination with RIBOSUM-like scoring matrices [33]
RSpredict	[62]	Takes into account sequence covariation and employs effective heuristics for accuracy improvement
(With pseudoknot)		
hxmatch	[73]	Computes consensus structures including pseudoknots based on alignments of a few sequences. The algorithm combines thermodynamic and covariation information to assign scores to all possible base-pairs, the base-pairs are chosen with the help of the maximum weighted matching algorithm
ILM	[52]	Uses mutual information and helix plot in combination with heuristic optimization
IPKnot	[57]	Uses MEG estimators with γ -centroid gains and heuristic probability distribution of RNA interactions together with integer linear programming to compute a decoded RNA secondary structure
MIfold	[12]	A MATLAB(R) toolbox that employs mutual information, or a related covariation measure, to display and predict common RNA secondary structure

To the best of my knowledge, this is a complete list of tools for the problem as of 17 June 2013. Note that tools for common secondary structure prediction from *unaligned* RNA sequences are not included in this list. See Table 2 for further details of the listed tools

Table 2**Comparison of tools (in Table 1) for common secondary structure predictions from aligned sequences**

Name	Software ^a		Used information ^b						Gain ^c	Prob. dist. ^d	
	SA	WS	TS	ML	CV	PI	MI	MR			
(Without pseudoknot)											
CentroidAlifold	✓	✓	✓	✓	✓	✓	✓	✓	✓	γ -cent	Any ^f
ConStruct	✓		✓		✓		✓	✓		na	na
KNetFold		✓	✓				✓			na	na
McCaskill-MEA	✓							✓		Contra	Av(Mc)
PETfold	✓	✓				✓				Contra	Pf+Av(Mc)
Pfold	✓			✓		✓				Delta	Pf
PhyloRNAalifold	✓				✓	✓				Delta	Ra
PPfold	✓			✓		✓			✓	Delta	Pf
RNAalifold	✓	✓	✓		✓					Delta	Ra
RSpredict	✓	✓	✓		✓					na	na
(With pseudoknot)											
hxmatch	✓		✓		✓					na	na
ILM	✓	✓	✓				✓			na	na
IPKnot	✓	✓	✓	✓	✓			✓		γ -cent	Any ^f
MIfold	✓ ^g						✓			na	na

^a Type of software available. SA stand alone, WS web server, TS thermodynamic stability (Subheading 2.1.1)^b In the “Information used” columns, ML machine learning (Subheading 2.1.2); CV covariation (Subheading 2.3); PI phylogenetic (evolutionary) information(Subheading 2.4); MI mutual information (Subheading 2.2); MR majorityrule (Subheading 2.5); EI experimental information (Subheading 2.1.3)^c In the column“Gain,” γ -cent: γ -centroid-type gain function (Subheading 3.3.2); contra: CONTRAfold-type gain function (Subheading 3.3.3)^d In the column “Prob. dist.,” Pf Pfold model (Subheading 3.2.2); Ra RNAalipffold model (Subheading 3.2.1); Av(Mc) averaged probability distribution with McCaskill model (Subheading 3.2.3); “+” indicates a mixture distribution (of several models). “na” means “Not available” due to no use of probabilistic models^e If the method proposed in [16] is used, experimental information derived from SHAPE [36] and PARS [31] is easily incorporated in CentroidAlign^f CentroidAlifold and IPKnot can employ a mixed distribution given by an arbitrary combination of RNAalipffold, Pfold, and an averaged probability distribution based on the McCaskill or CONTRAfold models^gMATLAB codes are available

2 Materials

Several pieces of information are generally utilized in tools and algorithms for predicting common RNA ondary structures. These will now be briefly summarized.

2.1 Fitness to Each Sequence in the Input Alignment

The common RNA secondary structure should be a representative secondary structure among RNA sequences in the alignment. Therefore, the fitness of a predicted common secondary structure to *each* RNA sequence in the alignment is useful information. In particular, in Evaluation Procedure 1, the fitness of a predicted common secondary structure to each RNA sequence is evaluated.

This fitness is based on probabilistic models for RNA secondary structures of individual RNA sequence, such as the energy-based and machine learning models shown in Subheadings 2.1.1 and 2.1.2, respectively. These models provide a probability distribution of secondary structures of a given RNA sequence. ($p(\theta|x)$ denotes a probability distribution of RNA secondary structures for a given RNA sequence x .)

2.1.1 Thermodynamic Stability: Energy-Based Models

Turner's energy model [38] is an energy-based model, which considers the thermodynamic stability of RNA secondary structures. This model is widely utilized in RNA secondary structure predictions, in which experimentally determined energy parameters [38, 39, 75] are employed. In the model, structures with a lower free energy are more stable than those with a higher free energy. Note that Turner's energy model leads to a probabilistic model for RNA sequences, providing a probability distribution of secondary structures, which is called the McCaskill model [40].

2.1.2 Machine Learning (ML) Models

In addition to the energy-based models described in the previous subsection, probabilistic models for RNA secondary structures based on machine learning (ML) approaches have been proposed. In contrast to the energy-based models, machine learning models can automatically learn parameters from training data (i.e., a set of RNA sequences with secondary structures). There are several models based on machine learning which adopt different approaches: (1) Stochastic context free grammar (SCFG) models [10]; (2) the CONTRAfold model [9] (a conditional random field model); (3) the Boltzmann Likelihood (BL) model [1–3]; and (4) non-parametric Bayesian models [56].

See Rivas et al. [51] for detailed comparisons of probabilistic models for RNA secondary structures.

2.1.3 Experimental Information

Recently, experimental techniques to probe RNA structure by high-throughput sequencing (SHAPE [36]; PARS [31]; FragSeq [64]) have enabled genome-wide measurements of RNA structure. Those experimental techniques stochastically estimate the flexibility of an RNA strand, which can be considered as a kind of *loop probability* for every nucleotide in an RNA sequence. Remarkably, secondary structures of long RNA sequences, such as HIV-1 [69], HCV (hepatitis C virus) [44], and large intergenic ncRNA (the steroid receptor RNA activator) [43], have been recently determined by combining those experimental techniques

with computational approaches. If available, such experimental information is useful in common secondary structure predictions, because they provide reliable secondary structures for each RNA sequence.

2.2 Mutual Information

The mutual information of the i th and j th columns in the input alignment is defined by

$$M_{ij} = \sum_{x,y} f_{ij}(XY) \log \frac{f_{ij}(XY)}{f_i(X)f_j(Y)} = KL(f_{ij}(XY) || f_i(X)f_j(Y)) \quad (1)$$

where $f_i(X)$ is the frequency of base X at alignment position i ; $f_{ij}(XY)$ is the joint frequency of finding X in the i th column and Y in the j th column, and $KL(\cdot || \cdot)$ denotes the Kullback–Liebler distance between two probability distributions. As a result, the complete set of mutual information can be represented as an upper triangular matrix: $\{M_{ij}\}_{i < j}$.

Note that the mutual information score makes no use of base-pairing rules of RNA secondary structure. In particular, mutual information does not account for consistent non-compensatory mutations at all, although information about them would be useful when predicting common secondary structures as described in the next subsection.

2.3 Sequence Covariation of Base-Pairs

Because secondary structures of ncRNAs are related to their functions, mutations that preserve base-pairs (i.e., covariations of a base-pair) often occur during evolution. Figure 3 shows an example of covariation of base-pairs of tRNA sequences, in which many covariations of base-pairs are found, especially in the stem parts in the tRNA structure.

The covariation of the i th and j th columns in the input alignment is evaluated by the averaged number of compensatory mutations defined by

$$C_{ij} = \frac{2}{N(N-1)} \sum_{(x,y) \in A} d_{ij}^{x,y} \Pi_{ij}^x \Pi_{ij}^y \quad (2)$$

where N is the number of sequences in the input alignment A . For an RNA sequence x in A , $\Pi_{ij}^x = 1$ if x_i and x_j form a base-pair and $\Pi_{ij}^x = 0$ otherwise, and

$$d_{ij}^{x,y} = 2 - \delta(x_i, y_i) - \delta(x_j, y_j) \quad (3)$$

where δ is the delta function: $\delta(a, b) = 1$ only if $a = b$, and $\delta(a, b) = 0$ otherwise.

For instance, RNAAlifold [65] uses the information of covariation in combination with the thermodynamic stability of common secondary structures.

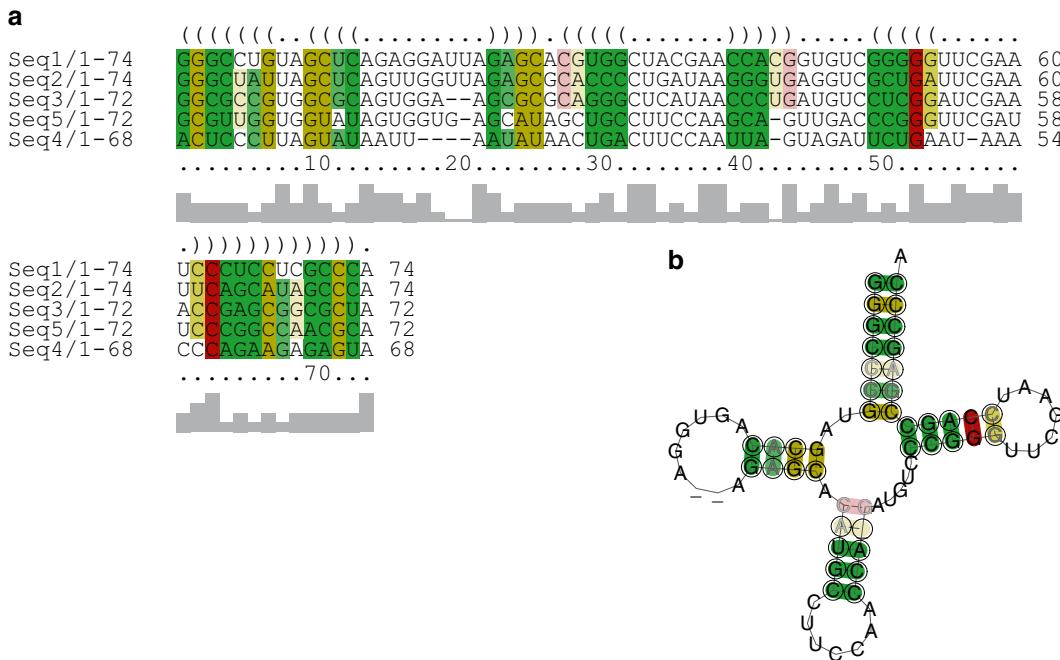


Fig. 3 An example of covariation of base-pairs in an alignment of tRNA sequences: **(a)** a multiple alignment of tRNA sequences and **(b)** a predicted common secondary structure of the alignment. The figures are taken from the output of an example on the RNAalifold [5] Web Server (<http://rna.tbi.univie.ac.at/cgi-bin/RNAalifold.cgi>)

2.4 Phylogenetic (Evolutionary) Information

Because most secondary structures of functional ncRNAs are conserved in evolution, the phylogenetic (evolutionary) information with respect to the input alignment is useful for predicting common secondary structures, and is employed by several tools. Pfold [34, 35] incorporates this information into probabilistic model for common secondary structures (Subheading 3.2.2) for the first time.

2.5 Majority Rule of Base-Pairs

Kiryu et al. [32] proposed the use of the majority rule of base-pairs in predictions of common secondary structures. This rule states that base-pairs supported by many RNA sequences should be included in a predicted common secondary structure. Specifically, Kiryu et al. utilized an averaged probability distribution of secondary structures among RNA sequences to predict a common RNA secondary structure from a given alignment (Subheading 3.2.3).

The aim of this approach is to mitigate alignment errors, because the effects of a minor alignment errors can be disregarded in the prediction of common secondary structures.

3 Methods

3.1 MEG Estimators

In this section, I classify the existing algorithms for common RNA secondary structure prediction (shown in Table 1) from a unified viewpoint, based on a previous study [23] in which the following type of estimator [18, 24] was employed.

$$\hat{y} = \operatorname{argmax}_{y \in S(A)} \sum_{\theta \in \mathcal{Y}} G(\theta, y) p(\theta | A) \quad (4)$$

where $S(A)$ denotes a set of possible secondary structures with length $|A|$ (the length of the alignment A), $G(\theta, y)$ is called a “gain function” and returns a measure of the similarity between two common secondary structures, and $p(\theta | A)$ is a probabilistic distribution on $S(A)$. This type of estimator is an MEG estimator; When the gain function is designed according to accuracy measures for target problems, the MEG estimator is often called a “maximum expected accuracy (MEA) estimator” [18] (see Note 2).

In the following, a common secondary structure $\theta \in S(A)$ is represented as an upper triangular matrix $\theta = \{\theta_{ij}\}_{1 \leq i < j \leq |A|}$. In this matrix $\theta_{ij} = 1$ if the i th column in A forms a base-pair with the j th column in A , and $\theta_{ij} = 0$ otherwise.

The choices of $p(\theta | A)$ and $G(\theta, y)$ are described in Subheadings 3.2 and 3.3, respectively.

3.2 Choice of Probabilistic Models $p(\theta | A)$

The probabilities $p(\theta | A)$ provide a distribution of common RNA secondary structures given a multiple sequence alignment A . This distribution is given by the following models.

3.2.1 RNAalifold Model

The RNAalifold model is a probabilistic version of RNAalifold [27], which provides a probability distribution of common RNA secondary structures given a multiple sequence alignment. The distribution on $S(A)$ is defined by

$$p^{(\text{RNAalifold})}(\theta | A) = \frac{1}{Z(T)} \exp\left(-\frac{E(\theta, A)}{kT} + \text{Cov}(\theta, A)\right) \quad (5)$$

where $E(\theta, A)$ is the averaged free energy of RNA sequences in the alignment with respect to the common secondary structure θ in A and $\text{Cov}(\theta, A)$ is the base covariation (cf. Subheading 2.3) with respect to the common secondary structure θ . The ML estimate of this distribution is equivalent to the prediction of RNAalifold.

Note that the negative part of the exponent in Eq. 5 is called the *pseudo* energy and it plays an essential role in finding ncRNAs from multiple alignments [15, 67].

3.2.2 Pfold Model

The Pfold model [34, 35] incorporates phylogenetic (evolutionary) information about the input alignments into a probabilistic distribution of common secondary structures:

$$p^{(\text{pfold})}(\theta | A) = p^{(\text{pfold})}(\theta | A, T, M) = \frac{p(A | \theta, T)p(\theta | M)}{p(A | T, M)} \quad (6)$$

where T is a phylogenetic tree, A is the input data (i.e., an alignment), M is a prior model for secondary structures (based on SCFGs; see Note 3). Unless the original phylogenetic tree T is obtained, T is taken to be the ML estimate of the tree, T^{ML} , given the model M and the alignment A .

3.2.3 Averaged Probability Distribution of Each RNA Sequence

Predictions based on RNAAlifold and Pfold models tend to be affected by alignment errors in the input alignment. To address this, averaged probability distributions of RNA sequences involved in the input alignment were introduced by Kiryu et al. [32]. This leads to an MEG estimator with the probability distribution

$$p^{(\text{ave})}(\theta | A) = \frac{1}{n} \sum_{x \in A} p(\theta | x) \quad (7)$$

where $p(\theta | x)$ is a probabilistic model for RNA secondary structures, for example, the McCaskill model [40], the CONTRAfold model [9], the BL model [1–3], and others [56]. Note that neither covariation nor phylogenetic information about alignments is considered in this probability distribution.

In [32], the authors utilized the McCaskill model as a probabilistic model for individual RNA sequences (i.e., they used $p(\theta | x)$ in Eq. 14), and showed that their method was more robust with respect to alignment errors than RNAAlifold and Pfold. Using averaged probability distributions for RNA sequences in an input alignment is compatible with Evaluation Procedure 1. See Hamada et al. [23] for a detailed discussion.

3.2.4 Mixture of Several Distributions

Hamada et al. [23] pointed out that arbitrary information can be incorporated into common secondary structure predictions by utilizing a *mixture* of probability distributions. For example, the probability distribution

$$p(\theta | A) = w_1 \cdot p^{(\text{pfold})}(\theta | A) + w_2 \cdot p^{(\text{alifold})}(\theta | A) + w_3 \cdot p^{(\text{ave})}(\theta | A), \quad (8)$$

where w_1 , w_2 , and w_3 are positive values that satisfy $w_1 + w_2 + w_3 = 1$, includes covariation (Subheading 2.3), phylogenetic tree (Subheading 2.4), and majority rule (Subheading 2.5) information.

In CentroidAlifold [23], users can employ a mixed distribution given by an arbitrary combination of RNAAlifold (Subheading 3.2.1), Pfold (Subheading 3.2.2), and an averaged probability distribution

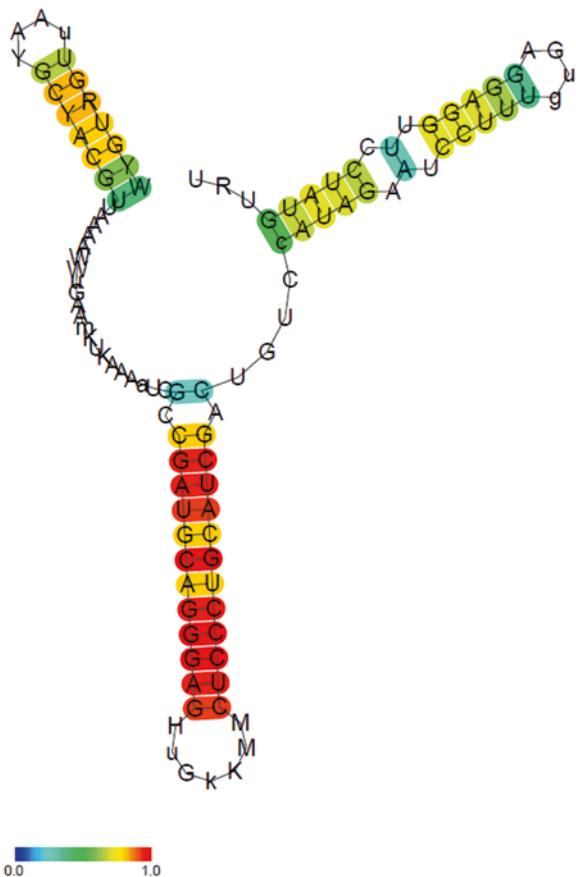


Fig. 4 Example of a predicted common secondary structure from the CentroidAlifold Web Server [55] (<http://www.ncrna.org/centroidfold>). The input is a multiple sequence alignment of the traJ 5' UTR. The *color* of a base-pair indicates the averaged base-pairing probabilities (among RNA sequences in the alignment) of the base-pair, where *warmer colors* represent higher probabilities

(Subheading 3.2.3) based on the McCaskill or CONTRAfold model. An example of a result from the CentroidAlifold Web Server is shown in Fig. 4, in which colors of base-pairs indicate the marginal base-pairing probability with respect to this mixture distribution. Hamada et al. also showed that computation of MEG estimators with a mixture distribution can be easily conducted by utilizing base-pairing probability matrices (see also the next section), when certain gain functions are employed. Moreover, computational experiments indicated that CentroidAlifold with a mixture model is significantly better than RNAAlifold, Pfold, or McCaskill-MEA.

3.3 Choice of Gain Functions $G(\theta, y)$

A choice of the gain function in MEG estimators corresponds to the decoding method (i.e., prediction of one final common secondary structure from the distribution) of common RNA secondary structures, given a probabilistic model of common secondary structures.

3.3.1 The Kronecker Delta Function

A straightforward choice of the gain function is the Kronecker delta function:

$$G^{(\text{delta})}(\theta, y) (= \delta(\theta, y)) = \begin{cases} 1 & \text{if } y \text{ is the exactly same as } \theta \\ 0 & \text{otherwise} \end{cases} \quad (9)$$

The MEG estimator with this gain function is equivalent to the “maximum likelihood estimator” (ML estimator) with respect to a given probabilistic model for RNA common secondary structures (cf. Subheading 3.2), which predicts the secondary structure with the highest probability.

3.3.2 The γ -Centroid-Type Gain Function [19]

It is known that the probability of the ML estimation is extremely small, due to the immense number of secondary structures that could be predicted; this fact is known as the “uncertainty” of the solution, and often leads to issues in bioinformatics [17]. Because the MEG estimator with the delta function considers only the solution with the highest probability, it is affected by this uncertainty. A choice of gain function that partially overcomes this uncertainty of solutions is the γ -centroid-type gain function [19]:

$$G_{\gamma}^{(\text{centroid})}(\theta, y) = \sum_{i,j} \left[\gamma I(\theta_{ij} = 1)I(y_{ij} = 1) + I(\theta_{ij} = 0)I(y_{ij} = 0) \right] \quad (10)$$

where $\gamma > 0$ is a parameter that adjusts the relative importance of the SEN and PPV of base-pairs in a predicted structure (i.e., a larger γ produces more base-pairs in a predicted secondary structure). This gain function is motivated by the concept that more true base-pairs (TP and TN) and fewer false base-pairs (FP and FN) should be predicted [19] when the entire distribution of secondary structures is considered. Note that, when $\gamma = 1$, the gain function is equivalent to that of the centroid estimator [8].

3.3.3 The CONTRAfold-Type Gain Function [9]

In conventional RNA secondary structure predictions, another gain function has been proposed (see Note 4), which is based on the number of accurate predictions of every single position (*not* base-pair) in an RNA sequence (see Note 5). The CONTRAfold-type gain function is

$$G_{\gamma}^{(\text{contra})}(\theta, y) = \sum_{i=1}^{|A|} \left[\gamma \sum_{j:j \neq i} I(\theta_{ij}^* = 1)I(y_{ij}^* = 1) + \prod_{j:j \neq i} I(\theta_{ij}^* = 0)I(y_{ij}^* = 0) \right] \quad (11)$$

where θ^* and y^* are symmetric extensions of θ and y , respectively (i.e., $\theta_{ij}^* = \theta_{ij}$ for $i < j$ and $\theta_{ij}^* = \theta_{ji}$ for $j < i$). This gain function is also applicable to MEG estimators for common secondary structure predictions. It should be emphasized that MEG estimators based on the CONTRAfold-type gain function (for any $\gamma > 0$) do not include centroid estimators, while the γ -centroid-type gain function includes the centroid estimator as a special case (i.e., $\gamma = 1$).

3.3.4 Remarks About Choice of Gain Function

From a theoretical viewpoint, the γ -centroid-type gain function is more appropriate for Evaluation Procedure 1 than either the delta function or the CONTRAfold-type gain function (see Note 6), which is also supported by several empirical (computational) experiments. See Hamada et al. [23] for a detailed discussion.

Another choice of the gain function is MCC (or F -score), which takes a balance between SEN and PPV of base-pairs, and the MEG estimator with this gain function leads to an algorithm that maximizes *pseudo*-expected accuracy [22]. In addition, the estimator with this gain function includes only one parameter for predicting secondary structure (see Note 7).

3.4 Computation of Common Secondary Structure Through MEG Estimators

3.4.1 MEG Estimator with Delta Function

3.4.2 MEG Estimator with γ -Centroid (or CONTRAfold) Type Gain Function

The MEG estimator with the delta function (that predicts the common secondary structure with the highest probability with respect to a given probabilistic model) can be computed by employing a CYK (Cocke–Younger–Kasami)-type algorithm. For example, see [65] for details.

The MEG estimator with the γ -centroid-type (or CONTRAfold-type) gain function is computed based on “base-pairing probability matrices” (BPPMs) (see Note 8) and Nussinov-style dynamic programming (DP) [9, 23]:

$$M_{i,j} = \max \begin{cases} M_{i+1,j} \\ M_{i,j-1} \\ M_{i+1,j-1} + S_{ij} \\ \max_k [M_{i,k} + M_{k+1,j}] \end{cases} \quad (12)$$

where $M_{i,j}$ is the optimal score of the subsequence $x_{i \dots j}$ and S_{ij} is a score computed from the BPPM(s). For instance, for the γ -centroid estimator with the RNAAlipffold model, the score S_{ij} is equal to $S_{ij} = (\gamma + 1)p_{ij}^{(\text{alipffold})} - 1$ where $p_{ij}^{(\text{alipffold})}$ is the base-pairing probability with respect to the RNAAlipffold model. This DP algorithm maximizes the sum of (base-pairing) probabilities $p_{ij}^{(\text{alipffold})}$ which are larger than $1/(\gamma + 1)$, and requires $O(|A|^3)$ time.

3.4.3 MEG Estimators with Averaged Probability Distributions

The MEG estimator with an averaged probability distribution (Subheading 3.2.3) can be computed by using averaged base-pairing probabilities, $\{p_{ij}\}_{i < j}$:

$$\hat{p}_{ij}^{(\text{ave})} = \frac{1}{n} \sum_{x \in A} p_{ij}^{(x)} \quad (13)$$

where

$$p_{ij}^{(x)} = \begin{cases} \sum_{\theta \in S(x)} I(\theta_{\tau(i)\tau(j)} = 1) p(\theta | x') & \text{if both } x_i \text{ and } x_j \text{ are not gaps} \\ 0 & \text{otherwise} \end{cases} \quad (14)$$

In the above, x' is the RNA sequence given by removing gaps from x and n is the number of sequences in the alignment A . The function $\tau(i)$ returns the position in x' corresponding to the position i in x .

The common secondary structure of MEG estimator with the γ -centroid gain function and the averaged probability distribution are computed by using the DP recursion in Eq. 12 where $S_{ij} = (\gamma + 1)p_{ij}^{(\text{ave})} - 1$. This procedure has a time complexity of $O(n|A|^3)$ where n is the number of sequences in the alignment.

3.4.4 MEG Estimators with a Mixture Distribution

The MEG estimator with a mixture of distribution (Subheading 3.2.4) and the delta function (Subheading 3.3.1) cannot be computed efficiently. However, if the γ -centroid-type (or CONTRAfold-type) gain function is utilized, the prediction can be conducted using a similar DP recursion to that in Eq. 12. For instance, the DP recursion of the γ -centroid-type gain function with respect to Eq. 8 is equivalent to the one in Eq. 12 with $S_{ij} = (\gamma + 1)p_{ij}^* - 1$ where

$$p_{ij}^* = w_1 \cdot p_{ij}^{(\text{pfold})} + w_2 \cdot p_{ij}^{(\text{alipffold})} + \frac{w_3}{n} \sum_{x \in A} p_{ij}^{(x)}. \quad (15)$$

In the above, $p_{ij}^{(\text{pfold})}$ and $p_{ij}^{(\text{alipffold})}$ are base-pairing probabilities for the Pfold and RNAAlipffold models, respectively, and $\{p_{ij}^{(x)}\}$ is a base-pairing probability matrix with respect to a probabilistic model for secondary structures of single RNA sequence x (McCaskill or CONTRAfold model). Note that the total computational time of CentroidAlifold with a mixture of distributions still remains $O(n|A|^3)$.

3.4.5 MEG Estimators with Probability Distribution Including Pseudoknots

Using probability distributions of secondary structures with pseudoknots in MEG estimators generally has higher computational cost [42]. To overcome this, for example, IPKnot [57] utilizes an approximated method for determining the probability distribution as along with integer linear programming for predicting a final common secondary structure.

3.5 A Classification of Tools for Problem 1

Table 1 shows a comprehensive list of tools for common secondary structure prediction from aligned RNA sequences (in alphabetical order within groups that do or do not consider pseudoknots (see Note 9)). To the best of my knowledge, Table 1 is a complete list of tools for Problem 1 as of 17 June 2013.

In Table 2, the tools in Table 1 are classified based on the considerations of Subheadings 2 and 3. The classification leads to much useful information: (1) the pros and cons of each tool; (2) the similarity (or dissimilarity) among tools; (3) which tools are more suited to Evaluation Procedure 1; and (4) a unified framework within which to design algorithms for Problem 1. I believe that the classification will bring a deeper understanding of each tool, although several tools (which are not based on probabilistic models and depend fundamentally on heuristic approaches) cannot be classified in terms of MEG estimators.

3.6 Discussion

3.6.1 Multiple Sequence Alignment of RNA Sequences

Predicting a multiple sequence alignment (point estimation) from unaligned sequences is not reliable because the probability of the alignment becomes extremely small. This is called the “uncertainty” of alignments which raises serious issues in bioinformatics [17]. In one *Science* paper [74], for instance, the authors argued that the uncertainty of multiple sequence alignment greatly influences phylogenetic topology estimations: phylogenetic topologies estimated from multiple alignments predicted by five widely used aligners are different from one another. Similarly, point estimation of multiple sequence alignment will greatly affect consensus secondary structure prediction.

In Problem 1, because the quality of the multiple sequence alignment influences the prediction of common secondary structure, the input multiple alignment should be given by a multiple aligner which is designed specifically for RNA sequences. Although strict algorithms for multiple alignments taking into account secondary structures are equivalent to the Sankoff algorithm [54] and have huge computational costs, several multiple aligners which are fast enough to align long RNA sequences are available: these are CentroidAlign [20, 76], R-coffee [71], PicXXA-R [53], DAFS [58], and MAFFT [30]. In those multiple aligners, not only nucleotide sequences but also secondary structures are considered in the alignment, and they are, therefore, suitable for generating input multiple alignments for Problem 1.

Because the common secondary structure depends on multiple alignment, an approach adopted in RNAG [70] also seems promising. This approach iteratively samples from the conditional probability distributions $P(\text{Structure} | \text{Alignment})$ and $P(\text{Alignment} | \text{Structure})$. Note, however, that RNAG does not solve Problem 1 directly.

3.6.2 Improvement of RNA Secondary Structure Predictions Using Common Secondary Structure

Although several studies have been conducted for RNA secondary structure predictions for a single RNA sequence [9, 19, 38, 47], the accuracy is still limited, especially for long RNA sequences. By employing comparative approaches using homologous sequence information, the accuracy of RNA secondary structure prediction will be improved. In many cases, homologous RNA sequences of

the target RNA sequence are obtained, and someone would like to know the common secondary structure of those sequences. Gardner and Giegerich [13] introduced three approaches for comparative analysis of RNA sequences, and common secondary structure prediction is essentially utilized in the first of these. However, if the aim is to improve the accuracy of secondary structure predictions, common secondary structure prediction is not always the best solution, because it is not designed to predict the optimal secondary structure of a specific target RNA sequence. If you have a target RNA sequence for which the secondary structure is to be predicted, the approach adopted by the CentroidHomfold [21, 25] software is more appropriate than a method based on common RNA secondary structure prediction.

3.6.3 How to Incorporate Several Pieces of Information in Algorithms

As shown in this review, there are two ways to incorporate several pieces of information into an algorithm for common secondary structure prediction. The first approach is to modify the (internal) algorithm itself in order to handle the additional information. For example, PhyloRNAAlifold [14] incorporates phylogenetic information into the RNAAlifold algorithm by modifying the internal algorithm and PPfold [63] modifies the Pfold algorithm to handle experimental information. The drawbacks of this approach are the relatively large implementation cost and the heuristic combination of the information.

On the other hand, another approach adopted in CentroidAlifold [23] is promising because it can easily incorporate many pieces of information into predictions if a base-pairing probability matrix is available. Because the approach depends on only base-pairing probability matrices, and does not depend on the detailed design of the algorithm, it is easy to implement an algorithm using a mixture of distributions.

Moreover, a method to update a base-pairing probability matrix (computed using sequence information only) which incorporates experimental information [16] has recently been proposed. The method is independent of the probabilistic models of RNA secondary structures, and is suitable for incorporating experimental information into common RNA secondary structure prediction. A more sophisticated method by Washietl et al. [68] can also be used to incorporate experimental information into common secondary structure predictions, because it produces a BPPM that takes experimental information into account.

3.6.4 A Problem that Is Mathematically Related to Problem 1

Problem 1, which is considered in this paper, can be extended to predictions of RNA–RNA interactions, another important task in RNA bioinformatics (e.g., [29, 50]).

Problem 2 (Common Joint Structure Predictions of Two Aligned RNA Sequences)

Given two multiple alignments A_1 and A_2 of RNA sequences, then predict a joint secondary structure between A_1 and A_2 .

Because the mathematical structure of Problem 2 is similar to that of Problem 1, the ideas utilized in designing the algorithms for common secondary structure predictions can be adopted in the development of methods for this new problem. In fact, Seemann et al. [60, 61] have employed similar idea, adapting the PETfold algorithm to Problem 2 (implemented in the PETcofold software). Note that the problem of (pairwise) alignment between two multiple *alignments* (cf. see [24] for the details) has a similar mathematical structure to Problem 1.

3.7 Conclusion

In this review, I focused on RNA secondary structure predictions from *aligned* RNA sequences, in which a secondary structure whose length is equal to the length of the input alignment is predicted. A predicted common secondary structure is useful not only for further functional analyses of the ncRNAs being studied but also for improving RNA secondary structure predictions and for finding ncRNAs in genomes. In this review, I systematically classified existing algorithms on the basis of (1) the information utilized in the algorithms and (2) the corresponding MEG estimators, which consist of a gain function and a probability distribution of common secondary structures. This classification will provide a deeper understanding of each algorithm.

4 Notes

- Reference common secondary structures are available for only reference multiple sequence alignments in the Rfam database [7] (<http://rfam.sanger.ac.uk/>).
- The gain $G(\theta, y)$ is equal to the accuracy measure $\text{Acc}(\theta, y)$ for a prediction and references, so the MEG estimator maximizes the expected accuracy under a given probabilistic distribution.
- The SCFG is based on the rules $S \rightarrow LS \mid L$, $F \rightarrow dFd \mid LS$, $L \rightarrow s \mid dFd$.
- Historically, the CONTRAfold-type gain function was proposed earlier than the γ -centroid-type gain function.
- This is not consistent with Evaluation Procedure 1, because accurate predictions of *base-pairs* with respect to reference structures are evaluated in it.
- The CONTRAfold-type gain function has a bias toward accurate predictions of base-pairs, compared to the γ -centroid-type gain function.
- The γ -centroid-type and CONTRAfold-type gain functions contain a parameter adjusting the ratio of SEN and PPV for a predicted secondary structure.

8. The BPPM is a probability matrix $\{p_{ij}\}$ in which p_{ij} is the marginal probability that the i th base x_i and the j th base x_j form a base-pair with respect to a given probabilistic distribution of secondary structures. For many probabilistic models, including the McCaskill model and the CONTRAfold model, the BPPM for a given sequence can be computed efficiently by utilizing inside–outside algorithms. See [40] for the details.
9. Tools for predicting common secondary structures *without* pseudoknots are much faster than those for predicting secondary structures *with* pseudoknots.

Acknowledgement

This work was supported in part by MEXT KAKENHI (Grant-in-Aid for Young Scientists (A): 24680031; Grant-in-Aid for Scientific Research (A): 25240044).

References

1. Andronescu M, Condon A, Hoos HH, Mathews DH, Murphy KP (2007) Efficient parameter estimation for RNA secondary structure prediction. *Bioinformatics* 23(13):19–28
2. Andronescu M, Condon A, Hoos HH, Mathews DH, Murphy KP (2010) Computational approaches for RNA energy parameter estimation. *RNA* 16(12):2304–2318
3. Andronescu MS, Pop C, Condon AE (2010) Improved free energy parameters for RNA pseudoknotted secondary structure prediction. *RNA* 16(1):26–42
4. Balik A, Penn AC, Nemoda Z, Greger IH (2013) Activity-regulated RNA editing in select neuronal subfields in hippocampus. *Nucl Acids Res* 41(2):1124–1134
5. Bernhart SH, Hofacker IL, Will S, Gruber AR, Stadler PF (2008) RNAalifold: improved consensus structure prediction for RNA alignments. *BMC Bioinform* 9:474
6. Bindewald E, Shapiro BA (2006) RNA secondary structure prediction from sequence alignments using a network of k-nearest neighbor classifiers. *RNA* 12(3):342–352
7. Burge SW, Daub J, Eberhardt R, Tate J, Barquist L, Nawrocki EP, Eddy SR, Gardner PP, Bateman A (2013) Rfam 11.0: 10 years of RNA families. *Nucl Acids Res* 41(Database issue):D226–D232
8. Carvalho LE, Lawrence CE (2008) Centroid estimation in discrete high-dimensional spaces with applications in biology. *Proc Natl Acad Sci USA* 105(9):3209–3214
9. Do CB, Woods DA, Batzoglou S (2006) CONTRAfold: RNA secondary structure prediction without physics-based models. *Bioinformatics* 22(14):e90–e98
10. Dowell RD, Eddy SR (2004) Evaluation of several lightweight stochastic context-free grammars for RNA secondary structure prediction. *BMC Bioinform* 5:71
11. Esteller M (2011) Non-coding RNAs in human disease. *Nat Rev Genet* 12(12):861–874
12. Freyhult E, Moulton V, Gardner P (2005) Predicting RNA structure using mutual information. *Appl Bioinform* 4(1):53–59
13. Gardner PP, Giegerich R (2004) A comprehensive comparison of comparative RNA structure prediction approaches. *BMC Bioinform* 5:140
14. Ge P, Zhang S (2013) Incorporating phylogenetic-based covarying mutations into RNAalifold for RNA consensus structure prediction. *BMC Bioinform* 14(1):142
15. Gruber AR, Findeiss S, Washietl S, Hofacker IL, Stadler PF (2010) Rnaz 2.0: improved noncoding RNA detection. *Pac Symp Biocomput* 15:69–79
16. Hamada M (2012) Direct updating of an RNA base-pairing probability matrix with marginal probability constraints. *J Comput Biol* 19(12): 1265–1276
17. Hamada M (2014) Fighting against uncertainty: an essential issue in bioinformatics. *Briefings Bioinform* 15(5):748–767
18. Hamada M, Asai K (2012) A classification of bioinformatics algorithms from the viewpoint

- of maximizing expected accuracy (MEA). *J Comput Biol* 19(5):532–549
19. Hamada M, Kiryu H, Sato K, Mituyama T, Asai K (2009) Prediction of RNA secondary structure using generalized centroid estimators. *Bioinformatics* 25(4):465–473
 20. Hamada M, Sato K, Kiryu H, Mituyama T, Asai K (2009) CentroidAlign: fast and accurate aligner for structured RNAs by maximizing expected sum-of-pairs score. *Bioinformatics* 25(24):3236–3243
 21. Hamada M, Sato K, Kiryu H, Mituyama T, Asai K (2009) Predictions of RNA secondary structure by combining homologous sequence information. *Bioinformatics* 25(12):i330–i338
 22. Hamada M, Sato K, Asai K (2010) Prediction of RNA secondary structure by maximizing pseudo-expected accuracy. *BMC Bioinform* 11:586
 23. Hamada M, Sato K, Asai K (2011) Improving the accuracy of predicting secondary structure for aligned RNA sequences. *Nucl Acids Res* 39(2):393–402
 24. Hamada M, Kiryu H, Iwasaki W, Asai K (2011) Generalized centroid estimators in bioinformatics. *PLoS ONE* 6(2):e16450
 25. Hamada M, Yamada K, Sato K, Frith MC, Asai K (2011) CentroidHomfold-LAST: accurate prediction of RNA secondary structure using automatically collected homologous sequences. *Nucl Acids Res* 39(Web Server issue):W100–W106
 26. Hofacker IL (2007) RNA consensus structure prediction with RNAalifold. *Methods Mol Biol* 395:527–544
 27. Hofacker IL, Fekete M, Stadler PF (2002) Secondary structure prediction for aligned RNA sequences. *J Mol Biol* 319(5):1059–1066
 28. Jager D, Pernitzsch SR, Richter AS, Backofen R, Sharma CM, Schmitz RA (2012) An archaeal sRNA targeting cis- and trans-encoded mRNAs via two distinct domains. *Nucl Acids Res* 40(21):10964–10979
 29. Kato Y, Sato K, Hamada M, Watanabe Y, Asai K, Akutsu T (2010) RactIP: fast and accurate prediction of RNA-RNA interaction using integer programming. *Bioinformatics* 26(18):i460–i466
 30. Katoh K, Toh H (2008) Improved accuracy of multiple ncRNA alignment by incorporating structural information into a MAFFT-based framework. *BMC Bioinform* 9:212
 31. Kertesz M, Wan Y, Mazor E, Rinn JL, Nutter RC, Chang HY, Segal E (2010) Genome-wide measurement of RNA secondary structure in yeast. *Nature* 467(7311):103–107
 32. Kiryu H, Kin T, Asai K (2007) Robust prediction of consensus secondary structures using averaged base-pairing probability matrices. *Bioinformatics* 23(4):434–441
 33. Klein RJ, Eddy SR (2003) RSEARCH: finding homologs of single structured RNA sequences. *BMC Bioinform* 4:44
 34. Knudsen B, Hein J (1999) RNA secondary structure prediction using stochastic context-free grammars and evolutionary history. *Bioinformatics* 15(6):446–454
 35. Knudsen B, Hein J (2003) Pfold: RNA secondary structure prediction using stochastic context-free grammars. *Nucl Acids Res* 31(13):3423–3428
 36. Low JT, Weeks KM (2010) SHAPE-directed RNA secondary structure prediction. *Methods* 52(2):150–158
 37. Luck R, Graf S, Steger G (1999) ConStruct: a tool for thermodynamic controlled prediction of conserved secondary structure. *Nucl Acids Res* 27(21):4208–4217
 38. Mathews DH, Sabina J, Zuker M, Turner DH (1999) Expanded sequence dependence of thermodynamic parameters improves prediction of RNA secondary structure. *J Mol Biol* 288(5):911–940
 39. Mathews DH, Disney MD, Childs JL, Schroeder SJ, Zuker M, Turner DH (2004) Incorporating chemical modification constraints into a dynamic programming algorithm for prediction of RNA secondary structure. *Proc Natl Acad Sci USA* 101(19):7287–7292
 40. McCaskill JS (1990) The equilibrium partition function and base-pair binding probabilities for RNA secondary structure. *Biopolymers* 29(6–7):1105–1119
 41. Meer EJ, Wang DO, Kim S, Barr I, Guo F, Martin KC (2012) Identification of a cis-acting element that localizes mrna to synapses. *Proc Natl Acad Sci* 109(12):4639–4644
 42. Nebel ME, Weinberg F (2012) Algebraic and combinatorial properties of common RNA pseudoknot classes with applications. *J Comput Biol* 19(10):1134–1150
 43. Novikova IV, Hennelly SP, Sanbonmatsu KY (2012) Structural architecture of the human long non-coding RNA, steroid receptor RNA activator. *Nucl Acids Res* 40(11):5034–5051
 44. Pang PS, Elazar M, Pham EA, Glenn JS (2011) Simplified RNA secondary structure mapping by automation of SHAPE data analysis. *Nucl Acids Res* 39(22):e151
 45. Pauli A, Rinn JL, Schier AF (2011) Non-coding RNAs as regulators of embryogenesis. *Nat Rev Genet* 12(2):136–149

46. Penn AC, Balik A, Greger IH (2013) Steric antisense inhibition of AMPA receptor Q/R editing reveals tight coupling to intronic editing sites and splicing. *Nucl Acids Res* 41(2):1113–1123
47. Proctor JR, Meyer IM (2013) COFOLD: an RNA secondary structure prediction method that takes co-transcriptional folding into account. *Nucl Acids Res* 41(9):e102
48. Puton T, Kozlowski LP, Rother KM, Bujnicki JM (2013) CompaRNA: a server for continuous benchmarking of automated methods for RNA secondary structure prediction. *Nucl Acids Res* 41(7):4307–4323
49. Qureshi IA, Mehler MF (2012) Emerging roles of non-coding RNAs in brain evolution, development, plasticity and disease. *Nat Rev Neurosci* 13(8):528–541
50. Richter AS, Backofen R (2012) Accessibility and conservation: general features of bacterial small RNA–mRNA interactions? *RNA Biol* 9(7):954–965
51. Rivas E, Lang R, Eddy SR (2012) A range of complex probabilistic models for RNA secondary structure prediction that includes the nearest-neighbor model and more. *RNA* 18(2):193–212
52. Ruan J, Stormo GD, Zhang W (2004) An iterated loop matching approach to the prediction of RNA secondary structures with pseudoknots. *Bioinformatics* 20(1):58–66
53. Sahraeian SM, Yoon BJ (2011) PicXAA-R: efficient structural alignment of multiple RNA sequences using a greedy approach. *BMC Bioinform* 12(Suppl 1):S38
54. Sankoff D (1985) Simultaneous solution of the RNA folding alignment and protosequence problems. *SIAM J Appl Math* 45:810–825
55. Sato K, Hamada M, Asai K, Mituyama T (2009) CENTROIDFOLD: a web server for RNA secondary structure prediction. *Nucl Acids Res* 37(Web Server issue):W277–W280
56. Sato K, Hamada M, Mituyama T, Asai K, Sakakibara Y (2010) A non-parametric bayesian approach for predicting rna secondary structures. *J Bioinform Comput Biol* 8(4):727–742
57. Sato K, Kato Y, Hamada M, Akutsu T, Asai K (2011) IPknot: fast and accurate prediction of RNA secondary structures with pseudoknots using integer programming. *Bioinformatics* 27(13):85–93
58. Sato K, Kato Y, Akutsu T, Asai K, Sakakibara Y (2012) DAFS: simultaneous aligning and folding of RNA sequences via dual decomposition. *Bioinformatics* 28(24):3218–3224
59. Seemann SE, Gorodkin J, Backofen R (2008) Unifying evolutionary and thermodynamic information for RNA folding of multiple alignments. *Nucl Acids Res* 36(20):6355–6362
60. Seemann SE, Menzel P, Backofen R, Gorodkin J (2011) The PETfold and PETcofold web servers for intra- and intermolecular structures of multiple RNA sequences. *Nucl Acids Res* 39(Web Server issue):W107–W111
61. Seemann SE, Richter AS, Gesell T, Backofen R, Gorodkin J (2011) PETcofold: predicting conserved interactions and structures of two multiple alignments of RNA sequences. *Bioinformatics* 27(2):211–219
62. Spirollari J, Wang JT, Zhang K, Bellofatto V, Park Y, Shapiro BA (2009) Predicting consensus structures for RNA alignments via pseudo-energy minimization. *Bioinform Biol Insights* 3:51–69
63. Sukosd Z, Knudsen B, Kjems J, Pedersen CN (2012) PPfold 3.0: fast RNA secondary structure prediction using phylogeny and auxiliary data. *Bioinformatics* 28(20):2691–2692
64. Underwood JG, Uzilov AV, Katzman S, Onodera CS, Mainzer JE, Mathews DH, Lowe TM, Salama SR, Haussler D (2010) FragSeq: transcriptome-wide RNA structure probing using high-throughput sequencing. *Nat Methods* 7(12):995–1001
65. Washietl S, Hofacker IL (2004) Consensus folding of aligned sequences as a new measure for the detection of functional RNAs by comparative genomics. *J Mol Biol* 342(1):19–30
66. Washietl S, Hofacker IL, Lukasser M, Huttenhofer A, Stadler PF (2005) Mapping of conserved RNA secondary structures predicts thousands of functional noncoding RNAs in the human genome. *Nat Biotechnol* 23(11):1383–1390
67. Washietl S, Hofacker IL, Stadler PF (2005) Fast and reliable prediction of noncoding RNAs. *Proc Natl Acad Sci USA* 102(7):2454–2459
68. Washietl S, Hofacker IL, Stadler PF, Kellis M (2012) RNA folding with soft constraints: reconciliation of probing data and thermodynamic secondary structure prediction. *Nucl Acids Res* 40(10):4261–4272
69. Watts JM, Dang KK, Gorelick RJ, Leonard CW, Bess JW, Swanstrom R, Burch CL, Weeks KM (2009) Architecture and secondary structure of an entire HIV-1 RNA genome. *Nature* 460(7256):711–716
70. Wei D, Alpert LV, Lawrence CE (2011) RNAG: a new Gibbs sampler for predicting RNA secondary structure for unaligned sequences. *Bioinformatics* 27(18):2486–2493
71. Wilm A, Higgins DG, Notredame C (2008) R-Coffee: a method for multiple alignment of non-coding RNA. *Nucl Acids Res* 36(9):e52

72. Wilm A, Linnenbrink K, Steger G (2008) ConStruct: improved construction of RNA consensus structures. *BMC Bioinform* 9:219
73. Witwer C, Hofacker IL, Stadler PF (2004) Prediction of consensus RNA secondary structures including pseudoknots. *IEEE/ACM Trans Comput Biol Bioinform* 1(2):66–77
74. Wong KM, Suchard MA, Huelsenbeck JP (2008) Alignment uncertainty and genomic analysis. *Science* 319(5862):473–476
75. Xia T, SantaLucia J, Burkard ME, Kierzek R, Schroeder SJ, Jiao X, Cox C, Turner DH (1998) Thermodynamic parameters for an expanded nearest-neighbor model for formation of RNA duplexes with Watson–Crick base-pairs. *Biochemistry* 37(42):14719–14735
76. Yonemoto H, Asai K, Hamada M (2013) CentroidAlign-Web: a fast and accurate multiple aligner for long non-coding RNAs. *Int J Mol Sci* 14(3):6144–6156

Chapter 3

A Simple Protocol for the Inference of RNA Global Pairwise Alignments

Eugenio Mattei, Manuela Helmer-Citterich, and Fabrizio Ferrè

Abstract

RNA alignment is an important step in the annotation and characterization of unknown RNAs, and several methods have been developed to meet the need of fast and accurate alignments. Being the performances of the aligning methods affected by the input RNA features, finding the most suitable method is not trivial. Indeed, no available method clearly outperforms the others. Here we present a simple workflow to help choosing the more suitable method for RNA pairwise alignment. We tested the performances of six algorithms, based on different approaches, on datasets created by merging publicly available datasets of known or curated RNA secondary structure annotations with datasets of curated RNA alignments. Then, we simulated the frequent case where the secondary structure is unknown by using the same alignment datasets but ignoring the known structure and instead predicting it. In conclusion, the proposed workflow for pairwise RNA alignment depends on the input RNA primary sequence identity and the availability of reliable secondary structures.

Key words RNA alignment, RNA structure comparison, RNA sequence–structure relationship, RNA functional annotation, Computational biology

1 Introduction

RNAs are complex molecules that can fold, by nucleotide pairing, into intricate secondary and tertiary structures, which play a leading role in influencing their function. Accordingly, the need for instruments for a structure-based functional characterization is becoming more pressing. In recent years, the development of new sequencing technologies [1] led to the discovery of several novel RNA transcript variants and entirely new classes of ncRNAs [2, 3]. The characterization and annotation of these molecules is in general facilitated by the search for similarities among annotated RNAs. Yet, RNAs are more subject to structure rather than sequence constraints in order to preserve their secondary structure upon genomic variations [4], and therefore sequence-only comparison can be often misleading when comparing homologous

but divergent RNA molecules. It was demonstrated, indeed, that the inclusion of secondary structure information is mandatory to improve comparison results when sequence identity drops under 50–60 % [5]. Hence, the search for evolutionary structural signatures inspired the development of many comparison tools [6]. Structure comparison plays an important role not only in RNA alignment and classification but also in ncRNA annotation, motifs finding and phylogenetic inference. Despite its importance, the structural alignment of RNAs is still an open issue. Several alignment tools have been developed using different kinds of approaches. Sankoff’s dynamic programming algorithm [7] represents the primary reference for structural RNA alignment but its high computational complexity ($O(N^6)$ in time and $O(N^4)$ in space) motivated the search for more efficient algorithms. Lower complexity has been reached by different methods using for example dynamic programming [8–10], formal grammars [11], genetic algorithms [12, 13], and tree-based encodings [14, 15].

Given two RNAs to be aligned, the selection of the best procedure is not trivial, given also the absence in the current literature of a thorough benchmark of the available algorithms. Herein, we propose a simple procedure that can guide the selection of the most reliable approach, which is motivated by testing several popular algorithms on different datasets and conditions. As shown by our results, the discriminating factors are the sequence identity of the two input RNAs and the availability of reliable secondary structures.

2 Materials

2.1 Computational Methods for RNA Alignment Tested in This Work

LocARNA [10] uses a folding and aligning strategy to perform pairwise alignments. The latest version (v. 1.7.10) can be downloaded from <http://www.bioinf.uni-freiburg.de/Software/LocARNA/> and requires the Vienna package [16] that can be downloaded from <http://www.tbi.univie.ac.at/~ronny/RNA/index.html>. LocARNA can be also accessed through a Web interface (<http://rna.informatik.uni-freiburg.de:8080/LocARNA/Input.jsp>). Detailed installation instructions are included in the package (see **Note 1**). LocARNA uses a two-step procedure to align RNAs. In the first place, a base pair probability matrix for each input sequence is computed using RNAfold (included in the Vienna package). Then the two matrices are used as a guide to find the optimal alignment.

RNA StrAT [15] employs a tree-based strategy to perform pairwise structural alignments. The latest version (v. 7.1) is available upon request; instructions and contact information can be found at <http://www-lbit.iro.umontreal.ca/rnastrat/>. RNA StrAT uses a three-step procedure to align RNA secondary structures. Firstly, the two input secondary structures are broken down into stems and stem-loop structures and then these substructures

are compared using a variant of the tree edit distance. The scores obtained from the previous calculation are then used to guide the alignment of the stem and stem-loop structures. During the alignment, also the nucleotidic sequence is taken into account to improve the alignment performances.

needle from the EMBOSS package [17] performs RNA sequence alignments using an implementation of the Needleman–Wunsch algorithm with affine gaps. *needle* can be downloaded from <http://emboss.open-bio.org/html/adm/ch01s01.html> (*see Note 2*); a detailed guide is included in the Web page.

2.2 Test Datasets

We built three datasets to test the alignment performances of the six selected alignment methods. We retrieved curated secondary structures from the RNA STRAND [18] and RNAspa [19] datasets. RNA STRAND integrates information about known RNA secondary structure of any type and from different organisms, retrieved from several public databases. Instead, the RNAspa dataset is a collection of curated secondary structures from Rfam. Datasets of curated sequence alignments were retrieved from RNASTAR [20] and bralibase II [5]. Bralibase II is a collection of RNA alignment datasets proposed for benchmarking of alignment algorithms. Among the available datasets supplied by bralibase, we selected the dataset 2 including pairwise tRNA alignments. RNASTAR includes refined Rfam alignments that were manually curated using structural information from the Protein Data Bank, PDB [21]. Since these curated datasets of alignments do not provide secondary structure annotation, we combined together structural datasets and secondary structure datasets to obtain a collection of curated alignments of known RNA structures. More specifically, we used RNA STRAND secondary structure for RNASTAR alignments, RNAspa secondary structure for bralibase alignments, and finally the remaining RNAs in RNAspa for Rfam alignments.

The sum-of-pairs (SPS) score [5] was employed as measure to evaluate the performances of the alignment methods. SPS is defined as the number of correct pairs (aligned nucleotide pairs found in the reference alignment, i.e., the correctly aligned positions) over the total number of predicted pairs, and it can be considered as a measure of the sensitivity of a method. An SPS score of 0 indicates two completely different alignments (i.e., the alignment reconstructed from the algorithm is completely wrong); conversely, a score of 1 indicates that the reconstructed alignment is identical to the reference alignment.

2.3 Hardware Requirements

No specific requirements are needed to handle and align RNA sequences and structures. The proposed methods work well on normal desktop computers running Linux. Nevertheless all the methods described in this paper have a Web-based interface that can be used instead of the command line version.

3 Methods

In order to select the best alignment tool to use according to the sequence identity of the two input RNAs and the availability of their secondary structure, we tested six different aligners using three different datasets. As a result, we propose a workflow that can be used as a guideline for the alignment of two given RNAs (Fig. 1).

3.1 Comparing Algorithms Performance

We run six different RNA alignment tools on the datasets presented in Subheading 2.2, namely *needle*, LocARNA, RNA StrAT, RNAforester (included in Vienna package), RNAdistance (also included in Vienna package), and gardenia [14]. These algorithms can be divided into three classes according to their approach: sequence-based (*needle*), folding and aligning (LocARNA), tree-based (all the others). We used *needle* as a measure of how much the secondary structure is important for the alignment of two RNAs; the less the sequence-based alignment is correct, the more the secondary structure can give a positive contribution in guiding the alignment.

Figure 2 shows the results of the structure-based aligners at different levels of sequence identity, measured as SPS (see Subheading 2.2). Specifically, when sequence identity is lower than

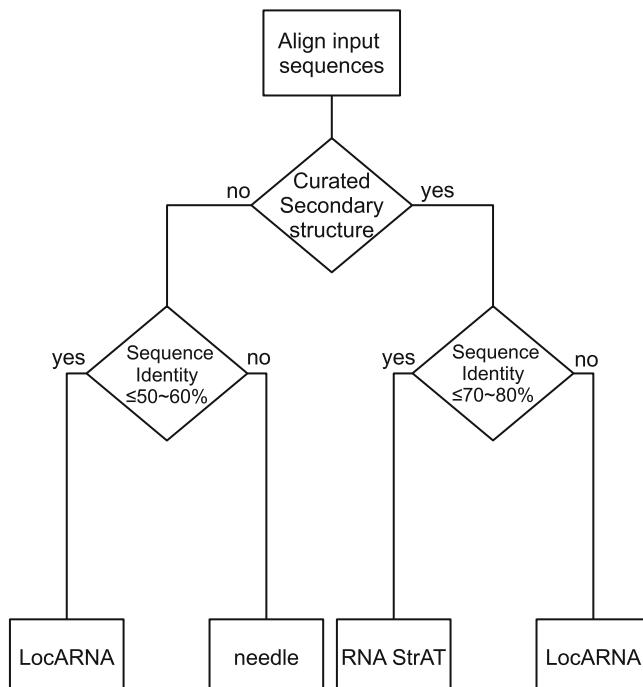


Fig. 1 The proposed workflow, showing the required steps to find the best alignment approach according to the features of the input RNA sequences

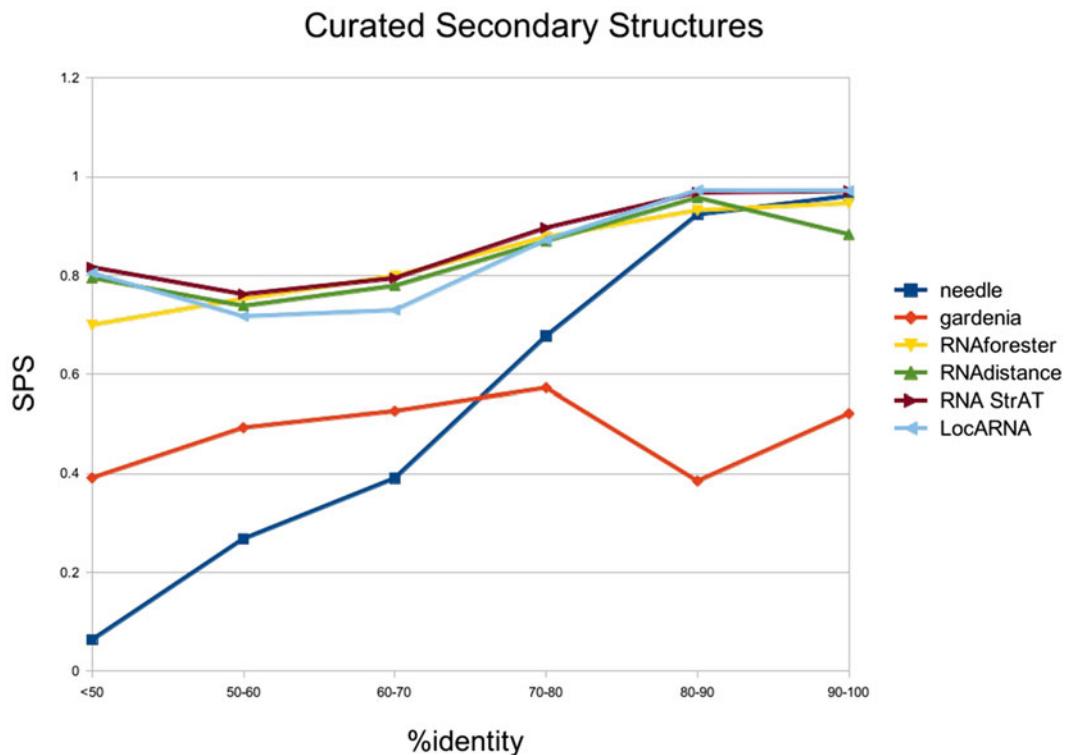


Fig. 2 Alignment performances of the six tested methods on datasets composed by curated structures and alignments. Employed methods are *needle*, LocARNA, RNA StrAT, gardenia, RNAforester, RNAdistance. Alignment accuracy is evaluated as the sum-of-pairs (SPS, described in Subheading 2.2), at different intervals of sequence identity as measured using *needle*

70–80 % RNA StrAT shows the best performances, while above 70–80 % LocARNA is the most accurate (*see Note 3*). Moreover, we run the same test using predicted secondary structures instead of curated ones in order to evaluate the performances in a case of unknown secondary structure, which is the most frequent setting (Fig. 3). In this test, the reference alignment is used as before, but the input structures passed to the alignment algorithms are predicted by RNA fold (included in Vienna package) instead of using the real or curated ones. Even if databases of RNA known structures are growing in size, in the majority of cases the user has only access to their sequences. Our results show that, when the structure is predicted, structural information leads to alignments that are worst than the ones obtained with a simple sequence alignment. LocARNA represents an exception because rather than using directly the predicted secondary structure, we run it using only sequence information and let it compute secondary structures while aligning. In conclusion, when the secondary structure is known we suggest using LocARNA for RNAs sharing more than 70–80 % sequence identity, or otherwise use RNA StrAT.

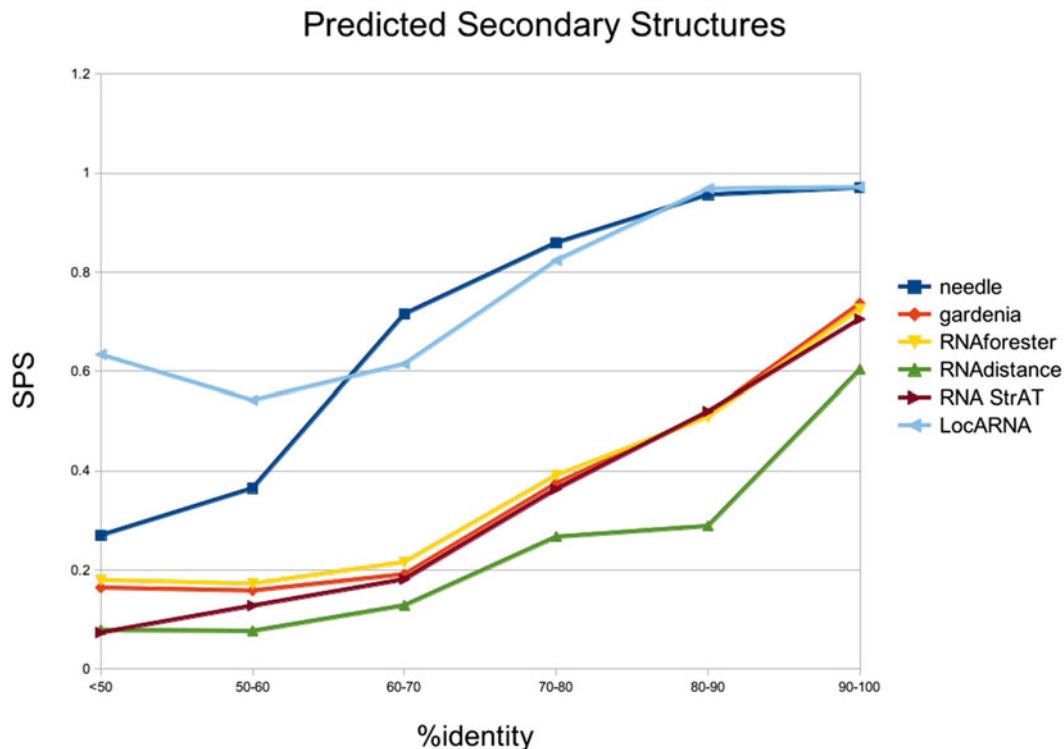


Fig. 3 Alignment performances of the six methods using predicted secondary structures obtained using RNAfold

For the more frequent cases where structural annotations are not available, LocARNA is the tool of choice, and sequence alignment can be also sufficient depending on the user needs.

3.2 Finding Sequence Identity with Needle

The command to perform a sequence alignment with *needle* is:

```
needle -asequence rna1 -bsequence rna2
```

where rna1 and rna2 are two plain text files containing the input sequences. The two input files can be in different formats; below is shown the widely used fasta format:

```
>S1
```

```
ACCAGGUGAAAUUCCUGGACCGACGGUUAAAGUCGG
```

The output is printed on screen but it can be saved into a file (in different formats) using the following command:

```
needle -asequence rna1 -bsequence rna2 -outfile out
```

3.3 Aligning Using LocARNA

The command to perform an alignment with LocARNA is:

```
locarna rna1 rna2
```

where rna1 and rna2 are two plain text files containing the input sequences (and, optionally, their secondary structure, or a set of structural constraints, as described later). The two input files must be formatted as shown below:

```
S1 ACCAGGUGAAAUUCCUGGACCGACGGUUAAAGUCCGG
```

Secondary structure annotation can be supplied in the standard dot-bracket notation as a new line. The input files must then be modified as shown below:

```
S1 ACCAGGUGAAAUUCCUGGACCGACGGUUAAAGUCCGG
#S .((((.....))).((((.....)).)))
```

The secondary structure is used by LocARNA to compute a base pair probability matrix satisfying structural constraints.

Among the parameters that can help in tuning the alignment, LocARNA allows the user to choose the substitution matrix with the switch ‘--ribosum-file=<path_to_the_matrix>’; default is the RIBOSUM85-60 matrix. Another useful parameter is ‘--clustal=<file>’ that save the output of the program using ClustalW [22] format in the file ‘my_out’. By default the output is printed on screen.

3.4 Aligning Using RNA STRAT

The command to run RNA StrAT is:

```
rnastrap rna1 rna2
```

where input plain text files rna1 and rna2 must have the format reported below:

```
>S1
ACCAGGUGAAAUUCCUGGACCGACGGUUAAAGUCCGG
.((((.....))).((((.....)).)))
```

No further parameters are available in the package.

4 Notes

1. LocARNA needs a Vienna installation to work. By using the ‘—enable-librna’ switch along with the ‘./configure’ LocARNA will search the Vienna installation in the default path. If your Vienna installation is not in the default path you should use the ‘—enable-librna PATH_TO_VIENNA’ switch.
For the sake of simplicity we suggest to install Vienna package by adding the Linux repository.
2. EMBOSS package in UBUNTU system is already included in the repository.
3. Table 1 shows the average computational time per sequence for each dataset and method. The computational time generally depends on the sequence length, which is different in the three

Table 1
Mean running time (in seconds) per sequence

	bralibase	RNAspa	RNAstrand
needle	0.01	0.01	0.01
gardenia	0.01	0.01	0.02
RNA StrAT	0.02	0.32	0.46
LocARNA	0.02	0.08	0.02
RNAdistance	0.01	0.01	0.01
RNAforester	0.03	0.73	0.81

datasets (bralibase: average length 76 nucleotides; RNAspa: 145 nucleotides; RNAstrand: 119 nucleotides). Other input RNA characteristics can also influence the running time of some algorithms, for example their structural complexity (i.e., how many secondary sub-structures the RNA contains).

Acknowledgements

This work was supported by the EPIGEN flagship project and PRIN 2010 (prot. 20108XYHJS_006) to MHC.

References

1. Mercer TR, Gerhardt DJ, Dinger ME et al (2012) Targeted RNA sequencing reveals the deep complexity of the human transcriptome. *Nat Biotechnol* 30:99–104. doi:[10.1038/nbt.2024](https://doi.org/10.1038/nbt.2024)
2. Cabili MN, Trapnell C, Goff L et al (2011) Integrative annotation of human large intergenic noncoding RNAs reveals global properties and specific subclasses. *Genes Dev* 25:1915–1927. doi:[10.1101/gad.17446611](https://doi.org/10.1101/gad.17446611)
3. Baker M (2011) Long noncoding RNAs: the search for function. *Nat Methods* 8:379–383. doi:[10.1038/nmeth0511-379](https://doi.org/10.1038/nmeth0511-379)
4. Burge SW, Daub J, Eberhardt R et al (2013) Rfam 11.0: 10 years of RNA families. *Nucleic Acids Res* 41:D226–D232. doi:[10.1093/nar/gks1005](https://doi.org/10.1093/nar/gks1005)
5. Gardner PP, Wilm A, Washietl S (2005) A benchmark of multiple sequence alignment programs upon structural RNAs. *Nucleic Acids Res* 33:2433–2439. doi:[10.1093/nar/gki541](https://doi.org/10.1093/nar/gki541)
6. Wan Y, Kertesz M, Spitale RC et al (2011) Understanding the transcriptome through RNA structure. *Nat Rev Genet* 12:641–655. doi:[10.1038/nrg3049](https://doi.org/10.1038/nrg3049)
7. Sankoff D (1985) Simultaneous solution of the RNA folding, alignment and protosequence problems. *SIAM J Appl Math* 45:810–825
8. Havgaard JH, Torarinsson E, Gorodkin J (2007) Fast pairwise structural RNA alignments by pruning of the dynamical programming matrix. *PLoS Comput Biol* 3:1896–1908. doi:[10.1371/journal.pcbi.0030193](https://doi.org/10.1371/journal.pcbi.0030193)
9. Harmanci AO, Sharma G, Mathews DH (2007) Efficient pairwise RNA structure prediction using probabilistic alignment constraints in Dynalign. *BMC Bioinformatics* 8:130. doi:[10.1186/1471-2105-8-130](https://doi.org/10.1186/1471-2105-8-130)

10. Will S, Reiche K, Hofacker IL et al (2007) Inferring noncoding RNA families and classes by means of genome-scale structure-based clustering. *PLoS Comput Biol* 3:e65. doi:[10.1371/journal.pcbi.0030065](https://doi.org/10.1371/journal.pcbi.0030065)
11. Dowell RD, Eddy SR (2006) Efficient pairwise RNA structure prediction and alignment using sequence alignment constraints. *BMC Bioinformatics* 7:400. doi:[10.1186/1471-2105-7-400](https://doi.org/10.1186/1471-2105-7-400)
12. Taneda A (2010) Multi-objective pairwise RNA sequence alignment. *Bioinformatics* 26:2383–2390. doi:[10.1093/bioinformatics/btq439](https://doi.org/10.1093/bioinformatics/btq439)
13. Notredame C, Higgins DG (1996) SAGA: sequence alignment by genetic algorithm. *Nucleic Acids Res* 24:1515–1524
14. Blin G, Denise A, Dulucq S et al (2007) Alignments of RNA structures. *IEEE ACM Trans Comput Biol Bioinformatics* 7:309–322. doi:[10.1109/TCBB.2008.28](https://doi.org/10.1109/TCBB.2008.28)
15. Guignon V, Chauve C, Hamel S (2005) An edit distance between RNA stem-loops. In: Consens MP, Navarro G (eds) SPIRE. Springer, Heidelberg, pp 335–347
16. Lorenz R, Bernhart SH, Höner Zu Siederdissen C et al (2011) ViennaRNA Package 2.0. *Algorithms Mol Biol* 6:26. doi:[10.1186/1748-7188-6-26](https://doi.org/10.1186/1748-7188-6-26)
17. Rice P, Longden I, Bleasby A (2000) EMBOSS: the European molecular biology open software suite. *Trends Genet* 16: 276–277
18. Andronescu M, Bereg V, Hoos HH, Condon A (2008) RNA STRAND: the RNA secondary structure and statistical analysis database. *BMC Bioinformatics* 9:340. doi:[10.1186/1471-2105-9-340](https://doi.org/10.1186/1471-2105-9-340)
19. Horesh Y, Doniger T, Michaeli S, Unger R (2007) RNAspa: a shortest path approach for comparative prediction of the secondary structure of ncRNA molecules. *BMC Bioinformatics* 8: 366. doi:[10.1186/1471-2105-8-366](https://doi.org/10.1186/1471-2105-8-366)
20. Widmann J, Stombaugh J, McDonald D et al (2012) RNASTAR: an RNA STructural Alignment Repository that provides insight into the evolution of natural and artificial RNAs. *RNA* 18:1319–1327. doi:[10.1261/rna.032052.111](https://doi.org/10.1261/rna.032052.111)
21. Berman HM, Kleywegt GJ, Nakamura H, Markley JL (2013) The future of the protein data bank. *Biopolymers* 99:218–222. doi:[10.1002/bip.22132](https://doi.org/10.1002/bip.22132)
22. Larkin MA, Blackshields G, Brown NP et al (2007) Clustal W and Clustal X version 2.0. *Bioinformatics* 23:2947–2948. doi:[10.1093/bioinformatics/btm404](https://doi.org/10.1093/bioinformatics/btm404)

Chapter 4

De Novo Secondary Structure Motif Discovery Using RNAProfile

Federico Zambelli and Giulio Pavesi

Abstract

RNA secondary structure plays critical roles in several biological processes. For example, many *trans*-acting noncoding RNA genes and *cis*-acting RNA regulatory elements present functional motifs, conserved both in structure and sequence, that can be hardly detected by primary sequence analysis alone. We describe here how conserved secondary structure motifs shared by functionally related RNA sequences can be detected through the software tool RNAProfile. RNAProfile takes as input a set of unaligned RNA sequences expected to share a common motif, and outputs the regions that are most conserved throughout the sequences, according to a similarity measure that takes into account both the sequence of the regions and the secondary structure they can form according to base-pairing and thermodynamic rules.

The method is split into two parts. First, it identifies candidate regions within the input sequences, and associates with each region a locally optimal secondary structure. Then, it compares candidate regions to one another, both at sequence and structure level, and builds motifs exploring the search space through a greedy heuristic. We provide a detailed guide to the different parameters that can be employed, and usage examples showing the different software capabilities.

Key words RNAProfile, RNA untranslated sequences, RNA secondary structure, UTR, Modtools, Posttranscriptional regulation

1 Introduction

RNA molecules play a large number of different roles in eukaryotic cells, other from the transfer of coding information performed by mRNAs and tRNAs. Examples are the posttranscriptional regulatory activity of miRNAs, the catalytic activity of ribozymes, and several others that have been identified only recently, like the ability to influence epigenetic states [1, 2]. Such great versatility is mainly due to the fact that, unlike DNA, RNA is a single stranded nucleic acid, and has the ability to fold into a three-dimensional structure through the formation of intramolecular base pairings. RNA structure can therefore act both in *cis* and in *trans*. While functional elements that depend from RNA structure cannot be identified by considering sequence information alone, in several

cases they can be however described by considering RNA secondary structure. Examples of these motifs are the secondary structure signals present in the untranslated regions (UTRs) of mRNA, responsible for posttranscriptional regulation of the gene expression. Several other classes of RNAs, such as rRNAs, tRNAs, microRNAs, and other noncoding small RNAs, as well as viral mRNAs, contain similar motifs with regulatory activity.

Thus, in the last few years, several bioinformatic algorithms and tools have been introduced to predict and compare RNA secondary structures (e.g., [3–6]). Also, several databases have been established and maintained to organize and exploit the wealth of information available on different class of RNAs and their associated secondary structures (e.g., [7–9]).

The problem of predicting shared structural features of RNA molecules, likely to correspond to functional elements, can be split into two separate problems. First of all, reliable models of the (secondary) structure of the RNAs should be built. Then, a set of RNAs sharing the same function should then present similarity at the sequence and most important, at the structure level, at least in some regions.

The prediction of the structure of a RNA sequence is usually based on the minimization of a given potential energy function (e.g., [10, 11]). If the prediction is limited to secondary structure, than the optimal secondary structure associated with the global minimum of the energy function can be obtained in feasible time. On the other hand, RNA folding is a dynamic process, that usually involves many intermediate states corresponding to local minima of energy. As a consequence, the actual *in vivo* structure of RNA molecules may not correspond to the most thermodynamically stable one. Thus, while single sequence energy-based prediction algorithms can exhaustively explore the space of possible secondary structure conformations, even if a very reliable energy function is employed the structure corresponding to a global minimum is not guaranteed to match the actual one.

On the other hand, the prediction of RNA secondary structure is more reliable when applied to predicting a “consensus structure” for a set of sequences sharing a similar structure [12]. However, in a set of functionally related RNAs, relevant structural similarity usually covers only a fraction of the sequences themselves, often just a single hairpin, whereas the rest of the overall structure exhibits more variability. Motifs found in mRNA-untranslated regions (UTRs) are typical instances of this behavior [13]. Thus, the problem of finding these functional motifs can be recast as motif finding aimed at discovering similar elements in sequences with related function, where similarity cannot be limited to primary sequence alone, but has to include also structural features (Fig. 1).

Herein we describe RNAProfile [3], a bioinformatic tool for finding conserved secondary structure motifs shared by a set of

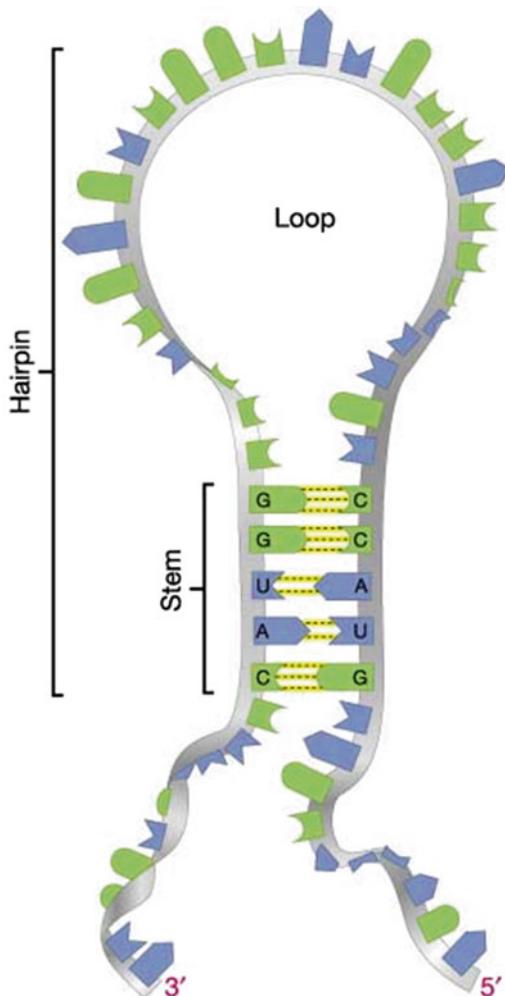


Fig. 1 A secondary structure *hairpin*, composed by a *stem* made by intramolecular base pairings and a *loop* with unpaired bases. Bases composing the loop have usually a functional role, and thus are generally more conserved in functional motifs than the paired ones

functionally related RNA sequences. RNAProfile bypasses the need of predicting the overall secondary structure of the sequences studied, and can identify similar regions within the sequences themselves, where similarity is measured by considering both sequence similarity and the optimal secondary structure associated with the regions.

2 Materials

RNAProfile is available at <http://www.beaconlab.it/modtools/> (current version is 2.2). It is possible to download either the source code, or precompiled binary files for Unix-like operating systems

(Unix, Linux, BSD, Mac OS X) and for Windows. In order to compile the source code, the C compiler “*gcc*” and the “*make*” utility should be available. For secondary structure prediction, RNAProfile includes functions for free energy calculation and secondary structure prediction taken from the Vienna RNA package [11]. The software is memory and time efficient, and can be run on any current PC configuration.

3 Methods

3.1 The Algorithm

We present here a summary of the main steps performed by the algorithm, while a more detailed description can be found in [3] and relative Supplementary Materials.

3.1.1 Candidate Regions Selection

Given a set of RNA sequences expected to share a structural motif, the first step is to identify a set of candidate regions from each sequence, containing potential secondary structure motifs. Since RNA secondary structure can be decomposed into hairpins, the only requested parameter is the number of hairpins expected to form the motif, with a single hairpin as default. No other constraints like size and number of features like loops, stacks, and connecting elements or any threshold for the folding energy are required. The prediction of the secondary structure of the input sequences is performed only locally. That is, given as input the number h of hairpins that have to be contained in the secondary structure of the motif, the algorithm selects from each input sequence the regions whose predicted *optimal* secondary structure contains exactly h hairpins. If this parameter is also not available, the analysis can be simply iterated starting from a single hairpin, and increasing the number h at each run. In the absence of further constraints, the algorithm examines all the possible substrings of each input sequence. The general idea is that a structural motif should correspond to a local free energy optimum for the region forming it, and its formation thus should depend solely on local interactions among the nucleotides of the region and not on the presence of other structures elsewhere in the sequence.

Moreover, searching for a fixed number of hairpins within each region of a RNA sequence of length n allows the reduction of computational complexity from the exponential time required to enumerate all the potential secondary structures that a RNA sequence can form [14] to a polynomial one. Notice also that the regions length is not predetermined, since the algorithm processes every possible substring of suitable size of the sequence in order to check if it contains the required number of hairpins. Finally, this approach takes also advantage from the fact that folding parameters used by energy-based RNA secondary structure methods are usually more reliable when applied to small regions of limited size (10–50 nt).

The algorithm does not need a priori free energy thresholds, so that each region with an associated optimal secondary structure with free energy lower than zero (lower than the unfolded state) is accepted as a candidate motif instance. Candidate regions from the same sequence may overlap one another.

3.1.2 Building Motifs

After being selected from the input sequences, together with their secondary structure, candidate regions are compared to one another, in order to build groups of “most similar” regions likely to form a conserved motif. These groups will contain exactly one region from each input sequence. Comparisons are made by computing pairwise alignment of the regions, with a scoring function that takes into account at the same time similarity at sequence and structure level. Given k input sequences of size n , the selection step returns $O(n)$ candidate regions per sequence, with $O(n^k)$ possible candidate motifs that can be built by selecting one region per sequence. Thus, exhaustive enumeration of all possible region combinations would be computationally unfeasible, and a greedy heuristic has been introduced in order to explore the solution space.

Briefly, the heuristic works as follows. Given a set of k sequences $S = \{S_1, S_2, \dots, S_k\}$ and their relative sets of associated regions $R = \{R_1, R_2, \dots, R_k\}$, RNAProfile first computes all the pairwise alignments between the regions from R_1 with those from R_2 . All the alignments are scored and ranked, and the p highest scoring alignments are kept. Each of the alignments is described with an alignment *profile*, that summarizes the sequence-structure similarity of the two aligned regions. At the second step, regions from R_3 are then aligned with the p profiles that were built at the previous step. The resulting alignments, that now contain three regions, are again scored, ranked, and the best p alignments are kept. This step is iterated until the last set of regions R_k has been reached. The alignments built at the last step will thus contain exactly one region per input sequence, and will correspond to the motifs output by the algorithm. Also, a *fitness* value is associated with each region included in an alignment, estimating how well the region fits the motif profile defined by the alignment itself. A positive fitness value is an indicator that the region is likely to represent an instance of the motif described by the profile, vice versa for negative fitness values. The rationale is that, since there are no guarantees that all the sequences of S will share an instance of the motif, a motif profile could include regions that do not represent instance of the motif, and hence have little similarity to the others building the motif.

When the motif is expected to appear only in a small subset (less than half) of the input sequences, the algorithm can be run employing a different method for the construction of motif profiles. For each i from 0 to $k - 1$ the selected regions R_i of a sequence S_i are aligned to all the regions R_j of the other sequences with $j > i$, and again the p highest scoring profiles are kept. Thus, instead of

saving the best alignments of regions from two sequences, the algorithm saves the highest scoring pairwise alignments obtained by comparing all possible pairs of candidate regions, with the sole condition that aligned regions come from different sequences. In the second iteration each of the highest scoring profiles is aligned with regions from the sequences that have not been used to build the profile itself, and so on until each profile contains a minimum number q of regions that can be set as input parameter. However, it is possible to output the highest scoring profiles obtained at each iteration, in order to inspect them: this strategy is suitable when the motifs appear in a few q of the k sequences, with $q < k/2$. The advantage of this method is that it is more likely to find motifs shared by a small subset of the input sequences. The main drawback is that it is prone to fall in local optima, if some input sequences are highly similar to one another, and that the computational complexity rises from $O(kn^2)$ to $O(k^2n^2)$ where again k is the number of input sequences and n is their length.

3.2 Running RNAPProfile

The command for using RNAPProfile with default settings is:

```
> ./rnaprofile -f<filename>
```

where <filename> is the input file.

3.2.1 Input

Input files can be of two types. The first and most common type is a *multifasta* file containing the RNA sequences to be analyzed. A careful selection of the sequences is important and may have a strong impact on the output. When possible, it is useful to avoid including long sequences (i.e., full mRNAs). Rather, it is advisable to include only the portion that is more likely to contain a functional motif. For example, when motifs appear within mRNA untranslated regions, it can be a good strategy to omit the coding portion of the transcripts, in order to exclude the bias induced by the higher sequence conservation usually found in coding sequences (*see Note*).

The second kind of input file accepted by RNAPProfile contains preprocessed sequences or regions, for which a secondary structure is already available. In this case, the secondary structure prediction and region selection steps of the algorithm are bypassed. In the latter case, the file format is as follows, with for each input sequence the sequence itself in FASTA format, followed by the secondary structure associated with its regions in bracket notation.

```
>Sequence1
CAGTCAGTACGTCTGACAGTCAGTACATGCTCGATGGTACGTATGCATCGTG
CATCAGTCTGAGTCAGTACTGACGTAGTCAGTCTGACTGACGTATGCAGTCTGA
!Sequence1
GTTCGTCCTCAGTGCAGGGCAAC
(((.((((.....)))))))
```

```

AACTTCAGCTACAGTGTAGCTAAGTT
((((((.((((((.....)))))))))))
CCACAGGCTCAGTGTGGTCTTGG
(((.((((((.....))))))))))
GCCTTCTGCACCAGTGTGTGTAAAGGC
((((((.((((((.....)))))))))))
GCCTTCTGCCAGTGTGTGTAAAGGC
((((((.((((((.....)))))))))))
TAATTGCAAACGCAGTGCGTTCAATTG
((((((.((((((.....)))))))))))
>Sequence2
CTACTGACAGTCAGTCATGCGTACAGTGTCACTGCAGTCAGTACCGTACGTA
CATGACGTCATGCATGCATGCAGTCAGTCATGCAGTCATGCATGCAGTCAGTCAG
!Sequence2
CCACAGGCTCAGTGTGGTCTTGG
(((.((((((.....))))))))
GCCTTCTGCACCAGTGTGTGTAAAGGC
((((((.((((((.....)))))))))))
GCCTTCTGCCAGTGTGTGTAAAGGC
((((((.((((((.....)))))))))))
TAATTGCAAACGCAGTGCGTTCAATTG
((((((.((((((.....)))))))))))

```

The regions of sequence *seqname* (defined in the FASTA header) must be introduced with a line beginning with ! (exclamation mark) followed by the same *seqname* as in the above example. The order in which regions appear in the file is unimportant and it is possible to put first all the sequences and then all their respective regions. If for one or more input sequences no regions are provided, RNAProfile will apply the secondary structure and region selection steps only on those sequences.

3.2.2 Parameters

Default settings have been determined by running several tests, with the aim of speeding up the computation obtaining at the same time reliable results. All input parameters can anyway be modified and fine tuned by users as follows.

-*A* <*int*>: run the program in “all versus all” mode, useful when only a subset of the input sequences is expected to share a motif as discussed in Subheading 3.1.2, or when the default run did not yield significant results. The algorithm stops when profiles containing <*int*> regions have been generated. The best profiles for each iteration are anyway output.

-*H* <*int*>: sets the number of hairpins that the secondary structure associated to each candidate region must contain. Default is a single hairpin. A good strategy can be to run RNAProfile on the same set of sequences increasing by one the value of *H* at each run, and stopping when the score for the best profile found given *n* hairpins is smaller than the one for the best profile with *n*–1 hairpins.

-v: verbose mode.

-r yes: process the sequences in random order. The default is to process the sequences in the same order with which they appear in the input file. This option is incompatible with **-A**. Different permutations of the input order are likely to produce different results, but, on the other hand, obtaining the same highest scoring motifs on different permutations is an indicator of the convergence of the method towards a set of highly conserved motifs.

-r <seed>: use *<seed>* (integer number) as seed for the random selection of sequences. Runs with the same seed will have identical output.

-P <int>: number of profiles to be kept at each step. The default is 100. Increasing this number allows for a more thorough exploration of the solution space, at the expense of computational time.

-l <int>: minimum length of the candidate regions. Default is 20 for single hairpin motifs and 10 more bases are added for each additional hairpin. It should be increased in case of complex structures and reduced when looking for small hairpins.

-L <int>: maximum length for candidate regions. Default is 40 for one hairpin and 20 more bases are added for each additional hairpin.

-o <filename>: the program performs only the first step, that is, secondary structure prediction and the identification of candidate regions, which are saved in file *<filename>* together with the predicted secondary structure.

-s <int>: do not compare regions differing in length for more than *<int>*. This can speed up the program by avoiding the alignment of regions of largely different size, and thus very unlikely to produce good results. Default is 10.

-i <int>: run the program consecutively for *<int>* times. To use in conjunction with **-r yes** in order to process the sequences in a different order at each run. Once again, finding the same highest scoring motif(s) in all or most of the runs is an indicator of the presence of a highly conserved motif in the sequences.

-e <float>: set an energy threshold for the secondary structure associated to candidate regions. Default is 0, and *<float>* should be lower than 0.

-p <int>: at each step, no more than *<int>* of the best profiles can be generated by using the same starting profile. Used to avoid premature convergence. Default is 10.

3.2.3 Output

Results are output into a file, whose name is shown on the screen at the end of the run. A result file has the following format:

```

ALIGNMENT RESULTS
Input file: mouse+human.ferritin.fna
Number of profiles saved at each step: 100 Max number of
profiles originating from the same profile (or region): 10
Region minimum length: 20 Region maximum length: 40
Energy threshold: 0
Max difference in length between regions: 10
Random alignment: yes (Seed used: 1065538283)

Best profiles:
Profile 1. Score: 2.66
(profile data here)

>gi|33859501|ref|NM_009653.1| Mus musculus aminolev-
ulenic acid synthase 2, mRNA
gGTTcGTCTcagtgcAGGGCAACa
.(((.((((.....))))))). (E: -7.2 Fitness: 4.6)
>gi|6753913|ref|NM_010240.1| Mus musculus ferritin
light chain 1 (Ft11), mRNA
cTTGcTTCAAcagtgtTTGAACGGa
.(((.((((.....))))))). (E: -1.8 Fitness: 9.0)
>gi|6753911|ref|NM_010239.1| Mus musculus ferritin
heavy chain (Fth), mRNA
cCTGcTTCAAcagtgcTTGAACGGa
.(((.((((.....))))))). (E: -4.4 Fitness: 8.1)
>gi|20149497|ref|NM_000146.2| Homo sapiens ferritin,
light polypeptide (FTL), mRNA
cTTGcTTCAAcagtgtTTGGACGGa
.(((.((((.....))))))). (E: -1.3 Fitness: 9.8)
>gi|4557298|ref|NM_000032.1| Homo sapiens aminolevu-
linate, delta-, synthase 2
cGTTcGTCTcagtgcAGGGCAACa
.(((.((((.....))))))). (E: -7.5 Fitness: 6.3)
>gi|507251|gb|L20941.1| HUMFERRITH Human ferritin
heavy chain mRNA, complete cds
cCTGcTTCAAcagtgcTTGGACGGa
.(((.((((.....))))))). (E: -3.9 Fitness: 8.8)
>gi|806340:c862-1 H.sapiens (24) Ferritin H pseudogene
aATTCTTatttGAAGGAATg
.((((((.(((....))))))). (E: -4.5 Fitness: -3.6)
>gi|182512|gb|J04755.1| HUMFERHX Human ferritin H
processed pseudogene, complete cds
ctAGTCATGCCatGATGACTGca
..((((((.(((....))))))). (E: -7.5 Fitness: -39.7)
>gi|806342|emb|X80336.1| HS5FERHPE H.sapiens (5)
Ferritin H pseudogene
TTCTTCaCCaaTCtcatGAGGaGAGGGa
((((((.((.((....))))))). (E: -5.8 Fitness: -321.6)

```

The input parameters are reported at the beginning of the file. Then, the highest scoring profiles are listed ranked by score. The output for each profile includes the frequency of each nucleotide in each column of the alignment, together with the list of regions used to build it, their secondary structure and folding energy (E). As described in Subheading 3.1.2, the *fitness* value can be used to discriminate regions that are true instances of the motif described by the profile (positive fitness value) from those less likely. The more negative the fitness, the more unlikely it is for the correspondent region to be a true instance of the associated motif profile.

3.3 Usage Examples

In these examples the algorithm has been run iteratively with default parameters, unless otherwise specified, starting with $h=1$ (h being the number of hairpins in candidate regions) and increasing h by 1 until the best profile found with h hairpins had a score lower than the best profile reported using $h-1$ hairpins.

3.3.1 Iron Responsive Element

The iron responsive element (IRE) is a well-known mRNA structural element (e.g., [15–17]) that has been also associated with neurodegenerative diseases (e.g., [18]). It is composed by a conserved hairpin structure present in the UTRs of transcripts coding for proteins involved in cellular iron metabolism. IREs usually present an unpaired nucleotide (cytosine) at the 5' of the stem, or a cytosine nucleotide and two additional bases at the 5' opposing one free 3' nucleotide. The iron responsive element is then a good benchmark for testing the functionality of RNAProfile. A dataset with the full mRNA sequences of human and mouse ferritin (light and heavy chain) and aminolevulinate synthase 2 was prepared. To add experimental noise, sequences from three human ferritin pseudogenes, hence not likely to contain an IRE motif, were also included in the dataset. As it can be seen in Fig. 2 the algorithm is able to identify correctly the IREs as building the highest scoring motif. Candidate motif regions from pseudogenes have instead a negative *fitness* value, strongly hinting that they are not actual instances of the IRE motif described by the profile.

3.3.2 Atypical IREs

Not all the IREs fall in the two types described in the previous example. A new dataset with sequences including atypical IREs found in the 5'-UTR of mRNAs from fruitfly, bullfrog, starfish and crayfish was prepared. Eight 5'-UTRs from plant ferritin mRNAs, lacking the IRE, were also added to the dataset to test the “all versus all” functionality of the algorithm. In fact, only one third of the sequences of this dataset actually contain an IRE. The algorithm was thus run with the $-A$ option active, and looking for motifs containing up to six regions (appearing in at most half of the sequences). As shown in Fig. 3, four regions containing four IRE instances form the highest scoring motif. The bottom helix has different length in the four instances, while the starfish and crayfish IREs share a base pair missing at the bottom of the topmost helix.

Iron Responsive Element

```
>NM_009653.1| Mus musculus aminolevulinic acid synthase 2
gGTTcGTCCTcagtgcAGGGCAACA
.(((.((((((.....)))))))..
(E: -7.2 Fitness: 5.7)

>NM_010240.1| Mus musculus ferritin light chain 1 (Ft11)
cTTGcTTCACcagtgtTTGAACGGG
.(((.((((((.....)))))))..
(E: -1.8 Fitness: 8.7)

>NM_010239.1| Mus musculus ferritin heavy chain (Fth)
cCTGcTTCACcagtgcTTGAACGGG
.(((.((((((.....)))))))..
(E: -4.4 Fitness: 9.5)

>NM_000146.2| Homo sapiens ferritin, light polypeptide (FTL)
cTTGcTTCACcagtgtTTGGACGGG
.(((.((((((.....)))))))..
(E: -1.3 Fitness: 8.5)

>NM_000032.1| Homo sapiens aminolevulinate, delta-, synthase 2
cGTTcGTCCTcagtgcAGGGCAACA
.(((.((((((.....)))))))..
(E: -7.5 Fitness: 7.3)

>L20941.1| HUMFERRITH Human ferritin heavy chain mRNA
cCTGcTTCACcagtgcTTGGACGGG
.(((.((((((.....)))))))..
(E: -3.9 Fitness: 9.4)

>gi|806340:c862-1 H.sapiens (24) Ferritin H pseudogene
GTCAcTCAAttctTTGATGGC
(((.((((((.....)))))))..
(E: -4.3 Fitness: -10.3)

>J04755.1| HUMFERHX Human ferritin H processed pseudogene
GAGacATTCTTcaccAAGAGTcCTC
(((.((((((.....)))))))..
(E: -3.4 Fitness: -25.1)

>gi|806342|emb|x80336.1|HS5FERHPE H.sapiens (5) Ferritin H pseudogene
cTGAAatctTCCTtccttcGGGAcTTC
.(((.((((((.....)))))))..
(E: -4.2 Fitness: -13.8)
```

Fig. 2 IRE motif occurrences reported by RNAProfile, with respective energy and fitness values. Notice the negative fitness values for the instances reported in pseudogenes

The crayfish element does not present the unpaired cytosine usually considered the distinctive signature of IREs. However, all these elements have been experimentally validated as functional IREs in [19].

3.3.3 Translation Control Element

The Nanos protein determines the correct anterior/posterior patterning in fruitfly embryos [20]. The translation of its mRNA is repressed in the bulk cytoplasm and activated only in the posterior domain. Translational control is dependent on a RNA secondary structure feature called *translation control element* (TCE) found in the 3'-UTR of Nanos mRNA. This structure is Y shaped and is bound by the protein Smaug leading to translational repression [21]. For this test RNAProfile was run on the 3'-UTR sequences of Nanos mRNA homologs from *D. melanogaster*, *D. virilis* and *D. simulans*. The score of the highest scoring profile remained stable when searching for motifs with one or two hairpins and the best profile found with one hairpin was contained within the best profile found with two hairpins. The latter captures the two hairpins forming the Y shape as seen in Fig. 4.

Best Four Regions

```
>gi|3559829|emb|Y15629.1|DMFER1 Drosophila melanogaster mRNA for ferritin
CCTTcTGCGCcagtgtGTGTAAGG
((((((.....))))))).) (E: -6.200 Fitness: 15.254)

>gi|2183236|gb|AF001984.1| Asterias forbesii ferritin mRNA, complete cds
GTTTGTgcgTTCGcagtgtCGGAacCAAGC
((((((.....))))..))).) (E: -3.800 Fitness: -1.267)

>gi|213691|gb|M12120.1|RANRCFA Bull frog ferritin mRNA, complete cds
cTTGcTTCAAcagtgtTTGAACGGa
.((((((.....))))))).) (E: -1.800 Fitness: 13.610)

>gi|1070378|emb|X90566.1|PLRNFERR P.leniusculus mRNA for ferritin
cGCTCgggTCGCcagtgtGTGAacGAGCt
.((((((.....))))..))).) (E: -8.500 Fitness: 5.847)
```

Fifth Region Added

```
>gi|20530724:1-100 Chlamydomonas reinhardtii pre-apoferritin (Fer1) mRNA,
cGTTTCGGggccCGGccAAGCt
.((((((.....))))..))).) (E: -6.200 Fitness: -68.032)
```

Sixth Region Added

```
>utr|BB120830|5OSA001337 5'UTR in Oryza sativa ferritin mRNA
gaCCGacTCCGGtgcggCCGGGcCGGgc
..(((...(((((.....))))..))).) (E: -11.600 Fitness: -36.524)
```

Fig. 3 Four IRE instances found in the “unconventional IRE” dataset building the highest scoring motif. Allowing the algorithm to reach the imposed limit of six regions per motif lowered the score of the motif itself by including two unlikely instances, as also shown by the low fitness of the two additional sequences

3'UTR in Drosophila nanos mRNA

```
>gi|157956:2472-2800 Drosophila melanogaster nanos (nos)
gaGCAGAGGCTctggcAGCTTTGCaGCGTtTATATAacatggaaaTATATATACGC
..((((((.....))))))).) (((((.(((((.....))))))).))).)

>gi|1184670:2500-2700 Drosophila virilis nanos (Dv nos)
ggAAGAACGCTctggcAGCTTTTaaGCGTtTATATAagaggtaTATATATGCGCgt
..((((((.....)))))).) (((((.(((((.....))))))).))).)

>gi|7716709:792-1073 Drosophila simulans strain sim1 nanos protein (nos) gene
gaGCAGAGGCTctggcAGCTTTGCaGCGTtTATATAacaagaaaTATATATACGCat
..((((((.....)))))).) (((((.(((((.....))))))).))).
```

Fig. 4 The Nanos TCE element identified by RNAProfile, composed of two hairpins building a Y-shaped secondary structure

3.4 Single Sequence RNA Secondary Structure Prediction

In some cases, it might be helpful to have a tool for secondary structure prediction available to work together with motif finding tools like RNAProfile. In fact, while functional RNA secondary structures may be different from the ones predicted according to minimal energy or other criteria, tools like mfold [22] can be constrained to predict structures that include fixed elements in some regions, like those identified by RNAProfile. In this way the predicted structure for the whole molecule might represent a more accurate representation of the possible *in vivo* secondary structure.

3.5 Secondary Structure Visualization Tools

RNAProfile output consists of a text file where the structure motifs are described using the conventional “dot-and-brackets” notation, where round brackets correspond to paired bases, and dots to the unpaired ones. While this method is practical, a graphical representation of the RNA motifs should be more suitable when producing figures for articles or oral presentations. Tools like PseudoViewer [23] and Jviz.Rna [24] can be very useful for this task, since they permit to draw a model of the secondary structure starting from the dot-bracket notation.

4 Note

As discussed in Subheading 3.2.1, a careful selection of the sequences to be analyzed is of the utmost importance for the correct identification of RNA motifs, for example by removing unnecessary portion of the sequences. Another point to be considered when preparing an input where most of the sequences are expected to contain a functional motif is to arrange them in phylogenetic order following a sort of decreasing similarity gradient. In fact, as described in Subheading 3.1.2, the greedy heuristic for the comparisons of regions proceeds by following the same order of the input sequences. Therefore, it may happen that processing very dissimilar sequences at the beginning could mask the presence of motifs that would be otherwise detected when the profiles are built more gradually. On the other hand, including highly similar sequences in the input should be avoided, since in this case sequence similarity will have the effect of “shadowing” similarity mainly due to secondary structure. In case of doubt, it might be advisable to use the `-r yes` option in conjunction with a suitable value for `-i`, in order to perform multiple tests on different permutations of the sequences. Highest scoring motifs appearing in all or most of the results are then the most likely candidates to correspond to actual functional motifs shared by the input sequences.

References

1. Sabin LR, Delás MJ, Hannon GJ (2013) Dogma derailed: the many influences of RNA on the genome. *Mol Cell* 49(5):783–794
2. Dieterich C, Stadler PF (2012) Computational biology of RNA interactions. *Wiley Interdiscip Rev RNA* 4(1):107–120
3. Pavesi G, Mauri G, Stefani M et al (2004) RNAProfile: an algorithm for finding conserved secondary structure motifs in unaligned RNA sequences. *Nucleic Acids Res* 32(10):3258–3269
4. Rabani M, Kertesz M, Segal E (2008) Computational prediction of RNA structural motifs involved in posttranscriptional regulatory processes. *Proc Natl Acad Sci U S A* 105(39):14885–14890
5. Hiller M, Pudimat R, Busch A et al (2006) Using RNA secondary structures to guide sequence motif finding towards single-stranded regions. *Nucleic Acids Res* 34(17):e117
6. Bafna V, Tang H, Zhang S (2006) Consensus folding of unaligned RNA sequences revisited. *J Comput Biol* 13(2):283–295
7. Mokrejs M, Vopálenský V, Kolenaty O et al (2006) IRESite: the database of experimentally verified IRES structures (www.iresite.org). *Nucleic Acids Res* 34(Database issue):D125–D130
8. Burge SW, Daub J, Eberhardt R et al (2013) Rfam 11.0: 10 years of RNA families. *Nucleic Acids Res* 41(Database issue):D226–D232
9. Grillo G, Turi A, Licciulli F et al (2010) UTRdb and UTRsite (RELEASE 2010): a collection of sequences and regulatory motifs of the untranslated regions of eukaryotic mRNAs. *Nucleic Acids Res* 38(Database issue):D75–D80
10. Reuter JS, Mathews DH (2010) RNAstructure: software for RNA secondary structure prediction and analysis. *BMC Bioinformatics* 11:129
11. Lorenz R, Bernhart SH, Höner Zu Siederdissen C et al (2011) ViennaRNA Package 2.0. *Algorithms Mol Biol* 6:26
12. Witwer C, Hofacker IL, Stadler PF (2004) Prediction of consensus RNA secondary structures including pseudoknots. *IEEE/ACM Trans Comput Biol Bioinformatics* 1(2):66–77
13. Mignone F, Gissi C, Liuni S et al (2002) Untranslated regions of mRNAs. *Genome Biol* 3(3):REVIEWS0004
14. Waterman MS (1995) Introduction to computational biology. Chapman & Hall, London
15. Hentze MW, Muckenthaler MU, Galy B et al (2010) Two to tango: regulation of mammalian iron metabolism. *Cell* 142(1):24–38
16. Tandara L, Salamunic I (2012) Iron metabolism: current facts and future directions. *Biochem Med* 22(3):311–328
17. Ma J, Haldar S, Khan MA et al (2012) Fe²⁺ binds iron responsive element-RNA, selectively changing protein-binding affinities and regulating mRNA repression and activation. *Proc Natl Acad Sci U S A* 109(22):8417–8422
18. Cahill CM, Lahiri DK, Huang X et al (2009) Amyloid precursor protein and alpha synuclein translation, implications for iron and inflammation in neurodegenerative diseases. *Biochim Biophys Acta* 1790(7):615–628
19. Huang TS, Melefors O, Lind MI et al (1999) An atypical iron-responsive element (IRE) within crayfish ferritin mRNA and an iron regulatory protein 1 (IRP1)-like protein from crayfish hepatopancreas. *Insect Biochem Mol Biol* 29(1):1–9
20. Lehmann R, Nüsslein-Volhard C (1991) The maternal gene nanos has a central role in posterior pattern formation of the *Drosophila* embryo. *Development* 112(3):679–691
21. Cruces S, Chatterjee S, Gavis ER (2000) Overlapping but distinct RNA elements control repression and activation of nanos translation. *Mol Cell* 5(3):457–467
22. Zuker M (2003) Mfold web server for nucleic acid folding and hybridization prediction. *Nucleic Acids Res* 31(13):3406–3415
23. Byun Y, Han K (2009) PseudoViewer3: generating planar drawings of large-scale RNA structures with pseudoknots. *Bioinformatics* 25(11):1435–1437
24. Wiese KC, Glen E, Vasudevan A (2005) JViz. Rna—a Java tool for RNA secondary structure visualization. *IEEE Trans Nanobioscience* 4(3):212–218

Chapter 5

Drawing and Editing the Secondary Structure(s) of RNA

Yann Ponty and Fabrice Leclerc

Abstract

Secondary structure diagrams are essential, in RNA biology, to communicate functional hypotheses and summarize structural data, and communicate them visually as drafts or finalized publication-ready figures. While many tools are currently available to automate the production of such diagrams, their capacities are usually partial, making it hard for a user to decide which to use in a given context. In this chapter, we guide the reader through the steps involved in the production of expressive publication-quality illustrations featuring the RNA secondary structure. We present major existing representations and layouts, and give precise instructions to produce them using available free software, including *jViz.RNA*, the *PseudoViewer*, *RILogo*, *R-chie*, *RNAplot*, *R2R*, and *VARNA*. We describe the file formats and structural descriptions accepted by popular RNA visualization tools. We also provide command lines and Python scripts to ease the user's access to advanced features. Finally, we discuss and illustrate alternative approaches to visualize the secondary structure in the presence of probing data, pseudoknots, RNA–RNA interactions, and comparative data.

Key words RNA visualization, Secondary structure, Graph drawing, Pseudoknots, Non-canonical motifs, Structure-informed multiple sequence alignments

1 Introduction

1.1 Context and Definitions

The secondary structure of RNA represents a discrete abstraction, or projection, of a three-dimensional structure restricted to (a subset of) its interacting positions. As such, it is naturally used within schematics depicting the general architecture of a given RNA conformation. It provides a context for various sequential information, including evolutionary conservation, accessibility to chemical/enzymatic probes, or predicted single-strandedness. In structural studies, it is also used as a scaffold to perform an interactive three-dimensional modeling by homology. Such schematics are finally used to illustrate functional scenario involving RNA, possibly interacting with another molecule or ligand, raising the need for the production of publication-quality diagrams.

Electronic Supplementary Material: The online version of this chapter (doi: 10.1007/978-1-4939-2291-8_5) contains supplementary material, which is available to authorized users

Formally, the **secondary structure** of an RNA is simply a set of base-pairs connecting some of the positions in an RNA sequence. Any base-pair will be denoted by a pair (i,j) , indicating that the nucleotide at position i in the sequence is paired with the nucleotide at position j . In its strictest definition, this set of base-pairs obeys additional constraints:

- **Only canonical interactions:** Any base-pair must be canonical (A–U, G–C, or G–U bases, interacting on Watson–Crick/Watson Crick edges in *cis* orientation);
- **Non-crossing base-pairs:** Any pair of base-pairs (i,j) and (i',j') should be free of crossing interactions ($i < i' < j < j'$ or $i' < i < j' < j$). This property forbids the representation of complex structural features like pseudoknots;
- **Single partner:** Any position may interact with at most a single distinct partner.

These restrictions lead to **tree-like objects** which can be drawn on the 2D plane, and are easy to analyze visually. They also ensure the computational tractability of the underlying algorithmic processing, while retaining the capacity to suggest the general architecture (helices, junctions...) adopted within a given conformation.

However, enforcing these restrictions leads to a loss of information which could deteriorate the quality of subsequent analyses. In such cases, a more detailed view of the base–base interaction network must be adopted. Some or all of the above constraints can be lifted, leading to the **extended secondary structure**, a representation which typically supports additional annotations, such as stacked bases, or interacting edges and orientation for base-pairs. Crossing helices, usually referred to as **pseudoknots**, using a topological analogy, are also supported by some tools, but their planar layout is usually challenging, and even intrinsically unfeasible for some RNA architectures.

1.2 Objectives of RNA Visualization

The secondary structure can be naturally represented as a graph whose vertices are individual nucleotides. In this representation, edges are expected to connect consecutive nucleotides in the sugar-phosphate backbone, or pairs of positions involved in a base-pair mediated by hydrogen bonds. Besides these formal requirements, additional properties have been identified as desirable by earlier work [25]:

Modularity Inspection of the drawing should suggest a decomposition of the structure into functional domains. For instance, helices should be easy to identify.

Robustness Structural similarity should be revealed by similar-looking representations. In particular, the introduction of few evolutionary events (insertions/deletions of single or paired nucleotides) should not lead to significant changes.

Clarity The resulting drawing should lend itself to an easy visual analysis. In particular, overlapping nucleotides and crossing

lines should be avoided, and a constant distance should separate backbone-consecutive nucleotides and base-pairs, respectively.

Realism Drawings should ideally constitute a projection of the three-dimensional structure. In this respect, relative distances between consecutive nucleotides and base-pairing partners should be proportional to that observed in three-dimensional structures.

Aesthetics Arguably the most subjective aspect of RNA visualization encompasses a large variety of criteria ranging from standard color schemes to a strict adherence to historical representations for classic families of RNAs (e.g., cloverleaf shape for transfer RNAs).

Unfortunately, satisfying combinations of these properties may be computationally intractable, or simply even impossible for intrinsic reasons. Consequently, many alternative representations and layout strategies have been proposed over the years, each arising as a tradeoff.

1.3 Existing Tools

Many tools are now available for visualizing the secondary structure of RNA. These tools differ on many aspects, depending on their intended application and functionalities. Among the distinguishing features of available tools, one denotes the presence of a graphical user interface, allowing for a convenient post-processing before resorting to a time-consuming general-purpose editor. Moreover, while certain formats may be converted into others in a lossless manner, most conversion may lose some format-specific data, and a support for a rich variety of input format is therefore desirable. The output format of a software is also of great importance, as generated figures typically require further post-processing before meeting the quality criteria expected by publishers. Therefore, vector formats such as Portable Document Format (PDF), or Scalable Vector Graphics (SVG) should always be preferred to bitmap formats. Table 1 summarizes the foremost tools available for visualizing the secondary structure. Which, of the available alternatives, should be preferred will typically depend on the intended application. We hope that this overview of existing tools and representations will assist the reader in his choice of the right visualization, and ultimately ease the production of more informative pictures.

1.4 Outline

In this chapter, we focus on the necessary steps involved in the production of publication-quality illustrations involving the RNA secondary structure. After introducing the main data sources and file formats, we describe a preferred workflow, and stress on the advantages of vector graphics manipulation. Then, we describe major proposed representations and layouts for the secondary structure. We also provide simple command lines to invoke main tools. We also describe how to address the specific visualization needs arising from exemplary use-cases: chemical probing experiments, RNA pseudoknots and interactions, and multiple-sequence alignments.

Table 1
Main features of selected tools offering a visualization of the RNA secondary structure

Name	Platform(s) [B威ML ^a]	Ease of use	Layouts	Interactivity	Pseudoknots	Ext. sec. str. ^b	Output ^c	References
jViZ.RNA	••••	++	Circular Linear Graph	+	+	•	EPS PNG	[29, 32]
PseudoViewer	•••••	++	Graph	++	◦	◦	EPS SVG GIF	[5]
RNAMovies	•••••	++	Graph	+	◦	◦	SVG PNG JPEG GIF	[17]
RILogo	•••••	+	Linear		◦	◦	SVG	[23]
R-chie	•••••	++	Linear	+	◦	◦	PDF PNG	[18]
RNAPlot	••••• ^d	+	Graph		◦	◦	PS SVG GML	[13]
RNAView	◦••••	+	Graph	+	•	◦	PS	[34]
RNAViZ	◦••••	+	Graph	++	+	◦	PDF ^e	[27]
R2R	◦••••	-	Graph	+	◦	◦	PDF SVG	[31]
S2S/Assemble	◦••••	+	Linear Graph	++	•	◦	SVG	[15, 16]
VARNA	•••••	++	Circular Linear Graph	++	•	◦	EPS SVG PNG JPEG GIF	[7]
xRNA	◦••••	+	Graph	++	+	◦	EPS	-

^a Indicates support (•) or lack of support (◦) for Browser-based, Windows, Mac and Linux executions

^b Indicates support (•) or lack of support (◦) for an extended secondary structure

^c Bold output formats indicate vector graphics, allowing for convenient post-processing

^d Only available from the Vienna RNA webserver, available at <http://rna.tbi.univie.ac.at/>

^e Export accessible through a print option, using a custom printer driver such as Adobe Distiller

2 Materials

Reflecting the diversity of functions and cellular processes involving non-coding RNAs, secondary structure data is organized within a maze of domain-specific databases, and encoded in a variety of file formats. We briefly mention these sources, and describe in further details the syntax of major file formats.

2.1 Data Sources

At the primary structure level, including sequences and alignments, the RFAM database [12] is the authoritative source of RNA data. Unfortunately this centralized authority is without a counterpart on the secondary structure level, leading to a wealth of specialized repositories focusing on RNAs of specific functions. The reason of this situation is mostly due to the young history of RNA computational biology, coupled with the fact that, up until recently, RNA structural data was relatively scarce compared to proteins. Consequently, databases have typically arisen from a variety of domain-specific efforts, which have not currently undergone standardization.

Of notable interest at the secondary structure level is the STRAND database [1], a collection of secondary structures found in, or inferred from, a variety of databases. However, the purpose of this database, which was initially assembled to train new energy models from existing sequence/structure data and benchmark computational prediction tools, conflicts with the goal of completeness, and secondary structure data must currently be sought within a variety of specialized databases. One should however note that recent initiatives, such as the RNA ontology consortium [21] or the RNA Central project [2], have led to propositions for more rational organizations of RNA structural data, and a solution to these issues may be found in the near future (Fig. 1).

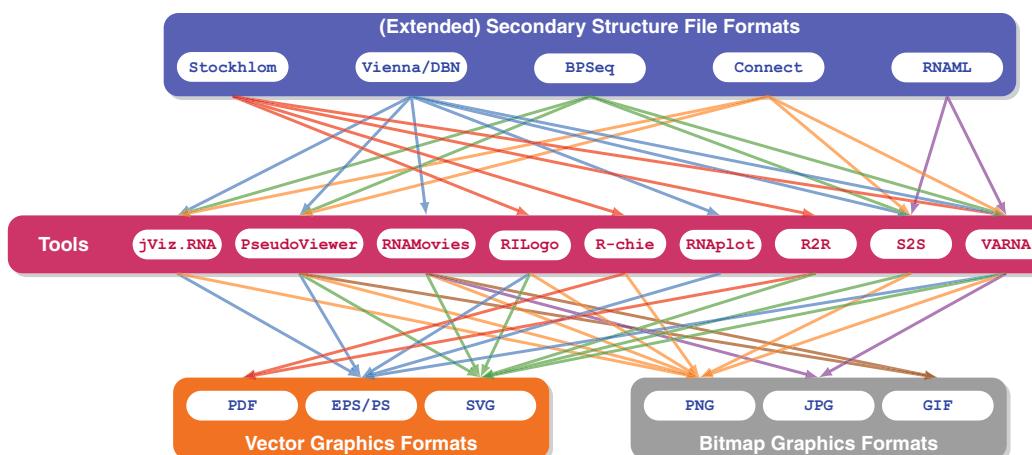


Fig. 1 Main input and output formats supported by the major visualization tools for the RNA secondary structure

```

# STOCKHOLM 1.0
#=GF ID      mir-22
#=GF AC      RF00653
...
o.latipes.1          CGUUG.CCUCACAGUCGUUCUCA.CUGGCU.AGCCUUUAUGUCCCACG..
Gasterosteus_aculeat.1  GGCUG.ACQUACAGCAGUUCUCA.CUGGCA.AGCCUUUAUGUCCUCAUCU
R.esox.1            AGCUGAGCACA...CAGUUCUCA.CUGGCA.GCCUUAAAGGUUCUGUAG
...
#=GC SS_cons        .<<<. <<..<<<<<<<<<..<<<..<<<<<.&<.....
#=GC RF             gGccg.acucaCagcaGuuCuuCa.cuGGCA.aGCuuuAuguccuuuaaa
o.latipes.1          CCCCACCGUAAAGCU.GC.CAGUUGAAGAGCUGUUGUG..UGUAACC
Gasterosteus_aculeat.1  ACCAGC..UAAAGCU.GC.CAGCUGAAGAACUGUUGUG..GUCGGCA
R.esox.1            ACAGGC..UAAACCU.GC.CAGCUGAAGAACUGCUCUG..GCCAGCU
...
#=GC SS_cons        ....>..>>>>>..>>..>>>>>>>>>..>>>>.
#=GC RF             acaaac..UaaaGCU.GC.CaGuuGaaGaaCugcuGug..gucggCu
//
```

Fig. 2 *Stockholm* formatted file fragment, excerpted from RFAM seed alignment for miRNA mir-22. RFAM ID: RF00653

2.2 RNA Secondary Structure File Formats

Reflecting the diversity of biological contexts and computational use-cases involving RNA secondary structure, many formats were proposed over time to support its description. Aside from the ubiquitous FASTA format, extended by the popular Vienna RNA software package [14] to include a dot-bracket encoding of the secondary structure, the STOCKHOLM format is used within RFAM to present a WUSS-formatted consensus secondary structure, possibly including pseudoknots. The BPSEQ and CONNECT formats represent an adjacency list, indicating the partner (if any) of each position.

Notably, the RNAML format [30] represents a unifying effort of the community to represent virtually any sort of RNA-related data. While this format is not widely used to represent the secondary structure because of its intrinsic verbosity, it is clearly the format of choice to represent and manipulate the extended secondary structure, including non-canonical base-pairs.

2.2.1 STOCKHOLM Format

STOCKHOLM format files are arguably the preferred representation for RNA multiple sequence alignments. As illustrated in Fig. 2, a STOCKHOLM file breaks the global alignment of a set of sequences into portions of bounded width. Unique identifying prefixes (Organism name/Accession ID) are associated with each portion. Additionally a set of mark-up lines, recognizable by their prefix `# =`, specify various additional information, including accession identifiers, experimental parameters, bibliographical references etc.

```
> Rat Alanine tRNA
GAGGAUUUAGCUUAUUAAAGCAGUUGAUUUGCAUUUAACAGAUGUAAGAUAUAGCUUACAGUCUUA
((((((...((((.....))))).((((.....))))...((((((...))))))))))))).
```

Fig. 3 Dot-parentheses (aka dot-bracket) notation for the minimal free-energy structure of the Rat Alanine tRNA, predicted using **RNAFold**

Two mark-up line types are especially relevant to RNA bioinformatics, the **# = GC RF** lines, which indicate a sequence consensus, and the **# = GC SS_cons**, which indicates the secondary structure consensus. The latter uses a parenthesis scheme to represent base-pairing positions using corresponding parentheses. Pseudoknots can also be represented using an additional alphabetical system (**A** matching with **a**, **B** with **b** ...).

2.2.2 Vienna RNA Dot-Parentheses (aka dot-bracket) and parenthesis and Pseudobase Notations

In this format, the RNA sequence is coupled with a dot-parenthesis string where matching pairs of opening and closing parentheses identify base-pairs. This expression is well parenthesized, meaning that any opening parenthesis can be unambiguously associated with a closing parenthesis, inducing a set of non-crossing base-pairs. For instance, in Fig. 3, the bases at first and ante-penultimate positions are base-paired.

Since matching pairs of parenthesis cannot unambiguously identify crossing base-pairs, pseudoknots cannot be represented strictly within this format. Therefore this format was extended by the PseudoBase to include support for multiple parentheses systems, allowing crossing interactions to be represented. In Fig. 4, two crossing helices, forming a H-type pseudoknot, are initiated by base-pairs at positions (1604,1623) and (1615,1630), respectively.

2.2.3 BPSeq Format

The BPSeq format is an alternative to the CT format introduced by the Comparative RNA Web site (CRW, hosted at the University of Texas Austin by the Robin Gutell Lab). It essentially consists in a simplified version of the CT format. As illustrated in Fig. 5 any BPSeq file starts with four self-explanatory lines identifying the data source and content, and is followed by the structure, specified as a sequence of space-separated triplets (x,y,z) , where: x

x Position;

y IUPAC code for base;

z Base-pairing partner position (0 if unpaired).

2.2.4 CONNECT (CT) Format

This format was introduced by Mfold [37], the historical tool for the ab-initio prediction of RNA secondary structure, and is still used to date by several prediction tools. After an initial header consisting of the sequence

	1590	1600	1610	1620	1630
#	123456789 123456789 123456789 123456789 123456				
\$	1590	AAAAAACUAUAGAGGGGGACUUAGCGCCCCCAAACCGUAACCCC=1636			
%	1590	:::(((((((:::((([])))):::)))::::::			

Fig. 4 Pseudobase notation for the Gag/pro ribosomal frameshift site of Bovine Leukemia Virus. *Source:* pseudobase, entry# PKB1

```

Filename: AM286415_b.bpseq
Organism: Yersinia enterocolitica subsp. enterocolitica 8081
Accession Numbers: AM286415
Citation and related information available at http://www.rna.ccbb.utexas.edu
1 U 0
...
117 U 0
118 U 236
119 G 235
120 C 234
121 C 233
122 U 232
123 G 231
124 G 230
...
230 C 124
231 C 123
232 A 122
233 G 121
234 G 120
235 C 119
236 A 118
...

```

Fig. 5 Fragment of a BPSEQ formatted 5s rRNA inferred using comparative modeling. *Source:* comparative RNA web site

length followed by a comment (*see* Fig. 6), each position is represented by six fields (*a, b, c, d, e, f*), each encoded in a fixed-width (eight characters) column:

- a* Position;
- b* IUPAC code for base;
- c* Position of previous base in the backbone (5'-3' order, 0 is used if first position);
- d* Position of next base in the backbone (5'-3' order, 0 is used if last position);
- e* Base-pairing partner position (0 if unpaired);

80		dG = -33.48 [Initially -35.60]			
1	U	0	2	80	1
2	G	1	3	79	2
3	G	2	4	78	3
4	G	3	5	77	4
5	A	4	6	76	5
6	U	5	7	0	6
7	G	6	8	75	7
...					
75	U	74	76	7	75
76	U	75	77	5	76
77	C	76	78	4	77
78	C	77	79	3	78
79	U	78	80	2	79
80	A	79	0	1	80

Fig. 6 CONNECT format for the Mfold 3.7 [37] predicted structure of the human let-7 pre-miRNA

```

HEADER      RNA          27-JUL-09      3IGI
TITLE       TERTIARY ARCHITECTURE OF THE OCEANOBACILLUS IHEYENSIS GROUP
TITLE       2 II INTRON
COMPND     MOL_ID: 1;
COMPND     2 MOLECULE: GROUP IIC INTRON;
COMPND     3 CHAIN: A;
...
ATOM    8009   P      U A 375      19.076  79.179 370.688  1.00 66.25      P
ATOM    8010   OP1    U A 375      18.815  77.862 371.313  1.00 83.22      O
ATOM    8011   OP2    U A 375      19.869  80.203 371.409  1.00 56.32      O
...
CONECT  8654 8520
CONECT  8655 8521
CONECT  8658 8531
MASTER    717    0   66    0    0   69    6 8656    2   123   33

```

Fig. 7 Fragment of a three-dimensional model for the Oceanobacillus iheyensis Group II intron (PDBID:3IGI)

f Position (duplicated).

This format can be gently abused to allow for the description of pseudoknots, although such motif cannot be predicted by Mfold.

2.2.5 PDB File Format

The PDB format is a comprehensive text-formatted representation of macromolecules used with the authoritative eponym repository of experimentally derived 3D models [3]. Originally introduced to represent protein structure, it has been enriched over the years to include fine details of the experimental protocol used for any structure derivation. However, it does not support detailed information regarding base-pairing position, and is mostly used by RNA software as a raw geometrical descriptor of a 3D model, as illustrated in Fig. 7.

```

<?xml version="1.0"?>
<!DOCTYPE rnaml SYSTEM "rnaml.dtd">
<rnaml version="1.0">
  <molecule id="xxx">
    <sequence>
      ...
      <seq-data>
        UGUGCCCGGC AUGGGUGCAG UCUAUAGGU...
      </seq-data>
      ...
    </sequence>
    <structure>
      <model id="yyy">
        <base> ... </base> ...
        <str-annotation>
          ...
          <base-pair>
            <base-id-5p><base-id><position>2</position></base-id></base-id-5p>
            <base-id-3p><base-id><position>260</position></base-id></base-id-3p>
            <edge-5p>+</edge-5p>
            <edge-3p>+</edge-3p>
            <bond-orientation>c</bond-orientation>
          </base-pair>
          <base-pair comment="">
            <base-id-5p><base-id><position>4</position></base-id></base-id-5p>
            <base-id-3p><base-id><position>259</position></base-id></base-id-3p>
            <edge-5p>S</edge-5p>
            <edge-3p>W</edge-3p>
            <bond-orientation>c</bond-orientation>
          </base-pair>
          ...
        </str-annotation>
      </model>
    </structure>
  </molecule>
  <interactions> ... </interactions>
</rnaml>

```

Fig. 8 RNAML formatted fragment of an all-atom 3D model for the Oceanobacillus iheyensis Group II intron (PDBID: 3IGI, also shown in Fig. 7), produced by the **RNAView** software. The base-pair XML sections are automatically annotated geometrically from the PDB file, and represent base-pairing positions. Such base-pairs may be non-canonical, i.e. they may involve non-standard pairs of nucleotides, interacting edges, or orientation. Here, positions 2 and 260 form a canonical base-pair (U–A, both Watson-Crick edges, *cis* orientation), while positions 4 and 259 form a non-canonical base-pair (U–U, Sugar/Watson-Crick edges, *cis* orientation)

2.2.6 RNAML Format

RNAML [30] is an XML format, introduced to address the dual need to unify data representations related to RNA, and to represent novel important features of its structure (e.g. non-canonical base-pairs and motifs). Although still challenged in its former goal by less structured, domain-specific, formats (arguably because of the intrinsic verbosity arising from its ambitious goals), RNAML has established itself as the format of choice for an enriched symbolic description of the tertiary structure, as illustrated in Fig. 8. Accordingly, it is currently supported by most automated methods capturing non-canonical interactions and motifs.

3 Methods

3.1 Producing Publication-Quality Pictures of RNA

The production of publication-quality illustrations can be greatly eased by the choice of adequate tools. Indeed, while one could in principle produce illustrative pictures of RNA by relying only on vector graphics software such as Inkscape or Adobe® Illustrator®, such a task would quickly be extremely time-consuming as soon as RNAs of moderate lengths are considered. In particular, this scenario would require spending a huge amount of time on trivial tasks, such as the layout of individual bases, or the reorientation of helices, and would not necessarily yield homogeneous output pictures.

For these reasons, current workflows attempt to keep using domain-specific tools as far as possible. As illustrated in Fig. 9, a variety of file formats (*see* Subheading 2.2) can be used to represent the secondary structure, possibly resulting from an automated annotation of a 3D all-atom model. Such files are then used, possibly in

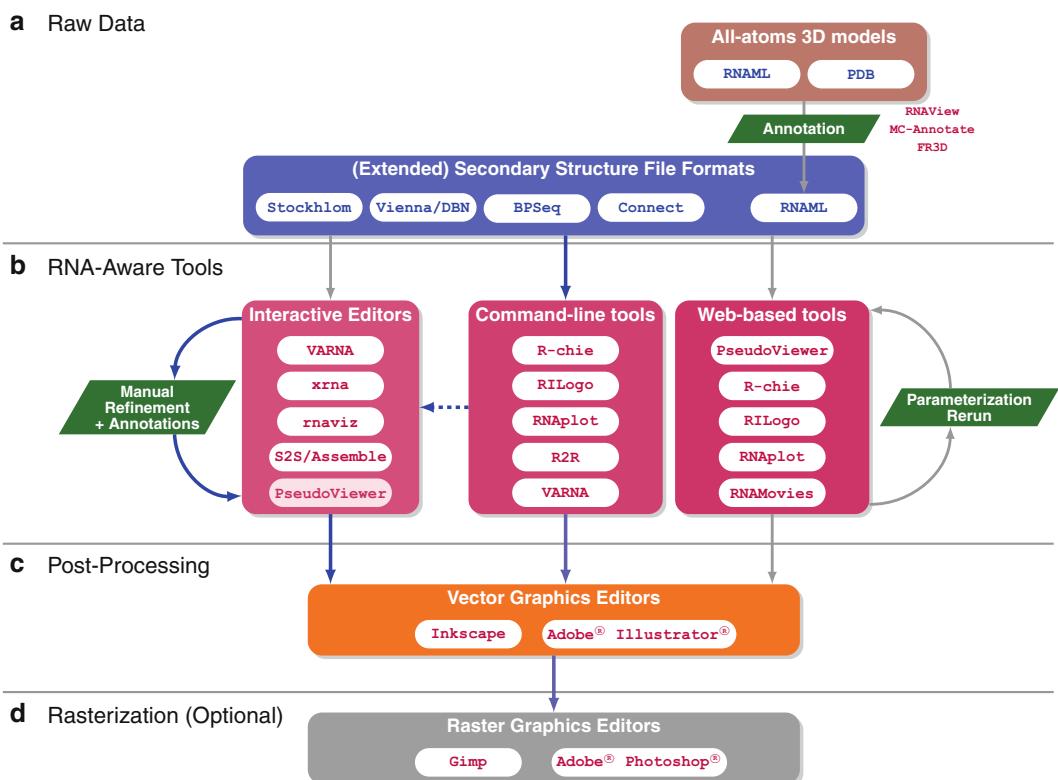


Fig. 9 Typical workflow for the production of publication-quality diagrams. The preferred execution (*blue arrows*, *dashed arrow* indicates a functionality which is not widely available) will start with a scripted production of a data-rich initial draft, followed by an interactive refinement within a specialized GUI, and conclude with a limited post-processing session using a general-purpose vector graphics editor

combination with additional data and annotations, as input for a large number of software which use some algorithm to produce an initial layout. This layout may then be further edited within a dedicated GUI, e.g. to refine the automated layout, or to add further annotations to the drawing. Finally, the output may be exported as a picture, encoded using a general purpose picture format. Vector format pictures may then be manually edited using tools such as Inkscape or Adobe® Illustrator®, typically to add legends or merge several pictures into panels.

One should always prefer vector formats (SVG, PDF, EPS/PS) over bitmap graphics (PNG, JPG). Indeed, the latter, having lost track of the underlying geometry, only allows for very basic translations and color adjustment while the former, by retaining a symbolic description of the picture, enables a convenient post-processing including rotations and grouping/dissociation of objects, and the possibility of correcting factual errors such as erroneous nucleotides. Finally, bitmap pictures of virtually any resolution can be generated from a vector graphics, while the production of a bitmap picture amounts to being blocked in a given resolution. Such a commitment is arguably dangerous before the final zoom level of the picture and the requirements of the publisher are known. Working with bitmap picture may lead users to create files of extreme resolutions, which are subsequently difficult to handle, and sometimes even store. Therefore, the rasterization of RNA secondary structure pictures should be postponed as much as possible in the figure-creation process.

3.2 Representations, Layouts, and Basic Usage

Many representations have been proposed to display the secondary structure of RNA, each with its strengths and limitations. In this section, we give an overview of the existing propositions and, for each representation and tool, we provide minimal command lines and/or python scripts to obtain them.

3.2.1 Linear Layout

The linear layout represents base-pairs as arcs drawn over and/or under a linearly drawn RNA sequence. It arguably constitutes one of the easiest ways to represent the secondary structure, and can be thought as a genomic perspective over the secondary structure. Among the strengths of this representation, one counts an easy identification of helices as sets of nested arcs, and a convenient (unbiased) representation of pseudoknots. This representation may also be easily adapted into a side-by-side visual comparison of two or several candidate structures for a given RNA, provided an alignment is available at the sequence level. In such a joint representation, shared base-pairs are easy to identify, as illustrated in Fig. 10.

However, the linear representation suffers from a poor density of information. Indeed, the height of arcs associated with base-pairs typically grows proportionally to the distance between its associated

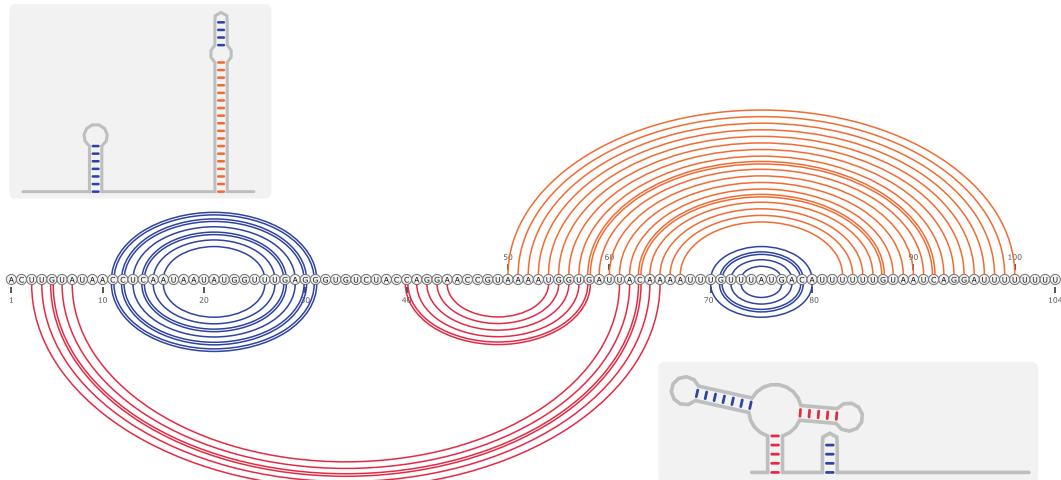


Fig. 10 ON (above) and OFF (below) states of the pbuE adenine riboswitch [19], jointly represented using a linear layout created using **VARNA** (see actual command in Note 1) [7], followed by minimal post-processing. Shared structural elements (blue arcs) are easily identified in this representation, a property which is not satisfied by their typical graph layout (gray boxes)

positions, leading to diagrams whose total area scales quadratically with the sequence length. Consequently, one must either accept to lose the fine details of the sequence/structure or resort to interactive visualization strategies based on zoom/pan navigation operations. Such strategies, however, will impede the visualization of long-range interactions, by forbidding the simultaneous visualization of—sequentially distant—interacting positions.

Such diagrams can be generated, for instance starting from a DBN file XXX.txt, using VARNA [7]:

```
# Running VARNA (Linear layout/SVG output)
java -cp VARNAvx-y.jar fr.orsay.lri.varna.applications.VARNAcmd
-i XXXX.yyy -o out.svg -algorithm line
```

or the RChie [18] software:

```
# Running R-chie (Linear layout/PDF output)
Rscript rchie.R --format1 "vienna" --pdf --output=out.pdf XXX.yyy
```

jviz.RNA [32] may also be used to generate such a representation in an interactive way.

3.2.2 Circular Layout

The circular layout constitutes a modified version of the linear layout, in which the sequence is drawn along a circle. In this representation, base-pairs are drawn either as arcs or as chords of the circle, as illustrated in Fig. 11.

This representation is usually more compact than its linear counterpart (albeit only by a constant factor), and puts more

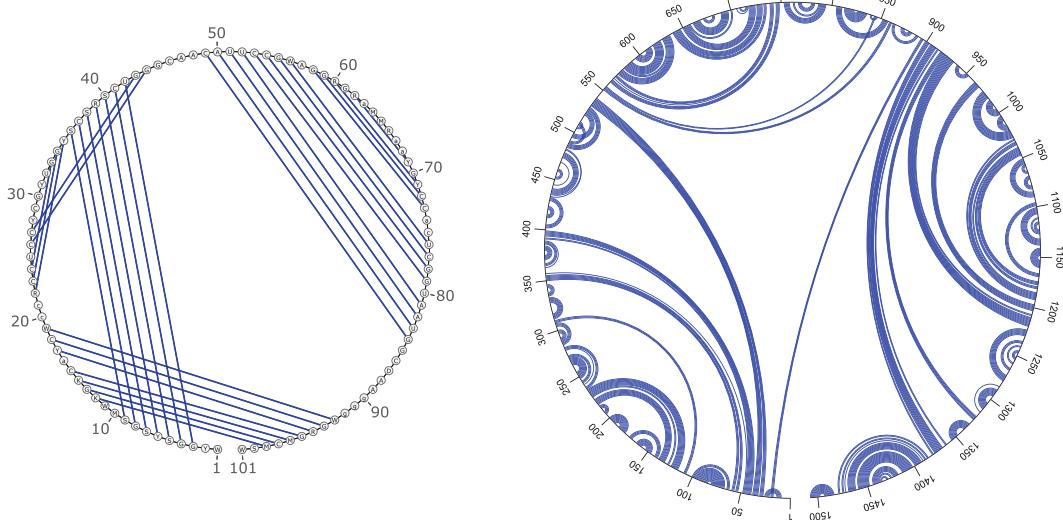


Fig. 11 *Left:* RFAM consensus sequence/structure for the Hepatitis delta virus ribozyme family (RFAM id: RF00094), drawn using **VARNA** [7]; *Right:* comparative model for the 16s ribosomal RNA in *Thermus thermophilus* (source: comparative RNA Web site [6]), drawn using **jViz.RNA** [32]

emphasis on long distance interactions. However, one must remain cautious in a visual analysis of such a diagram, as the height of an arc is no longer strictly proportional to its distance. This phenomenon has a particularly strong impact on the representation of helices involving both ends of an RNA. In this case, helices are represented by chords/arcs which are in close proximity to the circular support (see helix supported by (19,93) in Fig. 11, Left), and may erroneously be interpreted as a local structural feature.

This representation may also be of limited help to build one's intuition regarding the similarity of secondary structures. For instance, drawing two structures simultaneously using both inward and outward base-pair will not even assign similar-looking patterns to perfectly conserved helices. Furthermore, it does not easily allow for a visual realignment of homologous structures, as minor changes in the structure will lead to helices having different orientations. For these reasons, the circular layout is typically reserved for an automated broad overview of the structural organization in larger RNAs, mixing long- and short-range interactions.

Preferred software for producing this representation includes **jViz.RNA** [32] (using the GUI + cautious post-processing, *see Note 2*), or **VARNA** [7], invoked as:

```
# Running VARNA (Circular layout/PostScript output)
java -cp VARNAvx-y.jar fr.orsay.lri.varna.applications.VARNAcmd
-i XXXX.yyy -o out.eps -algorithm circular
```

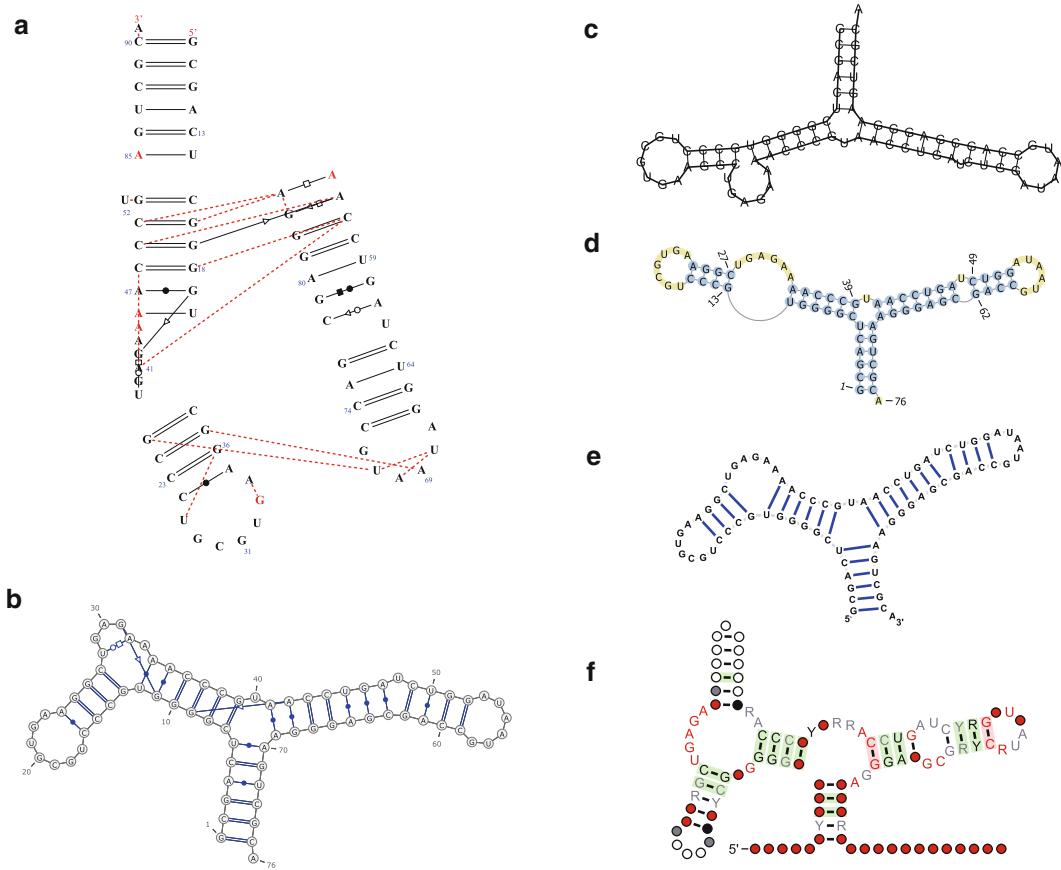


Fig. 12 Various layout strategies for the display of (extended/consensus) RNA secondary structures as (outer) planar graphs, also known as squiggle plots. *Source:* three-dimensional model of the TPP riboswitch (PDB id:2HOM), and associated RFAM family (RFAM: RF00059)

3.2.3 Squiggle Plots:

Planar Graph Representations

This representation attempts to draw the secondary structure as a schematics of its three-dimensional structure. Helices are almost universally represented as straight, ladder-like, segments. Besides this norm, layout strategies largely differ, especially with respect to the layout of multiples loops (3-way junctions and more), as summarized in Fig. 12. This representation is usually preferred to illustrate functional scenario.

RNAVieW (Fig. 12a) draws the secondary structure of an RNA (PDB model) as a direct 2D projection of the three-dimensional model, chosen to maximize the spread of the diagram. This strategy results in drawings which, despite being usually self-overlapping, are often indicative of the relative orientation of helices for small RNAs, as illustrated in Fig. 12a. The software is also mentioned in Subheading 2.2 (Fig. 8) for its capacity to annotate the base-pairs of a 3D model. After downloading/installing the software from <http://ndbserver.rutgers.edu>, a PostScript projection can be produced from an input RNA (PDB format file XXXX.pdb), by simply running the command:

```
# Running RNView on RNA 3D model (PDB format)
rnaview -p XXXX.pdb
```

Note that, in addition to the output PostScript file, this command also creates an RNAML file XXXX.pdb.xml, which contains the extended secondary structure for this 3D model.

VARNA (Fig. 12b) offers two types of layout strategies to render squiggle plots. The first one, named the radial strategy, aims at a consistent spacing of backbone-consecutive and paired nucleotides. To that purpose, unpaired nucleotides are positioned along an imaginary circle, whose radius is computed so that consecutive elements are always distant by a predefined distance. After downloading the latest version of VARNA as a JAR archive VARNAVx-y.jar from <http://varna.lri.fr>, the default radial strategy can be used to draw an extended secondary structure (denoted by an RNAML file, including non-canonical base-pairs, pseudoknots and multiple interactions per position, and output as an SVG file, through the **single-line** command:

```
# Running VARNA on the ext. sec. str. (RNAML -> Vector graphics)
java -cp VARNAVx-Y.jar fr.orsay.lri.varna.applications.VARNAcmd
    -i XXXX.pdb.xml -o YYYY.svg
```

The resulting layout, illustrated in Fig. 12b, does not guarantee non-overlapping diagrams. The result may therefore require further manual post-processing for larger RNAs, a task which can be performed within VARNA’s dedicated GUI, provided that the specific file format varna is chosen for the output:

```
# Running \Software{VARNA} (RNAML -> VARNA session file)
java -cp VARNAVx-y.jar fr.orsay.lri.varna.applications.VARNAcmd
    -i XXXX.pdb.xml -o YYYY.varna
```

Alternatively, the NAView algorithm [4] is implemented in VARNA, and usually ensures a non-overlapping layout, resulting in layouts which are similar to Fig. 12c. It can be invoked by setting the **algorithm** command-line option:

```
# Running VARNA (NAView algorithm -> PostScript output)
java -cp VARNAVx-y.jar fr.orsay.lri.varna.applications.VARNAcmd
    -i XXXX.pdb.xml -o YYYY.eps -algorithm naview
```

A *skeleton* view can also be produced, as shown in both subpanels of Fig. 10 (see Note 1 for precise commands), by using the combination of options below (replacing XXXX with your input file, YYYY with the output file, including an extension which will decide its type, and ZZZ with the length of the RNA):

```
# Running VARNA (NAVview algorithm -> PostScript output)
java -cp VARNAvx-y.jar fr.orsay.lri.varna.applications.VARNAcmd -i
XXXX -o YYYY -flat true -highlightRegion "1-ZZZ:fill=#AOAOAO,
outline=#AOAOAO, radius=10" -drawBases false -backbone "#AOAOAO"
-fillBases "#AOAOAO" -bpStyle simple -numPeriod 50
```

RNAPlot (Fig. 12c), a software which is part of the Vienna RNA package, also uses the NAVview algorithm for its default layout. As can be seen from Fig. 12c, a peculiarity of this layout is its unique orientation, going counterclockwise in the $5' \rightarrow 3'$ direction. After successful installation of the package, downloaded from <http://www.tbi.univie.ac.at/RNA/>, it is invoked on a simple DBN file (see Fig. 3 for an example) xxxx.yyy, by running the command:

```
# Running RNAPlot (NAVview algorithm)
RNAPlot < xxxx.yyy
```

The resulting PostScript drawing is output to a file zzz_ss.ps, where zzz is the first token found on the comment line of the input file (defaults to rna.ps if absent). A radial layout can also be specified through a dedicated option:

```
# Running RNAPlot (Radial algorithm)
RNAPlot --layout-type=0 < xxxx.yyy
```

The PseudoViewer (Fig. 12d) uses an original strategy for choosing the relative orientations of helices. As illustrated in Fig. 12d, pairs of consecutive helices, connected by a bulge or an interior loops, are drawn along the same axis. The resulting structures appear simpler, possibly at the cost of uneven distances along the backbone. Furthermore, multiple junctions/loops are stretched and oriented to completely forbid overlaps. Finally, this advanced layout strategy satisfactorily supports a very large class of pseudoknots. The software can be used online, as a web server or a web service, at <http://pseudoviewer.inha.ac.kr/>. Graphical User Interfaces (GUI) are provided for the Windows platform but, to the best of our knowledge, no command-line version is currently available.

jViz.RNA (Fig. 12e) implements an interactive layout strategy for squiggle plots, which relies on a spring model. Both backbone bonds and base-pairs are modeled as springs, each with a preferred length and tension. The layout is iteratively refined, simulating the springs reactions until an equilibrium is found. As illustrated in Fig. 12e, helices are not necessarily drawn in rectangular boxes, and are not even necessarily straight, leading longer helices to exhibit a *wavy* shape. However, a user may intervene during the process to *disentangle* the structure, or modify the spring tensions to obtain non-overlapping diagrams. Like the PseudoViewer, this software does not offer *offline* command-line capabilities, and

a dedicated GUI, freely available at <http://jviz.cs.sfu.ca>, must be used to produce drawings.

The R2R (Fig. 12f) software also uses a radial layout as its default algorithm. However, more sophisticated layout strategies can be specified, for instance in order to force angles to fall within a restricted list (typically $k \cdot \pi / 8, \forall k \in [0, 15]$). As illustrated in Fig. 12f, another interesting feature is the *flat* drawing of unpaired nucleotides along the exterior loop. Another interesting aspect is the display of comparative information regarding a structural family, represented as a STOCKHOLM-formatted multiple sequence alignment including a consensus structure. The installation and usage of R2R, described in detail in Subheading 3.5, is perhaps a bit more involved than its competitors, but the quality of the resulting picture is clearly worth the effort.

3.2.4 Tree Layout

In the absence of *crossing* interactions such as pseudoknots, RNA secondary structures can be unambiguously decomposed in a variety of ways, leading to tree-like objects. Such a decomposition of the conformation space is at the core of any dynamic programming scheme, a strategy for solving combinatorial optimization problems which is especially popular in RNA bioinformatics. For instance, atomic contributions in the Turner energy model [22] are entirely determined by the joint knowledge of internal nodes (a base-pair) in combination with their immediate children. Consequently, this representation may be useful to illustrate the principles underlying RNA algorithms.

Unfortunately, there are currently few available options to produce such a representation, and the most realistic option consists in using the general-purpose graph visualization software GraphViz [11], which unfortunately requires a DOT-formatted file as input. Consequently, a secondary structure, typically denoted by a dot-parenthesis expression (see Subheading 2.2.2), will require some conversion. We provide in Note 3 a minimal python (2.x) script to convert, and dump to the standard output as a DOT-formatted file, a secondary structure.

Invoking this code for a given sequence/structure, while capturing the output through standard I/O redirection, one obtains a file which can then be fed to the `dot` utility of GraphViz (See Fig. 13). This utility can then be configured to produce quality pictures in virtually any format, including many vector formats such as PDF and SVG.

3.3 Mapping Probing Data onto Secondary Structure Models

Footprinting techniques are widely applied to map the 2D and 3D structures of RNA using both chemical and/or enzymatic probes. They provide useful information about the relative accessibility of the RNA to a given probe at the nucleotide resolution. Depending on the physico-chemical conditions or the presence of different molecular partners (RNA, proteins, antibiotics, etc.), the chemical or enzymatic probing data give a snapshot of the RNA structure

(((((.....((((.....))))(((((.....))))....(((.....))))))))....

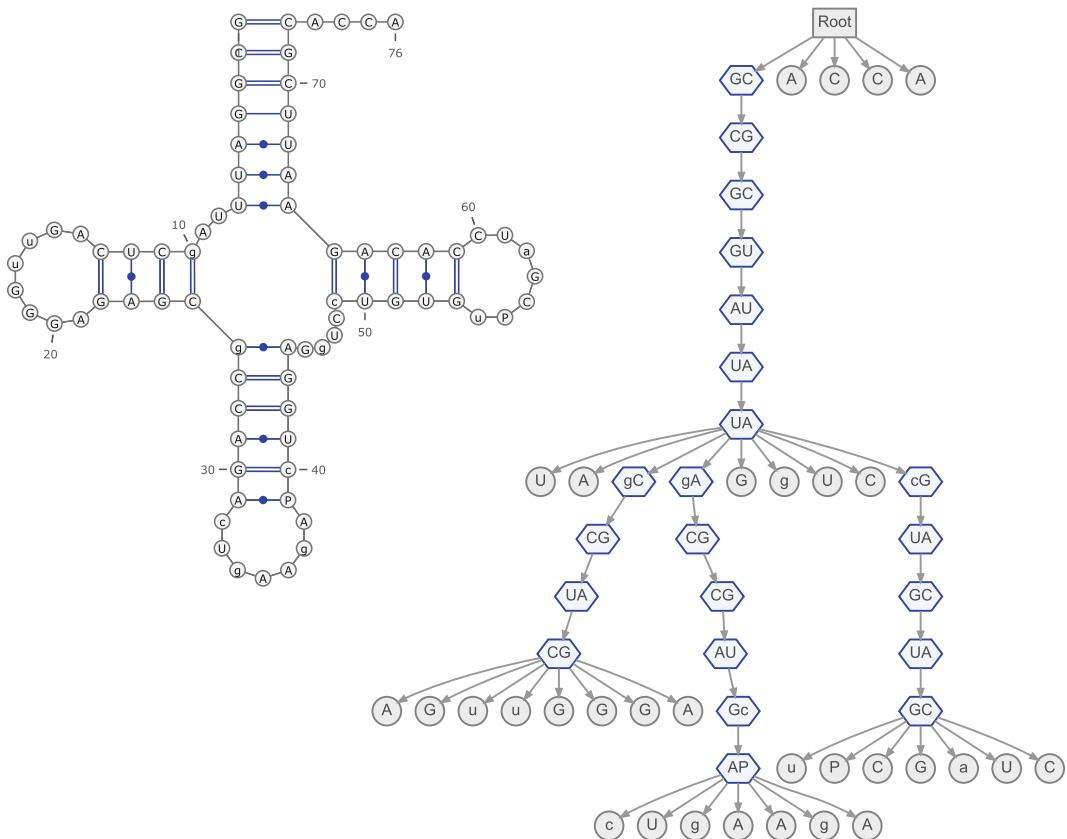


Fig. 13 Planar graph layout using **VARNA** [7] and tree representation using the default options of the **dot** [11] algorithm of **GraphViz** [10] of the secondary structure of a transfer RNA (PDB id: 1TN2:A, annotated using **RNAView** [34])

under specific conditions. They are used to study the conformational changes during RNA folding or induced upon protein or ligand binding [33]; they also nicely complement the structural data obtained on the conformational changes associated with RNA folding [28] or RNA/protein interactions [9].

With the development of high-throughput techniques using Selective 2'-Hydroxyl acylation Analyzed by Primer Extension (SHAPE) chemistry (SHAPE-seq [24]), UV-crosslinking or immunoprecipitation (HITS-CLIP [36]), emerging computational methods attempt to integrate such data into RNA structure predictions [20, 26, 35]. However, these predictions are never entirely accurate, and must be confronted with the experimental data in an iterative refinement. Visualization is therefore an important aspect in the process of generating primary 2D representations and models. Probing data can be represented using graphical

annotations (symbols, color maps and marks, labels, etc.) mapped onto a 2D structure.

In this use-case, we focus on probing data obtained on the H/ACA guide RNAs from the snoR9 family (RFAM: RF00065) in *Pyrococcus abyssi* [8]. Here, one may map chemical probing data onto a secondary structure by generating different representations for the same RNA molecule, corresponding to chemical probing data obtained under different conditions. Since the structure is used to provide a context for the interpretation of probing data, one will typically use a template for the secondary structure (sequence/data and layout) onto which the molecular properties are displayed. Chemical probing data can be represented in different ways:

- Symbols may be used to indicate the degree of accessibility to some chemical or enzymatic probe (Fig. 14b);
- A color map can also be used to superimpose the accessibility directly onto the secondary structure (Fig. 14c);
- The chemical probing data associated with conformational changes of the RNA (folding or interactions with ligands) can be displayed in different panels to provide a graphical indication of the condition-dependent changes, affecting the tertiary and secondary structures.

VARNA offers multiple features to ease the production of such diagrams. Specifically, an extensive array of command-line options can be used to produce a collection of pictures that share a general orientation. Three types of annotations are of particular interest:

- **Color maps:** A color map associates numerical values to each position in an RNA (`colorMap` option), and uses a (multiple) color gradient (`colorMapStyle` option) to associate colors to each position, based on their associated value. Minimal and maximal values can be manually input (`colorMapMax` and `colorMapMin` options), otherwise they are simply guessed from the values. For color maps, the tedious task of manually inputting the values (e.g., chemical reactivities), on a nucleotide-per-nucleotide basis, can be avoided by loading an external file containing all the data at once from within the GUI. Alternatively, an ad hoc script may format the command-line such that those values are directly provided to the software. This usage is illustrated by the code listed in **Note 4** whose execution produces Fig. 14c.
- **Glyphs:** Arrows, pins, or additional symbols are sometimes used to indicate significantly accessible (backbone) regions, or a difference in accessibility. Position-specific markers can be specified using the `chemProb` options in VARNA, as shown in Listings 4, resulting in the corresponding panels of Fig. 14 (Panels b–f).

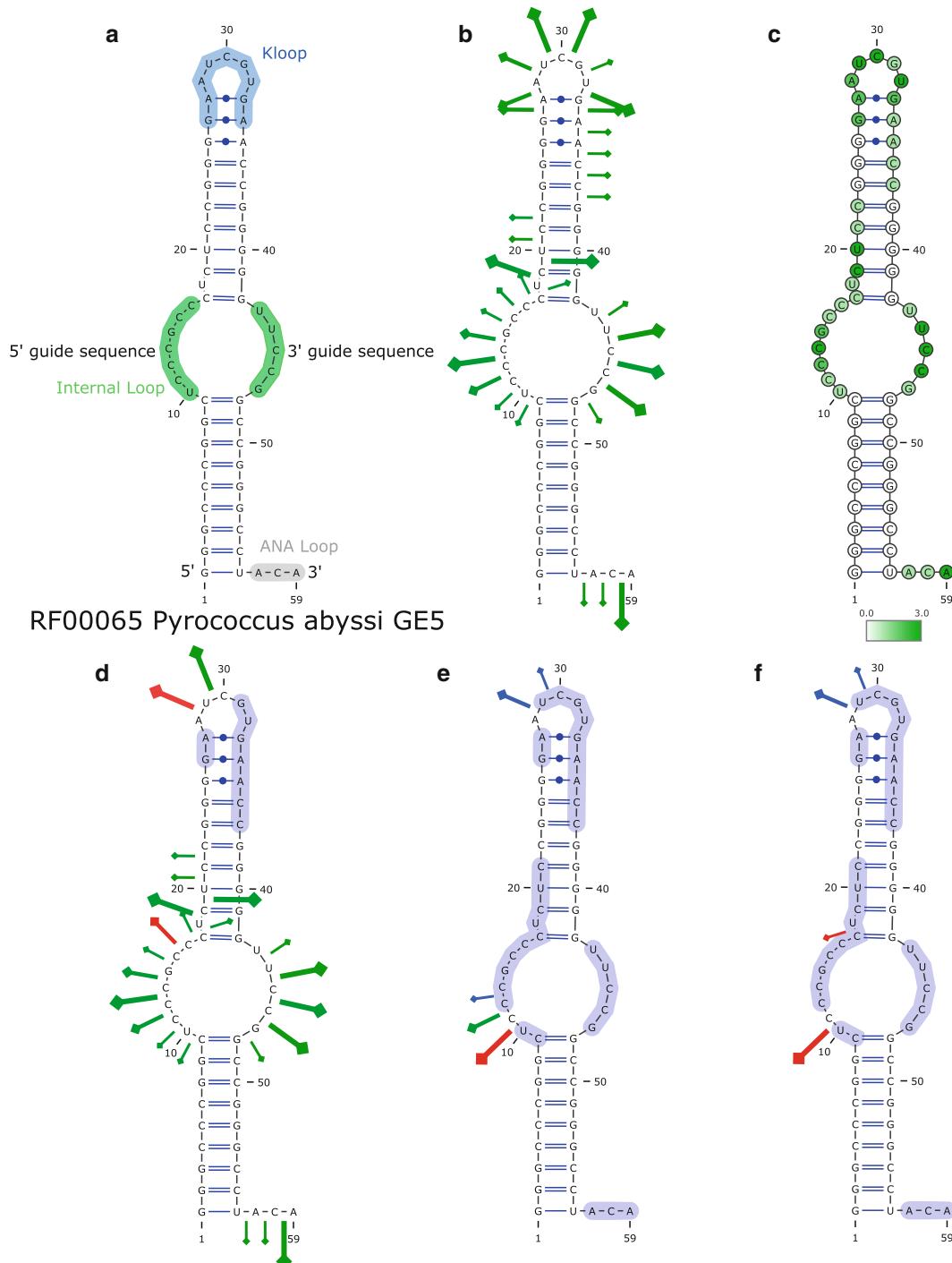


Fig. 14 2D structure representations integrating chemical probing of the Pab21 HACA sRNP from *P. abyssi* (RFAM: RF00065), as generated by **VARNA**. **(a)** 2D structure representation of the HACA RNA motif highlighting the key structural features. **(b)** Free RNA Chemical probing indicating the accessible positions for Pb cleavage on a scale from 1 to 3 in the intensity of the reaction. **(c)** Alternate representation showing the accessibility using a color map. **(d–f)** Modifications of the RNA probing data due to the successive binding of L7Ae, CBF5 and NOP10, and the RNA substrate. *Green pins*: initial probing on RNA; *red pins*: increase in reactivity; *blue pins*: decrease in reactivity; *violet regions*: newly protected positions

- **Highlighted Regions (a.k.a. sausages):** Regions highlights may be used to emphasize a change of reactivity observed on a portion of the backbone, possibly hinting towards a local conformational change. To produce such annotations, the highlightRegion option in VARNA may be used, and allows to specify the width and color of the region, as shown in panels d, e, and f of Fig. 14 (produced as shown in Note 4).

RNAplot can also be used to display accessibility values as a color map, using the --pre and --post options to *decorate* the PostScript output. Some basic examples can be found in the listing below:

```
# Running RNAplot from a DBN file 'XXX.txt'

# Marking nucleotide 10
RNAplot --post "10 cmark" < XXX.txt
# Drawing backbone for region (10,15) using red color (r=1,g=b=0) and line thickness 2
RNAplot --pre "10 15 2 1. 0. 0. omark" < XXX.txt
# Defining a custom PS macro and using it to fill base number 10 in blue (r=g=0, b=1)
RNAplot --pre "/cfmark {setrgbcolor newpath 1 sub coor exch get aload pop fsize 2 div 0 360
arc fill} bind def 10 0. 0. 1. cfmark" < XXX.txt
```

As can be seen in this last example, achieving even simple operations may require an extensive knowledge of the PostScript language, whose suffix-order for operations induces a steep learning curve. Furthermore, formatting the command line manually would quickly become tedious. Therefore, we propose in Listing 5.1 a minimal Python wrapper which, given a DBN formatted RNA and a list of values, invokes RNAplot to produce a *color map-decorated* output similar to that of Fig. 14c:

Listing 5.1 Using RNAplot to display probing values as a color map

```
import os,sys
def getColor(val, minval, maxval,col1=(1.,1.,1.),col2=(0.0,0.8,0.0)):
    (r1,g1,b1),(r2,g2,b2) = col1,col2
    span = float(maxval)-float(minval)
    k = (float(val)-float(minval))/(span if span!=0 else 1.0)
    l = 1.-k
    return (r1*l+r2*k,g1*l+g2*k,b1*l+b2*k)
def formatValuesAsPS(values):
    minval,maxval = min(values),max(values)
    valtab = [ "%s %s cfmark"%((i+1,"%.2f %.2f %.2f")%getColor(v,minval,maxval)) for i,v in
              enumerate(values)]
    return "".join(valtab)
# Invokes RNAplot to display a color map
# Arg1: path to DBN file
# Arg2: list of comma-separated values (eg "1,2,3")
if __name__ == "__main__":
    inputFile = sys.argv[1]
    values = sys.argv[2].split(",")
    extraMacro = "/cfmark {setrgbcolor newpath 1 sub coor exch get aload pop fsize 2 div 0 360
    arc fill} bind def"
    os.system("RNAplot --pre \"%s %s\" < %s"%(extraMacro,formatValuesAsPS(values),inputFile))
```



3.4 3Displaying Pseudoknots and Interactions

Pseudoknots and RNA–RNA interactions depart from the tree-like paradigm, and therefore constitute a challenge to computational methods. Most prediction algorithms address the problem by restricting the search space. However, visualization tools cannot arbitrarily enforce such restrictions. This narrows down the choice of suitable layouts and tools to few options.

3.4.1 Pseudoknots

Basic representations, such as the linear and circular representations, do not require a complicated layout, and therefore remain largely unaffected by the presence of pseudoknot. From a software designer perspective, supporting pseudoknot for these representations is then mostly a matter of being able to parse suitable input formats. *jViz.RNA* (Fig. 15b), R-chie and VARNA (Fig. 15d) may be used without much additional effort. In particular, the latter will also support more complicated structures, such as those featuring multiple partners for a given nucleotide, as illustrated by Fig. 15d.

However, a large subset of existing pseudoknots can be represented planarly as squiggle plots, i.e. in such a way that base-pairs and backbone are non-crossing. To create such representations, the *PseudoViewer* is the only fully satisfactory option. Pseudoknots are laid out within dedicated boxes, that are embedded in a tree-like general layout. As shown in Fig. 15a, the result is aesthetically pleasing even in the absence of user intervention, its only drawback being a consequent backbone distortion. Alternatively, the spring layout of *jViz.RNA* can be used to obtain decent squiggle plots, as illustrated in Fig. 15c usually after some manual disentangling from the user.

3.4.2 Interactions

The need for automated representations of RNA–RNA interactions is relatively recent, and has arisen from a recent increase of interest in the computational prediction of RNA–RNA interactions.

RILogo is currently the only automated tool that produces representations of RNA–RNA interactions. The web version requires as input either one or two multiple sequence alignments (Stockholm or a modified version of FASTA). In the case of a single file, the consensus secondary structure will use a special character & to indicate the end of the first RNA and the beginning of the second. The base-pairs which are split by the separator then be involved in the hybridization of the two RNAs. Alternatively, two Stockholm files may be specified, using one or several parentheses systems (**A/a, B/b ...**) to indicate the subset of base-pairs involved in the interaction as follows:

Listing 5.2 Fasta-like input file xxxx.txt expected by RILogo

```
>organism1
GCGGGGUUGCGC&AGGACCCACUCCU
>organism2
CGGCCCGGCCG&AGCGGGCCGCGCU
>structure
(((AABB)))&(((aabbb))))
```

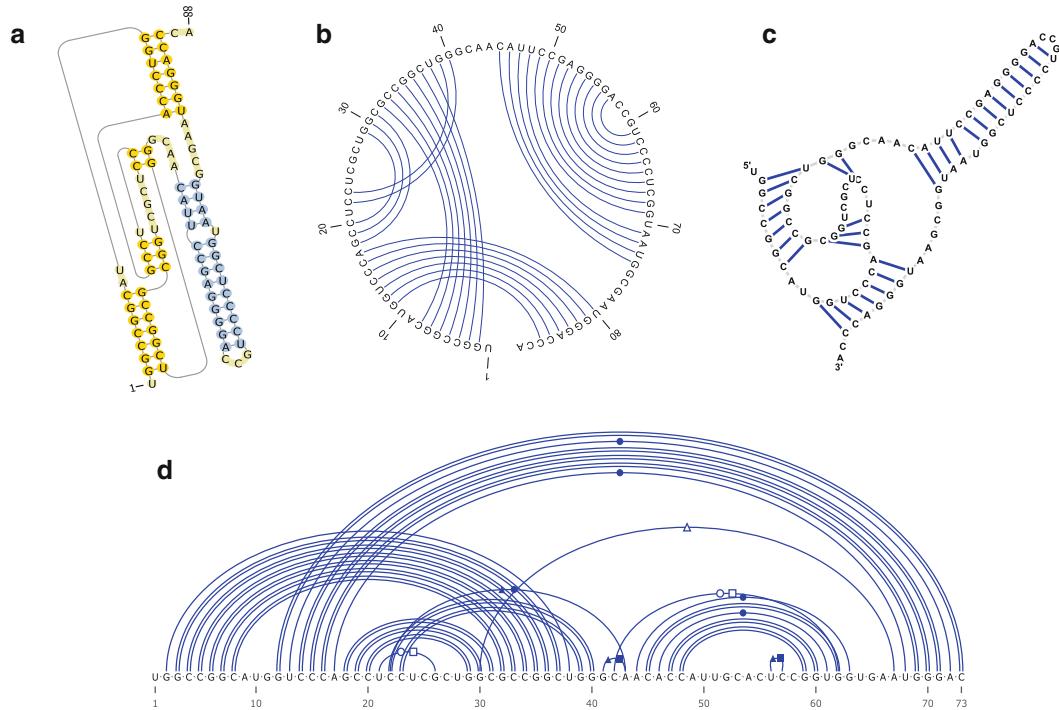
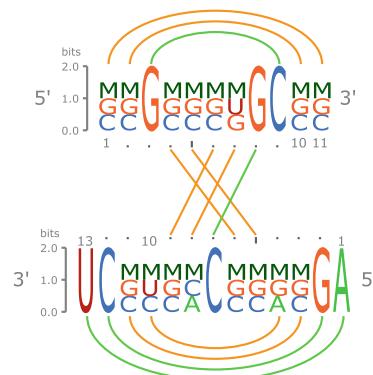


Fig. 15 Pseudoknotted secondary structure of the Hepatitis Delta virus ribozyme (PseudoBase entry PKB75), rendered as a planar squiggle plot by the **PseudoViewer** (a), as a circular diagram and as a *force-driven* spring layout by **jViz.RNA** (b and c). Extended secondary structure of a variant (PDB id: 1SJ3), drawn linearly by **VARNA** (d)

Listing 5.3 Invoking RILogo

```
rilogos XXXX.txt > output.svg
```



Feeding such a file to the **RILogo** webserver or software (both available at <http://rth.dk/resources/rilogo>) as illustrated in Listing 5.3 one obtains arc-annotated (linear) representations of the interacting families, similar to the ones above and shown in Fig. 16a.

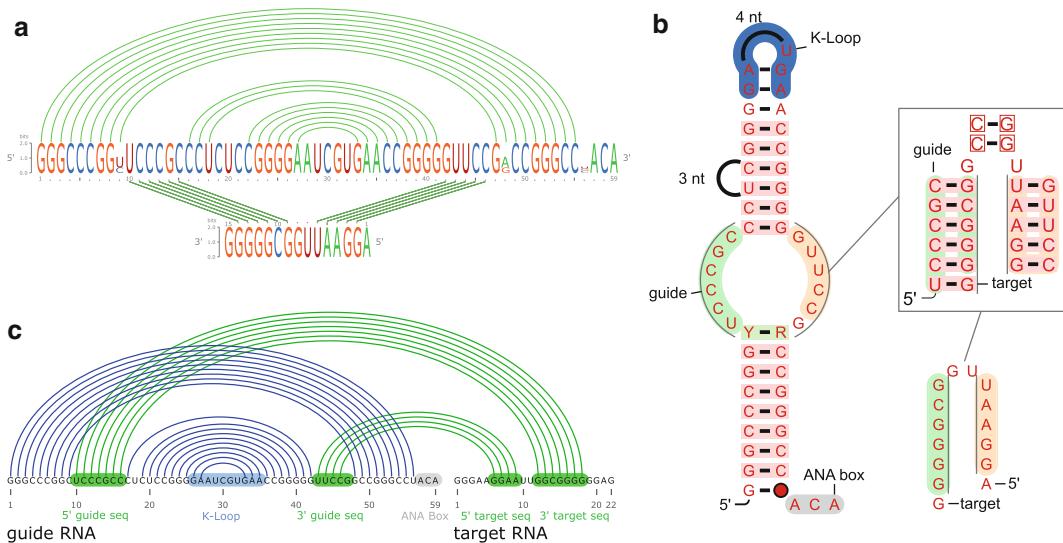


Fig. 16 Available representations for RNA–RNA interactions: consensus secondary structures of the H/ACA guide and its target sequence from the 16S rRNA (RFAM: RF00065) automatically generated by **RILogo** (Panel **a**), or as a manually edited pseudoknotted structure using **R2R** (Panel **b**) and **VARNA** (Panel **c**)

Neither R2R nor VARNA offers a native support for RNA–RNA interactions. However, it is possible to use some tricks to represent consensus sequence and secondary structures for RNA–RNA interactions. Within R2R, using templates and options designed for pseudoknots, one may generate some representations where both interacting RNAs are merged into the same alignment using some random sequence spacer. Then, the sequence spacer and its associated annotations may be manually removed using appropriate software (`Inkscape` or commercial alternatives) to display a clean representation of consensus sequences and secondary structures of two interacting RNAs (see Fig. 16b). VARNA may also be used in its linear representation to represent the interaction as a pseudoknot, possibly in combination with annotations, and manually post-process the output to obtain result similar to Fig. 16c.

3.5 Visualizing Structure-Informed RNA Alignments

The identification of novel families of structured ncRNAs relies heavily on the implementation of a comparative approach. This approach starts with a multiple sequence alignments, from which a draft consensus secondary structure is identified. This draft is then used to refine the alignment, and retrieve novel homologs, upon which the secondary structure can be further refined. Iterating these steps leads to a final structure-informed multiple sequence alignment, whose quality must be assessed before concluding on the existence of a new functional family. To that purpose, a global visualization of the conservation/covariation levels in the context of the structure is essential.

The R2R software is particularly suited to generate representations of structure-informed RNA alignments. Beside allowing to decorate a squiggle plot using conservation and covariation levels,

R2R also allows to define subfamilies of sequences, possibly associated with diverging secondary structures. Another unique feature of R2R is its definition of a complete set of directives to express preferences. These preferences may then be used to produce sets of similar-looking representations for individual sequences, helping in the visual identification of differences. Figure 17, a panel of secondary structure representations produced for the RF00065 RFAM family (*see Note 5* for precise commands), illustrates such capabilities. Here, the consensus structure of the RF00065 family is displayed using variable-length regions and color-shaded backbone regions for sequence elements that vary in size associated with the internal loop, terminal loop, or 3'-end regions of the RNAs. Each member of the family is also displayed with the full sequence to view the details of each member in the family. Two subfamilies corresponding to different sizes in the 5' guide sequence (7 or 8 nucleotides) of the internal loop are shown to zoom on the variations in the loop size.

A typical usage of R2R includes the following steps:

Step 1 (Initial Alignment) Start from a Stockholm file XXXX.sto featuring a multiple sequence alignment, along with a consensus secondary structure denoted in WUSS notation. The software will refuse a *blocked* alignment (in which sequences are broken into blocks of fixed lengths), and we provide in Listing 5.4 a minimal Python script to transform a blocked Stockholm file into a linear one.

Listing 5.4 Transforming an Stockholm-formatted RFAM multiple sequence alignment into an unblocked alignment accepted by R2R as input

```
import os,sys,string
def unblockStockholm(inpath,outpath):
    outfile = open(outpath,"w")
    (seqIDs,seqContents) = ([],{})
    for l in open(inpath):
        if len(l)>12 and ((not l.startswith("#")) or (l[:12] in ["#=GC SS_cons","#=GC RF      "])):
            (id,content) = (l[0:37].strip(),l[37:-1].strip())
            if id not in seqContents:
                seqIDs.append(id)
                seqContents[id] = ""
                seqContents[id] += content
            elif len(l)>1:
                for id in seqIDs:
                    outfile.write(string.ljust(id, 37)+seqContents[id]+"\n")
                seqIDs,seqContents = [],{}
                outfile.write(l)
    outfile.close()
# Transforms a 'blocked' STOCKHOLM file (eg RFAM alignment) into a linear one
# Arg1: path to 'blocked' STOCKHOLM file
# Arg2: path to 'linearized' STOCKHOLM output file
if __name__=="__main__":
    (inStoc,outFile) = (sys.argv[1],sys.argv[2])
    unblockStockholm(inStoc,outFile)
```

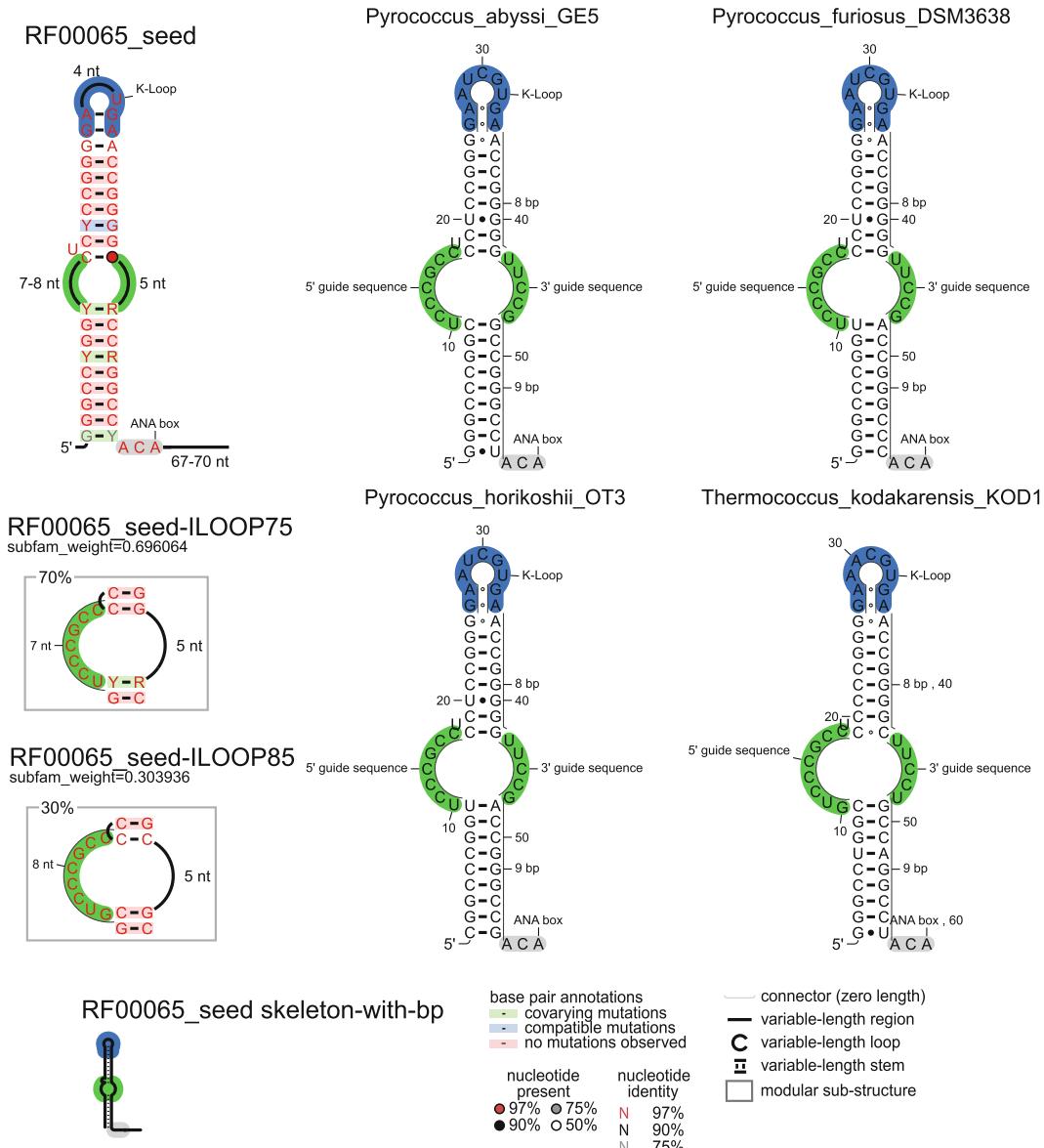


Fig. 17 Consensus sequence and secondary structures of the H/ACA guide RNA from the snoR9 family (RFAM: RF00065) generated by R2R (see Note 5 for a sequence of commands) from the RFAM seed alignment. Four individual secondary structures from the seed alignment are annotated to display the common structural features (internal loop, K-loop, ANA box) and the subtle differences in sequence and loop size. The modular sub-structures correspond to two subfamilies: ILOOP75 and ILOOP85 where the 5' guide sequence of the internal loop contains either 7 or 8 nucleotides, respectively

Step 2 (Computing statistics) Conservation and covariation levels must be computed, and appended to the alignment into a file XXXX.cons.sto by running the command

```
# Compute conservation levels
r2r --GSC-weighted-consensus XXXX.sto XXXX.cons.sto 3 0.97 0.9 0.75 4
0.97 0.9 0.75 0.5 0.1
```

The numerical values occurring in the above command correspond to the recommended thresholds and parameter values, and we direct to the manual any user with specific needs and expectations in this respect.

Step 3 (User preferences) The produced file may then be manually edited to specify preferences regarding the layout and annotations. To that purpose, *generic feature* lines, starting with **#GF =** can be added appended to the file prior to the terminal *nn* characters. A huge collection of such features is available, whose complete listing would perhaps require a dedicated chapter. We illustrate some of the most salient features in **Note 6** and **7**.

Step 4 (Batch metafile) Once these preferences have been expressed and stored within a Stockholm file XXXX.user.sto, a meta file must be defined. In a minimal setting, it may simply contain the name of the processed Stockholm input file. More complex examples, as shown in **Note 8**, may include several lines to produce multiple figures. These figures may either represent the content of multiple Stockholm files, focus on specific sequences/organisms (see `oneseq` keyword), or produce a coarse-grained *skeleton* view (see `skeleton-with-pairbonds` keyword). For instance, the following meta file produces:

1. a consensus drawing;
2. a drawing for *Pyrococcus furiosus*, adhering to the same layout rules;
3. a *skeleton* view of the consensus.

```
XXXX.user.sto
XXXX.user.sto oneseq Pyrococcus_furiosus
XXXX.user.sto skeleton-with-pairbonds
```

Step 5 (Running R2R) Once the meta file XXXX.meta is ready, the software can simply be run through either of the following commands:

```
r2r XXXX.meta XXXX.pdf
r2r XXXX.meta XXXX.svg
```

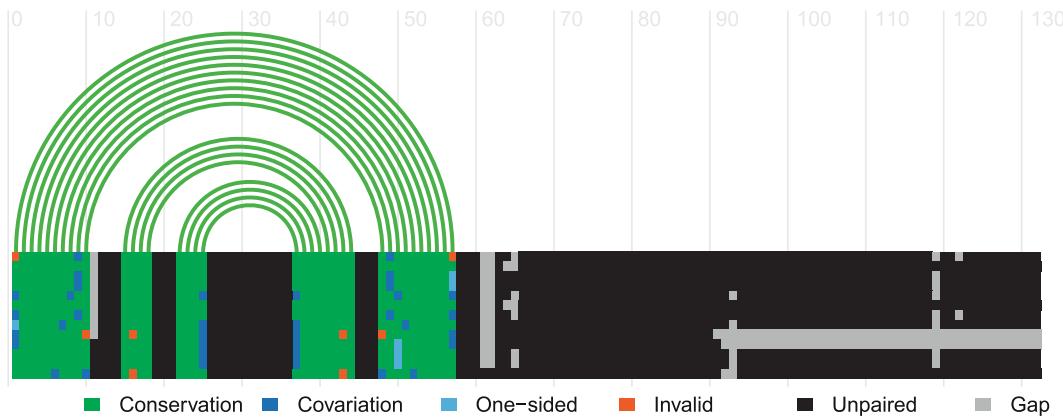


Fig. 18 Structure-informed representation of the RFAM multiple-sequence alignment of the snoR9 family (RFAM: RF00065), produced using **R-Chie**

to produce either a PDF or SVG output, amenable to further vector post-processing.

The representations offered by R2R allow for a critical inspection of a proposed consensus structure. However, they do not necessarily inform the viewer regarding the quality of the underlying alignment. Therefore, we advocate complementing this representation with sequence-focused alternatives produced by R-chie, an R package which uses a linear layout to draw one or several structures, stacked on top of a multiple sequence alignment. This sequence alignment is typically color-encoded to indicate conservation, insertion, compatibility, or contradiction with the consensus structure. R-chie can be invoked online, or downloaded at <http://www.e-rna.org/r-chie>. Once the necessary R dependencies are installed, it can be executed on a *gapped* FASTA multiple sequence alignment XXXX.txt and a Vienna/DBN consensus structure YYYY.txt via the following command:

```
Rscript rchie.R --msafile XXXX.txt --colour1 "#4DAF4A" --msacol "#00A651,#0072BC,#00BF2,#F15A22,#231F20,#AAAAAA,#DAGFAB" --pdf
--output="out.pdf" --format1 "vienna" YYYY.txt
```

Running the command produces a PDF file named out.pdf similar to that shown in Fig. 18.

4 Notes

- Execution of the command-line standalone version of VARNA to generate Arc representations (see Fig. 10 for graphical rendering).

```

# Top
java -cp VARNAvx-y.jar fr.orsay.lri.varna.applications.VARNAcmd \
-sequenceDBN "GGGCCCGGCUCGGCCUCUCCGGGGAAUCGUGAACCGGGGUCCGGCCGGGUAC" \
-structureDBN "((((((.....(.(((((((.....))))))))....))))...." \
-highlightRegion "10-16:radius=10,fill=#acf269,outline=#acf269;43-47:radius=10,fill=#acf269,
outline=#acf269" \
-o RF00065_testarc1.svg -algorithm line -drawBases False -spaceBetweenBases 0.6 -bpStyle line

# Bottom
java -cp VARNAvx-y.jar fr.orsay.lri.varna.applications.VARNAcmd \
-sequenceDBN "GGGCCCGGCUCGGCCUCUCCGGGGAAUCGUGAACCGGGGUCCGGCCGGGUAC" \
-structureDBN "((((((.....(.(((((((.....))))))))....))))...." \
-chemProb "10-11:glyph=pin,dir=out,intensity=3,color=#f90000;17-18:glyph=pin,dir=out,intensity
=1,color=#f90000;" \
-highlightRegion "26-27:radius=10,fill=#d2ccf4,outline=#d2ccf4;31-37:radius=10,fill=#d2ccf4,
outline=#d2ccf4" \
-o RF00065_testarc2.svg -algorithm line -drawBases False -spaceBetweenBases 0.6 -bpStyle line

```

2. jViz.RNA tends to create extremely large EPS files, which cannot be easily manipulated within existing vector graphics editors. This problem apparently originates from the way characters are output. Indeed, texts are apparently not exported as PostScript strings, but as splines defined using a large number of points. Besides increasing the file size, this means that the nucleotides and annotations cannot be easily manipulated within dedicated editors. Therefore, unless a bitmap picture is eventually sought and very limited post-processing is involved, we generally do not recommend using jViz.RNA to draw RNAs of length greater than 300 nts.
3. Minimal python script to convert a secondary structure, denoted by a dot-parenthesis notation, into a Graphviz input file (DOT format) amenable to a tree-like layout.

```

import sys
# Get custom styles by changing these lines (cf GraphViz manual)
STYLE_DEFAULT = "shape=\\"rectangle\\",style=filled,margin=\"0,0\\",fontsize=20,color=grey40,
    fontcolor=grey20,fillcolor=grey90,fontname=\"Helvetica\\"
STYLE_UNPAIRED = "shape=\\"circle\\",color=blue,fillcolor=aliceblue"
STYLE_PAIRED = "shape=\\"hexagon\\\""
STYLE_EDGES = "color=grey50"
# Converts an RNA sequence/secondary structure (dot-parenthesis notation) into a DOT-formatted
# GraphViz input, written into a previously opened file f
def drawAsDOT(seq,secstr,f):
    print >> f, "digraph rna{\n    node [%s];\n    edge [%s];\n    -1 [label=\"Root\"];"%(
        STYLE_DEFAULT,STYLE_EDGES)
    stack = [-1]
    for i in range(len(secstr)):
        k = stack[-1]
        if secstr[i]==":":
            stack.append(i)
            print >> f, "    %s -> %s;"%(k,i)
        elif secstr[i]==")":
            stack.pop()
            print >> f, "    %s [label=\"%s\",%s];"%(k,seq[k]+seq[i],STYLE_PAIRED)
        else:
            print >> f, "    %s -> %s;"%(k,i)
            print >> f, "    %s [label=\"%s\",%s];"%(i,seq[i],STYLE_UNPAIRED)
    print >> f, "}"
#Typical usage of this script (RNAToDOT.py):
#   python RNAToDOT.py (((...))) GGGAUACCC > rnaTree.dot
#   dot -Tpdf -o rnaTree.pdf rnaTree.dot
if __name__=="__main__":
    struct,seq = sys.argv[1],sys.argv[2]
    drawAsDOT (seq,struct,sys.stdout)

```

4. Execution of the command-line standalone version of VARNA for a series of representations (*see Fig. 14* for the graphical rendering).

```
# panel A
java -cp VARNAvx-y.jar fr.orsay.lri.varna.applications.VARNAcmd \
-sequenceDBN "GGGCCCGCUCCCGCCCUUCGGGGAAUCGUGAACCGGGGUUCGGCCGGCCUACA" \
-structureDBN "((((((.....(.((((((.....))))))))....))))...." \
-highlightRegion "10-16:radius=10,fill=#acf269,outline=#acf269;43-47:radius=10,fill=#acf269,
    outline=#acf269" \
-o RF00065_panelA.svg -algorithm radiate -flat true -drawBases False -spaceBetweenBases 0.6 \
-title "RF00065: Pyrococcus abyssi GE" -titleSize 12

# panel B
java -cp VARNAvx-y.jar fr.orsay.lri.varna.applications.VARNAcmd \
-sequenceDBN "GGGCCCGCUCCCGCCCUUCGGGGAAUCGUGAACCGGGGUUCGGCCGGCCUACA" \
-structureDBN "((((((.....(.((((((.....))))))))....))))...." \
-chemProb "9-10:glyph=pin,dir=out,intensity=1,color=#3e844b;10-11:glyph=pin,dir=out,intensity
    =1,color=#3e844b;" \
-o RF00065_panelB.svg -algorithm radiate -flat true -drawBases False -spaceBetweenBases 0.6

# panel C
java -cp VARNAvx-y.jar fr.orsay.lri.varna.applications.VARNAcmd \
-sequenceDBN "GGGCCCGCUCCCGCCCUUCGGGGAAUCGUGAACCGGGGUUCGGCCGGCCUACA" \
-structureDBN "((((((.....(.((((((.....))))))))....))))...." \
-colorMap "0;0;0;0;0;0;0;1;2;3;2;1;1;1;3;3;1;1;0;0;2;2;2;3;3;1;3;2;\n
1;1;1;1;0;0;0;0;1;3;3;3;1;0;0;0;0;0;0;0;1;1;3" \
-colorMapMax 3 -colorMapMin 0 -colorMapStyle green \
-o RF00065_panelC.svg -algorithm radiate -flat true -drawBases True -spaceBetweenBases 0.6

# panels D, E, F
java -cp VARNAvx-y.jar fr.orsay.lri.varna.applications.VARNAcmd \
-sequenceDBN "GGGCCCGCUCCCGCCCUUCGGGGAAUCGUGAACCGGGGUUCGGCCGGCCUACA" \
-structureDBN "((((((.....(.((((((.....))))))))....))))...." \
-chemProb "9-10:glyph=pin,dir=out,intensity=1,color=#3e844b;10-11:glyph=pin,dir=out,intensity
    =1,color=#3e844b;" \
-highlightRegion "26-27:radius=10,fill=#d2ccf4,outline=#d2ccf4;31-37:radius=10,fill=#d2ccf4,
    outline=#d2ccf4" \
-o RF00065_panelD.svg -algorithm radiate -flat true -drawBases False -spaceBetweenBases 0.6
```

5. Shell script to execute the R2R program:

```
#!/bin/tcsh -f
#
set R2RPATH =
if ( $R2RPATH == "" ) then
echo "Usage: please set the R2RPATH variable"
exit
endif

set r2r = "$R2RPATH/src/r2r"
set subfam = "perl $R2RPATH/src>SelectSubFamilyFromStockholm.pl"
set GSCparams = '3 0.97 0.9 0.75 4 0.97 0.9 0.75 0.5 0.1'

$r2r --GSC-weighted-consensus RF00065_seed.sto RF00065_seed.cons.sto $GSCparams

$subfam RF00065_seed.cons.sto IL00P75 > RF00065_seed-IL00P75.sto
$r2r --GSC-weighted-consensus RF00065_seed-IL00P75.sto RF00065_seed-IL00P75.cons.sto $GSCparams
$subfam RF00065_seed.cons.sto IL00P85 > RF00065_seed-IL00P85.sto
$r2r --GSC-weighted-consensus RF00065_seed-IL00P85.sto RF00065_seed-IL00P85.cons.sto $GSCparams

setenv R2R_DUMP_FILE RF00065_seed.txt
$r2r RF00065_seed.meta RF00065_seed.pdf
$r2r RF00065_seed.meta RF00065_seed.svg

$r2r --GSC-weighted-consensus RF00065_guide_target.sto RF00065_guide_target.cons.sto $GSCparams

$subfam RF00065_guide_target.cons.sto hybrid > RF00065_guide_target-hybrid.sto
$r2r --GSC-weighted-consensus RF00065_guide_target-hybrid.sto RF00065_guide_target-hybrid.cons.sto $GSCparams

setenv R2R_DUMP_FILE RF00065_guide_target.txt
$r2r RF00065_guide_target.meta RF00065_guide_target.pdf
$r2r RF00065_guide_target.meta RF00065_guide_target.svg
```



```

#=GF SUBFAM_PERL_PRED hybrid return 1;
#=GF SUBFAM_hybrid_R2R subst_ss _1 primary outline_only_bp
#=GF SUBFAM_hybrid_R2R box_nuc b rgb:200,0,0
#=GF SUBFAM_hybrid_R2R box_nuc f rgb:200,0,0
#=GF SUBFAM_hybrid_R2R box_nuc g rgb:200,0,0
#=GF SUBFAM_hybrid_R2R box_nuc c rgb:200,0,0
#=GF SUBFAM_hybrid_R2R outline_nuc E
#=GF SUBFAM_hybrid_R2R outline_nuc D
#=GF SUBFAM_hybrid_R2R outline_nuc m
#=GF SUBFAM_hybrid_R2R outline_nuc n
#=GF SUBFAM_hybrid_R2R outline_nuc i
#=GF SUBFAM_hybrid_R2R set_dir pos0 -90
#=GF SUBFAM_hybrid_R2R place_explicit b b-- +45 3 0 0 0 0
#=GF SUBFAM_hybrid_R2R place_explicit g g-- +90 1.5 0 0 0 90
#=GF SUBFAM_hybrid_R2R place_explicit c c-- 90 1 0 0 0 0
#=GF SUBFAM_hybrid_R2R place_explicit h h-- +45 3 0 0 0 90
#=GF SUBFAM_hybrid_R2R place_explicit m m-- 0 1 0 0 0 -90
#=GF SUBFAM_hybrid_R2R place_explicit o o-- 0 2.5 0 0 0 90
#=GF SUBFAM_hybrid_R2R tick_label i target
#=GF SUBFAM_hybrid_R2R tick_label a guide
#=GF SUBFAM_hybrid_R2R shade_along_backbone A rgb:193,255,193
#=GF SUBFAM_hybrid_R2R shade_along_backbone a rgb:193,255,193
#=GF SUBFAM_hybrid_R2R shade_along_backbone E rgb:193,255,193
#=GF SUBFAM_hybrid_R2R shade_along_backbone n rgb:193,255,193
#=GF SUBFAM_hybrid_R2R shade_along_backbone D rgb:255,228,196
#=GF SUBFAM_hybrid_R2R shade_along_backbone d rgb:255,228,196
#=GF SUBFAM_hybrid_R2R shade_along_backbone h rgb:255,228,196
#=GF SUBFAM_hybrid_R2R shade_along_backbone H rgb:255,228,196

#=GF R2R_oneseq Pa21-S892 shade_along_backbone KLOOP:K rgb:0,129,255
#=GF R2R_oneseq Pa21-S892 shade_along_backbone ILOOP:P rgb:193,255,193
#=GF R2R_oneseq Pa21-S892 shade_along_backbone ILOOP:Q rgb:193,255,193
#=GF R2R_oneseq Pa21-S892 shade_along_backbone ILOOP:X rgb:193,255,193
#=GF R2R_oneseq Pa21-S892 shade_along_backbone ILOOP:Y rgb:255,228,196
#=GF R2R_oneseq Pa21-S892 shade_along_backbone ANA:C rgb:200,200,200

//
```

7. Stockholm seed alignment of the snoR9 family (RFAM ID: RF00065) decorated with R2R labels to generate a series of consensus and secondary structures (*see* Fig. 17 for the graphical rendering).

```

# STOCKHOLM 1.0
#=GF ID      snoR9
#=GF AC      RF00065
#=GF DE      Small nucleolar RNA snoR9
#=GF AU      Bateman A, Daub J
#=GF GA      50.0
#=GF NC      49.8
#=GF TC      68.7
#=GF SE      Bateman A
#=GF SS      Published; PMID:12032319
#=GF TP      Gene; snRNA; snoRNA; CD-box;
#=GF BM      cmbuild -F CM SEED; cmcalibrate --mpi -s 1 CM
#=GF BM      cmsearch -Z 274931 -E 1000000 --toponly CM SEQDB
#=GF DR      SO:0000593 SO:C_D_box_snoRNA
#=GF DR      GO:0006396 GO:RNA processing
#=GF DR      GO:0005730 GO:nucleolus
#=GF RN      [1]
#=GF RM      12032319
#=GF RT      Noncoding RNA genes identified in AT-rich hyperthermophiles.
#=GF RA      Klein RJ, Misulovin Z, Eddy SR;
#=GF RL      Proc Natl Acad Sci U S A 2002;99:7542-7547.
#=GF CC      snoRNA R9 is a member of the C/D class of snoRNA which contain
#=GF CC      the C (UGAUGA) and D (CUGA) box motifs. R9 was identified in a
#=GF CC      computational screen in AT-rich hyperthermophiles [1]. R9 was
#=GF CC      found to overlap with the smaller snoRNA R19 which is currently a
#=GF CC      member of Pyrococcus C/D box snoRNA family Rfam:RF00095.
#=GF WK      http://en.wikipedia.org/wiki/Small_nucleolar_RNA_snoR9
#=GF SQ      5

#=GS Pyrococcus_furirosus    AC      AE009950.1/163991-163864
#=GS Pyrococcus_abysii_GE   AC      AJ248283.1/230575-230449
#=GS Pyrococcus_horikoshi   AC      BA000001.2/215834-215709
#=GS P.furirosus           AC      AF468960.1/1-128
#=GS Thermococcus_kodakar   AC      AP006878.1/47908-47779

Pyrococcus_furirosus          GGGCCCGGUU.
CCCGCCCCUCUCGGGGAAUCGUGAACCGGGGUUCCGACCGGGGCCACA..
AUGGGAUGAUGACCUUUUGCUUUACUGAACACAUAGAUGACCACGCCCUUCGCUGAC.CUAAAUAUUGAC
Pyrococcus_abysii_GE          GGGCCCGGUU.
CCCGCCCCUCUCGGGGAAUCGUGAACCGGGGUUCCGACCGGGGCCACA..
UUAUGAUGAACUUUUGCUUUUGCUGAUGUGGUGAUGAGCACGCCCUUCGCUGAUACUCUCUGGUCCAU
Pyrococcus_horikoshi          GGGCCCGGUU.
CCCGCCCCUCUCGGGGAAUCGUGAACCGGGGUUCCGACCGGGGCCACA..
GGAUGAAGAGACCUUUUGCUUUGCUGAGCAGAACAUAGAUGACCACGCCCUUCGCUGAC.CU.GCUAAUUGAC
P.furirosus                  GGGCCCGGUU.
CCCGCCCCUCUCGGGGAAUCGUGAACCGGGGUUCCGACCGGGGCCACA..
AUGGGAUGAUGACCUUUUGCUUUACUGAACACAUAGAUGACCACGCCCUUCGCUGAC.CUAAAUAUUGAC
Thermococcus_kodakar          GGGCCUGGGGUUCCGGCCUCCCAGGGGAAACGUGAACCGGGGUUCCUGCCAGGCCUACACCGGGGAUGAAGAGCUUUUGCUUUGCUGAC
..UGGAUGAGCACGCCCUUCACUGACCCGUACAGCUCU
#=GC SS_cons                 <<<<<<.....<.<<<<<<....>>>>>>>....>>>>>>..
.....  

#=GC RF                      gGGCCCGGUU.
CCCGCCCCUCUCGGGGAAUCGUGAACCGGGGUUCCGACCGGGGCCACA..
auguuAUGAUGAACUUUUGCUUUACUGAagagaUGAUGAgCACGCCCUUCGCUGAU.CUaaaUauUugAu
#=GR Pyrococcus_furirosus DEL_COLS ..  

.....  

#=GR Pyrococcus_abysii_GE DEL_COLS ..  

.....  

#=GR Pyrococcus_horikoshi DEL_COLS ..  

.....  

#=GR P.furirosus DEL_COLS ..  

.....  

#=GR Thermococcus_kodakar DEL_COLS ..  

.....  

#=GC R2R_LABEL               p.....1.....2.....3..4...QQQQqQQQ5...6RRRRrRRRs..
.7.....8  

#=GC R2R_XLABEL_KLOOP        .....KKKKKKKK.....  

.....  

#=GC R2R_XLABEL_ILOOP        .....k.....JJJJJ.....  

.....  

#=GC R2R_XLABEL_ILOOPL       .....i.....j.....  

.....  

#=GC R2R_XLABEL_ANA          CCC.....  

.....  

#=GC R2R_XLABEL_ANAL         C.....  

.....
```

```

#=GC SUBFAM_LABEL_TERM .....x.....  

.....  

#=GC SUBFAM_ILOOP75_R2R_LABEL -----<.....B>-----<.KLMNO.>-----  

-----  

#=GC SUBFAM_ILOOP85_R2R_LABEL -----<.....B>-----<.KLMNO.>-----  

-----  

#=GF R2R keep p s  

#=GF R2R var_backbone_range 1 2  

#=GF R2R var_backbone_range 3 4  

#=GF R2R var_backbone_range 5 6  

#=GF R2R var_backbone_range 7 8  

#=GF R2R shade_along_backbone ILOOP:I rgb:0,255,0  

#=GF R2R shade_along_backbone ILOOP:J rgb:0,255,0  

#=GF R2R shade_along_backbone KLOOP:K rgb:0,129,255  

#=GF R2R shade_along_backbone ANA:C rgb:200,200,200  

#=GF R2R tick_label ANAL:c ANA box  

#=GF R2R tick_label KLOOPL:k K-Loop  

#=GF SUBFAM_REGEX_PRED ILOOP75 TERM [ . ]  

#=GF SUBFAM_REGEX_PRED ILOOP85 TERM [A-Z]  

#=GF SUBFAM_ILOOP75_R2R no5  

#=GF SUBFAM_ILOOP75_R2R shade_along_backbone ILOOP:I rgb:0,255,0  

#=GF SUBFAM_ILOOP75_R2R tick_label ILOOPL:i 7 nt  

#=GF SUBFAM_ILOOP75_R2R outline_nuc ILOOP:I  

#=GF SUBFAM_ILOOP75_R2R var_backbone_range_size_fake_nucs 1 M M 5 nt  

#=GF SUBFAM_ILOOP75_R2R var_backbone_range_size_fake_nucs 1 B B  

#=GF SUBFAM_ILOOP75_R2R var_backbone_range_size_fake_nucs 1 K K  

#=GF SUBFAM_ILOOP75_R2R var_backbone_range_size_fake_nucs 1 L L  

#=GF SUBFAM_ILOOP75_R2R var_backbone_range_size_fake_nucs 1 N N  

#=GF SUBFAM_ILOOP75_R2R var_backbone_range_size_fake_nucs 1 O O  

#=GF SUBFAM_ILOOP85_R2R no5  

#=GF SUBFAM_ILOOP85_R2R shade_along_backbone ILOOP:I rgb:0,255,0  

#=GF SUBFAM_ILOOP85_R2R tick_label ILOOPL:i 8 nt  

#=GF SUBFAM_ILOOP85_R2R outline_nuc ILOOP:I  

#=GF SUBFAM_ILOOP85_R2R var_backbone_range_size_fake_nucs 1 M M 5 nt  

#=GF SUBFAM_ILOOP85_R2R var_backbone_range_size_fake_nucs 1 B B  

#=GF SUBFAM_ILOOP85_R2R var_backbone_range_size_fake_nucs 1 K K  

#=GF SUBFAM_ILOOP85_R2R var_backbone_range_size_fake_nucs 1 L L  

#=GF SUBFAM_ILOOP85_R2R var_backbone_range_size_fake_nucs 1 N N  

#=GF SUBFAM_ILOOP85_R2R var_backbone_range_size_fake_nucs 1 O O  

#=GF R2R_oneseq Pyrococcus_furiosus shade_along_backbone ILOOP:I rgb:0,255,0  

#=GF R2R_oneseq Pyrococcus_furiosus tick_label ILOOPL:i 5' guide sequence  

#=GF R2R_oneseq Pyrococcus_furiosus shade_along_backbone ILOOP:J rgb:0,255,0  

#=GF R2R_oneseq Pyrococcus_furiosus tick_label ILOOPL:j 3' guide sequence  

#=GF R2R_oneseq Pyrococcus_furiosus shade_along_backbone KLOOP:K rgb:0,129,255  

#=GF R2R_oneseq Pyrococcus_furiosus var_backbone_range_size_fake_nucs 10 7 8  

#=GF R2R_oneseq Pyrococcus_furiosus inline_nuc ILOOP:I  

#=GF R2R_oneseq Pyrococcus_furiosus inline_nuc ILOOP:J  

#=GF R2R_oneseq Pyrococcus_furiosus inline_nuc KLOOP:K  

#=GF R2R_oneseq Pyrococcus_furiosus outline_nuc Q  

#=GF R2R_oneseq Pyrococcus_furiosus outline_nuc q  

#=GF R2R_oneseq Pyrococcus_furiosus outline_nuc R  

#=GF R2R_oneseq Pyrococcus_furiosus outline_nuc r  

#=GF R2R_oneseq Pyrococcus_furiosus outline_nuc S  

#=GF R2R_oneseq Pyrococcus_furiosus tick_label q 8 bp  

#=GF R2R_oneseq Pyrococcus_furiosus tick_label r 9 bp  

#=GF R2R_oneseq Pyrococcus_abyssi_GE shade_along_backbone ILOOP:I rgb:0,255,0  

#=GF R2R_oneseq Pyrococcus_abyssi_GE tick_label ILOOPL:i 5' guide sequence  

#=GF R2R_oneseq Pyrococcus_abyssi_GE shade_along_backbone ILOOP:J rgb:0,255,0  

#=GF R2R_oneseq Pyrococcus_abyssi_GE tick_label ILOOPL:j 3' guide sequence  

#=GF R2R_oneseq Pyrococcus_abyssi_GE shade_along_backbone KLOOP:K rgb:0,129,255  

#=GF R2R_oneseq Pyrococcus_abyssi_GE var_backbone_range_size_fake_nucs 10 7 8  

#=GF R2R_oneseq Pyrococcus_abyssi_GE inline_nuc ILOOP:I  

#=GF R2R_oneseq Pyrococcus_abyssi_GE inline_nuc ILOOP:J  

#=GF R2R_oneseq Pyrococcus_abyssi_GE inline_nuc KLOOP:K  

#=GF R2R_oneseq Pyrococcus_abyssi_GE outline_nuc Q  

#=GF R2R_oneseq Pyrococcus_abyssi_GE outline_nuc q  

#=GF R2R_oneseq Pyrococcus_abyssi_GE outline_nuc R  

#=GF R2R_oneseq Pyrococcus_abyssi_GE outline_nuc r  

#=GF R2R_oneseq Pyrococcus_abyssi_GE outline_nuc S  

#=GF R2R_oneseq Pyrococcus_abyssi_GE tick_label q 8 bp  

#=GF R2R_oneseq Pyrococcus_abyssi_GE tick_label r 9 bp

```

```

##=GF R2R_oneseq Pyrococcus_horikoshi shade_along_backbone ILOOP:I rgb:0,255,0
##=GF R2R_oneseq Pyrococcus_horikoshi tick_label ILOOPL:i 5' guide sequence
##=GF R2R_oneseq Pyrococcus_horikoshi shade_along_backbone ILOOP:J rgb:0,255,0
##=GF R2R_oneseq Pyrococcus_horikoshi tick_label ILOOPL:j 3' guide sequence
##=GF R2R_oneseq Pyrococcus_horikoshi shade_along_backbone KLOOP:K rgb:0,129,255
##=GF R2R_oneseq Pyrococcus_horikoshi var_backbone_range_size_fake_nucs 10 7 8
##=GF R2R_oneseq Pyrococcus_horikoshi inline_nuc ILOOP:I
##=GF R2R_oneseq Pyrococcus_horikoshi inline_nuc ILOOP:J
##=GF R2R_oneseq Pyrococcus_horikoshi inline_nuc KLOOP:K
##=GF R2R_oneseq Pyrococcus_horikoshi outline_nuc Q
##=GF R2R_oneseq Pyrococcus_horikoshi outline_nuc q
##=GF R2R_oneseq Pyrococcus_horikoshi outline_nuc R
##=GF R2R_oneseq Pyrococcus_horikoshi outline_nuc r
##=GF R2R_oneseq Pyrococcus_horikoshi outline_nuc s
##=GF R2R_oneseq Pyrococcus_horikoshi tick_label q 8 bp
##=GF R2R_oneseq Pyrococcus_horikoshi tick_label r 9 bp

##=GF R2R_oneseq P.furiosus shade_along_backbone ILOOP:I rgb:0,255,0
##=GF R2R_oneseq P.furiosus tick_label ILOOPL:i 5' guide sequence
##=GF R2R_oneseq P.furiosus shade_along_backbone ILOOP:J rgb:0,255,0
##=GF R2R_oneseq P.furiosus tick_label ILOOPL:j 3' guide sequence
##=GF R2R_oneseq P.furiosus shade_along_backbone KLOOP:K rgb:0,129,255
##=GF R2R_oneseq P.furiosus var_backbone_range_size_fake_nucs 10 7 8
##=GF R2R_oneseq P.furiosus inline_nuc ILOOP:I
##=GF R2R_oneseq P.furiosus inline_nuc ILOOP:J
##=GF R2R_oneseq P.furiosus inline_nuc KLOOP:K
##=GF R2R_oneseq P.furiosus outline_nuc Q
##=GF R2R_oneseq P.furiosus outline_nuc q
##=GF R2R_oneseq P.furiosus outline_nuc R
##=GF R2R_oneseq P.furiosus outline_nuc r
##=GF R2R_oneseq P.furiosus outline_nuc s
##=GF R2R_oneseq P.furiosus tick_label q 8 bp
##=GF R2R_oneseq P.furiosus tick_label r 9 bp

##=GF R2R_oneseq Thermococcus_kodakar shade_along_backbone ILOOP:I rgb:0,255,0
##=GF R2R_oneseq Thermococcus_kodakar tick_label ILOOPL:i 5' guide sequence
##=GF R2R_oneseq Thermococcus_kodakar shade_along_backbone ILOOP:J rgb:0,255,0
##=GF R2R_oneseq Thermococcus_kodakar tick_label ILOOPL:j 3' guide sequence
##=GF R2R_oneseq Thermococcus_kodakar shade_along_backbone KLOOP:K rgb:0,129,255
##=GF R2R_oneseq Thermococcus_kodakar var_backbone_range_size_fake_nucs 10 7 8
##=GF R2R_oneseq Thermococcus_kodakar inline_nuc ILOOP:I
##=GF R2R_oneseq Thermococcus_kodakar inline_nuc ILOOP:J
##=GF R2R_oneseq Thermococcus_kodakar inline_nuc KLOOP:K
##=GF R2R_oneseq Thermococcus_kodakar outline_nuc Q
##=GF R2R_oneseq Thermococcus_kodakar outline_nuc q
##=GF R2R_oneseq Thermococcus_kodakar outline_nuc R
##=GF R2R_oneseq Thermococcus_kodakar outline_nuc r
##=GF R2R_oneseq Thermococcus_kodakar outline_nuc s
##=GF R2R_oneseq Thermococcus_kodakar tick_label q 8 bp
##=GF R2R_oneseq Thermococcus_kodakar tick_label r 9 bp

##=GF Makefile_skeleton-with-pairbonds
//
```

8. R2R meta files including the instructions to process the input files associated with the different representations generated in Figs. 16 and 17.

```

RF00065_guide_target.cons.sto
RF00065_guide_target.sto      oneseq    Pa21-S892
RF00065_guide_target-hybrid.cons.sto

```

```

RF00065_seed.cons.sto
RF00065_seed.sto      oneseq    Pyrococcus_furiosus
RF00065_seed.sto      oneseq    Pyrococcus_abysse GE
RF00065_seed.sto      oneseq    Pyrococcus_horikoshi
RF00065_seed.sto      oneseq    Thermococcus_kodakar
RF00065_seed.cons.sto  skeleton-with-pairbonds
RF00065_seed-ILOOP75.cons.sto
RF00065_seed-ILOOP85.cons.sto

```

Acknowledgements

The author wishes to thank the French *Centre National de la Recherche Scientifique* (CNRS) for its continued support, the NSF-funded RNA Ontology Consortium, and Jim Procter for stimulating discussions.

References

1. Andronescu M, Bereg V, Hoos H, Condon A (2008) RNA STRAND: the RNA secondary structure and statistical analysis database. *BMC Bioinform* 9(1):340. doi:10.1186/1471-2105-9-340
2. Bateman A, Agrawal S, Birney E, Bruford EA, Bujnicki JM, Cochrane G, Cole JR, Dinger ME, Enright AJ, Gardner PP, Gautheret D, Griffiths-Jones S, Harrow J, Herrero J, Holmes IH, Huang H-D, Kelly KA, Kersey P, Kozomara A, Lowe TM, Marz M, Moxon S, Pruitt KD, Samuelsson T, Stadler PF, Vilella AJ, Vogel J-H, Williams KP, Wright MW, Zwieb C (2011) RNACentral: a vision for an international database of RNA sequences. *RNA* 17(11):1941–1946. doi:10.1261/rna.2750811
3. Berman HM, Olson WK, Beveridge DL, Westbrook J, Gelbin A, Demeny T, Hsieh SH, Srinivasan AR, Schneider B (1992) The nucleic acid database. A comprehensive relational database of three-dimensional structures of nucleic acids. *Biophys J* 63(3):751–759. doi:10.1016/S0006-3495(92)81649-1
4. Bruccoleri RE, Heinrich G (1988) An improved algorithm for nucleic acid secondary structure display. *Comput Appl Biosci* 4(1):167–173. doi:10.1093/bioinformatics/4.1.167
5. Byun Y, Han K (2009) PseudoViewer3: generating planar drawings of large-scale RNA structures with pseudoknots. *Bioinformatics* 25(11):1435–1437. doi:10.1093/bioinformatics/btp252
6. Cannone JJ, Subramanian S, Schnare MN, Collett JR, D’Souza LM, Du Y, Feng B, Lin N, Madabusi LV, Muller KM, Pande N, Shang Z, Yu N, Gutell RR (2002) The comparative RNA web (CRW) site: an online database of comparative sequence and structure information for ribosomal, intron, and other RNAs. *BMC Bioinform* 3(2). doi: 10.1186/1471-2105-3-15
7. Darty K, Denise A, Ponty Y (2009) VARNA: interactive drawing and editing of the RNA secondary structure. *Bioinformatics* 25(15):1974–1975. doi:10.1093/bioinformatics/btp250
8. Fourmann J-B, Tillault A-S, Blaud M, Leclerc F, Branlant C, Charpentier B (2013) Comparative study of RNA conformational dynamics during assembly of two box H/ACA ribonucleoprotein pseudouridine-synthases revealing uncorrelated efficiencies in assembly and activity. *PLoS ONE* 8(7):e70313. doi:10.1371/journal.pone.0070313
9. Fourmy D, Yoshizawa S (2012) Protein-RNA footprinting: an evolving tool. *Wiley Interdiscip Rev RNA* 3(4):557–566. doi:10.1002/wrna.1119
10. Gansner ER, North SC (2000) An open graph visualization system and its applications to software engineering. *Softw – Pract Exp* 30(11):1203–1233. doi:10.1002/1097-024X(200009)30:11<1203::AID-SPE338>3.3.CO;2-E
11. Gansner ER, Koutsofios E, North SC, Vo KP (1993) A technique for drawing directed graphs. *IEEE Trans Softw Eng* 19(3):214–230. doi:10.1109/32.221135
12. Griffiths-Jones S, Bateman A, Marshall M, Khanna A, Eddy SR (2003) Rfam: an RNA family database. *Nucl Acids Res* 31(1):439–441. doi: 10.1093/nar/gkg006
13. Gruber AR, Lorenz R, Bernhart SH, Neuböck R, Hofacker IL (2008) The Vienna RNA websuite. *Nucl Acids Res* 36(Web Server issue):W70–W74. doi:10.1093/nar/gkn188
14. Hofacker IL, Fontana W, Stadler PF, Bonhoeffer LS, Tacker M, Schuster P (1994) Fast folding and comparison of RNA secondary structures. *Monatshefte für Chemie/Chem Mon* 125(2):167–188. doi:10.1007/BF00818163
15. Jossinet F, Westhof E (2005) Sequence to structure (S2S): display, manipulate and interconnect RNA data from sequence to structure. *Bioinformatics* 21(15):3320–3321. doi:10.1093/bioinformatics/bti504
16. Jossinet F, Ludwig TE, Westhof E (2010) Assemble: an interactive graphical tool to analyze and build RNA architectures at the 2D and 3D levels. *Bioinformatics* 26(16):2057–2059. doi:10.1093/bioinformatics/btq321

17. Kaiser A, Krüger J, Evers DJ (2007) RNA movies 2: sequential animation of RNA secondary structures. *Nucl Acids Res* 35(Web Server issue):W330–W334. doi:[10.1093/nar/gkm309](https://doi.org/10.1093/nar/gkm309)
18. Lai D, Proctor JR, Zhu JYA, Meyer IM (2012) R-CHIE: a web server and R package for visualizing RNA secondary structures. *Nucl Acids Res* 40(12):e95. doi:[10.1093/nar/gks241](https://doi.org/10.1093/nar/gks241)
19. Lemay J-F, Lafontaine DA (2007) Core requirements of the adenine riboswitch aptamer for ligand binding. *RNA* 13(3):339–350. doi:[10.1261/rna.142007](https://doi.org/10.1261/rna.142007)
20. Leonard CW, Hajdin CE, Karabiber F, Mathews DH, Favorov OV, Dokholyan NV, Weeks KM (2013) Principles for understanding the accuracy of SHAPE-directed RNA structure modeling. *Biochemistry* 52(4):588–595. doi:[10.1021/bi300755u](https://doi.org/10.1021/bi300755u)
21. Leontis NB, Altman RB, Berman HM, Brenner SE, Brown JW, Engelke DR, Harvey SC, Holbrook SR, Jossinet F, Lewis SE, Major F, Mathews DH, Richardson JS, Williamson JR, Westhof E (2006) The RNA ontology consortium: an open invitation to the RNA community. *RNA* 12(4):533–541. doi:[10.1261/rna.2343206](https://doi.org/10.1261/rna.2343206)
22. Mathews DH, Sabina J, Zuker M, Turner DH (1999) Expanded sequence dependence of thermodynamic parameters improves prediction of RNA secondary structure. *J Mol Biol* 288:911–940. doi:[10.1006/jmbi.1999.2700](https://doi.org/10.1006/jmbi.1999.2700)
23. Menzel P, Seemann SE, Gorodkin J (2012) RILogo: visualizing RNA–RNA interactions. *Bioinformatics* 28(19):2523–2526. doi:[10.1093/bioinformatics/bts461](https://doi.org/10.1093/bioinformatics/bts461)
24. Mortimer SA, Trapnell C, Aviran S, Pachter L, Lucks JB (2012) SHAPE-Seq: high-throughput RNA structure analysis. *Curr Protoc Chem Biol* 4(4):275–297. doi:[10.1002/9780470559277.ch120019](https://doi.org/10.1002/9780470559277.ch120019)
25. Muller G, Gaspin C, Etienne A, Westhof E (1993) Automatic display of RNA secondary structures. *Comput Appl Biosci* 9(5):551–561. doi:[10.1093/bioinformatics/9.5.551](https://doi.org/10.1093/bioinformatics/9.5.551)
26. Ouyang Z, Snyder MP, Chang HY (2012) SeqFold: genome-scale reconstruction of RNA secondary structure integrating high-throughput sequencing data. *Genome Res.* doi:[10.1101/gr.138545.112](https://doi.org/10.1101/gr.138545.112)
27. Rijk PD, Wuyts J, Wachter RD (2003) RnaViz 2: an improved representation of RNA second-
ary structure. *Bioinformatics* 19(2):299–300. doi:[10.1093/bioinformatics/19.2.299](https://doi.org/10.1093/bioinformatics/19.2.299)
28. Schlatterer JC, Brenowitz M (2009) Complementing global measures of RNA folding with local reports of backbone solvent accessibility by time resolved hydroxyl radical footprinting. *Methods* 49(2):142–147. doi:[10.1016/j.ymeth.2009.04.019](https://doi.org/10.1016/j.ymeth.2009.04.019)
29. Shabash B, Wiese KC, Glen E (2012) Improving the portability and performance of jViz.RNA – a dynamic RNA visualization software. In: PRIB, pp 82–93. doi:[10.1007/978-3-642-34123-6_8](https://doi.org/10.1007/978-3-642-34123-6_8)
30. Waugh A, Gendron P, Altman R, Brown JW, Case D, Gautheret D, Harvey SC, Leontis N, Westbrook J, Westhof E, Zuker M, Major F (2002) RNAML: a standard syntax for exchanging RNA information. *RNA* 8(6):707–717. doi:[10.1017/S1355838202028017](https://doi.org/10.1017/S1355838202028017)
31. Weinberg Z, Breaker RR (2011) R2R-software to speed the depiction of aesthetic consensus RNA secondary structures. *BMC Bioinform* 12:3. doi:[10.1186/1471-2105-12-3](https://doi.org/10.1186/1471-2105-12-3)
32. Wiese KC, Glen E, Vasudevan A (2005) JViz. Rna—a Java tool for RNA secondary structure visualization. *IEEE Trans Nanobiosci* 4(3):212–218. doi:[10.1109/TNB.2005.853646](https://doi.org/10.1109/TNB.2005.853646)
33. Xu Z, Culver GM (2009) Chemical probing of RNA and RNA/protein complexes. *Methods Enzymol* 468:147–165. doi:[10.1016/S0076-6879\(09\)68008-3](https://doi.org/10.1016/S0076-6879(09)68008-3)
34. Yang H, Jossinet F, Leontis N, Chen L, Westbrook J, Berman HM, Westhof E (2003) Tools for the automatic identification and classification of RNA base pairs. *Nucl Acids Res* 31(13):3450–3460. doi:[10.1093/nar/gkg529](https://doi.org/10.1093/nar/gkg529)
35. Zarringhalam K, Meyer MM, Dotu I, Chuang JH, Clote P (2012) Integrating chemical footprinting data into RNA secondary structure prediction. *PLoS ONE* 7(10):e45160. doi:[10.1371/journal.pone.0045160](https://doi.org/10.1371/journal.pone.0045160)
36. Zhang C, Darnell RB (2011) Mapping in vivo protein-RNA interactions at single-nucleotide resolution from HITS-CLIP data. *Nat Biotechnol* 29(7):607–614. doi:[10.1038/nbt.1873](https://doi.org/10.1038/nbt.1873)
37. Zuker M, Stiegler P (1981) Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. *Nucl Acids Res* 9:133–148. doi:[10.1093/nar/9.1.133](https://doi.org/10.1093/nar/9.1.133)

Chapter 6

Modeling and Predicting RNA Three-Dimensional Structures

Jérôme Waldspühl and Vladimir Reinharz

Abstract

Modeling the three-dimensional structure of RNAs is a milestone toward better understanding and prediction of nucleic acids molecular functions. Physics-based approaches and molecular dynamics simulations are not tractable on large molecules with all-atom models. To address this issue, coarse-grained models of RNA three-dimensional structures have been developed. In this chapter, we describe a graphical modeling based on the Leontis–Westhof extended base-pair classification. This representation of RNA structures enables us to identify highly conserved structural motifs with complex nucleotide interactions in structure databases. Further, we show how to take advantage of this knowledge to quickly and simply predict three-dimensional structures of large RNA molecules.

Key words Tertiary structure, RNA motifs, Extended secondary structure, Base-pair classification, Modeling, Prediction

1 Introduction

RNAs perform a broad range of catalytic and regulatory functions in cells, which often use the folding properties of these molecules. RNA structures are typically described at two levels of organization: the secondary structure, which enumerates the Watson–Crick and Wobble base pairs that create the backbone of the molecular structure; and the tertiary structure, which indicates the positions of each atom in the molecule.

The study of secondary structures and base-pairing properties can reveal fundamental insights into the functional mechanisms of RNAs such as frameshift elements [1] and riboswitches [2]. However, the information provided by this representation is sometimes not sufficient to describe the complexity of intermolecular and intramolecular interactions that govern RNA functions. A seminal example is the sarcin–ricin factor-binding loop, which possesses a sophisticated tertiary structure [3].

Classical approaches to predict and analyze three-dimensional molecular structures make use of molecular dynamic simulations [4]. This methodology has the potential to perform calculations on

all-atom models, but suffers from high computational complexity. Straightforward applications are thus limited to small molecules (approximately 50 nt.) on a short period of time with very small body motion. Coarse-grained modeling of RNA structures enables us to overcome some of these limitations [5–8], but the practical impact and theoretical horizon of these technologies remain unchanged. In addition, molecular dynamic simulations often require fine-tuning and can be challenging to run and interpret for nonexperts.

Recently, several research groups developed alternate strategies to model and predict RNA three-dimensional structures. One of the most successful approach, applied in the MC-Fold|MC-Sym pipeline [9], RNA2D3D [10] or 3dRNA [11], consists in predicting a secondary structure first, and use it to build a three-dimensional model. Other programs have employed fragment-assembly [12, 13] or conditional random fields techniques [14]. It is worth noting that, as we see in a previous chapter, comparative modeling techniques can also be applied to RNA molecules if one structural homolog has been already identified [15].

In this chapter, we introduce a versatile model for RNA structures, which enables us to describe essential features and fine-grained details of RNA three-dimensional structures while preserving the complexity of the representation to a minimum. This methodological advance is key to large-scale analysis and to better understanding of critical features of RNA molecular structures. In turn, this knowledge enables us to define new computational techniques to quickly and easily predict three-dimensional structures of large RNA molecules.

First, we describe the base-pair classification at the base of this model [16], and present rnaview [17]—a program that annotates automatically RNA three-dimensional structures. Then, we introduce the concept of RNA structural motifs and present recent databases and online resources based on this definition [18, 19]. Finally, show how to use this knowledge to predict RNA three-dimensional structures. The pipeline described in this chapter works in two steps. First, we predict secondary structures with RNAsubopt [20] and expand the prediction to a full base-pairing interaction network with RNA-MoIP [21]. Next, we use this graphical representation to build the three-dimensional structure of the RNA with MC-Sym [9].

2 Material

2.1 Sequence and Structure Data

The Nucleic Acids Database [22] is a repository of experimentally determined three-dimensional RNA structures maintained at Rutgers University, which is available at <http://ndbserver.rutgers.edu>. This is a specialized version of the Protein Data Bank [23],

which is available at <http://www.rcsb.org/pdb>. The structures are typically stored in files with the extension “.pdb”. In particular, the PDB files store the spatial coordinates of each atom in the molecule. Plain sequences can also be downloaded separately under the FASTA format.

In this chapter, we illustrate our methods on the sequence and experimentally determined tertiary structures of tRNA(Cys) of *Archaeoglobus fulgidus* [24]. The PDB ID of this molecule is 2DU3.

2.2 Automatic Annotation of 3D Structures

2.3 RNA Secondary Structure Prediction Tools

2.4 Insertion of RNA Motifs in Secondary Structure

2.5 Building RNA 3D Structures from a RNA Interaction Graphs

The rnaview software is used to identify all base-pairing interactions that represent in a RNA tertiary structure [17]. This program can be downloaded at: <http://ndbserver.rutgers.edu/ndbmodule/services/download/rnaview.html> and is currently available for Linux, UNIX, SUN, and MAC systems.

The prediction of RNA secondary structures is performed with the RNAfold program available in the Vienna RNA package [20, 25]. The software suite is available at <http://www.tbi.univie.ac.at/RNA/> and can run under LINUX, UNIX, MAC, and WINDOWS systems. In this chapter, we used the version 2.1.2 of the Vienna RNA package. Web services are also available at <http://rna.tbi.univie.ac.at/>.

We perform the insertion of RNA motifs inside predicted RNA secondary structures with the RNA-MoIP software [21] available at <http://csb.cs.mcgill.ca/RNA MoIP/>. Noticeably, this software requires installing the Gurobi solver (<http://www.gurobi.com/>), which is free for academic users.

We use the MC-Sym software to build tertiary structure from a base-pairing interaction network [9]. MC-Sym is part of the MC-tools package available at: <http://www.major.iric.ca/MajorLabEn/MC-Tools.html>.

3 Methods

3.1 Classification of Base-Pairing Interactions

In this section, we describe how to extract from a three-dimensional structure, the list of all base-pairing interactions. Then, we describe how to use databases of recurrent motifs to predict the tertiary structure of large RNA molecules.

Watson–Crick (C–G and A–U) and Wobble (G–U) base pairs are the most common type of interactions. They create a scaffold for the tertiary structure. Nonetheless, the analysis of RNA crystal structures revealed a diversity of base-pairing interactions that goes far beyond these canonical base pairs. In order to facilitate the description of RNA structures, Leontis and Westhof proposed a

complete nomenclature of base-pairing interactions [16] that aims to provide a better description of the complexity of the tertiary structure. Key to this model is to introduce a detailed representation of the base of nucleotides, which in turn enables us to define a more sophisticated classification of base-pairing interactions. Thereby, a base is abstracted with a right triangle modeling all edges of the molecules that can be potentially involved in an interaction with other nucleotides (*see* Fig. 1). The three interacting edges are: the Watson–Crick edge, the Hoogsteen edge, and the Sugar edge. The hypotenuse is associated with the Hoogsteen edge, and the vertex created by the Hoogsteen and Sugar edge represents the root of the base.

Base-pairing interactions between nucleotides can now occur between any of these edges. In addition, a complete description of these interactions must also specify the relative orientation of the bases (i.e., the glycosidic bond orientation). Hence, we indicate if the bases are oriented in the same or in opposite directions. These configurations are respectively named *trans* and *cis* configurations.

These parameters enable us to define a complete nomenclature of all possible base-pairing interactions between nucleotides. This catalog is shown in Fig. 1 and includes all 12 possible base pairs that are found in RNA structures.

In secondary structure diagrams, base pairs are often represented with specific links. C–G base pairs are represented with two parallel line, A–U base pairs with a single line and G–U base pairs with a circle. Leontis and Westhof also assigned a symbol to each base-pairing interaction of their classification. Watson–Crick edges are represented with a circle, Hoogsteen edges with a square and Sugar edges with a triangle. In addition, black symbols represent a *cis* orientation, and white symbols a *trans* orientation. This notation is illustrated at the bottom of each base pair in Fig. 1.

Using this nomenclature, RNA tertiary structures can be decomposed into a network of base-pairing interactions. Unlike classical secondary structures, nucleotides are now no longer restricted to interact with at most one other nucleotide. Instead, they can be involved in multiple interactions and create, for instance, base triples. Crossing interactions are also permitted. While this modeling does not obviously encompass all details of three-dimensional atomistic structures, it appears to store most of the information that one needs to reconstruct full tertiary structures [9].

3.2 Annotation of Base-Pairing Interactions from RNA 3D Structures

The rnaview program annotates automatically of all base-pairing interactions found in RNA tertiary structure using the Leontis–Westhof classification [16]. It can also produce an image of the interaction graph. The command line to run the program is:

```
rnaview -p input.pdb
```

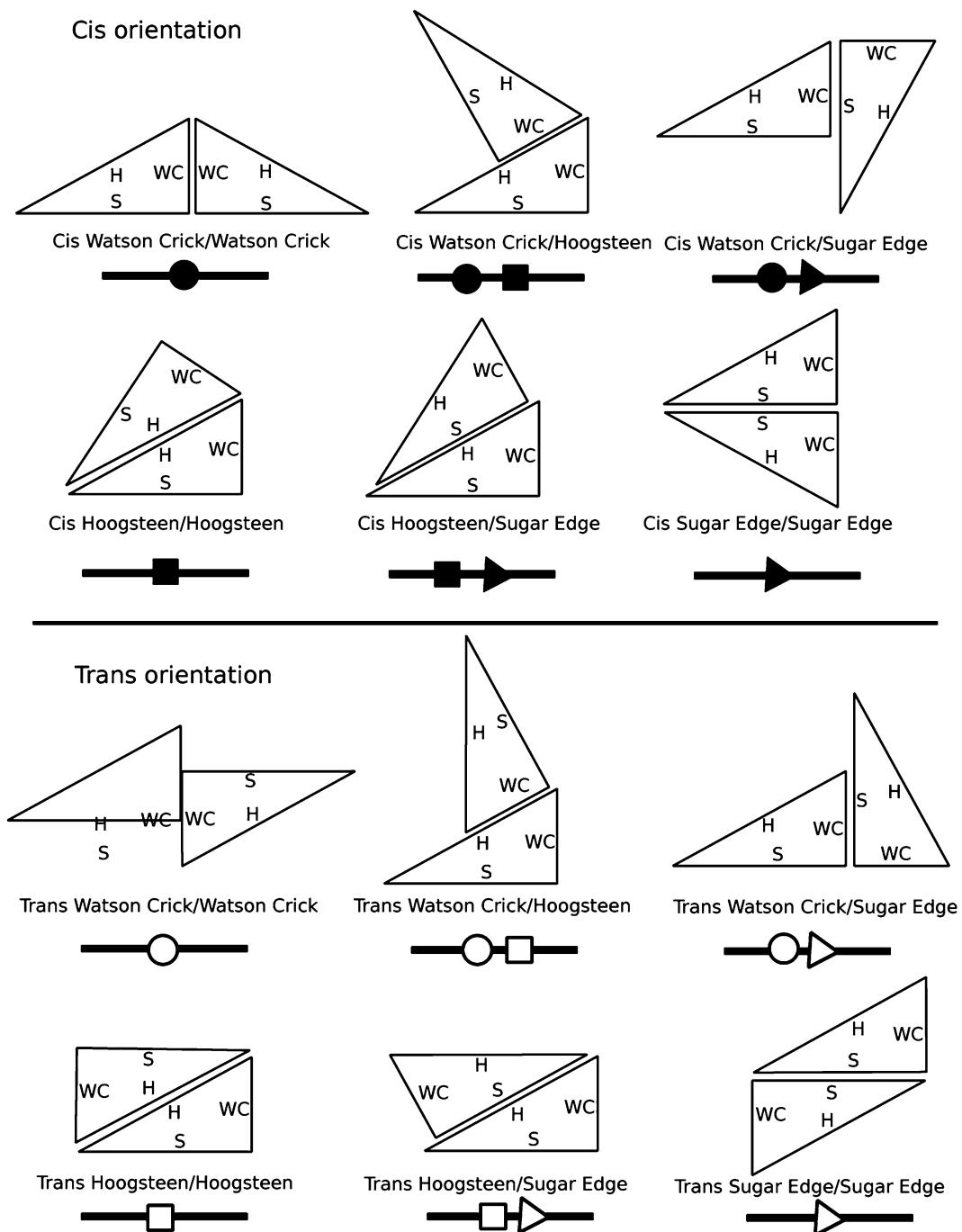


Fig. 1 Base modeling and Leontis–Westhof base-pair classification. The base of a nucleotide is represented with a *right triangle*. The hypotenuse represents the Hoogsteen edge (noted "H"), and the other sides are associated with the Watson–Crick edge (noted "W") and Sugar edge (noted "S"). This figure represents all 12 base-pair interactions with *cis* or *trans* orientation

where “input.pdb” is your input three-dimensional structure and the “-p” flag is an optional parameter that indicates to the program to create a visualization of the annotated structure. Figure 2 shows the graphical output of rnaview on the tRNA(Cys) from *Archaeoglobus fulgidus*.

The output files are stored in the same directory as the input file. The list of base-pairing interactions is stored in a new text file named after the input file and appended with the suffix “.out”. Similarly, if you used the “-p” option, rnaview also creates a postscript file with a drawing of the base-pairing interaction network.

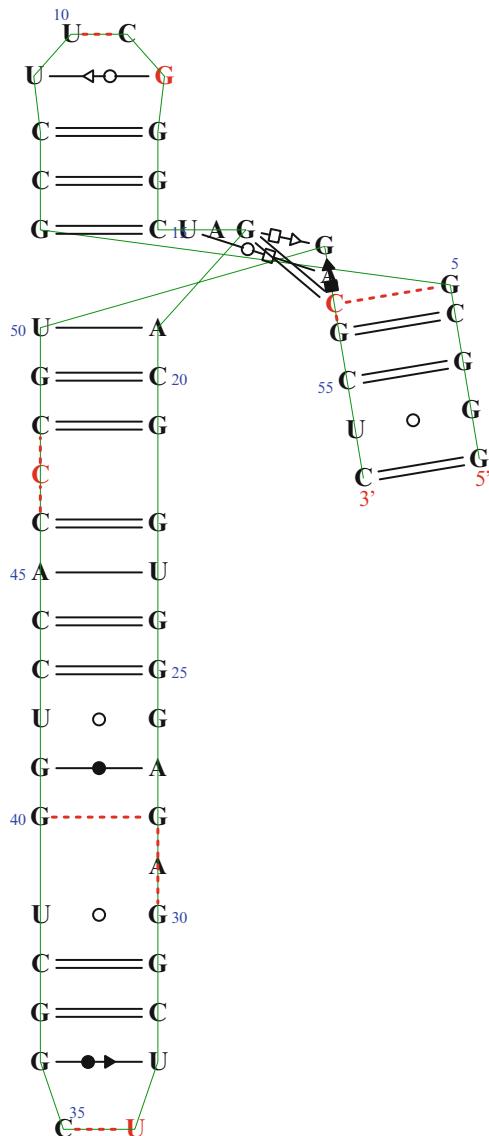


Fig. 2 rnaview annotation of a crystal structure of tRNA(Cys) from *Archaeoglobus fulgidus* (Fukunaga and Yokoyama [24])

The following code, calculated from the tRNA(Cys) from *Archaeoglobus fulgidus* (2DU3), illustrates a typical output of rnaview:

```
9_12, B: 9 U-G 12 B: S/W tran syn n/a
16_52, B: 16 U-A 52 B: W/H tran XXIV
18_51, B: 18 G-G 51 B: H/S tran n/a
18_53, B: 18 G-C 53 B: +/-cis syn XIX
```

The first column gives the indices, separated by an underscore, of the two nucleotides involved in the base pair. These indices are calculated by rnaview and correspond to their positions in the sequence(s) entered in the program. They may eventually differ from the indices stored in the PDB file, which are given in the third and fifth columns.

The letters in the second and sixth columns indicate the chain of the two nucleotides. In our case, the RNA molecule has a single chain named “B” in the PDB file. The types of the nucleotides forming the base pair are indicated in the fourth column.

Finally, the seventh and eighth columns describe the edges involved in the base pair (Watson–Crick, Hoogsteen, or Sugar edge), followed by the orientation of their interaction (*cis* or *trans*). For instance, in the example above the first row says “the nucleotide U at index 9, and the nucleotide G at position 12, form a *trans* base pair between the Sugar edge (S) of nucleotide U and Watson–Crick edge (W) of the nucleotide G.” It is worth noting that classical Watson–Crick base pairs are annotated with +/+ (C–G base pair) or -/- (A–U base pair). More details on the base-pair notation, including “marginal” interactions, can be found in refs. [17] or [26].

As expected, most base pairs represented in Fig. 2 are Watson–Crick base pairs. Nonetheless, we note that noncanonical base pairs tend to get concentrated in specific regions of the secondary structure backbone and create sophisticated local motifs. In fact, this observation is recurrent in most annotated structures. It will give rise to the notion of RNA motifs introduced in the next section.

Finally, it is worth noting that we can alternatively use the program MC-Annotate [27] to perform similar annotations. However, the classification used by MC-Annotate differs slightly from the one used in the motif databases described further.

3.3 RNA Motifs

The modeling of RNA tertiary structures into networks of base-pairing interactions revealed recurrent motifs conserved across families of structures. These structural patterns form a base toward better understanding of complex organizations of nucleotides inside hairpins, internal loops and beyond (See previous section for a definition of the secondary structure elements). Several methodologies and databases have been developed to mine and store these data. Among the most popular repositories, we count the RNA 3D

Hub maintained by the Bowling Green State University RNA group at <http://rna.bgsu.edu/rna3dhub/> [19], and the RNA 3D Motif database developed at the university of Paris-Sud and available at <http://rna3dmotif.lri.fr/> [18]. More recently, international institute for molecular and cell biology in Warsaw introduced a novel motif database RNABricks [28], available at <http://iimcb.genesisilico.pl/rnabricks/>.

Beside databases of annotated structures (i.e., the RNA Structure Atlas) and recurrent RNA 3D motifs (RNA 3D Motif Atlas), the RNA 3D Hub offers large suite of online tools and Web services. In particular, users can here retrieve local 3D motifs with WebFR3D (<http://rna.bgsu.edu/main/webapps/webfr3d/>) or align RNA tertiary structures with R3D Align (<http://rna.bgsu.edu/main/webapps/webr3dalight/>).

Each database developed its own local 3D motif format description, based on the Leontis–Westhof base-pair classification. RNA-MoIP use the format introduced with RNA3DMotif [18]. An example of an internal loop motif is provided below:

```
Bases: 26_G 27_A 28_G 29_A 30_G 39_U 40_G 41_G 42_U
       (39_U) --- C/C - --- (40_G)
       (30_G) --- 5/5 s --- (40_G)
       (30_G) --- +/+c --- (39_U)
       (40_G) --- C/C - --- (41_G)
       (29_A) --- C/C - --- (30_G)
       (28_G) --- 5/5 s --- (41_G)
       (28_G) --- C/C - --- (29_A)
       (41_G) --- C/C - --- (42_U)
       (26_G) --- +/+c --- (42_U)
       (27_A) --- C/C - --- (28_G)
       (26_G) --- C/C - --- (27_A)
```

The first line starting with the key word “Bases” indicates the nucleotides involved in this motif. For each nucleotide, we display its index and its type separated by an underscore. Each following line describes a base-pairing interaction in this motif. The syntax of a base pair is structured as follows. On both ends of the row, the two nucleotides (index and type) are shown between parentheses. Then, the type of the interaction is shown in the middle, surrounded by three dashes on each side. The first field indicates the edges involved and the second the base-pair orientation (“c” for *cis*, “t” for *trans*, and “s” for a stacking). An interaction annotated “C/C” stands for a backbone connection (i.e., consecutive nucleotides) and thus may not be considered as a base pair in our discussion. More information can be found at <http://rna3dmotif.lri.fr/help.html>.

RNA-MoIP needs to decompose these motifs into components to permit their insertion. Components are largest sequences

of contiguous nucleotides that belong to a motif. For instance, in the example above, we have two components ([26_G,27_A,28_G, 29_A,30_G] and [39_U,40_G,41_G,42_U]). In fact, the definition of components follows the classical secondary structure loop classification. Hairpins have one single component, bulges and internal loops have two components, and k-way junctions have k components. This definition has been introduced with RNA-MoIP to facilitate the description of integer programming equations.

Figure 3 illustrates the definition of motifs. In this example, we identify two motifs (a hairpin and a internal loop) inserted into a single stem. The figure shows the 3D structures and the base-pairing interaction graphs of each motif. We observe that the hairpin has one single component (i.e., one single stranded region), while the internal loop has two.

3.4 Prediction of RNA Tertiary Structures from Sequence Data

Modeling RNA tertiary structure is the first step toward predicting RNA 3D structures. We now describe how the knowledge accumulated in motif databases can be used together with RNA secondary structure prediction methods to predict the structure of large RNA molecules from sequence data only. The strategy presented here works in two steps. First, we predict a secondary structure using classical software such as RNAfold [20] and refine this

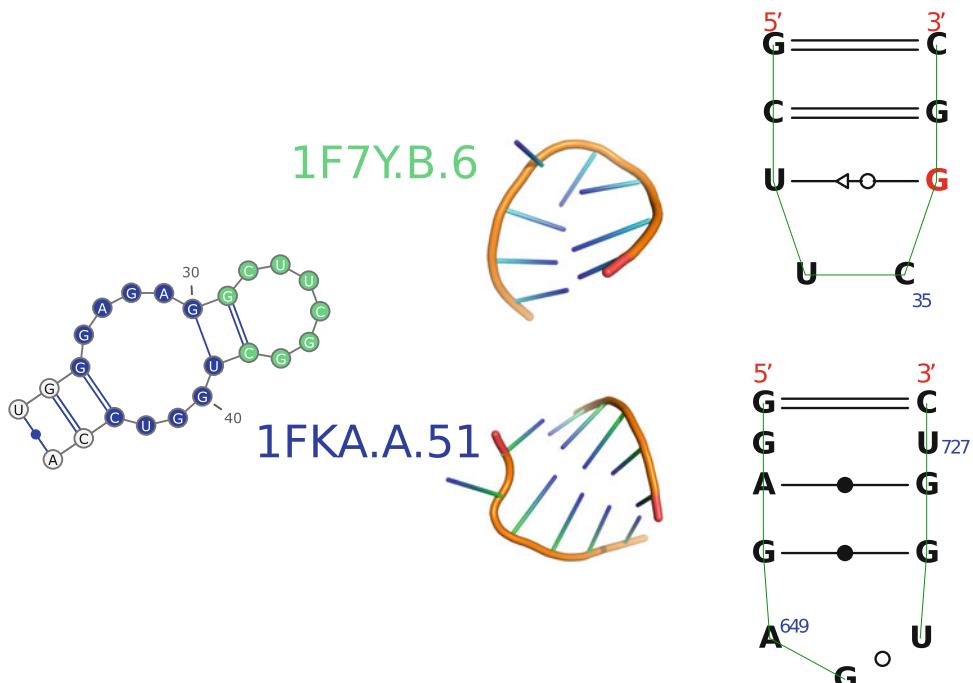


Fig. 3 Example of 3D RNA motif insertions in a secondary structure. In green, we show the position of the hairpin motif “1F7Y.B.6”, and in blue we indicate the position of the internal loop motif “1FK.A.51”. On the left side of the motif IDs, we display a 3D structure of the motif together with its base-pairing interaction graph

prediction by inserting RNA 3D motifs and adding noncanonical base pairs inside the predicted secondary structure with RNAMoIP [21]. Next, we use this extended secondary structure to reconstruct three-dimensional models of the molecule using the MC-Sym software [9].

3.5 Prediction of a Secondary Structure Scaffold

Our first objective is to create a base-pairing network from sequence data. Since the majority of base pairs in RNA structures are *cis* Watson–Crick base pairs, we first predict a secondary structure (without pseudo-knots) that will be used to build a scaffold of the interaction graph. Secondary structures (without pseudo-knot) can be deterministically predicted with RNAfold, or stochastically generated with RNAsubopt. The command line used to predict the minimum free energy (MFE) secondary structure with RNAfold is:

```
RNAfold --noPS < input.fasta
```

Where input.fasta is a text file (FASTA format recommended) that stores your input sequence. The--noPS flag is not mandatory, but it prevents the program to generate a postscript file drawing the predicted secondary structure. The program returns the input sequence with its MFE secondary structure in bracket format on the line below.

However, as discussed in previous chapters, single energy minimized structures do not always provide the best secondary structure prediction. Instead, it is recommended to perform a deeper exploration of the conformational space and to consider suboptimal structures [29, 30]. This approach to secondary structure prediction, originally implemented in mfold [30, 31] and Sfold [32], is available with the RNAsubopt program in the Vienna RNA package. The command line for running RNAsubopt is:

```
RNAsubopt -e 3 < input.fasta
```

Where the “-e” option specifies the depth of the suboptimal search. More specifically, this argument indicates the range (in kCal/mol) from the MFE, within which all suboptimal structures must be returned. Obviously, the values of that range dependent of the MFE of the sequence, and should be adjusted on a case-by-case basis. In our experiments, a value of 3 kCal/mol generated 25 secondary structures; which appears to provide a good representation of the suboptimal conformational landscape.

The list of suboptimal secondary structures generated by RNAsubopt (available at <http://csb.cs.mcgill.ca/RNAMoIP>) provides us a description of the set of potential secondary structure backbones. It will be used as it in the next step.

It is worth noting that other software could have been used to generate the initial secondary structures. RNAstructure [33, 34], mfold [30, 35], or Sfold [32, 36] make similar prediction than RNAsubopt. Further, recent software such as MC-Fold [9] and

RNAwol [37] have been developed to predict extended secondary structures (including all noncanonical base pairs) directly from sequence data.

3.6 Prediction of a Complete Base-Pairing Interaction Graph with Motif Insertion

We describe how to use RNA-MoIP [21] to insert local motifs into secondary structures generated with RNAsubopt. By default, RNA-MoIP aims to inserts motifs from a repository build with RNA3Dmotif [18]. This repository of nonredundant motifs is included in the distribution of RNA-MoIP and named “No_Redondance_DESC” (see Note 1). Nonetheless, advanced users can also either build themselves an up-to-date motif repository using RNA3Dmotif, or use databases available on the RNA 3D Hub [19].

RNA-MoIP is a flexible tool that allows modifications of the secondary structure to permit the insertion of motifs. In particular, the program has the capacity to remove a fixed amount of base pairs from the input secondary structure. This feature is particularly helpful if the predicted secondary structure has incorrectly predicted base pairs that prevent motifs to be inserted.

Lets assume that you work in a directory that contains the RNA-MoIP program (i.e., the python script named “RNAMoIP.py”) and that Gurobi has been properly installed. The command line to insert motifs in the sequence and secondary structure with RNA-MoIP is:

```
gurobi.sh RNAMoIP.py -s <sequence> -ss <structure_list>
-d <path_to_repository>
```

Where the argument <sequence> is a string representing the primary structure of the RNA sequence, <structure_list> is the name of the file that stores the secondary structures in bracket format generated by RNAsubopt, and <path_to_repository> is the location of the motif repository. It indicates the path to the motif repository stored in the directory named “CATALOGUE,” which is distributed with the RNA-MoIP package at <http://csb.cs.mcgill.ca/RNAMoIP/>. Assuming that the directory “CATALOGUE” is in your current directory, the value of <path_to_repository> is the string “./CATALOGUE/No_Redondance_DESC/”.

RNA-MoIP accepts an additional parameter to control the maximum number of base pairs that can be removed. This parameter can be adjusted with a float number (between 0 and 1) through the option –r. By default, RNA-MoIP allows up to 30 % (i.e., –r 0.3) of base pairs to be removed. This is a reasonable choice as the base-pair prediction positive predictive value (PPV) is roughly 60 % for classical secondary structure predictors such as RNAfold and mfold [38]. Nonetheless, users can decrease this value if they are confident in their predicted secondary structure.

Lets now run a concrete example on the tRNA(Cys) sequence, and lets call “RNAsubopt.out” the file storing the output of RNAsubopt. Then, the RNA-MoIP command line is:

```
gurobi.sh RNAMoIP.py -s
"GCCAGGGUGGCAGAGGGGCUUUGCGGGGACUGCAGAUCCGCUUACCCGGUUCGAA
UCCGGGCCUUGC"-ss RNAsubopt.out -d
"./CATALOGUE/No_Redondance_DESC/"
```

The program outputs first the usual output of Gurobi (the mathematical solver used by RNA-MoIP). We are interested by the output produced after the line starting with “Best objective.” It contains the secondary structure used to insert the motifs, the IDs of the inserted motifs, the positions of the deleted base pairs, and the score of the solution. The RNA-MoIP command above will output the following results:

```
Solution for the secondary structure:
((((((..((((..)))))).((((.....)))).....((((.....))))))))))

Optimal solution nb: 1

Corrected secondary structure: (((((..(((.....))))...
((((.....)))).....(((.....))))..))))))

C-2DU6.D.2-12-22-1
C-3CUL.D.6-51-61-1
C-2DU3.D.3-30-38-1
C-2DU5.D.1-6-10-1
C-2DU5.D.1-24-27-2
C-2DU5.D.1-41-48-3
C-2DU5.D.1-64-66-4
D-15-19
D-14-20
D-13-21
D-26-42
D-52-60
D-7-65
```

```
The optimal solutions has as value:
-663.0
```

The first line starts with “Solution for the secondary structure” and returns the best secondary structure extracted from the list of suboptimal structure, which has been used to insert to motifs. The next line (starting with “Optimal solution nb”) indicates how many suboptimal structures have been used. In our case, only one structure can be used to build the optimal solution.

The line starting with “Corrected secondary structure” shows the structure used by RNA-MoIP. Since RNA-MoIP may remove

some base pairs to insert motifs, this structure can be different to the one returned on the first line.

Then, the program outputs information about the inserted motifs and deleted base pairs. The rows starting with a “D” are the positions of the deleted base pairs. Hence, here the output tells us that the base pairs (15,19), (14,20), (13,21), (26,42), (52,60), and (7,65) have been removed to insert the motifs. We highlight in red the deleted base pairs.

((((((((..((((((...))))).((((.....)))).....((((.....)))))))))))

As indicated in the third line, the secondary predicted (or updated) by RNA-MoIP is:

(((((...(((.....))))..(((.....)))).....(((.....))))..))))))

The rows starting with a “C” display information about the inserted motifs. Each row indicates where a component of a motif (i.e., contiguous sequence of a local motif) has been used. The syntax of these lines is:

C-<ID>-<first_index>-<last_index>-<component_number>

Where <ID> is the identifier (or name) of the inserted motif, <first_index> is the first position and <last_index> is the last position of the insertion. The last value <component_number> indicates which component of the motif has been used.

In our example, four motifs have been inserted: three hairpins and one 4-way junction. For each motif, Table 1 shows the ID, type, and positions at which each component has been inserted. Figure 4 illustrates these results and shows where the motifs and components have been inserted by RNA-MoIP in the secondary structure.

The last line in the output returns the value of the objective function that has been optimized by RNA-MoIP. In our example, the value returned is -663.0. Details about this function can be found in ref. [21].

Table 1
Description of motifs inserted by RNA-MoIP in the suboptimal secondary structures generated by RNAsubopt for the tRNA(Cys) from *Archaeoglobus fulgidus*

Motif ID	Type	Indices of insertion sites
2DU6.D.2	Hairpin	[12, 22]
3CUL.D.6	Hairpin	[51,61]
2DU3.D.3	Hairpin	[30, 38]
2DU5.D.1	4-Way junction	[6,10], [24, 27], [41,48], [64,66]

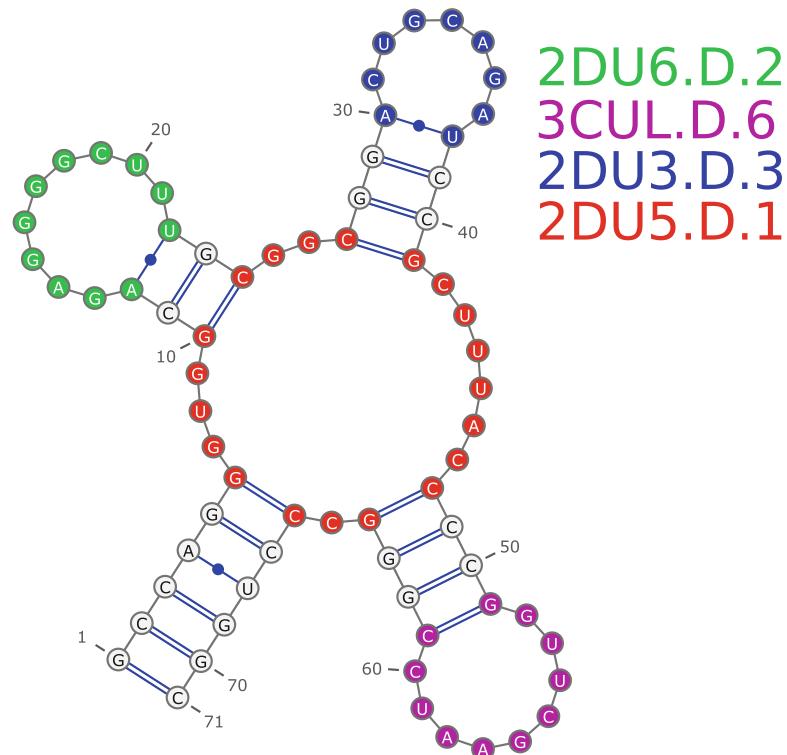


Fig. 4 Location of RNA motifs inserted by RNA-MoIP in the secondary structure of tRNA(Cys) from *Archaeoglobus fulgidus*. Three hairpins (2DU6.D.2 in green, 2CUL.D.6 in magenta, and 2DU3.D.3 in blue) and one 4-way junction (2DU5.D.1 in red) have been inserted. We note the 4 components of the 4-way junction, which correspond to the 4 single strands of the multi-loop

3.7 Reconstruction of Tertiary Structures from Base-Pairing Interaction Networks

Now, we describe how to use a base-pairing network to build three-dimensional structures with the MC-Sym software [9]. This operation requires writing scripts that will be used to run MC-Sym at <http://www.major.iric.ca/MC-Sym/>.

An MC-Sym script is composed of 6 parts. The first one provides a description of the sequence that is going to be modeled, and the second indicates the set of fragments to be used. The third part is the order in which the fragments will be merged. The fourth and fifth parts describe geometrical constraints to satisfy during the construction of the three-dimensional model. The last segment defines a set of rules for the space exploration. Every line starting by ‘//’ is a comment.

In this example, we show how to model the tRNA(Cys) molecule of *Archaeoglobus fulgidus*, a 71-nucleotides long RNA into which four motifs are inserted by RNAMoIP (See previous section). Each part of the script is described and explained separately.

Part 1: Sequence Definition

The first step defines the sequence we are modeling. The syntax is:

```
// ===== Sequence =====
sequence(r A1 GCCAGGGUGGCAGAGGGCUUUGCGCGACUGCAGA
UCCGCUUUACCCGGUUCGAAUCCGGGCCUGGC)
// (((((..((((..)))))))..(((.....)))).....(((((..)))))).))))))
// 1234567890123456789012345678901234567890123456789
0123456789012345678901
// 1 2 3 4 5 6 7
```

The keyword “sequence” states that we are defining a sequence. The program requires three arguments that are indicated between the parentheses. First, “r” specifies that we are working with an RNA sequence. Next, “A1” sets as identifier for the sequence “A” and defines the first position as “1.” Finally, we end by the sequence and close the parenthesis. The two lines of comment that follows are just here to improve readability.

Part 2: Fragment Definition

This is probably the most important and delicate step of the script. In MC-Sym, the basic units are called cycles. The second step aims to define which cycles and motifs will be used to build the structure. Cycles, motifs, and other structural elementary blocks used by MC-Sym are called fragments. The order in which they entered is not important, but we must ensure that every position is included in a fragment and that all fragments overlap.

In the following example, we illustrate how to insert a basic cycle (i.e., not a motif). Here, the cycle is a stack between two consecutive base pairs. This is the most common usage of cycles as most Watson–Crick and Wobble base pairs are predicted by secondary structure predictors and thus not covered by RNA motifs. The code below indicates how and where to map this cycle. In this case, a stacked base pairs at positions 1, 2, 70, and 71.

```
ncm_01=library(
  pdb("MCSYM-DB/2_2/GCGC/*R20*.pdb.gz")
  #1:#2, #3:#4<- A1:A2, A70:A71
  rmsd(0.1 sidechain && !(pse || lp || hydrogen)))
```

We start with a unique identifier (i.e., “ncm_01”), followed by the keyword “library.” It indicates to MC-Sym where to retrieve the basic fragments.

On the second line, we provide specifications of the files within that library. Here, our library is composed of PDB files, followed by the path “MCSYM-DB/2_2/GCGC/*_1.pdb.gz”. MCSYM-DB is the location of all standard library fragments. MC-Sym fragments are sorted according to the number of consecutive nucleotides in each component. Here, we have two times

two consecutive nucleotides (1–2 and 70–71). Therefore, we must look into the subfolder “2_2”. Next, we must explicitly tell which sequence of nucleotides are involved, in the order 5' to 3'. Since the nucleotides at positions 1, 2, 70, and 71 are respectively “G,” “C,” “G,” “C,” the next path is “GCGC.” We end by “*.pdb.gz” to consider all possible structures for this specific fragment. Specific subsets of those fragments can be chosen as specified in the MC-Sym documentation.

On the third line, we provide detailed information on the positions onto which the fragment will be mapped. The syntax is composed of two parts, separated by the symbol “<-”. The left-hand side indicates the segments associated with each consecutive component. Segments are represented with an interval, and nucleotides are numbered sequentially according to their position in the motif. The right-hand side defines the specific positions (still as intervals or segments) onto which these components/segments must match. In our sample code, we have two stretches of two nucleotides. We write it “#1:#2, #3:#4”, which also implies that the first part is composed of the two first nucleotides and the second one of the next two. On the right hand side, “A1:A2, A70:A71” describes which sequence must be mapped. The nucleotides at from position 1 to 2 for the first component, and the nucleotides at from position 70 to 71 for the second component. “A” is a unique identifier of the sequence to be used, as defined in the first part. The last line is a set of constraints for this specific fragment. Since this information is nonessential here, we kindly redirect the reader to the MC-Sym documentation for more details. It is also important to not forget to close the parenthesis opened after the keyword “library”

Now, we must specify how to declare the fragments inserted by RNAMoIP. The simplest one is a hairpin. Lets start with the hairpin “1DU6.D.2”, between positions “12” and “22”. The code is:

```
pdbs_hairpin_1=library(
    pdb ("~/reinharz/RNAMOIP-DB-VIEW3D/2DU6.D.2/*.pdb")
        #1:#11<- A12:A22
)
```

The syntax is very similar to those of the previous example. The first line is the same as for the stacked base pair: a unique identifier followed by the keyword “library.” However, the path to the library is different. All RNA-MoIP motifs are contained in a repository that is accessible with the path “~/reinharz/RNAMOIP-DB-VIEW3D/<Motif Identifier>”. The path must always end by “*.pdb” to indicate that we want to consider all possible structures in the folder. On the third line, we define the positions involved. In this case, the motif contains 11 consecutive nucleotides, thus the left hand side is “#1:#11”. Those nucleotides are at positions

“12” to “22” in our sequence, defining the right hand side. We end by closing the parenthesis of “library.”

Finally, motifs with multiple stretches of consecutive nucleotides are a slightly more complicated to specify. We illustrate that case with the 4-way junction “2DU5.D.1”. Using the same syntax as above, we write the code as follows.

```
pdb4way=library(
    pdb ("~/u/reinharz/RNAMOIP-DB-VIEW3D/2DU5.D.1/*.
    pdb")
        #1:#5, #6:#9, #10:#17, #18:#20 <- A6:A10, A24:A27,
    A41:A48, A64:A66
)
```

In this case, our motif is composed of 4 components, between positions “6” and “10,” “24” and “27,” “41” and “48,” and “64” and “66” in the input sequence. Each component of consecutive nucleotides needs to be described individually.

Part 3: Merging Fragments

Once all fragments are specified, we must define in which order they are going to be assembled together. One fragment must be selected to start the construction. All others will be sequentially added; with the constraint that each new fragment must overlaps with any of the previous one. The syntax to assemble the first two stacked base pairs, uses the unique names used in the fragments definitions (i.e., ncm_01 and ncm_02), and is written as follows.

```
structure=backtrack
(
    ncm_01
    merge(ncm_02 1.5)
)
```

It is important to always start with the keywords “structure=backtrack”. The assembly is indicated between two parentheses. The starting point of our model is “ncm_01” and must be written on the first line. On the second line we use the keyword “merge” followed, between parentheses, by an argument that indicates the next fragment, and another argument that specifies an “error” threshold that can be tolerated by the program to extend the structure. By default we use a value of “1.5”. More details on this parameter can be found in the MC-Sym documentation.

Part 4 and 5: Constraints

MC-Sym requires several general parameters and constraints to perform the three-dimensional reconstruction. A standard set of values is provided in the following code and can be used as it is.

```

// ===== Backtrack Restraints =====

clash
(
    structure
    1.5 !(pse || lp || hydrogen)
)
backtrack_rst
(
    structure
    width_limit=25 %,
    height_limit=33 %,
    method=probabilistic
)

//=====RiboRestraints=====

ribose_rst
(
    structure
    method=ccm,
    pucker=C3p_endo,
    threshold=2.0
)

```

Part 6: Space Exploration Parameters

Finally, the last step is to determine the boundaries of the space to explore. We also specify the time limit and the maximal number of structures to output. MC-Sym will stop as soon as one of those conditions is fulfilled. The other parameter options are standard and can be found in the documentation of MC-Sym.

```

// ===== Exploration Initialization =====

explore
(
    structure
    option(
        model_limit=1000,
        time_limit=30 m,
        seed=3210)
    rmsd(3.0 sidechain && !(pse || lp || hydrogen))
    pdb("structure" zipped)
)

```

Where, the key word “model_limit” specifies the maximal number of structure to generate, and “time_limit” the maximal amount of time allowed to run MC-Sym. Here, we use an upper bound of 30 min with a maximum of 1,000 structures. The time needed to generate structures can vary. In our benchmark [21], we have seen that about half an hour of computation was often enough to produce solutions with molecules with sizes up to one hundred nucleotides. Within this time frame, the maximal number of structures created will rarely reach one thousand in that time. Importantly, motifs with large number of components will highly restrict the exploration of the space, thus the time needed to explore the conformational space. In absence of any motif with three or more components, increasing the time limit to 2 days is a reasonable step. In that case, the number of structures could increase to five or even ten thousands, and we can expect much more diversity in the sample set.

It is difficult to determine in advance how many structure samples are necessary to provide a good representation of the conformational landscape. Numbers between 100 and 1,000 are good starts, but more might be occasionally needed. We acknowledge that large numbers of structures are difficult to analyze. Fortunately, as we will see in the next section, the MC-Sym server offers several tools to facilitate these tasks.

The key word “seed” is a random number used to initialize MC-Sym. Its value has little importance in the context of this discussion, and users can modify it if they wish.

The complete source code of this MC-Sym script is available at <http://csb.cs.mcgill.ca/RNAMoIP/>.

3.8 Retrieving, Optimizing, Analyzing and Visualizing Structures

Once submitted, the script will run on MC-Sym servers and the URL of a result page will be returned to the user. This page offers multiple options to optimize, analyze, and retrieve the results. To access these options, the user must access the web page “commands.html” located in the results directory.

The energy minimization of the MC-Sym structure is probably one of the most important options. We recommend any user to run it before analyzing or visualizing the results. The “steepest descent” option returns satisfactory results in a short time, but more sophisticated and slower techniques (e.g., simulated annealing) are also available.

Clustering of structures using the k-means algorithm is another useful option that enables the user to automatically identify the most significant structures in the set of structures returned by MC-Sym.

A PDB model of all predicted structures is available at the root of the directory. Each model can be visualized with molecular viewer such as PyMOL or Jmol. Figure 5 show an example of a structure predicted with our RNA-MoIP and MC-Sym pipeline, aligned with the experimental structure [24].



Fig. 5 3D structure of the tRNA(Cys) from *Archaeoglobus fulgidus* predicted with the RNA-MoIP + MC-Sym pipeline (in blue), aligned with its crystallographic model (Fukunaga and Yokoyama [24])

4 Notes

1. The motifs database used by RNA-MoIP includes motifs from the PDB model 2DU3 (tRNA(Cys) from *Archaeoglobus fulgidus*) database. Thus, the quality of the prediction achieved in our example illustrates the performance of RNA-MoIP in the best-case scenario. In applications on other (new) molecules, it will be very unlikely to have motifs from the same molecule in the motif repository. Fortunately, even in the absence of motifs, the benchmark performed in ref. [21] shows that the quality of the structure prediction remains high.

References

1. Bekaert M et al (2003) Towards a computational model for -1 eukaryotic frameshifting sites. *Bioinformatics* 19(3):327–335
2. Vitreschak AG et al (2004) Riboswitches: the oldest mechanism for the regulation of gene expression? *Trends Genet* 20(1):44–50
3. Szewczak AA et al (1993) The conformation of the sarcin/ricin loop from 28S ribosomal RNA. *Proc Natl Acad Sci U S A* 90(20): 9581–9585
4. Sponer J et al (2012) Chapter 6 molecular dynamics simulations of RNA molecules, in innovations in biomolecular modeling and simulations. *R Soc Chem* 2:129–155
5. Bernauer J et al (2011) Fully differentiable coarse-grained and all-atom knowledge-based potentials for RNA structure evaluation. *RNA* 17(6):1066–1075
6. Ding F et al (2008) Ab initio RNA folding by discrete molecular dynamics: from structure prediction to folding mechanisms. *RNA* 14(6): 1164–1173
7. Jonikas MA et al (2009) Coarse-grained modeling of large RNA molecules with knowledge-based potentials and structural filters. *RNA* 15(2):189–199
8. Poursina M et al (2011) Strategies for articulated multibody-based adaptive coarse grain simulation of RNA. *Methods Enzymol* 487:73–98
9. Parisien M, Major F (2008) The MC-fold and MC-Sym pipeline infers RNA structure from sequence data. *Nature* 452(7183):51–55

10. Martinez HM, Maizel JV Jr, Shapiro BA (2008) RNA2D3D: a program for generating, viewing, and comparing 3-dimensional models of RNA. *J Biomol Struct Dyn* 25(6):669–683
11. Zhao Y et al (2012) Automated and fast building of three-dimensional RNA structures. *Sci Rep* 2:734
12. Das R, Baker D (2007) Automated de novo prediction of native-like RNA tertiary structures. *Proc Natl Acad Sci U S A* 104(37):14664–14669
13. Das R, Karanicolas J, Baker D (2010) Atomic accuracy in predicting and designing noncanonical RNA structure. *Nat Methods* 7(4):291–294
14. Wang Z, Xu J (2011) A conditional random fields method for RNA sequence-structure relationship modeling and conformation sampling. *Bioinformatics* 27(13):i102–i110
15. Rother M et al (2011) ModeRNA: a tool for comparative modeling of RNA 3D structure. *Nucleic Acids Res* 39(10):4007–4022
16. Leontis NB, Westhof E (2001) Geometric nomenclature and classification of RNA base pairs. *RNA* 7(4):499–512
17. Yang H et al (2003) Tools for the automatic identification and classification of RNA base pairs. *Nucleic Acids Res* 31(13):3450–3460
18. Djelloul M, Denise A (2008) Automated motif extraction and classification in RNA tertiary structures. *RNA* 14(12):2489–2497
19. Leontis N, Zirbel CL (2012) Nonredundant 3D structure datasets for RNA knowledge extraction and benchmarking. In: Leontis N, Westhof E (eds) *RNA 3D structure analysis and prediction*. Springer, Berlin, pp 281–298
20. Hofacker IL et al (1994) Fast folding and comparison of RNA secondary structures. *Monatsh Chem* 125(2):167–188
21. Reinharz V, Major F, Waldspuhr J (2012) Towards 3D structure prediction of large RNA molecules: an integer programming framework to insert local 3D motifs in RNA secondary structure. *Bioinformatics* 28(12):i207–i214
22. Berman HM et al (1992) The nucleic acid database. A comprehensive relational database of three-dimensional structures of nucleic acids. *Biophys J* 63(3):751–759
23. Bernstein FC et al (1977) The Protein Data Bank: a computer-based archival file for macromolecular structures. *J Mol Biol* 112(3):535–542
24. Fukunaga R, Yokoyama S (2007) Structural insights into the first step of RNA-dependent cysteine biosynthesis in archaea. *Nat Struct Mol Biol* 14(4):272–279
25. Lorenz R et al (2011) ViennaRNA Package 2.0. *Algorithms Mol Biol* 6:26
26. Waugh A et al (2002) RNAML: a standard syntax for exchanging RNA information. *RNA Rep* 8(6):707–717
27. Lemieux S, Major F (2002) RNA canonical and non-canonical base pairing types: a recognition method and complete repertoire. *Nucleic Acids Res* 30(19):4250–4263
28. Chojnowski G, Walen T, Bujnicki JM (2014) RNA Bricks—a database of RNA 3D motifs and their interactions. *Nucleic Acids Res* 42(1):D123–D131
29. Ding Y, Chan CY, Lawrence CE (2005) RNA secondary structure prediction by centroids in a Boltzmann weighted ensemble. *RNA* 11(8):1157–1166
30. Zuker M (1989) On finding all suboptimal foldings of an RNA molecule. *Science* 244(4900):48–52
31. Zuker M, Mathews DH, Turner DH (1999) Algorithms and Thermodynamics for RNA Secondary Structure Prediction: A Practical Guide. In: Barciszewski J, Clark BFC (eds) *RNA Biochemistry and Biotechnology*. Springer, Netherlands, pp 11–43
32. Ding Y, Lawrence CE (2003) A statistical sampling algorithm for RNA secondary structure prediction. *Nucleic Acids Res* 31(24):7280–7301
33. Reuter JS, Mathews DH (2010) RNAstructure: software for RNA secondary structure prediction and analysis. *BMC Bioinformatics* 11:129
34. Bellaousov S et al (2013) RNAstructure: Web servers for RNA secondary structure prediction and analysis. *Nucleic Acids Res* 41(Web Server issue):W471–W474
35. Zuker M (2003) Mfold web server for nucleic acid folding and hybridization prediction. *Nucleic Acids Res* 31(13):3406–3415
36. Ding Y, Chan CY, Lawrence CE (2004) Sfold web server for statistical folding and rational design of nucleic acids. *Nucleic Acids Res* 32(Web Server issue):W135–W141
37. Honer zu Siederdissen C et al (2011) A folding algorithm for extended RNA secondary structures. *Bioinformatics* 27(13):i129–i136
38. Do CB, Woods DA, Batzoglou S (2006) CONTRAfold: RNA secondary structure prediction without physics-based models. *Bioinformatics* 22(14):e90–e98

Chapter 7

Fast Prediction of RNA–RNA Interaction Using Heuristic Algorithm

Soheila Montaseri

Abstract

Interaction between two RNA molecules plays a crucial role in many medical and biological processes such as gene expression regulation. In this process, an RNA molecule prohibits the translation of another RNA molecule by establishing stable interactions with it. Some algorithms have been formed to predict the structure of the RNA–RNA interaction. High computational time is a common challenge in most of the presented algorithms. In this context, a heuristic method is introduced to accurately predict the interaction between two RNAs based on minimum free energy (MFE). This algorithm uses a few dot matrices for finding the secondary structure of each RNA and binding sites between two RNAs. Furthermore, a parallel version of this method is presented. We describe the algorithm's concurrency and parallelism for a multi-core chip. The proposed algorithm has been performed on some datasets including CopA–CopT, R1inv–R2inv, Tar–Tar*, DIS–DIS, and IncRNA₅₄–RepZ in *Escherichia coli* bacteria. The method has high validity and efficiency, and it is run in low computational time in comparison to other approaches.

Key words RNA secondary structure, RNA–RNA interaction, Heuristic, Parallel, Minimum free energy

1 Introduction

RNA–RNA interaction is applied in the gene regulation [1]. The interaction between two RNAs is used in the treatment of many cancers, including colon, breast, and pancreatic by silencing the gene expression or genes involved in the development of malignancy. In most living organisms, noncoding RNAs (ncRNAs) bind to their target mRNAs to posttranscriptional regulation [2]. Furthermore, small interfering RNAs (siRNAs) and microRNAs (miRNAs) play an important role in the gene suppressing process [3].

RNA–RNA interaction prediction is one of the most challenging problems in bioinformatics that involves an interplay between the secondary structure of each molecule on the one hand, and the binding of the two molecules on the other hand [4].

The first method to predict RNA–RNA interaction is carried out by concatenating two RNA sequences as a single sequence and

using dynamic programming techniques [5, 6]. A similar approach has been presented based on a rigorous extension of secondary structure models to multistranded cases. It calculates the partition function of a secondary structure complex of multiple interacting RNAs [7].

Reduction of computational time might be achieved by ignoring all internal base pairings in both RNAs. This idea was first introduced by RNAhybrid [8] and UNAFold [9]. It was later implemented by RNAduplex from *Vienna package* [6].

Minimizing the free energy between two RNA molecules in a number of energy models with growing complexity was introduced in [10]. In both RNAup [11, 12] and intaRNA [13] the single regions on each RNA are used to find the interaction between two RNAs. In RNAPlex [3], the possible hybridization sites for an RNA in large RNA databases are found based on a slight simplification of the energy model. In this model, the loop energy is a function of the loop size. An approach based on the stochastic context free grammars was introduced [14]. In this algorithm, two real values called transition probability and emission probability are specified for each rule of the grammar.

InteRNA [15] computes the optimal set of interactions among all loops as well as single-stranded sequences in the original secondary structures of interacting RNA molecules. This method uses a heuristic approach for solving this problem. In another heuristic algorithm, highly accessible regions are predicted in each sequence. These regions include the loop regions in the native structure of RNA strand. Then the optimal non-conflicting interaction regions are found. In this algorithm, each accessible region can interact with at most two accessible regions from the other sequence [2].

Approximation algorithms were introduced in ref. [4]. An RNA–RNA interaction graph is created in which every edge represents a possible bond in or between two RNAs. A set of node disjoint edges is found to maximize the number of bonds. A statistical sampling algorithm [1] calculates the interaction probabilities for any given interval on the target RNA by grammars.

In this context, a heuristic method is presented for RNA–RNA interaction prediction problem. This algorithm has two main steps. Firstly, an RNA secondary structure (RSS) is constructed for each RNA sequence. Secondly, the interaction between two RNA secondary structures is computed. In order to make an RSS, a dot matrix is constructed for each RNA sequence. Each sub-diagonal in the dot matrix can be considered as a stem. Using some of these sub-diagonals, an RSS is generated. Then another dot matrix is created for finding the interaction. Each sub-diagonal can be seen as a part of a binding site between two RNAs. Finally, some of these sub-diagonals are chosen as the interaction regions between two RNAs. In these dot matrices, all possible base pairs in RNA secondary structure and RNA–RNA interaction are considered,

and thus each set of consecutive unpaired bases can interact with other complementary unpaired sets. Here, we also have implemented a modeling for RNA–RNA interaction prediction based on parallel heuristic approach. All diagonals in each dot matrix can be considered independently of each other. Therefore, the sub-diagonals on different diagonals and their minimum free energy (MFE) are calculated concurrently. The sub-diagonals are sorted in parallel based on their length and MFE.

2 Materials

2.1 Dataset

The proposed algorithm has been performed on some standard datasets such as CopA-CopT, R1inv-R2inv, Tar-Tar*, DIS-DIS, and IncRNA₅₄-RepZ in *Escherichia coli* bacteria [14]. These pair RNAs have kissing hairpin structures.

Before presenting our algorithm, some definitions and notations about RNA are presented. An RNA molecule consists of a sequence of four nucleotides: adenine (A), cytosine (C), guanine (G), and uracil (U). Thus, an RNA sequence is defined by $R = r_1 r_2 \dots r_n$ in 5'-3' direction, where $|R| = n$ and $r_i \in \{A, C, G, U\}$ ($1 \leq i \leq n$). On the other hand, the reverse R is indicated as $r_n r_{n-1} \dots r_1$ in 3'-5' direction. A subsequence $r_{ij} = r_i r_{i+1} \dots r_j$ from the sequence R is started from the position i and ended at the position j in 5'-3' direction. The reverse r_{ij} is defined as $r_j r_{j-1} \dots r_i$ in 3'-5' direction. The RNA structure is formed by the creation of hydrogen bonds between complementary bases (A-U, C-G).

An RNA secondary structure is composed of stems and single regions. Each stem contains some consecutive base pairs such as r_i, r_j and r_{i+1}, r_{j-1} . Let r_{ij} and r_{kl} be two subsequences from the sequence R that form a stem. So the subsequence r_{ij} is bonded to the reverse r_{kl} base to base. For the stem, each base in r_{ij} is represented by '(' and each base in r_{kl} is displayed with ')'. A single region is a loop or single strand. Each end of a loop is linked to a stem, while one end of a single strand is connected to a stem. For the single region, each base is represented by '.'. Thus the secondary structure of the RNA R is indicated with a sequence of characters '(', ')' and '.'.

An RNA–RNA interaction structure is composed of stems in each RNA and hybrids between the two RNAs. Each hybrid contains some consecutive base pairs between the two RNAs without any base pairs in each RNA. Let r_{ij}' and r_{kl}'' be two single regions of RNA secondary structures R' and reverse R'' , respectively, which are involved in a hybrid. Thus the subsequence r_{ij}' is bonded to the reverse r_{kl}'' base to base. For the hybrid, each base in r_{ij}' is shown by '[' and each base in the r_{kl}'' is declared by ']'.

In RNA–RNA interaction prediction problem, two RNA sequences $R' = r_1' r_2' \dots r_n' (|R'| = n)$ and $R'' = r_1'' r_2'' \dots r_m'' (|R''| = m)$ are given as inputs. The goal is to predict the interaction structure between R' and R'' that maximizes the sum of the length (number of 1's) of stems and hybrids. Moreover, the free energy of the interaction structure should be minimized.

2.2 Computational Hardware

The proposed method is run on Intel(R) Core(TM)2 Duo processor T6670 2.20 GHz with 4 GB RAM to predict the interaction structures. Parallel version of the algorithm implements parallelism on a multicore system. Experiments are performed on an Intel(R) core(TM) i7 2600k CPU 3.40 GHz with 8G RAM.

2.3 Computational Software

The heuristic method for RNA–RNA interaction prediction is performed in language C++. The parallel heuristic approach is implemented in language Matlab.

3 Methods

3.1 RNA Secondary Structure Prediction

In this section, a heuristic algorithm is presented for RNA–RNA interaction prediction. This method has two main steps. In the first step, an RNA secondary structure is constructed for each RNA sequence as follows:

1. A stem dot matrix $M_{n \times n}^R$ is constructed for the RNA sequence $R = r_1 r_2 \dots r_n (|R| = n)$ as follows:

$$M^R[i, j] = \begin{cases} 1 & \text{if } (r_i, r_j) \in \{(A, U), (U, A), (C, G), (G, C)\}, \\ 0 & \text{else} \end{cases}$$

where r_i and r_j ($1 \leq i, j \leq n$) are the i -th nucleotide in the sequence R and the j -th nucleotide in the reverse R , respectively.

2. In the stem dot matrix $M_{n \times n}^R$, each right-skewed consecutive value of 1's which is parallel to the main diagonal shows a stem sub-diagonal. A set of stem sub-diagonals for the RNA R is formed as follows:

$$D^R = \{ \langle i, j, k, l \rangle | 1 \leq i \leq k \leq n \ \& 1 \leq j \leq l \leq n \},$$

where (i, j) and (k, l) are the start and end positions of a stem sub-diagonal, respectively. Let $d^R \in D^R$ and $d^R = \langle i, j, k, l \rangle$. Then d^R shows how the subsequence r_{ik} is bonded to the reverse r_{jl} in R . If there are i' and j' where $i < i' < k$, $j < j' < l$ and $i' + j' = n + 1$, then d^R has to be removed and two stem sub-diagonals $\langle i, j, i', j' \rangle$ and $\langle i' + 1, j' + 1, k, l \rangle$ have to be inserted to the set D^R .

3. Length and MFE are computed for each stem sub-diagonal. The length of stem sub-diagonal is the number of 1's in it. MFE of each sub-diagonal is determined as the sum of the free energies of two adjacent base pairs in it, as follows:

$$\text{MFE}(s) = \sum_{(r_i, r_j), (r_{i+1}, r_{j-1}) \in s} e(r_{i,i+1}, r_{j-1,j}), \quad (1)$$

where s represents a stem or hybrid, and $e(r_{i,i+1}, r_{j-1,j})$ shows the free energy of two adjacent base pairs (r_i, r_j) and (r_{i+1}, r_{j-1}) . The MFEs of all two adjacent base pairs are depicted in Table 1.

4. Stem sub-diagonals in the array D^R are decreasingly sorted based on the length, then each set of the sub-diagonals with equal length is increasingly sorted based on MFE.
 5. The sub-diagonals are selected to form RNA secondary structure based on order of setting in the set D^R . Notice that the selected sub-diagonals should not have overlapping. Let D be a type of sub-diagonal sets and $d_1, d_2 \in D$, where $d_1 = \langle i_1, j_1, k_1, l_1 \rangle$ and $d_2 = \langle i_2, j_2, k_2, l_2 \rangle$. Overlapping between two sub-diagonals d_1 and d_2 is defined as follows:

$$\text{Overlap}(d_1, d_2) = \begin{cases} 1 & \text{if } \exists p : i_1 \leq p \leq k_1 \text{ & } i_2 \leq p \leq k_2, \\ 1 & \text{if } \exists p : j_1 \leq p \leq l_1 \text{ & } j_2 \leq p \leq l_2, \\ 0 & \text{else.} \end{cases}$$

After performing the algorithm, S represents the secondary structure of the RNA R . Accordingly, S' and S'' are the structures of RNAs R' and R'' , respectively. Each base pair in the stem regions is represented by '(' and ')'.

Table 1
MFE of all two adjacent base pairs

5'→3'	AA	AC	AG	AU	CA	CC	CG	CU	GA	GC	GG	GU	UA	UC	UG	UU
AA	-0.9
AC	-2.2	.
AG	-2.1	.	-0.6
AU	-1.1	.	-1.4	.
CA	-2.1
CC	-3.3
CG	-2.4	.	-1.4
CU	-2.1	.	-2.1
GA	-2.4	-1.3	.
GC	-3.4	-2.5	.	.
GG	-3.3	.	-1.5	-2.1	.	-0.5	.
GU	-2.2	.	-2.5	-1.4	.	1.3	.	.
UA	.	.	.	-1.3	-1
UC	.	.	-2.4	-1.5
UG	-2.1	.	-1	-1.4	.	0.3
UU	-0.9	.	-1.3	-0.6	.	-0.5

3.2 RNA–RNA Interaction Prediction

In the second step, interaction structure between the two structures S' and S'' is calculated as follows:

1. A hybrid dot matrix $M_{n \times m}^{R', R''}$ is made up of two secondary structures of RNA sequences $R' = r_1', \dots, r_n'$ and $R'' = r_1'', \dots, r_m''$ ($|R'| = n$, $|R''| = m$), as follows:

$$M^{R', R''}[i, j] = \begin{cases} 1 & \text{if } (r_i', r_j'') \in \{(A, U), (U, A), (C, G), (G, C)\} \& (s_i', s_j'') = ('.', '.', ') \\ 0 & \text{else,} \end{cases}$$

where (r_i', r_j'') are the i -th and j -th nucleotides and (s_i', s_j'') are the i -th and j -th structures in the sequence R' and reverse R'' , respectively ($1 \leq i \leq n$, $1 \leq j \leq m$).

2. In the hybrid dot matrix $M_{n \times m}^{R', R''}$, each right-skewed consecutive value of 1's which is parallel to the main diagonal shows a hybrid sub-diagonal. A set of hybrid sub-diagonals for the two RNAs R' and R'' is formed as follows:

$$D^{R', R''} = \{ \langle i, j, k, l \rangle | 1 \leq i \leq k \leq n \ 1 \leq j \leq l \leq m \},$$

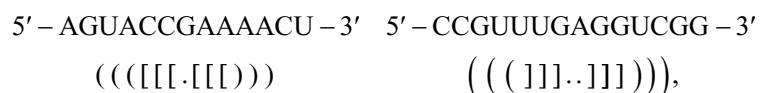
where (i, j) and (k, l) are the start and the end positions of the hybrid sub-diagonal, respectively. Each $d^{R', R''} = \langle i, j, k, l \rangle \in D^{R', R''}$ indicates that the subsequence r_{ik}' in R' is bonded to the r_{jl}'' in reverse R'' .

The remaining steps are similar to **steps 3, 4, and 5** in Subheading 3.1.

The time and space complexity of the algorithm are $O(k^2 \log k^2)$ and $O(k^2)$, respectively, where k indicates the sum of the length of two RNAs.

Example 1

Let $R' = \text{AGUACCGAAAACU}$ and $R'' = \text{CCGUUUGAGGUUCGG}$ be two RNA sequences. Interaction between the two RNAs can be shown as follows:



Here, each RNA has one stem. In the left hand RNA, one stem is found by the formation of bonding between AGU and the reverse ACU. Consecutive open brackets and their corresponding closing brackets are denoted as an interaction region between two RNAs. Here, there are two interaction regions between the sequences R' and R'' . The first one is formed by binding between ACC and the reverse GGU. The second one is generated by binding between AAA and the reverse UUU.

3.3 Parallel Algorithm

Here, parallel version of the heuristic algorithm is described. Let N be the number of created threads. Each dot matrix is constructed in parallel. Notice that an element (x, y) in the dot matrix is dependent on a set of elements (x, j) and (i, y) for each $1 \leq i, j \leq n$ as shown in Fig. 1. This dependency pattern allows that the main diagonal and other diagonals parallel to it can be considered independent of each other. So divide the diagonals between threads. Figure 2

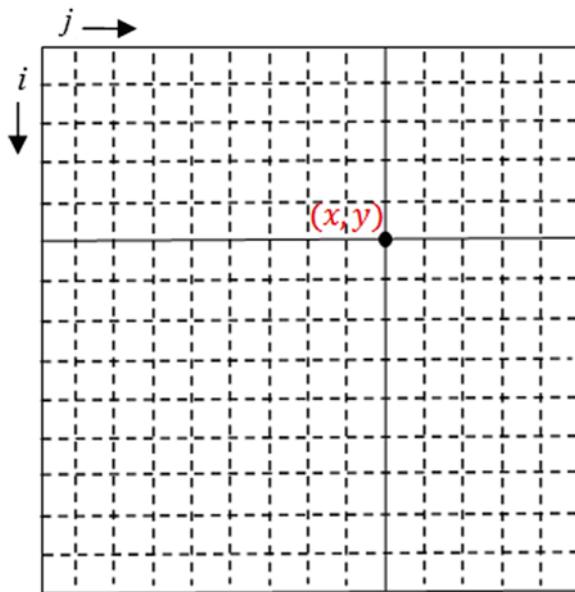


Fig. 1 Dependency of element (x, y) to a set of elements (x, j) and (i, y) for each $1 \leq i, j \leq n$

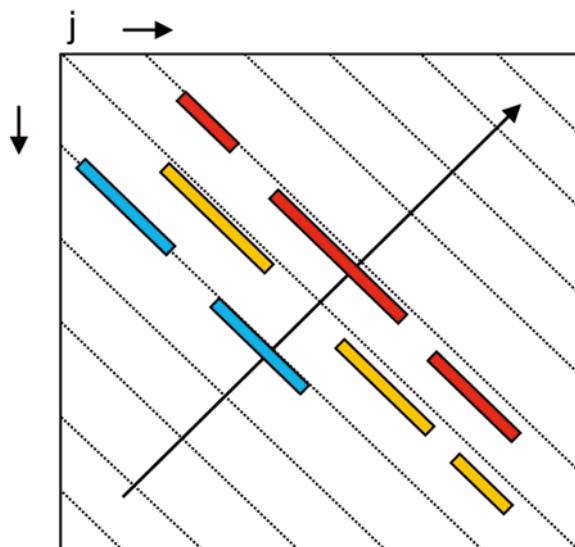


Fig. 2 Pattern of implemented computation

shows the sequence of computations of sub-diagonals. In this figure, some samples of sub-diagonals are shown in different colors on distinct diagonals.

Procedure Merge-Split in ref. [16] is called to sort the sub-diagonals decreasingly based on the length. The Merge-Split is recalled to sort the sub-diagonals with equal length increasingly based on the *MFE*.

Creating the dot matrices and extracting the sub-diagonals require time complexity $O(k^2/N)$, where k and N indicate the sum of the length of two RNAs and the number of threads, respectively. Sorting the arrays of sub-diagonals is run in order of $O(k^2 \log k^2/N)$. Therefore, the time complexity of the algorithm is $O(k^2 \log k^2/N)$. Because of using dot matrices, computational space is $O(k^2)$.

3.4 Algorithm Limitations

One of the limitations of the proposed algorithm is that we have first obtained the length of sub-diagonals and then computed their MFEs for RNA–RNA interaction prediction, while the interaction between two RNAs is formed based on MFE.

To evaluate the prediction accuracy of the method, sensitivity, specificity, and *F*-measure have been employed. Let *TP*, *FP*, and *FN* be the number of correctly predicted base pairs, the number of false predicted base pairs, and the number of unpredicted base pairs, respectively. So the sensitivity (*Sn*), specificity (*Sp*), and *F*-measure (*F*) are defined as follows:

$$Sn = TP / (TP + FN), \quad Sp = TP / (TP + FP), \quad F = (2 * Sn * Sp) / (Sn + Sp).$$

We have compared the prediction accuracy of our parallel method, TPIRNA, in these criteria with some similar approaches such as InRNAs, RNAUp [12] and EBM in Table 2. As it is shown, the

Table 2
Comparison of sensitivity, specificity, and *F*-measure

RNA–RNA pairs	Sensitivity (%)				Specificity (%)				<i>F</i> -measure (%)			
	TPIRNA	inRNAs	RNAUp	EBM	TPIRNA	inRNAs	RNAUp	EBM	TPIRNA	inRNAs	RNAUp	EBM
Tar-Tar*	100	100	100	100	100	87.5	83.3	93.3	100	93.3	90.9	96.5
R1inv-R2inv	100	90	100	90	100	90	77.8	94	100	90	87.5	92
DIS-DIS	100	100	100	78.6	100	100	100	78.6	100	100	100	78.6
CopA-CopT	96.74	100	55.6	90.09	78.18	84.6	65.2	80	86.86	91.7	60	85.1
IncRNA ₅₄₋ RepZ	86	87.5	75	91.7	75.51	79.2	85.7	83	80.41	83.1	80	87.1
Average	96.55	95.5	86.12	90.23	90.73	95.5	82.4	85.9	93.45	91.6	83.69	87.9

Table 3
Comparison of running times of TPIRNA with 1, 2, 4, and 8 threads and also EBM

RNA–RNA pairs	1-TPIRNA	2-TPIRNA	4-TPIRNA	8-TPIRNA	EBM
CopA-CopT	0.176	0.089	0.033	0.021	1206.47
DIS-DIS	0.162	0.075	0.016	0.011	382.13
Tar-Tar*	0.152	0.064	0.008	0.006	21.25
R1inv-R2inv	0.161	0.060	0.009	0.007	49.21
lncRNA ₅₄ -RepZ	0.202	0.089	0.030	0.023	1328.62
Average	0.17	0.037	0.019	0.013	597.39

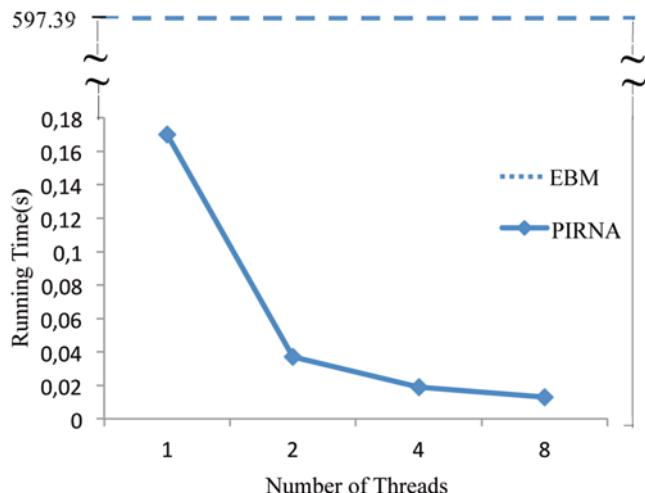


Fig. 3 Comparison of the average running times of TPIRNA using 1, 2, 4, and 8 threads and sequential EBM on our datasets

average accuracy of TPIRNA in the sensitivity, specificity, and *F*-measure is 96.55 %, 90.73 %, and 93.45 %, respectively. According to the table, the average accuracy of the proposed approach is higher than these algorithms.

Table 3 represents the running times achieved using 1, 2, 4, and 8 threads for TPIRNA in comparison to the sequential EBM on our datasets. TPIRNA computes interaction of CopA-CopT in 0.021 s with 8 threads, while EBM takes approximately 20 min. In the table, *i* -TPIRNA represents that TPIRNA uses *i* thread(s). Comparison of the average running times of TPIRNA using 1, 2, 4, and 8 threads and also EBM on our datasets is shown in Fig. 3.

Table 4
Comparison of time and space complexity of some methods

Algorithm	Time complexity	Space complexity
TPIRNA	$O(k^2 \log k^2 / N)$	$O(k^2)$
SPM	$O(n^3 m^3)$	$O(n^2 m^2)$
LM	$O(n^3 m^3)$	$O(n^2 m^2)$
inRNAs	$O(k^4 w)$	$O(k^2)$
RNAUp	$O(n^3 m)$	$O(n^2)$
EBM	$O(n^3 m^3)$	$O(n^2 m^2)$
App	$O(n^3 m^3)$	$O(n^2 m^2)$
Pairfold	$O(k^3)$	$O(k^2)$
IntaRNA	$O(nm + n\ell^3)$	$O(nm)$
Ripalign	$O(N^6)$	$O(N^4)$
PETcofold	$O(MI^3)$	

The dashed horizontal line represents the sequential running time of EBM. It is clear from the figure that TPIRNA with one thread performs much faster than EMB. The running time of TPIRNA is decreased when the number of threads is increased.

The time and space complexity of some algorithms such as Petcofold [17], IntaRNA [18], ripalign [19], Pairfold [5], App [4], EBM, RNAUp, LM [10], and SPM [10] is represented in Table 4.

According to the table, minimum time complexity of these algorithms except TPIRNA is $O(k^3)$, where k indicates the sum of the lengths of two RNAs. As it is shown, TPIRNA is run in a time complexity of $O(k^2 \log k^2 / N)$. Thus our algorithm is performed in lower computational time in comparison to other methods.

Acknowledgements

Special thanks to Professor Fatemeh Zare-Mirakabad and Nasrollah Moghadam-charkari for their valuable consultancy in improving this work.

References

1. Huang FWD, Qin J, Reidys CM et al (2010) Target prediction and a statistical sampling algorithm for RNA-RNA interaction. *Bioinformatics* 26:175–181
2. Salari R, Backofen R, Sahinalp SC (2010) Fast prediction of RNA-RNA interaction. *Algorithms Mol Biol* 5:5–15
3. Tafer H, Hofacker IL (2008) RNAPlex: a fast tool for RNA-RNA interaction search. *Bioinformatics* 24:2657–2663
4. Mneimneh S (2009) On the approximation of optimal structures for RNA-RNA interaction. *Trans Comput Biol Bioinform* 6:682–688
5. Andronescu M, Zhang ZC, Condon A (2005) Secondary structure prediction of interacting RNA molecules. *J Mol Biol* 345:987–1001
6. Bernhart S, Tafer H, Mückstein U et al (2006) Partition function and base pairing probabilities of RNA heterodimers. *Algorithms Mol Biol* 1–3
7. Dirks R, Bios J, Schaeffer JM et al (2007) Thermodynamic analysis of interacting nucleic acid strands. *Soc Ind Appl Math* 49:65–88
8. Rehmsmeier M, Steffen P, Hochsmann M et al (2004) Fast and effective prediction of microRNA/target duplexes. *RNA* 10: 1507–1517
9. Markham NR, Zuker M (2008) UNAFold: software for nucleic acid folding and hybridization. *Methods Mol Biol* 453:3–31
10. Alkan C, Karakoc E, Nadeau JH et al (2006) RNA-RNA interaction prediction and antisense RNA target search. *J. Comput Biol* 13:267–282
11. Mückstein U, Tafer H, Bernhart S et al (2009) Translational control by RNA-RNA interaction: Improved computation of RNA-RNA binding thermodynamics. *Bioinform Res Dev* 13:114–127
12. Mückstein U, Tafer H, Hackermuller J et al (2006) Thermodynamics of RNA-RNA binding. *Bioinformatics* 22:1177–1182
13. Busch A, Richter AS, Backofen R (2008) IntaRNA: efficient prediction of bacterial sRNA targets incorporating target site accessibility and seed regions. *Bioinformatics* 24: 2849–2856
14. Kato Y, Akutsu T, Seki H (2009) A grammatical approach to RNA-RNA interaction prediction. *Pattern Recogn* 42:531–538
15. Aksay C, Karakoc E, Kin Ho C et al. (2007) ncRNA discovery and functional identification via sequence motifs. Technical Report TR 1–9
16. Selim GA (1989) The design and analysis of parallel algorithms. ISBN 0-33-23005b-3
17. Seemann SE, Richter AS, Gesell T et al (2011) PETcofold: predicting conserved interactions and structures of two multiple alignments of RNA sequences. *Bioinformatics* 2:211–219
18. Chitsaz H, Salari R, Sahinalp SC et al (2009) A partition function algorithm for interacting nucleic acid strands. *Bioinformatics* 25:i365–i373
19. Li AX, Marz M, Qin J et al (2011) RNA-RNA interaction prediction based on multiple sequence alignments. *Bioinformatics* 4: 456–463

Part II

Analysis of High-Throughput RNA Sequencing Data

Chapter 8

Quality Control of RNA-Seq Experiments

Xing Li, Asha Nair, Shengqin Wang, and Liguo Wang

Abstract

Direct sequencing of the complementary DNA (cDNA) using high-throughput sequencing technologies (RNA-seq) is widely used and allows for more comprehensive understanding of the transcriptome than microarray. In theory, RNA-seq should be able to precisely identify and quantify all RNA species, small or large, at low or high abundance. However, RNA-seq is a complicated, multistep process involving reverse transcription, amplification, fragmentation, purification, adaptor ligation, and sequencing. Improper operations at any of these steps could make biased or even unusable data. Additionally, RNA-seq intrinsic biases (such as GC bias and nucleotide composition bias) and transcriptome complexity can also make data imperfect. Therefore, comprehensive quality assessment is the first and most critical step for all downstream analyses and results interpretation. This chapter discusses the most widely used quality control metrics including sequence quality, sequencing depth, reads duplication rates (clonal reads), alignment quality, nucleotide composition bias, PCR bias, GC bias, rRNA and mitochondria contamination, coverage uniformity, etc.

Key words Quality control, RNA-seq, High-throughput sequencing, Next-generation sequencing

1 Introduction

RNA-seq has led to a better understanding of the RNA universe by providing unprecedented opportunities to interrogate the transcriptome from different perspectives. These include gene expression profiling [1–4], new isoforms or alternative splicing identification and quantification [5–7], novel transcripts such as lncRNAs discovery [8–10], aberrant transcripts such as gene fusion identification [11–13], and variant calling [14–17]. However, current library preparation protocols of RNA-seq are still developing and possess several intrinsic biases and limitations, such as nucleotide composition bias, GC bias, and PCR bias, which could directly detriment many RNA-seq applications [18, 19].

In general, quality of RNA-seq experiments can be assessed at two different levels. Raw sequence based metrics, which check RNA-seq experiments at a “low level” because they do not require sequence alignments. These assessments include read (i.e., a consecutive sequence of nucleotides) quality, read duplication rate

(clonal reads), GC content, nucleotide composition bias, etc. However, raw sequence-based metrics largely focuses on evaluating the success of sequencing technologies and themselves alone cannot ensure the usability (biologic accuracy) of RNA-seq data. Therefore, checking RNA-seq data at a “higher level” is also imperative. These metrics include mapping statistics, coverage uniformity, saturation of sequencing depth, reads distribution over gene structure, ribosomal RNA contamination, reproducibility between biological replicates, etc.

1.1 Raw Sequence Quality

Phred quality score (Q) was originally developed by the program Phred to measure base-calling reliability from Sanger sequencing chromatograms [20, 21]. It is defined as $Q = -10 \times \log_{10}(P)$ where P is the probability of erroneous base calling. For example, a Phred quality score of 30 means the chance that this base is called incorrectly is 1 in 1,000. Although the Phred program is rarely used in next-generation sequencing field, Phred or Phred-like quality score has become widely accepted to characterize the quality of DNA sequences (see Note 1). Most often, Phred scores are reported as their corresponding ASCII characters (33–126 or “!” to “V”) (see Note 2 for FASTQ format), but SOLiD still uses numbers to represent quality scores.

There is no gold standard to tell if the quality of a particular sequence is good or bad, as this is really depending on the purpose of the study. For example, compared to expression profiling, variants calling tasks require much higher sequence quality. In general, scores over 30 indicate very good quality, 20–30 indicate reasonable good and <20 indicate poor quality. Parallel boxplots visualize “per nucleotide quality score” by summarizing Phred qualities for all reads at each position (Fig. 1a) [22, 23]. In addition, one can also calculate the average quality score per read (“per sequence quality score”) and check the quality score distribution of all sequences (Fig. 1b).

1.2 Nucleotide Composition and GC Content

GC content (or guanine-cytosine content) is the percentage of bases in a DNA sequence that are either guanine or cytosine. It is a simple way to measure nucleotide composition of DNA. The reason to use GC rather than AT (or AU in RNA) is that GC content carries more direct biologic meaning. GC pairs are more stable than AT (3 vs. 2 hydrogen bonds) which has implications in PCR experiments where the GC content of primers predicts their annealing temperature. Further, exons have much higher GC content than introns and intergenic regions and cytosine is the target of DNA methylation. People have found the dependence between read coverage and the GC content of reference genome in high-throughput sequence data. Therefore, evaluating GC content bias in RNA-seq data is of great importance to both transcript detection and abundance quantification [18].

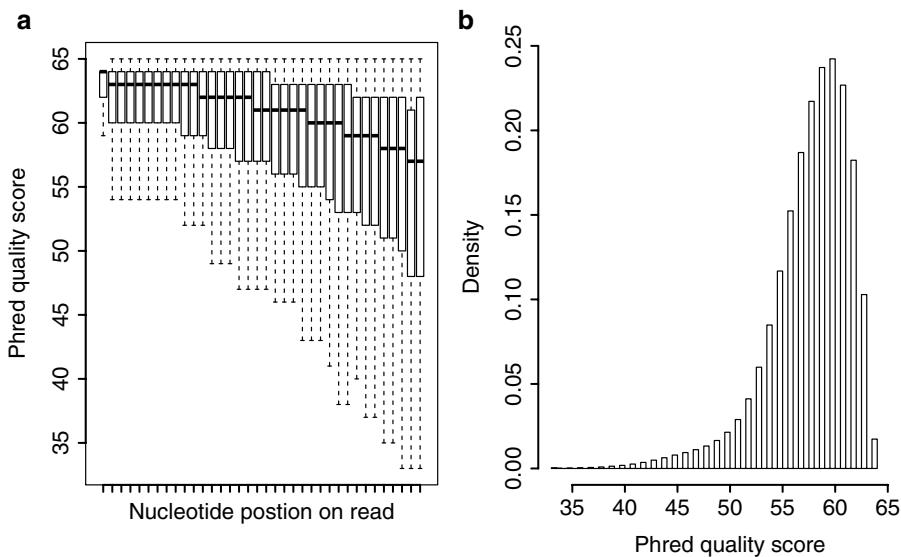


Fig. 1 (a) Parallel boxplot showing “per nucleotide quality score.” All reads are overlaid together, and then summarize Phred quality score (Y-axis) for each position of read from 5’ to 3’ end (X-axis). (b) “per sequence quality score” distribution. For each read, “per sequence quality score” is calculated as the average Phred quality score (X-axis) across all nucleotides

Assume RNA-seq reads were randomly sampled from expressed transcripts, when we pileup reads together and calculate the nucleotide composition (percentage of A, C, G, and T) at each positions or column, we expect little differences between columns. Random fluctuations will be cancelled out because the large sample size (i.e., hundreds of millions of reads). If we visualize nucleotide composition versus nucleotide positions in a diagram [22] the lines should be roughly flat at a value of 0.25 (Fig. 2a). In practical, the first 12 bases starting from 5’ end of reads exhibit large deviation from 0.25, this is due to the random hexamer priming during PCR amplification [19]. A serious bias indicates the existence of over-represented sequences, and such bias will influence coverage uniformity as well as transcripts abundance estimation. Per sequence GC content can be roughly used to measure the randomness of sequencing library as GC content of reads from random sequence library follows normal distribution with the mean equals to the overall GC content of the transcriptome (Fig. 2b). A poorly prepared or contaminated library will exhibit a skewed distribution.

1.3 Duplicate Sequences (PCR Duplication)

Read duplication rate is affected by read length, sequencing depth, transcript abundance and PCR amplification. Supposing the sequencing library is purely random and read length is 36 bp, the chance to get a duplicated read is $1/4^{72}$ (or 4.5×10^{-44}), this chance is still slim even if the sequencing depth reaches hundreds of millions.

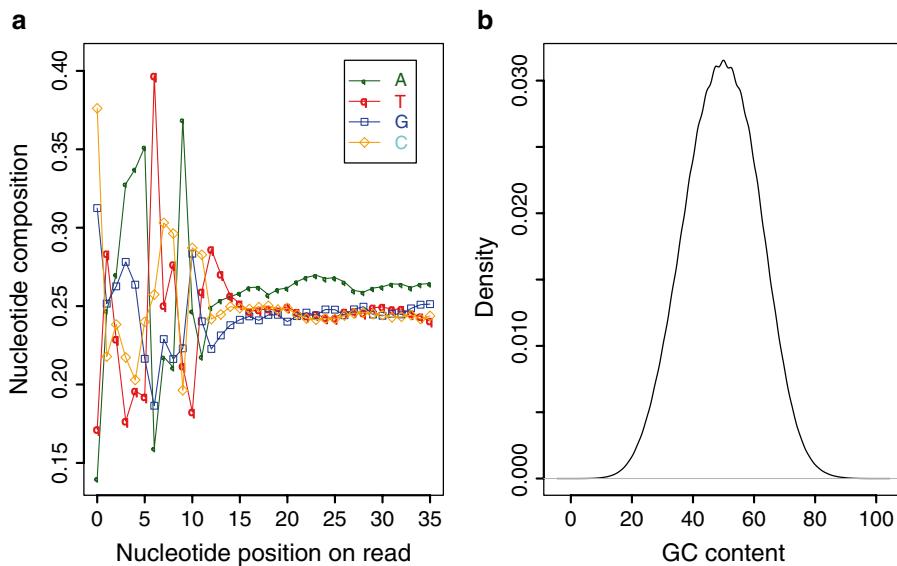


Fig. 2 (a) Diagram showing nucleotide composition bias at the beginning of reads. All reads are overlaid together, and then calculate nucleotide frequency (Y-axis) for each position of read (X-axis). Four nucleotides were indicated using different colors. (b) “per sequence GC content” distribution

Therefore, the majority of duplicated reads were artificially generated from PCR amplification [22]. And because of this, duplication rate is one way to check PCR amplification bias. To circumvent the huge memory requirement, tools such as FastQC will only track the first 200,000 short reads in each file for duplication level and creates a graph plotting the count of sequences with different degrees of duplication. By default, FastQC will raise a warning if there are more than 20 % of duplicated sequences in total and a failure if this number reaches over 50 % as the sequencing library is seriously biased and may not randomly sampling the target sequence.

1.4 Descriptive Statistics

Mapping statistics are the simplest and most intuitive way to assess if RNA sequencing was successful. These include mappability (number of reads aligned to reference genome), number of reads aligned to unique locations in the genome, and the number of splice mapped reads and number of reads mapped to mitochondria. It is difficult to derive reasonable or even empirical thresholds to determine if a particular RNA sequencing was successful or not, because these metrics really depend on read length, sequencing depth, bioinformatic analysis parameters, sample preparation protocol, and tissue type. For example, compared with shorter reads, longer reads will have better mappability, lower duplication rate, higher proportion that aligned to unique genome location, more spliced reads given the same sequencing depth. For the same sequencing depth and same read length, number of splice reads

may be dramatically different between two RNA-seq datasets simply because tissue origins are different. Muscle and heart tissues usually have much more mitochondria than other tissue types, and therefore mitochondria reads could be a problem if they account for a large proportion (i.e., >30 %), as this makes actual sequencing depth much lower than expected.

1.5 rRNA/tRNA Contamination

The goal of most RNA-seq studies is to interrogate functional message RNA (mRNA). However, structure RNAs such as Ribosomal RNA (rRNA) and transfer RNA (tRNA) are the most abundant RNA species and constitute 60–90 % of total RNA in a cell. To avoid having these RNAs dominate the sequencing data, it is necessary to remove these RNA species before preparing libraries for deep sequencing. Two approaches have been used to enrich mRNA. The first approach starts with total RNA that has been depleted of rRNA by using a set of oligos that binds to rRNA (such as RiboMinusTM), and the second method selects for transcripts by isolating poly-A RNA as the starting materials for the construction of sequencing libraries.

Even with ribosome depletion, a fair amount of ribosomal sequences may still remain in the raw data. Small amounts of rRNA contamination will not be a detriment to downstream analyses. However, a larger amount of ribosomal reads usually suggests rRNA depletion was inefficient or failed and additional sequencing may be necessary. Assessing rRNA contamination is straightforward; aligning reads to reference genome and then counts how many reads mapped to ribosome genes, or aligning reads directly to ribosomal RNA sequences.

1.6 Saturation Test of Sequencing Depth

RNA-seq experiments are diverse in their aims and design goals. The amount of sequencing needed for a given sample is determined by the goals of the experiment. For gene expression profiling, where we are interested to find quantitative differences of known genes between groups, modest sequencing depth is good enough (e.g., 30 million pair-end reads with length >30 bp for mammalian genomes). But for studies that involve investigation of alternative splicing, gene fusion detection and novel transcript identification, deeper read depths is required to be able to adequately cover not just the exons but also exon-exon junctions. It is recommended by ENCODE consortium that a minimum of 100–200 million 2×76 bp or longer reads is needed for mammalian genomes.

The saturation test is an approach to determine if current sequencing depth is deep enough to satisfy a particular purpose. It is fundamentally important because if sequencing was unsaturated, estimated gene expression metrics such as RPKM (Reads Per Kilobase exon per Million mapped reads) will be unstable and low abundant isoforms will be undiscovered. In practical, we resample 5, 10, ..., 100 % of the total mapped reads and RPKMs are

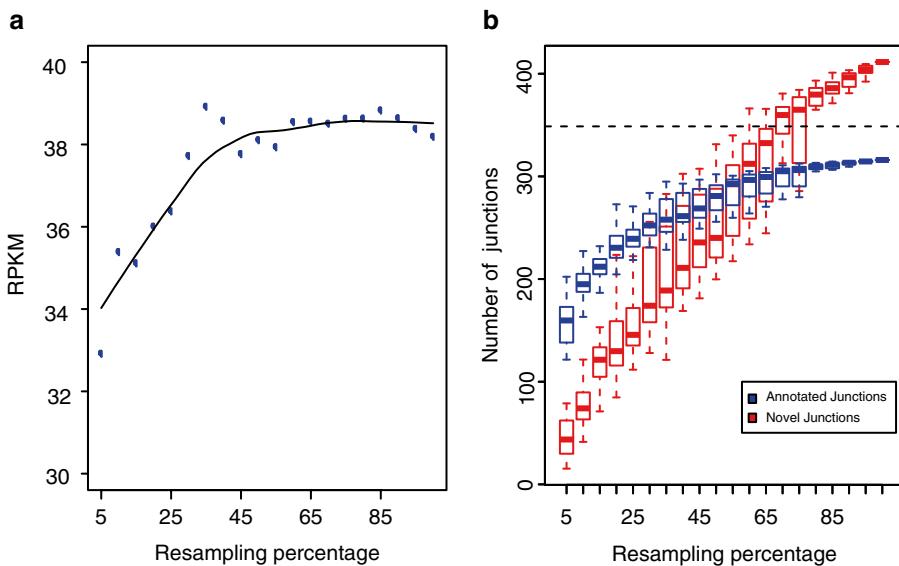


Fig. 3 Saturation test of sequencing depth. (a) Saturation test using RPKM (gene expression measurement). RPKMs were recalculated for each resampled subset (*blue dots*) to test if RPKM values enter a steady state (or saturated). (b) Saturation test using detected splice junctions (*blue*: annotated junction, *red*: novel junction). Horizontal dashed line indicates all annotated junctions encoded in reference gene models

subsequently recalculated using each subset. For a particular transcript, RPKM values may vary at the beginning with very small sample sizes, but finally could reach a plateau. The stable RPKM values indicate a saturated sequencing depth; otherwise, sequencing depth should be increased until RPKM values enter a stationary stage (Fig. 3a). Saturation test for splice junction is similar; splice junctions are detected for each resampled subset, and number of detected splice junctions will increase as the resample percentage increases, but finally will reach a fixed value. The junction saturation test is very important for alternative splicing analysis, as unsaturated sequencing depth would miss many low-abundance yet bona fide splicing junctions. Due to the sensitivity of RNA-seq, the number of identified novel splice junctions will increase as sequencing depth goes deeper, and therefore saturation rarely occurs for novel splice junctions even with billions of reads in mammalian genome (Fig. 3b).

1.7 Reproducibility Between Replicates

For RNA sequencing technology, depending on the goal of the experiment, replicates can be of two kinds, technical and biological. Technical replicates are replicates obtained from the same sample for purposes of studying batch effects and evaluating the technology, including background noise, differences in sequencing chemistry, instrument-to-instrument differences, etc. Biological replicates are the most desired form of replicates, as these provide us with the true variation among biological samples.

The use of such replicates comes into play for experiments that involve comparison of two or more groups for differential expression analysis. It is recommended to have at least two biological replicates per group in order to statistically determine the significantly differentially expressed genes.

Evaluating the reproducibility between replicates is straightforward. Most often scatter plots are used to visualize the reproducibility between expression measurements such as RPKM or FPKM (Fragment Per Kilobase exon per Million reads). Logarithm transformation of RPKM is necessary because of the large dynamic range of RPKM values. After logarithm transformation, expression values roughly follow a normal distribution and have a high Pearson's correlation coefficient (Fig. 4).

1.8 Coverage Uniformity

Gene body coverage describes the overall reads density over the mRNA regions (both UTR exon and CDS exon). Ideally, each base has the same chance to be sequenced, and each site within gene body has similar coverage. However, read density profiles can be affected by library preparation protocol, PCR amplification, RNA degradation, genome complexity and the underlying gene

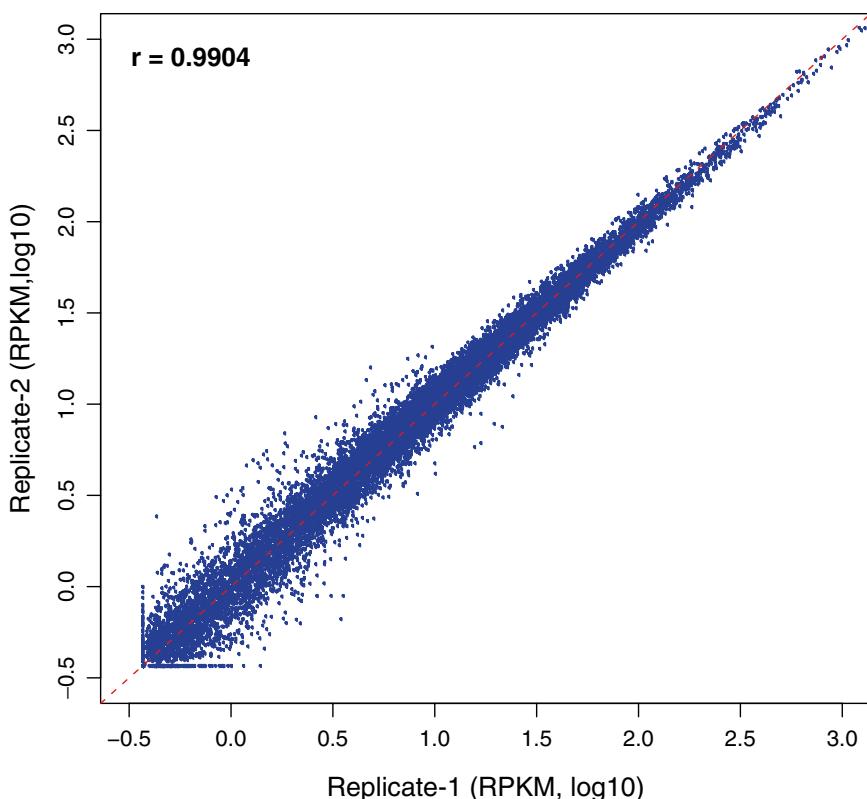


Fig. 4 Scatter plot showing reproducibility between two RNA-seq datasets (technical replicates). Each *blue dot* represents a gene, and the *red dashed line* is linear regression line

model used. For example, RNA-seq data using poly-A selection usually have higher coverage at 3' end. PCR amplification efficiency could be different for different DNA fragments site, this also introduce uneven coverage. Low DNA complexity (or repetitive) regions usually have higher coverage but this will depend on how multi-hit reads were processed. For RNA-seq data that are not strand specific, coverage profile is also affected by underlying reference gene model; for example, an uneven coverage occurs when two genes are overlapped in the genome and express differently. Coverage profile is the most intuitive way to check uniformity, by normalizing all annotated genes into the same scale, and then calculating coverage for each position (Fig. 5).

1.9 Reads Distribution (Intron, Exon, UTR, etc.)

After mapping reads to a reference genome, we can calculate the fraction of reads assigned to exons (including both UTR and CDS exons), introns and intergenic regions based on the provided gene model. In ideal conditions and for well-annotated organisms, most of reads in RNA-seq data should be mapped to exonic regions. However, in practice, a considerable amount of reads are mapped to intron or intergenic regions. Except for mapping artifacts, intergenic/intronic reads are mainly from DNA contamination, pre-mRNAs, new isoforms, or novel transcripts. Some UTR regions are overrepresented (i.e., higher reads density) because of DNA

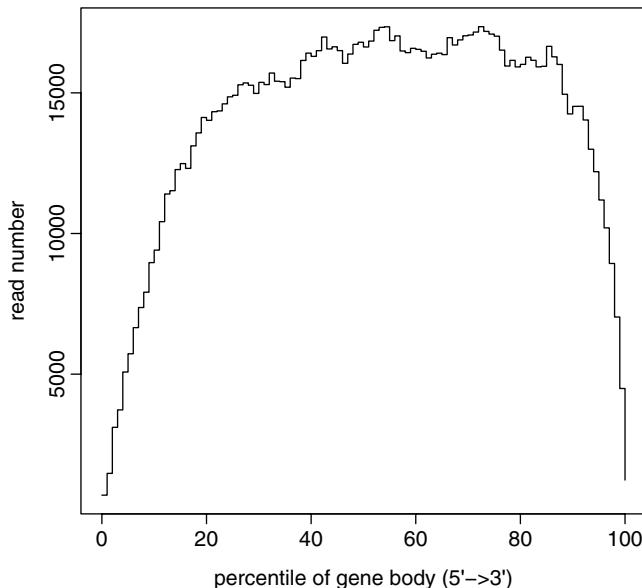


Fig. 5 Coverage uniformity over gene body. All transcripts were scaled into the same length (100 nucleotides) and then reads coverage (Y-axis) was calculated for each position (X-axis) from 5' to 3' end

repeats or PCR bias, but most often they have lower reads density due to RNA degradation. Specially, when poly-A RNA-seq protocol was used, reads are biased (i.e., overrepresented) in 3'UTR.

1.10 Strand Specificity

In mammalian genomes, many genes encoded on different strands are partially or completely overlapped. Therefore, conventional RNA-seq cannot decode the complex transcriptome due the lack of RNA polarity information. Strand specific RNA-seq experiments can overcome these limitations and is better suited for genome annotation (such as identify anti-sense noncoding RNA, demarcate the exact boundaries of adjacent genes transcribed on different strand), de novo transcriptome assembly, accurate digital gene expression and variant calling. Strandness is achieved by degrading one cDNA strand or by ligating distinct RNA adapters to the 5' and 3' ends of each RNA molecule prior to cDNA synthesis [24]. The specificity of a strand-specific protocol can be calculated by comparing the strand of reads to the strand of a reference gene. For example, if there is only one gene located in the forward strand in a particular region, one would expect the majority (>90 %) of reads mapped to this region will be forward reads (determined from protocol). Further, one can use spliced reads to measure strand specificity, because the strandness of spliced reads can be easily inferred from splicing motif (such as GT/AG).

2 Notes

1. Two different equations, standard Sanger format and Solexa/Illumina format (prior to v1.3) have been used to calculate Q [25]. After Illumina Pipeline v1.4, the quality scheme has been changed into standard Sanger scale.

$$Q_{\text{Sanger}} = -10 \times \log_{10} P; \quad Q_{\text{Solexa(prior_to_v1.3)}} = -10 \times \log_{10} \frac{P}{1-P}$$

2. FASTQ file stores nucleotide sequences and Phred qualities in the same file. For example:

```
@read_1 ← (read identifier)
CCGGCCCCAGGCTCCTGTCTCCCCCCCAGGTGTGGTGATGCCAGGCATG(sequence
+read_1 ← (read identifier, can be omitted)
@@;D?DADF=FHFBEBH+A2ADFGFH:;09?*/?BB=F)==FH3=@CH@?←quality
@read_2 ← (another read identifier)
CTGCTCTCTTGCTGATGGACAAGGGGGCATCAAACAGCTTCTCCTCTG
+read_2 ← (another read identifier, can be omitted)
CCCFHHHGHJJIJJJJJJJIJJJJJIJIJJFGIIJJIIIIIGJ
```

References

1. Mortazavi A, Williams BA, McCue K et al (2008) Mapping and quantifying mammalian transcriptomes by RNA-Seq. *Nat Methods* 5:621–628. doi:[10.1038/nmeth.1226](https://doi.org/10.1038/nmeth.1226)
2. Marioni JCJ, Mason CEC, Mane SMS et al (2008) RNA-seq: an assessment of technical reproducibility and comparison with gene expression arrays. *Gene Dev* 18:1509–1517. doi:[10.1101/gr.079558.108](https://doi.org/10.1101/gr.079558.108)
3. Wang Z, Gerstein M, Snyder M (2009) RNA-Seq: a revolutionary tool for transcriptomics. *Nat Rev Genet* 10:57–63. doi:[10.1038/nrg2484](https://doi.org/10.1038/nrg2484)
4. Wilhelm BT, Landry J-R (2009) RNA-Seq—quantitative measurement of expression through massively parallel RNA-sequencing. *Methods* 48:249–257. doi:[10.1016/j.ymeth.2009.03.016](https://doi.org/10.1016/j.ymeth.2009.03.016)
5. Wang ET, Sandberg R, Luo S et al (2008) Alternative isoform regulation in human tissue transcriptomes. *Nature* 456:470–476. doi:[10.1038/nature07509](https://doi.org/10.1038/nature07509)
6. Katz Y, Wang ET, Airoldi EM, Burge CB (2010) Analysis and design of RNA sequencing experiments for identifying isoform regulation. *Nat Methods* 7:1009–1015. doi:[10.1038/nmeth.1528](https://doi.org/10.1038/nmeth.1528)
7. Trapnell C, Williams BA, Pertea G et al (2010) Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation. *Nat Biotechnol* 28:511–515. doi:[10.1038/nbt.1621](https://doi.org/10.1038/nbt.1621)
8. Cabili MN, Trapnell C, Goff L et al (2011) Integrative annotation of human large intergenic noncoding RNAs reveals global properties and specific subclasses. *Gene Dev* 25:1915–1927. doi:[10.1101/gad.17446611](https://doi.org/10.1101/gad.17446611)
9. Guttmann M, Garber M, Levin JZ et al (2010) Ab initio reconstruction of cell type-specific transcriptomes in mouse reveals the conserved multi-exonic structure of lincRNAs. *Nat Biotechnol* 28:503–510. doi:[10.1038/nbt.1633](https://doi.org/10.1038/nbt.1633)
10. Prensner JRJ, Iyer MKM, Balbin OAO et al (2011) Transcriptome sequencing across a prostate cancer cohort identifies PCAT-1, an unannotated lincRNA implicated in disease progression. *Nat Biotechnol* 29:742–749. doi:[10.1038/nbt.1914](https://doi.org/10.1038/nbt.1914)
11. Kannan K, Wang L, Wang J et al (2011) Recurrent chimeric RNAs enriched in human prostate cancer identified by deep sequencing. *Proc Natl Acad Sci U S A* 108:9172–9177. doi:[10.1073/pnas.1100489108](https://doi.org/10.1073/pnas.1100489108)
12. Pflueger D, Terry S, Sboner A et al (2011) Discovery of non-ETS gene fusions in human prostate cancer using next-generation RNA sequencing. *Gene Dev* 21:56–67. doi:[10.1101/gr.110684.110](https://doi.org/10.1101/gr.110684.110)
13. Edgren H, Murumagi A, Kangaspeska S et al (2011) Identification of fusion genes in breast cancer by paired-end RNA-sequencing. *Genome Biol* 12:R6. doi:[10.1186/gb-2011-12-1-r6](https://doi.org/10.1186/gb-2011-12-1-r6)
14. Peng ZZ, Cheng YY, Tan BC-MB et al (2012) Comprehensive analysis of RNA-Seq data reveals extensive RNA editing in a human transcriptome. *Nat Biotechnol* 30:253–260. doi:[10.1038/nbt.2122](https://doi.org/10.1038/nbt.2122)
15. Bahn JHJ, Lee J-HJ, Li GG et al (2012) Accurate identification of A-to-I RNA editing in human by transcriptome sequencing. *Gene Dev* 22:142–150. doi:[10.1101/gr.124107.111](https://doi.org/10.1101/gr.124107.111)
16. Park EE, Williams BB, Wold BJB, Mortazavi AA (2012) RNA editing in the human ENCODE RNA-seq data. *Gene Dev* 22:1626–1633. doi:[10.1101/gr.134957.111](https://doi.org/10.1101/gr.134957.111)
17. Ramaswami G, Zhang R, Piskol R et al (2013) Identifying RNA editing sites using RNA sequencing data alone. *Nat Methods*. doi:[10.1038/nmeth.2330](https://doi.org/10.1038/nmeth.2330)
18. Benjamini Y, Speed TP (2012) Summarizing and correcting the GC content bias in high-throughput sequencing. *Nucleic Acids Res* 40:e72. doi:[10.1093/nar/gks001](https://doi.org/10.1093/nar/gks001)
19. Hansen KD, Brenner SE, Dudoit S (2010) Biases in Illumina transcriptome sequencing caused by random hexamer priming. *Nucleic Acids Res* 38:e131. doi:[10.1093/nar/gkq224](https://doi.org/10.1093/nar/gkq224)
20. Ewing B, Hillier L, Wendl MC, Green P (1998) Base-calling of automated sequencer traces using phred. I. Accuracy assessment. *Genome Res* 8(3):175–85
21. Ewing B, Green P (1998) Base-calling of automated sequencer traces using phred. II. Error probabilities. *Genome Res* 8(3):186–94
22. Babraham Bioinformatics – FastQC a quality control tool for high throughput sequence data. <http://www.bioinformatics.babraham.ac.uk/projects/fastqc/>
23. Wang L, Wang S, Li W (2012) RSeQC: quality control of RNA-seq experiments. *Bioinformatics*. Oxford, England. doi:[10.1093/bioinformatics/bts356](https://doi.org/10.1093/bioinformatics/bts356)
24. Levin JZ, Yassour M, Adiconis X et al (2010) Comprehensive comparative analysis of strand-specific RNA sequencing methods. *Nat Methods*. doi:[10.1038/nmeth.1491](https://doi.org/10.1038/nmeth.1491)
25. Cock PJ, Fields CJ, Goto N, Heuer ML, Rice PM (2010) The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants. *Nucleic Acids Res* 38(6):1767–71. doi:[10.1093/nar/gkp1137](https://doi.org/10.1093/nar/gkp1137)

Chapter 9

Accurate Mapping of RNA-Seq Data

Kin Fai Au

Abstract

The mapping of RNA-Seq data on genome is not the same as DNA-Seq data, because the junction reads span two exons and have no identical matches at reference genome. In this chapter, we describe a junction read aligner SpliceMap that is based on an algorithm of “half-read seeding” and “seeding extension.” Four analysis steps are integrated in SpliceMap (half-read mapping, seeding selection, seeding extension and junction search, and paired-end filtering), and all tuning parameters of these steps can be editable in a single configuration file. Thus, SpliceMap can be executed by a single command. While we describe the analysis steps of SpliceMap, we illustrate how to choose the parameters according to the research interest and RNA-Seq data quality by an example of human brain RNA-Seq data.

Key words Junction reads, Junction, SpliceMap, Half-read seeding, Seeding extension

1 Introduction

In most studies, mapping is the first step of RNA-Seq data analysis. RNA-Seq short reads can be mapped to reference genome or annotated transcriptome. The annotated transcriptomes such as RefSeq [1] and Ensembl [2] are the gene and gene isoform identifications accumulated from many previous research, but they are far from comprehensive even in human and mouse. Therefore, when RNA-Seq data is mapped to annotated transcriptome, some reads from undiscovered genes could be missed in the mapping process. Some others could be mapped to incorrect positions because the sequence similarity between undiscovered genes and annotated genes. The incorrect mapping would cause bias or even errors in the downstream analyses, such as abundance estimation. Because abundance estimation from sequencing data is determined by short read coverage. In many species, reference genome is better sequenced and more complete than transcriptome. Therefore, an accurate and comprehensive mapping of RNA-Seq data should be performed against reference genome.

There are two types of reads in RNA-Seq data: exon reads and junction reads. They should be considered separately when mapped

to reference genome. An exon read is generated from a single exon and thus it has identical match in genome. Therefore, exon reads can be mapped directly to genome by regular short read aligners, which is mostly used in DNA-Seq. A junction read spans two exons and thus its sequence is the combination of two exonic sequences separated by introns. Therefore, there is no identical match of junction read in the reference genome. Instead of regular short read aligners, we need a special aligner to map junction reads. Herein, we describe a detailed procedure of a aligner SpliceMap [3] to map junction reads to genome. Integrating with a regular short read aligner, such as Bowtie [4], ELAND (Cox, unpublished software), and SeqMap [5], SpliceMap maps both exon reads and junction reads to genome, without bias on the existing transcriptome annotations.

2 Material

2.1 RNA-Seq Dataset

In order to illustrate the procedure of SpliceMap, we use a human brain RNA-Seq data from Illumina's Human BodyMap 2.0 project. It was released with Ensembl (release 62). The RNA was from a 77-year-old Caucasian female. This RNA-Seq dataset contains 80,946,860 50 bp paired-end reads in FASTQ format. Each read in FASTQ format contains four lines:

```
@HWI-BRUNOP16X_0001:3:1:9465:1024#0/1
TTAACAGTCGAGAGTGTGCTGAGAACTTAGACGGGATTGGTAGGCCAAG
+HWI-BRUNOP16X_0001:3:1:9465:1024#0/1
TQFLTSSTTggggfJKTSUccccggggggggggggggf g f f g g g g e
```

The first line is the read name and begins with a symbol “@.” The second line is the raw sequence. The third line begins with a symbol “+” and it is the extra info that was added during sequencing or preliminary analysis. In this example, the third line is just a copy of the read name. The fourth line contains quality score of each base. The quality score encodes the probability that the corresponding base call is incorrect. There are three types of quality scores: phred-33, phred-64, and Solexa formats. Phred-33 format encodes Phred quality score from 0 to 93 using the ASCII characters from 33 to 126. Phred-64 format encodes Phred quality score from 0 to 62 using the ASCII characters from 64 to 126. Solexa format encodes Phred quality score from -5 to 62 using the ASCII characters from 59 to 126. Each character in quality score line presents an integer as a sequencing quality which can be converted to probability of false base call. Thus, please be aware that the same character in three different quality score format can encode different probabilities of false base call. Therefore, SpliceMap requires a proper setting of quality score format in the configuration file. Our human brain data set uses phred-64 format (*see Subheading 3*).

2.2 Reference Genome

In this chapter, we use hg19 as the reference human genome, which is in FASTA format, with each chromosome in a separate

file. These can be obtained from UCSC website <http://hgdownload.soe.ucsc.edu/downloads.html> (for hg19, the download link is <http://hgdownload.soe.ucsc.edu/goldenPath/hg19/bigZips/chromFa.tar.gz>). After unzipping, each chromosome is in its own .fa file. However, the “*_random.fa” and “*_hap*.fa” files can be deleted, as they are random chromosome segregations and haplotype sequences, respectively. Make sure all the genome files of different organisms are placed in separate directories.

In order to run SpliceMap with Bowtie, you also need to obtain the other kind of genome files: Bowtie index, which can be downloaded from <http://bowtie-bio.sourceforge.net/index.shtml> (download the same version as your genome files, probably UCSC). If the genome you are interested in is not listed, you may need to build your own index from the FASTA-formatted genome files by following the instructions at <http://bowtie-bio.sourceforge.net/tutorial.shtml>.

2.3 Computational Hardware

Mapping of high-throughput sequencing data requires high CPU and high memory usage. Linux or Unix-based computer is more efficient for this study. SpliceMap supports multi-threading, so multi-core computer cluster is recommended to use to decrease running time. The running speed is approximately proportional the number of threads used. The size of the raw data, intermediate files and final results (including coverage, junction detection and mapping results) is about 29, 32, and 33 GB for this example. Therefore, 100 GB of hard disk space is required for the mapping demonstration of this RNA-Seq data set. In general, the required disk space is proportional to the size of RNA-Seq, so the required storage space should be calculated accordingly. The memory usage varies at different steps of SpliceMap, but the peak usage is approximately proportional to the size of the reference genome. In this example, it requires at least 5 GB memory, but >10 GB is recommended.

2.4 Computational Software

SpliceMap uses a “half-read seeding” and “seeding-extension” algorithm to map junction reads. The first step is to find the alignment of the half read of junction read on genome. Therefore, a regular short read aligner is required. Bowtie, ELAND, and SeqMap are compatible with SpliceMap. In this chapter, Bowtie is used (version 0.12.7, complied by gcc version 4.1.2 20080704). The precompiled executable of Bowtie and prebuilt index of human hg19 genome can be downloaded from <http://bowtie-bio.sourceforge.net>.

SpliceMap is an integrated pipeline to map both exon reads and junction reads. The mapping of each read is independent, so more available threads can allow parallel mapping processes and reduce running time. SpliceMap 3.3.5.2 is used to in this chapter. You can download the source code and the precompiled executable files from SpliceMap homepage (www.stanford.edu/group/wonglab/SpliceMap/download.html). If the executive files are not compatible with your computer system, you can compile them from the source code on your own computer (*see Note 1*).

3 Methods

A junction read spans two exons and thus its sequence can be split into two exonic fragments. The mapping of junction read is the process to find where to split junction read and how to map two exonic fragments to genome. It is intuitive that either exonic fragment is longer or equal to the half length of original junction reads. Therefore, either half of a junction read comes from a single exon, and thus can be directly mappable to genome. This mapping hit becomes a seeding to narrow the search region of the remaining sequence. The second part is to extend the seeding alignment to complete the mapping of the corresponding exonic fragment and then find the mapping of the other exonic fragment in a small neighbor region on the same chromosome (Fig. 1a). SpliceMap contains four steps: half-read mapping, seeding selection, seeding extension and junction search, and paired-end filtering (Fig. 1b).

SpliceMap integrates all steps in an easy one-button tool. The parametric settings of each step are simply editable in the configuration file “run.cfg.” In order to use SpliceMap on your own data, you should follow these protocol:

1. Obtaining the genome files in the format “chr1.fa, chr2.fa, …” and also the corresponding Bowtie index (*see* Subheading 2.2).
2. Create an empty directory, this will be the working directory.
3. Copy “run.cfg” from the SpliceMap package to the working directory.
4. Edit run.cfg to include paths to your data files and genome directories.
5. Edit the parametric settings for each steps, according to the data quality and research interest
6. Execute “runSpliceMap run.cfg” while in your working directory.
7. After a certain time execution will conclude. You can find results in the “output” directory.

At first, the information of SpliceMap input (data and genome) should be edited correctly in run.cfg. The location of the reference genome is required:

```
# Directory of the chromosome files in FASTA format
# Each chromosome should be in a separate file (can be concatenated)
# ie. chr1.fa, chr2.fa, ...
# (single value)
genome_dir = /home/kinfai/genome/hg19/
```

Also the names of chromosome files can be defined optionally:

```
# Name of the chromosome file with wildcards
# If none is given the default of "chr*.fa" will be used [this is compatible with the UCSC genome files]
# If you genome file is concatenated, just type the file name
```

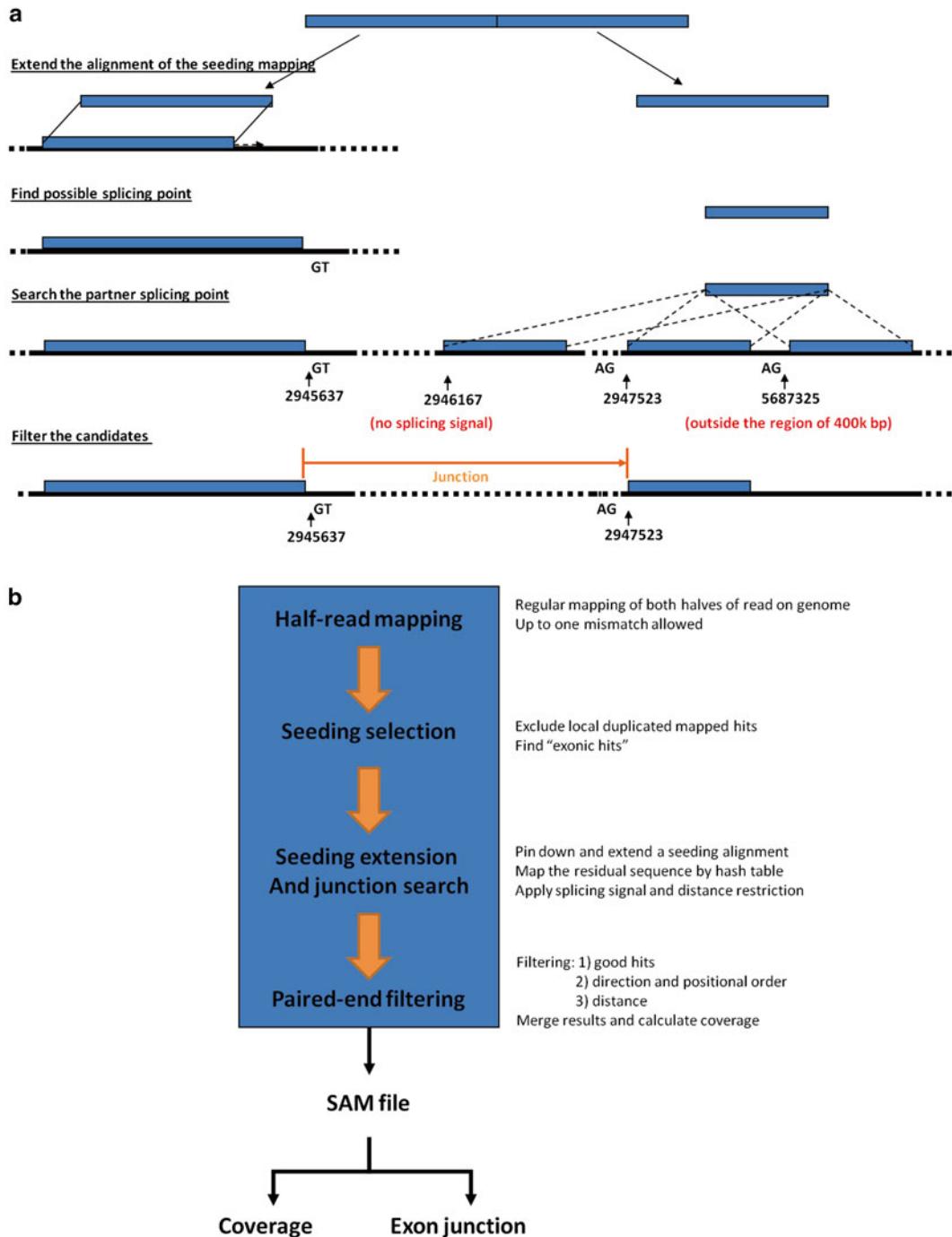


Fig. 1 (a) SpliceMap flowchart and (b) SpliceMap algorithm

```
# (optional)
# (single value)
chromosome_wildcard = chr*.fa
```

The other input of SpliceMap is the RNA-Seq data. The location, data format and the quality format (*see* Subheading 2.1) should be defined (*see Note 2*):

```
# These are the two lists of sequencer reads files.
# "reads_list2" can be commented out if reads are not
paired-end.
# Make sure the order of both lists are the same!
# Also, "reads_list1" must be the first mate of the
pair of reads.
# Note: pair-reads should be in the "forward-reverse" format.
# (multiple values)
> reads_list1
/scratch1/0-2-kinfai/50bp_mRNA_Seq/FCA/s_3_1_sequence.txt
<
> reads_list2
/scratch1/0-2-kinfai/50bp_mRNA_Seq/FCA/s_3_2_sequence.txt
<

# Format of the sequencer reads, also make sure reads are
# not split over multiple lines.
# Choices are: FASTA, FASTQ, RAW
# (single value)
read_format = FASTQ

# Format of the quality string if FASTQ is used
# Choices are:
# phred-33--Phred base 33 (same as Sanger format) [default]
# phred-64--Phred base 64 (same as Illumina 1.3+)
# solexa --Format used by solexa machines
# (single value)
quality_format = phred-64
```

The other parametric settings in run.cfg are to tune the performance of four steps of SpliceMap. It is illustrated in the following subsections by the example of human brain RNA-Seq data. Some parameters are required while some others are optional. If optional parameters start with the symbol “#,” they are commented out.

3.1 Half-Read Mapping

Many Second Generation Sequencing platforms provide reads that are not shorter than 50 bp. Thus, the half reads are not shorter than 25 bp, which are long enough for reliable alignment on genome by regular short read aligners. SpliceMap is compatible with Bowtie, ELAND, and SeqMap for this mapping. In this example, we select Bowtie:

```
# The short reads mapper used
# choices are "bowtie", "eland" or "seqmap"
# Bowtie is recommended
# this can be commented out if the mapping has already
been done and the
```

```
# appropriate ".t" files exist in the temp directory
[advanced].
# (single value)
mapper = bowtie
```

The tolerance of mismatches should be set based on the data quality (please *see* Chap. 8). Consider SNP and the other sequence variants between target sample and reference genome, the tolerance of mismatch on half-read seedings set to 1. However, if the qualities of some base positions are very low, this value (seed_mismatch) could be higher. As most current Second Generation Sequencing platform provides high-quality data, this number is not bigger than two generally.

```
# Maximum number of mismatches allowed in seeding for
junction search
# Choices are 0,1(default) or 2
# (optional)
# (single value)
seed_mismatch = 1
```

Low-quality bases or specific sequence (e.g., barcode or adaptor) may exists at two ends of some datasets, they should be clipped off by the following setting:

```
# Full read length
# SpliceMap will only use the first "full_read_length"
bp for mapping.
# If the read is shorter than "full_read_length", the
full read will be used before head clip.
# If you comment this out SpliceMap will use as many
as possible.
# This is for the case where the reads might have N's
at the end.
# It is always desirable to cut off the N's
# (optional)
# (single value)
full_read_length = 50

# Number of bases to clip off the head of the read
# This clipping is applied after "full_read_length"
# (optional)
# (single value)
head_clip_length = 0
```

In this example, there is no clipping of base, because the data qualities are high at all positions.

For instance, if we want to cut the last two bases off, we should set:

```
full_read_length = 48
```

If we want to cut the first three bases off, we should set:

```
head_clip_length = 3
```

Bowtie always requires a reference genome index for mapping. You can either download a prebuilt index or build the own index by (see Subheading 2.2). The path of the Bowtie index is required:

```
# Base of bowtie index, this should be the same genome
# as the
# chromosome files
# eg. if you bowtie files are "genome/hg18/genome.1.ewbt", ?
# then your base dir is "genome/hg18/genome"
# (single value)
bowtie_base_dir = /home/kinfai/program/tophat-1.3.2.
Linux_x86_64/index/Homo_sapiens/UCSC/hg19/Sequence/
BowtieIndex/genome
```

Bowtie can utilize multiple threads to speed up the mapping process. This can be set in the run.cfg as well:

```
# Number of threads to use for mapping
# Default value is 2
# (optional)
# (single value)
num_threads = 6
```

Bowtie also has an option to “try hard” to find as many hits as possible but it is slightly slower than regular mode. Thus, this is optional according to the research interest.

```
# Try hard?
# choices are "yes" or "no"
# Default value is yes. (it is not much slower, about 15%)
# I'm unsure if this is required, feel free to try with
# this option off and let me know your results.
# (optional)
# (single value)
try_hard = yes
```

3.2 Seeding Selection

The mapping hits of the half reads are used to narrow the search region of junction read mapping. But not all mapping hits of half reads are used. Because there exist many repeat regions in genome, especially mammalian genome, a half read may be mapped to multiple places. If all mapping hits are passed to the downstream mapping process, some half reads with extremely many repeats could increase the computational intensity greatly. To balance the running time and sensitivity, we need to set a tolerance limit of multiple mappability, which is determined by the complexity of the target genome. According to the experience of human and mouse RNA-seq analyses, we set the number of multi-hits not higher than 10:

```
# Maximum number of multi-hits
# If a 25-mer seed has more than this many multi-
# hits, it will be discarded.
# (optional) Default is 10
# (single value)
max_multi_hit = 10
```

If two repeat regions are very closed, then the half-read mapping hits from two regions could be likely crossed over each other in the downstream mapping process. This could introduce many incorrect mapping hits of junction reads. We need to set a constraint of the distance between mapping hits from the same half reads. Since the downstream mapping process is only performed within a region of 99.9th percentile of intron size (400,000 bp in human, *see* Fig. 2 and Subheading 3.3) around the half-read mapping hits, we exclude those hits that are within this region:

```
# Maximum intron size, this is absolute 99th-percentile maximum.
# Introns beyond this size will be ignored.
# (optional) If you don't set this, we will assume a mammalian genome (400,000)
# (single value)
max_intron = 400000
```

3.3 Seeding Extension and Junction Search

Half-read mapping hit is extended base by base until reach the canonical splice signal GT-AG. The extended hit is considered as a possible hit of one exonic fragment of junction read. Then, the other exonic fragment is searched in the neighbor regions. Two exonic fragments of a junction read are separated by an intron, so the search region is narrowed down to the 99.9th percentile of intron size (400,000 bp in human, *see* Fig. 2 and Subheading 3.2).

Based on the data quality, we set a mismatch tolerance of the entire read mapping. This value should be determined by the error rate (please *see* the chapter “Quality controls of massive RNA sequencing data”) but should not be smaller than seed_mismatch (*see* Subheading 3.1). If seed_mismatch is bigger than read_mismatch, then SpliceMap would output null result.

```
# Maximum number of mismatches allowed in entire read
# No limit on value, however SpliceMap can only identify reads with
# a maximum of 2 mismatches per 25bp.
# Default is 2.
# (optional)
# (single value)
read_mismatch = 2
```

Some very long read may have a few low-quality bases at two ends, while the remaining part are long and can be mapped reliably. In order to rescue these reads from the constraints of read_mismatch, an extra tolerance of unmappable termini is set. This parameter is different from the clipping parameters in Subheading 3.1. The base clipping should be used if some terminal bases are of very low qualities for majority of reads. If only small part of reads has quality problem at their terminus (especially for very long reads), then max_clip_allowed is used instead. In this chapter we use 50 bp reads as an example, so this optional parameter is commented out by starting with a symbol “#”:

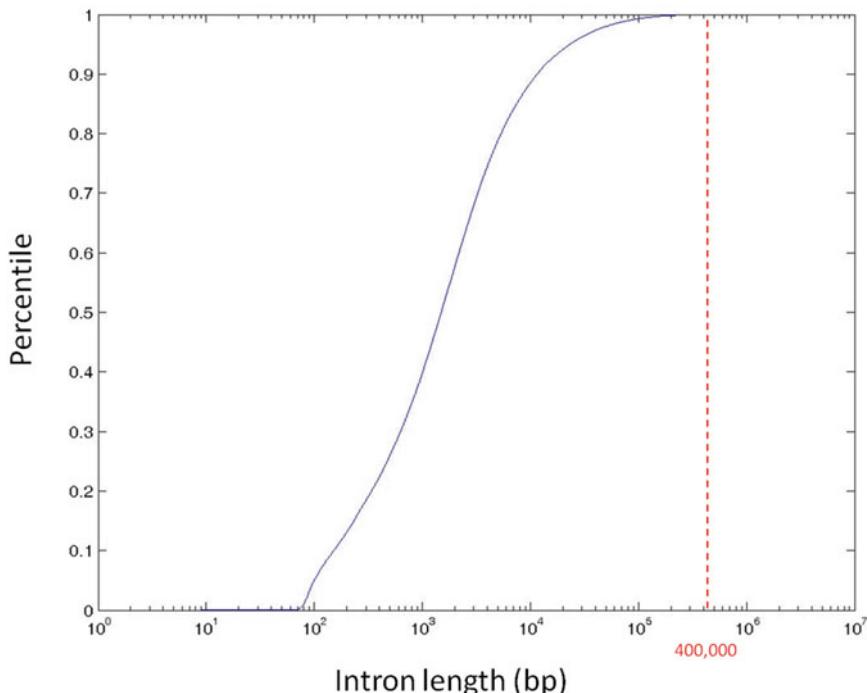


Fig. 2 The cumulative distribution function of intron lengths in human genome (RefSeq annotation)

```
# Maximum number of bases allowed to be soft clipped
# from the ends of reads during
# alignment. This is required as mismatches near
# junctions could cause parts of a
# a read to not map.
# Default is 40.
# (optional)
# (single value)
# max_clip_allowed = 40
```

The downstream processes after “Half-read Mapping” are performed chromosome by chromosome. Thus, multiple cores can be utilized to parallelize the mapping. This could require more memory but reduce the running time:

```
# Number of chromosomes to process at once, to take
# advantage of multi-core systems.
# This is not threading, so it will take extra mem-
# ory. However, running 2 at a time should be fine.
# (optional) Default = 1
# (single value)
num_chromosome_together = 3
```

3.4 Paired-End Filtering

Paired-end information can help to exclude some false mapping hits. If a paired-end data is input, the paired-end filter is applied by default. At first, three types of “good hits” are defined as: if two halves of a full read are mapped to two locations that differ by

exactly the half-read length, this mapping hit of the full read is called “exonic hit.” If a half-read mapping can be extended to reasonable long (not shorter than full length minus 10), this mapping hit is defined as “extension hit.” If a read can be mapped as a junction read by the processes above, then it is a “junction hit.” Three constraints are used for paired-end filtering:

1. Both hits from a pair of reads are good hits.
2. The mapping hits from a pair of reads are at the same chromosome and not separated by the 99.9th percentile of intron size (400,000 bp in human).
3. The directions and the positional order of the mapping hits from a pair of reads are consistent to the sequencing platform setting.

If both mapping hits of two mates of a pair of reads pass three constraints, then they are reported in a SAM file.

3.5 Mapping Results: SAM File

The mapping results are written in a SAM (The Sequence Alignment/Map) file “good_hits.sam” by SpliceMap (*see Note 3*). If a read is multiply mapped, then there will be multiple entries in the SAM file. Each line of the file good_hits.sam contains the following columns, separated by tab:

QNAME | FLAG | RNAME | POS | MAPQ | CIGAR | MRNM | MPOS | ISIZE | SEQ | QUAL | OPTIONAL

where QNAME is the name of each query copied from the FASTA or FASTQ file. A unique index is attached to the name;

FLAG is an integer that represents the information of read. The explanation of each flag can be found at <http://picard.sourceforge.net/explain-flags.html>;

RNAME is the name of reference sequence that the mapping hit locates. For example, the reference sequences of hg19 has chromosome names as “chr1, chr2,...”;

POS is the mapping location;

MAPQ is 255 if uniquely mapped and 0 if multiply mapped;

CIGAR is the mapping pattern (details can be found in <http://samtools.sourceforge.net/SAM1.pdf>);

MRNM and MPOS are the name of reference sequence and position that the mapping hit of the mate read (the other read of the pair) locates.

ISIZE is the distance between the mapping hits of two mates of the paired-end reads.

SEQ and QUAL are the raw sequence and the mapping quality.

OPTIONAL contains the following items:

XS—Strand of junction

XC—Number of bases clipped

NM—Number of mismatches in the read

MD—Describes location of mismatches

NH—Number of multi-hits

Here are the mapping hits of two mates of paired-end reads:

```
HWI-BRUNOP16X_0001:3:42:4915:16807#0[165352140] 65
chr1 14630 0 50M = 15006 327
TGGCTGTGTCATGTCAGAGCAATGGCCCAAGTCTGGGTCTGGGGGGAA
DFEEBADCD@<=:C@;>>45555FDDFFGGFGG@>?7?DFDFD@###
MD:Z:23C26 NM:i:1 XC:i:0 NH:i:5
HWI-BRUNOP16X_0001:3:42:4915:16807#0[165352140] 129
chr1 15006 0 33M757N17M = 14630 327
ACATCAAGGCCACCTTGGCTCGTGGCTCTCACTTGCTCCTGCTCCTC
@>>>AAAAA=;;=@ADA<@;@=8@A?A<25555DCFB?######
XS:A:-MD:Z:8TG40 NM:i:2 XC:i:0 NH:i:8
```

The FLAG's are 65 and 129, which mean that they are paired-end reads and they are the first read and the second read respectively. The CIGAR string of the first read is “50M,” so all 50 bases are mapped together. Thus it is an exonic read. The CIGAR string of the second read is “33M757N17M,” so the first 33 bases are mapped together while the remaining 17 bases are mapped together at a place 757 bases downstream. Thus, this is a junction read. Both MAPQ's are 0, so they are mapped to multiple places and the other hits are reported in additional lines. In the SAM file, some lines that begin with the symbol “@” are not the mapping hits but just additional information.

3.6 Junction Detection from Junction Read Alignment

The mapping hits of junction reads can be converted to junction detection and reported in a bed file “junction_color.bed.” The bed format is described in <http://genome.ucsc.edu/FAQ/FAQformat.html#format1>. SpliceMap tags each junction with a snippet of text describing the reliability of the junction (the fourth column in bed file). The format of this snippet is

$$(nR)[width_nNR](nUR/nMR)$$

nR—Number of reads supporting this junction.

width—Range of different right lengths supporting this junction, the larger the better.

nNR—Number of nonredundant reads supporting this junction.

nUR—Number of uniquely mappable reads supporting this junction.

nMR—Number of multiply mappable reads supporting this junction.

Using the parameters above, a number of junction filters packaged with SpliceMap can control the specificity of the outputted junctions. Different combinations of filters can be applied to tone the balance of specificity and sensitivity. These filters only work on the junctions (.bed files).

NnrFilter is the strictest filter. It filters junctions based on the number of nonredundant supporting reads. Nonredundant is defined as a read that is not mapped to exactly the same chromosome position. Empirical results show that using $nNR \geq 2$, will provide a specificity of about 90 %. The usage is:

```
nnrFilter infile.bed outfile.bed limit
```

where “limit” is the number of nonredundant supporting reads. Suggested value is 2.

UniqueJunctionFilter removes junctions which are solely supported by multiply mapped reads. That is, junction can remain if it has at least one uniquely mapped supporting read. Only junctions with $nNR = 1$ are targeted by this filter. The usage is:

```
uniqueJunctionFilter infile.bed outfile.bed
```

NeighborFilter removes “lonely junction,” which is defined as a junction with no exonic reads nearby. This filter seems to be able to remove the few artifact junctions that appear in the middle of nowhere. However, you may not want to use this filter if you are looking for low coverage junctions. Only junctions with $nNR = 1$ are targeted by this filter. The usage is:

```
neighborFilter good_hits.sam infile.bed outfile.bed [limit]
```

where “limit” defines the number of nucleotides upstream and downstream to check exonic reads nearby. The default value is 80 bp.

3.7 Results from Human Brain RNA-Seq Data

Using the settings above (6 threads for Bowtie and 3 cores for SpliceMap), it took 69,580.6 s (19 h) to finish all SpliceMap steps. In total, 134,961,718 mapping hits from 127,349,786 reads are reported in `good_hits.sam`. The mappable rate is 78.66 % ($127,349,786 / (2 \times 80,946,860)$). Then, 193,127 nonredundant junctions (`junction_color.bed`) are found from these mapping hits. 190,487 junctions pass the filter `uniqueJunctionFilter`:

```
uniqueJunctionFilter junction_color.bed junction_nUM_color.bed
```

Or we can apply `nnrFilter`:

```
nnrFilter junction_color.bed junction_nNR_color.bed 2
```

139,190 junctions remain after the nonredundant read filter.

4 Notes

1. After extracting `SpliceMap3352_linux-64.zip`, the folder should contain the following files and folders:

```
INSTALL src LICENSE bin run.cfg
```

Try running “./bin/runSpliceMap.” If you see the following output, the precompiled SpliceMap executive files are compatible with your computer and you may skip this step.

```
SpliceMap3352_linux-64 moo$ ./bin/runSpliceMap
===== Welcome to SpliceMap 3.3.5.2 (55) =====
Developed by Kin Fai Au and John C. Mu      http://
www.stanford.edu/group/wonglab/SpliceMap/

usage: ./runSpliceMap run.cfg
run.cfg -- Configuration options for this run, see
comments in file for details
See website for further details
```

However, if you see something like

```
SpliceMap3352_linux-64 $ ./bin/runSpliceMap
./bin/runSpliceMap:          /usr/lib/libstdc++.so.6:
version `GLIBCXX_3.4.9' not found (required by ./bin/runSpliceMap)
./bin/runSpliceMap:          /usr/lib/libstdc++.so.6:
version `GLIBCXX_3.4.11' not found (required by ./bin/runSpliceMap)
```

Then the C++ standard libraries in your Linux distribution are not compatible and you need to build SpliceMap from source code by following these instructions (for 64-bit systems):

- (a) Navigate to the “src” directory in the SpliceMap folder.
- (b) Type “./install.sh bin,” this will install SpliceMap into the example bin directory “bin.” Of course, you can install it anywhere you like in future.
- (c) Type “./install-bowtie.sh bin,” this will install Bowtie into the example bin directory
or these instructions (for 32-bit systems):
 - (a) Navigate to the “src” directory in the example folder.
 - (b) Type “./install-32.sh bin,” this will install SpliceMap into the example bin directory.
 - (c) Type “./install-bowtie-32.sh bin,” this will install Bowtie into the example bin directory.

You may copy the SpliceMap executive files to any location as long as all the executive files (including Bowtie) are in the same directory or path.

2. A single set of paired-end data always contains two files which are indexed by “1” and “2.” In the configuration file run.cfg, two files should be listed under “> reads_list1” and “> reads_list2,” respectively. If multiple sets of paired-end data are

input, the files listed under “> reads_list1” and “> reads_list2” should be in the consistent order. Here is an example:

```
# These are the two lists of sequencer reads files.
# "reads_list2" can be commented out if reads are
# not paired-end.
# Make sure the order of both lists are the same!
# Also, "reads_list1" must be the first pair.
# Note: pair-reads should be in the "forward-
# reverse" format.
# (multiple values)
> reads_list1
/scratch1/0-2-kinfai/50bp_mRNA_Seq/FCA/s_3_1_
sequence.txt
/scratch1/0-2-kinfai/50bp_mRNA_Seq/FCA/s_4_1_
sequence.txt
/scratch1/0-2-kinfai/50bp_mRNA_Seq/FCA/s_1_1_
sequence.txt
<
> reads_list2
/scratch1/0-2-kinfai/50bp_mRNA_Seq/FCA/s_3_2_
sequence.txt
/scratch1/0-2-kinfai/50bp_mRNA_Seq/FCA/s_4_2_
sequence.txt
/scratch1/0-2-kinfai/50bp_mRNA_Seq/FCA/s_1_2_
sequence.txt
<
```

3. In the output folder, there is a subfolder “debug_logs” which contains a few log files that record the working status of each step. You can learn the performances of each step and each chromosome from them. Thus, they are helpful for debugging if there are bugs and errors while SpliceMap is running.

References

1. Pruitt KD, Tatusova T, Maglott DR (2005) NCBI reference sequences (RefSeq): a curated non-redundant sequence database of genomes, transcripts and proteins. *Nucleic Acids Res* 33: 501–504
2. Curwen V, Eyras E, Andrews TD, Clarke L, Mongin E, Searle SM, Clamp M (2004) The Ensembl automatic gene annotation system. *Genome Res* 14:942–950
3. Au KF, Jiang H, Lin L, Xing Y, Wong WH (2010) Detection of splice junctions from paired-end RNA-seq data by SpliceMap. *Nucleic Acids Res* 38:4570–4578
4. Langmead B, Trapnell C, Pop M, Salzberg SL (2009) Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biol* 10:R25
5. Jiang H, Wong WH (2008) SeqMap: mapping massive amount of oligonucleotides to the genome. *Bioinformatics* 24:2395–2396

Chapter 10

Quantifying Entire Transcriptomes by Aligned RNA-Seq Data

Raffaele A. Calogero and Francesca Zolezzi

Abstract

Massive Parallel Sequencing methods (MPS) can extend and improve the knowledge obtained by conventional microarray technology, both for mRNAs and noncoding RNAs. Although RNA quality and library preparation protocols are the main source of variability, the bioinformatics pipelines for RNA-seq data analysis are very complex and the choice of different tools at each stage of the analysis can significantly affect the overall results. In this chapter we describe the pipelines we use to detect miRNA and mRNA differential expression.

Key words Coding genes, Differential expression, Annotated genes, RNAseq workflow

1 Introduction

In less than a decade RNA-seq changed the way of doing transcriptome analysis extending remarkably the knowledge derived by this kind of studies [1–6]. However, RNA-seq technology is still under evolution for both the wet and the dry sides. Now commercially available RNA-seq kits allow the analysis of the coding transcriptome of a single cell, directional sequencing that allows the quantification of transcripts located on the opposite strand of the same chromosomal locus, etc. Nevertheless, there are various criticalities in a RNA-seq experiment: (1) the biological question, (2) the experimental design, (3) the sequencing coverage, (4) the bioinformatics pipeline. Each of these points needs to be carefully addressed to obtain biologically meaningful results.

1.1 RNA-Seq Criticalities

The biological question is the first and probably the most important point to be considered. Although RNA-seq provides a massive amount of information if the experimental setting is not driven by a clear biological question, the extraction of valuable biological knowledge could be very challenging. The biological question is very tightly connected with the experimental design, since only in

the case of a clear biological question it is possible to correctly design the experiment. On the contrary, if the aim of the experiment is very vague (the so-called “fishing expedition experiment”), it will be very difficult to design the RNA-seq experiment correctly. While designing an experiment we need to keep in mind the familiar expression; “garbage in garbage out.” Sequencing coverage and replications are other critical issues. It is tricky to define which would be the ideal number of reads for any experiment as well as the number of biological replicates needed. The number of reads required for each experiment is highly dependent on the aim of the experiment itself. Estimating differential expression at gene level will need a much lower number of reads compared the number of reads needed to search for alternative splicing events or for the detection of aberrant genomic rearrangement leading to chimeric transcripts. However, if the sequencing depth is not sufficient to achieve the aim of the experiment, one of the advantages of RNA-seq is the possibility to increase, in a second time, the number of reads associated to one or more specific sample simply by running again the same library. This is feasible since the amount of cDNA library produced is not a limiting factor and the technical bias introduced during the sequencing procedure is negligible with respect to the biological variance.

It is evident that, as in gene expression microarray experiments, also in RNA-seq experiments the biological variability has to be taken into consideration [7]. Thus biological replicates are indeed needed. Looking at published RNA-seq experiments frequently the number of replications is very limited, e.g., two biological replicates for each experimental condition. This small number of replicates limits the statistical power of the experiments and, as a consequence, the statistics designed for RNA-seq are strongly biased by the lack of a good estimation of the biological variance [8].

A well known bottom neck is given by the complexity of the bioinformatics pipelines used for the RNA-seq analysis. We have highlighted that, in case of miRNA-seq, it is extremely important to have a clear insight of the tools used at each step of the pipeline [4]. This issue can be extended to any other RNA-seq application. Normalization and statistical tools are, for instance, a critical issue for differential expression analysis at gene level [8, 9]. A critical issue is also the detection of alternative splicing events. This field is still in an early phase because, in spite of the growing number of published methods on the detection of alternative splicing events, we know very little about the strengths and weaknesses of the different approaches. What at the moment is missing is a benchmark experiments that allow an efficient comparison among different pipelines.

In this chapter we present the approaches we use in our laboratory for miRNA-seq and gene/transcript-level RNA-seq.

2 Materials

2.1 Deep Sequencing Data

All the cDNA libraries were subjected to indexed sequencing run on an Illumina HiSeq 2000. We used 51 nts single end reads for miRNA-seq because the use of 50 nts long reads makes easier the detection and removal of linkers. In the case of mRNA-seq we use pair end sequencing runs of 2×51 cycles as the best compromise between sequencing cost and sequence specificity. Sequence directionality can provide valuable information specifically in the case whole transcriptome analysis is performed, i.e., rRNA depletion followed by sequencing of coding and noncoding RNAs.

2.2 Computational Hardware

Being a relatively small lab we preferred a multicore solution with respect to a cluster solution for RNA-seq analysis since management of multiple cores on a single machine is easier with respect to the management of a cluster solution. We run our analyses on two AMD machines (64/48 cores), 512 Gb RAM each running linux SUSE Enterprise 11/10. We decided also to invest significantly in storage. Our storage is made of 30×2 Tb SATA disks and 6×2 Tb hot spare disks configured as RAID 1+0, which guarantee a reasonable security on hardware failure for the data under analysis. We also have a 6×2 Tb SATA disks configured as RAID 5 for long storage of raw data, i.e., fastq files.

2.3 Computational Software

As aligner for RNA-seq projects we use STAR due to its elevated performances [10]. We use SHRiMP [11] for miRNA-seq projects, because it has specific alignment parameters for miRNAs and it was one of the aligner showing the best sensitivity in miRNA-seq benchmark experiments [4].

As reference for miRNA alignment we use the latest version of miRNA hairpins from miRbase (mirbase.org). The fasta file encompassing the hairpins of interest can be easily generated with the R script shown below:

```
download.file("ftp://mirbase.org/pub/mirbase/CURRENT/
               hairpin.fa.gz", "hairpin.fa.gz", mode="wb")
system("gunzip hairpin.fa.gz")
library(Rsamtools)
library(GenomicRanges)
library(Biostrings)
hairpins<- readRNAStringSet("hairpin.fa")
hsa<- hairpins[grep( "hsa", names(hairpins)) ]
hsa<-DNAStringSet(hsa)
writeXStringSet(hsa, "hsa.fa", format="fastq")
```

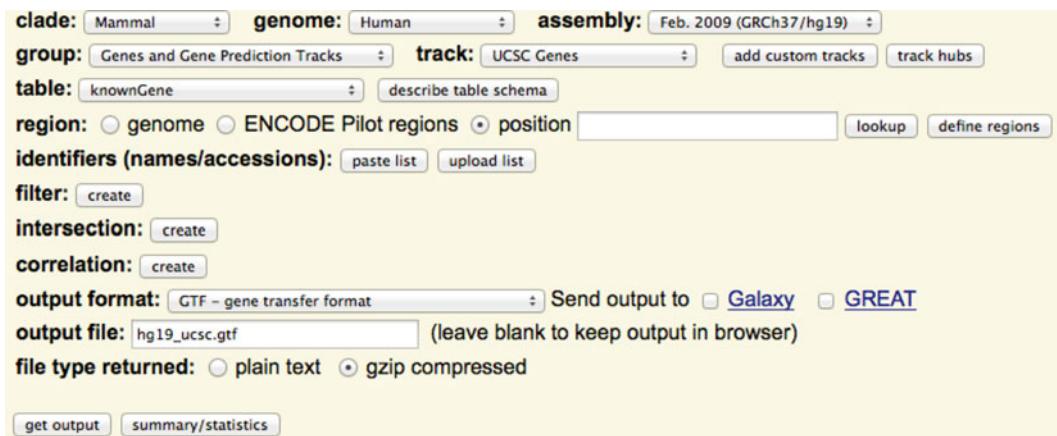


Fig. 1 Table browser at USCS genome browser. The table browser allows the exportation in various formats of annotation data. Specifically here it is shown the table selection to generate a GTF file for the human transcriptome

The above example is given for human hairpins. Changing “hsa” with another organism identifier of miRbase will produce the specific subset of hairpins. As reference for mRNA-seq we use the concatenated set of chromosomes downloadable at NCBI (<ftp://ftp.ncbi.nlm.nih.gov/genomes/>) and the UCSC annotation as GTF file. UCSC GTF annotation file can be easily generated using the table browser of the UCSC genome browser (<http://genome.ucsc.edu/cgi-bin/hgTables?command=start>) as shown in Fig. 1.

The majority of post alignment processing is done using R (<http://cran.r-project.org/>) and Bioconductor (<http://www.bioconductor.org/>) [12] frameworks (see Note 1).

3 Methods

3.1 Preprocessing

Fastq files are the standard output of high-throughput sequencers, including Illumina sequencers. Before aligning them to the reference genome it is important to check the overall quality of the data. This can be done using FASTQC (<http://www.bioinformatics.babraham.ac.uk/projects/fastqc/>), which is a stand-alone java tool that allows the quality check of fastq as well as bam files.

For miRNA data analysis it is essential to remove linkers from the fastq data. There are multiple tools to do it. We normally use the perl script provided by mirTools suite (<http://centre.bioinformatics.zj.cn/mirtools/adaptortrim.php>).

3.2 Sequence Alignment

The first step in a RNA-seq pipeline is the alignment of the fastq data to a reference dataset. miRNA data alignment is done with SHRiMP [11]. The reference database for the alignment is created with the following code:

```
$SHRIMP_FOLDER/utils/project-db.py --seed 001111100111  
111100, 0011111100111100, 001111111100111  
100, 001111111111001100, 001111111111110000 --h-  
flag --shrimp-mode ls hsa.fa
```

The above code creates a set of files named: hsa-ls.seed.0, hsa-ls.seed.1, hsa-ls.seed.2, hsa-ls.seed.3, hsa-ls.seed.4.

An example of the code needed to run SHRiMP is given below:

```
nohup $SHRIMP_FOLDER/bin/gmapper-ls --strata -o 1 --qv-  
offset 33 -Q -L hsa-ls myfile.fastq -N number_of_cores --mode  
mirna -w 170 % -E>myfile.sam 2>myfile.log &
```

In the case of RNA-seq pipeline the alignment is made with STAR. As first step it is necessary to generate the reference database. In this example hg19.fa is the file containing the concatenated set of all human chromosomes in fasta format and hg19.star is the folder in which the database is created:

```
nohup STARstatic --runMode genomeGenerate --genomeDir  
hg19.star --genomeFastaFiles hg19.fa --runThreadN number_  
of_cores &
```

The example below refers to the alignment of pair end fastq files against a reference database build in hg19.star folder.

```
nohup STARstatic --genomeDir hg19.star --readFilesIn my1R.  
fastq my2R.fastq --runThreadN number_of_cores --outFile  
NamePrefix ./mydata &
```

3.3 Postprocessing

In the case of miRNA-seq the sam file generated by SHRiMP is filtered to keep only the alignments with at least 16 contiguous perfect matches. The script below associates the number of detected counts to each miRNA present in the human hairpins dataset:

```
library(Rsamtools)  
library(Biostrings)  
asBam("my.sam", "my")  
mybam<- scanBam("my.bam", param=ScanBamParam(what=c("r  
name", "cigar")))  
cigar<- strsplit(mybam[[1]]$cigar, "S")  
cigar1<- sapply(cigar, function(x){  
    good<- sub("M", "", x[grep("M", x)])  
})  
cigar1<- as.numeric(cigar1)  
reads<- as.character(mybam[[1]]$rname[which(cigar1>= 16)])  
hairpins<- readRNAStringSet("hairpin.fa")  
hsa<- hairpins[grep("hsa", names(hairpins))]  
hsa<- DNAStringSet(hsa)
```

```

hsa.names<- strsplit(names(hsa), " ")
hsa.names<- sapply(hsa.names, function(x)x[1])
counts.mybam<- rep(NA, length(hsa.names))
names(counts.mybam)<- hsa.names
counts.mybam<- sapply(names(counts.mybam), function(x,y){
  length(which(y==x))
}, y=reads)

```

Running the above script for all the bam files under analysis it will be possible to generate the count table necessary for differential expression analysis. Since miRNA quantification rely on alignment over a very limited region of the miRNA hairpin no specific normalization approach is required but only library size normalization, which is embedded in the statistical model for differential expression analysis (*see* Subheading 3.4). Before differential expression we filter out low or not expressed miRNAs, e.g., in a two group experiment made in triplicate we remove all miRNAs being characterized by having less than 10 counts in at least 50 % of the analyzed samples.

In the case of mRNA-seq it is possible to quantify expression at various levels:

- Exon
- Transcript
- Gene

Depending on the choice of the quantification approach selected, a specific statistic is required for differential expression (*see* Subheading 3.4). Since exons, transcripts and genes are characterized by different size the sampling of the reads will be affected, in the sense that it is more likely that reads will map more frequently to a long exon compared to a short one. Furthermore, the size of the sequence sampling, i.e., the total amount of the sequenced reads for each sample, will also affect features coverage, where feature refers to exon, transcript, or gene. To compensate these biases Mortazavi [13] proposed RPKM (Reads Per Kilobase per Million mapped reads) as a method of quantifying gene/transcript expression from RNA sequencing data by normalizing for total read length and the number of sequencing reads. In case of pair-end reads it is instead used FPKM (Fragments Per Kilobase per Million mapped fragments, where a fragment). Recently it has been shown that RPKM/FPKM do not represent the best approach for data normalization [9]. In case of differential expression analysis, other normalization approaches, embedded in the differential expression model, are more indicated (*see* below Subheading 3.4).

The number of reads for a given transcript is not only determined by the transcript expression level since certain fragments are preferentially detected in the mRNA-seq data acquisition process,

leading to nonuniform detection of expression between transcripts [14–16]. Transcripts deconvolution, based on short reads data, is not computationally trivial and requires the use of sophisticated statistical models such as Cufflinks [17] or RSEM [18] in order to estimate, rather than count, expression levels of these transcripts. Since transcript expression estimation methods are still quite limited in performances we prefer to work, at least at the present time, with a more conservative approach thus counting reads associated to genes or to exons. It has to be noted that the mentioned approaches work for differential expression analysis but are not suitable for co-expression studies [19].

We use HTSeq (<http://www-huber.embl.de/users/anders/HTSeq/doc/count.html>) to generate gene-level counts for each of the sam file under analysis. A counts table can be imported in DESeq [20, 21] for differential expression analysis using the following function:

```
newCountDataSetFromHTSeqCount(sampleTable, directory=".")
```

For exon-level analysis we use two python scripts provided as a part of the DEXSeq [22] Bioconductor package: dexseq_prepare_annotation (usage: `python dexseq_prepare_annotation.py<in.gtf><out.gff>`) and dexseq_count.py (usage: `python dexseq_count.py--paired=PAIRED --stranded=no --minaqual=10 hg19.gff my.sam my.counts`), which use HTSeq to generate exon-level counts. These counting procedures require some knowledge on gene/exon annotation and results are, at a certain extent, dependent on the type of annotation used. At the present time the most used annotation databases are UCSC (genome.ucsc.edu) and ENSEMBL (www.ensembl.org). Routinely in our lab we use UCSC annotation, which is a transcript-oriented annotation. The GTF annotation file can be easily generated as shown in Fig. 1. Counts can be then imported in DEXSeq with the function:

```
read-HTSeqCounts(countfiles, design, flattenedfile)
```

where *countfiles* is a string vector containing the output files with the paths from `dexseq_count.py`, *design* is vector of factors with information corresponding to each of the countfiles and *flattenedfile* is the annotation file generated by `dexseq_prepare_annotation.py`.

3.4 Differential Expression Analysis

Differential expression analysis for miRNAs and for gene level can be done applying the same statistical methods. We have observed that the best tools for miRNA-seq differential expression are DESeq [20] and baySeq [4, 23]. Their use is relatively simple and a detailed description of the use of these tools is found by the following commands:

```
library(DESeq)
openVignette("DESeq")
```

```
library(baySeq)
openVignette("baySeq")
```

The counts for each experiment can be generated as indicated above in post-processing paragraph.

DESeq estimates the variance in digital data and tests for differential expression [20]. The method implemented in DESeq uses the mean as a good predictor of the variance; that is, genes with a similar expression level also have similar variance across replicates. Hence, it predicts the variance from the mean. This estimation is done by calculating for each gene, the sample mean and variance within replicates and then fitting a curve to this data. The statistics tests for differences between the base means of the two conditions.

baySeq is based on the NB (negative binomial) model. Specifically, it estimates the empirical distribution on the parameters of the NB distribution by bootstrapping from the data and the subsequent acquisition of posterior likelihoods, thus estimating the proportions of differentially expressed counts.

Both methods perform quite well on miRNA-seq data. Instead in an analysis involving a much larger set of features, as in the case of gene-level analysis, DESeq is often overly conservative [8]. Furthermore, the modification of the parameters in DESeq can have significant effects on the results of the differential expression analysis and the recommended parameters [20] are the best choice. baySeq could produce highly variable results in case the differentially expressed genes are up-regulated in one condition compared to the other [8]. It is important to note that the behavior of DESeq and baySeq in case of a very low number of replications, i.e., 2, results in a poor FDR control [8].

As indicated above for the detection of alternative splicing events we use an exon-level approach: DEXSeq [22]. The first point to be highlighted is how counts are associated to exons. Exons are fragmented on the basis of their association to a specific isoform (Fig. 2) and the read counts are associated to them on the

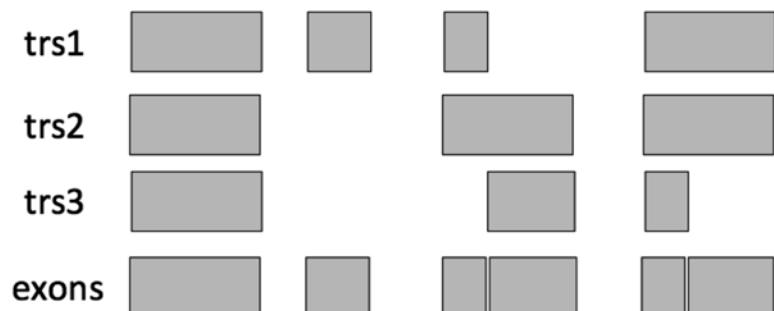


Fig. 2 DEXSeq exon binning. Exons are fragmented on the basis of their association to isoforms

basis of chromosomal location. Then DEXSeq uses the same approach of DESeq to normalize the libraries on the basis of their size and the variance of each exon is estimated using essentially the same approach used in edgeR [24]. In calculating the exon variance it is possible to filter the data selecting only genes characterized by a maximum number of exons (maxExon parameter) and exon's total sum of counts over all samples is higher than a user defined value (minCount). Specifically the minCount parameter allows reducing significantly the number of evaluated genes and thus reducing the number of multiple tests. Subsequently, a generalized linear model is fitted:

$$\text{sample} + \text{exon} + \text{condition} * I(\text{exon} == \text{exonID}) \quad (1)$$

and compared with the null model:

$$\text{sample} + \text{exon} + \text{condition} \quad (2)$$

Then the deviances of both fits are compared using a χ^2 -distribution and p -values are corrected by the Benjamini–Hochberg adjustment for multiple testing.

As in the case of DESeq a very detailed explanation of the procedure needed to run a differential expression analysis at exon level can be seen using the following code:

```
library(DESeq)
openVignette("DEXSeq")
```

A particularly useful function in the package is the possibility to generate an HTML report of the results using the following code:

```
DEXSeqHTML(myExonSet, FDR=0.1)
```

4 Note

1. The procedure described for miRNA-seq is fully implemented in oneChannelGUI [25] Bioconductor package. Since oneChannelGUI is a graphical interface, user can run the complete miRNA-seq analysis without the need to write any line of R code.

Acknowledgement

This study was funded by grants from the Epigenomics Flagship Project EPIGEN and FP7-Health-2012-Innovation-1 NGS-PTL Grant no. 306242. SInG, A*STAR, Singapore.

References

1. Maher CA, Kumar-Sinha C, Cao X, Kalyana-Sundaram S, Han B, Jing X, Sam L, Barrette T, Palanisamy N, Chinnaiyan AM (2009) Transcriptome sequencing to detect gene fusions in cancer. *Nature* 458(7234):97–101
2. McGettigan PA (2013) Transcriptomics in the RNA-seq era. *Curr Opin Chem Biol* 17(1):4–11
3. Arribere JA, Gilbert WV (2013) Roles for transcript leaders in translation and mRNA decay revealed by transcript leader sequencing. *Genome Res* 23(6):977–987
4. Cordero F, Beccuti M, Arigoni M, Donatelli S, Calogero RA (2012) Optimizing a massive parallel sequencing workflow for quantitative miRNA expression analysis. *PloS One* 7(2):e31630
5. Carrara M, Beccuti M, Lazzarato F, Cavallo F, Cordero F, Donatelli S, Calogero RA (2013) State-of-the-art fusion-finder algorithms sensitivity and specificity. *Biomed Res Int* 2013:340620
6. Carrara M, Beccuti M, Cavallo F, Donatelli S, Lazzarato F, Cordero F, Calogero RA (2013) State of art fusion-finder algorithms are suitable to detect transcription-induced chimeras in normal tissues? *BMC Bioinformatics* 14(Suppl 7):S2
7. Hansen KD, Wu Z, Irizarry RA, Leek JT (2011) Sequencing technology does not eliminate biological variability. *Nat Biotechnol* 29(7):572–573
8. Soneson C, Delorenzi M (2013) A comparison of methods for differential expression analysis of RNA-seq data. *BMC Bioinformatics* 14:91
9. Dillies MA, Rau A, Aubert J, Hennequet-Antier C, Jeamougin M, Servant N, Keime C, Marot G, Castel D, Estelle J et al (2012) A comprehensive evaluation of normalization methods for Illumina high-throughput RNA sequencing data analysis. *Brief Bioinform* 14(6):671–683
10. Dobin A, Davis CA, Schlesinger F, Drenkow J, Zaleski C, Jha S, Batut P, Chaisson M, Gingeras TR (2013) STAR: ultrafast universal RNA-seq aligner. *Bioinformatics* 29(1):15–21
11. Rumble SM, Lacroix P, Dalca AV, Fiume M, Sidow A, Brudno M (2009) SHRIMP: accurate mapping of short color-space reads. *PLoS Comput Biol* 5(5):e1000386
12. Gentleman RC, Carey VJ, Bates DM, Bolstad B, Dettling M, Dudoit S, Ellis B, Gautier L, Ge Y, Gentry J et al (2004) Bioconductor: open software development for computational biology and bioinformatics. *Genome Biol* 5(10):R80
13. Mortazavi A, Williams BA, McCue K, Schaeffer L, Wold B (2008) Mapping and quantifying mammalian transcriptomes by RNA-Seq. *Nat Methods* 5(7):621–628
14. Dohm JC, Lottaz C, Borodina T, Himmelbauer H (2008) Substantial biases in ultra-short read data sets from high-throughput DNA sequencing. *Nucleic Acids Res* 36(16):e105
15. Hansen KD, Brenner SE, Dudoit S (2010) Biases in Illumina transcriptome sequencing caused by random hexamer priming. *Nucleic Acids Res* 38(12):e131
16. Wu Z, Wang X, Zhang X (2011) Using non-uniform read distribution models to improve isoform expression inference in RNA-Seq. *Bioinformatics* 27(4):502–508
17. Trapnell C, Williams BA, Pertea G, Mortazavi A, Kwan G, van Baren MJ, Salzberg SL, Wold BJ, Pachter L (2010) Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation. *Nat Biotechnol* 28(5):511–515
18. Li B, Dewey CN (2011) RSEM: accurate transcript quantification from RNA-Seq data with or without a reference genome. *BMC Bioinformatics* 12:323
19. Mostafavi S, Battle A, Zhu X, Urban AE, Levinson D, Montgomery SB, Koller D (2013) Normalizing RNA-sequencing data by modeling hidden covariates with prior knowledge. *PloS One* 8(7):e68141
20. Anders S, Huber W (2010) Differential expression analysis for sequence count data. *Genome Biol* 11(10):R106
21. Anders S, McCarthy DJ, Chen Y, Okoniewski M, Smyth GK, Huber W, Robinson MD (2013) Count-based differential expression analysis of RNA sequencing data using R and Bioconductor. *Nat Protoc* 8(9):1765–1786
22. Anders S, Reyes A, Huber W (2012) Detecting differential usage of exons from RNA-seq data. *Genome Res* 22(10):2008–2017
23. Hardcastle TJ, Kelly KA (2010) baySeq: empirical Bayesian methods for identifying differential expression in sequence count data. *BMC Bioinformatics* 11:422
24. Robinson MD, McCarthy DJ, Smyth GK (2010) edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. *Bioinformatics* 26(1):139–140
25. Sanges R, Cordero F, Calogero RA (2007) oneChannelGUI: a graphical interface to bioconductor tools, designed for life scientists who are not familiar with R language. *Bioinformatics* 23(24):3406–3408

Chapter 11

Transcriptome Assembly and Alternative Splicing Analysis

**Paola Bonizzoni, Gianluca Della Vedova, Graziano Pesole,
Ernesto Picardi, Yuri Pirola, and Raffaella Rizzi**

Abstract

Alternative Splicing (AS) is the molecular phenomenon whereby multiple transcripts are produced from the same gene locus. As a consequence, it is responsible for the expansion of eukaryotic transcriptomes. Aberrant AS is involved in the onset and progression of several human diseases. Therefore, the characterization of exon–intron structure of a gene and the detection of corresponding transcript isoforms is an extremely relevant biological task. Nonetheless, the computational prediction of AS events and the repertoire of alternative transcripts is yet a challenging issue.

Hereafter we introduce *PIntron*, a software package to predict the exon–intron structure and the full-length isoforms of a gene given a genomic region and a set of transcripts (ESTs and/or mRNAs). The software is open source and available at <http://pINTRON.algolab.eu>. *PIntron* has been designed for (and extensively tested on) a standard workstation without requiring dedicated expensive hardware. It easily manages large genomic regions and more than 20,000 ESTs, achieving good accuracy as shown in an experimental evaluation performed on 112 well-annotated genes selected from the ENCODE human regions used as training set in the EGASP competition.

Key words Alternative splicing, Spliced alignment, Gene structure, Expressed isoforms, Software package

1 Introduction

Genes in eukaryotes may undergo to the Alternative Splicing (AS) mechanism and, therefore, they may produce *alternative* mRNAs (or *alternative isoforms*). AS is the major molecular mechanism responsible for the expansion of eukaryotic transcriptomes and it is involved in the onset and progression of several human diseases [1]. For this reason, two fundamental and challenging problems arise: (1) characterizing the gene exon–intron structure and (2) finding the set of alternative isoforms potentially expressed by a gene.

In the present work we introduce *PIntron*, a software package designed for solving these problems through *spliced alignments* of expressed transcripts to the genome. Its input data are the genomic sequence containing a *gene locus* and the corresponding set of

expressed sequences such as ESTs and/or full-length mRNAs. PIntron works as a pipeline composed of two main steps: the first computes the spliced alignments of input transcripts to the genomic sequence in order to reconstruct a consensus exon–intron gene structure [2], while the second step assembles potential full-length gene isoforms applying the graph-based method presented in [3]. Furthermore, PIntron tries to annotate the predicted isoforms identifying some features such as the coding sequence (CDS), the polyA tail, the polyadenylation signal (PAS), and the Nonsense-Mediated Decay signal (NMD) [4].

PIntron produces an output in two file formats: GTF (Gene Transfer Format) and JSON (JavaScript Object Notation). The GTF format is a widely used standard format for representing a set of full-length isoforms and can be easily visualized using genome browsers as the UCSC (<http://genome.ucsc.edu/>). The JSON format, instead, allows the representation of all details of a prediction (such as the set of inferred introns and all full-length isoforms along with all their characteristics) in a text-based format that is both human-readable and machine-parsable in a variety of programming languages.

PIntron is able to process complex gene structures or genes associated with a large cluster of expressed sequences. Its efficiency derives from the use of efficient data structures for exploring all possible spliced alignments of input transcripts with respect to the genome. Then, only those that support a parsimonious consensus exon–intron gene structure are retained. Further ad hoc refinement procedures are implemented to improve the prediction accuracy.

PIntron is customizable by a variety of input parameters (such as the minimum length of exons and introns) and also nonexpert users can execute the program without any problem.

The accuracy and efficiency of PIntron have been evaluated by comparing results with those from ASPic [5] and Exogean [6], which are two well-established tools for gene structure prediction.

Below we report technical details to install and run PIntron as well as the characteristics of input and output files.

2 Materials

2.1 Installation

PIntron has been developed and intensively tested on Linux and should easily run also on OS X. It can be downloaded from <http://pintron.algolab.eu> as a TAR or ZIP archive and the following additional software is required: Python v3.1 or newer, Perl v5 and a recent version of the standard GNU toolchain (the C compiler “gcc” and the build tool “make”, in particular). All these prerequisites can be easily installed on popular Linux distributions through their package manager.

PIntron is composed of several programs (each one performing a step of the prediction pipeline) managed and executed by the script “pintron”. The compilation step (for which detailed and up-to-date instructions are available at the PIntron website) prepares a directory containing the main script “pintron”, all needed executables and sample input files (human gene TP53).

In order to successfully execute the pipeline, we suggest to copy the directory containing the executables prepared at the compilation step in a single (dedicated) directory that we will indicate as the *binary directory*. Sometimes it may be convenient to create a subdirectory of the user home folder, called “pintron”, in which executables can be copied. In this case, no system administrative privileges are required.

2.2 Input Data

PIntron takes in input the genomic sequence of a gene locus and a set of expressed sequences (mostly ESTs and mRNAs). In particular, it requires two input files: a FASTA file containing the genomic sequence (*genomic file*) and a MultiFASTA file containing the EST/mRNA sequences (*transcript file*). All nucleotide sequences must be in uppercase letters.

In addition, the genomic file must contain the nucleotide sequence on the same strand of the input gene and its FASTA header must be in the format “>chrZZ:START:END:STRAND”, where ZZ is the reference chromosome, START and END are the start and end coordinates of the genomic region on the reference chromosome, and STRAND is +1 (or, simply, 1) for a gene on the plus strand of the chromosome or -1 for a gene on the minus strand of the chromosome.

An example of genomic file is shown in Fig. 1 for human TP53 gene. Header characteristics are highlighted in boxes.

The transcript file must contain input sequences in a MultiFASTA format (typically a UniGene file) and each header should include the substring “/gb=XXXXXX”, where XXXXXX is a unique identifier (like the GenBank identifier associated to UniGene sequences). The substring “/clone_end=YY” is optional and allows to specify the read strand (YY). In particular, YY=3' means plus strand (or 5'3'), while YY=5' means minus strand (or 3'5'). In the latter case, the input transcript should be reversed

```

chromosome      start position      end position      strand
>chr17:7570000:7600000:-1
AGTCTTCCATTACCAACGTGCTACCCCCG...
ACCCACCTCTCAAGGCTTATCTCTATCT...
...

```

Fig. 1 Example of genomic file (genomic.txt) for human TP53 gene

and complemented before the alignment step. If the strand is not specified, then the sequence is considered on plus strand by default. RefSeq mRNAs are always considered on plus strand. PIntron recognizes RefSeq mRNAs if the corresponding GenBank (gb) identifier starts with “NM_”. In any case, if no valid alignments are found using the suggested strand, the program will compute a spliced alignment on the opposite strand.

Figure 2 represents an extract of a valid transcript file from the UniGene cluster Hs.437460 associated to the human TP53 gene. Two boxes highlight the GenBank identifier and its strand.

The user can optionally provide the coding sequence (CDS) annotation for RefSeq mRNAs by means of an optional text file that must be stored in the *working directory* and must be called *cds*. Indeed, input RefSeq mRNAs are handled as full-length isoforms (and thus not involved in the assembly process like EST sequences) and the corresponding coding sequence is derived from the supplied *cds* file (if any).

The format of *cds* file is quite simple and an example is provided in Fig. 3 in which the first RefSeq sequence (NM_000546) is 2,591 bp long and composed of 11 exons; its coding sequence starts at position 203 and ends at position 1,384 (stop codon included). Its translation is composed of 393aa.

identifier	strand
>gnl UG Hs#S19946258 ... /gb=CN342738 /clone_end=5' /gi=... /ug=...	
GCTGCTGGCTCCGGGACACTTGCCTCGGGCTGGAGCGTGCTTCACGACGGTGA	
CACGCTTCCCTGGATTGGCAGCCAGCTGCCCTCGGGTCACTGCCATGGAGGAGCCGA	
GTCAGATCCTAGCGTCGAGCCCCCTCTGAGTCAGAACATTTCAGACCTATGGAAACT	

Fig. 2 Example of input transcript file (ests.txt) for human TP53 gene: EST sequence CN342738 from Unigene cluster Hs.437460

# of annotated mRNAs				
mRNA length				
start/end position of CDS	# of exons			
3				
2591				
NM_000546	203	1384	11	GATGGGATTGGGGTTTCCC...
2331				
NM_001126117	279	923	8	TGAGGCCAGGAGATGGAGGC...
2404				
NM_001126116	279	908	8	TGAGGCCAGGAGATGGAGGC...

Fig. 3 Example of CDS annotation file for human TP53 gene

In all cases in which the file *cds* is not present or the CDS annotation is not provided or the isoform has been obtained by assembling spliced ESTs, PIntron tries to annotate the coding sequence by computing the left-most *Open Reading Frame* (ORF) of at least 100 bp and having a *non-weak* context. If all predicted ORFs of at least 100 bp have a *weak* context, then the left-most ORF (if any) is reported. The ORF context [7] is given by a small window “XNNATGY” of seven bases around the start codon ATG (underlined). The context is *strong* if both X and Y are purines (A or G), *medium* if only one of X and Y is a purine, and *weak* if neither X nor Y is a purine. Symbol N stands for any base.

3 Methods

3.1 Description of the Pipeline

The PIntron pipeline (Fig. 4) is composed of two steps: (1) reconstruction of the exon–intron gene structure by computing transcript-to-genome alignments (spliced alignments), and (2) assembly of potentially expressed full-length isoforms.

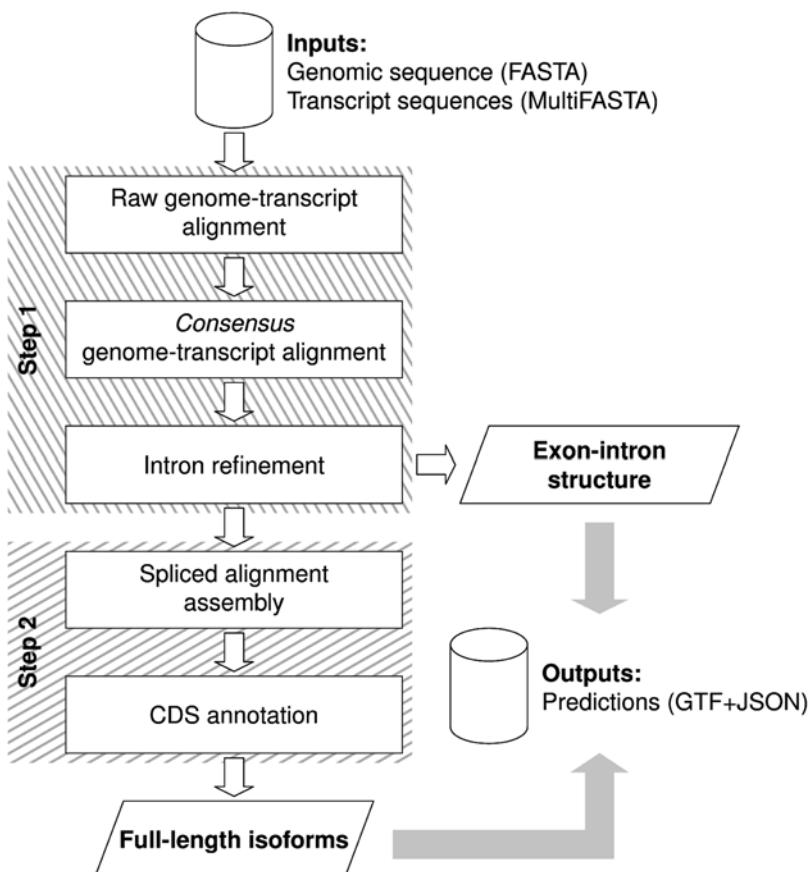


Fig. 4 The PIntron pipeline

The first step takes in input the genomic sequence (*genomic file*) and the expressed sequences (*transcript file*) and produces a set of spliced alignments (one for each expressed sequence) that are supposed to be compatible to an exon–intron gene structure. During this step, a pool of alignments are computed [2] and, then, only those biologically meaningful are retained and used to draft a consensus gene structure using the methodology proposed in [8]. This method exploits the inherent redundancy of information in a set of transcripts in order to select, among all possible alignments, only those which allow to infer splice site junctions largely supported by input data. Finally, the predicted introns are classified in U2 and U12, and putative splice sites are scored according to [9].

The alignment procedure of each input transcript is based on the efficient construction of maximal sequences of perfect transcript-genome matchings, called *embeddings*. An embedding reveals the basic “building blocks” of the spliced alignment. The embeddings are obtained from paths of a graph structure, called *embedding graph*, whose vertices are the perfect matching substrings of the genomic sequence and the input transcript. An embedding graph gives a compact and implicit representation of all the embeddings of a transcript with respect to the genome in input. This procedure is efficient and runs in time linear in the transcript length, in the genomic length, and in the size of the resulting alignments.

In the second step, PIntron reconstructs all potentially expressed isoforms by applying the graph-based method presented in [3], and annotates inferred transcripts adding some specific features such as the coding sequence (CDS), the polyA tail, and the polyadenylation signal (PAS).

Accuracy of PIntron has been assessed by comparing its performances with those [2] from ASPic [5] and Exogean [6]. In particular, the assessment has been conducted on 112 genes selected from the ENCODE human regions and used as training set in the EGASP competition [10]. Sensitivity and specificity of predictions have been computed with respect to the GENCODE annotation.

Our results indicate that PIntron performs better than ASPic and Exogean, except for the specificity at transcript level. In addition, it is the only tool to successfully compute a gene structure for all 112 genes (ASPic and Exogean failed to compute a gene structure for 19 and 8 genes, respectively).

Scalability of PIntron has been tested on 26 complex gene structures that are computationally intensive to analyze. Although Exogean is on average 30 % faster than PIntron, it is not able to compute all gene structures.

3.2 Execution

As explained in Subheading 2.1, PIntron is composed of several executables coordinated by a script called “pintron”. In the following, we assume that all the executables (together with the “pintron” script) are placed in a directory indicated by the placeholder

“<BINDIR>”. Please replace the placeholder with the full path of the binary directory (i.e., the directory where PIntron executables have been copied to). If the binary directory is in the system PATH, then it can be omitted.

3.2.1 Invocation and Main Parameters

The PIntron pipeline is executed by invoking the command “<BINDIR>/pintron”. Several options can be specified to customize default parameter values as (in parentheses the equivalent short form is reported):

- Options for input data:
 - --genomic=FILE (-g FILE), the input file containing the genomic sequence (default: “genomic.txt”).
 - --est=FILE (-s FILE), the input file containing the expressed sequences (default: “ests.txt”).
 - --organism=NAME (-n NAME), the name of the organism (default: “unknown”).
 - --gene=NAME (-e NAME), the gene name (default: “unknown”).
- Options for output data:
 - --output=FILE (-o FILE), the output file name (that will be created in JSON format) (default: “pintron-full-output.json”).
 - --gtf=FILE (-t FILE), the output file name in GTF format (default: “pintron-all-isoforms.gtf”).
 - --strict-GTF-compliance, to force PIntron to report only isoforms with an annotated CDS.
 - --logfile=FILE (-l FILE), the log file name (default: “pintron-pipeline-log.txt”).
 - --general logfile=FILE, the log file name for the main script (default: “pintron-log.txt”).
- Additional parameters:
 - --bin-dir=DIR (-b DIR), path to the binary directory including PIntron executables. This option has to be specified if the binary directory is not listed in the PATH variable.

Other advanced parameters (not discussed here) allow to adapt the alignment step of the pipeline to the characteristics of the data to analyze (*see Note 1*).

Two equivalent ways to launch PIntron on the provided sample files are:

```
<BINDIR>/pintron --bin-dir=<BINDIR> --genomic=genomic.txt --est=ests.txt --organism=human --gene=TP53 --output=pintron-full-output.json --gtf=pintron-all-isoforms.gtf
```

```
<BINDIR>/pintron -b <BINDIR> -g genomic.txt -s
ests.txt -n human -e TP53 -o pintron-full-
output.json -t pintron-all-isoforms.gtf
```

3.2.2 Managing Runtime Errors

The pipeline could prematurely terminate if a runtime error occurs. There are two common causes for runtime errors: wrong input file formats and exhaustion of computational resources. PIntron attempts to check if the input files are strictly compliant with their required format (as described in Subheading 2) and, in case it is not able to correctly parse them, terminates with a runtime error (or with unexpected/partial results). As a consequence, in case of runtime errors or unexpected results, it is important to check the formats of input files and rerun the pipeline. The other most common cause of runtime errors is the exhaustion of the computational resources. PIntron has been designed to run on standard mid-range workstations and, as such, has strict default limits on computational resources (running times and amount of memory). Default settings are conservative and prevent to deplete all computational resources of the workstation.

The inspection of log files (*see Note 2*) or the customization of default computational limits (*see Note 3*) could help in identifying and removing the causes of runtime error. However, if the user is not able to trace down the cause of a runtime error can report it using the issue tracker at PIntron website (<http://pintron.algolab.eu/>). The report should be as complete as possible and should include all data needed to reproduce the error. Moreover, results from all intermediate steps should be attached to the report. By default, these (temporary) files are deleted at the end of the execution. To prevent this, it is possible to use the option `--keep-intermediate-files (-k, in short)`.

3.3 Output Data

PIntron produces its output in two standard formats: GTF (Gene Transfer Format) and JSON (JavaScript Object Notation). More precisely, it outputs a file describing the predicted full-length isoforms in GTF format and a JSON file reporting all details of the prediction (such as the exon–intron gene structure and the predicted full-length isoforms along with their annotations). GTF is a well-known standard format in bioinformatics used to describe gene features, while the JSON format allows to describe all results into a unique file. The JSON format has been chosen since it is human-readable and very easy to parse by means of libraries that are available for almost all programming languages (Ruby, Python, JavaScript, Perl and so on).

3.3.1 The GTF Output File

A GTF file is composed of records with nine tab-separated fields, and each record represents a *feature*. Here we omit the details of the format and refer the reader to the official documentation (<http://mblab.wustl.edu/GTF22.html>).

The GTF output file is specified by the option `--gtf` and contains all predicted full-length isoforms if the execution option `--strict-GTF-compliance` has not been specified, or only the subset of the CDS-annotated isoforms, otherwise. A full-length isoform is described in the GTF output as composed of features on the genome. A feature can be an *exon*, a *5' untranslated region*, a *3' untranslated region*, a *coding sequence*, a *start codon*, or a *stop codon*.

Below, there is a description of GTF fields produced by PIntron:

- *seqname* is the reference chromosome.
- *source* is always the string “PIntron”, since PIntron is the program generating the feature.
- *feature* is the string describing the feature represented by the GTF record, and its value belongs to the set {“exon”, “3UTR”, “5UTR”, “CDS”, “start_codon”, “stop_codon”}.
- *start* is the (1-based) starting position of the feature on the plus strand of the reference chromosome (that is specified by the *seqname* field).
- *end* is the (1-based) ending position of the feature on the plus strand of the reference chromosome (that is specified by the *seqname* field).
- *score* is always a dot “.”, since PIntron does not produce this value.
- *strand* is the strand of the input gene.
- *frame* can assume one of the following values:
 - 0, if the value of the field *feature* is “start_codon” or “stop_codon” (the feature is a start or a stop codon).
 - 0, if the value of the field *feature* is “CDS” (the feature is a coding sequence), and its first base is the first base of a codon.
 - 1, if the value of the field *feature* is “CDS” (the feature is a coding sequence), and its first base is the second base of a codon.
 - 2, if the value of the field *feature* is “CDS” (the feature is a coding sequence), and its first base is the third base of a codon.
 - A dot “.” (the frame is not defined), if the value of the field *feature* is “exon” or “5UTR” or “3UTR”.
- *attribute* specifies the gene and the full-length isoform containing the feature. More in detail:
 - The value of the tag *gene_id* is the gene HUGO name specified by the execution option `--gene` (see Subheading 3.2).

```

chr17 PIntron exon      7578371 7578811 . - . gene_id "TP53"; transcript_id "TP53.25";
chr17 PIntron 5UTR     7578534 7578811 . - . gene_id "TP53"; transcript_id "TP53.25";
chr17 PIntron start_codon 7578531 7578533 . - 0 gene_id "TP53"; transcript_id "TP53.25";
chr17 PIntron CDS      7578371 7578533 . - 0 gene_id "TP53"; transcript_id "TP53.25";
chr17 PIntron exon      7577499 7577608 . - . gene_id "TP53"; transcript_id "TP53.25";
chr17 PIntron CDS      7577601 7577608 . - 2 gene_id "TP53"; transcript_id "TP53.25";
chr17 PIntron stop_codon 7577598 7577600 . - 0 gene_id "TP53"; transcript_id "TP53.25";
chr17 PIntron 3UTR     7577499 7577597 . - . gene_id "TP53"; transcript_id "TP53.25";

```

Fig. 5 Example of GTF records (features) for human TP53 gene

- The value of the tag *transcript_id* has the format “GENE.ID”, where GENE is the gene name (see the above tag *gene_id*) and ID is a progressive index associated to the full-length isoform containing the feature.

An example of GTF records produced by PIntron on the human TP53 gene is shown in Fig. 5, where the eight records (or features), related to one of the predicted isoforms, are reported.

3.3.2 The JSON Output File

A JSON file is a (*key*, *value*) dictionary, where *value* can be in turn a dictionary, a list, or simple value (like a string or a number). In the following, we describe the JSON structure of the PIntron output file, while we refer the interested reader to the official documentation (<http://www.json.org/>) for a complete description of the format.

We have chosen key names that are self-explanatory, and we have exploited the nesting nature of the JSON format to encode a gene and its sets of introns and of full-length isoforms. The JSON file produced by PIntron is specified by the execution option `--output` (see Subheading 3.2). The value of a key ending with a question mark (“?”) can be *true* or *false*, since it represents the answer to a given question.

The whole PIntron prediction is represented by a dictionary giving the predicted exon–intron gene structure and the full-length isoforms potentially expressed by the gene. In particular, the top-level dictionary gives the following data:

- The input genomic sequence (*key*=“genome”) that in turn is represented by a dictionary specifying:
 - The sequence identifier (*key*=“sequence_id”) through the FASTA header of the genomic file (without the symbol “>”).
 - The gene strand (*key*=“strand”), which is “+” if the gene is on the plus strand of the chromosome, otherwise it is “-”.
 - The sequence length (*key*=“length”).
- The number of input transcripts which have been successfully aligned to the genomic sequence (*key*=“number_of_processed_transcripts”).

- The pipeline version that produced that JSON file (*key*=“program_version”).
- The version of the JSON schema (*key*=“file_format_version”).
- The set of predicted introns (*key*=“introns”).
- The number (*key*=“number_of_predicted_isoforms”) and the set (*key*=“isoforms”) of predicted isoforms.

The values associated to the keys “introns” and “isoforms” are two dictionaries describing the set of the predicted introns and the set of the predicted full-length isoforms, respectively. In both these dictionaries, keys are progressive identifiers, while each associated value represents an *intron* and an *isoform*, respectively.

An *intron* is a dictionary giving the following data:

- The (1-based) start position of the intron both on the input genomic sequence (*key*=“relative_start”) and on the reference chromosome (*key*=“absolute_start”).
- The (1-based) end position of the intron both on the input genomic sequence (*key*=“relative_end”) and on the reference chromosome (*key*=“absolute_end”).
- The intron length (*key*=“length”).
- The 15 bp-long suffix of the donor exon (*key*=“donor_exon_suffix”), and the 15 bp-long prefix of the acceptor exon (*key*=“acceptor_exon_prefix”), flanking the intron.
- The 20 bp-long prefix (*key*=“prefix”) and the 20 bp-long suffix (*key*=“suffix”) of the intron.
- The number of input transcripts supporting the intron (*key*=“number_of_supporting_transcripts”).
- The set of the input transcripts supporting the intron (*key*=“supporting_transcripts”) as a dictionary whose keys are GenBank identifiers. The value associated to each GenBank identifier is a dictionary representing an input transcript by means of:
 - The 15 bp-long suffix (*key*=“donor_factor_suffix”) of the transcript factor aligned to the donor exon.
 - The 15 bp-long prefix (*key*=“acceptor_factor_prefix”) of the transcript factor aligned to the acceptor exon.
 - The (1-based) start (*key*=“donor_factor_start”) and end (*key*=“donor_factor_end”) positions of the transcript factor aligned to the donor exon; these endpoints are given over the transcript itself.
 - The (1-based) start (*key*=“acceptor_factor_start”) and end (*key*=“acceptor_factor_end”) positions of the transcript factor aligned to the acceptor exon; these endpoints are given over the transcript itself.

- The average transcript-genome alignment error at the donor (*key*=“donor_alignment_error”) and at the acceptor (*key*=“acceptor_alignment_error”) splice sites (computed on a 15 bp-long suffix of the donor exon and on a 15 bp-long prefix of the acceptor exon, respectively).
- The intron classification and the splice site scoring in donor and acceptor according to [9]. Precisely:
 - The intron type (*key*=“type”), where a value of 0 means a U12 intron, a value of 1 means a U2 intron, and 2 denotes an unclassified intron.
 - The score (*key*=“BPS_score”) and the position (*key*=“BPS_position”) of the *Branch Point Sequence (BPS)* from the donor splice site; if the BPS does not exist, then the score is 0, and the key “BPS_position” is not present.
 - The splice site scores in donor (*key*=“donor_score”) and acceptor (*key*=“acceptor_score”).
- The intron pattern (*key*=“pattern”), given by the first and the last dinucleotides concatenated into a unique string (the string “GTAG” denotes a canonical intron).
- When present, the repeat sequence (*key*=“repeat_sequence”), that is the sequence giving ambiguities at the intron splice sites as described in [11].

An *isoform* (the value related to a key in the dictionary “isoforms”) is a dictionary reporting the following data:

- The sequence (*key*=“sequence”) of the isoform and its length (*key*=“length”).
- If the isoform is a RefSeq mRNA (*key*=“from_RefSeq?”); if that is *true*, then its GenBank identifier is also given (*key*=“RefSeqID”).
- If the isoform is to be considered as a *reference* (*key*=“reference?”).
- A string describing the splicing variants with respect to the *reference* isoform (*key*=“variant_type”). When the isoform itself is the *reference*, then this string is “[*key*=“RefSeqID”] (Reference TR)”, where [*key*=“RefSeqID”] is the value (the null string, possibly) of the key “RefSeqID”.
- If the coding sequence (CDS) has been annotated on the isoform (*key*=“annotated_CDS?”); if that is *true*, then the following data are also given:
 - The length of the coding sequence (*key*=“CDS_length”) and its (1-based) start (*key*=“CDS_start”) and end (*key*=“CDS_end”) positions on the isoform (stop codon not included).

- If the CDS starts with the ATG codon (*key*=“start_codon?”) (a *false* value denotes an incomplete CDS).
 - If the CDS ends with a stop codon (*key*=“stop_codon?”) (a *false* value denotes an incomplete CDS).
 - If the start ATG codon, referred to the genome, is conserved with respect to the start ATG codon of the *reference* isoform (*key*=“reference_start_codon?”).
 - If the stop codon, referred to the genome, is conserved with respect to the stop codon of the *reference* isoform (*key*=“reference_stop_codon?”).
 - The length of the CDS translation (*key*=“protein_length”).
 - If the CDS translation is incomplete (*key*=“protein_incomplete?”).
 - If the CDS is in frame (*key*=“reference_frame?”) with respect to the CDS of the *reference* isoform.
- If a PolyAdenylation Signal (*key*=“PAS?”) and a polyA tail (*key*=“polyA?”) have been detected on the isoform according to [12].
 - The introns composing the isoform (*key*=“introns”), as a list of the keys of the dictionary “introns”.
 - The *Nonsense-Mediated Decay* flag (*key*=“NMD_flag”), where 1 means that the CDS has been annotated, the exon containing the stop codon is not the last one, and its 3’UTR suffix is longer than 50 bp. Otherwise (that is, the stop codon is not on the last exon or the 3’UTR exon suffix is at most 50 bp), the value is 0. When the CDS has not been annotated or the stop codon is missing, then the NMD flag is set to -1.
 - The number of exons composing the isoform (*key*=“number_of_exons”).
 - The set of exons composing the isoform (*key*=“exons”), as a list of exons represented each one by a dictionary giving the following data:
 - The (1-based) start (*key*=“relative_start”) and end (*key*=“relative_end”) positions on the input genomic sequence.
 - The (1-based) start (*key*=“absolute_start”) and end (*key*=“absolute_end”) positions on the reference chromosome.
 - The length of the exon on the genome (*key*=“length”).
 - The length of the exon on the transcript (*key*=“length_on_transcript”).
 - The exon sequence (*key*=“sequence”), which is considered on the genome, when the isoform is not a RefSeq mRNA,

otherwise the sequence is considered on the transcript. In the latter case, the length on the transcript (*key*=“length_on_transcript”) is the actual exon length.

- The length (*key*=“5UTR_length”) of the 5’UTR exon prefix (that is, the exon prefix belonging to the 5’UTR region). A value greater than 0 means that the exon has a prefix that belongs to the 5’UTR region. In that case, it is also given the start (*key*=“absolute_5UTR_start”) and the end (*key*=“absolute_5UTR_end”) positions on the reference chromosome of the 5’UTR exon prefix.
- The length (*key*=“3UTR_length”) of the 3’UTR exon suffix (that is, the exon suffix belonging to the 3’UTR region). A value greater than 0 means that the exon has a suffix that belongs to the 3’UTR region. In that case, it is also given the start (*key*=“absolute_3UTR_start”) and the end (*key*=“absolute_3UTR_end”) positions on the reference chromosome of the 3’UTR exon suffix.
- The cumulative length (*key*=“cumulative_length”) considered on the genome.
- The cumulative length (*key*=“cumulative_length_on_transcript”) considered on the transcript.
- The start (*key*=“start_codon_absolute_start”) and the end (*key*=“start_codon_absolute_end”) positions, on the reference chromosome, of the portion of the start codon (possibly the whole feature) belonging to the exon. This value exists only if the exon contains the start codon or a portion of it. In that case, the frame of the start codon (*key*=“start_codon_frame”) is also given, and is the number of the leading start codon bases that are located in the previous exon(s) of the isoform.
- The start (*key*=“stop_codon_absolute_start”) and the end (*key*=“stop_codon_absolute_end”) positions, on the reference chromosome, of the portion of the stop codon (possibly the whole feature) belonging to the exon. This value exists only if the exon contains the stop codon or a portion of it. In that case, the frame of the stop codon (*key*=“stop_codon_frame”) is also given, and is the number of leading stop codon bases that are located in the previous exon(s) of the isoform.
- The start (*key*=“CDS_absolute_start”) and the end (*key*=“CDS_absolute_end”) positions, on the reference chromosome, of the portion of the coding sequence (possibly the whole feature) belonging to the exon. This value exists only if the exon contains the coding sequence or a portion of it. In that case, the frame of the coding sequence

(*key*=“CDS_frame”) is also given, and is the number of the leading CDS bases that are located in the previous exon(s) of the isoform.

4 Notes

1. The spliced alignment step of the PIntron pipeline has several parameters that affect the resulting alignments. Their default values have been chosen to process most of the human genes and it could be necessary to change them in order to process genes with peculiar characteristics. The values of these parameters must be provided in an optional configuration file “config.ini” (placed in the *working directory*) with the following format:

<parameter>=<value>

where <parameter> is the name of a parameter and < value> is the provided value.

Some significant parameters are:

- *min-factor-length*, that is the minimum length of the common transcript-genome substrings used to compute the alignment (default: 15). In other words, it is the minimum length of exons expected for the input gene. Notice that PIntron is able to find exons shorter than this length (down to 3 nt) but they must be preceded and/or followed by exons longer than this length. This parameter affects the computational resources used during the alignment: lower values for *min-factor-length* corresponds to increased time and memory usage. For genes in highly repetitive regions or with long input transcripts (more than 100 k nucleotides), it might be necessary to set this parameter to 20 or more.
- *min-intron-length*, that is the minimum length allowed for an intron (default: 60).
- *max-intron-length*, that is the maximum length allowed for an intron, where 0 means that this parameter is ignored (default: 0).

An example of row is:

`min-factor-length="20"`

If PIntron is executed with the option *--keep-intermediate-files* (-k), then, after the execution, the file “config-dump.ini” will contain the values of all the parameters actually used in the spliced alignment step. The file can be then customized and, after renaming it to “config.ini”, can be used in subsequent invocations.

2. PIntron reports a log of its activities in two different files: the *general logfile* and the *pipeline logfile*. The first one, named

“pintron-log.txt” by default, records all the events and messages produced by the main script, while the second one, named “pintron-pipeline-log.txt” by default, records all the messages produced by the programs which compose the PIntron pipeline. The name (and the path, possibly) of these files can be changed using the two options `--general-logfile` and `--logfile` at the invocation of the main script “pintron”. The two logfiles are human-readable and their content (especially that of the pipeline logfile) could help in identifying the cause of a runtime error.

3. In case of runtime errors due to the limits imposed on the computational resources, and in case the system has more resources than those allowed by default, the user can change the default limits imposed on the most computationally expensive step—the alignment of the transcripts against the genomic sequence—using the following execution options when invoking the main script “pintron”:
 - `--set-max-factorization-time=INT`, sets a time limit (in minutes) for computing the spliced alignments (which is often the most computational expensive step). The default value is 60 (i.e., an hour).
 - `--set-max-factorization-memory=INT`, sets a memory limit (in MiB) for computing the spliced alignments (which is often the most computational expensive step). The default value is 3,000 MiB (approximately 3 GB).

References

1. Caceres J, Kornblihtt A (2002) Alternative splicing: multiple control mechanisms and involvement in human disease. *Trends Genet* 18:186–193
2. Pirola Y et al (2012) PIntron: a fast method for gene structure prediction via maximal pairings of a pattern and a text. *BMC Bioinformatics* 13:S2
3. Bonizzoni P et al (2009) Detecting alternative gene structures from spliced ESTs: a computational approach. *J Comput Biol* 16: 43–66
4. Green RE et al (2003) Widespread predicted nonsense-mediated mRNA decay of alternatively-spliced transcripts of human normal and disease genes. *Bioinformatics* 19(S1):i118–i121
5. Bonizzoni P, Rizzi R, Pesole G (2005) ASPIC: a novel method to predict the exon-intron structure of a gene that is optimally compatible to a set of transcript sequences. *BMC Bioinformatics* 6:244
6. Djebali S, Delaplace F, Crolius HR (2006) Exogean: a framework for annotating protein-coding genes in eukaryotic genomic DNA. *Genome Biol* 7(Suppl 1):S7
7. Kozak M (1986) Point mutations define a sequence flanking the AUG initiator codon that modulates translation by eukaryotic ribosomes. *Cell* 44:283–292
8. Bonizzoni P et al (2009) Minimum factorization agreement of spliced ESTs. *Lectures Notes in Bioinformatics (LNCS)*. Proceedings of 9th workshop on algorithms in bioinformatics WABI, vol 5724. pp 1–12
9. Sheth N et al (2006) Comprehensive splice-site analysis using comparative genomics. *Nucleic Acids Res* 34:3955–3967
10. Guigó R et al (2006) EGASP: the human ENCODE Genome Annotation Assessment Project. *Genome Biol* 7(Suppl 1):S2
11. Burset M, Seledtsov I, Solovyev V (2000) Analysis of canonical and non-canonical splice sites in mammalian genomes. *Nucleic Acids Res* 28:4364–4375
12. Zhang H et al (2005) PolyA_DB: a database for mammalian mRNA polyadenylation. *Nucleic Acids Res* 33(Suppl 1):D116–D120

Chapter 12

Detection of Post-Transcriptional RNA Editing Events

Ernesto Picardi, Anna Maria D'Erchia, Angela Gallo, and Graziano Pesole

Abstract

The advent of deep sequencing technologies has greatly improved the study of complex eukaryotic genomes and transcriptomes, providing the unique opportunity to investigate posttranscriptional molecular mechanisms as alternative splicing and RNA editing at single base-pair resolution. RNA editing by adenosine deamination (A-to-I) is widespread in humans and can lead to a variety of biological effects depending on the RNA type or the RNA region involved in the editing modification.

Hereafter, we describe an easy and reproducible computational protocol for the identification of candidate RNA editing sites in human using deep transcriptome (RNA-Seq) and genome (DNA-Seq) sequencing data.

Key words RNA editing, A-to-I editing, Deep sequencing, Bioinformatics, Genomics, RNA-Seq, DNA-Seq

1 Introduction

RNA editing is a posttranscriptional mechanism occurring in a wide range of organisms, affecting primary RNAs through the insertion/deletion or the modification of specific nucleotide [1]. In humans, the most common RNA editing event is the deamination of adenosine to inosine (A-to-I) by members of adenosine deaminase [2] family of enzymes acting on double stranded RNA (dsRNA) [1]. Editing events in both coding and noncoding transcripts can lead to a variety of biological effects depending on the RNA type (mRNA or ncRNA) or RNA portion undergoing editing (5'UTR, 3'UTR, CDS, intron) [3]. RNA editing within the 5'-3'-UTRs are associated to gene expression modulation (preventing the efficient ribosome binding or the recognition by small regulatory RNAs) or perturbed RNA stability, whereas modifications in coding protein regions may lead to amino acid replacements with functional consequences.

RNA editing deregulation in human has been associated to diverse neurological disorders and cancer, attesting its important

biological role in preserving the cellular homeostasis [4, 5]. In principle, the identification of RNA editing sites is straightforward and based on the direct comparison between transcripts and their corresponding genomic loci of origin. However, this naïve solution is not suitable for genome-wide investigations. Alternatively, massive transcriptome sequencing (RNA-Seq) in combination with whole genome sequencing (DNA-Seq) from the same individual (to exclude variations due to SNPs) provide now a novel exciting opportunity to explore at genome scale the RNA editing landscape of complex organisms as human [6, 7].

Hereafter, we describe a computational method to detect candidate RNA editing sites in human by using matched RNA-Seq and DNA-Seq data. The methodology is based on specific bioinformatic tools implemented in the package REDItools [8], freely available under the MIT license at <http://code.google.com/p/reditools/>.

2 Materials

REDItools are implemented in Python and tested on Linux (CentOS 6.0) and Mac OS X (10.8).

2.1 Prerequisites

2.1.1 Hardware

Computer (64-bit architecture) running Linux, Mac OS X or other UNIX-based operating systems with at least 4 GB of RAM and several GB of free disk space.

2.1.2 Short Read Mapper

Although a plethora of mapping programs has been released up to now, we obtained accurate results using GSNAp [9], freely available for Unix/Linux or Mac OS X systems from <http://research/pub.gene.com/gmap/> (*see Note 1* for basic installation on Linux/Unix systems). Nonetheless, other tools to perform the fast alignment of RNA reads onto the reference genome could be used, even though the selected mapping software may affect the detection of RNA editing [7]. Note that the mapping program should print out alignments in the standard SAM format [10].

2.1.3 Mandatory Software

REDItools requires the Python interpreter (at least version 2.6) (*see Note 2*). In Linux/Unix or Mac OS X it should be preinstalled. Alternatively, a copy can be downloaded from <http://www.python.org/download/> and installed according to instructions reported in the web page. To correctly read alignments in BAM files, the external python module pysam (at least version 0.6) needs to be installed (available from <https://github.com/pysam-developers/pysam>). In addition, SAMtools should be downloaded and installed from <http://samtools.sourceforge.net/> (version ≥0.1.18) [10].

2.1.4 Optional Software

Sometimes optional software could be required to improve final results. Indeed, mismapping errors, that are quite frequent and represent the first source of false RNA editing calls, can be mitigated aligning reads (only ones carrying mismatches) using Blat [7]. The complete Blat suite, including gfServer and gfClient executables as well as the related documentation, can be obtained from the following UCSC web page <http://hgdownload.cse.ucsc.edu/admin/exe/>.

Duplicated reads due to PCR could be marked in BAM files before running REDItools, even though this practice is still an open question (especially at RNA level).

Duplicated reads can be marked using the Picard MarkDuplicate tool.

```
$ java -Xmx2g -jar MarkDuplicates.jar INPUT=
myINPUT.bam OUTPUT=myOUTPUT.bam METRICS_FILE=
myMetrics.txt REMOVE_DUPLICATES=false ASSUME_
SORTED=true
```

The complete Picard package can be downloaded from <http://picard.sourceforge.net> (version ≥ 1.57).

2.2 Installation of REDItools

Once the mandatory software described in Subheading 2.1.3 has been installed, the last version of REDItools package can be downloaded from <http://code.google.com/p/reditools/>.

REDItools include three main scripts:

- REDItoolDnaRna.py: compares RNA-Seq and DNA-Seq data from the same sample;
- REDItoolKnown.py: explores known editing events in RNA-Seq experiments;
- REDItoolDenovo.py: predicts de novo RNA editing events using only RNA-Seq data;

Several accessory scripts to filter, annotate, sort, and convert REDItool tables are also included in the release.

In Unix/Linux or Mac OS X environment, the package can be installed in the following way:

```
$ gunzip reditools-1.0.x.tar.gz
$ tar -xvf reditools-1.0.x.tar
$ cd reditools-1.0.x
$ python setup.py install
```

To check the installation, download the testing dataset from <http://150.145.82.212/ernesto/reditools/data/testREDItools.tar.gz> and execute the following commands:

```
$ gunzip testREDItools.tar.gz
$ tar -xvf testREDItools.tar
$ cd testREDItools
$ REDItoolDnaRna.py -i rnaseq.bam -j dnaseq.
bam -f reference.fa -o outputfolder
```

Alternatively, each REDItool script can be executed independently, for example:

```
$ python REDItoolDnaRna.py -i testREDItools/rnaseq.bam -j testREDItools/dnaseq.bam -f testREDItools/reference.fa -o testREDItools/outputfolder
```

3 Methods

3.1 Mapping of RNA-Seq Reads

The mapping of RNA-Seq reads can be carried out using GSNAP and providing known splice junctions (from RefSeq, Gencode or other specialized databases) (*see Note 3*) to improve the global alignment process.

The command line for paired-end reads against the human genome assembly hg19 (*see Note 4*) is:

```
$ gsnap -d hg19 -s known-splicesites -E 1000 -n1 -Q -O--nofails -A sam--gunzip--split-output=outputGsnap -a paired R1.fastq.gz R2.fastq.gz
```

The option “-a paired” enables a procedure to remove adaptors in paired-end reads (useful in case of not preprocessed data). The options “-t” and “-B” take into account multiple working threads and available RAM for allocating genomic indexes, respectively. They speed up the mapping process and should be tuned according to the hardware running GSNAP (*see Note 5*). For human genome, 4 GB of RAM should be sufficient to preload indices in memory (option “-B 4”).

At the end of the run, GSNAP will create nine separate output files in the standard SAM format (thanks to the option--split-output), one for each alignment type (using specific filename suffixes) (*see Note 6*). For an accurate RNA editing detection, only the file with suffix “.concordant_uniq” will be taken into account for downstream analyses. Indeed, it includes unique and concordant alignments in order to exclude reads mapping to multiple genomic locations with the same number of mismatches.

3.2 SAM to BAM Conversion

REDItools accept pre-aligned reads in the standard BAM format (*see Note 7*). For this reason, after the mapping, SAM alignments need to be converted in the BAM format by means of SAMtools:

```
$ samtools view -bS outputGsnap.concordant_uniq > outputGsnap.concordant_uniq.bam
$ samtools sort outputGsnap.concordant_uniq.bam outputGsnap.concordant_uniq.sorted
$ samtools index outputGsnap.concordant_uniq.sorted.bam
```

Optionally, duplicated reads due to PCR can be marked before the RNA editing calling using the Picard MarkDuplicate tool, according to the command:

```
$ java -Xmx2g -jar MarkDuplicate.jar INPUT=
outputGsnap.concordant_uniq.sorted.bam
OUTPUT=outputGsnap.concordant_uniq.sorted.
nodup.bam METRICS_FILE=outputGsnap.concordant_
uniq.sorted.nodup.metrics.txt REMOVE_
DUPLICATES=false ASSUME_SORTED=true
```

The resulting BAM file should be indexed using SAMtools as above:

```
$ samtools index outputGsnap.concordant_uniq.
sorted.nodup.bam
```

3.3 Blat Correction (Optional)

Although GSNAP is quite accurate in aligning paired-end reads onto the complete human genome [11], several reads could be ambiguously mapped, leading to false mismatches and, thus, erroneous RNA editing calls. Misalignment errors can be mitigated realigning reads carrying mismatches by the classical Blat program. The list of problematic reads can be generated using the REDItoolBlatCorrection.py script and the following command line:

```
$ REDItoolBlatCorrection.py -i outputGsnap.
concordant_uniq.sorted.nodup.bam -f hg19.fa -F
hg19.2bit -o BlatCorrection -V -T
```

At the end of the run, the script will generate a directory named “BlatCorrection” and, inside, several “.bad” files including the list of reads prone to mismatching.

3.4 Detection of RNA Editing Candidates Using Matched DNA-Seq and RNA-Seq Data

REDItoolDnaRna.py is the main script to identify RNA editing candidates using matched DNA-Seq and RNA-Seq data. In particular, it inspects all genomic regions covered by RNA-Seq reads position-by-position looking for nucleotide changes between the reference genome and RNA sequences. DNA-Seq data are employed to support the presence of RNA editing events excluding potential SNPs or somatic mutations.

REDItoolDnaRna.py requires at least three input files: RNA-Seq alignments in BAM format, DNA-Seq alignments in BAM format and the reference genome in FASTA format. All inputs need to be indexed using SAMtools (*see Note 8*).

Once all input data are ready, a list of RNA editing candidates can be obtained with the following command line:

```
$ REDItoolDnaRna.py -i RNaseq.bam -j DNaseq.
bam -f hg19.fa -o myOUTPUT -b BlatCorrection -c
10,10 -m 20,20 -q 25.25 -u -a6-0 -v 3 -N 0.0
-n 0.1
```

where RNAseq.bam is the file of aligned RNA-Seq reads using the above described methodology (Subheadings 3.1 and 3.2), DNAseq.bam is the file of aligned DNA-Seq reads (*see Note 9*), hg19.fa is the reference human genome assembly hg19 in FASTA format, BlatCorrection is the optional directory containing problematic reads (obtained using the REDIttoolBlatCorrection.py script as suggested in Subheading 3.3) and myOUTPUT is the directory in which results will be stored.

According to the above instance, REDIttoolDnaRna.py will explore all genomic positions supported by at least 10 RNA-Seq reads and 10 DNA-Seq reads, filtering out bases with quality score lower than 25, reads with map quality lower than 20, sites in the first six bases of each read and positions with less than 3 independent reads supporting the RNA editing event.

REDIttoolDnaRna.py can infer the strand of each position when strand oriented RNA-Seq reads are provided (*see Note 10*) [7].

A detailed description of available parameters and options is in Table 1.

3.5 Exploring Known RNA Editing Sites in RNA-Seq Data

REDIttoolKnown.py script has been developed to explore the impact of RNA editing on a given RNA-Seq dataset using known editing events annotated in available databases as DARNED [12] and RADAR [13] or collected from supplementary materials of a variety of publications. REDIttoolKnown.py requires at least three input files: RNA-Seq alignments in BAM format, the reference genome in FASTA format and a list of known RNA editing sites (*see Note 11*).

Once all input data are ready, RNA editing can be explored by means of the following command line:

```
$ REDIttoolKnown.py -i RNAseq.bam -f hg19.fa  
-l known_editing_sites.txt.gz -o myOUTPUT
```

where RNAseq.bam is the file of aligned RNA-Seq reads generated by the methodology described in Subheadings 3.1 and 3.2, hg19.fa is the reference human genome assembly hg19 in FASTA format, known_editing_sites.txt.gz is the table file containing known RNA editing positions and myOUTPUT is the directory including all output files.

A detailed description of available parameters and options is in Table 2.

3.6 Output

Each execution of REDIttoolDnaRna.py or REDIttoolKnown.py creates an output folder containing a second folder in which scripts store results. The main output is a simple textual table called outTable_XXXX where XXXX is a random number generated at each run. The resulting REDIttool table comprises the following fields:

- Region: is the genomic region according to the reference genome.
- Position: is the exact genomic coordinate (1-based).

Table 1
Parameters required for REDIttoolDnaRna.py

Parameter	Description
-i	RNA-Seq BAM file
-j	DNA-Seq BAM files separated by comma or folder containing BAM files. <i>Note</i> that each chromosome must be present in a single BAM file only
-I	Sort input RNA-Seq BAM file
-J	Sort input DNA-Seq BAM file
-f	Reference genome in fasta format. <i>Note</i> that chromosome names must match chromosome names in BAMs files
-C	It indicates how many bases to load in RAM during the execution [100,000 by default]
-k	List of chromosomes to skip separated by comma. A file in which each line contains a chromosome name can also be provided
-t	It indicates how many processes should be launched [1 by default]
-o	Output folder in which all results will be stored [rediFolder_XXXX by default in which XXXX is a random number generated at each run]
-F	It indicates the name of the internal folder containing output tables
-M	If selected, pileup-like files are generated
-c	Minimum read coverage for DNA and RNA reads, respectively (dna,rna) [10,10 by default]
-Q	Fastq offset value (dna,rna) [33,33 by default]. For Illumina FASTQ 1.3+, 64 should be used
-q	Minimum quality score for DNA and RNA reads, respectively (dna,rna) [25,25 by default]. This option can influence the sequencing depth but is very useful to exclude sequencing errors
-m	Minimum mapping quality score for DNA and RNA reads, respectively (dna,rna) [25,25 by default]. This parameter can mitigate errors due to misplaced reads
-o	Minimum homopolymeric length for DNA and RNA reads, respectively (dna,rna) [5,5 by default]. Homopolymeric regions may confound alignment tools leading to misalignments. As a consequence, substitutions occurring in such regions of a predefined length (generally equal to or higher than five bases) should be excluded
-s	For strand oriented RNA-Seq reads, it indicates which read has the orientation of the RNA. Available values are: 1 for read1 in line with RNA; 2 for read2 in line with RNA. Option 1 is equivalent to fr-secondstrand library, whereas option 2 corresponds to fr-firststrand library. The strand information is essential to exclude substitutions occurring in antisense RNAs
-g	It specifies how strand should be deduced. Valid options are: 1 maxValue and 2 useConfidence (the strand is assigned if over a prefixed frequency set by -x option)
-x	Strand confidence value [0.70 by default]. It is used if -g 2 option is selected
-S	If selected, it performs the strand correction. Once the strand has been inferred, only bases according to this strand will be printed out

(continued)

Table 1
(continued)

Parameter	Description
-G	Infer strand by GFF annotation (must be in GFF and sorted, otherwise the -X option should be used). Sorting requires grep and sort unix executables
-K	GFF File with positions to exclude (must be in GFF and sorted, otherwise the -X option should be used). Sorting requires grep and sort unix executables
-T	Work only on given GFF positions (must be in GFF and sorted, otherwise the -X option should be used). Sorting requires grep and sort unix executables
-X	Sort annotation files. It requires grep and sort unix executables
-e	Exclude multi hits in RNA-Seq
-E	Exclude multi hits in DNA-Seq
-d	Exclude duplicated reads in RNA-Seq. PCR duplicates need to be marked in the input BAM
-D	Exclude duplicated reads in DNA-Seq. PCR duplicates need to be marked in the input BAM
-p	Use paired concordant reads only in RNA-Seq. It should be activated in case of paired end reads
-P	Use paired concordant reads only in DNA-Seq. It should be activated in case of paired end reads
-u	Consider mapping quality in RNA-Seq
-U	Consider mapping quality in DNA-Seq
-a	Trim x bases up and y bases down per read [0–0] in RNA-Seq
-A	Trim x bases up and y bases down per read [0–0] in DNA-Seq
-b	Blat folder containing problematic reads in RNA-Seq
-B	Blat folder containing problematic reads in DNA-Seq
-l	Remove substitutions in homopolymeric regions in RNA-Seq
-L	Remove substitutions in homopolymeric regions in DNA-Seq
-v	Minimum number of reads supporting the variation for RNA-Seq [3 by default]
-n	Minimum editing frequency for RNA-Seq [0.1 by default]
-N	Minimum variation frequency for DNA-Seq [0.1 by default]
-z	Exclude positions with multiple changes in RNA-Seq
-Z	Exclude positions with multiple changes in DNA-Seq
-w	Select RNA-Seq positions with defined changes (separated by comma, for example: AG,TC) [all by default]
-R	Exclude invariant RNA-Seq positions
-V	Exclude sites not supported by DNA-Seq

(continued)

Table 1
(continued)

Parameter	Description
-w	File containing splice sites annotations
-r	Number of bases near splice sites to explore [four by default]
--gzip	Gzip output files
-h, --help	Print out these options

Table 2
Parameters required for REDIttoolKnown.py

Parameter	Description
-i	RNA-Seq BAM file
-I	Sort input BAM file
-f	Reference genome in FASTA format. Note that chromosome names must match chromosome names in the BAM file
-l	List of known RNA editing events (<i>see Note 11</i>)
-c	It indicates how many bases to load in RAM during the execution [100,000 by default]
-k	List of chromosomes to skip separated by comma. A file in which each line contains a chromosome name can also be provided
-t	It indicates how many processes should be launched [1 by default]
-o	Output folder in which all results will be stored [rediFolder_XXXX by default in which XXXX is a random number generated at each run]
-F	It indicates the name of the internal folder containing output tables
-c	Minimum read coverage for RNA reads [10 by default]
-Q	Fastq offset value [33 by default]. For Illumina FASTQ 1.3+, 64 should be used
-q	Minimum quality score for RNA reads [25 by default]
-m	Minimum mapping quality score for RNA reads [25 by default]
-o	Minimum homopolymeric length [5 by default]
-s	For strand oriented RNA-Seq reads, it indicates which read has the orientation of the RNA. Available values are: 1 for read1 in line with RNA; 2 for read2 in line with RNA. Option 1 is equivalent to fr-secondstrand library, whereas option 2 corresponds to fr-firststrand library. The strand information is essential to exclude substitutions occurring in antisense RNAs
-g	It specifies how strand should be deduced. Valid options are: 1 maxValue and 2 useConfidence (the strand is assigned if over a prefixed frequency set by -x option)
-x	Strand confidence value [0.70 by default]. It is used if -g 2 option is selected

(continued)

Table 2
(continued)

Parameter	Description
-S	If selected, it performs the strand correction. Once the strand has been inferred, only bases according to this strand will be printed out
-G	Infer strand by GFF annotation (must be in GFF and sorted, otherwise the -X option should be used). Sorting requires grep and sort unix executables
-X	Sort annotation files. It requires grep and sort unix executables
-K	File with positions to exclude in the format chromosome name[TAB or space]coordinate
-e	Exclude multi hits
-d	Exclude duplicated reads in RNA-Seq. PCR duplicates need to be marked in the input BAM
-p	Use paired concordant reads only in RNA-Seq. It should be activated in case of paired end reads
-u	Consider mapping quality
-T	Trim x bases up and y bases down per RNA read [0–0]
-B	Blat folder containing problematic reads in RNA-Seq
-U	Remove substitutions in homopolymeric regions
-v	Minimum number of reads supporting the variation for RNA-Seq [3 by default].
-n	Minimum editing frequency [0.1 by default]
-E	Exclude positions with multiple changes
-P	File containing splice sites annotations
-r	Number of bases near splice sites to explore [4 by default]
-h	Print out these options

- Reference: is the nucleotide base in the reference genome.
- Strand: is the strand info with notation 1 for + strand, 0 for – strand and 2 for unknown or not defined strand.
- Coverage-qxx: is the depth per site at a given xx quality score.
- MeanQ: is the mean quality score per site.
- BaseCount[A,C,G,T]: is the base distribution per site in the order A, C, G, and T.
- AllSubs: is the list of observed substitutions at a given site, separated by a space. A character “-” is included in case of invariant sites.
- Frequency: is the observed frequency of substitution. In case of multiple substitutions, it refers to the first in the AllSubs field.

REDItoolDnaRna.py includes five additional columns to take into account information from DNA-Seq reads. Such columns are indicated as:

- gCoverage-qxx: is the depth per site at a given xx quality score in DNA-Seq.
- gMeanQ: is the mean quality score per site in DNA-Seq.
- gBaseCount[A,C,G,T]: is the base distribution per site in the order A, C, G, and T.
- gAllSubs: is the list of observed substitutions at a given site, separated by a space. A character “-” is included in case of invariant sites.
- gFrequency: is the observed frequency of substitution. In case of multiple substitutions, it refers to the first in the gAllSubs field.

An example of output table is shown in Table 3.

The output folder includes also a parameters.txt file that contains all selected options as well as the main command line.

3.7 Downstream Analyses

All output positions reported in REDItool tables can be sorted, filtered, and annotated using auxiliary scripts provided in the package. The complete list of available accessory scripts can be found at the web page <http://code.google.com/p/reditools/>.

An important issue in RNA editing detection is the exclusion of SNP sites also in case of DNA-Seq dataset. Indeed, positions well supported by RNA-Seq reads could not be equally supported by DNA-Seq reads and, thus, RNA editing candidates could be genuine SNPs. REDItools include the accessory FilterTable.py script that is able to filter out known SNPs as well as other preferred positions according to specialized databases as dbSNP (*see Note 12*). A typical command line is:

```
$ FilterTable.py -i myREDItoolTable -s dbsnp.gff.gz -S snp -E -p -o myREDItoolTable.nosnp
```

where myREDItoolTable is the output table from a REDItool script, dbsnp.gff.gz is the file containing genomic SNPs in GFF format (bgzipped using bgzip from SAMtools) and myREDItoolTable.nosnp is the final table. Option -S indicates the feature to filter out (according to GFF file), -E prevents the printing of filtered positions and -p reports simple statistics in the standard output.

FilterTable.py can also be used to filter out (or in) editing candidates in repetitive regions, as Alu rich regions, employing annotations stored in the RepeatMask table from UCSC. In this case, a sample command line is:

```
$ FilterTable.py -i myREDItoolTable -s dbsnp.gff.gz -F alu -E -p -o myREDItoolTable.alu
```

Table 3
Example of a REDitool output table

Region	Position	Reference	Strand	Coverage-q25	MeanQ [A,C,G,T]	AllSubs	Frequency	q25	gCoverage-	gBaseCount	
									gMeanQ [A,C,G,T]	gAllSubs	gFrequency
chr21	15412990	A	1	18	37.22	[3, 0, 15, 0]	AG	0.83	18	37.22	[3, 0, 15, 0]
chr21	15415901	A	1	13	37.15	[2, 0, 11, 0]	AG	0.85	13	37.15	[2, 0, 11, 0]
chr21	15423330	A	1	11	38.27	[4, 0, 7, 0]	AG	0.64	11	38.27	[4, 0, 7, 0]
chr21	15425640	A	1	8	36.12	[0, 0, 8, 0]	AG	1.00	8	36.12	[0, 0, 8, 0]
chr21	15456434	T	1	90	34.96	[0, 6, 1, 83]	TCTTG	0.07	90	34.96	[0, 6, 1, 83]
chr21	15461406	A	1	83	37.27	[73, 0, 10, 0]	AG	0.12	83	37.27	[73, 0, 10, 0]
chr21	15461417	A	1	90	36.26	[72, 0, 18, 0]	AG	0.20	90	36.26	[72, 0, 18, 0]
chr21	15461444	A	1	64	37.22	[26, 0, 38, 0]	AG	0.59	64	37.22	[26, 0, 38, 0]
chr21	15461479	A	1	70	36.96	[66, 0, 4, 0]	AG	0.06	70	36.96	[66, 0, 4, 0]
chr21	15461486	A	1	68	37.06	[61, 0, 7, 0]	AG	0.10	68	37.06	[61, 0, 7, 0]
chr21	15461503	A	1	76	37.26	[69, 0, 7, 0]	AG	0.09	76	37.26	[69, 0, 7, 0]
chr21	15461511	A	1	81	37.68	[55, 0, 26, 0]	AG	0.32	81	37.68	[55, 0, 26, 0]

Individual positions of REDItool tables can be annotated using the accessory AnnotateTable.py script and a favorite database as RefSeq or UCSC Known genes or Gencode (*see Note 12*). AnnotateTable.py script is quite useful to understand the gene and genomic context of detected RNA editing events. It can be launched with the following command line:

```
$ AnnotateTable.py -a refgene.gtf.gz -i myREDItoolTable -u -n RefGene -o myREDItoolTable.refgene
```

where myREDItoolTable is the output table from a REDItool script, refgene.gtf.gz is the file containing RefSeq annotations in GTF format (bgzipped using bgzip from SAMtools) and myREDItoolTable.refgene is the final annotated table.

Additional REDItool scripts can be used to format annotations, search in REDItool tables and select specific positions according to different criteria as the RNA-Seq or DNA-Seq coverage and the number of bases supporting the RNA editing change as well as the RNA editing frequency. Detailed documentation for accessory REDItool scripts is available at <http://code.google.com/p/reditools/> and <http://150.145.82.212/ernesto/reditools/doc>.

4 Notes

1. The GSNAp installation on Unix/Linux systems is quite simple. Download the latest version from <http://research-pub.gene.com/gmap/> and execute the following commands:

```
$ gunzip gmap-gsnap-2014-02-28.tar.gz
$ tar -xvf gmap-gsnap-2014-02-28.tar
$ cd gmap-gsnap-2014-02-28
$ ./configure
$ make
$ make check
$ make install
```

Executables will be installed in /usr/local/bin, whereas genome databases in /usr/local/share. If you don't have root privileges, you can install GSNAp and genome databases in your \$HOME using the command:

```
$ ./configure--prefix=/your/path--with-gmapdb=/path/to/gmapdb
```

If you plan to use reads longer than 250 bases as those generated by the MiSeq Illumina sequencer, you need to define the MAX_READLENGTH variable at the compile time.

```
$ ./configure MAX_READLENGTH=300
```

2. REDItools are designed to work with python versions 2.6 and superior. Python 3.x is not yet supported. You can check the python installation on your computer as well as the release version using the command:

```
$ python--version
```

If correctly installed you will see something like:

```
Python 2.7.2
```

3. A list of known splice sites can be obtained from RefSeq or UCSC gene annotations (also Gencode annotations are a good source of splice sites, <http://www.gencodegenes.org/>). For human RefSeq splice sites, first download the annotation table from UCSC according to the right genome assembly. If you are aligning to genome hg19, you can retrieve the table from <ftp://hgdownload.cse.ucsc.edu/goldenPath/hg19/database/refGene.txt.gz>. This track can be handled using the programs psl_splicesites and iit_store, included in the GSNAp-GMAP package.

```
$ gunzip -c refGene.txt.gz | psl_splicesites  
-s 1>my.splicesites  
$ cat my.splicesites | iit_store -o  
splicesites
```

In order to be used, this file (that will have the .iit extension) has to be moved into the maps subdirectory of your gmapdb genome by doing:

```
$ cp splicesites.iit /path/to/gmapdb/  
<genome>/<genome>.maps
```

Splice sites can be included in every GSNAp run through the -s flag. For example:

```
$ gsnap -d<genome>-s splicesites
```

4. GSNAp indexes for human genome hg19 can be generated using the utility gmap_build included in the GSNAp-GMAP package. The human genome hg19 can be downloaded from UCSC <http://hgdownload.soe.ucsc.edu/goldenPath/hg19/bigZips/chromFa.tar.gz>.

Indices can be built by doing:

```
$ gunzip chromFa.tar.gz  
$ tar -xvf chromFa.tar  
$ cd chromFa  
$ cat *.fa>hg19.fa  
$ gmap_build -d hg19 -c chrM hg19.fa
```

A folder called hg19 will be created in your gmapdb directory defined at the compiling time.

5. If GSNAp is running on a computer with multicore architecture, you can speed up the mapping process using the -t flag followed by the number of required threads. In a computer with a quadcore CPU, for instance, you can run GSNAp using -t 2, in order to reserve two threads for the mapping.

6. In GSNAP, the `--split-output` option creates separate output files for each output type, adding specific suffixes. For example, the suffix `.concordant_uniq` refers to paired-end reads aligning concordantly to a single position in the genome. The complete list of suffixes is available in the README file of GSNAP (<http://research-pub.gene.com/gmap/src/README>).
7. The BAM format is the binary version of the SAM format. More details about SAM and BAM formats can be found at <http://samtools.sourceforge.net/>.
8. BAM and FASTA files can be indexed by SAMtools:

```
$ samtools index myBAM.bam
$ samtools faidx myFASTA.fa
```

Note that BAM files need to be sorted before the indexing:

```
$ samtools sort myBAM.bam myBAM.sorted
```

9. DNA-Seq reads can be mapped using again GSNAP but excluding `-s` and `-N` flags that are specific for RNA-Seq data.
10. Current RNA-Seq libraries protocols can generate strand oriented and non strand-oriented RNA-Seq reads. Strand oriented reads can greatly improve the detection of RNA editing, mitigating the effect of antisense RNA. According to the protocol used to generate the RNA-Seq library, there are two main scenarios: (1) the left-most end of the fragment (in transcript coordinates) is the first sequenced (or only sequenced for single-end reads); (2) the right-most end of the fragment (in transcript coordinates) is the first sequenced (or only sequenced for single-end reads). REDItools can take into account the strand through `-s` flag. A value of 1 corresponds to the first scenario. On the contrary, a value of 2 is equivalent to the second scenario.
11. Known RNA editing sites have to be provided in simple textual files containing three tabulated columns: (1) the genomic region, (2) the coordinate (1-based) and the strand (+ or -). Editing tables have to be sorted and indexed by tabix before their use in REDItools. In Linux/Unix environments, the sorting can be easily done by the command:

```
$ sort -k1,1 -k2,2n mytable.txt>mytable.sorted.txt
$ bgzip mytable.sorted.txt
$ tabix -b 2 -e 2 mytable.sorted.txt.gz
```

Alternatively, editing tables can be sorted and converted in the right format for REDItools using the auxiliary script `tableToTabix.py`:

```
$ tableToTabix.py -i mytable.txt -c 2 -e 2
```

Known RNA editing sites can be downloaded for human genome hg19 assembly from DARNED (http://beamish.ucc.ie/data_hg19.txt) or RADAR (http://www.stanford.edu/~gokulr/database/Human_AG_all_hg19.txt) databases.

12. REDItools can use annotations in GTF format that includes nine tab-delimited fields:

- Chromosome name.
- Source (for example hg19_refseq).
- Feature (for example CDS, exon, snp...).
- Start coordinate (1-based).
- End coordinate (1-based).
- Score (use a dot if unknown).
- Strand (+ or -, use + if unknown).
- Frame for CDS only (a dot if unknown).
- Attributes. There are two mandatory attributes: gene_id and transcript_id separated by semicolons. For example: gene_id “C7orf55”; transcript_id “NM_197964”.

Before their use in REDItools, annotations in GTF have to be sorted and indexed using the accessory script GFFtoTabix.py:

```
$ GFFtoTabix.py -i myAnnotation.gtf
```

Further details are in <http://150.145.82.212/ernesto/redditools/doc/>.

Acknowledgments

This work was supported by the Italian Ministero dell’Istruzione, Università e Ricerca (MIUR): PRIN 2009 and 2010; Consiglio Nazionale delle Ricerche: Flagship Project Epigen, Medicina Personalizzata and Aging Program 2012–2014. It was also supported by the Italian Ministry for Foreign Affairs (Italy–Israel actions) to E.P. and AIRC to A.G.

References

1. Gott JM, Emeson RB (2000) Functions and mechanisms of RNA editing. *Annu Rev Genet* 34:499–531
2. Gerhard DS, Wagner L, Feingold EA, Shenmen CM, Grouse LH, Schuler G, Klein SL, Old S, Rasooly R, Good P, Guyer M, Peck AM, Derge JG, Lipman D, Collins FS, Jang W, Sherry S, Feolo M, Misquitta L, Lee E, Rotmistrovsky K, Greenhut SF, Schaefer CF, Buetow K, Bonner TI, Haussler D, Kent J, Kiekhaus M, Furey T, Brent M, Prange C, Schreiber K, Shapiro N, Bhat NK, Hopkins RF, Hsie F, Driscoll T, Soares MB, Casavant TL, Scheetz TE, Brownstein MJ, Usdin TB, Toshiyuki S, Carninci P, Piao Y, Dudekula DB, Ko MS, Kawakami K, Suzuki Y, Sugano S, Gruber CE, Smith MR, Simmons B, Moore T, Waterman R, Johnson SL, Ruan Y, Wei CL, Mathavan S, Gunaratne PH, Wu J, Garcia AM, Hulyk SW, Fuh E, Yuan Y, Snead A, Kowis C, Hodgson A, Muzny DM,

- McPherson J, Gibbs RA, Fahey J, Helton E, Ketteman M, Madan A, Rodrigues S, Sanchez A, Whiting M, Madari A, Young AC, Wetherby KD, Granite SJ, Kwong PN, Brinkley CP, Pearson RL, Bouffard GG, Blakesly RW, Green ED, Dickson MC, Rodriguez AC, Grimwood J, Schmutz J, Myers RM, Butterfield YS, Griffith M, Griffith OL, Krzywinski MI, Liao N, Morin R, Palmquist D, Petrescu AS, Skalska U, Smailus DE, Stott JM, Schnerch A, Schein JE, Jones SJ, Holt RA, Baross A, Marra MA, Clifton S, Makowski KA, Bosak S, Malek J (2004) The status, quality, and expansion of the NIH full-length cDNA project: the Mammalian Gene Collection (MGC). *Genome Res* 14(10B):2121–2127
3. Keegan LP, Gallo A, O'Connell MA (2001) The many roles of an RNA editor. *Nat Rev Genet* 2(11):869–878. doi:[10.1038/35098584](https://doi.org/10.1038/35098584)
 4. Silberberg G, Lundin D, Navon R, Ohman M (2012) Deregulation of the A-to-I RNA editing mechanism in psychiatric disorders. *Hum Mol Genet* 21(2):311–321. doi:[10.1093/hmg/ddr461](https://doi.org/10.1093/hmg/ddr461)
 5. Gallo A, Locatelli F (2011) ADARs: allies or enemies? The importance of A-to-I RNA editing in human disease—from cancer to HIV-1. *Biol Rev Camb Philos Soc* 87(1):95–110
 6. Picardi E, Horner DS, Chiara M, Schiavon R, Valle G, Pesole G (2010) Large-scale detection and analysis of RNA editing in grape mtDNA by RNA deep-sequencing. *Nucleic Acids Res* 38(14):4755–4767. doi:[10.1093/nar/gkq202](https://doi.org/10.1093/nar/gkq202)
 7. Ramaswami G, Lin W, Piskol R, Tan MH, Davis C, Li JB (2012) Accurate identification of human Alu and non-Alu RNA editing sites. *Nat Methods* 9(6):579–581. doi:[10.1038/nmeth.1982](https://doi.org/10.1038/nmeth.1982)
 8. Picardi E, Pesole G (2013) REDIttools: high-throughput RNA editing detection made easy. *Bioinformatics* 29(14):1813–1814. doi:[10.1093/bioinformatics/btt287](https://doi.org/10.1093/bioinformatics/btt287)
 9. Wu TD, Nacu S (2011) Fast and SNP-tolerant detection of complex variants and splicing in short reads. *Bioinformatics* 26(7):873–881
 10. Li H, Handsaker B, Wysoker A, Fennell T, Ruan J, Homer N, Marth G, Abecasis G, Durbin R (2009) The sequence alignment/map format and SAMtools. *Bioinformatics* 25(16):2078–2079
 11. Engstrom PG, Steijger T, Sipos B, Grant GR, Kahles A, Alioto T, Behr J, Bertone P, Bohnert R, Campagna D, Davis CA, Dobin A, Gingeras TR, Goldman N, Guigo R, Harrow J, Hubbard TJ, Jean G, Kosarev P, Li S, Liu J, Mason CE, Molodtsov V, Ning Z, Ponstingl H, Prins JF, Ratsch G, Ribeca P, Seledtsov I, Solovyev V, Valle G, Vitulo N, Wang K, Wu TD, Zeller G (2013) Systematic evaluation of spliced alignment programs for RNA-seq data. *Nat Methods* 10(12):1185–1191. doi:[10.1038/nmeth.2722](https://doi.org/10.1038/nmeth.2722)
 12. Kiran A, Baranov PV (2010) DARNEd: a Database of RNA EDiting in humans. *Bioinformatics* 26(14):1772–1776. doi:[10.1093/bioinformatics/btq285](https://doi.org/10.1093/bioinformatics/btq285)
 13. Ramaswami G, Li JB (2014) RADAR: a rigorously annotated database of A-to-I RNA editing. *Nucleic Acids Res* 42(Database issue):D109–D113. doi:[10.1093/nar/gkt996](https://doi.org/10.1093/nar/gkt996)

Chapter 13

Prediction of miRNA Targets

**Anastasis Oulas, Nestoras Karathanasis, Annita Louloupi,
Georgios A. Pavlopoulos, Panayiota Poirazi, Kriton Kalantidis,
and Ioannis Iliopoulos**

Abstract

Computational methods for miRNA target prediction are currently undergoing extensive review and evaluation. There is still a great need for improvement of these tools and bioinformatics approaches are looking towards high-throughput experiments in order to validate predictions. The combination of large-scale techniques with computational tools will not only provide greater credence to computational predictions but also lead to the better understanding of specific biological questions. Current miRNA target prediction tools utilize probabilistic learning algorithms, machine learning methods and even empirical biologically defined rules in order to build models based on experimentally verified miRNA targets. Large-scale protein downregulation assays and next-generation sequencing (NGS) are now being used to validate methodologies and compare the performance of existing tools. Tools that exhibit greater correlation between computational predictions and protein downregulation or RNA downregulation are considered the state of the art. Moreover, efficiency in prediction of miRNA targets that are concurrently verified experimentally provides additional validity to computational predictions and further highlights the competitive advantage of specific tools and their efficacy in extracting biologically significant results. In this review paper, we discuss the computational methods for miRNA target prediction and provide a detailed comparison of methodologies and features utilized by each specific tool. Moreover, we provide an overview of current state-of-the-art high-throughput methods used in miRNA target prediction.

Key words MiRNA target prediction, Computational tools, Databases, High-throughput methods, Biological features of miRNA target recognition

1 Introduction

1.1 State of the Art

MicroRNAs (miRNAs) belong to a recently identified group of the large family of noncoding RNAs [1]. The mature miRNA is usually 19–27 nt long and is derived from a larger precursor that folds into an imperfect stem-loop structure. The mode of action of the mature miRNA in mammalian systems is dependent on complementary base pairing primarily to the 3'-UTR region of the target mRNA, thereafter causing the inhibition of translation and/or the degradation of the mRNA.

Searching through all human genes (~25,000) and/or other species for novel miRNA gene targets is a complicated task for which fast, flexible, and reliable identification methods are required. Currently available experimental approaches working towards this goal are complex and sub-optimal [2]. Inefficiencies result from various sources, including difficulty in isolating certain miRNAs by cloning due to low expression, stability, tissue specificity, and technical difficulties of the cloning and repression assay procedures, while selecting the right 3'UTR to investigate is often a challenging task of its own. Computational prediction of miRNA gene targets from 3'UTR genomic sequences is an alternative technique which offers a much faster, cheaper and effective way of identifying putative miRNA gene targets. Moreover, by predicting the location of a miRNA gene target, these methods enable experimental biologists to concentrate their efforts on genomic regions more likely to contain novel genes that undergo miRNA regulation, thus facilitating the discovery process.

Due to the lack of negative data in this specific biological problem, the performance of current miRNA target prediction tools is largely dependent on the overall number of predicted targets. Some tools are very efficient in predicting true target sites (high sensitivity), but at the same time display an extremely large number of overall predictions (low specificity) [3–6]. In contrast, other tools display an overall high specificity and a relatively low sensitivity [7–9]. In order to provide an estimation of a false positive rate, false or mock miRNAs are often generated by randomly shuffling the nucleotide sequence of experimentally supported miRNAs [10]. Performing target prediction with these “mock” miRNAs can provide an estimation of the overall false positive rate of a miRNA target prediction tool.

Accurate prediction of novel miRNA gene targets requires the consideration of certain characteristic properties of the miRNA–target mRNA interaction. These properties are based on either experimental [11–14], or computational evidence [15–19] and can be used to build a classification scheme or predictive model. For example, the foremost nucleotides at the 5' region of a mature miRNA sequence are considered crucial for recognizing and binding to the target mRNA. Initial research performed by Kiriakidou et al. [20] has shown that almost consecutive complementarity of the first 9 miRNA nucleotides to the 3'UTR of protein coding genes is a prerequisite for translational repression. Moreover, Lewis et al. [7] showed that complementary motifs to nucleotides 2–7 of miRNA (commonly referred to as the miRNA seed region) remain preferentially conserved in several species in a statistically significant manner [21, 22].

In general, it is believed that binding of at least seven consecutive Watson–Crick (WC) base pairing nucleotides between the

foremost 5' region of the miRNA and the mRNA target is required for sufficient repression of protein production [7, 20].

Based on the above mentioned evidence, miRNA target prediction programs rely heavily on sequence complementarity of the miRNA seed region (nucleotides 2–7) to the 3'UTR sequences of candidate target genes for identifying putative miRNA binding sites [8, 23]. Furthermore, most prediction tools make use of thermodynamics and evolutionary conservation at the binding site, in order to minimize false positives (increase specificity) [8, 23, 24]. Some tools utilize additional features such as binding site structural accessibility [4, 25, 26], nucleotide composition flanking the binding sites [27, 28] or proximity of one binding site to another within the same 3' UTR [27, 29].

In summary, the general features employed for miRNA target prediction are: (1) sequence complementarity at the 5' region of the mature miRNA, better known as the seed region and commonly characterized by nucleotides 2–7, (2) secondary structure of the miRNA–target mRNA hybrid molecule and the overall thermodynamics of the interaction expressed in free energy (ΔG) and (3) species conservation observed via the use of full genome sequence alignments.

In addition to computational tools, large-scale, high-throughput transcriptomic and proteomic methods, such as microarrays and pSILAC, have recently been used, often in conjunction with computational tools, for the identification of novel miRNA gene targets [6, 30]. These methods are particularly useful as they can provide accurate protein repression data or gene expression data that may be correlated or anti-correlated to miRNA expression. Moreover, if such data is coupled to computational tools, it can facilitate rapid and precise detection of novel miRNA gene targets, while at the same time giving greater credence to computational predictions. One must keep in mind that such proteomic data may only provide indirect evidence for target genes, as the signal obtained may be due to downstream effects and not a consequence of a direct interaction between miRNA and predicted target gene.

Next-Generation Sequencing (NGS) methods have also been used for the prediction of miRNA genes, their mature sequences [31] and their downstream targets. One drawback of these techniques is that multiple small RNA sequences are often missed due to technical difficulties of the sequencing methodology, such as library construction. Moreover, not all small RNA and under expressed mRNA sequences detected by NGS methods are true miRNAs and targets respectively, unless, some physical interaction between the two sequences can be attributed to them. Experimental verification of miRNA targets can be achieved via the use of luciferase assays whereby the miRNA is expressed in vitro, while simultaneously expressing and monitoring the target messenger RNA linked to a luciferase reporter gene [32–34]. This assay provides an

experimental verification of a direct interaction between the mature miRNA and the target gene and furthermore provides evidence that regulation is mediated via the miRNA silencing pathway. However, the extent to which this interaction takes place in the intact system *in vivo* cannot be inferred from luciferase assays alone.

Currently, miRNA target prediction tools are freely available and are commonly built on sophisticated algorithms (i.e., machine learning) trained to recognize certain biological features of miRNA–target mRNA interactions. Validation of computational methodologies is achieved using protein repression information from large-scale proteomic studies (i.e., pSILAC) [30] as well as experimentally verified miRNA gene targets form online databases, like Tarbase (v5) [35]. Results from these analyses are used to compare prediction accuracy of existing target prediction tools. It is rare that target prediction tools are assessed for their ability to identify *de novo* biologically significant interactions. This can be achieved by directing predictions and high-throughput experiments towards answering specific biological questions [36]. Such approaches can provide raw material for experimental biologists to investigate interesting biological questions, such as the molecular basis of a disease like cancer.

1.2 The Biology of miRNA Target Prediction

1.2.1 Background

As mentioned earlier, the mode of action of the mature miRNA in mammalian systems is dependent on complementary base pairing to the 3'UTR region of the target mRNA, thereafter causing the inhibition of translation and/or the degradation of the mRNA. Despite the body of evidence supporting a role of miRNAs in cancer, their exact mechanism of action remains to be elucidated because the downstream targets of miRNAs implicated in cancer have yet to be defined. Towards this goal, miRNA target prediction tools can offer a first indication as to which target genes are regulated by miRNAs, thus providing new insights regarding their specific functions and guiding future experiments.

1.2.2 Biological Features of miRNA–mRNA Interactions

Early work on the principles of microRNA target recognition was conducted by Julius Brennecke et al. [14]. Based on experimental data, they separate the target sites on two classes based on their base pairing motifs, “5' dominant” which are sites that depend critically on the pairing to the miRNA 5' end and “3' compensatory” which include seed matches of four to six base pairs and seeds of seven to eight bases that contain G–U base pairs, single nucleotide bulges, or mismatches and a extensive pairing to the 3'end of the miRNA.

Additionally, they distinguish two subgroups of 5' dominant sites, canonical and seed. Canonical sites have good pairing to both 5' and 3' ends of the miRNA. Their repression efficiency was not affected by a mismatch at position 1, 9, 10 or at the 3' region, but

any mismatch in positions 2–8 strongly reduce the magnitude in target regulation. Seed matches can be as few as four central nucleotides. Seed sites, on the other hand, have good 5'pairing but little or no 3'pairing. In that case, seed matches should be more than 6 nucleotides long and any G–U base pairs cause a clear reduction in the efficiency of the regulation [14].

In more recent studies, “canonical,” “marginal,” and “atypical” categories were defined [27, 37]. Canonical are 7–8 nucleotide seed-match sites, which include two 7mers and one 8mer. The 7mer-m8 site contains the seed match augmented by a match to miRNA nucleotide 8. The 7mer-A1 site contains the seed match augmented by an A at target position 1. The 8mer site comprises the seed match flanked by both the match at position 8 and the A at position 1. Marginal are 6 nt sites matching the seed region. There are two groups, 6mer which refer to sites with perfect 6 nt match to the miRNA seed (miRNA nucleotides 2–7) and offset 6mer which refer to 6mer starting from the third nucleotide. Atypical are sites with productive 3'pairing and are divided into 3'-supplementary and 3'compensatory. In the first case, Watson–Crick pairing usually centering miRNA nucleotides 13–16 supplements a 6–8 nt seed site. At least 3–4 well-positioned contiguous pairs are typically required for increased efficacy, which explains why 3'-supplementary sites are atypical. On the other hand, in 3'-compensatory sites, Watson–Crick pairing usually centering on miRNA nucleotides 13–16 can compensate for a seed mismatch and thereby create a functional site [37].

More recently, “centered sites” have been described. They are a class of miRNA target sites that lack both perfect seed pairing and 3'-compensatory pairing, and instead have 11–12 contiguous Watson–Crick pairs to miRNA nt 4–15 [38]. In addition, based on reporter and microarray results, it was suggested that the centered site is a miRNA target site capable of downregulation comparable with that observed for single 7 nt seed sites. Although they are much less abundant, than both seed-matched sites and sites with 3'-supplementary pairing, centered sites are present in numbers similar to 3'-compensatory sites and could help explain the preferential conservation observed in the central region of most miRNAs [38].

In 2012, Sung Wook (Chi et al. [39]) identified an alternative binding mode used by miRNAs, by analyzing Ago HITS-CLIP data. G-bulge sites (positions 5–6) are often bound and regulated by miR-124 in mice brain. It was shown that bulged sites comprise more than 15 % of all Ago–miRNA interactions in mouse brain and are evolutionarily conserved. Position 6 is called the “pivot” nucleotide and suggests a model in which a transitional “nucleation bulge” leads to functional bulge mRNA–miRNA interactions, expanding the number of potential miRNA regulatory sites [40].

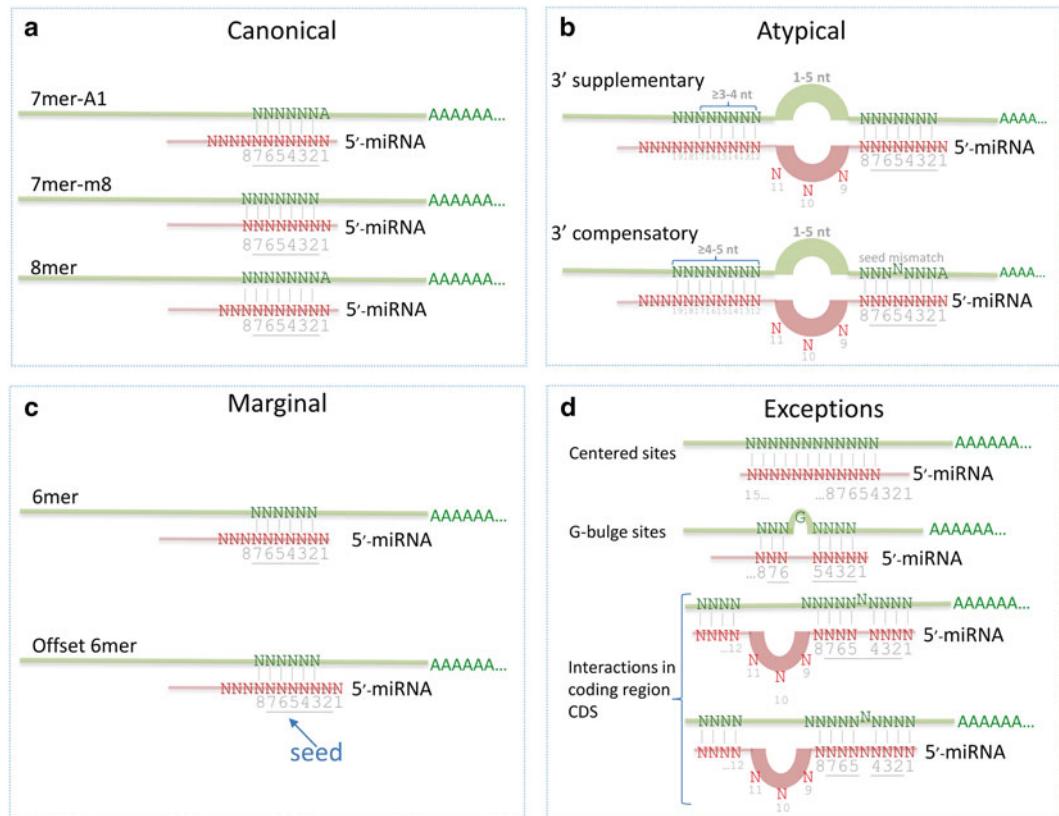


Fig. 1 Categories of miRNA–mRNA interactions. **(a)** Canonical are 7–8 nt seed-matched sites. They are divided into three groups, 7mer-A1, 7mer-m8, and 8mer. **(b)** Atypical are sites with productive 3' pairing. They consist of two groups 3' supplementary and 3' compensatory. **(c)** Marginal are 6 nt sites matching the seed region. **(d)** Recently characterized interactions, which do not belong to other categories, “centered” and “G-bulge” sites or occur in the mRNA's coding region (CDS), are shown here. *Vertical dashes* indicate contiguous Watson–Crick pairing. *Underlined numbers* on the miRNA sequence represent the seed site

One-sided or two-sided bulge in the seed region or even seedless sites have also been identified in 2008 and 2009 [41, 42]. In the first case, Nanog, Oct4, and Sox2 coding regions are targeted by miRNAs and embryonic stem cell differentiation modulated by these interactions [41]. In the second case, miR-24 regulates MYC and other cell-cycle genes by base pairing to “seedless” 3'UTR sites with extensive base pairing elsewhere in the sequence [42].

A visual representation of the different types of target site categories is shown in Fig. 1.

1.2.3 Biological Features Used by miRNA Target Prediction Algorithms

A summary of the features which are widely used by miRNA target prediction methods is described below. In general, these features can be grouped into three categories: (1) Sequence—this feature allows for the detection of base pairing between the miRNA and

the target site and moreover pinpoints the exact location of the matching nucleotides (e.g., perfect “seed” match (nucleotides 2–7 at the 5' part of a miRNA) or compensatory binding at the 3' region); (2) Thermodynamics—the minimum free energy (ΔG) of the miRNA–mRNA hybrid structure is very important for predicting target sites. Programs such as RNAcofold [43] or RNAhybrid [44, 45] can successfully predict hybrid structures between two RNA molecules based on matching base pairing and furthermore yield a minimum free energy for the overall structure. Since miRNAs bind to their targets in a very specific and stable manner, it is anticipated that the ΔG will be low. In fact, minimum free energy calculations from experimentally predicted targets sites display a very low value for ΔG [46] analogous to structures, which display stable binding and interaction; (3) Conservation—target sites are functional sequences in the 3'UTR of the transcribed mRNA. This fact makes target sites subject to evolutionary conservation across various organisms. A conservation analysis using *multiz* full genome alignment files provided by UCSC (<http://genome.ucsc.edu/>) shows that ~70 % of experimentally predicted human miRNA targets sites are in fact conserved across eight other organisms studied (including non-mammalian species). When used as a feature for predicting new target sites, conservation may provide additional support for predictions.

2 Materials

Standard PC equipped with linux or windows (depending on tool requirements) can be used to run all tools locally and standard browsers can also be employed to access Web applications and databases. For more computationally intensive tasks (i.e., RNA-seq data assembly) a PC cluster can be utilized. An example of a PC cluster for such jobs submissions is shown below:

CPU: 12 CPUs (2 * 6-core Intel® Xeon® X5680 @3.33 GHz)
(2010 model).

RAM: 64 GB (for 1 node—for memory intensive jobs), 48 GB (for any additional nodes).

Storage: each node has 500 GB to 1 TB storage.

Machine Type: x86_64.

Operating System Linux, 2.6.35-gentoo-r15.

Accessible via ssh from central master/login node.

Job submission via PBS the queuing system on the master node of the cluster.

Torque and Maui (resource manager and job scheduler) can be used on the master node of the cluster (more information: https://www.gridpp.ac.uk/wiki/Torque_and_Maui).

3 Methods

3.1 miRNA Online Databases

One of the earliest online databases for miRNA is miRBase [47] (<http://www.mirbase.org>). The current version of miRBase (v19) provides an elaborate repository which includes ~19,000 miRNA entries, ~22,000 mature miRNA sequences from over 160 species. Moreover, all entries are extensively annotated using information such as hairpin and mature sequences, genomic location, and relevant paper(s) supporting the entry. In addition to facilitating functional information for each miRNA entry, miRBase provides links to popular miRNA target prediction tools and databases, such as microRNA.org [48], DIANA-microT [49] and TargetScan [23, 50], as well as miRecords [51], DIANA-TarBase [52]. MirBase is also equipped with additional information like information about miRNA families and clusters, external links to Entrez Gene [53], Rfam [54], and HGNC [55] as well as links to deep sequencing experiments describing miRNA entries.

Databases archiving expression data for miRNAs in various tissues and cell lines are becoming an absolute necessity in the new era of RNAi. Data of this sort are often used in combination to prediction algorithms, and are ideal for obtaining initial clues and directions for biologically driven hypotheses that can be concurrently investigated in greater depth. One of the first databases describing such data is smiRNAdb (<http://www.mirz.unibas.ch/cloningprofiles>) [56]. smiRNAdb provides an elaborate miRNA expression map for humans and rodents [56] derived from sequencing ~250 small RNA libraries from 26 different cell types and organs. This database offers various analysis steps and tools in order to compare expression levels and profiles of miRNAs under different experimental conditions, such as clustering analysis and principal component analysis.

Although computational prediction of miRNA gene targets is a valuable asset, predictions should be supported by experimental verification. Techniques used to verify target prediction results can be either small or large scale [57]. The most widely used small-scale, wet lab experimentation techniques are reporter gene assays, followed by small-scale miRNA and/or mRNA expression assessment via northern blotting or qPCR. ELISA immunoassays or Western blotting can also be used to detect changes in protein concentration of gene targets. High-throughput proteomics methods like stable isotope labeling with amino acids in cell culture/SILAC and transcriptomics methods like microarrays and sequencing-based methodologies, such as HITS-CLIP, PAR-CLIP, Degradome-Seq and RNA-Seq, can further be used for large-scale miRNA target validation (further described in detail below).

Numerous databases that contain curated experimentally verified miRNA targets are currently readily available. The first comprehensive repository of experimentally validated targets was

Tarbase [52] (<http://www.microrna.gr/tarbase>). Currently, the version of Tarbase (v6.0) provides an elaborate repertoire of over 65,000 miRNA–gene interactions, includes 21 different species and, although initially contained only results from small scale experiments (i.e., luciferase assays), now includes an extensive index of target interactions obtained from large-scale, high-throughput experiments, such as microarrays and proteomics as well as interactions derived from sequencing data from HITS-CLIP and PAR-CLIP experiments.

Recently, tools that make use of text-mining techniques have been put into effect to mine through the enormous amount of biological literature available online. They provide a faster and easier way to collect useful and reliable information; however, this information has to be curated manually before depositing it online. These techniques lead to creation of databases like miRTarBase [58] (<http://mirtarbase.mbc.nctu.edu.tw>), which provides another large manually curated repertoire of experimentally verified targets. It contains ~4,200 miRNA gene interactions (release 2.5, 10/2011) from 14 species. The targets are obtained by ~1,400 articles and are usually validated by reporter gene, western blot and microarray experiments. miRTarBase also provides direct links to miRBase, Entrez and others databases as well as the UCSC genome browser.

miRecords [51] (<http://mirecords.biolead.org>), also makes use of text-mining techniques in combination with manual curation and currently hosts over 2,200 miRNA–gene interactions entries from nine different animal species. MirSel [59] (<http://services.bio.ifi.lmu.de/mirsel/>) is a similar database that contains thousands of entries and like miRecords allows for search and filtering capabilities based on miRNAs or genes of choice, species, and validation method. One main difference between the two databases is that MirSel allows for a more automated extraction of associations between microRNAs and genes from the biomedical literature. These databases also provide information derived from prediction tools which can be integrated into queries together with other search criteria. Links are also provided to external databases such as RefSeq, miRBase, mir2Disease, and others. Another cumulative database is StarBase [60] (<http://starbase.sysu.edu.cn>). StarBase focusses on sequencing experiments (i.e. PAR-CLIP, HITS-CLIP, and Degradome-Seq) derived from six different species and currently includes tens of thousands of miRNA–target interactions. In addition, StarBase provides several Web-based software like [miRPathway](#), [miRGO](#), [ClipSearch](#), [DegradomeSearch](#) to explore microRNA regulatory networks.

3.2 miRNA Target Prediction Tools in Detail

This section describes some widely used microRNA target prediction tools in detail with particular emphasis to the target site categories each tool is specialized to recognize. We aim to provide an overview of the most recently published tools and also to describe

tools that employ markedly different methodologies and prediction features. We trust that the selection of these tools is a sufficient sample to obtain an elaborate overview of the current state-of-the-art tools and algorithms for miRNA target prediction. It is also important to be able to compare the different types of tools and assess their individual performance (for details on tool comparison methods *see Notes 1–3*). Moreover most of these tools support a graphic user or Web interface that permits fast target prediction querying and also user intervention at numerous stages in the prediction pipeline (*see Note 4*).

3.2.1 PicTar (<http://pictar.mdc-berlin.de/>)

PicTar is a target prediction tool suitable for predicting “5'-dominant” target sites. However, it allows for targets with imperfect seed matches given that they pass a heuristically defined binding-energy threshold. Additionally, PicTar implements a maximum likelihood approach to incorporate the combinatorial nature of miRNA targeting. The program also implements cross-species conservation constraints. It requires conservation between at least five species for the portion of the target site that binds to the miRNA seed. Conservation amounts to a seed match occurring at overlapping positions in a cross-species UTR alignment. A more recent version of PicTar provides precompiled target predictions on the mouse genome based on comparative analyses of 17 vertebrate genomes [8].

3.2.2 TargetScan/ TargetScanS (<http://www.targetscan.org/index.html>)

TargetScan/TargetScanS is also appropriate for predicting “5'-dominant” target sites. It requires perfect complementarity with a miRNA seed and five general features of site context that help site efficacy:

- AU-rich nucleotide composition near the site,
- proximity to sites for co-expressed miRNAs,
- proximity to residues pairing to miRNA nucleotides 13–16,
- positioning within the 3' UTR at least 15 nt from the stop codon,
- positioning away from the center of long UTRs.

The model also identifies non-conserved sites because it predicts site efficacy without recourse to evolutionary conservation [23, 50].

3.2.3 MicroRNA.org (<http://microrna.org>)

MicroRNA.org incorporates both miRNA expressions and predicted miRNA targets identified by the miRanda algorithm [61] based on sequence complementarity between the mature miRNA and the target site, and binding energy of the miRNA/mRNA duplex [61]. Targets scored with mirSVR [48].

Evolutionary conservation of the target site sequence and site position in aligned 3'UTRs of homologous sequences are the two

main factors to calculate conservation. miSVR utilizes a support vector regression scoring scheme trained on a set of nine miRNA transfection experiments performed on HeLa cells [48]. Scoring is calculated using a weighted sum of context and sequence features such as A/U content flanking the target site, target site accessibility and position, 3'UTR length, base pairing and a conservation score.

3.2.4 PITA (<http://genie.weizmann.ac.il/pubs/mir07/>)

PITA uses standard settings to identify initial seeds for each miRNA in 3'UTRs, applies a target accessibility model to each putative site, and then combines sites for the same miRNA to find a total interaction score for the miRNA and the UTR. This model also adds a new dimension to miRNA target prediction, namely, the “flank sites.” “Flank sites” are sites around the seed, which are required to be unpaired. It has been found that a flank of 3 upstream and 15 downstream nucleotides results in the optimal the model performance which surpasses all other methods. The model’s performance is actually better than other methods even without using the “3-15 flank” option [4].

3.2.5 Diana-microT (<http://diana.cslab.ece.ntua.gr/microT/>)

Diana-microT uses a dynamic programming algorithm and like other tools requires seed complementarity to determine predictions. However, it does allow for some imperfect seed complementarity given there is compensatory binding in the 3' regions of the miRNA. Diana-microT also utilizes evolutionary conservation between various organisms to further provide supporting evidence for the credence of a predicted miRNA gene target [20, 49].

3.2.6 TargetProfiler (<http://mirna.imbb.forth.gr/Targetprofiler.html>)

TargetProfiler uses a hidden Markov model trained on sequence, structure, and conservation features derived from experimentally verified miRNA–mRNA interaction from Tarbase. Its allows scanning of user defined target sequences and novel miRNAs as well as querying on pre-compiled data derived from scanning all human 3'UTR sequences against all miRNA genes [62].

3.3 miRNA Functional Annotation Tools

The aspect of obtaining functional annotation for miRNAs of interest is very important and therefore numerous tools and databases have been developed in order to shed some light on the biological significance of predicted miRNA–mRNA interactions. Some common methodologies adopted by these tools are the use of network analysis to obtain functional modules of miRNAs and target genes. Statistical analysis using gene set enrichment and online databases like GO and KEGG is also commonly being employed, often in combination to network analysis, to obtain functional information for miRNA and target gene interactions taking into consideration multiplicity and co-operativity of binding. Some recently published tools, available as Web applications, that perform functional annotation are described below:

DIANA-miRPath [63] and miTalos [64]. These tools are available as standalone tools or as downstream analysis steps for miRNA target prediction pipelines. They make use of predicted as well as validated miRNA-mRNA target interactions from multiple online sources, which are subsequently subjected to pathway/network analysis. Another available tool is GeneTrail [65], which acts as an extension to existing target prediction pipelines. GeneTrail performs gene set enrichment using information derived from multiple online databases such as TRANSPATH, TRANSFAC, KEGG and GO. MiRGator [66, 67] (<http://mirgator.kobic.re.kr>) provides a more composite and integrated system for functional annotation of miRNAs. MiRGator performs statistical enrichment analysis of target genes using databases like KEGG pathways, GO, BioCarta and GenMAPP and integrates this information with expression data derived from large-scale experiments. Moreover, some well-established network analysis tools like Cytoscape [68] now support various plugins (i.e., MiRScape) for miRNA-target gene network analysis and visualization, gene set enrichment analysis, merging of networks from miRNA-mRNA and protein-protein interactions and links to a plethora of online databases.

In addition to Web server applications, there are also several online databases that store functional information and associations for miRNAs and target genes. One of these is miR2Disease [69] (<http://www.mir2disease.org>), which utilizes text-mining to extract useful functional information from biological literature and after manual curation it provides a database which connects miRNA function with numerous pathological diseases. miR2Disease currently contains over 3,000 miRNA-disease connections for ~350 miRNAs and ~160 human diseases.

Similar to mir2Disease is HMDD [70] (<http://202.38.126.151/hmdd/mirna/md>). HMDD also retrieves functional information related to miRNAs and human disease from the available literature. Yet another database that aims to provide a link between miRNAs and disease is PhenomiR [71] (<http://mips.helmholtzmuenchen.de/phenomir>). This database focuses more on miRNA which suffer from some kind of deregulation possibly associated with some genetic perturbation event like deletion or amplification. Hence a direct association is made between altered miRNA expression and specific pathological disorders. Another interesting online database is Patrocles [72] (<http://www.patrocles.org/>). Patrocles is a database of polymorphic miRNA-target interactions, which, as the name suggests, is a metaphoric association to Patrocles who was killed by Hector because he went to battle wearing the armor of Achilles. Likewise, several mRNAs may suffer mutations or single site polymorphisms (SNPs) that render them as targets for miRNAs that otherwise would not regulate the specific mRNA. This mistaken identity phenomenon has been observed for Texel MSTN mRNA which mistakenly became the target of miRNAs because of its disguise using a target octamer

motif borrowed from genuine target genes. Recently miRenvironment [73] (<http://202.38.126.151/> hmdd/tools/miren.html) has been developed aiming to provide an association between miRNA expression and environmental factors such as irradiation, viral infections, and drug administrations.

3.4 High-Throughput Tools and Approaches for Detecting miRNA–mRNA Interactions

Currently, the use of cutting edge high-throughput technologies, like transcriptomics and proteomics, has a high impact on the field of miRNA target prediction. Previous studies have used microarrays and pSILAC combined with computational tools, for the identification of novel miRNA gene targets [6, 30]. These methodologies allow for expression of genes, either at the mRNA or protein level, to be correlated to miRNA gene expression. By downregulating or over-expressing miRNAs of interest it is possible to observe the regulatory effect of potential target genes using these large-scale studies. The use of these methods in parallel to computational predictions allows for computational results to be validated via high-throughput methods and vice versa.

Next-Generation Sequencing (NGS) methods are now been used for the prediction of miRNA genes and/or their mature sequences [31] as well as obtain an expression profile that hints to downstream targets. However, these techniques pose certain problems mostly attributed to technical difficulties of the sequencing methodology such as library construction. Thus, due to the limit of sequencing depth the non-abundant miRNA as well as the ones which were expressed in a specific developmental stage cannot be detected [74]. Nowadays, a massive parallel sequencing technology has been developed which, aids in resolving most of the above mentioned drawback. Using cutting-edge technology after constructing 187 libraries from diverse developmental stages, specific tissue and cell-types, mutant conditions, and/or Argonaute immunoprecipitations, the modENCODE project managed to sequence mature miRNAs from *Drosophila melanogaster* in unprecedented depth. Remarkably, the results of modENCODE project revealed that several novel miRNA loci can be found in antisense strands as well as in coding and untranslated regions of mRNAs, giving new insights in miRNA biogenesis [75, 76]. Another example of a massive parallel sequencing was derived from ten different tissues of adult mice. Li et al. [76] managed to characterize, 87,369,704 and 252,003 sequencing reads derived from 887 mature and 281 precursor miRNAs respectively. Their analysis revealed new aspects of miRNA/pre-miRNA processing and modification. Furthermore, based on the sequences of both mature and precursor miRNAs miRGrep was developed, a miRNA discovery pipeline which uncover 239 known mouse pre-miRNAs and 41 novel potential pre-miRs [76].

In parallel, well-defined computational algorithms may analyze large-scale, high-throughput sequencing datasets and validate whether the small-RNA sequences “reads” are true miRNAs.

Algorithms can evaluate the “reads” by analyzing their compatibility with features of miRNA such as miRNA biogenesis and miRNA secondary structure [31, 74]. MiRDeep is one of the most reliable algorithms which utilize Bayesian statistics to score the fit of the reads to the biological model of miRNA biogenesis. Recently, Friedlander et al. [74] evaluated the updated algorithm miRDeep2 with NGS using a human cell line before and after knocking down the miRNA biogenesis pathway. The vast majority of the novel miRNAs predicted by miRDeep2 and expressed in this cell line were indeed specifically downregulated. MiRDeep2 may predict novel miRNAs with the accuracy of 98.6–99.9 % [74, 77]. MIRENA, another computational algorithm looks for miRNA sequences by exploring a multidimensional space defined by only five parameters characterizing acceptable pre-miRNAs and detects new miRNAs by homology to known miRNAs or from deep sequencing data [78].

Nevertheless, not all small RNA and under-expressed mRNA sequences detected by NGS methods are true miRNAs and targets respectively, unless, some physical interaction between the two sequences can be attributed to them. Ultraviolet (UV) cross-linking and immunoprecipitation (CLIP) was developed to identify specific protein–RNA interaction. CLIP relies on the principle that *in vivo* protein–RNA interactions can be preserved and then mapped, using ultraviolet C (UVC) light which can induce the formation of covalent bonds. After that, cells can be lysed and protein–RNA complexes can be immunoprecipitated with a specific antibody for the protein of interest. Next, several additional molecular steps are undertaken to obtain the RNA sequence [79]. Functional miRNAs are loaded into Argonaute protein and then bound to their target gene in order to silence it. Therefore, this functional complex made up of Argonaute–mRNA–miRNA can be verified through CLIP technology. High-throughput sequencing of RNAs isolated from cross-linking immunoprecipitation (HITS-CLIP) have identified functional protein–RNA interaction sites. Chi et al. [39] used this method in order to discover miRNA–target interactions and Argonaute proteins bound to mRNA molecules in mouse brain [39, 74]. Furthermore, Hafner et al. [80] developed an improved version of CLIP method, called Photoactivatable-Ribonucleoside-Enhanced Cross-linking and Immunoprecipitation (PAR-CLIP). PAR-CLIP uses photoactivatable nucleotide analogues which can enhance the efficiency of cross-linking. Upon exposure to UVA light (365 nm) these nucleotides lead to base transition—from thymidine to cytidine—at the cross-linked sites and mutation analysis can therefore reveal binding sites with great accuracy and precision [77, 80]. At the moment, several databases of large-scale miRNA–target interactions using the HITS-CLIP and PAR-CLIP have been developed by different research groups [39, 60, 81]. High-throughput PAR-CLIP sequences can be analyzed by computational tools such as miRTarCLIP, a novel

computational approach which has been developed by Chou et al. [77] to identify miRNA–target interactions [77].

With the advent of high-throughput technologies the field of miRNA target prediction has also been extended to encompass identification of target genes from next-generation sequencing (NGS) datasets.

Mirtools [82] is a RNA-Seq data analysis tool that works as a web application as well as a standalone command-line program. It makes use of online genomes as a reference for mapping RNA-Seq reads. Thereafter, it is capable of locating known as well as novel miRNA genes within the mapped reads using information from miRBase and also performs miRNA target prediction using the prediction tool miRanda[61].

Tools like miRanalyzer [83] (<http://bioinfo2.ugr.es/miRanalyzer/miRanalyzer.php>) are equipped with various steps for the analysis of small RNA deep sequencing experiments. These steps include read assessment and alignments construction using a tool called Bowtie [84]. miRanalyzer supports the detection of miRNA gene targets using pre-compiled data for miRNA gene targets identified using miRanda [61] and TargetScan [23, 50].

wapRNA [85] (<http://waprna.big.ac.cn>) is suited for the analysis of mRNA-Seq/miRNA-Seq data from SOLiD and Solexa technologies. It supports 10 species and can perform data analysis steps for raw RNA-Seq data like read filtering and mapping to reference genomes using Corona_Lite (SOLiD data) and BWA (Solexa data). Furthermore, wapRNA is capable of predicting miRNA targets using RNAhybrid [44, 45] and miRanda [61].

For additional bibliography and information regarding the experimental verification of miRNA gene targets using both high-throughput and smaller scale wet lab techniques *see Notes 5–8*. A comprehensive guideline of the major steps in the miRNA target prediction pipeline are outlined in **Note 9**.

4 Notes

1. State-of-the-art tools are assessed using large-scale datasets as well as experimentally verified data from online databases, and their performance is recorded. Specifically, data from PAR-CLIP [77] or pSILAC [30] can provide additional credence to computationally predicted target genes. Furthermore, pairwise ROC curves are commonly used for comparison between target prediction tools. Tools that show a higher true positive rate and low levels of false positive rate (high sensitivity and high specificity) obtain a higher area under the curve and are considered to be the state of the art.
2. Another way to compare state-of-the-art tools is to use experimentally verified miRNA targets from online databases like

- Tarbase and therefore obtain an indication as to which tool achieves a significant improvement in prediction accuracy.
3. Moreover, assessment of the number of common predictions or overlap between the tools can be performed to highlight the uniqueness of each tool and stress that each tool provides a valuable source of computational miRNA targets.
 4. State-of-the-art tools are publicly available and often include precompiled predictions for all miRNAs and gene targets stored in relational databases that can be queried by the user via a Web interface. Furthermore, they commonly provide an optimized prediction algorithm for ad hoc predictions that makes use of several biologically meaningful features of miRNA-target mRNA interactions, combined with a user friendly interface which allows for user flexibility in threshold adjustments and assists in the identification of interactions of interest. The URLs for some of these tools [86] are shown in Table 1, and most tools allow for user intervention at various steps of the prediction pipeline and can take into account both conserved and not conserved miRNA targets in accordance to user preferences.
 5. The capacity of prediction tools to predict miRNA targets with high accuracy is the first major part of the prediction analysis pipeline on the road to discovering new knowledge that is biologically meaningful and contributes in answering specific biological questions. At this stage it is important to take research one step further and perform experimental verification on computational predictions of biological significance. In previous work [87] it has been shown that the use of prediction algorithms to identify potential targets miRNAs can directly lead to the experimental validation of these targets and the generation of new biologically significant knowledge [88]. Importantly, supporting evidence from recent deep sequencing studies can provide expression data for specific miRNA genes and their targets and correlate these to specific expression signatures of biological conditions of interest [89].
 6. PAR-CLIP experiments are expected to revolutionize the research area of high-throughput miRNA target discovery due to their ability to assess the direct interaction of miRNAs and target mRNAs at a large scale, something previously performed by expression correlation. Computational identification of highly significant and evolutionary conserved target binding sites for miRNA in the specific genes of interest using target prediction tools provides the initiative for performing high-throughput validation experiments as well as reporter gene assays.
 7. In line with this, luciferase reporter assay results can show that specific genes of interest are targeted by specific miRNAs and this is depicted by decreased activity of the reporter gene in

Table 1

Databases and tools utilized for miRNA target predictions, including function and attributed to each application as well as link to the site where the application can be found

Name of system	Type (Database, tool, etc)	Website
miRDB	Tool—target prediction	http://www.mirdb.org
microRNA.org	Tool—target prediction	http://microrna.org
PicTar	Tool—target prediction	http://pictar.mdc-berlin.de
TargetScan	Tool—target prediction	http://www.targetscan.org
RNA22	Tool—target prediction	http://cm.jefferson.edu/rna22v1.0-homo_sapiens
TargetProfiler	Tool—target prediction	http://mirna.imbb.forth.gr/Targetprofiler.html
DIANA-microT-CDS	Tool—target prediction	http://www.microrna.gr/microT-CDS
TargetMiner	Tool—target prediction	http://www.isical.ac.in/~bioinfo_miu/targetminer20.htm
PITA	Tool—target prediction	http://genie.weizmann.ac.il/pubs/mir07/
GeneTrail	Tool—gene set analysis	http://genetrail.bioinf.uni-sb.de
miRGator	Tool—Integrated system for functional investigation of miRNAs	http://mrgator.kobic.re.kr
miTalos	Tool—prediction and visualization of miRNA signaling pathways	http://mips.helmholtz-muenchen.de/mitalos
DIANA-miRPath	Tool—prediction and visualization of miRNA pathways	http://www.microrna.gr/miRPathv2
wapRNA	Tool—RNA-Seq data analysis	http://waprna.big.ac.cn
DSAP	Tool—RNA-Seq data analysis	http://dsap.cgu.edu.tw
miRAnalyzer	Tool—RNA-Seq data analysis	http://bioinfo2.ugr.es/miRAnalyzer/miRAnalyzer.php
miRCat	Tool—RNA-Seq data analysis	http://srna-tools.cmp.uea.ac.uk/animal/cgi-bin/srna-tools.cgi?rm=input_form&tool=mircat
mirTools	Tool—RNA-Seq data analysis	http://122.228.158.106/mr2_dev/
miRBase	Database—miRNA repository	http://www.mirbase.org
S-MED/CC-MED	Database—miRNA expression data	http://www.oncomir.umn.edu/SMED and http://www.oncomir.umn.edu/colon/basic_search.php

(continued)

Table 1
(continued)

Name of system	Type (Database, tool, etc)	Website
smirnaDB	Database—miRNA expression data	http://www.mirz.unibas.ch/cloningprofiles
miRNEYE	Database—miRNA expression data	http://mirneye.tigem.it
microRNA.org — miRNA Expression	Database—miRNA expression data	http://www.microrna.org/microrna/getExprForm.do
miRTarBase	Database—of manually curated mirna targets	http://mirtarbase.mbc.nctu.edu.tw
DIANA-TarBase	Database—of manually curated mirna targets	http://www.microrna.gr/tarbase
miRecords	Database—of manually curated mirna targets	http://mirecords.biolead.org
miRSel	Database/tool—text mined and manually curated targets	http://services.bio.ifi.lmu.de/mirsel/
StarBase	Database—of sequence related miRNA experiments	http://starbase.sysu.edu.cn
PhenomiR	Database—miRNA associated phenotypes	http://mips.helmholtz-muenchen.de/phenomir
miR2Disease	Database—miRNA associated diseases	http://www.mir2disease.org
HMDD	Database—miRNA associated diseases	http://202.38.126.151/hmdd/mirna/md
miRenvironment	Database—validated miRNA environmental associations	http://202.38.126.151/hmdd/tools/miren.html
Patrocles	Database—of polymorphic miRNA-target interactions	http://www.patrocles.org/

wild-type binding site conditions and the increased activity in mutated binding site conditions. Moreover, addition of anti-sense LNAs for specific miRNAs to wild-type conditions is yet another experiment that can provide supportive evidence for a direct interaction of miRNA and target, given that reduced regulation is observed by increased reporter gene activity. When designing these experiments care must be taken to correctly address all case scenarios. One important aspect to be considered is that more than one miRNA may compete for the same

target site. In fact, a target site under investigation should be scanned for potential targeting by other known miRNAs.

8. Moreover, using publicly available RNA-Seq [74], full genome tiling array [90], and next-generation sequencing data [31] it is possible to obtain addition information regarding the expression of known miRNA in the tissue or cell line under investigation. This can further direct ones' experimental design and guide experimentalists in performing the correct series of validation experiments. Competition between two miRNAs for the same target site can lead to deviations in luciferase activity during LNA silencing of miRNAs.
9. At this stage we believe that a comprehensive ranking framework may be valuable for the experimental biologist who has a specific interest in miRNA gene target prediction and verification. We provide an intuitive way of ranking miRNA gene targets according to the features provided by target prediction tools.
 - (a) Initially, it is necessary to view the score assigned to the prediction (i.e., trained HMM score [62]). This will provide an initial indication as to the likelihood of the prediction.
 - (b) Next the free energy (ΔG) of the interaction may provide additional support for the stability of the hybrid molecule and further strengthen predictions (the lower the ΔG the more stable the interaction). Even manual inspection of the secondary structure, as provided by some tools [9, 23, 62] may prove to be an asset to the experienced miRNA specialists.
 - (c) As previously documented [9, 23, 62], conservation is a crucial filtering step in the miRNA target and gene prediction pipelines. The more evolutionary conserved a stretch of DNA the higher the chances that it generates functional, transcribed, RNA sequences. Therefore, conservation is yet another parameter which should be taken in consideration when selecting a potential miRNA target for experimental verification. In this perspective, conservation score provided by numerous target prediction tools, may prove to be helpful.
 - (d) Another additional feature which is provided by most tools is the number of target sites predicted on the same 3'UTR by the same miRNA. If a miRNA targets a specific 3'UTR more than once this increases the chances that the given interactions are in fact functional, as in evolutionary terms multiple target sites would not exist in the same 3'UTR unless they were of some functional purpose. Similarly if a given target site is a hotspot for multiple miRNAs then this also increases chances that one of these is in fact regulatory for similar reasoning as described above.

- (e) The use of high-throughput (PAR-CLIP, RNA-Seq, pSILAC) data can then provide additional validity to computational results. Functional annotation using online databases and network analysis, combined with statistical tools, like gene set enrichment analysis and hypothesis testing, can allow the expert RNA scientist to obtain an indication of functional modules present in the specific analysis being conducted. Moreover meaningful biological question or hypothesis can be investigated and used to direct additional research and experiments.
- (f) Finally small scale experiments can further validate and narrow down the biological question under hand and allow for fine tuning of the analysis pipeline.

References

1. Lee RC, Feinbaum RL, Ambros V (1993) The *C. elegans* heterochronic gene lin-4 encodes small RNAs with antisense complementarity to lin-14. *Cell* 75(5):843–854
2. Huttenhofer A, Vogel J (2006) Experimental approaches to identify non-coding RNAs. *Nucleic Acids Res* 34(2):635–646
3. Miranda KC, Huynh T, Tay Y et al (2006) A pattern-based method for the identification of MicroRNA binding sites and their corresponding heteroduplexes. *Cell* 126(6):1203–1217
4. Kertesz M, Iovino N, Unnerstall U et al (2007) The role of site accessibility in microRNA target recognition. *Nat Genet* 39(10):1278–1284
5. Griffiths-Jones S, Grocock RJ, van Dongen S et al (2006) miRBase: microRNA sequences, targets and gene nomenclature. *Nucleic Acids Res* 34(Database issue):D140–D144
6. Betel D, Wilson M, Gabow A et al (2008) The microRNA.org resource: targets and expression. *Nucleic Acids Res* 36(Database issue):D149–D153
7. Lewis BP, Burge CB, Bartel DP (2005) Conserved seed pairing, often flanked by adenines, indicates that thousands of human genes are microRNA targets. *Cell* 120(1):15–20
8. Krek A, Grun D, Poy MN et al (2005) Combinatorial microRNA target predictions. *Nat Genet* 37(5):495–500
9. Maragakis M, Reczko M, Simossis VA et al (2009) DIANA-microT web server: elucidating microRNA functions through target prediction. *Nucleic Acids Res* 37(Web Server issue):W273–W276
10. Maragakis M, Alexiou P, Papadopoulos GL et al (2009) Accurate microRNA target prediction correlates with protein repression levels. *BMC Bioinform* 10:295
11. Khvorova A, Reynolds A, Jayasena SD (2003) Functional siRNAs and miRNAs exhibit strand bias. *Cell* 115(2):209–216
12. Lee Y, Jeon K, Lee JT et al (2002) MicroRNA maturation: stepwise processing and subcellular localization. *EMBO J* 21(17):4663–4670
13. Lim LP, Glasner ME, Yekta S et al (2003) Vertebrate microRNA genes. *Science* 299(5612):1540
14. Brennecke J, Stark A, Russell RB et al (2005) Principles of microRNA-target recognition. *PLoS Biol* 3(3):e85
15. Helvik SA, Snove O Jr, Saetrom P (2006) Reliable prediction of Drosha processing sites improves microRNA gene prediction. *Bioinformatics* 23(2):142–149
16. Hertel J, Stadler PF (2006) Hairpins in a Haystack: recognizing microRNA precursors in comparative genomics data. *Bioinformatics* 22(14):e197–e202
17. Lim LP, Lau NC, Weinstein EG et al (2003) The microRNAs of *Caenorhabditis elegans*. *Genes Dev* 16(8):991–1008
18. Sewer A, Paul N, Landgraf P et al (2005) Identification of clustered microRNAs using an ab initio prediction method. *BMC Bioinform* 6:267–281
19. Yousef M, Nebozhyn M, Shatkay H et al (2006) Combining multi-species genomic data for microRNA identification using a Naive Bayes classifier. *Bioinformatics* 22(11):1325–1334
20. Kiriakidou M, Nelson PT, Kouranov A et al (2004) A combined computational-experimental approach predicts human microRNA targets. *Genes Dev* 18(10):1165–1178
21. Friedman RC, Farh KK, Burge CB et al (2009) Most mammalian mRNAs are conserved targets of microRNAs. *Genome Res* 19(1): 92–105

22. Witkos TM, Koscińska E, Krzyzosiak WJ (2011) Practical aspects of microRNA target prediction. *Curr Mol Med* 11(2):93–109
23. Lewis BP, Shih IH, Jones-Rhoades MW et al (2003) Prediction of mammalian microRNA targets. *Cell* 115(7):787–798
24. Enright AJ, John B, Gaul U et al (2003) MicroRNA targets in *Drosophila*. *Genome Biol* 5(1):R1
25. Long D, Lee R, Williams P et al (2007) Potent effect of target structure on microRNA function. *Nat Struct Mol Biol* 14(4):287–294
26. Marin RM, Vanicek J (2011) Efficient use of accessibility in microRNA target prediction. *Nucleic Acids Res* 39(1):19–29
27. Grimson A, Farh KK, Johnston WK et al (2007) MicroRNA targeting specificity in mammals: determinants beyond seed pairing. *Mol Cell* 27(1):91–105
28. Schmidt T, Mewes HW, Stumpflen V (2009) A novel putative miRNA target enhancer signal. *PLoS One* 4(7):e6473
29. Baek D, Villen J, Shin C et al (2008) The impact of microRNAs on protein output. *Nature* 455(7209):64–71
30. Selbach M, Schwanhausser B, Thierfelder N et al (2008) Widespread changes in protein synthesis induced by microRNAs. *Nature* 455(7209):58–63
31. Friedlander MR, Chen W, Adamidi C et al (2008) Discovering microRNAs from deep sequencing data using miRDeep. *Nat Biotechnol* 26(4):407–415
32. Cimmino A, Calin GA, Fabbri M et al (2005) miR-15 and miR-16 induce apoptosis by targeting BCL2. *Proc Natl Acad Sci U S A* 102(39):13944–13949
33. Mayr C, Hemann MT, Bartel DP (2007) Disrupting the pairing between let-7 and Hmga2 enhances oncogenic transformation. *Science* 315(5818):1576–1579
34. Sylvestre Y, De Guire V, Querido E et al (2007) An E2F/miR-20a autoregulatory feedback loop. *J Biol Chem* 282(4):2135–2143
35. Papadopoulos GL, Reczko M, Simossis VA et al (2009) The database of experimentally supported targets: a functional update of TarBase. *Nucleic Acids Res* 37(Database issue):D155–D158
36. Lee Y, Yang X, Huang Y et al (2010) Network modeling identifies molecular functions targeted by miR-204 to suppress head and neck tumor metastasis. *PLoS Comput Biol* 6(4):e1000730
37. Bartel DP (2009) MicroRNAs: target recognition and regulatory functions. *Cell* 136(2):215–233
38. Shin C, Nam JW, Farh KK et al (2010) Expanding the microRNA targeting code: functional sites with centered pairing. *Mol Cell* 38(6):789–802
39. Chi SW, Zang JB, Mele A et al (2009) Argonaute HITS-CLIP decodes microRNA-mRNA interaction maps. *Nature* 460(7254):479–486
40. Chi SW, Hannon GJ, Darnell RB (2012) An alternative mode of microRNA target recognition. *Nat Struct Mol Biol* 19(3):321–327
41. Tay Y, Zhang J, Thomson AM et al (2008) MicroRNAs to Nanog, Oct4 and Sox2 coding regions modulate embryonic stem cell differentiation. *Nature* 455(7216):1124–1128
42. Lal A, Navarro F, Maher CA et al (2009) miR-24 Inhibits cell proliferation by targeting E2F2, MYC, and other cell-cycle genes via binding to “seedless” 3'UTR microRNA recognition elements. *Mol Cell* 35(5):610–625
43. Hofacker IL (2003) Vienna RNA secondary structure server. *Nucleic Acids Res* 31(13):3429–3431
44. Rehmsmeier M, Steffen P, Hochsmann M et al (2004) Fast and effective prediction of microRNA/target duplexes. *RNA* 10(10):1507–1517
45. Kruger J, Rehmsmeier M (2006) RNAhybrid: microRNA target prediction easy, fast and flexible. *Nucleic Acids Res* 34(Web Server issue):W451–W454
46. Rusinov V, Baev V, Minkov IN et al (2005) MicroInspector: a web tool for detection of miRNA binding sites in an RNA sequence. *Nucleic Acids Res* 33(Web Server issue):W696–W700
47. Kozomara A, Griffiths-Jones S (2010) miR-Base: integrating microRNA annotation and deep-sequencing data. *Nucleic Acids Res* 39(Database issue):D152–D157
48. Betel D, Koppal A, Agius P et al (2010) Comprehensive modeling of microRNA targets predicts functional non-conserved and non-canonical sites. *Genome Biol* 11(8):R90
49. Reczko M, Maragakis M, Alexiou P et al (2012) Functional microRNA targets in protein coding sequences. *Bioinformatics* 28(6):771–776
50. Garcia DM, Baek D, Shin C et al (2011) Weak seed-pairing stability and high target-site abundance decrease the proficiency of lsy-6 and other microRNAs. *Nat Struct Mol Biol* 18(10):1139–1146
51. Xiao F, Zuo Z, Cai G et al (2009) miRecords: an integrated resource for microRNA-target interactions. *Nucleic Acids Res* 37(Database issue):D105–D110
52. Vergoulis T, Vlachos IS, Alexiou P et al (2012) TarBase 6.0: capturing the exponential growth of miRNA targets with experimental support. *Nucleic Acids Res* 40(Database issue):D222–D229

53. Maglott D, Ostell J, Pruitt KD et al (2005) Entrez Gene: gene-centered information at NCBI. *Nucleic Acids Res* 39(Database issue):D52–D57
54. Griffiths-Jones S, Moxon S, Marshall M et al (2005) Rfam: annotating non-coding RNAs in complete genomes. *Nucleic Acids Res* 33(Database issue):D121–D124
55. Seal RL, Gordon SM, Lush MJ et al (2011) genenames.org: the HGNC resources in 2011. *Nucleic Acids Res* 39(Database issue):D514–D519
56. Landgraf P, Rusu M, Sheridan R et al (2007) A mammalian microRNA expression atlas based on small RNA library sequencing. *Cell* 129(7):1401–1414
57. Thomson DW, Bracken CP, Goodall GJ (2011) Experimental strategies for microRNA target identification. *Nucleic Acids Res* 39(16):6845–6853
58. Hsu SD, Lin FM, Wu WY et al (2010) miRTarBase: a database curates experimentally validated microRNA-target interactions. *Nucleic Acids Res* 39(Database issue):D163–D169
59. Naeem H, Kuffner R, Csaba G et al (2010) miRSel: automated extraction of associations between microRNAs and genes from the biomedical literature. *BMC Bioinform* 11:135
60. Yang JH, Li JH, Shao P et al (2010) starBase: a database for exploring microRNA-mRNA interaction maps from Argonaute CLIP-Seq and Degradome-Seq data. *Nucleic Acids Res* 39(Database issue):D202–D209
61. John B, Enright AJ, Aravin A et al (2004) Human MicroRNA targets. *PLoS Biol* 2(11):e363
62. Oulas A, Karathanasis N, Louloupi A et al (2012) A new microRNA target prediction tool identifies a novel interaction of a putative miRNA with CCND2. *RNA Biol* 9(9):1196–1207
63. Vlachos IS, Kostoulas N, Vergoulis T et al (2012) DIANA miRPath v. 2.0: investigating the combinatorial effect of microRNAs in pathways. *Nucleic Acids Res* 40(Web Server issue):W498–W504
64. Kowarsch A, Preusse M, Marr C et al (2011) miTALOS: analyzing the tissue-specific regulation of signaling pathways by human and mouse microRNAs. *RNA* 17(5):809–819
65. Backes C, Keller A, Kuentzer J et al (2007) GeneTrail—advanced gene set enrichment analysis. *Nucleic Acids Res* 35(Web Server issue):W186–W192
66. Cho S, Jun Y, Lee S et al (2010) miRGator v2.0: an integrated system for functional investigation of microRNAs. *Nucleic Acids Res* 39(Database issue):D158–D162
67. Nam S, Kim B, Shin S et al (2008) miRGator: an integrated system for functional annotation of microRNAs. *Nucleic Acids Res* 36(Database issue):D159–D164
68. Shannon P, Markiel A, Ozier O et al (2003) Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome Res* 13(11):2498–2504
69. Jiang Q, Wang Y, Hao Y et al (2009) miR2Disease: a manually curated database for microRNA deregulation in human disease. *Nucleic Acids Res* 37(Database issue):D98–D104
70. Lu M, Zhang Q, Deng M et al (2008) An analysis of human microRNA and disease associations. *PLoS One* 3(10):e3420
71. Ruepp A, Kowarsch A, Schmidl D et al (2010) PhenomiR: a knowledgebase for microRNA expression in diseases and biological processes. *Genome Biol* 11(1):R6
72. Hiard S, Charlier C, Coppiepers W et al (2010) Patrocles: a database of polymorphic miRNA-mediated gene regulation in vertebrates. *Nucleic Acids Res* 38(Database issue):D640–D651
73. Yang Q, Qiu C, Yang J et al (2011) miREnvironment database: providing a bridge for microRNAs, environmental factors and phenotypes. *Bioinformatics* 27(23):3329–3330
74. Friedlander MR, Mackowiak SD, Li N et al (2012) miRDeep2 accurately identifies known and hundreds of novel microRNA genes in seven animal clades. *Nucleic Acids Res* 40(1):37–52
75. Berezikov E, Robine N, Samsonova A et al (2011) Deep annotation of *Drosophila melanogaster* microRNAs yields insights into their processing, modification, and emergence. *Genome Res* 21(2):203–215
76. Li N, You X, Chen T et al (2013) Global profiling of miRNAs and the hairpin precursors: insights into miRNA processing and novel miRNA discovery. *Nucleic Acids Res* 41(6):3619–3634
77. Chou CH, Lin FM, Chou MT et al (2013) A computational approach for identifying microRNA-target interactions using high-throughput CLIP and PAR-CLIP sequencing. *BMC Genomics* 14(Suppl 1):S2
78. Mathelier A, Carbone A (2010) MiReNA: finding microRNAs with high accuracy and no learning at genome scale and from deep sequencing data. *Bioinformatics* 26(18):2226–2234
79. Konig J, Zarnack K, Luscombe NM et al (2011) Protein-RNA interactions: new genomic technologies and perspectives. *Nat Rev Genet* 13(2):77–83
80. Hafner M, Landthaler M, Burger L et al (2010) Transcriptome-wide identification of RNA-binding protein and microRNA target sites by PAR-CLIP. *Cell* 141(1):129–141

81. Khorshid M, Rodak C, Zavolan M (2011) CLIPZ: a database and analysis environment for experimentally determined binding sites of RNA-binding proteins. *Nucleic Acids Res* 39(Database issue):D245–D252
82. Wu J, Liu Q, Wang X, Zheng J, Wang T, You M, Sheng Sun Z, Shi Q (2013), miTools 2.0 for non-coding RNA discovery, profiling, and functional annotation based on high-throughput sequencing. *RNA Biol* 10(7): 1087–1092
83. Hackenberg M, Rodriguez-Ezpeleta N, Aransay AM (2011) miRanalyzer: an update on the detection and analysis of microRNAs in high-throughput sequencing experiments. *Nucleic Acids Res* 39(Web Server issue):W132–W138
84. Langmead B, Trapnell C, Pop M et al (2009) Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biol* 10(3):R25
85. Zhao W, Liu W, Tian D et al (2011) wapRNA: a web-based application for the processing of RNA sequences. *Bioinformatics* 27(21):3 076–3077
86. Vlachos IS, Hatzigeorgiou AG (2013) Online resources for miRNA analysis. *Clin Biochem* 46(10–11):879–900. doi:[10.1016/j.clinbiochem.2013.03.006](https://doi.org/10.1016/j.clinbiochem.2013.03.006), Epub 2013 Mar 18. Review. PMID: 23518312 [PubMed—indexed for MEDLINE]
87. Oulas A, Boutla A, Gkirtzou K et al (2009) Prediction of novel microRNA genes in cancer-associated genomic regions – a combined computational and experimental approach. *Nucleic Acids Res* 37(10):3276–3287
88. Simoneau M, Aboulkassim TO, LaRue H et al (1999) Four tumor suppressor loci on chromosome 9q in bladder cancer: evidence for two novel candidate regions at 9q22.3 and 9q31. *Oncogene* 18(1):157–163
89. Han Y, Chen J, Zhao X et al (2011) MicroRNA expression signatures of bladder cancer revealed by deep sequencing. *PLoS One* 6(3): e18286
90. Kapranov P, Cheng J, Dike S et al (2007) RNA maps reveal new RNA classes and a possible function for pervasive transcription. *Science* 316(5830):1484–1488

Chapter 14

Using Deep Sequencing Data for Identification of Editing Sites in Mature miRNAs

Shahar Alon and Eli Eisenberg

Abstract

Deep sequencing has many possible applications; one of them is the identification and quantification of RNA editing sites. The most common type of RNA editing is adenosine to inosine (A-to-I) editing. A prerequisite for this editing process is a double-stranded RNA (dsRNA) structure. Such dsRNAs are formed as part of the microRNA (miRNA) maturation process, and it is therefore expected that miRNAs are affected by A-to-I editing. Indeed, tens of editing sites were found in miRNAs, some of which change the miRNA binding specificity. Here, we describe a protocol for the identification of RNA editing sites in mature miRNAs using deep sequencing data.

Key words miRNA, A-to-I editing, Deep sequencing, Bioinformatics, Genomics

1 Introduction

Adenosine to inosine (A-to-I) editing is catalyzed by enzymes of the adenosine deaminase that act on RNA (ADAR) family, which posttranscriptionally changes adenosine to inosine, the latter being treated by cell machinery similar to guanosine [1]. As ADARs bind to double-stranded RNA (dsRNA), they may act on the double-strand formation of the primary transcript of a microRNA (miRNA) gene (pri-miRNA) [2]. A-to-I editing of pri-miRNA can affect the processing of the pri-miRNA to precursor miRNA (pre-miRNA) or the processing of the pre-miRNA to mature miRNA [2]. The mature miRNA is a single-stranded RNA, ~21 bases long, that can bind to partially complementary targets located in the 3' untranslated region of specific mRNAs. Its binding blocks the translation or triggers degradation of the target mRNAs [3]. Pri-miRNA and pre-miRNA editing events may lead to an edited version of the mature miRNAs. Bases 2–8 at the 5' end of the mature miRNA were found to be critical for the target recognition [3]. Therefore, if the editing occurs in this recognition site, known as the “seed” region, a change in the set of target genes is expected. A striking

example is mouse miR-376 in which editing in the recognition site alters the target specificity of the miRNA and profoundly affects cellular processes [4]. Editing outside the recognition site can also be of interest as it may affect the loading of the mature miRNAs to the RNA-induced silencing complex (RISC), thereby altering the effectiveness of the mature miRNAs [1]. It should be mentioned that it was recently discovered that mature miRNAs undergo large-scale RNA modifications in the form of adenylation and uridylation in the 3' end [5]. However, the functional significance of these changes is not fully understood.

In addition to A-to-I editing, it is possible that other types of RNA editing modify the mature miRNA sequence. Proper utilization of deep sequencing data has the potential to unravel the full extent of A-to-I editing in miRNAs as well as other types of RNA editing [6–8]. However, technical problems in the analysis of deep sequencing datasets may result in a false detection of miRNA modification events [9]. Indeed, recent deep sequencing studies have raised a debate surrounding the extent of these other types of modifications in the human transcriptome in general and in miRNAs in particular [10–13]. Here we describe a protocol which utilizes deep sequencing data to characterize RNA editing in mature miRNAs while avoiding technical data analysis problems [13].

2 Materials

1. Hardware: 64-bit computer running Linux (including other UNIX-based operating systems and Cygwin inside Windows) or Mac OS with at least 8 GB of RAM (preferably 16GB).
2. Short read alignment program: In this protocol we use Bowtie [14] which can be downloaded for Linux (*see Note 1*) or Mac OS from <http://bowtie-bio.sourceforge.net/index.shtml>. The Bowtie Web site provides installation instructions.
3. Genome indexes: The alignment program Bowtie uses genome indexes to speed up computation time. Pre-built genome indexes for many organisms can be downloaded via the Bowtie Web site (mentioned above). The Mouse genome indexes (UCSC mm9) are needed for the example below. Therefore, they should be downloaded and placed in the “indexes” folder inside the Bowtie folder.
4. Software: The Perl programming language should be preinstalled. Perl for Linux or Mac OS can be downloaded from ActiveState <http://www.activestate.com/activeperl>. The ActiveState Web site provides installation instructions.
5. Mathematical package: The Perl package Math-CDF is required for the Perl scripts. This package can be downloaded from CPAN (<http://www.cpan.org/>). The CPAN Web site provides installation instructions.

6. Scripts: Three Perl scripts are required for this protocol:

http://www.tau.ac.il/~elieis/miR_editing/Process_reads.pl
http://www.tau.ac.il/~elieis/miR_editing/Analyze_mutation.pl
http://www.tau.ac.il/~elieis/miR_editing/Binomial_analysis.pl

7. miRNA definitions files: For the example below, download these three text files which contain data about mouse miRNAs: the mature/star sequences, the alignment of the pre-miRNA sequence against the genome, and the secondary structure of the pre-miRNA:

http://www.tau.ac.il/~elieis/miR_editing/Mature_file.txt
http://www.tau.ac.il/~elieis/miR_editing/Mouse_pre_miRNA_aligned_against_the_genome.txt
http://www.tau.ac.il/~elieis/miR_editing/RNAfold_file_mouse.txt

One may place these files in the same working folder as the Perl scripts for the purpose of the example below.

8. Sequencing data: This protocol refers to the output of Illumina sequencing machines. As the following analysis relies on the quality scores of the sequencing reads, the Fastq formatted output of the sequencing run is required. To run the example given below, download the dataset SRR346417 from the SRA NCBI <http://www.ncbi.nlm.nih.gov/sra/> (see Note 2). Place the downloaded file in the same folder as the Perl scripts.

3 Methods

3.1 Filtering Low-Quality Reads and Trimming Sequence Adapters

The Fastq file of the sequencing reads could be the raw reads without any filter, or filtered reads using Illumina software tools. In the latter case, proceed directly to Subheading 3.2. Otherwise, continue with this step. Raw sequencing reads likely contain parts of the adapter sequence. Therefore, these sequences must be identified and removed. Moreover, low-quality reads (as defined by the read quality score) are unlikely to be informative and therefore should be removed. There are several published tools for raw Fastq filtering [15, 16], or one can use our in-house filtering script “Process_reads.pl.” The script allows the user to define “low-quality reads” by choosing (a) a quality score cutoff and (b) the maximum number of locations allowed to have lower quality score compared to the chosen cutoff. For example, one can use a cutoff of 20 for the Phred quality score (which ranges from 0 to 40) and the maximum number of locations can be set to 3 [13]. The script also trims the adapters (see Note 3) and the resulting trimmed read can be filtered if it is too long or too short (as defined by the user). For example, as the expected length of mature miRNAs is ~21 bases, one can remove reads with length longer than 28 bases or shorter than 15 bases.

As an example, we will use the publicly available dataset of mature miRNAs from the mouse cerebellum (accession number SRR346417).

1. Use the head command to view the Fastq file (*see Note 4*):

```
$ head -100 sra_data.fastq
```

it is clear that these are raw, 36 bases long, reads.

2. Run the Perl script “Process_reads.pl” (*see Note 5*):

```
$ perl Process_reads.pl sra_data.fastq sra_data_filtered.fastq
```

the output on the screen should indicate that from ~25 million reads, ~19 million reads passed the filtering procedure.

RUN TIME: ~10 min (using Intel Core i7 processor).

3.2 Aligning the Reads Against the Genome

The filtered and trimmed reads should be aligned against the genome of interest and not against an miRNA sequence database [13]. We require unique best alignment, that is, the reads cannot be aligned to other locations in the genome with the same number of mismatches (Fig. 1). Lastly, we recommend using only alignments with up to one mismatch (in our datasets, allowing up to two mismatches did not aid in detecting more editing sites; instead, it added unreliable alignments). These steps taken together solve, by and large, the cross mapping problem that significantly hinders identification of true editing sites in mature miRNAs [9, 13].

The last two bases (the 3' end) of mature miRNA undergo extensive adenylation and uridylation [5]. It is thus recommended that these bases will not be considered in the alignment. Naturally, doing so prevents detection of editing in these locations. However, not taking this measure and still demanding low number of mismatches will severely reduce the number of alignments obtained.

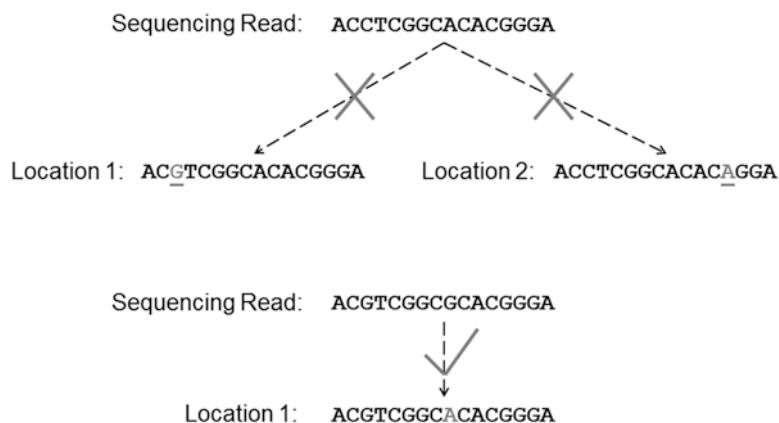


Fig. 1 A schematic representation of the alignment procedure. The sequencing reads should not be aligned to more than one location in the genome with the same number of mismatches

The following command line runs the alignment program Bowtie in a manner compatible with the above considerations (*see Notes 6–8*):

```
$ bowtie -p 8 -n 1 -e 50 -a -m 1 --best --strata --trim3 2 The_bowtie_folder/indexes/mm9 sra_data_filtered.fastq > sra_data_filtered.output
```

The output on the screen should indicate that from ~19 million reads, ~8 million reads were uniquely aligned against the mouse genome.

RUN TIME: ~1 min (using Intel Core i7 processor with eight threads)

3.3 Mapping the Mismatches Against the Pre-miRNA Sequences

The purpose of this step is to move from reads aligned against the genome (the end-point of the previous step) to counts of each of the four possible nucleotides at each position along the pre-miRNA sequence, for all the pre-miRNAs. Performing this transformation will allow us to focus our analysis only on bona fide miRNA and to use, in the following step, binomial statistics to detect significant modifications inside them.

We use pre-built files for the transformation: the alignment of the pre-miRNA sequences against the genome and the mature/star sequences of miRNAs. Taken together, these files give the location of the pre-miRNA and the mature/star miRNA inside the genome (*see Notes 9 and 10*).

The pre-built files and the Bowtie output file from the previous step are the input for the script “Analyze_mutation.pl.” A key user-defined input for this script is the minimum quality score allowed in the location of the mismatch in order for it to be counted. Naturally, the higher this number is, the lower the probability for sequencing error. However, taking very high quality score filter will give small number of counted modifications. We suggest using 30 as the minimum quality score allowed (*see also below and [13]*).

Running this script on large sequencing datasets (large Bowtie output files) requires extensive internal memory resources. A way around this hurdle is to divide the sequencing dataset into several smaller files; afterwards the output files can be merged (*see Subheading 3.4*). We suggest dividing the Bowtie output file if the number of reads is higher than 2.5 million when using 8 GB of RAM.

1. As in our example the Bowtie output file has ~8 million reads, dividing the file is required (*see Note 11*):

```
$ split -l 2500000 sra_data_filtered.output part_
```

This command will divide the Bowtie output file into 4 files: part_aa, part_ab, part_ac, and part_ad.

2. The script “Analyze_mutation.pl” can now be run:

```
$ perl Analyze_mutation.pl part_aa main_output_a.txt
```

RUN TIME: ~20 min (using Intel Core i7 processor with 8 GB of RAM).

3. Run the other parts of the Bowtie output file:

```
$ perl Analyze_mutation.pl part_ab main_output_b.txt
```

```
$ perl Analyze_mutation.pl part_ac main_output_c.txt
```

```
$ perl Analyze_mutation.pl part_ad main_output_d.txt
```

RUN TIME: ~60 min (using Intel Core i7 processor with 8 GB of RAM).

The main output of this script is a text file containing the counts of each of the four possible nucleotides at each position along all the pre-miRNA sequences. This output file is later used for the binomial test (*see Note 12*).

The binomial statistics requires the number of successes (the number of mismatches of a given type in a given position), the number of trials (the total number of counts in the given position), and the sequencing error probability. The sequencing error probability can be estimated using the Phred score. For example, if only mismatches with Phred score of 30 are allowed, the expected base call error rate should be 0.1 % in each position. The aligned read data can be used in order to examine the actual sequencing error in the retained reads (*see Note 13*).

3.4 Using Binomial Statistics to Remove Sequencing Errors

In this step binomial statistics is performed on the output file of the previous step to separate sequencing errors from statistically significant modifications (Fig. 2).

Importantly, binomial statistics do not require any arbitrary expression level filter. It is well suited even for low-expressed miRNAs with low number of sequencing reads, and the *P*-values computed reflect the absolute number of reads detected, small or large as the case may be.

This analysis is performed for every position (except the last two positions of the miRNA due to the extensive adenylation and uridylation) in every mature/star miRNA separately. As multiple tests are performed, the resulting *P*-value for each position must be corrected accordingly. The script performing this analysis, “Binomial_analysis.pl,” gives the user an option to use Bonferroni or Benjamini–Hochberg corrections.

1. Run the script “Binomial_analysis.pl” using the files from the previous step (*see Notes 14 and 15*):

```
$ perl Binomial_analysis.pl main_output_a.txt main_output_b.txt main_output_c.txt main_output_d.txt > binomial_output.txt
```

RUN TIME: <1 min (using Intel Core i7 processor).

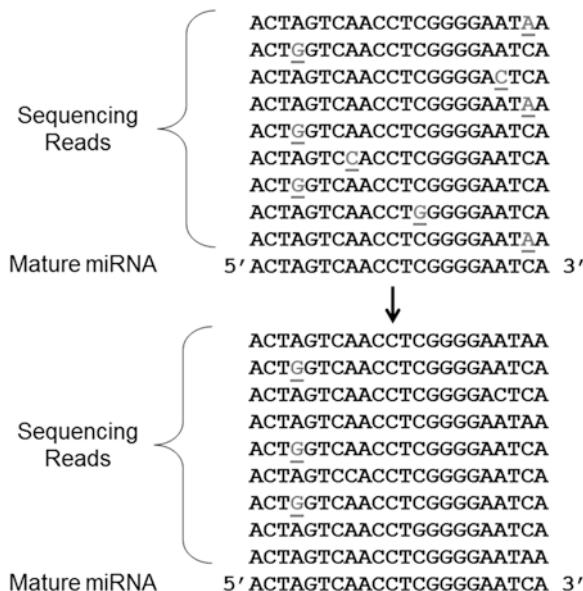


Fig. 2 A schematic representation of the mismatch filtering step. Sequencing errors, 3' adenylation and uridylation events, and SNPs are removed, leaving only possible RNA editing events

2. View the output file “binomial_output.txt” using text editor. The locations of the significant modification are given as well as the statistical description of the modifications (*see Note 16*). Overall, in this example 38 modifications should appear (*see Note 17*). Most of the modifications (30 out of the 38) are A-to-G, which can be a result of A-to-I editing. What can be the nature of the other types of modifications? Among the possible explanations are the following: (a) problems in the definition of the miRNA (*see Note 18*), (b) 5' adenylation and uridylation (*see Note 19*), (c) SNPs or somatic mutations (*see Subheading 3.5*), or (d) non-A-to-I RNA editing events.

3.5 Removing SNPs from the List of Statistically Significant Modifications

Known SNPs may be filtered from the statistically significant modifications detected in the previous step. This can be done using Galaxy (<http://galaxy.psu.edu/>).

1. In the Galaxy site choose the “Use Galaxy” option.
2. Use the “Get Data” link, and then “UCSC Main” to download data from the UCSC Table Browser.
3. To download the mouse SNP dataset, choose “Mouse” in the “genome” field and “Variation and Repeats” in the “group” field. Make sure that the “track” field is set to “SNPs” and the “region” field is set to “genome.” Then press “get output” and afterwards press “send query to Galaxy.”

4. Upload the output file of the previous step, “binomial_output.txt,” using the “Get Data” link, and then “Upload File” link. Choose “interval” in the “File Format” field and locate the file using the “Browse” button. Choose “Mouse mm9” in the “Genome” field and then press “Execute.”
5. Subtract the genomic locations of the SNPs from the locations of the identified mismatches. Use “Operate on Genomic Intervals” and then “Subtract.” Choose to subtract the SNP dataset from the identified mismatch dataset, make sure that the minimal overlap is set to 1 base, and then press “Execute.” After a while the subtracted file should appear as a new tab in the “History” panel (right side of the screen).
6. View the resulting subtracted file. The file should appear in the upper right corner of the screen under the “History” tab. It can be viewed using the eye icon. In this case, the resulting file should be identical to the file “binomial_output.txt,” meaning that none of the identified modifications are known SNPs.

RUN TIME: ~30 min (the execution time depends on the speed of the Internet connection and the current load of the Galaxy servers).

TOTAL RUN TIME: ~2 h (using Intel Core i7 processor with 8 GB of RAM).

4 Notes

1. By Linux we also refer to UNIX-based operating systems and Cygwin inside Windows.
2. Downloading the sequencing data from the NCBI SRA: (a) Type the run accession number (starts with SRR) or the experiment accession number (starts with SRX). Alternatively, type the study accession number (starts with SRA) or search by keywords and then choose the experiment of interest. (b) Use the “Run” link to choose the run of interest. (c) Choose “Filtered Download.” (d) Choose FASTQ as the “Download Format” and press “Download.” (e) Unzip the file “sra_data.fastq.gz” using gunzip command (in Linux and Mac). The “Filtered Download” tool can also allow the user to filter the obtained reads as explained in the NCBI SRA site. The download process can be made faster by using the Aspera plug-in using the link in the NCBI SRA site.
3. Although Illumina small RNA Adapters are universal, different versions of the protocol may result in variations in the adapter sequences. The user should check the adapter sequences given in the Perl script “Process_reads.pl” and change them if needed. Viewing and editing the Perl script can be done by using a text editor.

4. Commands meant to be executed from the Linux command line are prefixed with a “\$” character. The commands are meant to be run from the example working directory.
5. Edit the Perl script “Process_reads.pl” using a text editor if a change in the default parameters is required.
6. Note that the allowed Phred score in the mismatch position is not limited (-e 50). This means that even mismatches with high quality score will be allowed to align.
7. The -p option of Bowtie allows the use of multiple threads in parallel. The number of available threads is naturally dependent on the hardware.
8. By default, the Fastq file is treated by Bowtie as written in Phred+33 format. Phred+33 format means that the number 33 is added to the Phred score and the result is translated into single ASCII character. Note that previous versions of the Illumina software produced Fastq files in Phred+64 format. Therefore, it is recommended to view the Fastq file (using, for example, the Linux “head” command) to determine the Phred to ASCII conversion. If the Fastq file is in Phred+64 format, the flag --phred64-quals should be added to the Bowtie execution line.
9. Pre-built files for human are given below:

http://www.tau.ac.il/~elieis/miR_editing/Human_pre_miRNA_aligned_against_the_genome.txt

http://www.tau.ac.il/~elieis/miR_editing/RNAfold_file_human.txt

Open the Perl script “Analyze_mutation.pl” using a text editor and change the names of the pre-built miRNA files accordingly.

10. The miRBase release 17 was used for the human and mouse definition files. Human and mouse definition files for miRBase release 18 are available in

http://www.tau.ac.il/~elieis/miR_editing/Release18.zip

The results of the example described in this protocol are not affected by using the latter release.

It is simple to create the definition files for various other species. The miRNA sequences can be downloaded from miRBase (<http://www.mirbase.org/>), and the alignment of the pre-miRNA sequence against the genome can be achieved using Bowtie. Finally, RNAfold [17] can be used to identify the secondary structure of the pre-miRNA.

11. The -l option in the Linux split command allows dividing a file by the number of lines. Splitting by line (and not by size) is required in order to prevent cutting the output lines in the end of the file.

12. This output representation can also be used to detect significant variation from the miRBAse mature/star definitions (in terms of beginning and end locations). Only modifications inside the mature/star sequence as defined in miRBAse are analyzed in the script. Additional output file is a text file containing the number of reads for each mature/star miRNA; this file (termed “miRNA_expression.txt”) can be used for expression analysis.
13. Merging the data from all the mature and star miRNAs, the rate of any type of mismatch from the 12 possible mismatches can be calculated. In this calculation the location of the mismatch should also be taken into account, as biological modifications appear in the 3' of the miRNA and sequencing quality is lower towards the end of the read. Therefore, the calculation should be performed separately in each position along the mature/star miRNA. The results are presented in the main output file of “Analyze_mutation.pl.” The rate relevant for the statistical analysis to follow is the maximal mismatch rate of any single type of mismatch. If it is higher than the rate expected from the Phred score, the higher estimate should be used [13].
14. The Perl script “Binomial_analysis.pl” can process any number of input files. The counts of each of the four possible nucleotides at each position along the pre-miRNA sequence, for all the pre-miRNAs, are merged and analyzed together. Naturally, the input files should all be created using the same miRNA definition files (*see* Subheading 3.3).
15. Additional input file for the script “Binomial_analysis.pl” is the alignment of the pre-miRNA against the genome. This file is required to translate the location of the mismatch to genomic coordinates. *See* Subheading 2, item 7, and Note 9 for pre-built files for human and mouse. If change in organism is required, open the Perl script using a text editor and change the names of the pre-built miRNA files accordingly.
16. The genomic locations in the output of “Binomial_analysis.pl” are in the format, Chromosome (tab) Start location (tab) End location, for example chr16 (tab) 77599184 (tab) 77599185. The mismatch is located in the End position, that is, in base 77599185 in this example. This format, which includes Start location and End location with 1 base separating them, was chosen to be compatible with the genomic representation of SNPs in the UCSC Table Browser (*see* Subheading 3.5).
17. The identity of the detected modifications depends on the specific parameters used in “Binomial_analysis.pl”: the estimate of the sequencing error probability, the multiple correction type, and the *P*-value used as a cutoff after multiple correction. Here, the parameters used were 0.001, Bonferroni correction, and 0.05, respectively.

18. miRNAs with inconsistent definitions can be identified by visually inspecting the distribution of the reads along the pre-miRNAs in question. This data can be viewed in the output files of “Analyze_mutation.pl,” for example, “main_output_a.txt.” In addition, miRBase (<http://www.mirbase.org/>) can be used to view the distribution of reads along each pre-miRNA using publicly available deep sequencing datasets. For example, miR-5105 was identified as modified in position 22, but the distribution of the reads along the pre-miRNAs does not have a single peak around the mature miRNA sequence. Instead, the reads appear to come from multiple regions inside the pre-miRNA, which may suggest that this is not a bona fide miRNA. The distribution of reads along the pre-miRNA of miR-5115 indicates that the actual location of the mature miRNA is significantly different from the location mentioned in miRBase. Therefore, the modifications in miR-5105 and miR-5115 should be taken with caution.
19. We note that some low-abundance isomirs display 5' sequence modifications similar to the biological modifications reported at the 3' of mature miRNAs. We previously identified several isomers that start one or two bases upstream from a different isomer and display sequence modifications in the 5' end in the form of adenylation and uridylation. In all these events the abundance of the modified isomer was at least an order of magnitude lower than the unmodified isomer [13]. The script “Binomial_analysis.pl” automatically identifies these events and discards them from the analysis. However, the file “binomial_output.txt” contains three modifications which are likely to be a result of 5' adenylation and uridylation: in miR-126, miR-337, and miR-23b. Viewing the distribution of reads along the pre-miRNAs (*see Note 18*), it can be seen that the modifications in miR-337 and miR-23b are in isomirs which are more than an order of magnitude lower compared to the expression levels of the main isomer. They were not filtered out because they are located in positions 4 and 3 along the mature miRNA sequence, respectively, whereas the maximal position to discard is set to 2 in “Binomial_analysis.pl.” miR-126 was not filtered out because the fold change between the 5' isomir and the main isomer was only ~5 whereas the default is set to 10 in “Binomial_analysis.pl.” These parameters can be easily changed by editing “Binomial_analysis.pl” using a text editor.

Acknowledgments

This work was supported by a grant from the Israel Science Foundation [379/12] and a grant from the United States-Israel Binational Science Foundation (grant number 2009/290), Jerusalem, Israel, to E.E. S.A. was supported by a Clore Fellowship.

References

1. Nishikura K (2010) Functions and regulation of RNA editing by ADAR deaminases. *Annu Rev Biochem* 79:321–349
2. Yang W, Chendrimada TP, Wang Q et al (2006) Modulation of microRNA processing and expression through RNA editing by ADAR deaminases. *Nat Struct Mol Biol* 13:13–21
3. Bartel DP (2004) MicroRNAs: genomics, biogenesis, mechanism, and function. *Cell* 116: 281–297
4. Kawahara Y, Zinshteyn B, Sethupathy P et al (2007) Redirection of silencing targets by adenosine-to-inosine editing of miRNAs. *Science* 315:1137–1140
5. Burroughs AM, Ando Y, de Hoon MJL et al (2010) A comprehensive survey of 3' animal miRNA modification events and a possible role for 3' adenylation in modulating miRNA targeting effectiveness. *Genome Res* 20:1398–1410
6. Ekdahl Y, Farahani HS, Behm M et al (2012) A-to-I editing of microRNAs in the mammalian brain increases during development. *Genome Res* 22:1477–1487
7. Warf MB, Shepherd BA, Johnson WE et al (2012) Effects of ADARs on small RNA processing pathways in *C. elegans*. *Genome Res* 22:1488–1498
8. Vesely C, Tauber S, Sedlazeck FJ et al (2012) Adenosine deaminases that act on RNA induce reproducible changes in abundance and sequence of embryonic miRNAs. *Genome Res* 22:1468–1476
9. de Hoon MJL, Taft RJ, Hashimoto T et al (2010) Cross-mapping and the identification of editing sites in mature microRNAs in high-throughput sequencing libraries. *Genome Res* 20:257–264
10. Li M, Wang IX, Li Y et al (2011) Widespread RNA and DNA sequence differences in the human transcriptome. *Science* 333:53–58
11. Lin W, Piskol R, Tan MH et al (2012) Comment on “Widespread RNA and DNA sequence differences in the human transcriptome.”. *Science* 335:1302
12. Piskol R, Peng Z, Wang J et al (2013) Lack of evidence for existence of noncanonical RNA editing. *Nat Biotechnol* 31:19–20
13. Alon S, Mor E, Vigneault F et al (2012) Systematic identification of edited microRNAs in the human brain. *Genome Res* 22: 1533–1540
14. Langmead B, Trapnell C, Pop M et al (2009) Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biol* 10:R25
15. Stocks MB, Moxon S, Mapleson D et al (2012) The UEA sRNA Workbench: a suite of tools for analysing and visualising next generation sequencing microRNA and small RNA datasets. *Bioinformatics* 28:2059–2061
16. Zhu E, Zhao F, Xu G et al (2010) mirTools: microRNA profiling and discovery based on high-throughput sequencing. *Nucleic Acids Res* 38:W392–W397
17. Zuker M, Stiegler P (1981) Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. *Nucleic Acids Res* 9:133–148

Chapter 15

NGS-Trex: An Automatic Analysis Workflow for RNA-Seq Data

Ilenia Boria,* Lara Boatti,* Igor Saggesse, and Flavio Mignone

Abstract

RNA-Seq technology allows the rapid analysis of whole transcriptomes taking advantage of next-generation sequencing platforms. Moreover with the constant decrease of the cost of NGS analysis RNA-Seq is becoming very popular and widespread. Unfortunately data analysis is quite demanding in terms of bioinformatic skills and infrastructures required, thus limiting the potential users of this method.

Here we describe the complete analysis of sample data from raw sequences to data mining of results by using NGS-Trex platform, a low user interaction, fully automatic analysis workflow. Used through a web interface, NGS-Trex processes data and profiles the transcriptome of the samples identifying expressed genes, transcripts, and new and known splice variants. It also detects differentially expressed genes and transcripts across different experiments.

Key words Next-generation sequencing, RNA-Seq, Transcriptome profile, Differentially expressed genes, Gene expression profiling, NGS-Trex

1 Introduction

RNA-Seq technology enables to fix a snapshot of RNA expressed in a cell taking advantage of next-generation sequencing platforms. RNA-Seq is emerging as the major quantitative transcriptome profiling system [1, 2] but it also allows the detection of differentially expressed genes and transcripts across different experiments [3]. Moreover—unlikely microarray technique—it is not biased by a priori knowledge, thus making possible the identification of unannotated transcripts and splice variants.

Reads obtained after sequencing may be assembled into transcripts or can be used to identify known genes and transcripts available in databases. The first approach is mainly used when a reference genome is not available for the organism under investigation.

*Author contributed equally with all other contributors.

This technique is known as de novo transcriptome assembly. On the other hand whenever a reference genome (and its annotation) is available this information can be used to drive the assembly and the identification process. The latter case is known as genome-guided transcriptome analysis and is the focus of this manuscript.

A conceptual RNA-Seq analysis pipeline is relatively easy to be drawn, and four main steps can be identified: Filtering/Demultiplexing, Mapping, Annotation, and Data Mining.

First of all, raw sequencing data must be filtered to remove low-quality reads and cloning linkers. Moreover, since samples are often multiplexed in a single sequence run, a demultiplexing process is required to subdivide the reads in their respective sample datasets. High-quality reads are then mapped onto the reference genome and are compared to annotation to assign them to known genes and transcripts. By comparing reads with gene models it is possible to quantify known elements and to detect unannotated features such as splice variants. Finally differential expression at both gene and transcript level can be estimated by comparing reads assigned to genes in different datasets.

While the conceptual analysis pipeline to deal with NGS data is relatively easy to be drawn, the actual implementation can be challenging. Big efforts in terms of infrastructure and specific skills are in fact required to handle the large amount of data produced by high-throughput sequencing, thus making difficult for researchers to fully exploit the potential of RNA-Seq/NGS technologies.

To date several software packages and bioinformatics tools have been developed to address this issue; they can be categorized into command-line-based packages, GUI-based software, and web-based applications. Command-line- and GUI-based stand-alone packages require dedicated hardware resources and software installation, while web-based tools are ready to be used. Currently available web-based systems include Galaxy [4], GeneProf [5], and NGS-Trex [6] all of them with various advantages and drawbacks.

In the following section we will show the steps required to analyze an RNA-Seq dataset using NGS-Trex system available at www.ngs-trex.org.

2 Materials

2.1 RNA-Seq Datasets

For our sample analysis we will process raw RNA-Seq data obtained by Birzele et al. [7]. Briefly, the authors have used Illumina's Genome Analyzer platform to explore the differences in transcriptome profile between resting and activated human CD4+ T-helper and regulatory T-cells, including differential expression of genes and splice variants.

Raw sequence files, corresponding to the four samples examined by Birzele et al., are available at the NCBI Sequence Read Archive (SRA) under accession number SRP006674. To download sequences

Table 1

List of datasets described by Birzele et al. Each dataset is the result of the forward and reverse paired end sequences merged together

Dataset	Accession number	Description	Number of reads
SRR192532	SRR192532_1	mRNAseq of resting Human CD4 Th cells	15,277,248
	SRR192532_2		
SRR192536	SRR192536_1	mRNAseq of activated Human CD4 Th cells	10,335,712
	SRR192536_2		
SRR192425	SRR192425_1	mRNAseq of resting Human Treg cells	13,655,820
	SRR192425_2		
SRR192534	SRR192534_1	mRNAseq of activated Human Treg cells	8,237,652
	SRR192534_2		

in fastq format it is more easy to use the European Nucleotide Archive (ENA) hosted at the European Bioinformatic Institute (www.ebi.ac.uk/ena) (see Note 1). By inserting the SRP code of the dataset in the “text search” form the user is redirected to a table with a link to the fastq files. For each dataset two distinct files—containing the forward and the reverse paired end sequences (file 1 and file 2)—are available. We need to download both files for each dataset. Table 1 summarizes the downloaded files.

2.2 Computational Software

As described in 6 NGS-Trex is an automatic system to store, analyze, and mine RNA-Seq data. It is available through a simple web interface (www.ngs-trex.org) and allows the user to upload raw sequences files in fastq/fasta format and easily obtain an accurate characterization of the transcriptome profile after the setting of few parameters. The system is also able to assess differential expression at both gene and transcript level (i.e., splicing isoforms) by comparing the expression profile of different samples (e.g., normal and disease status).

3 Methods

The overall procedure to analyze RNA-Seq data with NGS-Trex is depicted in Fig. 1. It can be summarized into three major steps: (1) upload of raw sequences; (2) analysis; (3) results exploration and retrieval.

NGS-Trex organizes high-throughput sequencing data into *projects* and each project includes one or more datasets or samples.

To use the system we need to login using a demo account as described in the homepage or we can obtain our personal credentials.

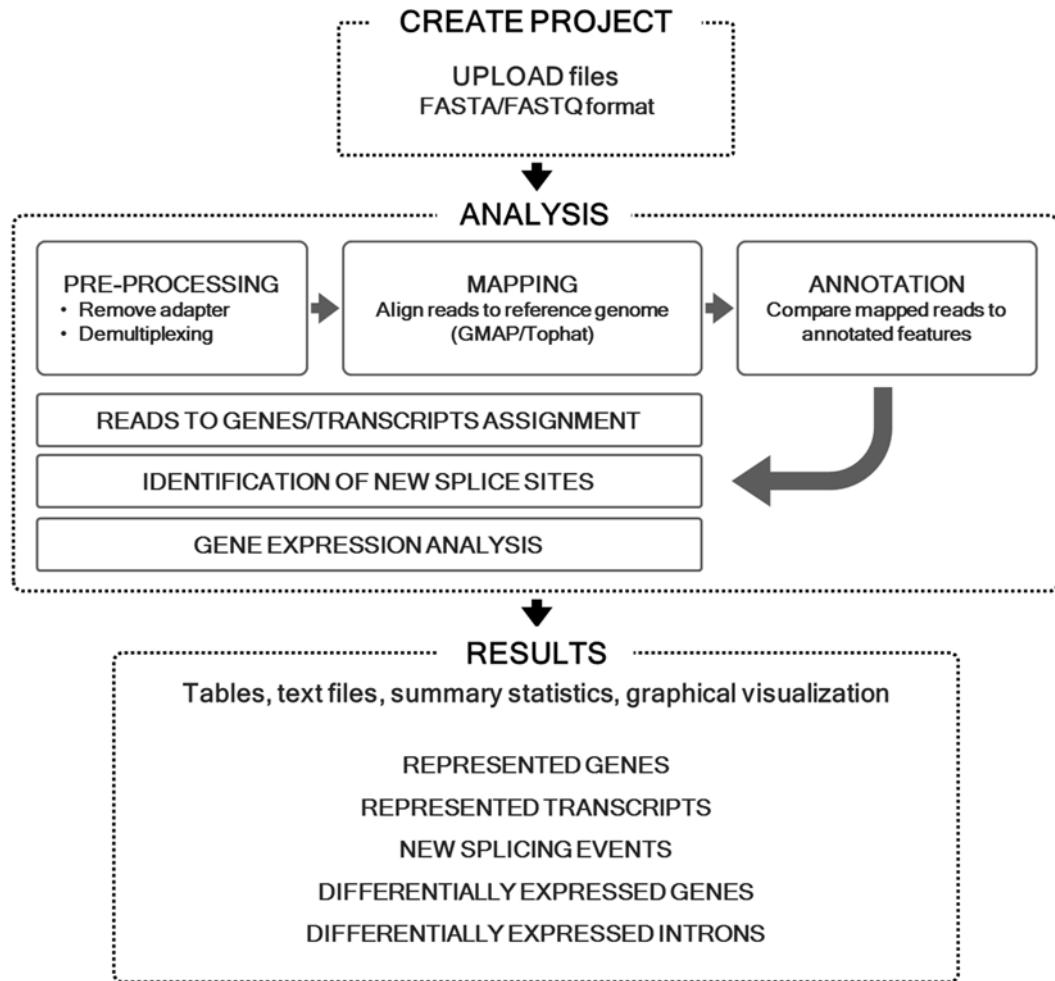


Fig. 1 Schema of NGS-Trex procedure for RNA-Seq data analysis (image modified from original in NGS-Trex web page)

Demo account is fully functional but has two main limitation: (1) the size of each sample is limited to 10 Gb (2) projects and files are “world accessible” so anyone can read and modify our data.

3.1 Data Submission

3.1.1 File Upload

The first step is to upload files to NGS-Trex server. This is the only procedure requiring an external tool. Indeed we need an FTP client (such as FireFTP, FileZilla, or any other client supporting sftp transfer protocol) to upload sequences. The exact procedure depends on the client used, but we have to provide the server name (www.ngs-trex.org) and our credentials (username and password). Only fasta and fastq formats (*.fa, *.tfa, *.fasta, *.fna, *.fq, *.fastq) are supported by the system.

3.1.2 Create a New Project

By clicking the *Create New Project* link we can fill a form with the project name (we enter “SRP006674”), a brief description (“Cell line datasets—Birzele et al. [7] NAR 39(18)”), and we can select the reference genome to be used for the analysis (we select “Homo sapiens (NCBI Build 36)” to use the same release used in Birzele work). On form submission we are redirected to the project main page. This page includes three sections: (1) the project summary: with the description of the project and a summary of already analyzed dataset, (2) the list of datasets belonging to the project (which is empty on newly created projects), and (3) the *Add Dataset* form.

3.1.3 Create Datasets

At this stage we need to create the required datasets by selecting the *Add Dataset* panel.

The *Select file to upload* link opens a window that shows a list of the previously uploaded files. It is also possible to upload files through this interface but—given the size of the files involved—it is much more reliable to use the ftp approach described before. As previously noticed, each dataset has two files. NGS-Trex does not explicitly deal with paired end sequences but treats forward and reverse sequences as single stranded. For this reason we need to assign the two distinct files to the same dataset. There is no need to preprocess files as we can select both files (pressing “ctrl” key—“cmd” key on mac OS—while selecting files). For example we can select “SRR192536_1” and “SRR192536_2”, we then right-click on one of the highlighted files and press on “Select” menu item. We then assign “CD4_activated” as label and “CD4 Th activated—Birzele et al. [7]” as short description and press “*Add Dataset*” button.

We can repeat the same procedure for all the other datasets.

Now Dataset panel of SRP006674 project shows a table with our four dataset: “CD4_activated”, “TREG_activated”, “CD4_resting”, and “TREG_resting”. We are ready to analyze our data.

3.2 Analysis

Although NGS-Trex analysis can be run with default options, it is possible to tune many parameters through a simple form accessible via the *Set Analysis Params* link provided—for each sample—in the *Datasets panel* of the project main page. Through this form it is possible to optimize the three main steps of NGS-Trex analysis workflow: (1) sequences preprocessing, (2) mapping criteria to align reads to the reference genome, and (3) annotation strategy to assign mapping reads to annotated genes and transcripts.

3.2.1 Sequences Preprocessing

With sequence preprocessing we can instruct the system to remove cloning linkers and define the rules to deal with reverse stranded reads.

“*Which kind of data do you want to analyze?*” Based on the platform used to obtain the raw data, we can currently select “454” or

“Illumina”. This option controls the algorithm used for the mapping of the reads onto the reference genome.

“*Cut-off value for percentage of read length that should be of given quality*” (default: 70 %). This threshold controls the number of bases of the reads that satisfy the quality threshold defined in the “*Cut-off value for quality score for high-quality filtering*” value (see Note 2).

“*Do you have linker/cloning sequences to remove?*” By answering Yes a form where we can insert the sequence of the 5' and 3' linkers appears. (Both linkers have to be written in 5'-3' orientation.) It is also possible to trim some extra nucleotides between the linker and the main sequence. The system searches for exact matches between the provided linker sequences and the reads.

“*Look for linker in Reverse (if not found in forward)?*” If sequences are not oriented we can expect to find 5' and 3' linkers also with a reverse order.

“*Reverse complement sequence if linker found reversed?*” If linkers are found with reverse order the read can be simply trimmed (maintaining unchanged its orientation) or it can be reverse-complemented to reflect the correct order of linkers. This option can be very critical for the following Annotation step (as described below) as it allows proper orientation of the reads thus making more reliable the Annotation procedure.

For the four datasets of the SRP006674 project, we used the default options in trimming step (since we have no adapters to remove). No quality filtering was applied as described in Birzele et al.

3.2.2 Mapping

“*Which trimmed sequences should be mapped?*” (default: *Map all Reads (also those without linkers)*). We can define which sequences have to be compared to genome after the trimming step choosing among the available options. Indeed we can proceed with the analysis using all reads or we can filter out the reads on the bases of the number of linker identified. In our example this option is irrelevant and we leave the default value.

“*Only consider Reads mapping with similarity*” (default: 95 %). We can define the cut-off value for minimum sequence similarity between the reads and the genome.

“*Only consider Reads mapping with coverage*” (default: 90 %). We can set the minimum overlap of the read with the genome. The value can be set as percentage or as number of nucleotides referred to the length of trimmed read.

“*Allow a mismatch of up to N nt at the 5' end of the read*”. This parameter filters out the reads with more than N unaligned bases at the 5' end. “0” is the most restrictive option while “-1” does not apply any restriction.

“Allow a mismatch of up to N nt at the 3' end of the read”. Like the previous filter but referred to the 3' end of the read.

“Are indels allowed?” We can define whether taking into account also indels within the alignment.

“Only consider Reads with N multiple mappings” (default: 5). This parameter defines the maximum number of times the same read is allowed to map to the genome (useful to remove repeated elements). If we do not want to apply any filter on multiple mapping we can set 0.

For our datasets we used the default values proposed by the NGS-Trex system meaning that among the reads mapped with TopHat, we selected those aligning to the *Homo sapiens* (NCBI Build 36) genome with a similarity of at least 95 % and an overlap of at least 90 % and having a number of multiple matches less than 5 over the genome. Indels have been accepted.

3.2.3 Annotation

The annotation process is a multistep procedure to assign reads to gene-specific clusters.

NGS-Trex performs the assignment with three degrees of confidence “*P*”, “*G*”, and “*T*” (Fig. 2) on the basis of few parameters reported in the annotation section of the analysis setup page.

A read is tagged as “*proximal*”—*P*—if it maps to a region flanking the gene (surrounding) and extending as much as a bounding limit. It is tagged as “*genic*”—*G*—if it overlaps gene coordinates by satisfying some threshold values (described below). Finally genic reads are aligned to RefSeq transcripts and tagged as “*transcript read*”—*T*—if they show contiguous mapping to transcripts.

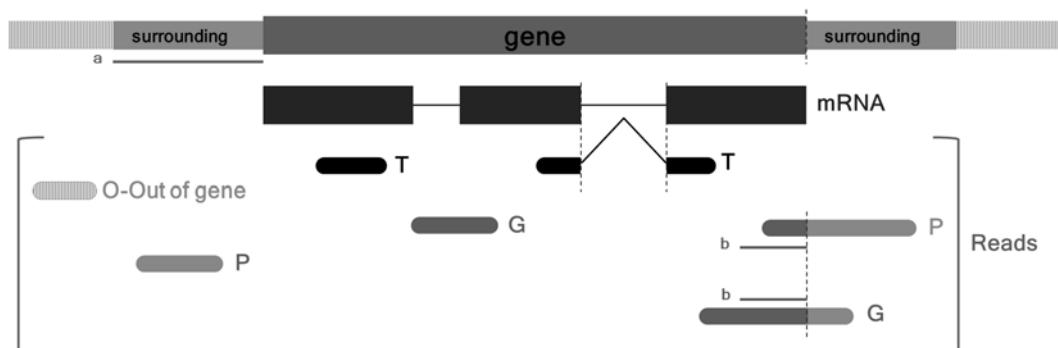


Fig. 2 Read to gene assignment schema reporting proximal reads as “*P*”, genic reads as “*G*”, transcript reads as “*T*”, and extragenic reads as “*O*”. The bounding limit for “*P*” reads is labeled as “*a*” and the minimum overlap with the gene for “*G*” assignment is labeled as “*b*” (image modified from original by Boria et al. (2013))

More in details:

“Gene upstream/downstream region classify Read as ‘Proximal’”. Bounding limit upstream and downstream a gene to classify a read as “P”. Default value is 0 nt which means that “P” reads are not considered.

“Minimum overlap to classify Read as ‘Genic’”. Minimum overlap between a read and a gene to classify the read as “G”. Default value (1 nt) is very tolerant as it implies that a read overlapping a gene with a single nucleotide is labeled as “G”.

“Minimum similarity between Read and Transcript” (default 95 %). Similarity cut-off value between read and RefSeq sequence to label the read as “T”.

“Extension to assign Reads to Transcript” (default: 10 nt). A Read overlapping the RefSeq transcript at its 5' or 3' end can extend it up to N nt. If the read extends the transcripts exceeding this threshold it is not assigned to the transcript but it is treated as a putative (extended) variant of the annotated transcript.

“Trim to assign Reads to Transcript” (default: 3 nt). The maximum number of allowed un-aligned nucleotide at reads ends. Unlike previous filter, this one is restricted to reads not aligning at RefSeq ends.

All sequences mapping outside gene coordinates beyond the defined surrounding region are tagged as “O”—*out of gene*.

“Are sequences oriented?” If reads are oriented (default: N) the relative orientation between the read and the gene is used to optimize the assignment process. This can help the assignment of multiple mapping reads or overlapping genes which are two main problems of the annotation process [8].

By selecting Υ (default value) in “*Try to solve ambiguities:*” field, NGS-Trex attempts to solve ambiguities by using relative orientation of genes and reads and the confidence level of read assignment, being the order from higher to lower “T”, “G”, “P” (Fig. 2). See 6 for detail.

3.2.4 Sequences Post-processing

Once filled the analysis setup form as described, let us click the “*Upload parameters*” button and the “*submit job to analysis queue*”. The progresses of each dataset analysis will be shown in the corresponding status column.

3.3 Data Mining

Upon the completion of the analysis process (required time depends on the size of the datasets and on the load of the server), the status column of the datasets table is set to “*Done*” enabling us to explore the results.

A first overview of the obtained results is displayed as a summary table in the project main page. For each analyzed dataset the table shows the total number of reads in the sample, the reads

average length, the number of mapping reads, and the number of genes identified in the sample. More information can be obtained from “Dataset” panel.

3.3.1 Statistics

By clicking the “*Statistics*” link we can access—for each dataset—several statistical summaries:

“*Analysis Parameters*”. Displays a table summarizing all the parameters used for the analysis.

“*Mapping*”. Flowchart showing the flow of reads among the different steps of the mapping procedure. This diagram makes it easy to monitor the quality of the sequences, as it shows how many reads do not map (or map with low quality) with the reference genome. Information about rRNA like sequences, about repetitive sequences, and spliced reads is also reported.

“*Annotation*”. Flowchart which describes reads assignment at gene and transcript level.

“*Results Overview*”. Table with a summary of some information about the dataset like the number of identified genes, transcripts, new introns, and differentially expressed genes. In Table 2 the example of results overview for the dataset “CD4_activated”.

Table 2
Results overview for dataset “CD4_activated” as reported in *Statistic* page of NGS-Trex

Represented genes	19,688
Represented transcripts	24,463
Differentially expressed genes	15,366
Differentially expressed genes (min number of reads mapping gene region = 10; $pValue < 0.05$)	13,956
Differentially expressed genes (min number of reads mapping known exons = 10; $pValue < 0.05$)	11,061
Differentially expressed genes (min number of reads mapping gene region = 10; $pValue < 0.01$)	12,431
Differentially expressed genes (min number of reads mapping known exons = 10; $pValue < 0.01$)	10,127
Differentially expressed introns ($pValue < 0.05$)	10,379
Differentially expressed new introns ($pValue < 0.05$)	132
Differentially expressed introns ($pValue < 0.01$)	5,169
Differentially expressed new introns ($pValue < 0.01$)	54
New introns	574
Canonic D/A sites	574

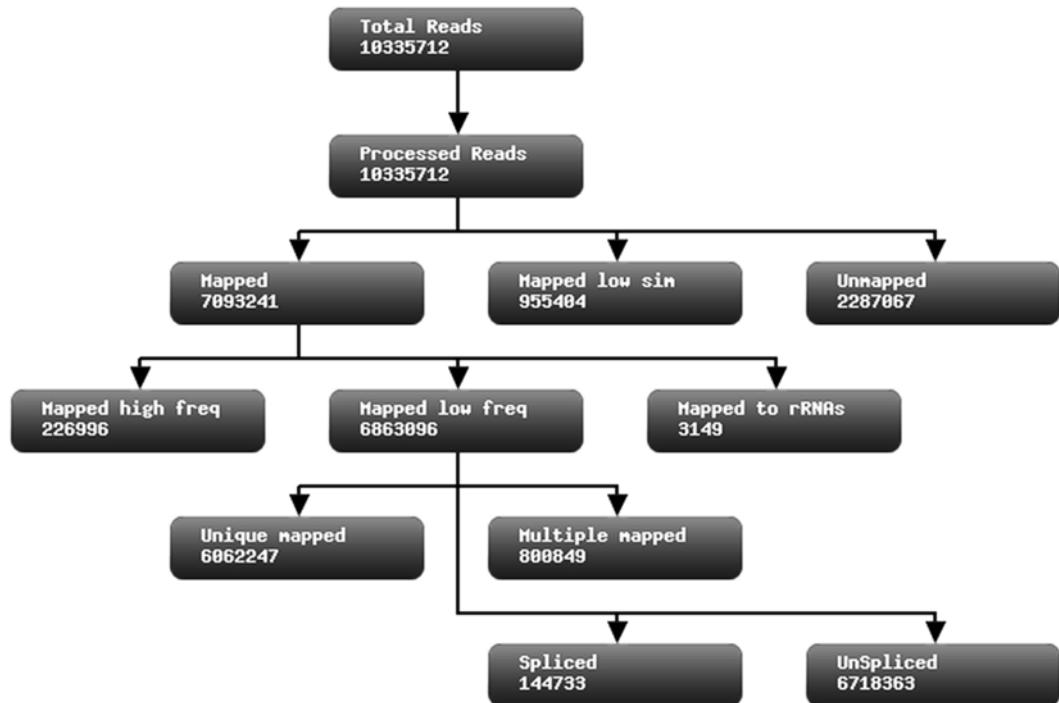


Fig. 3 “*Mapping*” diagram for the dataset “CD4_activated” as reported in the *Statistics* page of NGS-Trex

Let us explore the “*Mapping*” diagram for the dataset “CD4_activated” (Fig. 3). Starting from 10,335,712 “*Total Reads*”, 10,335,712 “*Processed Reads*” were filtered by the preprocessing step and next mapped to the *Homo sapiens* (NCBI Build 36) genome. We obtained 7,093,241 “*Mapped*” reads satisfying mapping criteria (Subheading 3.2.2), 955,404 “*Mapped low sim*” reads aligning with similarity and overlap values below the defined thresholds, and 2,287,067 “*Unmapped*” reads for which no matches have been found in the reference genome.

Before submitting them to the annotation process, the “*Mapped*” reads were filtered to remove those sequences mapping more than five times to the genome, “*Mapped high freq*”, and those aligning to ribosomal RNAs, “*Mapped to rRNAs*”. The resulting 6,863,096 “*Mapped low freq*” sequences represent the total number of reads submitted to the annotation step. Among these sequences 6,062,247 map to one single genome position, “*Unique mapped*”, while 800,849 map to more than one genome position, “*Multiple mapped*”. Moreover 144,733 reads show a spliced alignment: “*Spliced*”, while 6,718,363 map with ungapped match: “*UnSpliced*”.

As shown in the “Annotation” panel of the “CD4_activated” dataset statistics, we obtained 698,448 reads mapping out of gene coordinates, “*Intergenic*”, and 6,164,648 reads “*Genic*” reads including 5,088,923 reads labeled “T”, 1,075,725 labeled “G”, and 0 tagged “P” as described in Subheading 3.2.3.

3.3.2 Query Results

Through the “Query Results” link of the project navigation bar (top left side of the page), it is possible to obtain the most useful information by performing several queries on data. Again the “Query Result” page offers several panels that we will explore in details.

In “Query Gene” panel we can search a single gene using either the Entrez Gene Identifier (EntrezGeneID) or the HUGO accession.

The result is a short summary showing the count of reads assigned to the gene and the count of both annotated and new introns identified by the analysis. Furthermore the differential expression pattern of the selected gene within the different datasets is provided.

In “Advanced Search” panel we can filter the genes with three indexes: depth, coverage, and focus index. Depth is the maximum number of overlapping reads assigned to a gene. Coverage is the total number of reads assigned to the gene. Focus index is the ratio between depth and coverage: if all reads are distributed along the gene the depth value is lower than the coverage and the focus index is low, if all reads are focused on the same region the depth value is similar to the coverage value and the focus index tends to 1.

“Reads mapping within the gene boundaries but not overlapping with known exons should be considered?” To carry out the search we must also decide whether we want to use all reads assigned to genes (G,P,T) answering “Yes” or (with a more restrictive approach) to limit the search to reads labeled as “T” according to the previously described classification (by answering “No”). “All selected samples should satisfy criteria on the same gene?” option searches genes satisfying the filtering criteria in all selected samples (“Yes”) or in at least one of the selected samples (“No”).

For each gene the result table shows: the eg_id, the HUGO name, the sequence coverage, the sequencing depth, the focus index, and the RPKM value.

“DE Genes” and “DE Introns” panels are very similar. They allow the query differentially expressed genes between a reference and the others datasets. We need to select the reference sample, *pValue*, minimum number of reads, and relative expression level (i.e., we can select genes o introns which are overexpressed in reference dataset). As for the advanced search it is possible to limit the query

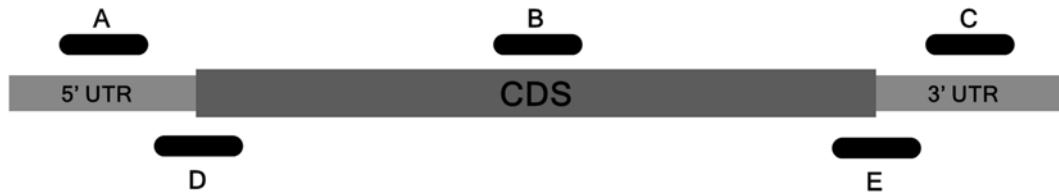


Fig. 4 Relative position of reads with respect to CDS and UTRs regions (Reads 5' UTR = **A**, Reads 5' UTR-CDS = **D**, Reads CDS = **B**, Reads 3' UTR-CDS = **E**, Reads 3' UTR = **C**) in *Transcripts* panel (image modified from original in NGS-Trex web page)

to reads assigned to annotated transcripts (T) or to use all reads assigned to the gene (G,T,P). DE introns can also be queried on specific donor acceptor sites. “Exclude if whole gene shows same pattern?” allows to hide differentially expressed introns if the differential expression is exhibited also at gene level (*see Note 3*).

Output table shows several useful information such as raw number of reads, fold change value, *pValue*. If the output refers to DE splice site also genomic coordinates and donor/acceptor sites are shown.

“*New introns*”. With this query form we can obtain a list of unannotated splicing events. In addition to the basic parameters described for the other panels, we can select whether to consider only splice sites with canonical donor and acceptor dinucleotides.

The resulting table shows the chromosomal location (chromosome name, chromosome start, chromosome end), the strand, the eg_id and the HUGO name of the gene it belongs to, the sequence around the 5' splice site (Donor) and the 3' splice site (Acceptor), and the donor/acceptor dinucleotides (D/A).

“*Transcripts*”. Transcripts query panel allows us to investigate the relation between reads and known transcripts by identifying the relative position of reads with respect to CDS and UTRs regions. The most critical parameter is “*Which reads should be considered?*” form. We can select among six different options as shown in Fig. 4.

Using this tool we get a table reporting for each transcript: the eg_id and the HUGO name of the gene, the RefSeq accession number identifying the transcript, the rna length, the rna strand, the coding sequence location on the transcript (cds start and cds end), the number of reads supporting the transcript, and the mapping position of reads related to the transcript.

We can also limit the result only considering reads aligning with reference transcript with no indels.

For all query panels the results are shown as tables and can be downloaded as tab-delimited text file clicking the “*Download Table*” button at the top right of the “*results*” page. We can also visualize our data through a simple genome browser accessible via

the “eg_id” link corresponding to the genes shown in tables and simply download the reads assigned to a gene or mapped onto a genomic region.

For example if we want to identify genes overexpressed in “TREG_activated” dataset compared to the other datasets we can use the “*DE Genes*” panel. Specifically we select all the samples in the datasets table of the project main page, click the “*Query Results*” link of the project navigation bar, and select the “*DE Genes*” panel.

We set “TREG_activated” as “*Reference Sample*” and “Significantly more frequent in Reference” as “*Gene expression*” pattern. Moreover we choose “0.05” as “*pValue*” threshold and “1,000” as “*Minimum Reads number*” and we select “Y” in “*Reads mapping within the gene boundaries but not overlapping with known exons should be considered?*”

By submitting the query we obtain a table reporting 1,148 genes differentially expressed in the reference sample. If we order the table by descending fold change (fc) value and we consider the top five listed genes, we find that three (CTLA4, IL2RA, IKZF2) out of them have already been reported by Birzele et al. as Treg-specific markers.

4 Notes

1. We downloaded RNA-Seq datasets in fastq format from ENA at EBI since NGS-Trex does not deal with SRA files.
2. Together with nucleotide sequences, NGS platforms typically produce a per-base quality score which provides the probability that a given base is called incorrectly by the sequencer. The quality scores range from 0 to 40, where higher numbers report more confident base calls.
3. The probability of a gene X to be differentially expressed within the reference dataset A compared to dataset B is computed by the cumulative hypergeometric distribution as follows:

$$p\text{Value} = PN, n; M, m = k = \frac{m}{n} \cdot \frac{n}{N} - \frac{k}{M} \cdot \frac{N}{M}$$

Where N is the total number of reads supporting genes within the two datasets, n the total number of reads supporting the gene X within the two dataset, M the number of reads supporting genes within the dataset B , and m the number of reads supporting the gene X within the dataset B . Overrepresentation is defined to be significant when $p\text{Value}$ is less than 0.01 or than 0.05. The Benjamini–Hochberg procedure is used to compute adjusted $p\text{Values}$.

References

1. Wang RL, Biales A, Bencic D, Lattier D, Kostich M, Villeneuve D, Ankley GT, Lazorchak J, Toth G (2008) DNA microarray application in eco-toxicology: experimental design, microarray scanning and factors affecting transcriptional profiles in a small fish species. *Environ Toxicol Chem* 27:652–663
2. Wang L, Feng Z, Wang X, Zhang X (2010) DEGseq: an R package for identifying differentially expressed genes from RNA-Seq data. *Bioinformatics* 26:136–138
3. Mutz KO, Heilkenbrinker A, Lönné M, Walter JG, Stahl F (2013) Transcriptome analysis using next-generation sequencing. *Curr Opin Biotechnol* 24(1):22–30
4. Goecks J, Nekrutenko A, Taylor J, Team G (2010) Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome Biol* 11(8):R86
5. Halbritter F, Vaidya HJ, Tomlinson SR (2011) GeneProf: analysis of highthroughput sequencing experiments. *Nat Methods* 9:7–8
6. Boria I, Boatti L, Pesole G, Mignone F (2013) NGS-Trex: Next Generation Sequencing Transcriptome profile explorer. *BMC Bioinformatics* 14(Suppl 7):S10
7. Birzele F, Fauti T, Stahl H, Lenter MC, Simon E, Knebel D, Weith A, Hildebrandt T, Mennerich D (2011) Next-generation insights into regulatory T cells: expression profiling and FoxP3 occupancy in Human. *Nucleic Acids Res* 39(18):7946–7960
8. Costa V, Angelini C, De Feis I, Ciccodicola A (2010) Uncovering the complexity of transcriptomes with RNA-Seq. *J Biomed Biotechnol* 2010:1–20

Chapter 16

e-DNA Meta-Barcoding: From NGS Raw Data to Taxonomic Profiling

Fosso Bruno, Marzano Marinella, and Monica Santamaria

Abstract

In recent years, thanks to the essential support provided by the Next-Generation Sequencing (NGS) technologies, Metagenomics is enabling the direct access to the taxonomic and functional composition of mixed microbial communities living in any environmental niche, without the prerequisite to isolate or culture the single organisms. This approach has already been successfully applied for the analysis of many habitats, such as water or soil natural environments, also characterized by extreme physical and chemical conditions, food supply chains, and animal organisms, including humans. A shotgun sequencing approach can lead to investigate both organisms and genes diversity. Anyway, if the purpose is limited to explore the taxonomic complexity, an amplicon-based approach, based on PCR-targeted sequencing of selected genetic species markers, commonly named “meta-barcodes”, is desirable. Among the genomic regions most widely used for the discrimination of bacterial organisms, in some cases up to the species level, some hypervariable domains of the gene coding for the 16S rRNA occupy a prominent place.

The amplification of a certain meta-barcode from a microbial community through the use of PCR primers able to work in the entire considered taxonomic group is the first task after the extraction of the total DNA. Generally, this step is followed by the high-throughput sequencing of the resulting amplicons libraries by means of a selected NGS platform. Finally, the interpretation of the huge amount of produced data requires appropriate bioinformatics tools and know-how in addition to efficient computational resources.

Here a computational methodology suitable for the taxonomic characterization of 454 meta-barcode sequences is described in detail. In particular, a dataset covering the V1–V3 region belonging to the bacterial 16S rRNA coding gene and produced in the Human Microbiome Project (HMP) from a palatine tonsils sample is analyzed. The proposed exercise includes the basic steps to manage raw sequencing data, remove amplification and pyrosequencing errors, and finally map sequences on the taxonomy.

Key words Metagenomics, e-DNA, Meta-barcoding, Microbiome, Taxonomy, Pyrosequencing, 16S rRNA

1 Introduction

Metagenomics is today one of the most advanced and fast growing research areas. It has opened an unprecedented comprehensive window on the microbial world, which can now be revealed also in all those parts which remained obscure due to the inability to

cultivate in the laboratory about 99 % of environmental species. Metagenomics bypasses these limits via the direct extraction and investigation of the total genetic material from any environmental niche without the preliminary need to isolate the individual organisms.

The recent amazing advances in next generation sequencing (NGS) technologies have strongly supported the development of this methodological approach by allowing the analysis of increasingly large and complex microbial communities through the production of massive amounts of sequence data in a short time and at lower cost. To date the NGS-based metagenomics has been applied for multiple purposes, such as inferring the health status of environments through the analysis of their microbial biodiversity [1], studying the human microbiome in physiological and pathological conditions [2], analyzing the human nutrition, following the food's path, from production to digestion, and its impact on human physiology [3], and finally discovering novel biomolecules and microorganisms that could be fruitfully introduced in industrial biotechnological processes [4, 5].

Even if in the last years Illumina and, to a lesser extent, SOLID NGS platforms are gaining increasing popularity in metagenomic projects due to their high level of throughput, Roche 454 is still the most competitive platform thanks to the length of its reads [6] which, in the latest version GS FLX Titanium XL +, grow up to ~1,000 bp. In this regard it should be noted that the length of the produced sequences has a significant influence on the accuracy of their next annotation, in particular in situations where reliable reference sequences belonging to the same species residing in the sample are not available. In this case 454 longer reads allow obtaining better results mainly based on the comparison with orthologous genomes. On the other hand the higher output per run of Illumina or SOLID could provide enough coverage to analyze more complex samples and better detect the rare variants.

A common shortcoming in the protocols of all the above-mentioned NGS technologies is represented by the errors, introduced during the amplification step, which could lead to an overestimation of biodiversity by creating false variant sequences. Also a certain intrinsic degree of inaccuracy of the sequencing chemistry contributes to this bias. Recently, several studies have tried to assess the sequencing errors and artifacts produced by the different NGS platforms in metagenomic studies and to compare their overall ability to describe the original microbial community. For example, the main bias associated with Roche 454 is due to the generation of errors in homopolymers (three or more identical consecutive bases). The Illumina protocol seems not to have the same limitation although even this platform is affected by several systematic errors in base calling. Anyway, Luo et al. [7] have shown that the Roche 454 FLX Titanium and the Illumina

Genome Analyzer (GA) II, despite their obvious differences in read length and sequencing protocols, provide a comparable overview of the biodiversity of the analyzed sample.

Organisms and gene functions of the microbiome living in a specific environment can be both addressed by a shotgun sequencing of its entire metagenome. Conversely, if the purpose of the analysis is limited to investigate the taxonomic complexity of the microbial community, an environmental DNA (e-DNA) meta-barcoding [8] approach could constitute an effective and less expensive solution. It involves the culture-independent PCR-targeted sequencing of a selected genetic taxonomic marker (meta-barcode) that should be amplified by means of universal primers and cycles conditions in virtually all the species belonging to the explored taxonomic range.

An optimal meta-barcode should possess three major features: (1) be ubiquitous in the group of organisms of interest (e.g., Bacteria, Fungi), (2) its sequence structure should be marked by domains with very different degrees of variability (hypervariable regions, able in discriminating at lower taxonomic ranks, flanked by primers-annealing very conserved regions) and finally, (3) its size should be compatible with the sequences produced by the most recent versions of NGS platforms.

According to these requirements, selected regions of the internal transcribed spacer (ITS) of the ribosomal RNA (rRNA) gene cluster and of the 16S rRNA gene are commonly used to identify fungal and bacterial taxa, respectively [9–12]. Depending on the selected region and on its length, the analysis will allow to reach a certain depth in the taxonomic classification. For instance, the analysis of partial 16S rRNA gene sequences provides a direct and relatively inexpensive access to bacterial communities biodiversity achieving, in the case of the recent versions of 454 technology, a resolution down to the genus level [13]. 16S rRNA surveys provide information about relative taxa abundance too [14]. A limitation of the 454 pyrosequencing is that the amount of obtained partial 16S rRNA gene sequences might be inadequate to represent the entire biodiversity of the environmental sample. In particular, the rare species may be difficult to detect. Illumina overcomes this problem thanks to its higher throughput but at the same time the shorter length of its reads can adversely affect the accuracy of the assignment to the appropriate taxonomic classes [13].

Taking into account the current progress of biomolecular technologies, the potential power of e-DNA meta-barcoding is limited basically by two aspects. The first one is its dependency on PCR which, unfortunately, can intrinsically introduce errors as substitutions and/or insertions/deletions and bias due to the different propensity of universal primers to anneal to the same genomic region in different taxa. In addition, the rare species could be overwhelmed by the most abundant ones, which are favored

during the amplification step. The second aspect concerns some issues regarding the bioinformatic analysis of the large amount of amplicons sequences produced through NGS and the computational challenges it entails.

Currently, the bioinformatic methods used to assign the metagenomic sequences to their taxonomic classes [15] adopt essentially three approaches:

- In *similarity-based methods*, the taxonomic characterization of a metagenomic dataset is commonly based on its comparison to curated collection of sequences with known function and origin [10]. These methods offer high accuracy, even for short sequences, but they suffer three general shortcomings: (1) the time needed for the similarity analysis grows with the size of the chosen reference database, (2) the absence of reference sequences for unknown species may affect the assignment accuracy [16], and (3) the availability of comprehensive and well annotated reference database requires considerable investments [8]. Among the largest reference resources currently available for rRNA gene-based identification of microorganisms there are Greengenes [17], Ribosomal Database Project II (RDP II) [18], and SILVA [19]. ITSOneDB has been recently developed as reference resource for meta-barcoding analysis based on ITS1, a specific intergenic region of rRNA gene cluster in Fungi [10].
- In *composition-based methods*, several genome features, such as GC content, codon-usage, or oligonucleotides frequencies, are used to assign the metagenomic sequences to taxonomically annotated reference genomes. The reference data are modelled using different methods, such as the interpolated Markov models (IMMs), the naïve Bayesian classifiers, and the k -means/ k -nearest-neighbor algorithms. The principal shortcoming of these approaches is the high requirement of computational power and time for the models building step. Despite this phase, the metagenomic classification is generally faster than that performed by means of similarity-based methods. Finally, composition-based methods offer a good classification accuracy, even for short sequences, that can be further improved using some statistical methods (e.g., bootstrap or cross-validation).
- In *phylogenetic-based methods*, the metagenomic sequences are placed onto a reference phylogenetic tree according to a model of evolution such as that implemented in maximum likelihood (ML), Bayesian or neighbor-joining (NJ) methods. Nowadays, although these methods promise a high accuracy, the computational demands for their application is often too high.
- Once meta-barcodes have been sequenced by means of NGS technologies, before applying one of the methods listed above

to assign the reads to their original taxonomic ranks, many other different tools should be used for filtering and denoising the raw data [13, 14, 16]. These preliminary bioinformatic steps are essential in order to obtain a consistent taxonomic inference.

Here, a simple and comprehensive protocol for the analysis of NGS reads from 16S rRNA gene is described in all its steps from the raw data to the taxonomic classifications of bacterial organisms. The procedure refers, in particular, to the analysis of data produced by the 454 platform but it can be applied, except for the denoising step, to Illumina sequences too.

2 Materials

2.1 HMP (Human Microbiome Project) Data Set

The proposed protocol is here executed by starting from a publicly available dataset of bacterial 16S rRNA gene sequences stored in the NCBI Sequence Read Archive (SRA) collection. In particular, the considered dataset has been produced by the Human Microbiome Project (HMP) in which several body sites, including gastrointestinal and female urogenital tracts, oral cavity, nasal and pharyngeal tracts, and skin, were studied by pyrosequencing the V1–V3 and V3–V5 hypervariable regions of 16S rRNA gene. The SRR331235 datarun covering the V1–V3 region obtained from palatine tonsils sample of a male subject was chosen to be submitted to the exercise described below. The forward primer ATTACCG CGGCTGCTGG was used for the unidirectional pyrosequencing as reported in SRA collection experiment page at the link <http://www.ncbi.nlm.nih.gov/sra?term=SRR331235>. Moreover a nine-nucleotide tag (TCGAGGAAC) was inserted upstream of the V1–V3 region [20] in order to discriminate the reads belonging to the different samples concurrently sequenced in the same run.

The SRR331235 datarun was downloaded as a binary sra file of about 5.3 Mb. The dataset consists of 4,504 reads with an average length of 535.80 nt.

Since the data provided by the 454 technology are in the binary Standard Flowgram Format (SFF), the downloaded sra file was converted in the sff by using the *sratoolkit* (see Note 1). The aim of this step was to reproduce in the exercise the case in which the researcher has to analyze directly the data produced by this type of platform.

The information included in the sff file can be visualized using the *Roche SFFtools* by converting it in a textual flat file, as extensively described in the next section. In particular, two principal sections can be found in this file: a common header section including some general information about the sequencing output (such as the number of reads stored in the file and the number of nucleotide flows performed during the sequencing) [21] and a read-specific

section for each of the generated sequences. In the read-specific section the following information are stored:

- The *Flowgramm* field provides information about the light signal measured for each nucleotide flow (e.g., a 0.0 signal is measured if there is no nucleotide incorporation while a 1.0 signal indicates just a single nucleotide addition) [21];
- The *Flow Index* field contains the progressive index associated to each nucleotide flow providing base addition in the called sequence;
- The *Bases* field contains the nucleotide sequence;
- The *Quality Scores* field contains information about the accuracy of the sequencing. The quality score is estimated for each of the bases in the sequence and its reported values are in Phred-score format and represent the $-10\log_{10}$ of the probability that there is a base-call error (e.g., a Phred-score of 30 means that there is a probability of 0.001 that the called base is wrong) [22].

During the protocol computation two other common biological formats are produced:

- The *fasta file format* is a common standard in computational biology consisting in a description line that starts with a “>” character followed by one or more lines in which the sequence is reported. An example of fasta format is shown below:

```
>GMF8DCB01AT97I
ATTACCGCGGCTGCTGGCACGTAGTTAGCCGTGGCTTCTGGT
TAGATAACCGTCAAGGCACACAGGGGATAGG
```

- The *fastq file format* stores both the sequence and the base-call quality information in a single file. Each sequence is described by four lines: a description line starting with “@” digit reporting the sequence accession number, a single line containing the sequence, a third line always starting with a “+” character and the ASCII encoded quality-score line. Below an example is shown:

```
@GMF8DCB01AT97I
ATTACCGCGGCTGCTGGCACGTAGTTAGCCGTGGCTTCTGGT
TAGATAACCGTCAAGGCACACAGGGGATAGG
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIHHHIIII
IIIIIIIIIIIIIIIIIIIC5555=GF@@
```

2.2 Computational Hardware

The management and the processing of large metagenomic datasets require powerful storage and computational capabilities therefore Linux and Unix-based machines are recommended.

The storage of data from a typical 454 GS FLX Titanium run usually takes up to 2 GB, and then it is necessary to consider additional free space for the analysis. Just think that by performing the proposed protocol starting from a flat sff file of about 14 MB, nearly 150 MB of data are produced just during the first step of the analysis, the denoising phase.

With regard to the computational capabilities it is reasonable to carry out all the analysis on a multi-processor/core machine with at least 20 GB of RAM.

2.3 Computational Software

The protocol described below includes the use of the following software:

1. SFFtools, a part of Roche Data Analysis package, is not freely available but 454 users can request it from <http://454.com/products/analysis-software/index.asp>. In order to use SFFtools to manage the sff file, the installation of the whole Roche Data Analysis package is required on a Linux operating system by using root privileges.
2. AmpliconNoise, available at <http://code.google.com/p/ampliconnoise/>, which is a collection of programs used to reduce the base errors intrinsically introduced during the 454 amplicons sequencing procedure [23]. It is supported on Linux and Mac OS X platforms but a particular attention is required during the installation and the configuration of the package because it is required the setting of some environment variables (as fully described in the documentation file provided with the installation package). Moreover, AmpliconNoise is designed to work on multi-processors/cores system using the Message Passing Interface (MPI) libraries that must be previously installed and configured. The MPI libraries allow parallelising the process on multi-processor/core machines for the High Performance Computing (HPC).
3. FastQC, a Java-based tool, which is available at <http://www.bioinformatics.babraham.ac.uk/projects/fastqc/> and used for a simple and quick quality check of sequences outputted directly by the sequencer (also referred as raw data).
4. RDP classifier is a naïve Bayesian classifier for fast taxonomic assignment of sequences and confidence estimation (Assignment procedures and confidence values estimation are amply described in Subheading 3.3). It can be downloaded from <http://rdp-classifier.sourceforge.net>.
5. Python, an object-oriented programming language available at <http://www.python.org>. It is preinstalled in several Linux-based Operating Systems (e.g., Ubuntu) and in Mac OS X but not in Windows Operating Systems. During the proposed exercise two Python modules are used: BioPython (http://biopython.org/wiki/Main_Page) and Numpy (<http://www.numpy.org>).

In order to correctly follow the protocol, the previous listed modules must be installed and operating properly. It is important to note that the Python modules installation procedures may change depending on the operating system and it is recommended to follow the installation instruction available in the aforementioned module Web page.

3 Methods

Conceptually we can define three main steps, listed below, in the taxonomic assignment of meta-barcodes, previously amplified from the microbial community and sequenced by 454 technology.

- Raw data management and evaluation: the 454 raw data are converted in file format suitable for the subsequent analysis;
- Denoising: the AmpliconNoise suite is applied to remove the errors, introduced in the sequences by both amplification and pyrosequencing processes, which could leads to incorrect inference of microbiota diversity;
- Assignment of cleaned sequences to their taxonomic group.

The entire protocol can be performed on a bash terminal. The operations performed directly into the bash environment are preceded by the “\$” prompt while those executed into the Python shell are specified by the “>>>” prompt.

3.1 Raw Data Management and Evaluation

The first step of the protocol consists in the conversion of the binary sff file produced by the sequencing platform 454 into two different file formats, a textual flat file and a fastq file, and in the statistical evaluation of the raw data.

The conversion of the sff file into a flat textual file is performed by the Roche SFFtools software. The produced file will be submitted to the denoising step. Some alternative methods that are shown in **Note 2** are available to manage the sff file.

The command used in the bash terminal to achieve this conversion is the following:

```
$ sffinfo SRR331235.sff > SRR331235.sff.txt
```

The sff is also converted into a fastq file suitable to perform the raw data quality check by means of FastQC. This operation is carried out in the Python shell by using the BioPython module and consists of two steps:

- (a) the invocation of the BioPython SeqIO command class, which is useful to manage biological data files, by using the following commands:

```
$ Python
```

And then:

```
>>>from Bio import SeqIO
```

- (b) the sff file conversion in a fastq file by using the command SeqIO.convert that requires four instructions: the file to be converted, its format, the output file name and the final file format. The command to perform this conversion is the following:

```
>>>SeqIO.convert("SRR331235.sff","fastq","SRR331235.fastq","fastq")
```

Afterwards, the produced fastq file is processed by means of FastQC, by using the following command:

```
$ fastqc SRR331235.fastq
```

FastQC performs several statistical evaluations on the sequencing data and builds an html report file divided in 12 subsections. The descriptions of the latter are included in several different html files (you can open them using any Internet browser) contained in the “help” folder, available within the FastQC package.

For example, in one of the 12 subsections obtained in this exercise, the “Basic Statistics,” the dataset under investigation results to consist of 4,504 sequences ranging from 86 to 1,049 nucleotides and with a GC content of 52 %. Furthermore, in the “Per base sequence quality” subsection it is shown a chart of the quality values ranges across all bases at each sequence position in the fastq file related to the analyzed dataset (Fig. 1). In Fig. 1 it is possible to observe a reduction of the quality at the 3' end. This behavior is due to the intrinsic limits of the sequencing technology which tends to lose sensitivity in the last steps of the process.

3.2 Denoising

The second step of the entire protocol, called denoising, consists in the application of AmpliconNoise. AmpliconNoise is a suite of programs designed to reduce the errors introduced in the sequences by both amplification and pyrosequencing before mapping them on the taxonomy.

In order to exhaustively explain all the procedures of this phase, the manual procedure is shown but it is possible to perform them by means of bash scripts that are supplied in the installation package. Moreover, AmpliconNoise can be alternatively applied within the Mothur suite [24] (*see Note 3*).

The processing starts with a filtering step applied to remove low quality reads using the Perl script FlowMinMax.

In the bash terminal, type:

```
$ FlowsMinMax.pl TCGAGGAACATTACCGCGGCTGCTGG
test_book < SRR331235.sff.txt
```

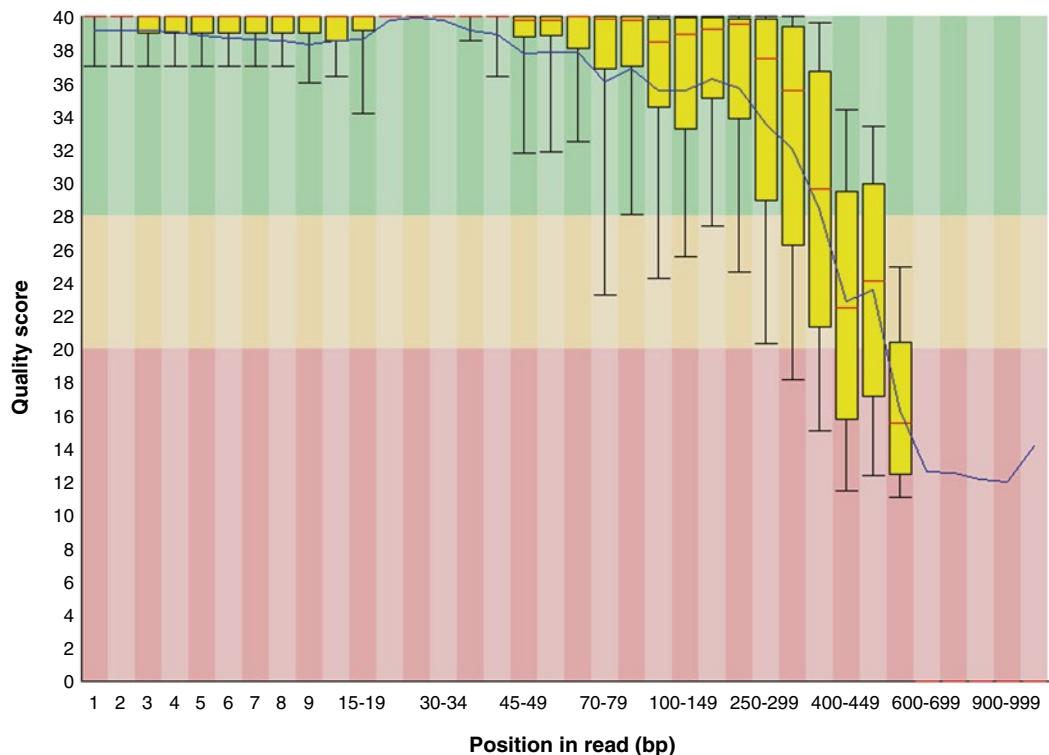


Fig. 1 The per base sequence quality chart produced by FastQC for the dataset analyzed in this exercise (data run SRR331235 produced by the Human Microbiome Project (HMP))

FlowMinMax requires two arguments, the tag (TCGAGGAAC), eventually added in order to discriminate the sequences belonging to different samples processed in the same sequencing run, followed by the primer(s) sequence (s) (ATTACCGCGGCTGCTGG), used for the sequencing of the selected meta-barcode, and an output argument (test_book) to process the flat textual sff file derived from the previous data conversion step. FlowMinMax works on both sequences and flowgrams that generated them. As regards the analysis of sequences, an exact match is searched between each of them and both the primer and the specific tag. If the match is not found the read is discarded. As regards the flowgrams, FlowMinMax verifies if there are ambiguous flow signals or if no signals are registered for all the four nucleotide in the region corresponding to the first 360 flows of pyrosequencing where reliability should be higher [20]. If one of these two conditions is verified the corresponding read is discarded. Moreover all flowgrams are truncated at 720th flow (if the 454 GS FLX Titanium sequencing technology has been adopted) due to an evident decrease in the quality of the sequences after this point [25].

After the application of the filters described above, FlowMinMax provides two output files: test_book.dat in which the

retained flowgrams are collected and test_book.fasta containing the corresponding sequences.

In this exercise 3,229 flowgrams and the related sequences pass the filtering step. In particular, in the test_book.dat file the following information are reported: in the first line the number of flowgrams which passed the filter followed by the number of base flows retained after the cutoff procedure (in this case 3,229 flowgrams and 720 flows) are reported. In the subsequent lines the reads accession numbers, their corresponding flowgrams and number of flows are shown.

The second step of AmpliconNoise, called dereplication, consists in removing the redundant sequences in order to get a more slender dataset and reduce the computational power required for the next bioinformatic steps.

The dereplication command to type on the bash terminal is the following:

```
$ FastaUnique -in test_book.fa > test_book_U.fa
```

FastaUnique joins all the identical sequences and returns a file test_book_U.fa in which the fasta description line of each “representative” sequence is followed by a number indicating the amount of sequences identical to it. An example of line obtained in this exercise, is:

```
> GMF8DCB01BB75Z_2
```

where “2” represent the number of identical sequences including the “representative” one.

In the case of the dataset used for the exercise, FastaUnique identifies 1,636 unique sequences.

Moreover, the information about the dereplication is stored in a map file (test_book.map) in which for each line is reported the accession number of the “representative” sequence followed by the accession number of all the reads identical to it.

Example: In the exercise case, the dereplication information related to the unique sequence GMF8DCB01BB75Z_2 is the following:

```
330,154,GMF8DCB01BB75Z,2:330,2965:GMF8DCB01BB75Z,GMF8DCB01B09TA
```

In particular, several data are stored in the shown line:

- 330 is the original ID of the representative sequences. Simply it means that in the test_book.fa file GMF8DCB01BB75Z was the 330th sequence;
- 154 is the new ID associated to the representative sequence and it means that in the test_book_U.fa file produced by FastaUnique, GMF8DCB01BB75Z is the 154th sequence;
- 2 is the number of sequences that are represented by GMF8DCB01BB75Z;

- 330 and 2965 are the original ID of the two sequences represented by GMF8DCB01BB75Z;
- GMF8DCB01BB75Z and GMF8DCB01B09TA are the accession number of the joined sequences.

The unique sequences reported in the fasta file (test_book_U.fa) produced by FastaUnique are then clustered according to their mutual similarity typing three different commands in the bash terminal. This process lead to reduce the computational weight in the subsequent removal of pyrosequencing errors.

The three commands to be sequentially entered are:

```
$ mpirun -np 8 NDist -i -in test_book_U.fa > test_book_U_I.ndist
$ FCluster -a -w -in test_book_U_I.ndist -out test_book_U_I > test_book_U_I.fcout
$ SplitClusterEven -din test_book.dat -min test_book.map -tin test_book_U_I.tree -s 2500 -m 250 > test_book_U_I.stats
```

Ndist requires two arguments, the fasta input file “-in” and the “-i” flag to print the output in a format suitable for the FCluster running. The exact Needleman–Wunsch algorithm is implemented in NDist to perform pairwise alignments with a fixed gap penalty score of 1.5. Then, the program produces a distance file (test_book_U_I.ndist) by applying the QuikDist algorithm [26] which assigns to both the mismatch and the gap a penalty score of 1. The genetic distance calculated on the basis of these computed differences for each pair of sequences is normalized both for the alignment length of each pair of sequences and the total number of comparisons performed in the dataset.

FCluster performs a hierarchical clustering starting by the distance file produced by NDist (-in). As default, it performs a complete linkage clustering. This is a hierarchical clustering method that sets the distance between two clusters as the maximum distance observed between the cluster members.

In the exercise case, sequences that are the result of the dereplication steps are considered and, for this reason, an average linkage accounting for the amount of sequences represented by the unique one is used, by typing the arguments, “-a -w”. In the average linkage clustering procedure the distance between two clusters corresponds to the average distance between the cluster members.

Three output are produced by FCluster:

- a file (test_book_U_I.list) in which the genetic distances which guide the clustering are listed. In particular, for each value of distance the corresponding number N of formed clusters is reported and for each cluster the number of included sequences is indicated;

- a file (test_book_U_I.otu) in which the clusters sizes, measured at a distance separation of 0.01(e.g., 0.00, 0.01, 0.02 and so on) by the QuickDist algorithm implemented in Ndist, are stored;
- a file (test_book_U_I.tree) where clusters are mapped on a hierarchical tree in newick format.

SplitClusterEven splits the flowgrams (-din), stored in the test_book.dat file by using as reference the map file test_book.map (-min), both produced by FlowMinMax, and by following the hierarchical tree (-tin) produced by FCluster. The arguments “-s” and “-m” control the cluster maximum and minimum dimension. The data for each cluster are stored in different folder labelled as C000,...,C00N+. In the exercise case just one cluster is generated (C000) using default “-s” and “-m” parameters.

In order to reduce the computational time, in this exercise the MPI libraries (mpirun) are used and FCluster is applied on eight nodes (-np argument) (*see Note 4*).

The clustered data obtained in the previous steps are then processed in order to decrease the pyrosequencing errors noise.

The commands to type are:

```
$ mpirun -np 8 PyroDist -in C000/C000.dat -out
C000/C000 > C000/C000(fout
$ FCluster -in C000.fdist -out C000_X > C000(fout
$ mpirun -np 8 PyroNoiseM -din C000.dat -out
C000_s60_c01 -lin C000_X.list -s 60.0 -c 0.01 >
C000_s60_c01.pout
```

In order to reduce the pyrosequencing noise, AmpliconNoise tries to estimate the “real sequences” and their relatives abundance directly from the raw data. For this reason, it starts with an initial clustering, performed by PyroDist and FCluster. In particular, PyroDist computes a distance matrix, working directly on flowgrams by taking in input the dat file (-din), in order to produce data suitable for FCluster computation (C000(fout). FCluster performs an average linkage clustering on the distance matrix produced by PyroDist (see before for the software details).

Then, the procedure continues by applying the Expectation Maximization algorithm (EM) [27] implemented in PyroNoiseM. The EM algorithm is based on the idea of maximizing the probability that a flowgram in a cluster was generated from the estimated real sequence for that cluster. Accordingly, the algorithm moves the flowgrams among the clusters and re-computes the probability. The EM computation ends when no more shifts are allowed because the estimated probability is already maximized. In the PyroNoiseM command-line the flowgrams to be processed are indicated using the “-din” flag while the clustering data produced by FCluster on PyroDist data are supplied with the argument “-lin”. Two parameters guide the clustering procedure:

“-c” is the cutoff for the initial clustering step and “-s” rules the cluster size (60.0 is a good optimization between noise removal and computational requirements). At the end of the PyroNoiseM processing, a number of files labelled with the “-out” parameter (C000_s60_c01) are produced:

- a fasta file where the denoised sequences associated with their accession numbers, called as *output_label_index_weight*, are reported. *Weight* indicates the number of raw sequences associated to each denoised one and *index* is just a number which identifies univocally the specific denoised sequences (C000_s60_c01_cd.fa);
- a quality file for the denoised sequences (C000_s60_c01_cd.qual). It is a fasta-like format file where, for each denoised sequence, the accession number and the estimated Phred-score for each base is annotated;
- a mapping file in which the accession numbers of the raw reads are associated with their corresponding denoised sequences. (C000_s60_c01_cd.mapping);
- a directory (C000_s60_c01) containing a fasta file for each denoised sequence. In each file the unique sequences, produced by the Perl script FastaUnique, and the corresponding reads are listed.

The sequence and mapping data produced by PyroNoiseM are then copied and renamed in the test_book_pyro_denoised.fa and test_book_pyro_denoised.mapping files, respectively, using the following commands:

```
$ cp C000/C000_s60_c01_cd.fa > test_book_pyro_denoised.fa
$ cp C000/C000_s60_c01.mapping > test_book_pyro_denoised.mapping
```

Subsequently, the denoised sequences stored in the test_book_pyro_denoised.fa file are truncated at 400 nt (see the AmpliconNoise documentation) by typing:

```
$ Truncate.pl 400 < test_book_pyro_denoised.fa > test_book_pyro_denoised_T400.fa
```

The sequences contained in the produced test_book_pyro_denoised_T400.fa file are processed in order to remove the barcode and the primer sequences before the SeqNoise application by means of *sed*, a stream editor which works using regular expressions.

In the exercise case, it is possible to substitute (s) the barcode and the primer at the 5' end (^) of each sequence with an empty string by typing:

```
$ sed 's/^TCGAGGAACATTACCGCGGCTGCTGG//' test_book_pyro_denoised_T400.fa > test_book_pyro_denoised_T400_P.fa
```

The `test_book_pyro_denoised_T400_P.fa` file is produced as output.

The last three steps of the denoising procedure are applied to remove the amplification errors noise. The corresponding commands to enter in the terminal are:

```
$ mpirun -np 8 SeqDist -in test_book_pyro_denoised_T400_P.fa > test_book_pyro_denoised_T400_P.seqdist
$ FCluster -in test_book_pyro_denoised_T400_P.seqdist -out test_book_pyro_denoised_T400_P > test_book_pyro_denoised_T400_P.fcout
$ mpirun -np 8 SeqNoise -in test_book_pyro_denoised_T400_P.fa -din test_book_pyro_denoised_T400_P.seqdist -lin test_book_pyro_denoised_T400_P.list -min test_book_pyro_denoised.mapping -out test_book_denoised_T400_s25_c08 -s 25.0 -c 0.08 > test_book_pyro_denoised_T400_s30_c08_snout
```

As discussed before for pyrosequencing noise reduction, the amplification errors are removed by using an initial clustering, performed by SeqDist and FCluster, to infer the “real sequences” and then by applying an EM algorithm implemented in SeqNoise.

SeqDist works directly on the sequences, outputted by the PyroNoiseM computation and then truncated using the Perl script Truncate.pl, in order to generate a distance matrix (`test_book_pyro_denoised.seqdist`) in a format suitable for FCluster. It works using an implementation of Needleman–Wunsch algorithm to perform pairwise alignment starting from a fasta file containing the pyro-denoised sequences (-in). For the distance matrix computation a different weight is given to homopolymer (4.0) and single (15.0) gaps.

FCluster works as already described and, in particular, it performs a hierarchical clustering based on the distance matrix produced by SeqDist.

SeqNoise removes the amplification noise using an EM algorithm clustering procedure. It takes in input the distance matrix produced by SeqDist (-din), a weighted fasta file (-in) with the associated mapping file (-min) and the initial hierarchical clustering data produced by FCluster (-lin). In particular, the weighted fasta file indicated using the “-in” flag is the file produced by PyroNoiseM and then processed by the Perl script Truncate.pl. In this file the description line of each denoised sequence is formed as *output_label_index_weight* where *weight* indicates the number of raw sequences clustered in each denoised one. The raw reads included in the denoised one are listed in the mapped file indicated by the “-min” flag and produced by PyroNoiseM. Two parameters guide the clustering procedure: “-c” (0.08) for the initial cutoff and “-s”

(30.0) for the clustering size (for more information refers to the AmpliconNoise documentation). The “-out” flag (test_book_denoised_T400_s25_c08) is used to label the output data produced by SeqNoise which are contained in the following files:

- a fasta file where the denoised sequence associated with their accession number formed as *output_label_index_weight* are reported. *Weight* indicates the number of raw sequences clustered in each denoised one and *index* is just a number which identifies univocally the specific cluster (test_book_denoised_T400_s25_c08_cd.fa);
- a mapping file in which the accession numbers of the sequences produced by PyroNoiseM computation are associated with their corresponding denoised sequences obtained by SeqNoise (test_book_denoised_T400_s25_c08.mapping);
- a directory (test_book_denoised_T400_s25_c08) containing a fasta file for each sequence derived from the SeqNoise denoising procedure. In each file the unique sequences and the corresponding reads, all used to generate the denoised sequences, are listed;
- if the –min parameter is added to the command-line, an optional mapping file (test_book_denoised_T400_s25_c08_cd.mapping), in which the accession numbers of the raw reads are associated with the corresponding denoised sequence derived from the full denoising procedure, is created.

In the exercise described here, after the complete denoising procedure, AmpliconNoise identifies 399 real sequences, that are theoretically unaffected by sequencing and amplification errors, starting from the 3,229 reads that passes the initial filtering step.

3.3 Taxonomic Classification

The third step of the protocol consists in the taxonomic classification of the denoised sequences by using the RDP classifier.

RDP classifier [28] is a naïve Bayesian classifier which works rapidly on short sequences and without the need to align them to a taxonomically annotated reference database. It is developed by the same team which maintains RPD II [29] and it is able to assign sequences from kingdom to genus levels.

As other textual Bayesian classifiers, it works in a space of features consisting in all the 8 nt long sub-string that could be found in a sequence whatever their position. In this system $W = \{w_1, \dots, w_d\}$ is defined as the set of all possible 8 nt long sub-string that can be retrieved in a given sequence. Given a collection of N sequences, $n(w_i)$ represents the number of times that w_i is observed in the set N . The expected probability to observe w_i in considering the entire set N is:

$$P_i = \frac{n(w_i) + 0.5}{N + 1}$$

If a classification at genus level is chosen, the conditional probability of observing a specific genus given the single w_i can be calculated.

Let us consider a training dataset (M) of sequences belonging to the genus G . $m(w_i)$ is the number of sequences in M containing the word w_i . The Conditional probability that a sequence belonging to G contain w_i is:

$$P(w_i | G) = \frac{m(w_i) + P_i}{M + 1}$$

The joint probability $P(S|G)$ that a sequence S (e.g., the 16S rDNA variable region V1–V3), belonging to a specific genus G , contains the entire set of possible words, $V = \{v_1, \dots, v_f\}$, in which it can be splitted can be calculated as follows:

$$P(S | G) = \prod P(v_i | G)$$

If a sequence S with unknown taxonomic origin is considered, the likelihood that S belongs to a genus G can be estimated by using the Bayesian theorem as follows:

$$P(G | S) = \frac{P(S | G)P(G)}{P(S)}$$

Assuming that all genera are equally probable the terms $P(G)$ and $P(S)$ can be ignored. A sequence is assigned to the genera with the highest probability score but the actual numerical probability estimate is ignored.

For each query sequence, the collection of all eight-character sub-sequences (words) in the query is first calculated and then 100 bootstrap trials are performed. So, for each bootstrap trial, a subset of one-eighth of the words is randomly chosen (with replacement) and the words in this subset are used to calculate the joint probability. The number of times a genus is selected out of 100 bootstrap trials is used as an estimate of confidence in the assignment to that genus. By default a confidence threshold of 0.8 is used to assign a sequence to a genus [28].

Some features of the dataset to be analyzed must be verified prior to apply RDP classifier, in order to optimize the choice of the confidence value threshold. Indeed, it has been demonstrated that a threshold value of 0.5 can be used without loss of assignment accuracy for short sequences (<250 nt) [30]. Therefore, it is convenient to extract the required information about the denoised sequences by using Python shell.

Below the commands to be typed are reported. The sentences starting with “#” are comments used to describe the operation and they are not processed by Python.

```

$ Python
#First let us importing the class commands needed
to make the analysis
>>>from Bio import SeqIO
>>>import numpy
>>>import fpformat
#an empty list is created to store the infor-
mation about the sequence lengths
>>>size = []
# the length of each sequence is stored in the
sizes list by using the multifasta parser SeqIO.
parse
>>>for sequence in SeqIO.parse("test_book_
denoised_T400_s25_c08_cd.fa","fasta"):
    size.append(len(sequence))
#the average length is estimated using numpy.
mean command and the computed value is rounded
#to 2 decimal positions
>>>mean = fpformat.fix(numpy.mean(size),2)
#the standard deviation is estimated using
numpy.std command and the computed value is
rounded
#to 2 decimal positions
>>>sd = fpformat.fix(numpy.std(size),2)
#estimation of the minimum length
>>>min_length = min(size)
#estimation of the maximum length
>>>max_length = max(size)
#simply the estimated information are printed
>>>print "The denoised sequences are",len(size)
The denoised sequences are 399
>>>print "the average length is",mean," and the
standard deviation is ",sd
the average length is 345.62 and the standard
deviation is 43.60
>>>print "the shortest sequence is long",min_
length,"while the longest ",max_length
the shortest sequence is long 205 while the lon-
gest 374

```

The previous operations provide the information that the dataset considered in the exercise contains sequences shorter than 250 nt. Accordingly, a confidence threshold values of 0.5 is fixed for the RDP classifier computing by using the “--conf” parameter. The flag “--hier_outfile” will allow to produce the output file containing the assignment count for each annotated taxonomical rank and the flag “--assign_outfile” will specify the output file containing the assignment details for each sequence.

The output file created with the “--hier_outfile” flag is tabular and includes five fields:

- Tax id: the taxonomy ID of the rank to which the sequences are assigned;
- Lineage: the full lineage path from root to the listed taxonomic rank;
- Name: the name of the considered taxonomic rank;
- Rank: the taxonomic class to which the considered rank corresponds (e.g., class, order, family, genus);
- The number of reads assigned to the considered rank.

In summary, the tabular file obtained using the flag “--hier_outfile” provide a summary of the taxonomic assignment obtained with RDP classifier and it can be easily managed by means the most popular spread sheet programs.

The output file, obtained using the flag “--assign_outfile” is tabular and includes the following fields:

- accession number of the analyzed sequences;
- the full lineage path resulting from the taxonomic assignment and the confidence values estimated for each rank in the path.

In the exercise case, since the assigned sequences resulted from several clustering and denoising steps, this file must be parsed in order to associate to each rank the actual number of reads instead their derived denoised sequence:

Following are the bioinformatics commands to perform the latest described operations.

In particular, in order to perform the RDP-classifier, type:

```
-jar $ java -Xmx1g MultiClassifier.jar --ier_outfile=hierarchical_assignment.txt --sign_outfile=reads_classification.txt --onf=0.5 test_book_denoised_T400_s25_c08_cd.fa
```

Then the entire output assignment file (reads_classification.txt) can be parsed and analyzed by means of a Python script, called “extract_data.py” in order to create the submentioned tabular file in which the number of actual reads is associated to each rank. The obtained file is in csv format and it is easily manageable using common spread sheet programs.

First, the “extract_data.py” script must be written in the current working directory. Below the commands included in it are listed:

```
l = open("read_classification.txt")
name2rank = {}
name2reads = {}
for line in l.readlines():
    line = line.strip()
    fields = line.split("\t")
```

```

        acc = fields[0]
        s = acc.split("_")
        reads = int(s[-1])
        stop = len(fields)
        index = 2
        while index < stop:
            node = fields[index]
            rank = fields[index+1]
            conf = float(fields[index+2])
            if conf >= 0.5:
                name2rank.setdefault(rank, set())
                name2rank[rank].add(node)
                name2reads.setdefault(node, 0)
                name2reads[node] += reads
                index += 3
            else:
                index = stop
    #now we can print the data in a csv file
    csv=open("data_summary_results.csv","w")
    for rank in name2rank.keys():
        csv.write(rank+"\n")
        for node in name2rank[rank]:
            csv.write(node+"\t"+str(name2reads[node])+"\n")

```

The script can be applied simply by typing “python extract_data.py” and in few seconds it produces the “data_summary_results.csv” file.

By observing the produced data, *Firmicutes* (1,099 reads, 34.03 %) and *Proteobacteria* (1,083 reads, 33.54 %) result the most represented phyla, while *Bacteroidetes*, *Fusobacteria* and *Actinobacteria* correspond only to 15.89 % (513 reads), 20.13 % (650 reads) and 6.41 % (207 reads), respectively. Really, these data are consistent with those obtained in previous human microbiota surveys [31, 32].

4 Notes

1. The *sratoolkit* is a collection of tools provided for accessing the information compressed in the sra files. The tools package is available at <http://trace.ncbi.nlm.nih.gov/Traces/sra/sra.cgi?view=software>. In order to extract the 454 data stored in the downloaded .sra file, the *sff-dump* has been applied as described below:

```
sff-dump SRR331235.sra -O sff_file_dir/
SRR331235.sff
```

the “-O” option is used to redirect the output in folder (in our case “`sff_file_dir`”).

2. As reported in AmpliconNoise documentation, there are some alternative methods to manage the sff file such as by using the free software Flower [33] or the `process_sff.py` script included in QIIME [34]. Moreover, the Mothur suite provides the command `sffinfo`.
3. In Mothur suite, PyroNoise and SeqNoise are re-implemented in `shhh.flows` and `shhh.seqns` commands, respectively.
4. All the AmpliconNoise analysis is computationally very heavy. For this region, the several steps are performed by using the MPI libraries in order to parallelise the job on 8/10 nodes. By means of these measures, the pyro- and seq-noise run times can be reduced to 30 and 20 min respectively, if the tested dataset is considered.

References

1. Hemme CL, Deng Y, Gentry TJ et al (2010) Metagenomic insights into evolution of a heavy metal-contaminated groundwater microbial community. *ISME J* 4(5):660–672
2. Ottman N, Smidt H, de Vos WM et al (2012) The function of our microbiota: who is out there and what do they do? *Front Cell Infect Microbiol* 2:104
3. Dutton RJ, Turnbaugh PJ (2012) Taking a metagenomic view of human nutrition. *Curr Opin Clin Nutr Metab Care* 15(5):448–454
4. Knight R, Jansson J, Field D et al (2012) Unlocking the potential of metagenomics through replicated experimental design. *Nat Biotechnol* 30(6):513–520
5. Barnard D, Casanueva A, Tuffin M et al (2010) Extremophiles in biofuel synthesis. *Environ Technol* 31(8–9):871–888
6. Shokralla S, Spall JL, Gibson JF et al (2012) Next-generation sequencing technologies for environmental DNA research. *Mol Ecol* 21: 1794–1805
7. Luo C, Tsementzi D, Kyripides N et al (2012) Direct comparisons of Illumina vs. Roche 454 sequencing technologies on the same microbial community DNA sample. *PLoS One* 7:e30087
8. Taberlet P, Coissac E, Pompanon F et al (2012) Towards next-generation biodiversity assessment using DNA metabarcoding. *Mol Ecol* 21(8):2045–2050
9. Blaalid R, Kumar S, Nilsson RH et al (2013) ITS1 versus ITS2 as DNA metabarcodes for fungi. *Mol Ecol Resour* 13(2):218–224
10. Santamaría M, Fosso B, Consiglio A et al (2012) Reference databases for taxonomic assignment in metagenomics. *Brief Bioinform* 13(6):682–695
11. Tringe SG, Hugenholtz P (2008) A renaissance for the pioneering 16S rRNA gene. *Curr Opin Microbiol* 11(5):442–446
12. Nilsson RH, Kristiansson E, Ryberg M et al (2008) Intraspecific ITS variability in the kingdom fungi as expressed in the international sequence databases and its implications for molecular species identification. *Evol Bioinform Online* 4:193–201
13. Teeling H, Glöckner FO (2012) Current opportunities and challenges in microbial metagenome analysis—a bioinformatic perspective. *Brief Bioinform* 13(6):728–742
14. Gilbert JA, Field D, Swift P et al (2010) The taxonomic and functional diversity of microbes at a temperate coastal site: a 'multi-omic' study of seasonal and diel temporal variation. *PLoS One* 5(11):e15545
15. Bazinet al, Cummings MP (2012) A comparative evaluation of sequence classification programs. <http://drum.lib.umd.edu/handle/1903/13346>
16. Simon C, Daniel R (2011) Metagenomic analyses: past and future trends. *Appl Environ Microbiol* 77(4):1153–1161
17. DeSantis TZ, Hugenholtz P, Larsen N et al (2006) Greengenes, a chimera-checked 16S rRNA gene database and workbench compatible with ARB. *Appl Environ Microbiol* 72(7):5069–5072
18. Cole JR, Chai B, Marsh TL et al (2003) Ribosomal Database Project. The ribosomal database project (RDP-II): previewing a new

- autoaligner that allows regular updates and the new prokaryotic taxonomy. *Nucleic Acids Res* 31(1):442–443
19. Pruesse E, Quast C, Knittel K et al (2007) SILVA: a comprehensive online resource for quality checked and aligned ribosomal RNA sequence data compatible with ARB. *Nucleic Acids Res* 35(21):7188–7196
20. Roche Applied Sciences (2008) Genome sequencer data analysis software manual. Roche Diagnostics GmbH, Germany
21. Metzker ML (2010) Sequencing Technologies - the Next Generation. *Nat Rev Genet* 11(1): 31–46
22. Ewing B, Green P (1998) Base-calling of automated sequencer traces using phred II error probabilities. *Genome Res* 8(3):186–194
23. Quince C, Lanzen A, Davenport RJ et al (2011) Removing noise from pyrosequenced amplicons. *BMC Bioinformatics* 12:38
24. Schloss PD (2009) A high-throughput DNA sequence aligner for microbial ecology studies. *PLoS One* 4(12):e8230
25. Balzer S, Malde K, Lanzén A et al (2010) Characteristics of 454 pyrosequencing data-enabling realistic simulation with flowsim. *Bioinformatics* 26(18):i420–i425
26. Huse SM, Huber JA, Morrison HG et al (2007) Accuracy and quality of massively parallel DNA pyrosequencing. *Genome Biol* 8(7): R143
27. Chuong BD, Batzoglou S (2008) What is the expectation maximization algorithm? *Nat Biotechnol* 26(8):897–899
28. Wang Q, Garrity GM, Tiedje JM et al (2007) Naïve bayesian classifier for rapid assignment of rRNA sequences into the new bacterial taxonomy. *Appl Environ Microbiol* 73(16):5261–5267
29. Cole JR, Wang Q, Cardenas E et al (2009) The Ribosomal Database Project: improved alignments and new tools for rRNA analysis. *Nucleic Acids Res* 37(Database issue):D141–D145. doi:[10.1093/nar/gkn879](https://doi.org/10.1093/nar/gkn879)
30. Claesson MJ, O'Sullivan O, Wang Q et al (2009) Comparative analysis of pyrosequencing and a phylogenetic microarray for exploring microbial community structures in the human distal intestine. *PLoS One* 4(8):e6669
31. Gosalbes MJ, Abellan JJ, Durbán A et al (2012) Metagenomics of human microbiome: beyond 16s rDNA. *Clin Microbiol Infect* 18(4):47–49
32. Andersson AF, Lindberg M, Jakobsson H et al (2008) Comparative analysis of human gut microbiota by barcoded pyrosequencing. *PLoS One* 3(7):e2836
33. Malde K (2011) Flower: extracting information from pyrosequencing data. *Bioinformatics* 27(7):1041–1042
34. Caporaso JG, Kuczynski J, Stombaugh J et al (2010) QIIME allows analysis of high-throughput community sequencing data. *Nat Methods* 7(5):335–336

Chapter 17

Deciphering Metatranscriptomic Data

**Evguenia Kopylova, Laurent Noé, Corinne Da Silva,
Jean-Frédéric Berthelot, Adriana Alberti, Jean-Marc Aury,
and Hélène Touzet**

Abstract

Metatranscriptomic data contributes another piece of the puzzle to understanding the phylogenetic structure and function of a community of organisms. High-quality total RNA is a bountiful mixture of ribosomal, transfer, messenger and other noncoding RNAs, where each family of RNA is vital to answering questions concerning the hidden microbial world. Software tools designed for deciphering metatranscriptomic data fall under two main categories: the first is to reassemble millions of short nucleotide fragments produced by high-throughput sequencing technologies into the original full-length transcriptomes for all organisms within a sample, and the second is to taxonomically classify the organisms and determine their individual functional roles within a community. Species identification is mainly established using the ribosomal RNA genes, whereas the behavior and functionality of a community is revealed by the messenger RNA of the expressed genes. Numerous chemical and computational methods exist to separate families of RNA prior to conducting further downstream analyses, primarily suitable for isolating mRNA or rRNA from a total RNA sample. In this chapter, we demonstrate a computational technique for filtering rRNA from total RNA using the software SortMeRNA. Additionally, we propose a post-processing pipeline using the latest software tools to conduct further studies on the filtered data, including the reconstruction of mRNA transcripts for functional analyses and phylogenetic classification of a community using the ribosomal RNA.

Key words Metatranscriptomics, High-throughput sequencing, 16S rRNA phylogenetic analysis

1 Introduction

Metatranscriptomics is the study of total RNA sequenced from a group of microorganisms directly extracted from their native environment. It successfully exploits the well-known RNA-Seq technique to transcriptome profiling based on leading high-throughput sequencing technologies, such as Illumina Solexa (Illumina Inc.), 454 (Roche Diagnostics), and Ion semiconductor sequencing (Ion Torrent Systems Inc.). RNA-Seq offers key advantages over previous technologies, such as array-based (like tiling [1]) and tag-based (like SAGE [2] or CAGE [3]). First, it is not limited to

detecting transcripts that correspond to existing genomic sequences. It is then particularly attractive for non-model organisms without an existing genome assembly. It can reveal the precise location of splicing junctions, transcription start and stop, to a single-base resolution. Furthermore, 30-bp short reads from RNA-Seq give us information about how two exons are connected, whereas longer reads or pair-end short reads should reveal connectivity between multiple exons. These factors make RNA-Seq useful for studying complex transcriptomes.

When applied to whole environments, RNA-seq can sequence millions of uncultured organisms in parallel and provide an authentic representation of species richness within a community at the time of the sampling. Phylogenetic structure of a community is primarily established using the 16S and 18S ribosomal RNA (rRNA) genes, which remain highly conserved among different species of bacteria, archaea and eukarya [4]. Other inquiries can be made regarding the functionality of a community by studying the messenger RNA (mRNA) of the actively transcribed genes. Thus, it is of primary interest to sort out the rRNA from the mRNA in the total RNA for answering the most basic questions regarding the species composition, gene regulation, and protein information of a metatranscriptome. Subject to prokaryotic or eukaryotic cells, the rRNA content can represent up to 80–85 % of total RNA and the protein coding mRNA as little as 1–6 % [5]. If one wants to focus exclusively on mRNA, there exist prior-to-sequencing methods to help isolate and enrich mRNA from the total RNA. These methods focus on the depletion of rRNA with technologies such as subtractive hybridization (Life Technologies), exonuclease digestion [6] and the duplex-specific nuclease treatment (DSN) [7]. However, even selective removal of rRNA genes does not guarantee definite depletion, and some rRNA genes can still remain in the metatranscriptomic sample. Alternatively, if the goal is sort apart RNA without any loss of information, in the case of studying rRNA for species identification, noninvasive methods for separating RNA are also preferred. Such methods implement a bioinformatics pipeline to catalog families of RNA using publicly available RNA databases. To this end, we introduce SortMeRNA [8], which is the fastest existing tool for filtering out rRNA from a total RNA dataset. The underlying algorithm takes into account the high-conservation property of rRNA sequences and achieves a high sensitivity.

This chapter is organized as follows. Following this introduction, in Subheading 2 we list the tools used for data cleaning and sorting metatranscriptomic reads. In Subheading 3, we describe the steps for filtering RNA from a total metatranscriptome using the software SortMeRNA. Lastly, in Subheading 3.3 we summarize the latest software tools developed to further process and analyze sorted metatranscriptomic data, including methods for assembly, mapping, functional analysis, and taxonomic analysis.

2 Materials

2.1 Software

The software tool SortMeRNA is a fast and accurate filter for identifying and sorting rRNA in a set of metatranscriptomic reads. SortMeRNA is written in C++ and freely distributed under the GPL license as a stand-alone version or as a Galaxy wrapper. Both distributions, including the user manual with installation instructions, can be downloaded from <http://bioinfo.lifl.fr/RNA/sortmerna/>. SortMeRNA supports multi-threading and has been tested on Linux (Ubuntu, Fedora, CentOS, and Debian) and Mac OS 10.6.8 systems. For compilation, a g++ compiler version 4.3 or higher is required. Our experiments were performed using version 1.8 of SortMeRNA, on an Intel(R) Xeon(R) CPU W3520 2.67 GHz machine with 8 GB of RAM and one thread.

2.2 Ribosomal RNA Databases

To filter out rRNA, SortMeRNA performs a search against a database of known rRNAs, such as the public rRNA databases SILVA [9], Greengenes [10], Rfam [11], and RDP-II [12], that provides access to millions of rRNA sequences. SortMeRNA can be used with any of these databases, any fraction of them, or with any custom database. The only requirement is that sequences are in the FASTA format. For easier use, the distribution package of SortMeRNA also includes eight nonredundant representative databases for 16S bacteria and archaea, 18S eukarya, 23S bacteria and archaea, 28S eukarya, 5S bacteria/archaea, and 5.8S eukarya, which were derived from the raw SILVA rRNA database sets using the tools ARB [13] and UCLUST [14]. We first remove contaminated sequences, such as chimeric rRNA and partial mRNA, and then use the tool UCLUST to construct a nonredundant database according to the identity threshold of the curated sequences.

2.3 Metatranscriptomic Reads

In the following Subheading 3, we use SortMeRNA to filter rRNA from a total metatranscriptomic dataset ERA236149 sequenced from a marine environment. Total RNA was chemically fragmented and converted into single-stranded cDNA using random hexamer priming. Then, the second strand was generated to create double-stranded cDNA. The library was prepared following Illumina's protocol and sequenced using 101 base-length read chemistry on the Illumina HiSeq2000. After quality and adapters cleaning, we obtain 20,968,598 useful reads longer than 30 nucleotides.

Cleaning the reads is an important step to enhance software performance and the quality of the results [15] (Table 1). Our procedure is as follows:

1. Filter the raw data to remove any clusters that have “too much” intensity corresponding to bases other than the called base.

Table 1
List of examples of relevant software which can be used for performing read quality control

Filter	Program
Vector/Adapter clipping	TagCleaner [16], FASTX-Toolkit
Low quality trimming	PRINSEQ [17], FASTX-Toolkit
Low complexity regions	dust [18]
Read length	PRINSEQ
Clone removal	PRINSEQ
Error correction	Coral [19]

2. Remove adapters and primers on the whole read and low quality nucleotides from both ends (while quality value is lower than 20) and then continue next steps with the longest sequence without adapters and low quality bases.
3. Remove sequences between the second unknown nucleotide (N) and the end of the read. This step is done on trimmed (adapters + quality) reads.
4. Discard reads shorter than 30 nucleotides after trimming.
5. Remove reads that mapped onto run quality control sequences (PhiX genome).

3 Methods

3.1 Command-Line

3.1.1 Input

SortMeRNA takes two mandatory inputs:

1. A FASTA or FASTQ file of reads.
2. A FASTA file for the database of full-length rRNA sequences.

The file of rRNA sequences must be first indexed using the command “buildtrie” and then the index is automatically loaded when “sortmerna” is launched. For example, we are interested in filtering out all bacterial and archaeal 16S rRNA from the metatranscriptomic dataset ERA236149 using SILVA database. Firstly, we construct the index using the rRNA databases provided in the distribution package:

```
buildtrie --db ./rRNA_databases/silva-bac-16s-database-id85.fasta
buildtrie --db ./rRNA_databases/silva-arc-16S-database-id95.fasta
```

This has to be done only once if one wants to process several sets of reads.

3.1.2 How to Run SortMeRNA

We filter the 16S rRNA from our ERA236149 dataset using our two previously indexed databases. The command-line is as follows:

```
sortmerna -n 2 --db ./rRNA_databases/silva-bac-16s-database-id85.
           fasta ./rRNA_databases/silva-arc-16S-database-id95.fasta --I
           ERA236149.fastq --accept rrna --bydbs --other nonrrna --log bilan
```

Where the options correspond to:

Input:

- n <N> is the number <N> of databases to search,
- db <string> is for a space separated list of <N> databases,
- I <filename> is for the input reads,

Output:

- accept <filename without extension> is for the output file(s) of matching reads,
- bydbs is to output the matching reads to <N> different files (one for each database),
- other <filename without extension> is the output file for non-matching reads,
- log <filename> gives a statistics file for the percentages of reads matching to each of the <N> databases.

Paired reads: paired reads are managed using the options--paired-in and--paired-out. Although SortMeRNA does not use parity information during filtering, when run in multi-threading mode, the output of matching and non-matching reads can lose the parity order of the original file. To maintain parity of reads, if one read matches and the other does not, the option--paired-in will output the paired reads into the--accept file, whereas the option--paired-out will output the paired reads into the--other file. If neither of the options are set, by default the output of SortMeRNA may not maintain parity order.

3.1.3 Output

From our example above, there will be four output files: two files for rRNA classified reads for each 16s-bacteria, 16s-archaea database, one file for the non-rRNA reads, and one file for the overall statistics of the ERA236149 dataset tested above. The sorting of ~21 million reads was performed 1.5 h using one thread. From the statistics file, approximately 27 % of these reads were classified as 16S rRNA to one of two reference databases.

3.1.4 Advanced Options

Memory management: The size of ERA236149.fastq is 5.2 GB, by default SortMeRNA works internally with 1 GB of memory allocated for the reads, without any required user intervention to split the file into multiple parts. If the user has sufficient memory to load all 5.2 GB of reads into memory, the option -m <M> allows

to set the memory limit to M bytes. Likewise, the memory limit can be reduced to under 1 GB for systems with RAM restrictions.

Multi-threading: The parameter `-a <A>` will use A number of threads, it is set to $A=1$ by default.

3.2 Galaxy

SortMeRNA is also available through Galaxy [20], which is an open, Web-based platform for computational research in the life sciences. Galaxy provides users with a graphical work-flow management system, and emphasizes accessibility, reproducibility and transparency. Most of the same options are available in Galaxy as the stand-alone version of SortMeRNA (excluding the accept by database--bydbs option).

3.2.1 Installation

The Galaxy wrapper for SortMeRNA is available on the Toolshed, the place for sharing Galaxy tools with any Galaxy instance. Any local administrator may install SortMeRNA by browsing the Toolshed, and Galaxy takes care of downloading and installing SortMeRNA, as well as indexing the ribosomal RNA databases provided with SortMeRNA.

3.2.2 Input

The Galaxy main graphical interface has three parts (*see* Fig. 1): the left panel lists all the available tools; the right panel is the history of

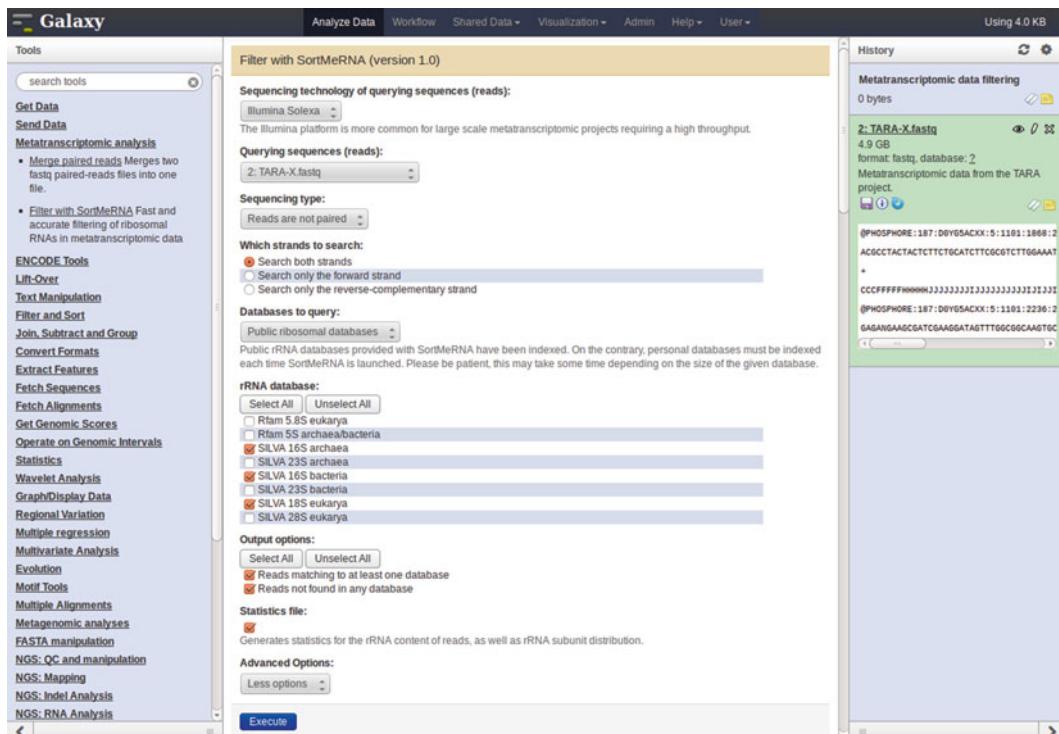


Fig. 1 Galaxy interface for SortMeRNA

all the steps of the current analysis and their outputs. The central panel displays all of the input, output and parameter criteria for launching SortMeRNA.

The input FASTA or FASTQ file of reads must be first imported to the Galaxy history. Small files may be uploaded from the local hard drive, by selecting the “Upload file from your computer” in the “Get data” section; bigger files can be transferred from an accessible URL or by FTP. Once done, our file “ERA236149.fasta” and its associated metadata are visible in the right panel history.

We bring up SortMeRNA by selecting “Filter with SortMeRNA” in the left panel. We select the desired reads file from the history—only FASTA/FASTQ files can be selected. Most of the options selected by default are appropriate for our experiment; we tick the 16S databases. We want to retrieve both the classified reads and non-rRNA reads, as well as the statistics file, so we tick all three options.

3.2.3 Output

The SortMeRNA outputs are automatically imported to the history as Galaxy datasets, and are recognized as FASTA/FASTQ files, which can be used for further analysis inside Galaxy.

3.2.4 Advanced Options

It is possible to use personal databases to filter our reads against. We must first import them to Galaxy as FASTA files, just like the reads file. In the SortMeRNA interface, we then select “Databases from your history” as “Databases to query,” and add as many files as we want. The indexation of the rRNA sequences file will be automatically invoked by Galaxy before running SortMeRNA—note however that the index must be computed each time, which is time-consuming. Alternatively, more databases can be added to the default list by the Galaxy administrator.

3.3 Further Analyses

Once SortMeRNA has been run, one might want to perform further analyses of the filtered dataset. In this section we touch on various stages of filtered data manipulation, such as the reconstruction of full-length RNA from short reads, taxonomic assignation for ribosomal RNA, and functional analysis of messenger RNA using well-maintained protein databases. Please refer to Fig. 2 for an illustration of the pipeline. Many of the tools shown in this pipeline are also available in Galaxy.

3.3.1 Assembly

De novo assembly of metatranscriptomic data is in the early stages of development, and up to now there does not exist a tool *specifically* designed to assemble metatranscriptomic data. It is seldom conducted on the total RNA sample, largely because assembly of rRNA genes is a very resource intensive task and susceptible to chimera formation from the variable and conserved regions on the SSU rRNA molecule [21, 22]. However, if the user has a sorted file of rRNA reads, they can perform full-length reconstruction

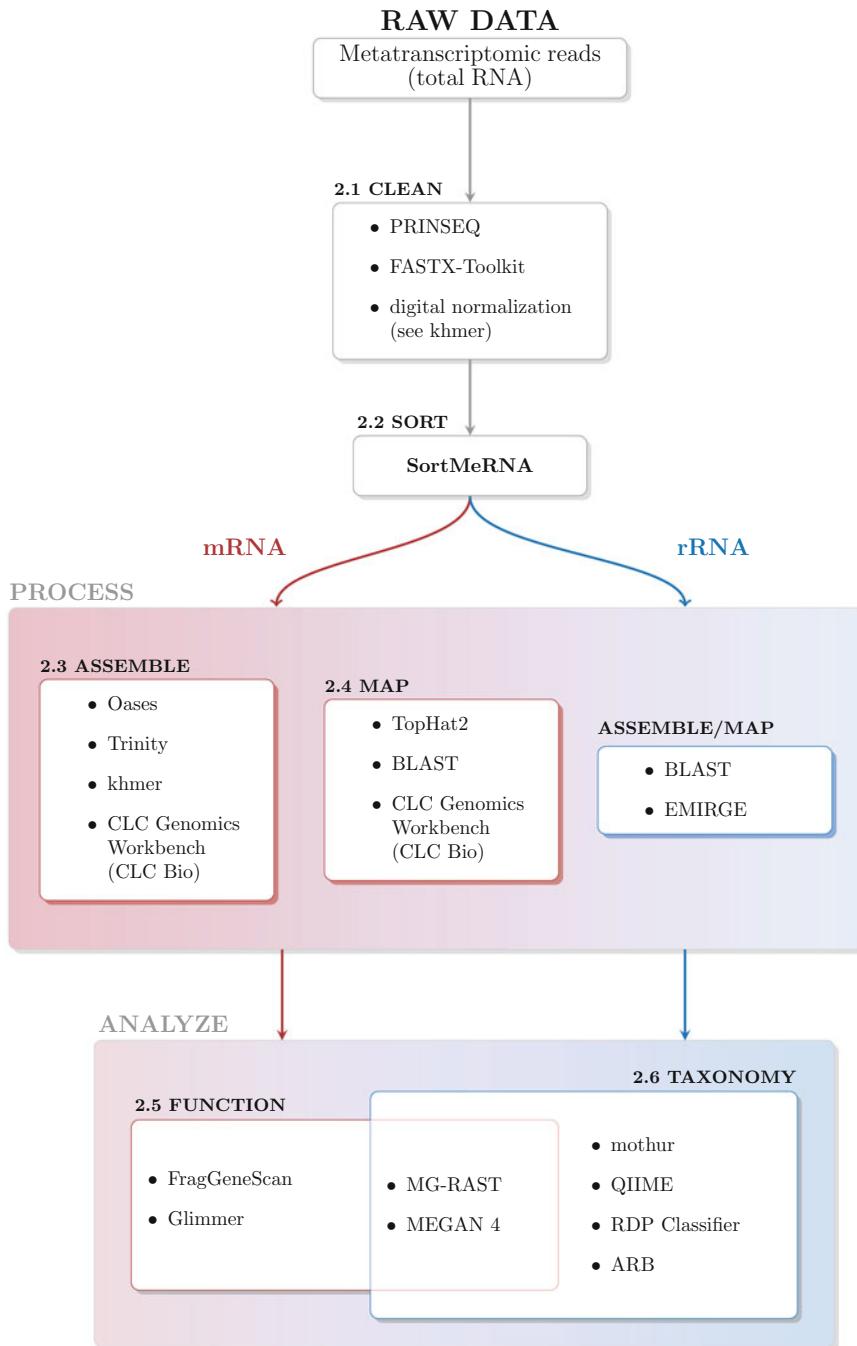


Fig. 2 Metatranscriptomic analysis flowchart

using the program EMIRGE [23], which can also provide relative abundances of rRNA in the community sample. For reconstructing full-length mRNA transcripts from a metatranscriptome, we describe two work-around methods applicable for raw or

normalized data. The first approach is to assemble short read sequencing data into longer contigs using bioinformatic tools such as the CLC Genomics Workbench (CLC Bio) or SOAPdenovo [24]. Afterward, the longer contigs can be further assembled using Newbler (454 Life Sciences) (designed for 454 pyrosequencing, known to work with Illumina contigs [25]), or merged together using AMOS Minimus2 [26] to reconstruct optimal-length transcripts. The second approach is to use a transcriptome assembler such as Oases [27] or Trinity [28], which can detect alternative splicing events and duplicated genes, and predict distinct full-length transcripts through assembly merging. If the reads will be used for assembly, reducing high-coverage data and selecting reads with low error rates can greatly improve the quality and computing performance. The program khmer [29] includes a single-pass digital normalization prefilter to even out random sampling variation of your metatranscriptome by removing redundant and erroneous sequences. Likewise, filtering out low-abundance k-mers (-d option in SOAPdenovo) will reduce the computational requirements for all de novo assemblers using a de Bruijn graph approach.

3.3.2 Mapping

Metatranscriptomic read mapping assists in recovering the arrangement of short RNA reads with respect to long reference sequences, such as genes, metagenomic contigs/ORFs, complete genomes, or rRNA molecules. This analysis can be particularly useful for metatranscriptomes of low level coverage and high diversity, where no significant overlap exists between the sequenced reads to facilitate de novo assembly. The most mature and informative tool for local sequence alignment is BLAST [30], although for large-scale high-throughput metatranscriptomic projects, the execution time is substantially slower in comparison with new-generation aligners and therefore more practical to be integrated in downstream analyses on smaller sized data. The most popular read-mapping software applying is Bowtie [31], which is the core alignment tool behind many cutting-edge read mapping methods, such as TopHat2 [32] and EMIRGE, due to its small memory footprint and rapid alignment. In addition to Bowtie1, Bowtie2 [33] supports both local and gapped alignment, which makes it more practical for reads prone to homopolymer errors such as those produced by 454 and Ion Torrent technologies.

3.3.3 Functional Analysis of mRNA

For assessing the potential functions of mRNA sequences, it is routine to search the reads or assembled transcripts (*see* Subheading 3.3.1) for homologs in a public reference database such as the NCBI non-redundant protein database (NCBI-nr) [34], the RDP-II database, KEGG (for enzymes and pathway networks) [35], or SEED (protein sequences with known functional roles) [36]. The MG-RAST [37] server and MEGAN4 [38] are two services providing comprehensive tools for the annotation and functional

analysis of metagenomes and metatranscriptomes (prokaryotes only for MG-RAST), including comparative analysis between multiple community datasets. Both services have been documented to generate comparable results [39]; however, MEGAN4 can illustrate hierarchical representations of functional assignments and supports easy local installation for scientists not inclined to upload their unpublished data. The tool MEGAN4 uses the KEGG classification to calculate molecular interaction and reaction pathways including metabolism and cellular processes for both prokaryotes and eukaryotes, and the SEED classification to assign each read to the highest scoring gene with a known functional role. Alternatively, MG-RAST provides access to computational power needed for high-quality annotations through a CLOUD client.

Gene prediction programs such as FragGeneScan [40] and Glimmer [41], provide an alternative ab initio approach to detect novel genes in sufficiently long assembled contigs when no close homologues of coding regions can be found in a reference database. The underlying method of these tools utilizes Hidden Markov models, which are trained on different species of prokaryotes, eukaryotes, viruses, phages, and plasmids to recognize protein coding and noncoding regions of input DNA sequences. Given the length of genes in prokaryotes can easily exceed 1,000 nucleotides [42], the user should be aware of the restricted identification of open reading frames (ORFs) when working with very short reads, since the in-frame start and stop codons may be split between reads.

3.3.4 Taxonomic Classification of rRNA

Due to the high conservation of rRNA genes between species and their presence in all cells, the 16S and 18S small subunit rRNA serve as signatures to infer whole organism phylogeny [43]. For a reliable representation of species diversity, short reads covering rRNA variable regions are more accurately classified than those covering higher-conserved regions. Therefore, tools which are more sensitive to capture sequence variation will render a higher taxonomic resolution of a given metatranscriptome. In the following two paragraphs, we describe two methods for rRNA classification, either by using the raw unassembled short reads, or with an assembled set of full-length rRNA genes.

Short rRNA fragments (reads): MEGAN4 implements visual and interactive phylogenetic trees from a given set of rRNA reads, and allows the user to carefully examine the alignment of each read with its assigned species. To use MEGAN4, the user must first run a BLASTN search to compare a set of reads to the SILVA database and then import the BLAST results file to calculate taxonomic classification using the NCBI taxonomy. Although the searching step can be computationally expensive for large set of reads and relaxed parameter settings, the BLAST algorithm can be efficient at finding the reads covering variable regions in low coverage samples.

The integrated software suite mothur [44] provides one of the most complete packages to preprocess and assign 16S pyrosequencing or Illumina reads to operational taxonomic units (OTUs), measure species diversity and display the results using Venn diagrams, heat maps, and dendrograms. The open source tool QIIME [45] provides most of the similar community analysis tools as mothur, however with more advanced visualization representations for network analyses, histograms, and phylogenetic trees. Lastly, the RDP Classifier [46] uses a fast, alignment-free naïve Bayesian algorithm to classify both short and full-length prokaryotic rRNA sequences into higher-order taxonomic ranks and the results can be input to QIIME for computing diversity estimates.

Full-length rRNA sequences: Following the reconstruction of full-length rRNA genes from a pool of isolated short rRNA reads, we can apply classical phylogenetic procedures to align them to existing rRNA databases. The ARB package provides a graphic environment for visualizing phylogenetic trees, aligning sequences, and editing primary and secondary structures of both SSU and LSU rRNA genes. The phylogenetic tree building tools include both phenetic and cladistic methods, allowing for an exhaustive study of relationships among a group of organisms and their pathways of evolution.

Acknowledgments

This research was supported by the French National Agency for Research (grant ANR-2010-COSI-004) and the French National Sequencing Center (Genoscope).

References

- Kapranov P et al (2007) RNA maps reveal new RNA classes and a possible function for pervasive transcription. *Science* 316(5830):1484–1488
- Velculescu VE et al (1995) Serial analysis of gene expression. *Science* 270(5235):484–487
- Shiraki T et al (2003) Cap analysis gene expression for high-throughput analysis of transcriptional starting point and identification of promoter usage. *Proc Natl Acad Sci U S A* 100(26):15776–15781
- Janda JM, Abbott SL (2007) 16S rRNA gene sequencing for bacterial identification in the diagnostic laboratory: pluses, perils, and pitfalls. *J Clin Microbiol* 45(9):2761–2764
- Sorek R, Cossart P (2010) Prokaryotic transcriptomics: a new view on regulation, physiology and pathogenicity. *Nat Rev Genet* 11(1):9–16
- Boissinot K, Huletsky A, Peytavi R et al (2007) Rapid exonuclease digestion of PCR-amplified targets for improved microarray hybridization. *Clin Chem* 53(11):2020–2023
- Yi H, Cho YJ, Won S et al (2011) Duplex-specific nuclease efficiently removes rRNA for prokaryotic RNA-seq. *Nucleic Acids Res* 39(20):e140
- Kopylova E, Noe L, Touzet H (2012) SortMeRNA: fast and accurate filtering of ribosomal RNAs in metatranscriptomic data. *Bioinformatics* 28(24):3211–3217
- Quast C, Pruesse E, Yilmaz P et al (2013) The SILVA ribosomal RNA gene database project: improved data processing and web-based tools. *Nucleic Acids Res* 41(D1):D590–D596
- DeSantis TZ, Hugenholtz P, Larsen N et al (2006) Greengenes, a chimera-checked 16S rRNA gene database and workbench compatible

- with ARB. *Appl Environ Microbiol* 72(7): 5069–5072
11. Griffiths-Jones S, Bateman A, Marshall M et al (2003) Rfam: an RNA family database. *Nucleic Acids Res* 31(1):439–441
 12. Cole JR, Wang Q, Cardenas E et al (2008) The Ribosomal Database Project: improved alignments and new tools for rRNA analysis. *Nucleic Acids Res* 37:D141–D145
 13. Ludwig W, Strunk O, Westram R et al (2004) ARB: a software environment for sequence data. *Nucleic Acids Res* 32(4):1363–1371
 14. Edgar RC (2010) Search and clustering orders of magnitude faster than BLAST. *Bioinformatics* 26(19):2460–2461
 15. Brown CT, Howe A, Zhang Q et al (2013) A reference-free algorithm for computational normalization of shotgun sequencing data. <https://www.e-biogenouest.org/resources/46>
 16. Schmieder R, Lim YW, Rohwer F et al (2010) TagCleaner: identification and removal of tag sequences from genomic and metagenomic datasets. *BMC Bioinformatics* 11:341
 17. Schmieder R, Edwards R (2011) Quality control and preprocessing of metagenomic datasets. *Bioinformatics* 27(6):863–864
 18. Morgulis A, Gertz EM, Schäffer AA et al (2006) A fast and symmetric DUST implementation to mask low-complexity DNA sequences. *J Comput Biol* 13(5):1028–1040
 19. Salmela L, Schroder J (2011) Correcting errors in short reads by multiple alignments. *Bioinformatics* 27(11):1455–1461
 20. Goecks J, Nekrutenko A, Taylor J, The Galaxy Team (2010) Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome Biol* 11(8):R86. doi:[10.1186/gb-2010-11-8-r86](https://doi.org/10.1186/gb-2010-11-8-r86)
 21. Radax R, Rattei T, Lanzen A et al (2012) Metatranscriptomics of the marine sponge *Geodia barretti*: tackling phylogeny and function of its microbial community. *Environ Microbiol* 14(5):1308–1324
 22. Fan L, McElroy K, Thomas T (2012) Reconstruction of ribosomal RNA genes from metagenomic data. *PLoS One* 7(6):e39948. doi:[10.1371/journal.pone.0039948](https://doi.org/10.1371/journal.pone.0039948)
 23. Miller CS, Baker BJ, Thomas BC et al (2011) EMIRGE: reconstruction of full-length ribosomal genes from microbial community short read sequencing data. *Genome Biol* 12(5):R44
 24. Luo R, Liu B, Xie Y et al (2012) SOAPdenovo2: an empirically improved memory-efficient short-read de novo assembler. *GigaScience*. doi:[10.1186/2047-217X-1-18](https://doi.org/10.1186/2047-217X-1-18)
 25. Mason OU, Hazen TC, Borglin S et al (2012) Metagenome, metatranscriptome and single-cell sequencing reveal microbial response to Deepwater Horizon oil spill. *ISME J* 6(9): 1715–1727
 26. Sommer DD, Delcher AL, Salzberg SL et al (2007) Minimus: a fast, lightweight genome assembler. *BMC Bioinformatics*. doi:[10.1186/1471-2105-8-64](https://doi.org/10.1186/1471-2105-8-64)
 27. Schulz MH, Zerbino DR, Vingron M et al (2012) Oases: robust de novo RNA-seq assembly across the dynamic range of expression levels. *Bioinformatics* 28(8):1086–1092
 28. Grabherr MG, Haas BJ, Yassour M et al (2011) Full-length transcriptome assembly from RNA-Seq data without a reference genome. *Nat Biotechnol* 29:644–652
 29. Pell J, Hintze A, Canino-Koning R et al (2012) Scaling metagenome sequence assembly with probabilistic de Bruijn graphs. *Proc Natl Acad Sci U S A*. doi:[10.1073/pnas.1121464109](https://doi.org/10.1073/pnas.1121464109)
 30. Altschul SF, Gish W, Miller W et al (1990) Basic local alignment search tool. *J Mol Biol* 215(3):403–410
 31. Langmead B, Trapnell C, Pop M et al (2009) Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biol* 10:R25
 32. Kim D, Pertea G, Trapnell C et al (2013) TopHat2: accurate alignment of transcriptomes in the presence of insertions, deletions and gene fusions. *Genome Biol* 14(4):R36
 33. Langmead B, Salzberg SL (2012) Fast gapped-read alignment with Bowtie 2. *Nat Methods* 9(4):357–359
 34. Pruitt KD, Tatusova T, Maglott DR (2005) NCBI Reference Sequence (RefSeq): a curated non-redundant sequence database of genomes, transcripts and proteins. *Nucleic Acids Res* 33:D501–D504
 35. Kanehisa M, Goto S, Sato Y et al (2012) KEGG for integration and interpretation of large-scale molecular datasets. *Nucleic Acids Res* 40:D109–D114
 36. Overbeek R, Begley T, Butler RM et al (2005) The subsystems approach to genome annotation and its use in the project to annotate 1000 genomes. *Nucleic Acids Res* 33(17): 5691–5702
 37. Meyer F, Paarmann D, D’Souza M et al (2008) The metagenomics RAST server—a public resource for the automatic phylogenetic and

- functional analysis of metagenomes. *BMC Bioinformatics* 9:386. doi:[10.1186/1471-2105-9-386](https://doi.org/10.1186/1471-2105-9-386)
38. Hudson DH, Mitra S, Ruscheweyh HJ et al (2011) Integrative analysis of environmental sequences using MEGAN4. *Genome Res* 21(9):1552–1560
39. Mitra S, Rupek P, Richter DC et al (2011) Functional analysis of metagenomes and metatranscriptomes using SEED and KEGG. *BMC Bioinformatics* 12(Suppl 1):S21
40. Rho M, Tang H, Ye Y (2010) FragGeneScan: predicting genes in short and error-prone reads. *Nucleic Acids Res* 38(20):e191
41. Delcher AL, Bratke KA, Powers EC et al (2007) Identifying bacterial genes and endosymbiont DNA with Glimmer. *Bioinformatics* 23(6):673–679
42. Lin X, Hong C, Xiaohua H et al (2006) Average gene length is highly conserved in prokaryotes and eukaryotes and diverges only between the two kingdoms. *Mol Biol Evol* 23(6):1107–1108
43. Lane DJ, Pace B, Olsen GJ et al (1985) Rapid determination of 16S ribosomal RNA sequences for phylogenetic analyses. *Proc Natl Acad Sci U S A* 82(20):6955–6959
44. Schloss PD, Westcott SL, Ryabin T et al (2009) Introducing mothur: open-source, platform-independent, community-supported software for describing and comparing microbial communities. *Appl Environ Microbiol* 75(23):7537–7541
45. Caporaso JG, Kuczynski J, Stombaugh J et al (2010) QIIME allows analysis of high-throughput community sequencing data. *Nat Methods* 7(5):335–336
46. Wang Q, Garrity GM, Tiedje JM et al (2007) Naive Bayesian classifier for rapid assignment of rRNA sequences into the new bacterial taxonomy. *Appl Environ Microbiol* 73(16): 5261–5267

Chapter 18

RIP-Seq Data Analysis to Determine RNA–Protein Associations

Federico Zambelli and Giulio Pavesi

Abstract

Next-generation sequencing (NGS) technologies have opened new avenues of unprecedented power for research in molecular biology and genetics. In particular, their application to the study of RNA-binding proteins (RBPs), extracted through immunoprecipitation (RIP), permits to sequence and characterize all RNAs that were found to be bound in vivo by a given RBP (RIP-Seq). On the other hand, NGS-based experiments, including RIP-Seq, produce millions of short sequence fragments that have to be processed with suitable bioinformatic tools and methods to recover and/or quantify the original sequence sample. In this chapter we provide a survey of different approaches that can be taken for the analysis of RIP-Seq data and the identification of the RNAs bound by a given RBP.

Key words Immunoprecipitation, Next-generation sequencing, RNA binding protein, RIP-Seq, CLIP-Seq, PAR-CLIP

1 Introduction

Next-generation sequencing (NGS) technologies have opened new avenues of unprecedented power for research in molecular biology and genetics. Other than their straightforward application to the de novo sequencing or resequencing of genomes, genomic loci, or transcriptomes, they can be used with excellent results to the characterization and/or quantification of the sequences contained in a given DNA or RNA sample [1]. Indeed, in the case of RNAs, the ever decreasing cost of sequencing has made NGS the method of choice not only for the characterization of transcriptomes and identification of alternative splicing, but also for simpler tasks like the quantification of transcript abundance, rapidly replacing oligonucleotide microarrays as the de facto standard for these applications [2]. In the latter case, as in the traditional Sanger method, sequencing is usually performed after the isolated RNA has been converted into a cDNA library.

RNAs in cells are associated with RNA-binding proteins (RBPs) to form ribonucleoprotein (RNP) complexes [3]. RBPs determine the structure and interactions of the RNAs, and their biogenesis, stability, function, and cellular localization. Eukaryotic genomes can encode hundreds of RBPs (thousands in vertebrates), each of which has unique RNA-binding specificity. Indeed, another broad field of application of NGS technologies is the characterization of DNA or RNA samples, extracted for example through an ImmunoPrecipitation (IP) experiment [4]. That is, given a DNA- or RNA-binding protein of interest for which suitable antibodies are available, the protein can be extracted from cells together with DNA or RNA sequences that were bound to it *in vivo*. Since RBP binding occurs simultaneously in the cells analyzed thousands of times for the same RNA molecule, once separated from the protein the DNA or RNA fragments bound should be found in many copies, and thus be enriched in the sample extracted. NGS technologies offer the most straightforward solution to the problem of identifying enriched DNA/RNA fragments in the sample, that is, to sequence the sample itself. The more times a given fragment is sequenced, the more likely it is to be actually bound *in vivo* by the protein investigated [5, 6].

The main issue that in general one has to face with NGS technologies applied to IP samples is that at the present time they do not permit to sequence the entire DNA or RNA molecules obtained from the IP, but only a short fragment at either or both 5' ends. A single sequencing lane of Illumina/Solexa platform, which has become the method of choice for this kind of experiments, produces millions of sequence reads whose length does not exceed 75–100 bps. Thus, bioinformatic expertise is needed first to try and reconstruct from the short sequence reads which were the original DNA or RNA sequences bound by the protein, and also if the respective abundance estimated from the short NGS sequence reads can be considered to be due to a (statistically) significant enrichment [7].

2 Materials

We list in the following bioinformatic tools and software that can be used to process data from RIP-Seq and similar experiments. Details on the purpose and application of each one can be found in Subheading 3.

TopHat [8]: a fast splice junction mapper for RNA-Seq reads. It aligns RNA-Seq reads to mammalian-sized genomes using the ultrahigh-throughput short read aligner Bowtie [9], and then analyzes the mapping results to identify splice junctions between exons. Source code and executables available at <http://ccb.jhu.edu/software/tophat/>.

GSNAP [10]: another mapping software especially devised for mapping short reads derived from RNA-Seq, taking into account the presence of introns. Available at <http://research-pub.gene.com/gmap/>.

RSEM [11]: A software package including tools for the mapping of RNA-Seq reads against a reference transcriptome annotation, and the quantification of the expression level of each transcript. It includes EBSeq [12], for the identification of differentially enriched transcripts and/or genes. Available at <http://deweylab.biostat.wisc.edu/rsem/>.

DESeq [13]—DEXSeq [14]: R packages to analyze count data from high-throughput sequencing assays such as RNA-Seq and test for differential expression and/or differential exon usage. Take as input RNA-Seq reads aligned to the genome with tools like TopHat. In order to be run, R and Bioconductor need to be installed beforehand. Available at <http://www.bioconductor.org>.

Cufflinks [15]: a series of tools for RNA-Seq analysis, including transcript assembly, transcript level estimation, and testing for differential expression and regulation in RNA-Seq samples. It accepts as input RNA-Seq reads aligned to the genome with tools like TopHat. Available at <http://cufflinks.ccb.umd.edu/>.

RIPSeeker [16]: Infer and discriminate RIP peaks from the alignment of RIP-seq sequences to the genome. It also provides a suite of bioinformatic tools addressing issues ranging from post-alignment processing to visualization and annotation. In order to be run needs R and Bioconductor to be installed beforehand. Available at <http://www.bioconductor.org>.

PARalyzer [17]: Aligns sequencing reads generated by the PAR-CLIP (Photoactivatable-Ribonucleoside-Enhanced Cross-linking and Immunoprecipitation) protocol to the genome, and identifies RBP binding sites by exploiting T=>C transitions induced by cross-linking. Available at https://ohlerlab.mdc-berlin.de/software/PARalyzer_85/.

3 Methods

3.1 RIP-Seq and Its Variants

In general, the steps that have to be followed for an IP experiment are similar regardless of the protein studied or the antibody employed. From a bioinformatic point of view, the main difference is on whether and how the protein is cross-linked (fixed) to the DNA/RNA before the IP [4]. For RBPs, this sometimes may lead to different choices in the bioinformatic treatment of the data. Sequencing of the RNA sample with cross-linking of RBPs is often found in literature referred to as CLIP-Seq or HITS-CLIP [18]. Also, PAR-CLIP [17] is based on the incorporation of photoreactive ribonucleoside analogs, such as 4-thiouridine (4-SU) and 6-thioguanosine (6-SG) into nascent RNA transcripts.

Cross-linking of the RBP to RNAs is then performed by irradiation of the cells by UV light of 365 nm. The key feature of PAR-CLIP is that the cross-linked sites are revealed by thymidine to cytidine ($T \Rightarrow C$) transitions in the cDNAs prepared from immunopurified RNPs of 4-thiouridine-treated cells, and hence can be identified by comparing the sequence reads obtained to the reference genome/transcriptome sequences, and highlighting the transitions induced by cross-linking.

On the other hand, even in the presence of highly specific antibodies, IP experiments tend to be noisy (*see Note 1*). That is, in theory nucleotide sequences isolated from the IP should be only those that were bound by the DNA- or RNA-binding protein, and thus, the only ones that were sequenced. But, in practice, IP'ed samples contain a variable but sizable fraction of spurious sequences that are due to a number of different factors, from poor specificity of the antibody employed, to contamination from DNA of the lab technician performing the experiment. It is thus common practice to perform in parallel with the IP a control experiment, aimed exactly at extracting only aspecific DNA or RNA reproducing experimental noise. Different strategies can be employed in the design of a control. Ideally, one should perform exactly the same IP with the same antibody on the same cell line or tissue, but where the protein studied is not expressed, that is, the corresponding gene has been knocked out or down. Since in most of the cases this approach is unfeasible, another idea is to perform an IP using an antibody recognizing a protein not present in the cells studied. Mouse or rabbit anti-IgG antibodies are often employed for this task. Enriched DNA or RNA sequences will be then determined by comparing enrichment in the IP'ed sample against enrichment in the control sample (*see Note 2*) [7].

3.2 Bioinformatic Analysis of RIP-Seq Data

Regarding the bioinformatic analysis of sequences derived from a RIP-Seq experiment, we are still far from having a de facto standard, and just a handful of methods have been published so far. On the other hand, as discussed in the previous sections, a RIP-Seq (or CLIP-Seq or PAR-CLIP) experiment shares many similarities to either full transcriptome sequencing (RNA-Seq)—since what is sequenced is a subset of the transcriptome itself—or sequencing applied to IP'ed chromatin (DNA bound by a protein, ChIP-Seq), with the difference that IP'ed sequences in this case are whole RNAs. Since for these two fields of NGS application literature for tools and methods is booming, one possibility is to treat RIP-Seq sequences as if they were derived from a RNA-Seq or ChIP-Seq experiment, and apply to data produced the methods devised for the latter (*see Note 3*).

Some general principles, however, remain the same, and regardless of how they were isolated RNAs retrotranscribed into cDNAs that have to be sequenced are fragmented at random

(either before or after retro-transcription). Then, fragments of suitable size (usually 200 bp) are selected, sequenced at either (single-end) or both (paired-end) strands producing sequence reads of 50–100 bps. At this point, the rationale is simple: once each sequence read has been assigned to its original transcript or gene, then the abundance of the transcript in an RNA sample should be proportional to the number of reads derived from it in the sample, normalized by the RNA length. “Enrichment” for a transcript or gene in an IP’ed sample can therefore be evaluated by comparing the normalized count of sequence reads derived from it in the IP with the read count obtained in the control experiment, and by evaluating whether the difference in abundance can be considered to be “significant” and not the effect of random fluctuations of the data [7].

3.3 Sequencing RNAs: Mapping Against the Genome

As with most NGS applications for which the genome of the organism studied is available, the first step of the analysis consists on the mapping of sequence reads on the genome, that is, finding on the genome the corresponding sequence region or, in case of RNAs, the one that had been transcribed in the RNA itself. The sequence read length currently available allows for unambiguous mapping of nearly all the sequence reads not coming from repetitive DNA regions, and sequence quality is good enough to allow at most two or three substitutions per read.

Another aspect that has to be kept in mind is that if mature RNAs are sequenced, a sizable fraction of the sequence reads obtained will cover an exon–exon junction on the RNA. Hence, mapping reads of this kind without allowing for insertions on the genome will not find any match, since the alignment of the read would have to be split by the intervening intron(s), as shown in Fig. 1. However, fast and reliable sequence mapping tools for

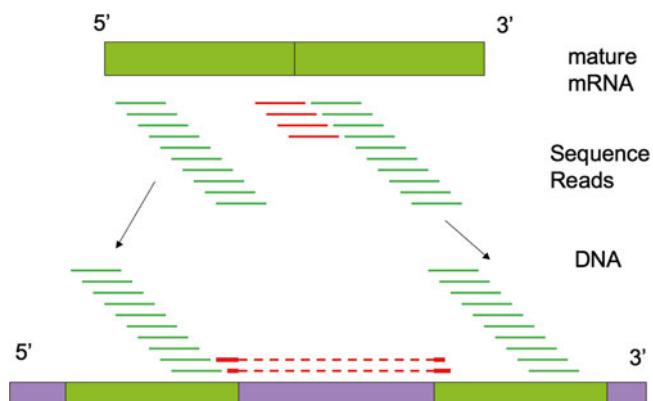


Fig. 1 Result of the mapping of RNA-Seq reads derived from mature mRNAs on the genome. Reads that covered exon–exon junctions will be split on the genome, and their mapping will encompass the intervening intron

RNA-Seq experiments have been designed explicitly taking into account this fact, such as TopHat and GSNAp. Tools of this kind perform first a mapping without insertions on the genome, identifying candidate “exons” corresponding to genomic regions covered by the mapped reads. Then, they align those reads that were not mapped at the first round trying to split them on the candidate exons identified at the first step. Further details can be found in the RNA-Seq chapter of this volume. The mapping strategy to be used (with or without insertions) depends on the RBP studied, since some bind mature RNAs while others, like splicing factors, pre-(m)RNAs. In case of doubt, we anyway advise the reader to employ TopHat or GSNAp or a similar tool allowing for the mapping of reads spanning exon-exon junctions.

The reconstruction of exons and transcripts from the mapping of the reads on the genome is not essential for organisms with a reference gene annotation complete with splicing isoforms, or if no genome but only a collection of transcript sequences is available. In cases like these, the straightforward choice is to map the sequence reads directly against the transcriptome, allowing only substitutions, with tools like Bowtie. In this case reads mapping to multiple positions (usually corresponding to different transcripts) should not be discarded, since the different transcripts usually correspond to isoforms of the same gene sharing some exons.

3.4 RIP-Seq: Using Read Counts

Starting from the considerations outlined in the previous sections, at the end of the mapping phase the mapping coordinates of each sequence read can be crossed with the genomic coordinates of exons, and of the respective transcripts or genes to which they belong. This step, which can be performed by simple in-house developed scripts, or by using a module contained in more general-purpose analysis tools, yields the basic information needed to assess enrichment, that is, a “read count” associated with each gene or transcript annotated on the genome, in turn proportional to the enrichment of the transcript in the samples sequenced. At this point, if a control experiment is available, a simple but effective strategy we implemented with good results [19], borrowing from the analysis of SAGE data [20], evaluates differential enrichment for a gene or transcript with respect to the control by using a 2×2 Chi-Square test. This kind of test is able to compare the relative enrichment of the gene or transcript across the two experiments, that is, it evaluates whether the relative enrichment of the gene differs, and if enough reads are associated to the gene to assume significant difference and not a random fluctuation of the data deriving from low read counts. Since a separate statistical test is performed on each gene or transcript, suitable corrections for multiple testing, like Bonferroni or Benjamini–Hochberg, should be then applied to the resulting p -values, yielding family-wise error rates and false discovery rates.

3.5 Treating RIP-Seq as RNA-Seq

RNA-Seq, that is, sequencing of whole transcriptomes, has become the de facto standard also for expression studies, aimed not only at the reconstruction of the repertoire of alternative splicing and RNAs expressed by a given cell line or tissue but more simply, given an existing gene/transcript annotation, at the quantification of the transcript level of genes, possibly at the single isoform level. The application of RNA-Seq has boomed in the last few years, and several different approaches have been introduced for its analysis. For example, if a reliable gene annotation is available, DESeq identifies differentially expressed genes starting from read counts associated with them, with an approach similar to the one described in the previous section. Instead of a Chi-Square test a negative binomial distribution is employed, and mean and variance of read counts are linked by local regression. DEXSeq performs similar computations at the single exon level, in order to identify differentially enriched alternative splicings. RSEM starts from the mapping of reads against a reference transcriptome, and outputs for each gene or transcript of the annotation abundance estimates (including estimation of the relative abundance of alternative transcripts of the same gene), 95 % credibility intervals, and visualization files. Differentially enriched genes or isoforms can be in turn identified through the EBSeq package included in the RSEM software distribution.

Also, the Cufflinks pipeline is currently one of the most widely used approaches in RNA-Seq analysis. Without delving into the details, starting from the mapping of sequence reads against the genomes, an estimate of the transcript level of each gene, and of each annotated isoform of each gene can be computed, expressed as Fragments Per Kilobase of exon per Million fragments mapped (FPKM), a measure which is already normalized with respect to the transcript length and the overall number of sequence reads produced by an experiment. FPKM values associated with a given gene or transcript can be compared across two or more samples, and tested for significant differences denoting “under-” or “over-expression” in the different conditions. Cufflinks returns plenty of additional information, together with the FPKM values, like the raw read counts associated with genes and transcripts, estimates of the statistical significance of difference in gene (or isoform) expression, or alternative promoter usage.

Regardless of the approach, it is straightforward to see how methods for RNA-Seq analysis can be applied to a sample extracted by RIP-Seq. The reliability of the results depends on the distribution of the reads that were produced in the RIP-Seq sample, that is, how the RBP-bound RNAs are enriched with respect to the control, which in turn depends on the specificity of the antibody employed. Also, RNA-Seq analysis tools need a suitable sequencing depth, and at least two replicates, to calculate the significance of enrichment in a reliable way. An interesting feature of this kind of analysis is that the FPKM (or analogous) values are computed at

the single transcript and isoform level, that is, return an abundance estimate and differential expression for each of the alternative transcripts of a gene. Thus, this might be of great interest if the RBP studied is for example a splicing factor, or it is known to or suspected to bind only some specific alternatively spliced transcripts.

3.6 Treating RIP-Seq as ChIP-Seq

Instead of RNAs, Chromatin Immunoprecipitation (ChIP) experiments permit to isolate DNA fragments bound by a protein of interest, for example a transcription factor or a histone carrying a given post-translational modification. The fragments that undergo sequencing are usually of 200–300 base pairs, and each can be sequenced at either 5' end of the double strand. Thus, once mapped onto the genome, sequence reads obtained in this way tend to define the boundaries of the actual binding regions of the protein. That is, the binding site is defined by a cluster of reads mapping on the forward strand of the genome within a few base pairs from one another, followed by another cluster mapping in the same way on the antisense strand about 200–300 base pairs downstream, that is, the two clusters mark the 5' ends of the original DNA fragments extracted from the IP (*see* Fig. 2).

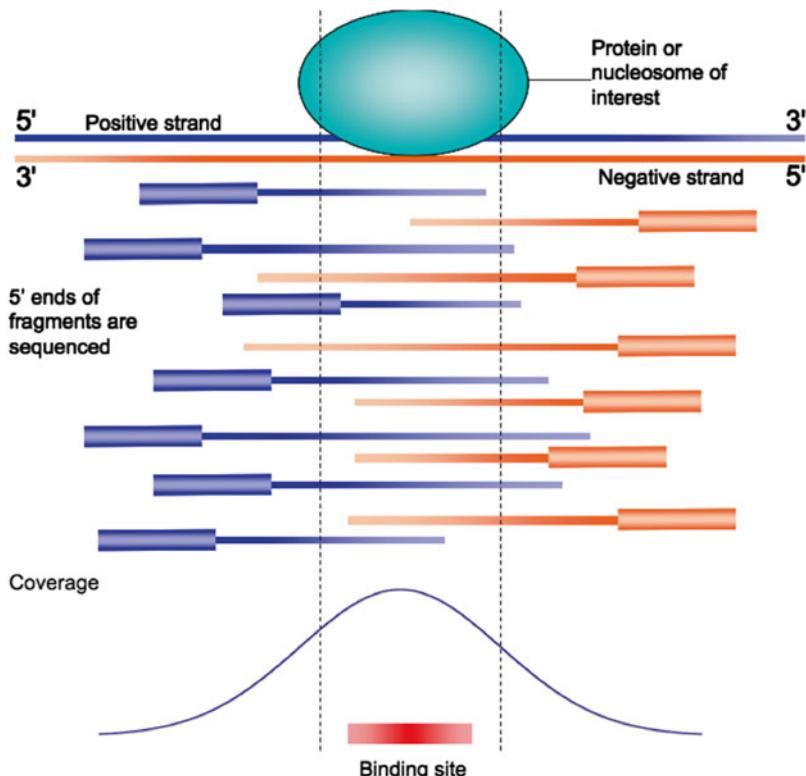


Fig. 2 A typical ChIP-Seq enrichment profile corresponding to a DNA bound region. The actual protein-binding site is usually located in correspondence to a local maximum (peak) of the sequencing coverage plot in the region

“Peak calling” algorithms, which identify DNA regions significantly enriched, differ in the strategies used to assess enrichment (over a control experiment), but nearly every single tool takes advantage of the bimodality of sequence read enrichment on the genome in correspondence of binding sites [7]. In RIP-Seq this feature is not present, and instead of presenting a well-defined area of enrichment as in ChIP-Seq corresponding to the protein binding site, read enrichment on the genome is spread all over the exons of bound RNAs, with all the transcribed region that should be enriched with respect to control. Thus, an attempt can be made nevertheless, but perhaps methods aimed at identifying broader regions of enrichment [21] are more suitable than “peak-callers” fine-tuned and optimized for the detection of isolate peaks corresponding to transcription factor binding, like MACS [22].

3.7 RIPSeeker

A recent tool that has been explicitly designed for RIP-Seq data analysis is RIPSeeker, which however shares some similarities with other ChIP-Seq analysis methods, especially those developed to identify regions enriched for histone modifications. Once again, the starting point of the analysis is the mapping of sequence reads on the genome performed with TopHat, allowing for read split mapping across different exons. The genome is then in turn divided into “bins” of a given size, and the read count in each bin is computed. The general idea is to first identify candidate RBP-bound enriched bins by training a hidden Markov model (HMM) with an Expectation Maximization procedure. Thus, the trained HMM yields for each bin the probability of being a “RBP-bound region,” or not. Once the HMM has been trained, the genome is scanned with the HMM itself, yielding the Viterbi (highest likelihood) partition of the genome into “RBP-bound” or “not bound” regions given by merging of consecutive “bound” or “unbound” bins. Comparison to other methods for RNA-Seq or ChIP-Seq seems to demonstrate the advantages offered by this tool. As for other methods for NGS data processing, the usage is quite straightforward, needing as input only the result of the read mapping, with most of the needed parameters calculated automatically (like bin size) or set to suitable default values.

3.8 PARalyzer

As already described, cross-linking performed in PAR-CLIP experiments induces a T=>C transition in the sequenced cDNAs. Thus, bioinformatic analysis can exploit this feature in the identification of the bound RNAs. PARalyzer is a tool devised for this task. Reads are first aligned to the genome, with up to two mismatches restricted to T to C conversions. Reads overlapping on the genome by at least a single nucleotide are grouped together. Within each of these read-groups, PARalyzer generates two smoothed kernel density estimates, one for T to C transitions and one for non-transition events. PARalyzer reports as interaction sites read groups

whose nucleotides maintain a minimum read depth (i.e., the nucleotides have to appear in at least a minimum number of reads), and where the likelihood of T to C conversion is higher than non-conversion (that is, a significant number of T's in a given base pair of the genome are converted to C's in sequenced reads). It should be noticed that in this case it is possible to infer also the most likely binding site positions on the RNA, which should correspond to the T=>C conversions, rather than only determining which are the RNAs bound by the protein studied.

4 Notes

1. The noisiness of IP experiments, either ChIP or RIP, makes hard or often impossible to obtain final and clear results by using just a single sequencing sample, even if compared to a control. Indeed, the usual situation is that many “interesting” targets for the protein studied fall into a sort of “gray area”, for which the significance of the enrichment changes for example according to the statistical framework employed or simply the *p*-value and FDR thresholds chosen.
2. It is common practice nowadays to perform replicates of an experiment, perform the bioinformatic analysis of the data on each replicate, and merge the results at the end. Biological replicates account for variability in the IP experiment, while technical replicates re-sequence the same starting material and account for variability and error in sequencing. While the sequencing steps can be nowadays considered to be reliable, at least for RNA or IP sequencing, biological replicates are useful to identify cases in which “something went wrong” in the IP. Our advice is, given also the decreasing cost of sequencing experiments, and the possibility to multiplex different samples on the same sequencing lane, to perform at least three biological replicates. The final merging of the results can be then performed according to different criteria, e.g., pooling together the read counts and/or FKPM values of the replicates and performing significance tests on these values, or process the replicates separately with a “majority vote” at the end (a given gene, transcript, or region has to pass significance thresholds in two experiments out of three). The latter strategy is more suitable in cases where one replicate shows little correlation to the others.
3. The best results can be obtained by applying more than one single analysis tool, like for example raw read count comparison, together with the FPKM analysis of Cufflinks, and a RIP-Seq specific method like RIPseeker. A critical comparison of the results, supported by experimental validation, will highlight the key features of the experiment performed and of the RBP investigated.

References

1. Horner DS, Pavesi G, Castrignanò T et al (2010) Bioinformatics approaches for genomics and post genomics applications of next-generation sequencing. *Brief Bioinform* 11(2):181–197
2. Marioni JC, Mason CE, Mane SM et al (2008) RNA-seq: an assessment of technical reproducibility and comparison with gene expression arrays. *Genome Res* 18(9):1509–1517
3. Glisovic T, Bachorik JL, Yong J et al (2008) RNA-binding proteins and post-transcriptional gene regulation. *FEBS Lett* 582(14):1977–1986
4. Collas P (2010) The current state of chromatin immunoprecipitation. *Mol Biotechnol* 45(1):87–100
5. Furey TS (2012) ChIP-seq and beyond: new and improved methodologies to detect and characterize protein-DNA interactions. *Nat Rev Genet* 13(12):840–852
6. Yuan J, Muljo SA (2013) Exploring the RNA world in hematopoietic cells through the lens of RNA-binding proteins. *Immunol Rev* 253(1):290–303
7. Pepke S, Wold B, Mortazavi A (2009) Computation for ChIP-seq and RNA-seq studies. *Nat Methods* 6(11 Suppl):S22–S32
8. Trapnell C, Pachter L, Salzberg SL (2009) TopHat: discovering splice junctions with RNA-Seq. *Bioinformatics* 25(9):1105–1111
9. Langmead B, Trapnell C, Pop M, Salzberg SL (2009) Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biol* 10(3):R25
10. Wu TD, Nacu S (2010) Fast and SNP-tolerant detection of complex variants and splicing in short reads. *Bioinformatics* 26(7):873–881
11. Li B, Dewey CN (2011) RSEM: accurate transcript quantification from RNA-Seq data with or without a reference genome. *BMC Bioinform* 12:323
12. Leng N, Dawson JA, Thomson JA, Ruotti V, Rissman AI, Smits BM, Haag JD, Gould MN, Stewart RM, Kendziorski C (2013) EBSeq: an empirical Bayes hierarchical model for inference in RNA-seq experiments. *Bioinformatics* 29(8):1035–1043
13. Anders S, Huber W (2010) Differential expression analysis for sequence count data. *Genome Biol* 11(10):R106
14. Anders S, Reyes A, Huber W (2012) Detecting differential usage of exons from RNA-seq data. *Genome Res* 22(10):2008–2017
15. Trapnell C, Roberts A, Goff L et al (2012) Differential gene and transcript expression analysis of RNA-seq experiments with TopHat and Cufflinks. *Nat Protoc* 7(3):562–578
16. Li Y, Zhao DY, Greenblatt JF (2013) RIPSeeker: a statistical package for identifying protein-associated transcripts from RIP-seq experiments. *Nucleic Acids Res* 41(8):e94
17. Corcoran DL, Georgiev S, Mukherjee N et al (2011) PARalyzer: definition of RNA binding sites from PAR-CLIP short-read sequence data. *Genome Biol* 12(8):R79
18. Licatalosi DD, Mele A, Fak JJ et al (2008) HITS-CLIP yields genome-wide insights into brain alternative RNA processing. *Nature* 456(7221):464–469
19. Mihailovic M, Wurth L, Zambelli F et al (2012) Widespread generation of alternative UTRs contributes to sex-specific RNA binding by UNR. *RNA* 18(1):53–64
20. Michiels EM, Oussoren E, Van Groenigen M et al (1999) Genes differentially expressed in medulloblastoma and fetal brain. *Physiol Genomics* 1(2):83–91
21. Micsinai M, Parisi F, Strino F et al (2012) Picking ChIP-seq peak detectors for analyzing chromatin modification experiments. *Nucleic Acids Res* 40(9):e70
22. Zhang Y, Liu T, Meyer CA et al (2008) Model-based analysis of ChIP-Seq (MACS). *Genome Biol* 9(9):R137

Part III

Web Resources for RNA Data Analysis

Chapter 19

The ViennaRNA Web Services

Andreas R. Gruber, Stephan H. Bernhart, and Ronny Lorenz

Abstract

The ViennaRNA package is a widely used collection of programs for thermodynamic RNA secondary structure prediction. Over the years, many additional tools have been developed building on the core programs of the package to also address issues related to noncoding RNA detection, RNA folding kinetics, or efficient sequence design considering RNA-RNA hybridizations. The ViennaRNA web services provide easy and user-friendly web access to these tools. This chapter describes how to use this online platform to perform tasks such as prediction of minimum free energy structures, prediction of RNA-RNA hybrids, or noncoding RNA detection. The ViennaRNA web services can be used free of charge and can be accessed via <http://rna.tbi.univie.ac.at>.

Key words RNA secondary structure prediction, RNA-RNA interaction prediction, Noncoding RNA detection, Sequence design

1 Introduction

The ViennaRNA package started as a small collection of tools aiming to provide fast implementations of algorithms for prediction and comparison of RNA secondary structures [1]. Over the years, the collection of algorithms and tools has substantially grown [2], now providing a comprehensive set of programs for thermodynamic RNA secondary structure prediction, including programs for efficient prediction of RNA-RNA interactions. In order to open this program package to a broad community, the ViennaRNA group provided web interfaces to the most commonly used tools early on [3]. In 2008, the original ViennaRNA secondary structure server was replaced by a revamped server attributing to recent developments in web programming and offering web interfaces to an extended set of tools [4].

In its current form, the ViennaRNA web services (<http://rna.tbi.univie.ac.at>) offer easy and user-friendly access to a large variety of computational tools useful to the RNA community. Services can basically be grouped into five categories as illustrated in Fig. 1.

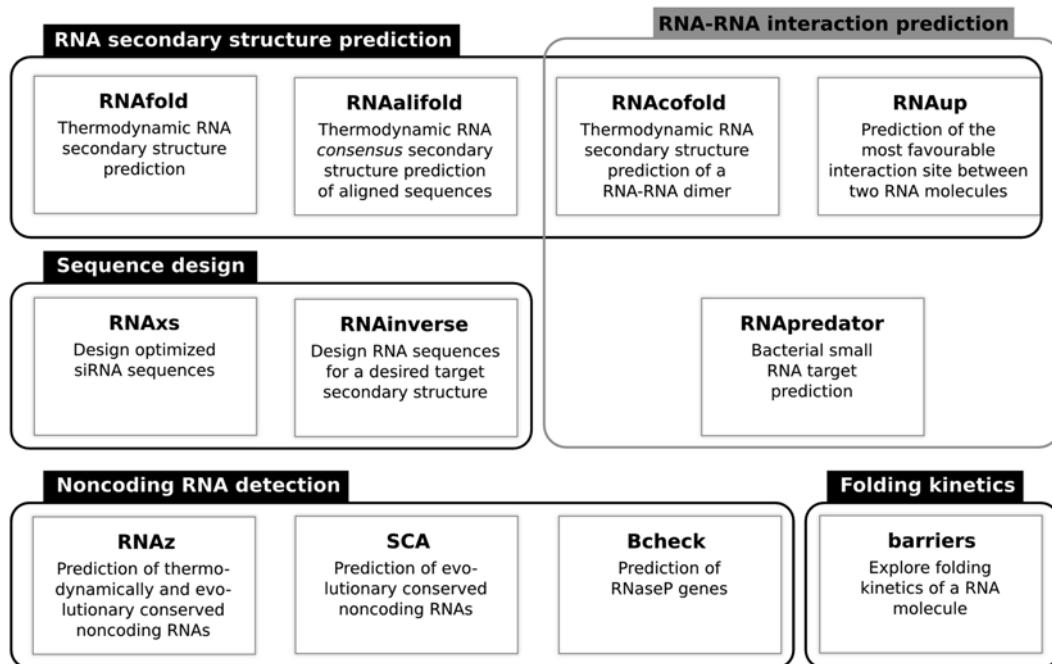


Fig. 1 Outline of the ViennaRNA web services. Based on their functionalities, programs are grouped into five categories

The core set of web services consists of tools for thermodynamic RNA secondary structure prediction including the prediction of RNA-RNA interactions. We briefly outline individual tools below and give detailed descriptions of the programs in Subheading 3.

The most highly accessed service is the RNAfold server. It allows users to determine the minimum free energy (MFE) RNA secondary structure of a single DNA or RNA sequence, while the RNAalifold server predicts the consensus structure of a set of aligned sequences [5, 6]. Exploring folding kinetics of an RNA molecule of interest can easily be accomplished with the barriers server [7]. For prediction of RNA-RNA interactions, the ViennaRNA web services offer access to several tools implementing different algorithms to address this complex problem. RNAcofold computes the secondary structure of an RNA-RNA dimer by a simple algorithm that concatenates the two sequences [8]. On the contrary, RNAup first computes the energy needed to open existing secondary structures in both molecules and then reports the single most favorable interaction site [9]. RNApredator tackles a problem of particular interest to the bacterial community, namely, the genome-wide prediction of RNA molecules targeted by bacterial small RNAs mediated by RNA-RNA hybridization [10]. On the side of sequence design, RNAinverse is an algorithm that aims to find RNA sequences that can adopt a desired RNA secondary

structure, while RNAXs is an efficient tool for the design of optimal siRNA sequences based on a set of target sequences [11]. The ViennaRNA group was among the first to develop efficient tools for prediction of noncoding RNAs [12, 13]. For the purpose of noncoding RNA gene finding, the SCA server screens alignments of RNA sequences for signs of evolutionarily conserved secondary structures [14]. The RNAz approach is more sophisticated, employing a machine learning approach to predict noncoding RNAs based on both thermodynamic stability and evolutionary conservation [13, 15, 16]. With Bcheck, the ViennaRNA web services also offer a specialized tool for RNaseP gene finding [17].

2 Materials

2.1 Hardware and Software Requirements

Use of the ViennaRNA web services does not require installation of particular software other than a common web browser that supports JavaScript. We recommend, however, to use modern web browsers such as Mozilla Firefox, Google Chrome, or Safari since these browsers natively support rendering of image files in SVG format and do not require installation of additional plug-ins.

2.2 Input Data Formats

Tools from the ViennaRNA web services require input of DNA or RNA sequences in either FASTA or ClustalW format, two widely used formats for storing sequence information. The user can paste sequences into the corresponding field or upload files via a file upload dialog. Unless DNA thermodynamic parameters are specifically selected, sequences will automatically be converted to RNA sequences (replacing T residues by U) upon submission of the job. The programs of the ViennaRNA package do not have any intrinsic restriction on the length or number of sequences that can be processed; certain restrictions do apply for the web services, though. Limitations have been chosen gracefully and are stated on the web pages of the individual services.

3 Methods

3.1 Secondary Structure Prediction with the RNAfold Web Server

3.1.1 Description and Scope of the Program

The RNAfold server offers an interface to the most basic secondary structure prediction program within the ViennaRNA web services. In short, the RNAfold algorithm uses a set of thermodynamic energy parameters in a dynamic programming framework to calculate the minimum free energy (MFE) and its associated secondary structure of a single RNA molecule. In addition, the partition function of the equilibrium ensemble of all secondary structures is computed which in turn allows to infer base pair probabilities and reliability measures for the MFE prediction.

3.1.2 Description of Input Data and Program Options

The input sequence for the RNAfold web server can be uploaded from the client computer as FASTA file or simply typed/pasted into the input field of the web interface. Without changing the default options, the server computes the MFE and partition function for the ensemble of canonical structures, i.e., structures without helices of size 1. However, structures having such isolated base pairs can be included in the prediction by unchecking the “avoid isolated base pairs” option. Since a weak GU pair at the end of a helix is likely to unfold, especially if the helix branches off a large loop, exclusion of such conformations usually yields more stable structures. Therefore, for some predictions, it might be a good idea to use the appropriate checkbox in the basic options. The auxiliary constraint folding feature of RNAfold allows for an even bigger restriction of the secondary structure space. Here, a pseudo-dot-bracket notation consisting of structure constraint symbols allows to specify whether a nucleotide must be unpaired (symbol: x), should be paired with another unspecified base (symbols “|”, “<”, “>”), or has a predefined pairing partner (symbols “(”, “)”).

It has to be noted that the web server allows for MFE prediction only; however, relying on this single representative of the ground state may be misleading. Unless the sequence length is very small, the partition function and the ensemble properties derived from the partition function (1) provide a reliability measure of the prediction and (2) take into account that an RNA molecule in thermodynamic equilibrium is in constant flux between several distinct low-energy structures and the ground state.

3.1.3 Description of the Program Output

Upon successful computation of the results, the comprehensive prediction results page not only displays the thermodynamic properties of the sequence but also comprises an interactive graphical representation of the predicted secondary structure. An example of the results page is given in Fig. 2.

The top of the results page lists the minimum free energy and its associated secondary structure in dot-bracket notation and ensemble properties like the Gibbs free energy of the ensemble, frequency of the MFE structure within the ensemble, and the ensemble diversity. Along with the MFE structure, an additional representative of the equilibrium ensemble is given by the centroid structure, i.e., the structure with the smallest distance to the Boltzmann-weighted set of all structures. By definition, this structure comprises all base pairs with a probability $p_{ij} > 0.5$. While the frequency of the MFE structure reflects its equilibrium probability, the ensemble diversity gives information about the folding potential of the nucleic acid sequence. Lower values indicate that the ensemble is dominated by a single, low free energy structure, whereas several diverse structures with almost equal stability give rise to a larger value. A complete picture of the pairing potential is given in the dot plot which depicts the base pair probabilities in the

Results for minimum free energy prediction

The optimal secondary structure in dot-bracket notation with a minimum free energy of **-28.00** kcal/mol is given below.

[color by base-pairing probability | color by positional entropy | no coloring]

```
1      GGGCUAUUAGCUCAGUUGGUUAGAGCGCACCCUGUAAGGGUGAGGUCGCUGAUUCAGCAUAGCCA
1      ((((((((.....))))).((((.....))))....((((.....))))))).
```

You can download the minimum free energy (MFE) structure in [[Vienna Format](#) | [Ct Format](#)]

You can get thermodynamic details on this structure by submitting to our [RNAAeval](#) web server.

Results for thermodynamic ensemble prediction

The free energy of the thermodynamic ensemble is **-29.19** kcal/mol.

The frequency of the MFE structure in the ensemble is **14.53** %.

The ensemble diversity is **18.09**.

You may look at the **dot plot** containing the base pair probabilities [[EPS](#) | [PDF](#) | [IMAGE CONVERTER](#)].

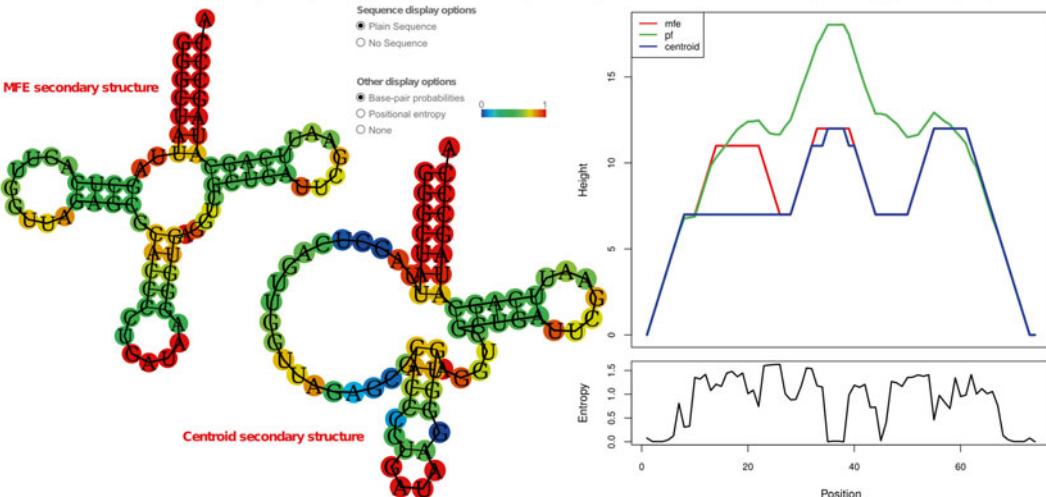
The centroid secondary structure in dot-bracket notation with a minimum free energy of **-17.40** kcal/mol is given below.

[color by base-pairing probability | color by positional entropy | no coloring]

```
1      GGGCUAUUAGCUCAGUUGGUUAGAGCGCACCCUGUAAGGGUGAGGUCGCUGAUUCAGCAUAGCCA
1      (((((.....((((.....))))....((((.....))))))).
```

Graphical output

The structure below is colored by base-pairing probabilities. For unpaired regions the color denotes the probability of being unpaired.



An equivalent RNAfold command line call would have been

```
RNAfold -p -d2 --noLP < test_sequenc.fa > test_sequenc.out
```

Fig. 2 Sample RNAfold output depicting the minimum free energy secondary structure, the centroid structure, and graphical representations thereof

form of a matrix representation. Here, the probabilities are indicated in the upper triangle of the matrix by a black square with its size proportional to the pairing probability. In contrast to this ensemble-wide insight, the lower half of the matrix reflects the base pairs present in the MFE structure. The dot plot can be either viewed/downloaded as PDF or PostScript file or be converted to a bitmap image format. It is worthwhile to note that the PostScript image is a simple text file from which the numerical values of the base pair probabilities can be extracted quite easily. While the dot plot representation is information rich, it may be difficult for an

untrained user to easily draw conclusions from it. Therefore, the structure representatives given in the results can be annotated by the base pair probability or positional entropy by a color code ranging from blue to red indicating low and high values, respectively. For the probability annotation, base pairs in the dot-bracket string are coded by their respective equilibrium probabilities, while unpaired nucleotides are coded by their probability of being unpaired. Thus, structure parts being coded by red color are more reliable than those with blue or green color. A similar measure, which, however, takes all pairing possibilities of a nucleotide into account, is the positional entropy. Again, red color indicates high reliability of the prediction, whereas blue or green colors indicate parts of the structure with low reliability.

The default graphical output consists of two interactive secondary structure plots of both structure representatives as well as a mountain plot which subsumes all thermodynamic data of the prediction. While the user can select the type of color-coded data in the interactive plots, each graphical representation can also be downloaded in either vector graphic format such as PDF or PostScript or as a bitmap graphic via the image converter. Finally, the mountain plot superpositions MFE structure, centroid structure, and base pair probabilities in a single plot. In this two-dimensional plot, the *x*-axis displays the sequence position while the *y*-axis, the height of the mountain, is given by the number of base pairs enclosing the respective position. This means that for the MFE and centroid structure, plateaus represent unpaired nucleotides, i.e., loops, whereas slopes represent the 5' or 3' side of helices, depending on whether they face upward or downward, respectively. Additionally, the mountain plot is accompanied by a plot of the positional entropy.

3.2 Prediction of Consensus Secondary Structures with the RNAalifold Web Server

3.2.1 Description and Scope of the Program

For certain classes of functional RNA molecules, secondary structure elements are often evolutionarily more conserved than the underlying primary sequence. The accuracy of single-sequence RNA secondary structure prediction tools is limited, irrespective of the approach or method employed [18]. Consensus structure prediction of a set of (aligned) sequences is thus a commonly used way to increase the predictive power of structure prediction tools and to spot evolutionarily conserved, often functional structural motifs. In addition to thermodynamic folding, consensus structure prediction relies on signals of sequence variation. In particular, consistent mutations (one base in a base pair is changed with respect to a reference sequence) and compensatory mutations (both bases are changed, e.g., a G-C base pair in one sequence and an A-U base pair in the other) are counted as supporting evidence for a conserved secondary structure, while base pairs that can only be formed in a single or few sequences contradict the formation of a conserved structural motif.

RNAalifold follows the align first, fold later strategy of consensus structure prediction. That means it takes a multiple sequence alignment as input and predicts the consensus structure by minimizing a pseudo-free energy composed of the mean free energy of the structures of the single sequences and the number of consistent and compensatory mutations [6].

3.2.2 Description of Input Data and Program Options

The input for RNAalifold consists of multiple sequence alignments in FASTA or ClustalW format. Information on the phylogenetic tree underlying the alignment is currently not considered by the algorithm. It must be noted that the quality of the structure prediction is directly influenced by the quality of the alignment. For average pairwise sequence identities above 75 %, alignments generated by simple sequence-only-based alignment tools such as ClustalW (<http://www.ebi.ac.uk/Tools/msa/clustalw2>) are sufficient for good results. For a set of highly divergent sequences, we recommend using an alignment program that considers both sequence and secondary structures such as LocARNA (<http://rna.informatik.uni-freiburg.de/LocARNA>).

RNAalifold has two ways of scoring evolutionary conservation of base pairs. The first approach simply counts occurrences of canonical base pairs and counterexamples for each base pair. The algorithm then only considers base pairs with less than three counterexamples. The other scoring scheme uses RIBOSUM scores, which are substitution matrices for RNA sequences, to compute an alignment-like score for each possible base pair. The main advantage of the RIBOSUM score is that it does not penalize counterexamples as harshly as the simple scoring scheme. For alignments with many sequences, this feature will allow RNAalifold to infer a consensus structure even if there are some counterexamples, possibly due to faulty alignment choices. The new version of RNAalifold [6] also introduced an alternate way of dealing with gap characters in the aligned sequences, with gaps not contributing to the energy computation. We recommend the default setting which is the new RNAalifold version with RIBOSUM scoring, but other variants can be chosen as well.

Many of the parameters of RNAfold apply to RNAalifold as well, but there are two parameters unique to RNAalifold. The parameter “Weight of covariance term” controls the relative weighting of energy contributions to contributions from structural conservation supporting base pairs. The default value of 1 implies equal contribution, while a value below 1 will down-weigh the conservation supporting score.

On the other side, “Penalty for non-compatible sequences” controls the impact of counterexamples. If this value is set to 0, non-supporting base pairs will not be penalized. Both parameters are accessible via the advanced options field.

3.2.3 Description of the Program Output

The RNAalifold output page provides the consensus secondary structure in dot-bracket notation and the computed minimum free energy, which actually is a pseudo-energy with contributions coming from both sequence-averaged free energy minimization and base pair conservation bonus energies. Each of the individual contributions is listed separately. The textual representation of the secondary structure is assisted by graphical output in the form of a structure-annotated alignment and a visualization of the secondary structure highlighting consistent and compensatory mutations (Fig. 3).

3.3 RNA-RNA Interaction Prediction: RNACofold

3.3.1 Description and Scope of the Program

RNAcofold is an algorithm for prediction of the hybrid structure formed between two RNA molecules [8]. The approach is simple: it concatenates the two sequences and computes the MFE structure as if it was for a single sequence. The only difference to single-molecule secondary structure prediction is that all structural features containing the end of the first and the start of the second molecule (called “cut point”) are treated as exterior loops. The approach is versatile enough to predict multiple interaction sites, while it is fast enough to be computed for molecules of several thousand nucleotides.

The drawback of this approach is, however, that it cannot predict structures that would be considered as pseudo-knots, i.e., crossing base pairs, in the single structure case. This excludes many intermolecular structures known in nature, e.g., the kissing hairpin structural motif. On the other hand, this ansatz makes it possible to easily compute many of the features known from single-sequence folding, mainly via exploiting partition function-related information. In detail, partition function calculations can be used to predict equilibrium concentrations of the dimers dependent on some initial monomer concentrations.

3.3.2 Description of Input Data and Program Options

RNAcofold takes two RNA sequences termed as sequence A and sequence B as its input and will, by default, not only compute the partition function and base pair probabilities of the heterodimer (i.e., the structure formed by a dimer of the two input sequences, hence denoted as AB) but also of the homodimers (AA and BB), which are the structures that two molecules of sequence A or sequence B will form. Binding of two molecules is always a concentration-dependent process. Therefore, the RNAcofold server offers two ways of predicting concentrations of dimers. It either varies the monomer concentrations using predefined values or uses a list of concentrations provided by the user. In the latter case, two values separated by a white space have to be provided by line.

3.3.3 Description of the Program Output

The RNAcofold output page, by default, displays the heterodimer structure in dot-bracket notation with an “&” sign separating the two sequences. In addition, dot plots of the heterodimer structure

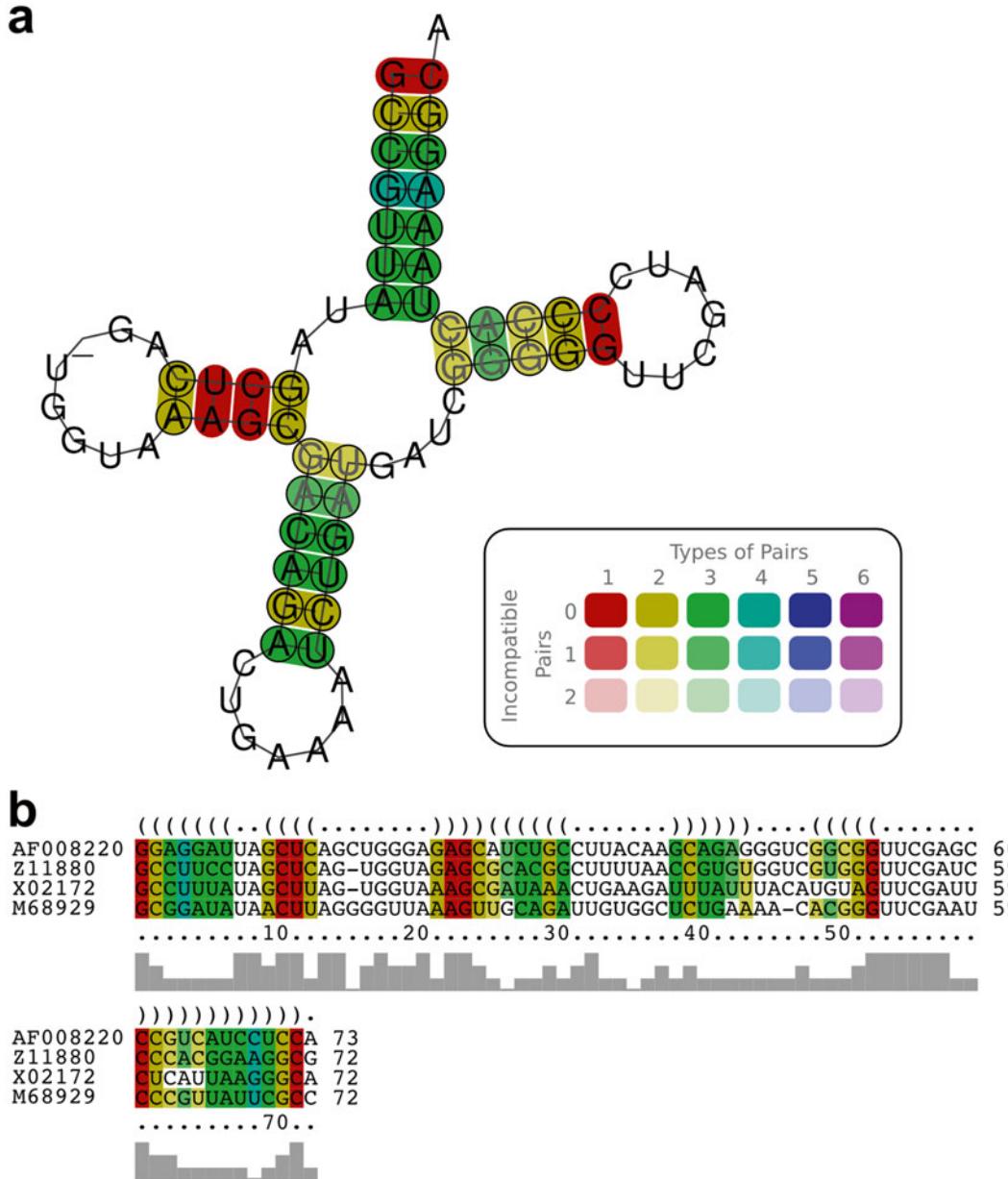


Fig. 3 Sample RNAalifold output. **(a)** Visualization of the consensus secondary structure predicted by RNAalifold. The number of different types of base pairs observed for a particular pair of positions in the alignment is color coded. *Pale colors* indicate contradicting evidence, meaning that not all sequences in the alignment can form the particular base pair. *Red* indicates base pairs that show conservation at nucleotide level in all sequences, while base pairs colored in *green*, for example, are supported by compensatory/consistent mutations; in this case, three different types of base pairs are observed. **(b)** Structure-annotated multiple sequence alignment. The same color-coding scheme as discussed above applies here

(AB) as well as the two monomer structures (A, B) and the two homodimer structures (AA, BB) are provided. In the dot plots of the dimers, a cross sign is used to indicate the cut point that separates the two molecules. The output page also contains ensemble free energies of all five species, as well as the energy that is gained by forming the heterodimer compared to the two monomers denoted as “Delta G for heterodimer binding.”

If the option “Vary concentration” was selected, concentration dependency plots showing the relative concentrations for each dimer (AA, BB, AB) and the monomers (A, B) will be generated. If the user specified concentrations, results will be provided in form of a table.

3.4 RNA-RNA Interaction Prediction: RNAup

3.4.1 Description and Scope of the Program

RNAup takes another approach for the prediction of intermolecular RNA-RNA interactions [9]. It basically models RNA-RNA interaction as a two-step process, which resembles a simplified approach of what is actually happening *in vivo* when RNA-RNA hybrids form. In the first step, the interaction sites have to be made accessible (i.e., single stranded) in order to engage in intermolecular base pairing. In the second step, the supposed interaction sites of two RNA molecules can then form base pairs between each other. The RNAup algorithm follows this model by first computing the opening energy for every sequence window of length w . Computation of interaction energies for all intervals with a maximum length of W is then accomplished in a second step. Lastly, results from these two steps are merged to obtain the total interaction energy, which is minimized to find the best site of interaction. This approach allows any form of structure around the interaction sites, thus being able to also predict for example the kissing hairpin structural motif. However, the drawback of this approach is that only a single interaction site can be taken into account. Therefore, RNAup is mainly used for the prediction of interactions of short RNAs on long target RNAs such as bacterial sRNAs on mRNAs.

3.4.2 Description of Input Data and Program Options

As RNACofold, RNAup takes two sequences as an input. Unique to RNAup are two parameters controlling aspects of the interaction prediction. The default option of the parameter “length of the unstructured region” is set to 4, meaning that RNAup will calculate the probabilities of being unpaired for stretches of 4 consecutive nucleotides. Note that via a simple transformation of these probabilities, one can also easily calculate the energy needed to open existing secondary structures. The other parameter “maximal length of the region of interaction” controls the size of the interaction region that is reported, set to 25 nt by default. RNAup reports probabilities of being unpaired by considering contributions from all loop types. In the advanced options field, the user can specifically select that, for example, also probabilities of being unpaired in a hairpin loop (H), interior loop (I), external loop (E), or multi-loop (M) are being reported.

3.4.3 Description of the Program Output

The RNAUp output page will report the location of the interaction sites in both sequences and the interaction secondary structure in dot-bracket notation. The interaction free energy is composed of multiple contributions (energy gained from duplex formation and energies needed to open existing secondary structures in the two molecules), which are separately listed. The output is also accompanied by two plots depicting the interaction free energy (red) and the energy needed to open existing secondary structure along the longer of the two sequences. The original RNAUp output text file is also provided, which lists detailed energy values for each nucleotide and, if selected, detailed energy contributions of individual loop types.

3.5 RNA Design with the RNAinverse Web Server

3.5.1 Description and Scope of the Program

While RNA secondary structure prediction is a computationally well-defined problem with the solution guaranteed to be optimal given the set of energy parameters, the inverse problem of finding a sequence that adopts a given secondary structure can only be solved in terms of heuristic approaches. RNAinverse implements such a heuristic approach of finding sequences that can fold into a given secondary structure and is thus a valuable tool for designing RNA sequences. The RNAinverse algorithm starts with a randomly selected nucleotide sequence and keeps changing nucleotides until the RNA sequence adopts the desired fold. The algorithm has two modes of operandi. In the MFE mode, RNAinverse searches for sequences that adopt the desired structure as their minimum free energy structure, while in the partition function mode, the frequency of the structure in the ensemble of potential secondary structures a sequence can adopt is maximized. While finding sequences that adopt the desired structure usually succeeds within a few iterations of the algorithm, it has to be noted that the longer and complex the desired structure is, the search becomes more difficult, and the algorithm might not be able to find a sequence meeting the criteria. In that case, the sequence with the minimal edit distance to the desired structure of the sequences generated is reported.

3.5.2 Description of Input Data and Program Options

Input for the RNAinverse server is the desired secondary structure in dot-bracket notation (*see* Subheading 3.10. for more details) and optionally an RNA sequence string of equal length. If no sequence is provided, RNAinverse will generate a random sequence. Note that when specifying the sequence, the letter “N” can be used to indicate a random base. Certain restrictions also apply to the secondary structure. A hairpin loop has to consist of at least three unpaired bases, and crossing base pairs, i.e., pseudo-knots, are not allowed. A sample input of a sequence-structure pair is given below:

```
. (((((....))))...(((....))).  
NANNNNNNNNNNUNNNNNNNNNNNNNNNNN
```

In this example, the desired secondary structure consists of two small hairpin structures with each four base pairs. The first base pair in the first stem is set to be an A-U pair. As noted before, depending on the initial (random) sequence provided, the algorithm may fail to find a secondary structure meeting the desired criteria. Instead of running RNAinverse several times, we advise the user to set the parameter “number of sequences to be generated” to a higher number than 1, thereby increasing chances that a valid sequence is found.

3.5.3 Description of the Program Output

The results page lists the generated sequences and the minimum free energy of the predicted structure. Links to the RNAfold server that automatically paste the sequence into the input field are provided for each sequence. This allows the user to explore the predicted secondary structure and folding potential of the sequences in more detail. We do recommend to perform this analysis, especially for sequences listed under “Results for thermodynamic ensemble prediction.” The procedure of optimizing the frequency in the thermodynamic ensemble may lead to deviation of the minimum free energy structure from the desired structure. When multiple sequences were generated, results are presented in sortable tables, which allows convenient sorting, e.g., by free energy.

3.6 Exploring Folding Kinetics with the Barriers Server

3.6.1 Description and Scope of the Program

Some RNA molecules are known to take a very long time or even never adopt their ground-state structure after transcription. These RNAs usually get trapped in a so-called metastable conformation, either due to co-transcriptional folding or while refolding from a denatured state. Approaches like RNAfold that consider the RNA molecule to be in thermodynamic equilibrium can therefore not be used to predict this behavior since they exclude folding kinetics by design. However, when all secondary structures an RNA molecule can adopt are known, they can be interpreted as points in a landscape with their corresponding free energy marking the elevation. A move set, e.g., inserting/removing a base pair, then defines the neighborhood of any given structure. Finally, folding kinetics is given by how RNA molecules explore this landscape which in turn can be modeled as a Markov process. Given an initial population density of the structure space, this allows to mathematically solve the problem of the population density for an arbitrary point in time. However, the number of secondary structures an RNA molecule can fold into grows exponentially with its sequence length making the solution impractical to compute even for very short RNAs of about 50 nt. Thus, a widely used approach to circumvent this limitation is to heavily shrink the state space by lumping secondary structure states together in the so-called macro states. One possibility of macro-state generation is to use barrier trees where all structures that reach the same local minimum via consecutive steepest descent moves belong to the same macro state.

Structures connecting the macro states are considered saddle points and represent the energy barriers separating the local minima. Taken together, the local minima, the saddle points, and the energy barriers define the leaves, internal nodes, and edge lengths between the nodes of a so-called barrier tree, respectively. Still, this approach relies on the enumeration of the secondary structures and thus is applicable for RNA molecules up to a length of some 100 nt only.

The barriers web server employs the programs RNAsubopt [19], barriers [7], and treekin [20] in order to generate a set of about ten million low free energy structures, compute the barrier tree and transition rate matrix for the Markov process, and solve the master equation of the system, respectively.

3.6.2 Description of Input Data and Program Options

A single RNA sequence with at most 100 nt in length may be used as input for the barriers server. In order to keep the barrier tree simple, the user may set the maximum number of leaves (default 50). Furthermore, local minima that do not exhibit an energy barrier higher than a certain user-defined threshold can be merged. This allows for keeping the number of leaves in the barrier tree small even for RNAs with high diversity of low-energy structures. The number of secondary structures generated by RNAsubopt can be limited again by omitting structures with isolated base pairs, which is the default, or removing all structures where a weak GU pair terminates a helix.

3.6.3 Description of the Program Output

Among the results of the barriers web server calculations are the number of secondary structures obtained by RNAsubopt as well as the resulting barrier tree in graphical form. Additionally, the refolding path between any of the ten lowest free energy macro states can be displayed with an animated SVG image. In this image, the energy profile in the form of a height plot and a circular and a regular structure representation are shown for each intermediate structure along the refolding path.

The folding kinetics section of the output consists of the downloadable transition rate matrix of the Markov process and four options that control the kinetic simulation. First, the initial structure, i.e., the structure that has a population density equal to 100 % at the start of the simulation, can be chosen from the ten lowest free energy macro states. The time span of the simulation, which is measured in arbitrary time steps, can be set via the “Start time” and “Stop time” values. Usually, the default values should suffice; however, for RNAs which populate a metastable state for too long, the stop time may have to be set to a value considerably higher than one million. In any case, the “Stop time” has to be set such that the plot displays macro-state number 1 with highest density at the end of the simulation. If no further changes appear in the population density over time, the system has reached equilibrium.

To keep the simulation plot clean from local minima which are sparsely populated, the treekin options allow for ignoring all macro states which do not exceed a predefined threshold.

3.7 Noncoding RNA

Gene Prediction with the RNAz Server and Related Tools

3.7.1 Description and Scope of the Program

Detection of functional RNA structures and noncoding RNA (ncRNA) genes is a challenging task. Simply folding the sequence of interest and assessing its minimum free energy is not a sufficiently good approach, since the MFE is highly dependent on the G + C content of the sequence and the length of the sequence. Nevertheless, functional RNA structures often show elevated thermodynamic stability when compared to a set of randomized sequences of the same length and base composition [13]. Moreover, functional RNAs are also often found to be evolutionarily conserved [12]. Aiming for accurate prediction of specifically the class of thermodynamically stable and evolutionarily conserved RNA structures, the ViennaRNA group developed RNAz [13, 15] and the RNAz server for easy and user-friendly access to the RNAz toolbox [16].

In detail, the RNAz approach measures thermodynamic stability in terms of a normalized *z*-score of folding energies. It indicates how many standard deviations the structure of a given sequence is more or less stable than that of random sequences of the same length and base composition. Negative *z*-scores thus indicate that a sequence is more stable than expected by chance. The second component is the evaluation of the structural conservation, which requires that the input for RNAz is an alignment of at least two sequences. Structural conservation is measured by the structure conservation index (SCI), which evaluates structural conservation indirectly through folding energies rather than on the structure level itself (*see* ref. 14 for details). In theory, a SCI above 1 indicates structural conservation supported by compensatory mutations. The SCI is, however, strongly influenced by the alignment quality with respect to secondary structures, and as a rule of thumb, a SCI close to or above the average pairwise identity of the sequences in the alignment can be considered as good. While this is valuable information to know when evaluating individual RNAz predictions in detail, we generally advise the user to assess predictions via the RNA class probability, which is calculated by support vector machine classification combining thermodynamic stability and structural conservation.

3.7.2 Description of Input Data and Program Options

Detection of functional RNA structures with the RNAz server is a multistep process. As stated before, RNAz requires multiple sequence alignments as input. Alignments can be provided in various formats such as ClustalW or MAF (<http://genome.ucsc.edu/FAQ/FAQformat.html>) but have to consist of at least two sequences and should not be less than 50 nt in length. Multiple alignments in ClustalW have to be separated by a header line starting with “CLUSTAL W.” After successful upload of the alignments to the server, the user will be redirected to the analysis options page (a screenshot is shown in Fig. 4). The RNAz version hosted at the

You have uploaded 1 alignments in CLUSTALW format.

Basic

Slice alignments longer than	200	?
Window size	120	?
Step size	40	?
Reading direction	both-strands	?
Use reference sequence	<input type="checkbox"/>	?

Filtering

Maximum fraction of gaps	0.25	?
Maximum fraction of masked letters	0.10	?
Minimum number of sequences	2	?
Minimum mean pairwise identity	50	?

Choose subset

Maximum number of sequences	6	?
Number of samples	1	?
Target mean pairwise identity	80	?

[Restore Defaults](#)

[« Back](#) [Proceed »](#)

Fig. 4 Screenshot of the RNAz analysis options page. For each field, detailed help is available by clicking on the question mark icons

server is limited to analyzing alignments with at most six sequences and a maximal alignment length of 200 nt. Longer alignments can, however, be uploaded and will be preprocessed with the parameters set on the analysis options page. In detail, long alignments are sliced into smaller alignments given a user-defined window size and step size. Default values have been proven to be adjusted well but may require to be set to different values when screening, for example, for ncRNAs of a certain size. The RNAz pipeline will also filter alignments and remove sequences that have too many repeat-masked letters (denoted by lowercase letters) or gaps. If there are more than six sequences left after filtering, optimized algorithms in the RNAz package will automatically select a set of appropriate sequences according to user-controlled values, e.g., the value specified in the target mean pairwise identity field. Finally, on the last page, the user can select between various output options and leave an e-mail address for notification once computation is completed.

3.7.3 Description of the Program Output

The RNAz output page is clearly structured providing links to detailed analyses for each alignment analyzed. In particular, for each alignment screened, the SCI, z -score, and RNA class probability are reported among other features. Each prediction is accompanied by a structure-annotated alignment and visualization of the consensus secondary structure (see Subheading 3.2.3 for detailed description of these plots). We would like to draw attention of the reader to the excellent help pages of the RNAz server, which discuss many of the analysis options and the RNAz output in great detail. In default mode, RNAz screens both the alignment in sense and antisense direction for putatively conserved functional structures. Considering the hit with the higher probability of the two screened windows is a first good choice. For a more detailed analysis, we refer the user to specialized tools such as RNAstrand ([21]; <http://www.bioinf.uni-leipzig.de/Software/RNAstrand>). Results are kept on the server for 30 days and can be accessed by the URL outlined on the results page. RNAz is also available as a stand-alone program and can be obtained from <http://www.tbi.univie.ac.at/~wash/RNAz>. For additional information on noncoding RNA gene detection, we refer the interested reader to ref. 22.

3.7.4 Related Tools for Noncoding RNA Prediction

While RNAz considers both thermodynamic stability and evolutionary conservation, the structure conservation analysis (SCA) server exclusively evaluates multiple sequence alignments with respect to conserved structural elements [14]. This is done by calculating various measures like the SCI, base pair distance, or tree-editing distances and comparing the values obtained from the alignment of choice against a set of randomized alignments generated by shuffling alignment columns while preserving gap patterns [12]. The output page features a graphical representation of the calculated scores together with z -scores and empirical p -values for statistical assessment. Note that for similarity measures like the SCI, a positive z -score indicates structural conservation, while for distance measures like the base pair distance, a negative z -score is indicative of structural conservation.

The Bcheck web server is a specialized tool for finding RNaseP genes [17]. The input simply consists of a FASTA sequence one wants to screen for a potential RNaseP gene, while the output page contains the predicted subsequence together with secondary structure annotation derived from fitting the RNaseP covariance model.

3.8 Other Services Hosted at the ViennaRNA Web Services

So far, we have described the most commonly used programs of the ViennaRNA package and tools for ncRNA detection. The ViennaRNA web services also offer access to a series of other specialized tools.

RNAPredator is a tool for prediction of target interactions of bacterial sRNAs with mRNA molecules on a genome-wide scale [10]. The algorithm makes use of RNAPlex, a fast approach for computing RNA-RNA interactions considering accessibility of the target site, which allows genome-wide screens to be performed within a few minutes. The RNAPredator output page outlines the predicted interactions in detail including enrichment evaluation in gene ontology terms.

RNAxs is an algorithm that aims at the optimal design of siRNAs targeting selected mRNA molecules [11]. Among other design criteria like seed matching, RNAxs also considers target site accessibility. For a detailed protocol on how to use the RNAxs web server, we refer the interested reader to ref. 23.

A very recent addition to the ViennaRNA web services is the CMCompare web server [24]. It allows comparison and quality assessment of Infernal RNA family models (<http://rfam.sanger.ac.uk/>). In addition to these programs, the database AREsite is also hosted on the ViennaRNA web services [25]. AREsite is an online resource for the investigation of AU-rich elements (AREs) in vertebrate mRNA UTR sequences. In addition to localization of AREs, AREsite also reports local accessibility values and evolutionary conservation profiles.

3.9 For the Advanced User: Additional Parameter Options

Interfaces to the core programs of the ViennaRNA package all have an advanced options panel. Clicking on it will expand the field, showing options to control energy parameters, dangling ends, and folding temperature. Default RNA energy parameters are those of the Turner 2004 model [26]. Usage of the Turner 1999 model is only recommended when reevaluating the folding of RNA sequences from publications that used the Turner 1999 energy model. Folding of DNA sequences is also possible, which can be done by selecting “DNA parameters” (Matthews model, 2004). Folding of DNA/RNA hybrids is currently not implemented. Turner 2004 parameters were measured at 37 °C, which is also set as default temperature. Other temperatures can be entered and energy parameters will be extrapolated accordingly, but please note that for temperatures far from 37 °C, results will increasingly become unreliable. Dangling ends are nucleotides that stack onto the ends of helices. Programs of the ViennaRNA package have several options for including dangling ends in the computations. The default setting is that dangling end energies are considered for both sides of a helix in any case. Other options are to turn off dangling end energy contributions or to extend the dangling end model to also include coaxial stacking of adjacent helices in a multi-loop. For some programs, it is also possible to specify that the RNA sequence should be treated as a circular RNA molecule. As circular RNA molecules cannot adopt all conformations of their linear counterparts due to tension in the exterior loop, a switch to correct the prediction for this case is available.

3.10 Dot-Bracket Notation and Other Formats for Storing RNA Secondary Structures

Programs of the ViennaRNA web services generate or require as input RNA secondary structures in dot-bracket notation. The dot-bracket notation, or sometimes referred to as Vienna format, is a simple string notation for RNA secondary structures. An opening bracket corresponds to the 5' base of a base pair, while a closing bracket denotes the 3' base in the base pair. An unpaired base is indicated by a dot. This notation is intuitive since it follows mathematical rules for parenthesizing. An example of the dot-bracket notation for the secondary structure of a tRNA showing the typical cloverleaf fold is given below:

```
GGGCUAUUAGCUCAGUUGGUUAGAGCGCACCCUGAUAGGGUGAGGCUGCUGAUUCGAAUUCAGCAUAGCCCA
((((((.....))))).((((.....)))).....((((.....)))))).
```

The dot-bracket notation is also widely supported by programs other than the ViennaRNA package; nevertheless, results pages of the ViennaRNA web services also provide output in ConnecT (CT) format for interchange with other programs.

3.11 Insights into Energy Contributions with RNAeval Web Server

Underlying the secondary structure prediction algorithms of the ViennaRNA package is the so-called loop-based energy model or nearest-neighbor model. The principle of this model is that any RNA secondary structure can uniquely be decomposed into a set of loops. In this model, stacked base pairs that build the helices of RNA secondary structures are also represented as loops, as a special case of interior loops with no unpaired nucleotide between two base pairs. For users interested in the energy contributions of individual structural elements to the total minimum free energy, we recommend the RNAeval server. It takes as input a sequence-structure pair and lists the individual energy contributions according to the loop-based energy model.

3.12 Obtaining High-Quality Images for Publication

Images displayed on the results pages are either in a bitmap format such as PNG or in a vector format such as SVG. We do not recommend to use images in PNG format for figures in publications, but advise the user to instead download the corresponding file in PDF or EPS format, which preserves the quality of the image when scaled. Images in these file formats can easily be postprocessed and edited with free image tools such as Inkscape. If this is not a viable option for the user, results pages also provide links named “IMAGE CONVERTER” to obtain PNG or GIF files at higher, user-specified resolutions than those displayed on the results pages.

3.13 The ViennaRNA Package: Stand-Alone Version

A stand-alone version of the ViennaRNA package can be obtained from <http://www.tbi.univie.ac.at/RNA>. The ViennaRNA package has been developed and tested under the operating system Linux and therefore is best operated under Linux or a Unix-like

environment. For most use cases, off-the-shelf hardware is sufficient. For details on installation and execution of programs, we refer the interested reader to another article in the Methods in Molecular Biology series [27]. Note that at the end of each results page there is an equivalent command line call outlined to conduct the same analysis with a local installation of the ViennaRNA package.

3.14 Related Web Services

The Nucleic Acids Research web server index (<http://www.oxford-journals.org/nar/webserver/subcat/10/90>) lists many tools for the purpose of RNA secondary structure prediction and related tasks and offers a good starting point for exploring other tools.

Among the service listed, we would like to specifically highlight the Freiburg RNA tools (<http://rna.informatik.uni-freiburg.de>), which is a comprehensive set of programs that provides among other services also excellent tools for the prediction of RNA-RNA interactions [28].

Acknowledgments

This work was supported in part by the University of Basel, the Austrian FWF project “SFB F43 RNA regulation of the transcriptome,” and the University of Leipzig.

References

- Hofacker IL, Fontana W, Stadler P, Bonhoeffer S, Tacker M, Schuster P (1994) Fast folding and comparison of RNA secondary structures. *Monatsh Chem* 125:167–188
- Lorenz R, Bernhart SH, Höner Zu Siederdissen C, Tafer H, Flamm C, Stadler PF, Hofacker IL (2011) ViennaRNA Package 2.0. *Algorithm Mol Biol* 6:26
- Hofacker IL (2003) Vienna RNA secondary structure server. *Nucleic Acids Res* 31: 3429–3431
- Gruber AR, Lorenz R, Bernhart SH, Neuböck R, Hofacker IL (2008) The Vienna RNA web-suite. *Nucleic Acids Res* 36:W70–W74
- Hofacker IL, Fekete M, Stadler PF (2002) Secondary structure prediction for aligned RNA sequences. *J Mol Biol* 319:1059–1066
- Bernhart SH, Hofacker IL, Will S, Gruber AR, Stadler PF (2008) RNAalifold: improved consensus structure prediction for RNA alignments. *BMC Bioinform* 9:474
- Flamm C, Hofacker IL, Stadler PF, Wolfinger MT (2002) Barrier trees of degenerate landscapes. *Z Phys Chem* 216:155
- Bernhart SH, Tafer H, Mückstein U, Flamm C, Stadler PF, Hofacker IL (2006) Partition function and base pairing probabilities of RNA heterodimers. *Algorithm Mol Biol* 1:3
- Mückstein U, Tafer H, Hackermüller J, Bernhart SH, Stadler PF, Hofacker IL (2006) Thermodynamics of RNA-RNA binding. *Bioinformatics* 22:1177–1182
- Eggenhofer F, Tafer H, Stadler PF, Hofacker IL (2011) RNApredator: fast accessibility-based prediction of sRNA targets. *Nucleic Acids Res* 39:W149–W154
- Tafer H, Ameres SL, Obernosterer G, Gebeshuber CA, Schroeder R, Martinez J, Hofacker IL (2008) The impact of target site accessibility on the design of effective siRNAs. *Nat Biotechnol* 26:578–583
- Washietl S, Hofacker IL (2004) Consensus folding of aligned sequences as a new measure for the detection of functional RNAs by comparative genomics. *J Mol Biol* 342:19–30
- Washietl S, Hofacker IL, Stadler PF (2005) Fast and reliable prediction of noncoding RNAs. *Proc Natl Acad Sci U S A* 102:2454–2459

14. Gruber AR, Bernhart SH, Hofacker IL, Washietl S (2008) Strategies for measuring evolutionary conservation of RNA secondary structures. *BMC Bioinform* 9:122
15. Gruber AR, Findeiß S, Washietl S, Hofacker IL, Stadler PF (2010) RNAz 2.0: improved noncoding RNA detection. *Pac Symp Biocomput* 2010:69
16. Gruber AR, Neuböck R, Hofacker IL, Washietl S (2007) The RNAz web server: prediction of thermodynamically stable and evolutionarily conserved RNA structures. *Nucleic Acids Res* 35:W335–W338
17. Yusuf D, Marz M, Stadler PF, Hofacker IL (2010) Bcheck: a wrapper tool for detecting RNase P RNA genes. *BMC Genomics* 11:432
18. Rivas E (2013) The four ingredients of single-sequence RNA secondary structure prediction: a unifying perspective. *RNA Biol* 10:59–70
19. Wuchty S, Walter F, Hofacker IL, Schuster P et al (1999) Complete suboptimal folding of RNA and the stability of secondary structures. *Biopolymers* 49:145–165
20. Wolfinger MT, Svrcek-Seiler AW, Flamm C, Hofacker IL, Stadler PF (2004) Efficient computation of RNA folding dynamics. *J Phys Math Gen* 37:4731
21. Reiche K, Stadler PF (2007) RNAstrand: reading direction of structured RNAs in multiple sequence alignments. *Algorithm Mol Biol* 2:6
22. Washietl S (2010) Sequence and structure analysis of noncoding RNAs. *Methods Mol Biol* 609:285–306
23. Hofacker IL, Tafer H (2010) Designing optimal siRNA based on target site accessibility. *Methods Mol Biol* 623:137–154
24. Eggenhofer F, Hofacker IL, Höner Zu Siederdissen C (2013) CMCompare web-server: comparing RNA families via covariance models. *Nucleic Acids Res* 41: W499–W503
25. Gruber AR, Fallmann J, Kratochvill F, Kovarik P, Hofacker IL (2011) ARESite: a database for the comprehensive investigation of AU-rich elements. *Nucleic Acids Res* 39:D66–D69
26. Mathews DH, Disney MD, Childs JL, Schroeder SJ, Zuker M, Turner DH (2004) Incorporating chemical modification constraints into a dynamic programming algorithm for prediction of RNA secondary structure. *Proc Natl Acad Sci U S A* 101: 7287–7292
27. Hofacker I, Lorenz R (2014) Predicting RNA structure: advances and limitations. *Methods Mol Biol* 1086:1
28. Smith C, Heyne S, Richter AS, Will S, Backofen R (2010) Freiburg RNA tools: a web server integrating INTARNA, EXPARNA and LOCARNA. *Nucleic Acids Res* 38: W373–W377

Chapter 20

Exploring the RNA Editing Potential of RNA-Seq Data by ExpEdit

**Mattia D'Antonio, Ernesto Picardi, Tiziana Castrignanò,
Anna Maria D'Erchia, and Graziano Pesole**

Abstract

Revealing the impact of A-to-I RNA editing in RNA-Seq experiments is relevant in humans because RNA editing can influence gene expression. In addition, its deregulation has been linked to a variety of human diseases. Exploiting the RNA editing potential in complete RNA-Seq datasets, however, is a challenging task. Indeed, no dedicated software is available, and sometimes deep computational skills and appropriate hardware resources are required. To explore the impact of known RNA editing events in massive transcriptome sequencing experiments, we developed the ExpEdit web service application. In the present work, we provide an overview of ExpEdit as well as methodologies to investigate known RNA editing in human RNA-Seq datasets.

Key words RNA editing, A-to-I editing, Deep sequencing, Bioinformatics, Genomics, RNA-Seq

1 Introduction

RNA editing is a posttranscriptional mechanism whereby genetic information is enzymatically overwritten at specific sites to generate functional transcripts, different from their corresponding genomic templates [1]. In mammals and, in particular, in humans, the A-to-I conversion is the most frequent type of RNA editing catalyzed by the ADAR (adenosine deaminase acting on RNA) family of enzymes. Inosines are commonly recognized as guanosines by cell molecular machineries as well as sequencing enzymes.

RNA editing has a relevant biological role in preserving the cellular homeostasis even though several aspects are yet unknown. Thanks to bioinformatics methods, thousands of editing changes have detected and collected in freely available specialized databases as DARNED [2] and RADAR [3]. These extraordinary collections can be employed to provide RNA editing snapshots in whole transcriptome experiments through the RNA-Seq technology. Indeed, a large amount of RNA-Seq data is now publicly available

through the SRA repository at NCBI. In this context, we have developed ExpEdit, a web application for exploring RNA editing in humans at known or user-specified sites supported by RNA-Seq experiments [4].

A great benefit of web applications, such as ExpEdit, is the possibility to analyze RNA-Seq data requiring neither technical competences nor expensive computational resources (usually a computing server or a computational cluster). In addition, results are provided in organized interfaces for easy and intuitive visualizations.

In this work, we provide an explanation of ExpEdit as well as methodologies to efficiently explore known human RNA editing in RNA-Seq datasets.

2 Materials

Results reported in this paper are based on raw sequencing reads obtained from the study published by [5] and stored into NCBI SRA under the accession SRP002274. It consists of RNA-Seq data from the human brain obtained through the Illumina Genome Analyzer II sequencer. We picked three of available six paired-end lanes (SRR039628/29, SRR039630/31 and SRR039632/33). Each lane comprises on average 7.8 million reads with an individual length of 50 bp.

2.1 Required Software and Hardware Resources

The management of large datasets generated from deep sequencing experiments typically requires powerful hardware resources. Hundred GBs of free disk space are needed to store inputs, outputs and intermediate temporary files. Bioinformatics software is often expensive in terms of computational resources and requires a large amount of RAM (16–24 GB at least) and several CPUs/cores to be executed in a reasonable time.

To handle several inputs as different lanes from the same experiment, analysis workflows need to be parallelized on multiple servers or serialized to analyze one input a time in full automation. Web applications overcome these hardware requirements, and the final user only needs a web browser and a stable Internet connection.

Input files can be directly uploaded or transferred providing a web link. In the first case, they have to be stored locally, and the upload process could be time consuming depending on file size and connection speed. Alternatively, inputs can also be transferred providing a web link to the location of data stored on remote servers. Using web links, the transfer is independent on the user connection speed. A further benefit of web applications is that analysis results can be downloaded by the user and stored locally to perform off-line investigations.

For our aim and thus the investigation of known RNA editing events in RNA-Seq datasets, the user needs a standard desktop or laptop computer with a stable Internet connection and an updated browser like the Internet Explorer, Firefox, or Safari.

ExpEdit service is available at the following web page <http://bioinformatics.cineca.it/expedit>, and a free registration is required to upload data and start computational analyses.

3 Methods

ExpEdit is designed to integrate in-house developed scripts as well as open source analysis tools into a single pipeline. It automatically handles and manages all required computational tasks. Every submission and retrieval operation is performed through an interactive web-based graphical user interface (GUI) verified on the main popular browsers. The GUI is written in PHP: Hypertext Preprocessor (PHP) language using HyperText Markup Language (HTML), and jQuery, combined with HTML5 and CSS3 standards, to enable a better user interaction.

ExpEdit takes as input short-read datasets produced by Illumina sequencing platforms in several standard file formats as FASTQ [6], SRA [7], or BAM [8]. The user can also upload these files in compressed archives (several compressed formats are also managed, such as .zip, .tar, .gzip, or .bz2) or provide ftp/http links to facilitate and speed up the upload process. In case of ftp/http links, the Wget application is used to retrieve input files.

The ExpEdit analysis pipeline can be summarized in three main steps: (1) quality assessment, (2) genome alignment, and (3) RNA editing calling. The first two steps are skipped if BAM inputs are provided.

Initially, ExpEdit performs an overall quality check (quality control or QC) of input data because read quality is critical for reliable analyses and may sensibly affect downstream results. For each FASTQ file, ExpEdit runs the FastQC program generating HTML report pages with detailed QC results organized in tables and graphs. Such reports include relevant information about raw reads such as quality score distributions, nucleotide distributions, or overrepresented sequences.

The ExpEdit pipeline implements also an additional application for quality control: the NGS QC Toolkit. In contrast to FastQC, it can remove contaminants as primers or adaptors and trim read regions with low quality (generally with a Phred score lower than 20) [9].

After the quality assessment, ExpEdit proceeds with the mapping onto the reference human genome. This is a critical step because erroneous alignments can seriously affect RNA editing detection and quantification. Genome mapping is performed by

means of TopHat [10, 11] program based on Bowtie [12]. TopHat handles both single and paired-end reads and can perform spliced alignments as required for human RNA-Seq experiments.

Alternatively, pre-aligned reads obtained by other software can be uploaded in ExpEdit as SAM/BAM format. Users are also completely free to download TopHat alignments, refine them off-line, and reload modified SAM/BAM files.

At the end of genome mapping, ExpEdit starts the RNA editing analysis launching in-house scripts, developed in Python programming language that makes use of the SAMtools package. In particular, unique read alignments are converted from SAM/BAM format to the pileup format, and genomic positions are explored site by site. During the parsing of pileup file, genomic positions are compared with those stored in specialized databases as DARNED [2]. Known positions subject to RNA editing are then printed out as well as the nucleotide distribution of supporting reads and the RNA editing level. A specific filter to exclude bases with a prefixed Phred-like quality score (by default fixed to 20) is also implemented. In addition, read nucleotides supporting a specific genomic position can be filtered according to the strand to take into account the effect of antisense transcription (recommended for strand-oriented reads). Individual positions are also annotated including gene name and RNA regions (CDS, intron, or UTR) or info about single nucleotide polymorphisms (SNPs).

Finally, ExpEdit automatically produces output tables that can be dynamically sorted, filtered, and exported as Excel files for off-line downstream analyses.

ExpEdit can also explore RNA editing at custom genomic positions by uploading a text file containing a list of candidates.

3.1 Hands On: Using the ExpEdit GUI

Every ExpEdit action can be executed through an interactive GUI available at <http://bioinformatics.cineca.it/expedit/>, and whatever popular web browser can be used to perform a complete analysis. Hereafter, we provide guidelines to create an ExpEdit study/project, upload data, run an entire analysis, and inspect results.

3.1.1 Creating a New Study/Project

The first step to submit a dataset to ExpEdit is the creation of a new project/study that will contain one or more input files as well as analysis results. To register a new study, a few information are required such as a title, a description and an access level (private, group, or public) (Fig. 1).

3.1.2 Uploading Input Files

The ExpEdit upload engine offers several options: web upload, web link, and Dropbox. The FTP protocol is planned but not yet implemented. Web Upload supports up to 12 GB file size on 64-bit operating systems and up to 2 GB on 32-bit operating systems. It is implemented in JavaScript and allows the user to upload several

Study archive

Create a new study

Name	<input type="text"/>
Description	<input type="text"/>
Access	private

In order to create a new study you should give it a unique name and write a short description about its subject.

Create new study

Fig. 1 Creation of a new ExpEdit project. Through the web interface, the user can create new projects by providing minimal information

files at once, handling a fully automated upload queue. The user can follow the upload progress and interact with the system adding or removing files also during the transfer. Alternatively, the user can choose to upload data providing a web link (HTTP, HTTPs, and FTP protocols are allowed). In this case, one or more links have to be provided, and the system will manage the download using an internal queue. An e-mail will notify the completion of downloads. A third option consists of the use of the Dropbox plug-in (Fig. 2).

The ExpEdit upload facility supports several input formats such as text-based raw sequences (i.e., .fastq or .fq), pre-aligned data (i.e., .bam or .sam), compressed reads (i.e., .sra), and compressed archives (i.e., .zip, .gz, .tar, or .bz2).

For paired-end RNA-Seq reads, each pair needs to be specified using a drag-and-drop tool provided by the GUI. The same tool can be used to reorder lanes (if needed). Finally, a unique label has to be assigned to each file or pair of files. Such label will be used instead of filename in the results pages.

Once input files have been imported into the study, users will be allowed to request for a new analysis.

3.1.3 Creating a New Analysis

To create a new analysis, one or more files have to be selected. Then, a name and a description can be provided. Before launching ExpEdit, several parameters can be tuned to customize the analysis, even though default parameters are suitable for many experiments.

ExpEdit includes four categories of parameters:

1. Common to many pipeline modules such as the human reference database (hg18 and hg19 assemblies are allowed).
2. For quality check and filtering.

Please choose how to upload your files.

Alert: this Web Upload supports up to **12GB** file size if you are running a 64-bit operating system, and up to **2GB** file size if you are running a 32-bit operating system.

Web Upload Web link Dropbox FTP

Add file by Web Upload

+ Add files...

lane2.fastq	28.66 MB	<div style="width: 20%;"><div style="width: 100%;"> </div></div>	<input type="button" value="Start"/>	<input type="button" value="Cancel"/>
lane1.fastq	664.49 MB	<div style="width: 10%;"><div style="width: 100%;"> </div></div>	<input type="button" value="Start"/>	<input type="button" value="Cancel"/>

119.16 kbit/s | 12:49:29 | 0.79 % | 5.47 MB / 693.15 MB

When your uploads are completed click on the Continue button to process your files.
Please wait until all your uploads are completed.
If you uploaded compressed files, decompression will be executed automatically after the 'Continue' button.

Fig. 2 Uploading files in ExpEdit. Input files can be added to an existing project by the web interface or by providing a web link or by importing data from a Dropbox account

3. For reads mapping through main TopHat options and allowing the alignment on both genome and transcriptome by providing a set of gene model annotations and/or known transcripts in GTF2 or GFF3 format. Different sequencing libraries are admitted in order to handle strand-oriented reads. Standard parameters such as the max number of allowed mismatches or multihits, the gap max length, and the min and max intron length can also be specified.
4. To detect RNA editing sites allowing the selection of known databases or the upload of a custom list of RNA editing candidates (the list must contain a site per row and three tab-separated fields including genomic region, coordinate and strand). A Phred-like quality score for filtering out supporting nucleotides below a predefined value (by default equal to 20) can be provided. Read nucleotides can also be filtered according to the strand to take into account the effect of antisense transcription (recommended for strand-oriented reads).

Figure 3 shows a screenshot of the ExpEdit web form to create an analysis and tune parameters.

New Analysis

RNA Editing with alignment performed by Tophat
Detection of A-to-I editing sites

Label	<input type="text" value="Analysis label"/>
Description	<input type="text" value="Analysis description"/>

Pipeline parameters (You can safely use defaults)

Common parameters

Database	<input type="text" value="hg19"/> <small>Choose the database for the alignment</small>
----------	--

Detection of RNA-Editing sites

Darned	<input type="text" value="hg19_darned"/> <small>Choose the Darned Database</small>
Min_quality	<input type="text" value="25"/> <small>Minimum Phred quality score to filter in read nucleotides supporting each reference position (min 15, max 50) (default: 25)</small>
Strand_based_filter	<input type="text" value="no"/> <small>Filter read nucleotides supporting each reference position according to its strand. (This filter is recommended for strand oriented reads)</small>
Editing_candidates	<input type="button" value="Scegli file"/> Nessun file selezionato <small>Custom list of editing candidates. The list must contain one position per row and the following fields separated by a tab/space character: genomic region (format chrN where N is the number of the chromosome or X/Y) genomic coordinate (1-based and according to the hg18 version of the human genome) genomic strand (+ or -) gene name (this field is optional)</small>
<small>Example: chr17 75874471 + DNAH2 chr3 48303723 + ZNF589</small>	

Genome alignment

Reference-GTF	<input type="text" value="hg18_gtf"/> <small>Reference GTF used to guide assembly</small>
--min-anchor	<input type="text" value="8"/> <small>Min anchor length around junctions (default: 8)</small>

Fig. 3 Creation of a new analysis. The analysis workflow can be configured tuning several parameters

3.1.4 Monitoring an Analysis

Once all parameters have been customized, the analysis can be submitted to the queue system. The execution process is fully automated with a complex dispatching architecture integrated with a TORQUE Resource Manager. The user can follow the analysis progress through a monitor page, which displays the list of pipeline steps and the intermediate output results (these files can be visualized and/or downloaded, also during the analysis execution). For each step, the running status and the used algorithm and its version are shown (Fig. 4). In case of execution fault, the running status is marked as error. At the completion of each step of the pipeline, all generated output files are listed and displayed in the monitoring page. In this way, the user can visualize or download all intermediate files. The command line and the estimated running time per task are also provided.

3.1.5 Analysis Results

At the end of the pipeline, all results are parsed and stored into a dedicated and optimized database. In addition, an e-mail containing a link to the available results will be automatically sent to the user.

Step(s) list

File	OriginalFile	Label	ExtraInfo	Download
file1.fastq.R1	SRR787271_1.fastq	brain	Short-reads: 39856205 x2 HQ reads: 36165072 x2	Download
file1.fastq.R2	SRR787271_2.fastq			Download

Fig. 4 Monitoring an analysis. During the execution, each computational task can be inspected to monitor the job status and intermediate output files

Results are organized in two sections (Fig. 5). The “quality checks” section refers to results from the quality control analysis obtained by FastQC and NGS QC Toolkit. The “editing sites” section reports a summary of all detected RNA editing sites and gives an overview of results. By clicking on these summaries, a detailed list of sites is displayed (*see Subheading 3.1.6*) (Fig. 5). Otherwise, the user can create one or more operations of intersection and union (*see Subheading 3.1.7*).

3.1.6 Result Details

Final ExpEdit results are provided in a detailed table that can be filtered, sorted and downloaded (Fig. 6). Every site in the table is identified by a genomic position (chromosome, coordinate, and strand) annotated with relevant biological information. Given a reference base, ExpEdit reports the total number of reads supporting the specific site and the number of reads supporting each

Analysis EditingBrain: results page

Quality checks Editing Sites

The meaning of Quality Columns is explained [here](#)

File	Label	Checks And Filtering	Basic Statistics	Per Base Sequence Quality	Per Sequence Scores	Per Sequence Content	Per Base GC Content	Per Sequence GC Content	Per Base N Content	Sequence Length Distribution	Sequence Duplication Levels	Overrepresented Sequences	Kmer Content	Quality Summary
1	brain		PASS	PASS	PASS	WARN	FAIL	WARN	WARN	PASS	FAIL	FAIL	FAIL	QC pair1
			PASS	PASS	PASS	FAIL	FAIL	WARN	PASS	PASS	FAIL	WARN	FAIL	QC pair2

Lane	Label	#Editing Site	...With Cov≥10	...And Edit%>0
1	brain	108874	11872	1674

Fig. 5 Visualization of summary results. After each run, ExpEdit shows simple summary tables (quality checks on the *top* and detected editing sites on the *bottom*)

Editing sites found for 'brain'

Filter, browse and download results

Sample: tissue1

Page Info

4 elements
1 pages
10 rows-per-page

List of selected Filters

Supp._reads	≥ '10'
Chr	= 'chr14'
Reference	= 'G'

Download Reset Filters Choose Filters

per page elements 10 Select page: 1

Click on a column title to order this table

UID	Gene	Position	Strand	Ref	Supp. Reads	#A	#C	#G	#T	Edit Perc	Source	Repeat Class	Repeat Name	SNP	RNA Position	Transcript And Localization
1	MAP4K5	chr14:50053585	+	G	17	0	0	17	0	0	D	sprRNA	7SLRNA	-	-	(0)
2	SLC38A6	chr14:61517256	+	G	16	0	0	16	0	0	D	-	-	rs74825213	CDS exon	NM_001172702 (CDS) NM_153811 (CDS) other 1...
3	MOAP1	chr14:93649014	-	G	231	0	0	231	0	0	D	SINE	AluSz	-	3UTR	NM_022151 (3UTR)
4	EVL	chr14:100606916	+	G	13	1	0	12	0	0.08	D	SINE	AluSq2	-	intron	NM_016337 (intron)

Fig. 6 Visualization of detailed results. RNA editing site is shown in dynamic tables

nucleotide. The editing percentage is defined as the ratio between the number of Gs over the number of As + Gs. An editing percentage equal to 100 means that all reads support the editing event. A percentage equal to 0 means that RNA editing is unlikely.

Results can be sorted in any order (ascending and descending) and filtered with custom thresholds to facilitate the identification of functionally significant variants. Filters can be combined to produce complex queries, and resulting tables can be exported as textual/Excel files for off-line downstream analyses.

3.1.7 Intersections and Unions

When two or more analyzed samples are available, union and intersection operations can be performed. A union operation may be helpful to merge together results obtained from biological replicates, while intersections may be useful to filter out aspecific results. Unions and intersections can be combined together to build complex operations, involving multiple lanes.

3.2 Case of Study

In this section, we provide an example of ExpEdit analysis using raw sequencing data obtained from the study published by [5] and stored into NCBI SRA database under the accession SRP002274.

Input reads can be uploaded in ExpEdit using the web link facility and the following URLs:

```
http://ftp-trace.ncbi.nlm.nih.gov/sra/sra-instant/reads/ByRun/sra/SRR/SRR039/SRR039628/SRR039628.sra
http://ftp-trace.ncbi.nlm.nih.gov/sra/sra-instant/reads/ByRun/sra/SRR/SRR039/SRR039629/SRR039629.sra
http://ftp-trace.ncbi.nlm.nih.gov/sra/sra-instant/reads/ByRun/sra/SRR/SRR039/SRR039630/SRR039630.sra
http://ftp-trace.ncbi.nlm.nih.gov/sra/sra-instant/reads/ByRun/sra/SRR/SRR039/SRR039631/SRR039631.sra
http://ftp-trace.ncbi.nlm.nih.gov/sra/sra-instant/reads/ByRun/sra/SRR/SRR039/SRR039632/SRR039632.sra
http://ftp-trace.ncbi.nlm.nih.gov/sra/sra-instant/reads/ByRun/sra/SRR/SRR039/SRR039633/SRR039633.sra
```

SRA archives will be automatically extracted by the ExpEdit preprocessing facility and converted into regular FASTQ files.

Now a new analysis can be created and executed by selecting the human genome assembly hg19/build37 and tuning parameters in the following way:

- Quality check and filtering: default values for cutoffs, length (70) and quality (20). No primer/adaptor library has to be selected.
- Genome alignment: the paired-end library has to be configured as unstranded with a mate inner distance of 200 bp and a mate standard deviation of 20. Default values should be used

for other parameters, i.e., max-multihits (1), segment-mismatches (2), max-intron-length (500,000), min-intron-length (70), min-anchor(8), read-edit-dist (2), read-gap-length (2), segment-length (25), and splice-mismatches (0).

- Detection of RNA editing sites: Phred quality filter, 20, no strand-based filter. No custom list of editing candidates has to be provided.

The first analysis step will consist in the run of FastQC version 0.10.1 that will be executed in about 13 min. All inputs should show the same trend with qualities decreasing at 3' end and Phred score dropping below the threshold of 20 in the last 4–5 positions. Sequence duplication level should be between 34 and 35 %, without specific overrepresented sequence.

In the second step, NGS QC Toolkit version 2.3 will be executed in about 35 min and the output should report statistics similar to FastQC.

The mapping of high-quality reads to the human genome by TopHat version 2.0.9 will be executed in about 3 h. Each lane should show an alignment rate above 90 %, and TopHat should report a rate of concordant alignment of 81.6, 81.7 and 81.1 % for each sample.

Finally, the ExpEdit computational core for RNA editing calling will be executed in about 4 h. Results obtained from this step will be parsed and stored into a MySQL database.

The complete pipeline should take less than 8 h. Input files (six files in three lanes for a total file size of 11 GB) will produce at the end 130 GB of intermediate output files.

Final tables should reveal on average 800 RNA editing sites with coverage higher than or equal to 10 and about 320 positions with an RNA editing percentage higher than 0.

Since all lanes are biological replicates of the same brain sample, the union operation can be executed to merge together these results.

All tables can be downloaded for off-line inspection by Excel or browsed to look at specific RNA editing events.

Acknowledgments

This work was supported by the Italian Ministero dell'Istruzione, Università e Ricerca (MIUR), PRIN 2009 and 2010, and Consiglio Nazionale delle Ricerche, Flagship Project Epigen, Medicina Personalizzata, and Aging Program 2012–2014.

References

1. Gott JM, Emeson RB (2000) Functions and mechanisms of RNA editing. *Annu Rev Genet* 34:499–531
2. Kiran A, Baranov PV (2010) DARNED: a DAtabase of RNAs EDiting in humans. *Bioinformatics* 26(14):1772–1776. doi:[10.1093/bioinformatics/btq285](https://doi.org/10.1093/bioinformatics/btq285), Epub 2010 Jun 14
3. Ramaswami G, Li JB (2014) RADAR: a rigorously annotated database of A-to-I RNA editing. *Nucleic Acids Res* 42(D1):D109–D113. doi:[10.1093/nar/gkt996](https://doi.org/10.1093/nar/gkt996)
4. Picardi E, D'Antonio M, Carrabino D, Castrignano T, Pesole G (2011) ExpEdit: a webserver to explore human RNA editing in RNA-Seq experiments. *Bioinformatics* 27(9):1311–1312. doi:[10.1093/bioinformatics/btr117](https://doi.org/10.1093/bioinformatics/btr117)
5. Au KF, Jiang H, Lin L, Xing Y, Wong WH (2010) Detection of splice junctions from paired-end RNA-seq data by SpliceMap. *Nucleic Acids Res* 38(14):4570–8. doi:[10.1093/nar/gkq211](https://doi.org/10.1093/nar/gkq211)
6. Cock PJ, Fields CJ, Goto N, Heuer ML, Rice PM (2010) The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants. *Nucleic Acids Res* 38(6):1767–71. doi:[10.1093/nar/gkp1137](https://doi.org/10.1093/nar/gkp1137)
7. Leinonen R, Sugawara H, Shumway M, Collaboration., International Nucleotide Sequence Database (2011) The sequence read archive. *Nucleic Acids Res* 39(Database issue): D19–D21
8. Li H, Handsaker B, Wysoker A, Fennell T, Ruan J, Homer N, Marth G, Abecasis G, Durbin R, 1000 Genome Project Data Processing Subgroup (2011) The sequence alignment/map format and SAMtools. *Bioinformatics* 25(16):2078–2079. doi:[10.1093/bioinformatics/btp352](https://doi.org/10.1093/bioinformatics/btp352)
9. Patel RK, Jain M (2012) NGS QC toolkit: a toolkit for quality control of next generation sequencing data. *PLoS One* 7(2):e30619. doi:[10.1371/journal.pone.0030619](https://doi.org/10.1371/journal.pone.0030619)
10. Trapnell C, Pachter L, Salzberg SL (2009) TopHat: discovering splice junctions with RNA-Seq. *Bioinformatics* 25(9):1105–1111. doi:[10.1093/bioinformatics/btp120](https://doi.org/10.1093/bioinformatics/btp120)
11. Kim D, Pertea G, Trapnell C, Pimentel H, Kelley R, Salzberg SL (2011) TopHat2: accurate alignment of transcriptomes in the presence of insertions, deletions and gene fusions. *Genome Biol* 14(4):R36. doi:[10.1186/gb-2013-14-4-r36](https://doi.org/10.1186/gb-2013-14-4-r36)
12. Langmead B, Salzberg S (2012) Fast gapped-read alignment with Bowtie 2. *Nat Methods* 9(4):357–359. doi:[10.1038/nmeth.1923](https://doi.org/10.1038/nmeth.1923)

Chapter 21

A Guideline for the Annotation of UTR Regulatory Elements in the UTRsite Collection

Matteo Giulietti, Giorgio Grillo, Sabino Liuni, and Graziano Pesole

Abstract

Gene expression regulatory elements are scattered in gene promoters and pre-mRNAs. In particular, RNA elements lying in untranslated regions (5' and 3'UTRs) are poorly studied because of their peculiar features (i.e., a combination of primary and secondary structure elements) which also pose remarkable computational challenges. Several years ago, we began collecting experimentally characterized UTR regulatory elements, developing the specialized database UTRsite. This paper describes the detailed guidelines to annotate *cis*-regulatory elements in 5' and 3' UnTranslated Regions (UTRs) by computational analyses, retracing all main steps used by UTRsite curators.

Key words UTR, *cis*-Regulatory elements, RNA secondary structure, Sequence alignments, Perl, Motif pattern

1 Introduction

In the post-genomic era, the identification of gene regulatory elements and the understanding of their biological role in modulating gene expression represent a big challenge. Gene regulatory elements are generally located in noncoding regions of eukaryotic genomes and act at transcriptional and/or posttranscriptional level. Many DNA elements in promoter regions are known to control gene transcription by interacting with transcription factors and several databases and software to classify and identify these elements have been developed [1]. At posttranscriptional level, instead, the control of gene expression rely on RNA elements that are scattered in whole pre-mRNA sequences and are involved in many processes such as splicing, RNA folding, capping, polyadenylation, nucleocytoplasmic transport, translation efficiency, subcellular localization, stability and determine the fate of mature mRNA in the cell [2]. Regarding the splicing, we recently developed SpliceAid [3, 4] and SpliceAid-F [5] databases that collect data about human splicing factors and their experimentally assessed

pre-mRNA binding sites. For regulatory elements lying in 5' and 3' UnTranslated Regions (5'UTRs, 3'UTRs), we have developed two specialized database, UTRdb and UTRsite, collecting eukaryotic UTR sequences and experimentally characterized UTR-specific regulatory signals, respectively [6]. In spite of transcription and splicing factor binding sites, RNA elements in UTRs are much less investigated. Indeed, their biological activities often require a combination of primary and secondary structure elements that are hard to predict correctly. Computational methods and available tools for predicting RNA secondary structure are reviewed in [7, 8]. This paper describes detailed computational approaches and guidelines to characterize and annotate UTR regulatory *cis*-elements following same steps adopted by UTRsite curators. In addition, we show the annotation process through an example concerning the characterization of the DNA polymerase beta stem loop II element [9].

2 Materials

2.1 Hardware and Software

The annotation workflow described in this paper can be executed on a regular computer without specific hardware resources. However, it must have a stable Internet connection to download executables or to access dedicated Web services. Although whatever operating system as Linux, Mac OS X, or Windows is suitable, the Perl interpreter needs to be preinstalled as well as the BioPerl package. An updated Perl interpreter for Windows and Mac OS X can be downloaded from www.activestate.com/activeperl. Linux users can install a copy via apt or yum tools. BioPerl package can be downloaded from www.bioperl.org.

2.2 Web Servers and Databases

PubMed at www.ncbi.nlm.nih.gov/pubmed will be used to search scientific literature. BLAST tool [10] at <http://blast.ncbi.nlm.nih.gov/Blast.cgi> will be employed to retrieve all orthologous sequences of UTR elements under investigation. DIALIGN [11] at <http://bibiserv.techfak.uni-bielefeld.de/dialign/> is a multiple sequence alignment (MSA) tool that differently from other MSA tools, is useful to detect local homologies in sequences with low overall similarity. LocARNA [12] at <http://rna.informatik.uni-freiburg.de:8080/LocARNA/Input.jsp> is a program to predict a consensus secondary structure from a set of unaligned RNAs by computing multiple alignments (*see Note 1*). PatSearch tool [13, 14] at <http://itbtools.ba.itb.cnr.it/patsearch> is a program able to search for specific patterns in nucleotide sequences. It is useful to detect sequence regions involved in secondary structure or to build position-weight matrices. UTRsite [6] is a specialized database of functional elements in 5' or 3' UTRs, available at <http://utrsite.ba.itb.cnr.it>.

3 Methods

3.1 Literature Search

The UTRsite entries describe the known regulatory elements located in 5'UTR or 3'UTR or both, whose functional role and structure have been experimentally assessed and reported in literature. Therefore, PubMed database is the starting point for UTRsite data collection and to identify new regulatory elements. The literature search is a crucial and time-consuming process.

The literature information related to UTR regulatory elements can be obtained using a variety of PubMed queries providing keywords like “UTR” or “untranslated regions” or “5’UTR” or “3’UTR” and different combinations among them (by means of Boolean operators AND, OR, and BUT NOT). Limits can also be activated to reduce the number of records and false positives, increasing the search specificity. Papers describing the same functional element are examined in order to extract all available information about primary sequence, secondary structure, RNA localization (5’ or 3’UTR), biological role, RNA-binding proteins and experimental procedures. In general, the functional characterization of UTR regulatory elements includes gene expression assays using recombinant or modified UTRs, site-specific mutagenesis, RNase protection, and chemical probing experiments.

Here, we present and explanatory example focusing on the DNA polymerase beta stem loop II regulatory element (POLB) which has been identified in a PubMed search using aforementioned terms [9, 15–17]. Related literature describes this element as a 3'UTR *cis*-acting element involved in the posttranscriptional regulation of the DNA Polymerase beta gene. It is 42-nt long and has a stem-loop secondary structure that is fundamental for the interaction between the gene transcript and the Hax-1 protein which, in turn, is responsible for at least two mechanisms: localization in the perinuclear space and stabilization of the otherwise unstable mRNA.

3.2 Detection and Extraction of Orthologous Sequences

Once the UTR regulatory element has been identified from literature and the relative sequence has been retrieved, the NCBI BLAST tool should be used to search for all available orthologous sequences. We suggest to perform BLAST searches using as query the whole mRNA sequence including the element (or its RefSeq identifier) or the UTR sequence alone or simply the extracted motif sequence. Multiple queries are essential because the length of the motif alone is generally short and, thus, BLAST may provide many not significant and undesirable alignments. All detected database hits must be carefully checked in order to define a reliable set of orthologous sequences, all containing the regulatory elements under investigation, and then investigate its conservation profile at level of primary and secondary structure that will be used to obtain an optimal definition of the regulatory pattern.

In the case of DNA polymerase beta stem loop II element, we used as query the *Rattus norvegicus* RefSeq id NM_017141, corresponding to the sequence used in [9] to experimentally assess the activity of the UTR element. In the NCBI BLAST tool, we selected the *blastn* algorithm, and *refseq_rna* as database. Other parameters were unchanged. After the search, all significant hits can be selected and downloaded in a FASTA-formatted text file (see Note 2). Given the list of Accession IDs obtained by the BLAST search, UTRs can be extracted from RefSeq mRNAs using the in house Perl script *UTR_extractor.pl* according to the following code:

```
#!/usr/bin/perl
use Bio::DB::GenBank;
if (! $ARGV[0]) {print "usage: UTR_extractor.pl [input
list file] [5UTR or 3UTR] \n";}
open (FILEIN, "<", "./$ARGV[0]");
open (FILEOUT, ">", "./output.txt");
@IDs = <FILEIN>;
foreach $id (@IDs) {
    $gb = new Bio::DB::GenBank;
    if (($id =~/_/) and ($id =~/\./)) {$seq = $gb->get_
Seq_by_version($id);} #RefSeq version (eg.
XM_003257165.2)
    if (($id =~/_/) and ($id !~ /\./)) {$seq = $gb->get_
Seq_by_acc($id);} #RefSeq (eg. XM_003257165)
    if (($id !~/_/) and ($id !~ /\./)) {$seq = $gb->get_
Seq_by_gi($id);} # GI (eg. 441611727)
    for my $feat ($seq->get_SeqFeatures) {
        if ($feat->primary_tag eq "gene") {
            $gene_seq = $feat->seq->seq();
            $gene_start = $feat->location->start;
            $gene_end = $feat->location->end;}
        if ($feat->primary_tag eq "CDS") {
            $CDS_start = $feat->location->start;
            $CDS_end = $feat->location->end;}
    }
    if ($ARGV[1] eq "5UTR") {
        $UTR5=substr ($gene_seq, $gene_start-1, $CDS_start-
$gene_start);
        print ">gi|$seq->primary_id()."|"$.seq->display_
id()."|5'UTR of ".$seq->desc()."\\n$UTR5\\n";
        print FILEOUT ">gi|$seq->primary_id()."|"$.seq-
>display_id()."|5'UTR      of      ".$seq->desc()."\\
n$UTR5\\n";}
    if ($ARGV[1] eq "3UTR") {
        $UTR3=substr ($gene_seq,   $CDS_end,   $gene_end-
$CDS_end);
        print ">gi|$seq->primary_id()."|"$.seq->display_
id()."|3'UTR of ".$seq->desc()."\\n$UTR3\\n";
        print FILEOUT ">gi|$seq->primary_id()."|"$.seq-
>display_id()."|3'UTR      of      ".$seq->desc()."\\
n$UTR3\\n";}
}
```

The above script can be executed by the command:

```
$ perl UTR_extractor.pl [input list file] [5UTR or 3UTR]
```

where [input list file] is the file containing a list of identifier (RefSeq accessions or GIs) and [5UTR or 3UTR] is a flag to specify which UTR end to extract. The script is able to retrieve sequences from GenBank using BioPerl and provide UTRs in a FASTA-formatted text file.

3.3 Alignment of Orthologous Sequences

In the next step, retrieved UTR sequences needs to be multi-aligned. For this task we suggest the tool DIALIGN paying attention to the content of the input file in order to exclude mRNAs that do not have an annotated UTR (for example mRNAs from predicted RefSeq). In addition, it could be useful to include in the input the sequence of the element obtained by literature to facilitate its localization in orthologous sequences and improve the final multi-alignment.

In case of very divergent sequences, the alignment software could produce biased results. So to limit false alignments, we suggest working in a hierarchical way aligning first sequences belonging to close related species and, then, adding further sequences one a time. Although alignment programs are very accurate, manual refinement is almost always needed to obtain a reliable multiple alignment.

In our example on DNA polymerase beta stem loop II regulatory element we used DIALIGN with default parameters. Only 30 sequences out of about 100 orthologous mRNAs derived from BLAST searches could be aligned with the rat element. This is mainly due to the presence of mRNAs without annotated full-length 3'UTRs. Some sequences, however, have to be discarded for insufficient sequence similarity to the rat element.

The multiple alignment is also important to deduce the taxonomic range as showed in Fig. 1. In this case, all 30 sequences belong to the class Mammalia.

3.4 Inferring the Secondary Structure

Multiple alignments generated by DIALIGN can be used to predict potential secondary structures and verify the correspondence between a consensus secondary structure and the one proposed by the literature. To this task, the LocARNA software should be employed. It takes as input a multiple alignment and tries to infer a consensus secondary structure.

Following our example on DNA polymerase beta stem loop II element, the inferred consensus secondary structure differs from the one predicted in literature. Indeed, the latter consists in 13 bp stem structure (Fig. 2a), while the former includes only a 9 bp stem and an asymmetric internal loop (Fig. 2b).

3.5 Generation of the Motif Pattern and Assessment of its Sensitivity/Specificity

Once a reliable multiple alignment is obtained, taking also into account the secondary structure of the UTR element, a motif pattern can be modelled using the PatSearch software (*see Note 3*) and its specific syntax rules. First, the multiple alignment has to be

gi 148747153	Rat (ARTICLE)	UU <u>AUUG</u> CU -UA ACCUUUUGC UAU GU <u>AAGGGU</u> -UG- GG UGUUUU
gi 229576830	Mus	UU <u>AUUC</u> CU -UA ACCUUUUGC UAU GU <u>AAGGGU</u> UUUU GG UGUUCU
gi 354482349	cricetus	UU <u>AUUC</u> CU -UA ACCUUUUGC UAU GU <u>AAGGGU</u> -UUU GG UGUUUU
gi 345525589	Homo	UU <u>AUUC</u> CU -UA ACCUUUUGC UAU GU <u>AAGGGU</u> CUUU GG UGUUUU
gi 77736472	Bos	UU <u>AUUC</u> CU -UA AGCUUUGC UAU GU <u>AAGGGU</u> CUGU GG UGUUUU
gi 332241021	Nomascus	UU <u>AUUC</u> CU -UA ACCUUUUGC UAU GU <u>AAGGGU</u> CUUU GG UGUUUU
gi 297682784	Pongo	UU <u>AUUC</u> CU -UA ACCUUUUGC UAU GU <u>AAGGGU</u> CUUU GG UGGUUU
gi 397505583	Pan_pan	UU <u>AUUC</u> CU -UA ACCUUUUGC UAU GU <u>AAGGGU</u> CUUU GG UGUUUU
gi 114619949	Pan_tro	UU <u>AUUC</u> CU -UA ACCUUUUGC UAU GU <u>AAGGGU</u> CUUU GG UGUUUU
gi 383872890	Macaca	UU <u>AUUC</u> CU -UA ACCUUUUGC UAU GU <u>AAGGGU</u> CUUU AG UGUUUU
gi 511912828	Mustela	UU <u>AUUC</u> CU -UA ACCUUUUGC UAU GU <u>AAGGGU</u> CUUU GG UGUUUU
gi 410956337	Felis	UU <u>AUUC</u> CU -UA ACCUUUUGC UAU GU <u>AAGGGU</u> CUUU GG UGUUUU
gi 478534251	Ceratotherium	UU <u>AUUC</u> CU -UA ACCUUUUGC UAU GU <u>AAGGGU</u> CUUU GG UGUUUU
gi 359321510	Canis	UU <u>AUUC</u> CU -UA ACCUUUUGC UAU GU <u>AAGGGU</u> CUUU GG UGUUUU
gi 296222126	Callithrix	UU <u>AUUC</u> CU -UA ACCUUUUGC UAU GU <u>AAGGGU</u> CUUU GG UGUUUU
gi 504165666	Ochotona	UU <u>AUUC</u> CU GUA ACCUUUUGC UAU GU <u>AAGGGU</u> CUUU GG UGUUUU
gi 472393040	odobenus	UU <u>AUUC</u> CU -UA ACCUUUUGC UAU GU <u>AAGGGU</u> CUUU GG UGUUUU
gi 466083816	orcinus	UU <u>AUUC</u> CU -UA ACCUUUUGC UAC GU <u>AAGGGU</u> CUGU GG UGUUUU
gi 344281573	Loxodonta	UU <u>AUUC</u> CU -UA ACCUUUUGC UAU GU <u>AAGGGU</u> CUUU GG UGUUUU
gi 301765967	Ailuropoda	UU <u>AUUC</u> CU -UA ACCUUUUGC UAU GU <u>AAGGGU</u> CUUU GG UGUUUU
gi 470598371	Tursiops	UU <u>AUUC</u> CU -UA ACCUUUUGC UAC GU <u>AAGGGU</u> CUAU GG UGUUUU
gi 403303647	Saimiri	UU <u>AUUC</u> CU -UA ACCUUUUGC UAU GU <u>AAGGGU</u> CUUU GG UGUUUU
gi 471397463	Trichechus	UU <u>AUUC</u> CU -UA ACCUUUUGC UCU GU <u>AAGGGU</u> CUCU GG UGUUUU
gi 507946886	Condylura	UU <u>AUUC</u> CU -UA ACCUUUUGC UAU GU <u>AAGGGU</u> CUUU GG UGUUUU
gi 507652452	Octodon	UU <u>AUUC</u> CU -UA ACCUUUUGC UGU GU <u>AAGGGU</u> GUUU GG UGUUUU
gi 395857476	Otolemur	UU <u>AUUC</u> CU -UA ACCUUUUGC UAU GU <u>AAGGGU</u> CUUU GG UGUUUU
gi 505795849	sorex	UU <u>AUUC</u> CU -UA ACCUUUUGC UAU GU <u>AAGGGU</u> CUUU GG UGUUUU
gi 426256601	Ovis	UU <u>AUUC</u> CU -UA AGCUUUGC UAU GU <u>AAGGGU</u> CUGU GG UGUUUU
gi 426359502	Gorilla	UU <u>AUUC</u> CU -UA ACCUUUUGC UAU GU <u>AAGGGU</u> CUUU GG UGUUUU
gi 402878107	Papio	UU <u>AUUC</u> CU -UA ACCUUUUGC UAU GU <u>AAGGGU</u> CUUU AG UGUUUU

Fig. 1 Multiple alignment generated by DIALIGN and manually refined. The *first column* reports the GenBank GI identifier and the organism. The character “l” in the alignment is used to distinguishing stems from loops. Mismatched nucleotides in stems are *underlined*. Colored nucleotides are used to facilitate the alignment visualization

checked in order to mark conserved and variable positions. In case of variable positions, the IUPAC nomenclature should be used (e.g.: Y means C or T; B means C, G, or T, and so on).

Each pattern can be composed by individual pattern units identified by p1, p2, pn. Individual units can be strings like “p1 = ACGYYAB” or complex pattern unit like “p2 = ACGYYAB[1,2,3]” allowing mismatch, deletions and insertions. A pattern unit can also include a range descriptor like “1...5”. For example, the pattern “AACU 1...5 GGAU” will recognize all sequences starting with AACU, followed by 1–5 uncharacterized nucleotides and ending with GGAU. The range descriptor is quite useful to describe alignments gaps or not conserved nucleotides. In addition, PatSearch program includes important rules to take into account RNA secondary structures through pairing rules as “r1 = {AU, UA, CG, GC, GU, UG}” allowing also GU base pairing. The pattern “p1 = ACCUAAGCC 3...10 r1~p1”, for instance, facilitate the identification of sequences composed by the string ACCUAAGCC

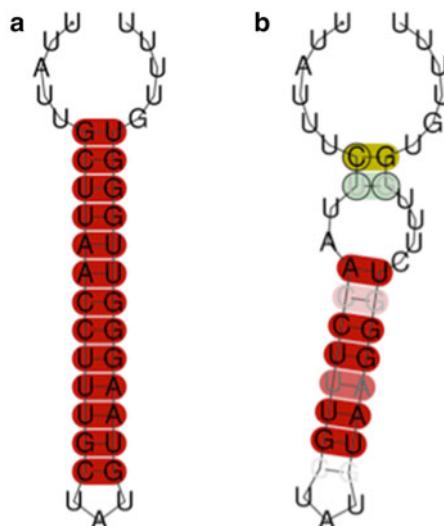


Fig. 2 The secondary structure of the rat DNA polymerase beta stem loop II element (a) and the inferred consensus structure from the multiple alignment (b)

followed by 3–10 bases acting as a loop and GG(C/U)UU(A/G) GGU as a second side of the stem (*see Note 4*). Complex PatSearch rules can also be defined. For a complete description of PatSearch syntax, please refer to [13, 14].

In our example on the DNA polymerase beta stem loop II regulatory element, the following pattern can be defined:

```
r1={AU,UA,CG,GC,UG,GU}
UUAUUK[0,1,0]    p1=CY    UA[0,0,1]    p2=ASCUUUGC    UVY
r1~p2[1,0,0]    CUUU[1,2,0]  r1~p1[1,0,0]  UGKUYU
```

where

UUAUUK[0,1,0]: is the first unpaired tract, where a deletion of 1 nt is allowed;

p1 = CY: is the first side of the first stem tract;

UA[0,0,1] : is the first side of the asymmetric internal loop, where an insertion of 1 nt is allowed;

p2 = ASCUUUGC: is the first side of the second stem tract;

UVY: is the external loop;

r1 ~ p2[1,0,0]: is the second side of the second stem tract, where a mismatch is allowed;

CUUU[1,2,0]: the second side of the asymmetric internal loop, where a mismatch and a deletion of up to 2 nt are allowed;

r1 ~ p1[1,0,0]: is the second side of the first stem tract, where a mismatch is allowed;

UGKUYU: is the second unpaired tract.

The correctness and statistical significance of PatSearch patterns can be assessed using shuffled datasets maintaining the same nucleotide composition of the original multiple alignment and applying a simple chi-square test. If the pattern is statistically significant, it can be used to search for similar patterns in other sequences and results could be good candidates for further functional investigations.

Using the above pattern for the DNA polymerase beta stem loop II regulatory element against all mammalian UTR sequences stored in UTRdb dataset (comprising over 260,000 sequences), PatSearch detects 24 UTRs with $p < 0.0001$.

3.6 Submit the Annotated UTR Element to UTRsite

Significant patterns and new UTR elements can be submitted to the UTRsite Web service (*see Note 5*). Each entry in UTRsite has several sections including all relevant info characterizing the element as those detected using the above methodology.

The UTRsite entry corresponding to the DNA polymerase beta stem loop II regulatory element is shown in Fig. 3 and can be directly inspected at the following Web page <http://utrsite.ba.itb.cnr.it/index.php/UTRSite%20signal/Signal/action/view/frmUTRSiteID/U0049>.

4 Notes

1. Potential alternatives to LocARNA are: (a) RNAAlifod [18] at <http://rna.tbi.univie.ac.at/cgi-bin/RNAAlifold.cgi> that predicts a consensus secondary structure from a set of pre-aligned sequences; (b) Infernal [19] at <http://infernal.janelia.org/> that builds consensus RNA profiles called covariance models (CMs), which is not available as Web service but needs to be downloaded and locally installed; (c) RNAG [20] at <http://ccmbweb.ccv.brown.edu/cgi-bin/rnag.pl>, a Gibbs sampler for predicting RNA secondary structure for unaligned sequences.
2. The mRNA sequences can be downloaded directly from BLAST result page. Alternatively, the retrieval of FASTA-formatted mRNAs from GenBank is feasible by providing a list of IDs (accessions or GIs) to the EFetch E-utility of NCBI Entrez through an URL. For example: http://eutils.ncbi.nlm.nih.gov/entrez/eutils/efetch.fcgi?db=nucleotide&rettype=fasta&retmode=text&id=NM_123,XM_456,NM_789,... where “db=nucleotide” indicates the NCBI database, “rettype=fasta” specifies the output sequence format and “id=NM_123, XM_456,NM_789,...” represent the list of IDs separated by commas.

UTRSite

user: guest [login](#)

Home | Signal Manager

Signal Manager :: View

:: General Information	
ID	U0049
Creation Date	26 July 2013
Updating Date	24 October 2013
Standard Name	DNA polymerase beta stem loopII regulatory element (POLB)
UTRSite Pattern	R1=[AU,UA,CG,GC,UG,GU] UUAAUK[0,1,0] P1=CY UA[0,0,1] P2=ACSUUUGC UVY R1-P2[1,0,0] CUUU[1,2,0] R1-P1[1,0,0] UGKUYU
Random Expectation	0.0001016410 hits/kb
:: Taxonomy	
UTR Region	3'
Taxon Range	Mammalia
:: Description	
Description	This signal is a cis-acting element involved in the post-transcriptional regulation of the DNA Polymerase beta gene and is located in the 3'UTR. It is characterized by a sequence of 42 nt, which form a stem-loop secondary structure. This hairpin is fundamental for the interaction between the transcript of the gene and the Hax-1 protein, which is responsible for at least two mechanisms: localization in the perinuclear space and stabilization of the otherwise unstable mRNA.
:: Features	
Feature Key	POLB
:: Database Cross-references	
Link	RFAM: RF01455
:: Transcript(s) / Gene(s)	
UTR(s)	Description: 3'UTR in <i>Rattus norvegicus</i> polymerase (DNA directed), beta (Polb); Species: <i>Rattus norvegicus</i> ; Link: UTRef: 3RNOR028468; Ref: [U1]
Transcript(s)	Description: <i>Rattus norvegicus</i> polymerase (DNA directed), beta (Polb), mRNA; Species: <i>Rattus norvegicus</i> ; Link: RefSeq: NM_017141; Ref: [T1]
Gene(s)	Description: Polb polymerase (DNA directed), beta; Species: <i>Rattus norvegicus</i> ; Link: EntrezGene: 29240; Ref: [G1]
:: Protein(s)	
Binding Protein(s)	Description: HS1-associating protein X-1 (Hax-1); Species: <i>Rattus norvegicus</i> ; Link: UniProt: Q7TSE9; Ref: [B1]
:: References	
ID	[1]
Authors	Sarnowska E, Grzybowska EA, Sobczak K, Konopinski R, Wilczynska A, Szwarc M, Sarnowski TJ, Krzyzosiak WJ, Siedlecki JA
Title	Hairpin structure within the 3'UTR of DNA polymerase beta mRNA acts as a post-transcriptional regulatory element and interacts with Hax-1
Citation	Nucleic Acids Res. 35(16):5499-510
Year	2007

Fig. 3 UTRsite entry (Accession ID U0049) for the DNA polymerase beta stem loop II regulatory element

3. PatSearch executables for Linux and Mac OS X can be required to Graziano Pesole (graziano.pesole@uniba.it) or Giorgio Grillo (giorgio.grillo@ba.itb.cnr.it).
4. We suggest testing the correctness of PatSearch syntax performing the pattern searching against all sequence of the multiple alignment.
5. To submit new UTR elements, users can follow the above methodology and then contact the UTRsite staff by e-mail following links inside the main Web page.

References

1. Wang C, Zhang MQ, Zhang Z (2013) Computational identification of active enhancers in model organisms. *Genomics Proteomics Bioinformatics* 11:142–150
2. Keene JD (2007) RNA regulons: coordination of post-transcriptional events. *Nat Rev Genet* 8:533–543
3. Piva F, Giulietti M, Burini AB et al (2012) SpliceAid 2: a database of human splicing factors expression data and RNA target motifs. *Hum Mutat* 33:81–85
4. Piva F, Giulietti M, Nocchi L et al (2009) SpliceAid: a database of experimental RNA target motifs bound by splicing proteins in humans. *Bioinformatics* 25:1211–1213
5. Giulietti M, Piva F, D'Antonio M et al (2013) SpliceAid-F: a database of human splicing factors and their RNA-binding sites. *Nucleic Acids Res* 41:D125–D131
6. Grillo G, Turi A, Licciulli F et al (2010) UTRdb and UTRsite (RELEASE 2010): a collection of sequences and regulatory motifs of the untranslated regions of eukaryotic mRNAs. *Nucleic Acids Res* 38:D75–D80
7. Mathews DH, Moss WN, Turner DH (2010) Folding and finding RNA secondary structure. *Cold Spring Harb Perspect Biol* 2:a003665
8. Hamilton RS, Davis I (2011) Identifying and searching for conserved RNA localisation signals. *Methods Mol Biol* 714:447–466
9. Sarnowska E, Grzybowska EA, Sobczak K et al (2007) Hairpin structure within the 3'UTR of DNA polymerase beta mRNA acts as a post-transcriptional regulatory element and interacts with Hax-1. *Nucleic Acids Res* 35:5499–5510
10. Johnson M, Zaretskaya I, Raytselis Y et al (2008) NCBI BLAST: a better web interface. *Nucleic Acids Res* 36:W5–W9
11. Morgenstern B (2004) DIALIGN: multiple DNA and protein sequence alignment at BiBiServ. *Nucleic Acids Res* 32:W33–W36
12. Smith C, Heyne S, Richter AS et al (2010) Freiburg RNA tools: a web server integrating INTARNA, EXPARNA and LOCARNA. *Nucleic Acids Res* 38:W373–W377
13. Grillo G, Licciulli F, Liuni S et al (2003) PatSearch: a program for the detection of patterns and structural motifs in nucleotide sequences. *Nucleic Acids Res* 31:3608–3612
14. Pesole G, Liuni S, D'Souza M (2000) PatSearch: a pattern matcher software that finds functional elements in nucleotide and protein sequences and assesses their statistical significance. *Bioinformatics* 16:439–450
15. Grzybowska EA, Zayat V, Konopinski R et al (2013) HAX-1 is a nucleocytoplasmic shuttling protein with a possible role in mRNA processing. *FEBS J* 280:256–272
16. Nowak R, Siedlecki JA, Kaczmarek L et al (1989) Levels and size complexity of DNA polymerase beta mRNA in rat regenerating liver and other organs. *Biochim Biophys Acta* 1008:203–207
17. Konopinski R, Nowak R, Siedlecki JA (1996) Alternative polyadenylation of the gene transcripts encoding a rat DNA polymerase beta. *Gene* 176:191–195
18. Hofacker IL (2007) RNA consensus structure prediction with RNAAlifold. *Methods Mol Biol* 395:527–544
19. Nawrocki EP, Eddy SR (2013) Infernal 1.1: 100-fold faster RNA homology searches. *Bioinformatics* 29:2933–2935
20. Wei D, Alpert LV, Lawrence CE (2011) RNAG: a new Gibbs sampler for predicting RNA secondary structure for unaligned sequences. *Bioinformatics* 27:2486–2493

Chapter 22

Rfam: Annotating Families of Non-Coding RNA Sequences

Jennifer Daub, Ruth Y. Eberhardt, John G. Tate, and Sarah W. Burge

Abstract

The primary task of the Rfam database is to collate experimentally validated noncoding RNA (ncRNA) sequences from the published literature and facilitate the prediction and annotation of new homologues in novel nucleotide sequences. We group homologous ncRNA sequences into “families” and related families are further grouped into “clans.” We collate and manually curate data cross-references for these families from other databases and external resources. Our Web site offers researchers a simple interface to Rfam and provides tools with which to annotate their own sequences using our covariance models (CMs), through our tools for searching, browsing, and downloading information on Rfam families. In this chapter, we will work through examples of annotating a query sequence, collating family information, and searching for data.

Key words Noncoding RNA, Rfam, Multiple sequence alignment, Secondary structure, Covariance model, Infernal, Families, Clans

1 Introduction

The Rfam database (<http://rfam.sanger.ac.uk>) is a collection of noncoding RNA sequences, grouped into “families” and “clans” on the basis of nucleotide sequence and secondary structure homology. This collection is not restricted to one type of ncRNA or one model organism but includes many different types of ncRNA genes, *cis*-regulatory elements, and intronic elements, all of which are annotated in as many species as possible. Rfam release 11.0 (Aug 2012) contains 2,208 ncRNA families (comprising 1,881 ncRNA genes, 317 *cis*-regulatory elements, and 10 catalytic introns) and 106 clans (encompassing 321 families). These 2,208 families annotate over 6 million sequence regions in the public nucleotide databases.

Rfam relies on an underlying sequence database known as “Rfamseq” which we derive from the European Nucleotide Archive (ENA) [1]. This database is searched for homologous sequences and annotated with matching Rfam families. For release 11.0, Rfamseq contains the entire STD (standard annotated assembled sequence) and WGS (whole genome shotgun sequencing) divisions of ENA (EMBL release 110), approximately 89 million

sequences (over 274.9×10^9 bp), and includes genome sequences for 3,110 unique species (101 eukaryotic, 1,747 bacterial, 123 archaeal, and 1,139 viral genomes).

1.1 Families

An Rfam family is a collection of nucleotide sequences that we believe have conserved sequence and secondary structure and is derived from a common ancestor. Each family is represented by a multiple sequence alignment annotated with a conserved secondary structure (referred to as the “seed”) and a probabilistic model (covariance model, CM).

The seed alignment is a minimal set of sequences, which are chosen as representative members of the family. Membership of the seed is curated by Rfam; however this alignment is derived from at least one experimentally validated example from the literature. The seed secondary structure annotation is obtained from the source literature or predicted using software tools such as WAR (Webserver for aligning structural RNAs) [2] and mfold [3]. The Infernal software [4] suite is used to generate the CM from the seed and used to search the Rfamseq databases for new homologues. All predicted homologues in Rfamseq are collated with the seed sequences into a single large alignment termed the “full.” We refer our readers to two comprehensive reviews on the use of covariance models for annotating ncRNA [5, 6] and other Rfam publications for more details on the process of building and annotating Rfam families [7].

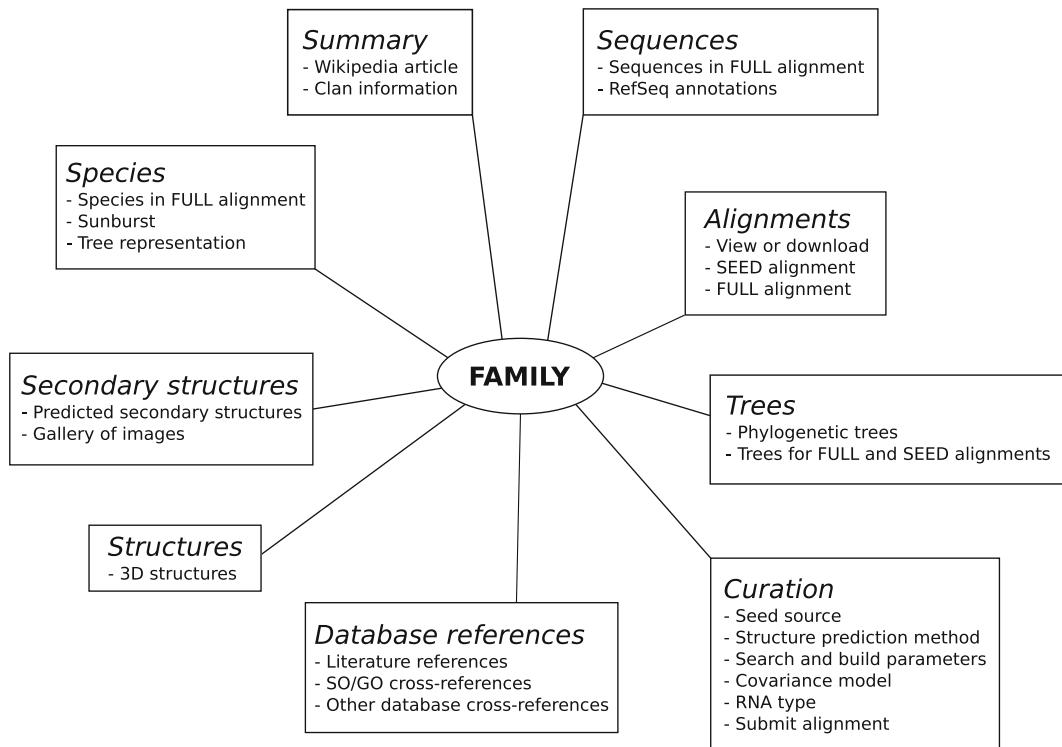
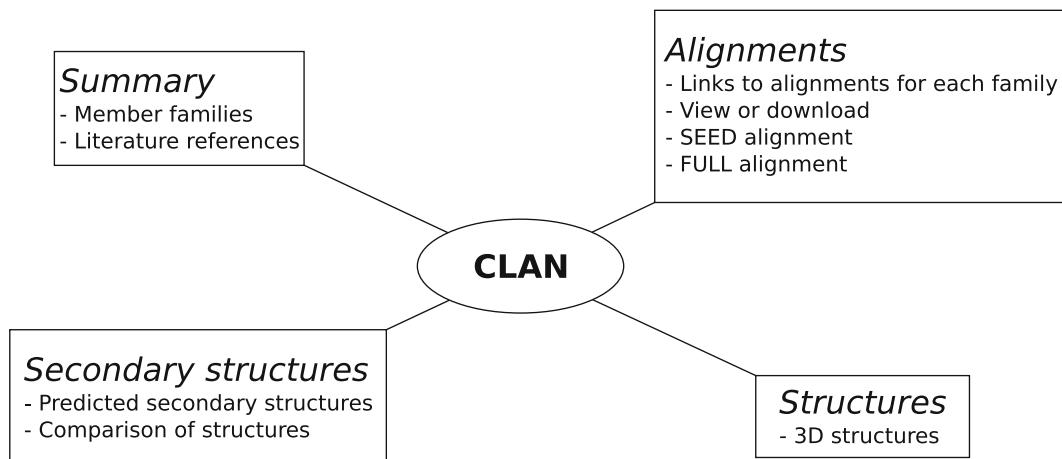
Each family is assigned a stable accession number of the form RFXXXXX, a family identifier (ID), a short description, and a ncRNA type classification. These terms are important and can be used to identify families of interest throughout the Web site. Rfam collates and generates a range of information about each family, which is found in the nine different tabs belonging to each family page (described in more detail in Subheading 3.2.1). An overview of the information available for families is provided in Fig. 1.

1.2 Clans

Rfam has implemented the use of “clans” to group-related families since Rfam release 10.0 in 2010 [8]. Families within a clan are proposed to be derived from a common ancestor or be related in sequence and structure, but are maintained as separate families due to distinct functions.

Evidence for relationship between families is established using computational tools (SCOOP (Simple Comparison Of Outputs Program) [9], PRC (Profile comparer) [10]), and curation-based approaches (literature searches and curation from specialized databases). Within a clan, the same sequence may be a member of more than one family, which is contrary to our usual curation rule that multiple families cannot annotate the same sequence region.

Clans are given stable accessions of the format CLXXXXX and identifiers, as for families. For each clan, we provide a curation summary that includes supporting literature, and we collate secondary structure images, PDB (Protein Data Bank) mappings, and alignments for each of the families in the clan. Users can access these using the four tabs provided on each clan page (Fig. 2).

**Fig. 1** Overview of family pages**Fig. 2** Overview of clan pages

2 Materials

2.1 Computational Software

The Web site is compatible with all recent browsers and makes use of several Java applets, such as Jalview for viewing sequence alignments and AstexViewer for visualizing three-dimensional structures.

2.2 Example Sequence

In this chapter, we provide a guide to using the Rfam Web site, based on an example sequence which is a short region from the genome of *Clostridium tetani* E88 (AE015927.1/1226050-1226250), annotated as belonging to family RF00162, in clan CL00012. This sequence can also be found on our ftp site: ftp://ftp.sanger.ac.uk/pub/databases/Rfam/RNA_Methods_Mol_Biol/C_tetani_AE015927_fragment.fa

```
>AE015927.1/1226050-1226250 Clostridium tetani E88, complete genome
AAATAAAAAAACACTTCGATAAGAAGTGTATAGCAAAA
TAAACTTTCTTATCTTCAGGAAATTCTGCTGGGA
GTTAGCACCTTATACCTTGATAGGTTGCTGGGT
TTCATAGGGCCAGTCCCCTCAACCGCTCTGATAAGAT
ATTTTAAATTATTTATTCAATTAAAGCA
TATTATATATTAAATTATTC
```

3 Methods

There are multiple ways to access data using the Rfam Web site; Figure 3 summarizes some of the suggested routes to finding information about a family, clan, or sequence of interest.

3.1 Sequence Searching

3.1.1 Searching with a Nucleotide Sequence

You can access the main sequence search page using the *SEARCH* link in the navigation bar, which appears at the top of every page in the Rfam Web site. The interactive sequence search can only be used for nucleotide sequence of less than 10,000 bases. You can see documentation on the restrictions on sequences under the *More...* link. If you have multiple sequences or if you prefer to retrieve results as plain text, you can use the *Batch search* facility, which runs searches offline and returns results via email.

In the interactive sequence search, the query sequence is scanned against a library of Rfam family sequences using WU-BLAST with an *E*-value threshold of 1.0 (*see Note 1*). Any matches are then searched against the corresponding CM using the hand-curated bit score gathering threshold (GA) for the family (*see Note 2*).

Paste the example sequence into the query box and press *Submit*—your results should be returned within a few seconds. The results page summarizes the number of hits and the query sequence used. You are also provided with a URL to bookmark if you wish to return to this result later.

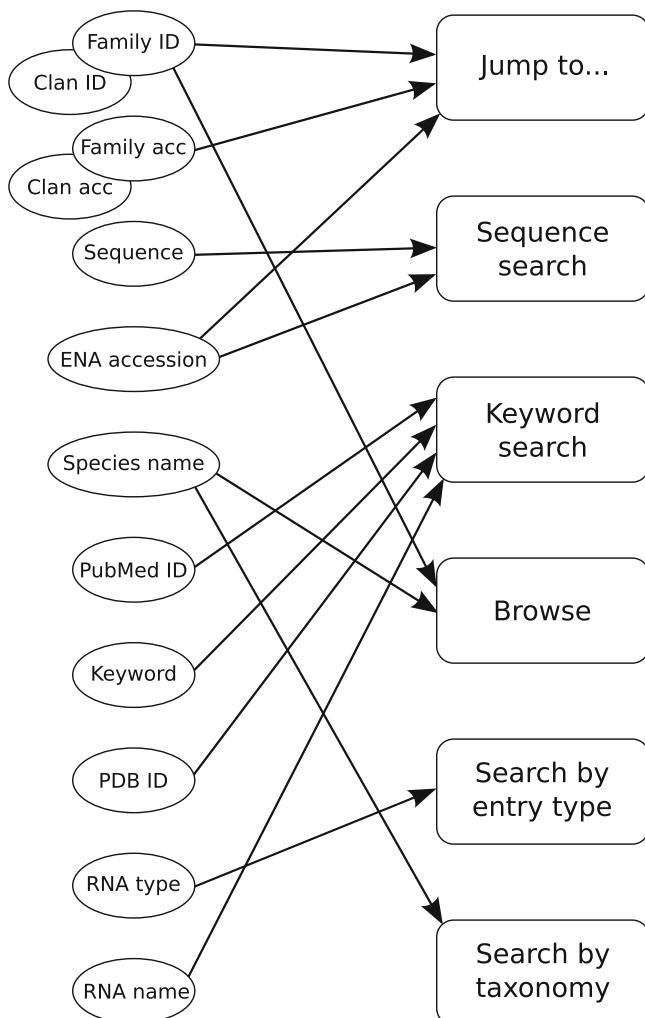


Fig. 3 Searching Rfam

Our example sequence returns matches to two families RF00162 (family ID “SAM”) and RF01725 (family ID “SAM-I-IV-variant”). Consider the coordinates of the matches (the start and end values) and the strand. The model for RF00162 aligns to nucleotides 151-48 of our query sequence on the minus (–) strand, and the RF01725 model aligns to nucleotides 153-47 also on the minus strand, so both models are aligned to the same/overlapping region of our query sequence.

If we consider the quality of these matches using the bit scores and *E*-values (see Note 2), we can see that our query sequence scores much more highly against the RF00162 model than RF01725 (103.6 and 24.2 bits, respectively, and *E*-values of 3e-26 and 0.0025, respectively). We may therefore conclude that our sequence is a close homologue of sequences in RF00162 and a distant homologue of those in RF01725.

a

Rfam matches

Show or hide all alignments.

Id	Accession	Start	End	Bits score	E-value	Strand	Show/hide alignment
SAM	RF00162	151	48	103.6	3e-26	-	Hide
#NC V							
#SS {((((((.,.,<<--<<< >>---->>,<<<-<<<< >>---->>>>,.,.,<<< >>>.,.,.,<<<							
#CM 1 cuuuauUcAGAGAGGGcgAGGGGAcuGGCCUAUGAgCCUgGAAACCwuuaauaaaaaaagaaaAUUGUCcaAUUCAGcaggaaaaaaaaccuAAGAUuaag 108							
#MATCH UCUUUAUCAAGAGGGGAGGGGGACUUGGUAGAACC CIGAACCU:i A AAA +:i:AAAAGGUUC:AA UCCAGCAGGA AA++CCUGAAAGUAGAA							
#SEQ 151 AUUUAUCAAGAGGGGGAGGGGGACUUGGUAGAACC CAGCAACCU:AUACAAAAGUAUAAGGUUCUAACUCCAGGA--AUUUCUAGAAAGAAA 48							
#PP *****9..66778*****							
SAM-I-IV-variant							
SAM-1725							
153 37 24.2 0.0025 -							
Show							

b

Rfam matches

Show or hide all alignments.

Id	Accession	Start	End	Bits score	E-value	Strand	Show/hide alignment
SAM	RF00162	151	48	103.6	3e-26	-	Show
SAM-1725	RF01725	153	37	24.2	0.0025	-	Hide
#NC V							
#SS {((((((.,.,<<--<<< >>---->>,<<<-<<<< >>---->>>>,.,.,<<< >>>.,.,.,<<<							
#CM 1 aaaaaaAUCAAGAGGGgg{ 6}*cccAccACCCGGcoaaaaa.ggCA.aGUUGGCc.aaaa.gGUUGGcc.aaaa.gGAAaaaAaagaaaaa 95							
#MATCH A+A+ :AUCAAGGGGG GGG: :CCCC:CAACCU:i A++AA :i:i:A AGGGGU+ GA :: IC + AA Gi AA AAA+ A+AAA+							
#SEQ 153 AUUUAUCAAGAGGGGGAGGGGGACUUGGUAGAACC CAGCAACCU:AUACAAAAGUAUAAGGUUCUAACUCCAG---caga-AAUUUCUAGAAAGAAAAGUUUAUUUU 37							
#PP *****88886...6.6888*****55555699998*****37776554...,44355.333444544446667777888899999999							

Fig. 4 Sequence search results

After running a search, you can view the alignment of your query sequence to the model and the consensus sequence, as taken from the output of CMalign (Fig. 4a), by clicking on the *Show* button in the far right column. There are six lines in this alignment: #NC, #SS, #CM, #MATCH, #SEQ, and #PP. These few lines condense large amounts of information from the Infernal output, and users are referred to the Infernal manual for a full and detailed explanation of this and all possible characters [11].

Briefly, in our example case, the #SS line shows the predicted secondary structure of the query sequence in dot bracket notation. The different closed bracket notations (“(”) and “<>”) are used by Infernal to indicate different structural features. In this example the model contains three stem loops closed by another stem of base pairing. Finally, “_” characters mark hairpin loops, and “,” characters mark single-stranded residues in multifurcation loops.

The #CM line shows the consensus sequence of the covariance model. The highest scoring residue at each position from the seed alignment is shown. Uppercase residues are highly conserved; lowercase residues are weakly or not conserved. In the example model, note the level of sequence conservation in the base-paired regions and the increased variation in the last two loop regions. The #SEQ line shows our query sequence and coordinates as aligned to the model. “-” characters indicate deletions in the query sequence with respect to the model. In the example sequence, these occur in the single-stranded, less well-conserved regions.

The #MATCH line is the important indicator of alignment bit score. Each position in the alignment is scored according to the consensus sequence. For base-paired regions, if the observed pair in the query sequence is the highest-scoring possible pair, both

residues are shown in uppercase; if a pair has an acceptable compensatory pair, both residues are annotated by “:” characters. Spaces indicate positions with a negative contribution to the alignment score. A similar convention is implemented for single-stranded regions, with the highest scoring residues noted in uppercase, acceptable scoring residues indicated with “+,” and spaces indicating negatively scoring positions. Looking at our example, we can see that the majority of positions are scored with uppercase, or positively scoring characters, and our sequence is a very good match to the RF000162 model.

The #NC line marks the positions of significant negatively scoring pairs with a “v” character. In our example, base pairing between residues 151 and 48, which is a predicted A:A base pairing, is flagged up.

The #PP line is used to annotate the expected accuracy of each aligned residue. “*” characters indicate 95–100 % accuracy, 9 indicates 85–95 % accuracy, and so on with 0 indicating 0–5 % accuracy. You can see in the example the majority of the sequence is aligned with 95–100 % accuracy, and the lower confidence region occurs in the less well-conserved loop region.

If we now view the alignment of our query to the RF01725, we can clearly see that there is much poorer confidence in the alignment (#PP line) and poorer scoring in the #MATCH line (Fig. 4b).

3.1.2 Searching Using a Sequence Accession

Rather than searching for matches to ENA sequences, the Rfam Web site allows you to quickly look up the sequence using its ENA accession. Note that the sequence must exist in the ENA version from which the current Rfamseq derives. You can use the *Look up sequence* box under the sequence search form, or you can use the *Jump to* field found on every tabbed page in the Rfam Web site (*see Notes 3 and 4*).

Using the *Look up sequence* box, we can see that there are 181 annotations in Rfam for our example sequence, AE015927.1 (this number is for the full sequence rather than the fragment that we searched in Subheading 3.1.1). The results are displayed in a tabular format, ordered by default according to the order of coordinates on the query sequence. You can sort the table on the different columns of data by clicking on the table headers (*see Note 5*).

Browsing down the results to the match with coordinates 1226200-1226097, we find the SAM annotation that we identified by searching the sequence above. If we now sort the table by Rfam familyID and scroll down to the SAM riboswitches, we can see that Rfam annotates four different instances of the SAM family (RF00162) on this sequence. Note the bit scores and coordinates of the different annotations (three are annotated on the minus strand and one on the positive).

3.2 Finding Family Information

3.2.1 Family Pages

Entering the accession RF00162 or the family ID SAM in the *Jump to* box will take us directly to the summary tab of the family page for this specific family (*see* Fig. 1). The accession, identifier, and description are displayed prominently at the top of all family pages. If the family is in a clan, the other members of the clan will also be displayed in the header of the summary page. We can see immediately that family RF00162 belongs to clan CL00012 (clan ID “SAM”), which contains two other families: SAM-I-IV-variant (RF01725) and SAM-IV (RF00634).

Since 2007, Rfam has linked every family to the article in Wikipedia that best describes that family [12]. The Wikipedia article is displayed in the main body of the *Summary* tab and should provide the user with preliminary background information, links, and literature sources that are relevant to this ncRNA family.

If you browse through the different tabs (Fig. 1), you can collate pertinent information on the family itself and the sequences in the full alignment, such as, under the *Alignments* tab, the number of sequences in the seed and full alignments (433 and 4,757, respectively). In the same tab, you can view or download the seed or full alignments and sequences in several formats, with the option of having sequences annotated using sequence accessions or species identifiers.

You can browse multiple graphical representations of the predicted secondary structure and sequence conservation for the seed, under the *Secondary structures* tab. In addition to the stem-loop representations, we also provide R-chie diagrams [13] and a link for viewing the structure interactively using the VARNA applet [14].

You can find the predicted phylogenetic trees for seed and full alignments in the *Trees* tab, while under the *Species* tab, you can browse, download, and align different taxonomic subsets of sequences from the full alignment (*see* Subheading 3.2.2).

Any PDB sequence mappings we have made to this family can be viewed in the *Structures* tab. Note that multiple PDB sequence regions may be mapped to the same ENA sequence region. In the case of RF00162, for example, we map 20 fragments from different PDB sequences and chains onto nine regions from five different ENA sequences. You can view these sequence fragments in the context of the three-dimensional PDB structure using the AstexViewer applet [15], which allows you to manipulate and rotate the structure to view the sequence region and toggle between different views of surface transparency.

The *Database references* tab provides external database cross-references, such as links to Gene Ontology (GO) and Sequence Ontology (SO) [16, 17]. Under the *Curation* tab, you will find important information about the curation of the Rfam family, including our family type classification, family source references, authors, and family building parameters, and you can also download the covariance model. Note that our example family has been

given the type classification “*Cis*-reg; riboswitch.” Also under the *Curation* tab, you will see the GA for this family is 44 bits. The bit score from our earlier query sequence search (Subheading 3.1.1) was 103.6 bits, indicating that our query sequence is well above the GA and a very good homologue of this family. At the bottom of the curation page, you can also view the distribution of all of the bit scores we obtain for every sequence in this family when we search Rfamseq. Again you can see that our query sequence, with a bit score of 103.6, lies to the far right of this range, indicating a very good match to the model.

Browsing through the links on the *Summary page* to the two other families in this clan, SAM-I-IV-variant (RF01725) and SAM-IV (RF00634), you will find they are also classified as *Cis*-reg; riboswitch. If you view the distribution of bit scores for RF01725 under the *Curation* tab, you will see that the match between this family and our query sequence (see above; Subheading 3.1.1) is 24.2 bits, which is just above the trusted bit score cutoff.

Following the link to the clan page for CL00012 (or using this clan accession in the *Jump to* box) will take you to the main clan page, where we also provide the curation summary, with appropriate source references and a collation of the alignments and PDB mappings for all three families in this clan. You can use the *Secondary structures* tab here to view the seq-cons stem-loop secondary structure images for any two clan members, displayed side by side.

3.2.2 Species Distribution and Downloading a Subset of Sequences

Two alternative representations of the species distribution in the full alignment are provided under the *Species* tab in the family page: the “sunburst” visualization (the default) and an interactive tree representation. Only the sunburst tool is described here.

Briefly, the sunburst contains segments colored according to taxonomic kingdom. Details on how it is generated can be found under the *More...* link. Looking at the sunburst for our example family RF00162, we can see immediately that it is predominantly bacterial, with a significant number of unclassified sequences. As the mouse is moved over the nodes within the sunburst, a tooltip is displayed, showing the relevant taxonomic level, the number of sequences, and the number of species for that node. The full taxonomic lineage of the current node is displayed in the sunburst control panel at the top right. The size of each segment of the sunburst can be scaled according either to the number of species or the number of sequences in the family within this taxonomic node. Toggle between these two options using the radio button to see the difference this makes to the representation.

Using the sunburst, we can also select and download sequences in the family. We can select sequences from our example species, *Clostridium tetani*, by clicking the *Clostridium tetani* segment in the outer ring (species level). You can zoom into the sunburst

using the size slider to make it easier to find the right node. When you identify *C. tetani*, the tooltip will indicate “Clostridium tetani [species], 1 sequence, 1 species.” After clicking to select the node, the segment turns red, and the selected sequences are registered at the bottom of the control panel on the right. You can download these sequences in FASTA format or have them aligned to the CM. Select *Align to CM* for our *C. tetani* selection and save the resulting Stockholm formatted alignment of the four SAM annotations indicated in Subheading 3.1.2.

The sunburst tool allows you to choose any arbitrary set of species: you can select or exclude sequences for any taxonomic division or combination of divisions you choose. For example, you may choose to download all of the regions for all of the Clostridia taxonomic class (425 sequences from 265 species) or for all of the phylum Firmicutes (2,218 sequences from 677 species). Alternatively, you could choose to select all of the phylum Firmicutes excluding all the Clostridia apart from *C. tetani*.

For comparison, if you browse to the sunburst for family RF01725, you can clearly see the difference in the taxonomic distribution between these two families. If you locate the Firmicutes, you will see that there are only five sequences from five species present in this phylum in RF01725.

3.3 Text Searching

3.3.1 ncRNA Type (Type Search)

Under the main *SEARCH* tab, select the *Entry type* tab, you are presented with a hierarchical representation of the Rfam family types. There are three top level categories to this structure, under which all families are classified: gene, *cis*-reg, and intron. Use the *More...* link to read more details about our classifications and what they mean within Rfam.

Our example family is of the type “*Cis*-reg; riboswitch;.” This means that our family is a *cis*-regulatory element of the subtype riboswitch. To view all the families that Rfam annotates in the “*Cis*-reg; riboswitch;” category, simply check the box next to riboswitch and press *Submit*. In the results page, you will see we have 26 families in this category. We already know our example family RF00162 is in a clan with two others in this list, but some of these other riboswitch families may also be of interest. This simple list of checkboxes is very flexible and allows you to select/deselect any combination of ncRNA categories depending on what data you are looking for.

3.3.2 Annotations by Species, Class, and Phylum (Taxonomy Searching)

Under the main *SEARCH* link, select the *Taxonomy* tab. The operation of this taxonomy search box is relatively complex and is explained in more detail under the *More...* link. Briefly, to view all of the family annotations that Rfam makes to *Clostridium tetani* (regardless of sequence accession), you can simply add *Clostridium tetani* to the search box, and you will be taken to a summary of all families that we annotate in any sequence from this species. You can also check the lower box on this page to identify families that

are unique to this species or taxonomic group. In the case of *Clostridium tetani*, we annotate 32 different families. Note that this search will only identify the families which we have identified in this species, but will not list the individual or multiple annotations of each family or each sequence. You would need to peruse each family to see the individual annotations to this species. See Subheading 3.2.2 for downloading taxonomic subsets of sequences for a single family or use the Rfam BioMart (Subheading 3.5).

You can also widen the search to a more generic taxonomic level and, for example, use the taxonomic class “Clostridia” in this search box. This will identify all of the Rfam families in any species that belong to this taxonomic class (in this case, we find 93 different families). Increasing the taxonomic range to phylum level “Firmicutes,” we find 180 families are identified in species in this phylum. Again you should bear in mind that this result will not list the individual annotations per species, and you would need to pursue individual families to identify these species and sequences.

3.3.3 Genome Annotation, by Species Name or by Sequence Accession (Jump to and Browse)

If you know the sequence accession of your genome of interest (e.g., the *Clostridium tetani* ENA accession is AE015927.1), you can use the *Jump to* box to go directly to genome pages for that species (in the case of higher eukaryotes, you would use a chromosome sequence accession, e.g., CM000209.2, for mouse chromosome 1).

If you do not know the genome accession or chromosome accession for your species of interest, you can browse the list of complete genomes that are annotated by Rfam, via the *BROWSE* link in the navigation bar (see Subheading 3.4).

The genome page gives the total number of annotations (in this case 181), the number of different families (32) found on sequences from that genome, and some basic information and statistics on the genome itself. To view the annotations, visit the chromosome tab. Since this is a bacterial genome and a complete sequence, there is only one chromosome sequence. In the case of higher eukaryotes, you would find the multiple chromosomes listed here separately. From the chromosome tab, you can choose to view the annotations or download them in GFF format.

3.3.4 PubMed ID, Author, PDB Structure, and Others (Keyword Search)

All of our curated family and clan data are indexed, so you can use multiple criteria to identify families of interest. The *Keyword search* box appears in the top right corner of the every page in the Rfam Web site, as well as under *Keyword* tab in the main search page. The keyword search allows you to perform a simple text search against the textual data from several different sections of the Rfam database:

- Rfam: accessions, identifiers, descriptions, type, comment lines from Rfam families
- Wikipedia: text from Wikipedia articles linked to Rfam families
- Literature: literature reference titles, authors, and PubMed identifiers

- PDB: sequence mappings
- Clans: accessions, identifiers, and family associations

If your keyword is found for a family across any of these categories, the family will be listed in the results. Because of the volume and diversity of textual information in Rfam, a text search can return a large number of hits. Furthermore, the keyword search will add wildcards to the end of your keyword, so the shorter your keyword/string, the wider the possible set of matches and the larger the number of matching families. This may result in matches that do not interest you (sensitivity versus specificity), but it can also be an extremely useful way to quickly identify families of interest. Results are ordered according to the number of sections of the database in which the search string appears, so that families for which the string appears in multiple sections will be listed first, giving a crude indication of the likely relevance (or ubiquity) of the search terms.

In the curation tab of the family page for our example family, RF00162, you will see that “Henkin T” is credited as one of the authors of the family. We can find other families that may be relevant by using this name as a search string in the keyword search. Searching with “Henkin” will return seven families, of which the top result is RF00162, since the author’s name appears in three sections of the database for that family (*see Note 6*). However, there are four other families that have a match in the “literature” section, and if you peruse the family information for each of these families, you will find this author under the *Database references* tab. Two of these four families are also riboswitches. In the remaining two families, the reference to this author can be found only in the associated Wikipedia article (most likely in the references or further reading section).

As another example, we may wish to look up families/sequences relating to a known structure in the PDB, e.g., 3NPB (the crystal structure of YbxF bound to the SAM-I riboswitch aptamer). You can use this PDB accession as a keyword search term, and in this case, you will be taken directly to the family RF00162 since that is the only family that maps to this specific PDB sequence. In general, if there is only a single matching family in the results of a keyword search, you will be redirected to the family page immediately, rather than being presented with a table of results with only a single row.

3.4 Browsing

As an alternative to directed searching, you can use the *BROWSE* link in the navigation toolbar to view our families listed under five different criteria: families (listed by family ID), via annotated genomes (listed by species name), as clans (listed by clan ID), families with structures (listed by family ID), and families with Wikipedia articles (listed by Wikipedia article title).

3.5 BioMart

BioMart is an open-source data management system [18], providing a powerful Web-based interface to complete databases such as Rfam. The Rfam BioMart can be accessed via the link in the navigation toolbar. The BioMart provides a user-friendly means of performing sophisticated queries on our database and offers much more flexibility than the searches which are built into the Rfam Web site. All of the text search examples above could equally well be performed using the BioMart filters, with the advantage that you have greater control over the content and format of the results.

3.6 RESTful Interface

We provide a RESTful interface to enable users to interact programmatically with the services provided by Rfam. A detailed description of this is beyond the scope of this chapter, but full documentation, including sample queries and scripts, is available via our help pages.

3.7 FTP

Our ftp site (<ftp://ftp.sanger.ac.uk/pub/databases/Rfam/>) contains all of our release data and database files for the current and previous releases of Rfam. It is important to note that many of these files are very large. A detailed description of the ftp site can be found in the Rfam Web site help documentation.

4 Notes

1. This chapter is based on Rfam release 11.0. In subsequent Rfam releases and with the adoption of Infernal 1.1, the BLAST prefilters will be replaced with faster and more sensitive HMMER3-based filters.
2. The bit score (also known as the log-odds score) is the quality score generated for every sequence that the Infernal software tries to match against the model. It is the log-odds ratio of the probability that the sequence was generated by the CM versus a random model of background sequence. Very simply, it is a measure of how well the sequence matches the model, with higher bit scores indicating higher confidence in the match to the model. When searching against Rfamseq and building a family, we manually curate a bit score threshold (termed the gathering threshold or cutoff (GA)) that in our opinion best discriminates between “real” homologues and the background distribution of false hits in the database. Sequences scoring above this threshold are considered members of the family, those below it are excluded.
3. Each Infernal sequence search and resultant bit score has an associated *E*-value. The *E*-value is the statistical significance of the hit, i.e., the number of hits expected to score this high in a database of this size (measured by the total number of nucleotides) if the database contained only nonhomologous random sequences [11].

4. The *Jump to* box: this field appears on the homepage and in the sidebar of all tabbed pages, such as the family, clan, and genome pages. It provides quick access to the data pages in the site, allowing you to enter any of a wide range of accessions or identifiers and be taken directly to the appropriate page. This can be extremely useful when you know exactly what you are looking for. The *Jump to* field will only accept valid, accurate identifiers. It does not perform a search like the keyword search and does not accept multiple values or search terms.
5. In cases where the ENA sequence is a whole genome sequence, the *Jump to* box will perform differently from the *Look up sequence* box, directing you to the page for the genome rather than simply collating the list of annotations (see Subheading 3.3.3 for using the *Jump to* box to view a genome). Furthermore, the *Jump to* box will perform differently depending on whether the sequence accession version is used. The example AE015927.1 will take you to the genome pages, while AE015927 will take you to the collated annotations.
6. The majority of tabular data is presented in tables that are sortable by the data in each of the different columns. Clicking on a column header will perform an alphanumerical sort on all of the data in the table based on the data in the chosen column.
7. If you include the author's initial in this search using "Henkin T" as opposed to "Henkin," you will obtain the same set of seven families in the results, but with slightly different scoring profiles across the sections of the database.

Acknowledgments

The Rfam database is funded by the Wellcome Trust (grant number WT077044/Z/05/Z).

References

1. Cochrane G, Alako B, Amid C, Bower L, Cerdeno-Tarraga A, Cleland I, Gibson R, Goodgame N, Jang M, Kay S et al (2013) Facing growth in the European Nucleotide Archive. *Nucleic Acids Res* 41(Database issue):D30–D35
2. Torarinsson E, Lindgreen S (2008) WAR: Webserver for aligning structural RNAs. *Nucleic Acids Res* 36(Web server issue): W79–W84
3. Zuker M (2003) Mfold web server for nucleic acid folding and hybridization prediction. *Nucleic Acids Res* 31(13):3406–3415
4. Nawrocki EP, Kolbe DL, Eddy SR (2009) Infernal 1.0: inference of RNA alignments. *Bioinformatics* 25(10):1335–1337
5. Gardner PP (2009) The use of covariance models to annotate RNAs in whole genomes. *Brief Funct Genomic Proteomic* 8(6): 444–450
6. Nawrocki EP, Eddy SR (2013) Computational identification of functional RNA homologs in metagenomic data. *RNA Biol* 10(7):1170–1179
7. Burge SW, Daub J, Eberhardt R, Tate J, Barquist L, Nawrocki EP, Eddy SR, Gardner

- PP, Bateman A (2013) Rfam 11.0: 10 years of RNA families. *Nucleic Acids Res* 41(Database issue):D226–D232
8. Gardner PP, Daub J, Tate J, Moore BL, Osuch IH, Griffiths-Jones S, Finn RD, Nawrocki EP, Kolbe DL, Eddy SR et al (2011) Rfam: Wikipedia, clans and the “decimal” release. *Nucleic Acids Res* 39(Database issue):D141–D145
 9. Bateman A, Finn RD (2007) SCOOP: a simple method for identification of novel protein superfamily relationships. *Bioinformatics* 23(7):809–814
 10. Madera M (2008) Profile comparer: a program for scoring and aligning profile hidden Markov models. *Bioinformatics* 24(22):2630–2631
 11. Infernal User Guide. [ftp://selab.janelia.org/
pub/software/infernal/Userguide.pdf](ftp://selab.janelia.org/pub/software/infernal/Userguide.pdf)
 12. Daub J, Gardner PP, Tate J, Ramskold D, Manske M, Scott WG, Weinberg Z, Griffiths-Jones S, Bateman A (2008) The RNA WikiProject: community annotation of RNA families. *RNA* 14(12):2462–2464
 13. Lai D, Proctor JR, Zhu JY, Meyer IM (2012) R-CHIE: a web server and R package for visualizing RNA secondary structures. *Nucleic Acids Res* 40(12):e95
 14. Darty K, Denise A, Ponty Y (2009) VARNA: interactive drawing and editing of the RNA secondary structure. *Bioinformatics* 25(15):1974–1975
 15. Hartshorn MJ (2002) AstexViewer: a visualisation aid for structure-based drug design. *J Comput Aided Mol Des* 16(12):871–881
 16. Ashburner M, Ball CA, Blake JA, Botstein D, Butler H, Cherry JM, Davis AP, Dolinski K, Dwight SS, Eppig JT et al (2000) Gene ontology: tool for the unification of biology. The Gene Ontology Consortium. *Nat Genet* 25(1):25–29
 17. Mungall CJ, Batchelor C, Eilbeck K (2011) Evolution of the Sequence Ontology terms and relationships. *J Biomed Inform* 44(1):87–93
 18. Zhang J, Haider S, Baran J, Cros A, Guberman JM, Hsu J, Liang Y, Yao L, Kasprzyk A (2011) BioMart: a data federation framework for large collaborative projects. *Database* 2011: bar038

Chapter 23

ASPicDB: A Database Web Tool for Alternative Splicing Analysis

**Mattia D'Antonio, Tiziana Castrgnanò, Matteo Pallocca,
Anna Maria D'Erchia, Ernesto Picardi, and Graziano Pesole**

Abstract

Alternative splicing (AS) is a basic molecular phenomenon that increases the functional complexity of higher eukaryotic transcriptomes. Indeed, through AS individual gene loci can generate multiple RNAs from the same pre-mRNA. AS has been investigated in a variety of clinical and pathological studies, such as the transcriptome regulation in cancer. In human, recent works based on massive RNA sequencing indicate that >95 % of pre-mRNAs are processed to yield multiple transcripts. Given the biological relevance of AS, several computational efforts have been done leading to the implementation of novel algorithms and specific specialized databases. Here we describe the web application ASPicDB that allows the recovery of detailed biological information about the splicing mechanism. ASPicDB provides powerful querying systems to interrogate AS events at gene, transcript, and protein levels. Finally, ASPicDB includes web visualization instruments to browse and export results for further off-line analyses.

Key words Alternative splicing, Transcriptome, Transcription, Alternative isoforms, Database, Web tool, Transcript prediction, Bioinformatics, Transcriptomics

1 Introduction

There has been a growing interest toward alternative splicing (AS) in recent years, as demonstrated by large-scale studies revealing that the majority (>95 %) of human multi-exonic genes undergo alternative splicing events [1–3] and showing AS as the main molecular phenomenon responsible for the increased complexity of eukaryotic transcriptomes. Splice sites have a great biological value because their alternative usage can be used as a robust molecular signature of biodiversity, human pathologies including cancer, and to trace cell developmental stages [4].

To investigate AS in eukaryotes, we have developed ASPicDB [5, 6], a specialized database designed to store reliable alternative splicing annotations predicted using ASPic [7, 8] algorithm and its recent evolution PIntron [9].

ASPicDB has also been conceived to handle AS biological information in different ways. Indeed, customized web interfaces enable complex queries to retrieve AS annotations at transcript, gene, and protein levels. Several interactive tables and graphical overviews allow a punctual inspection of results. Database search is also improved through sequence-based queries using BLAST.

Along the text, the term ASPicDB will be interchangeably used to describe both the relational database and the related web application.

2 Materials

ASPicDB is freely available at the following address: www.caspur.it/ASPicDB.

This resource is a relational database built with MySQL and featuring an ad hoc web interface that allows the user to fully retrieve/visualize any information collected in the database. Web applications are particularly convenient to wrap databases, since they can significantly simplify and enhance the query and visualization phases. A customized web interface can help the user to query the dataset without worrying about the data organization and other technical issues. Moreover, a graphical user interface can show the results in a more organized and clear fashion and allows the integration of data with further information, statistics, graphical representations, annotations, cross-linking, and other useful tools to help the user to browse and interpret the dataset.

Since ASPicDB is available through a web interface, the user only needs a regular computer or laptop with an updated web browser to run queries and visualize results. Specific dataset can be exported as flat files (*see* Subheading 3.5 below) for further analysis and cross-validation with different tools. In this case a standard text editor could be required. Depending on file size, text manipulation could be time and memory expensive. For text files greater than 10 MB we recommend to avoid spreadsheet-based tools.

3 Methods

ASPicDB stores AS annotations predicted using ASPic [7, 8] algorithm and its recent evolution PIntron [9]. Such algorithm deduces all nonredundant sets of alternative isoforms for a given gene, providing as input its genomic sequence and related expressed sequences (mostly ESTs and full-length cDNAs).

Data contained in ASPicDB can be divided in two main subsets: predictions calculated by the ASPic/PIntron algorithm and annotations associated with predictions.

ASPic/PIntron predictions are based on multi-alignments of expressed sequence tags (ESTs) [7] from UniGene clusters [10] of a variety of organisms (*Homo sapiens*, *Mus musculus*, *Rattus norvegicus*, *Bos taurus*, *Gallus gallus*, and *Arabidopsis thaliana*). Predicted full-length isoforms are automatically annotated employing several external databases such as GenBank [11], RefSeq [12], UniProtKB/Swiss-Prot [13], Protein Data Bank (PDB) [14], Pfam [15], and UniRef90 [16].

Current ASPicDB release also integrates info from the SpliceAid-F database [17] collecting known splicing factors and their binding sites.

ASPicDB can be queried through three different search forms: (1) the *simple search* form allowing basic queries on the most common fields, such as specific genes; (2) the *advanced search* form designed to perform complex queries on all available predictions and annotations, allowing the interrogation of dedicated tracks at gene, transcript, exon, intron, or protein levels; and (3) the BLAST [18] form permitting sequence queries on the entire collection of transcripts or proteins.

We provide usage guidelines for each form in the following subsections.

3.1 Simple Search Form

The simple search form comprises three different panels as shown in Fig. 1. The user can decide to perform a query providing a gene ID or a gene ontology term or whatever keywords of interest.

a

Search for Gene	
Organism:	Human
Accession:	Hugo NSMCE1 More
<input type="button" value="Search!"/>	

b

Search for Keywords	
Keyword:	apoptosis
<input type="button" value="Search!"/>	

c

Search for Gene Ontology	
Gene Ontology Id Search:	0015248
Term Search:	All calcium ion transport
<input type="button" value="Search!"/>	

Fig. 1 Overview of the simple query form

In the first panel, the following gene identifiers are supported:

- HGNC [19] (e.g., NSMCE1)
- UniGene [10] (e.g., Hs.481720)
- RefSeq [12] (e.g., NM_152713)
- Entrez Gene [20] (e.g., 3248)
- MIM [21] (e.g., 202300)

The central panel allows the selection of one or more genes matching specific query keywords. The match is mainly restricted to gene descriptions extracted from the GenBank database.

Finally, the third panel allows the interrogation of genes based on certain ontology terms [22] related to cellular components, molecular functions, or biological processes. The form admits also term substrings (such as *calcium ion*) in association with one of the three already mentioned semantic areas.

Search Results are presented in tabular form including different fields associated with the found gene such as accession ID, description, organism, genomic coordinates, transcript number, alternative proteins, and total number of splicing events. A specific link allows the access to the graphical visualization about predictions and annotations.

3.2 Advanced Search Form

The advanced search form allows the combination of different search criteria in order to increase the specificity of the interrogation. Five different panels are available to perform queries on different biological aspects, and each panel contains a set of common and specific features.

Every module includes the following three fields: *Organism*, *Accession*, and *Coordinates*. Only the *Organism* field is mandatory, and, actually, queries on multiple organisms are not feasible. Through the *Coordinates* fields, users can select genomic regions in which the search can be performed.

In addition to all common filters, in the gene section (Fig. 2), users can query:

- Keywords matching gene descriptions (e.g., “apoptosis”)
- RNA regions in which splicing events are located (i.e., 5' UTR; CDS; 3' UTR)
- Type of splicing events (e.g., exon skipping, intron retention, etc.)
- Specific donor-acceptor splice sites (e.g., GT-AG)
- Intron class (e.g., U2, U12)
- Number of alternative proteins and transcripts (range)
- Number of independent splicing events (range)

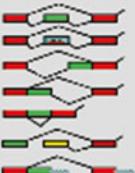
Search Gene entries fulfilling one or more of the following criteria:			
Organism	<input type="button" value="Human"/>		
Accession	<input type="button" value="Hugo"/>	<input type="text"/>	
Keyword	<input type="text"/>		
Coordinates	<input type="text"/> Chr:	<input type="text"/> Start:	<input type="text"/> End:
Alternative splicing event localized in	<input type="radio"/> 5'UTR <input type="radio"/> CDS <input type="radio"/> 3'UTR <input checked="" type="radio"/> All		
Type of splicing event	 <ul style="list-style-type: none"> <input type="checkbox"/> Exon Skipping <input type="checkbox"/> New Exon <input type="checkbox"/> Alternative 3' End <input type="checkbox"/> Alternative 5' End <input type="checkbox"/> Intron Retention <input type="checkbox"/> Alternative Initiation <input type="checkbox"/> Alternative Termination 		
Donor-Acceptor site	<input type="text"/> Donor: ...	<input type="text"/> Acceptor: ...	
Intron Class	<input type="text"/>		
Number of alternative transcripts	<input type="text"/> ≥	<input type="text"/> ≤	
Number of alternative proteins	<input type="text"/> ≥	<input type="text"/> ≤	
Number of independent splicing events	<input type="text"/> ≥	<input type="text"/> ≤	
NMD-related transcripts	<input type="checkbox"/> PTC+	<input type="checkbox"/> PTC-	
Search for genes with specific expression profile in normal/tumor cells of a given tissue			
Expression pattern: Genes	<input type="text"/>		
From the following tissue(s)	<input type="text" value="All Tissues"/> <input type="text"/>		
Select a p-value threshold:	<input type="text"/>		

Fig. 2 Overview of the advanced query form (Gene Tab)

- Presence of nonsense-mediated decay (NMD) transcripts

In contrast, in the transcript sections, users can search alternative isoforms of a certain gene. Although several fields are shared with the gene search form, the transcript section allows also to:

- Retrieve a specific transcript by its signature [23].
- Filter by the presence of annotated poly(A) tails and poly(A) signals (according to the PolyA_DB [24] standard).
- Filter by CDS, 5' UTR, and 3' UTR length.
- Filter by the number of exons.

The exon retrieval form enables the user to spot single exons that match to a set of personalized criteria, such as:

- Minimal and maximal exon length
- Exon class (initial, internal, terminal)
- Features of flanking introns (donor-acceptor sites, intron class)
- Number of ESTs supporting flanking introns
- Presence of poly-A tail or poly-A signal annotated in the exon

To retrieve particular splice sites (introns), the user can customize the following filters:

Specific donor-acceptor sites

- Intron class (U12, U2, not determined)
- Number of ESTs supporting the intron
- Intron length (interval)
- Minimal and maximal repeat site length in the intron sequence
- Presence of ambiguous splicing sites (i.e., the intron boundaries cannot be precisely defined due to an ambiguous alignment between cDNA and gDNA)

Finally, the protein search form is divided in two sections: the first one allows to search proteins with structural or functional properties; the second one is able to look for genes encoding alternative proteins matching user-defined criteria.

The first section allows to query for:

- Pfam class
- Protein class (e.g., globular, transmembrane)
- Number of functional domains
- Cellular localization for globular proteins (e.g., nucleus, cytoplasm, mitochondria)
- Presence of GPI anchors and signal peptides

The second panel contains a set of checkboxes to select features requested to be different in the alternative encoded proteins for the resulting genes. For instance, by selecting “subcellular localization,” the application will fetch all genes that encode for alternative proteins that differ in subcellular localization (e.g., some proteins localized in nucleus and others in cytoplasm).

3.3 BLAST Search Form

An integrated BLAST search enables sequence queries to interrogate the full set of ASPicDB alternative transcripts and proteins. Alignment results are listed in a table reporting the five most significant matches, in which every row provides additional information such as the matched transcript identifier, the sequence

query length, and the matching score. Results are also linked to the corresponding ASPicDB entries.

This facility could be very useful in combination with experimental techniques to investigate specific transcripts or embedded features in diverse conditions.

3.4 Graphical Visualization of Results

ASPicDB allows graphical representation of results facilitating the inspection of annotations in an easy and intuitive way.

Query results, obtained through the above-described forms, are linked to detailed web pages showing all available information for each extracted entry (i.e., gene, transcript, protein, exons, and introns). The main result page contains eight subsections. Of these, three provides a different dynamic image with mouseover tooltips.

The *Gene Information* table contains an overview of genomic characteristics for the gene under investigation (accession IDs, description, coordinates, the number of ESTs contained in the UniGene cluster). Furthermore, the number of transcripts, proteins, and splicing events is also summarized along with a set of web links to orthologous genes (processed in ASPicDB for different organisms), input and output raw files, and additional public data from widespread databases (e.g., ASAP2 [25], ASTD [26], AceView [27]).

In the *Gene Structure view*, the user can visualize the predicted gene structure. The image displays predicted introns and exons other than the reference transcript from RefSeq database.

Figure 3 reports an example of structure view for the KRT82 (keratin 82) human gene. A further but more complex example is for the TP63 human gene shown in Fig. 4, because it contains several alternative transcripts. Introns, as the number 13 in Fig. 4,

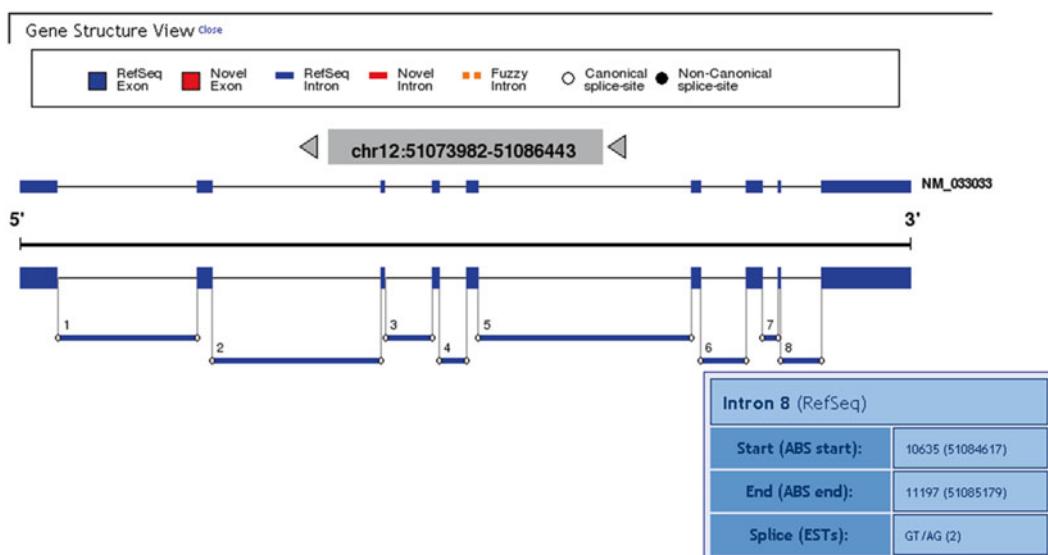


Fig. 3 Structure view for the human keratin 82 (KRT82) gene

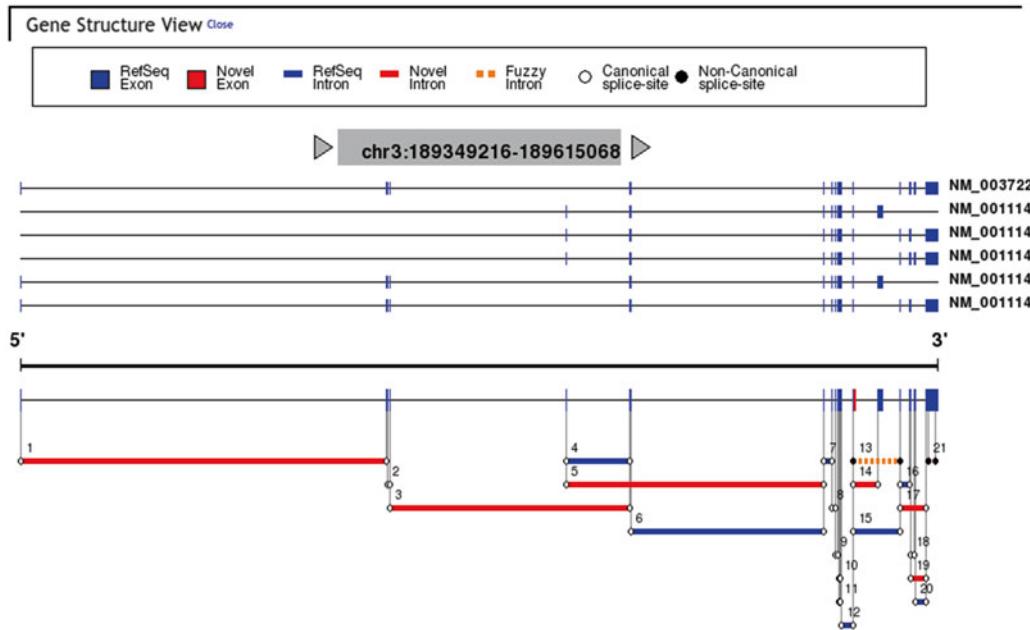


Fig. 4 Structure view for the human tumor protein p63 (TP63)

that do not have canonical splicing sites (black dots at ends) are represented by orange dotted lines and are labeled “fuzzy introns.”

The *Predicted Transcripts* section provides a graphical representation of the predicted transcripts. Only the most reliable introns (i.e., those supported by a perfectly aligned EST with canonical donor and acceptor sites or those supported by two or more ESTs) are used for the assembly process. Figure 5 shows a “Transcript View” snapshot for the ADSL human gene where exons (thick-colored strips) and introns (black junction regions) are clearly identifiable, along with other functional regions such as 5' UTR (yellow strips) and 3' UTR (sky blue). Interestingly, mass-spectrometry identified peptide sequences [28] supporting specific coding exons or splice junctions are represented in the top of the Transcript View as small green boxes.

The mouseover on specific exons or introns activates the tooltip with additional information, while a click opens a popup page containing the genomic sequence of the selected element. Furthermore, two icons to the right of each transcript are linked to the genomic sequence of the transcript and the corresponding protein (if the transcript has a coding sequence).

A 3' terminal arrow marks polyadenylated transcripts that may also contain a poly(A) site (AAUAAA or its accepted variants), and a hyphen marks the position of mapping CAGE [29] tags which identify transcription initiation sites. In Fig. 6, five of the alternative transcripts of the BRAF gene are shown. Since the fourth doesn't contain an annotated CDS, its exons are displayed in gray

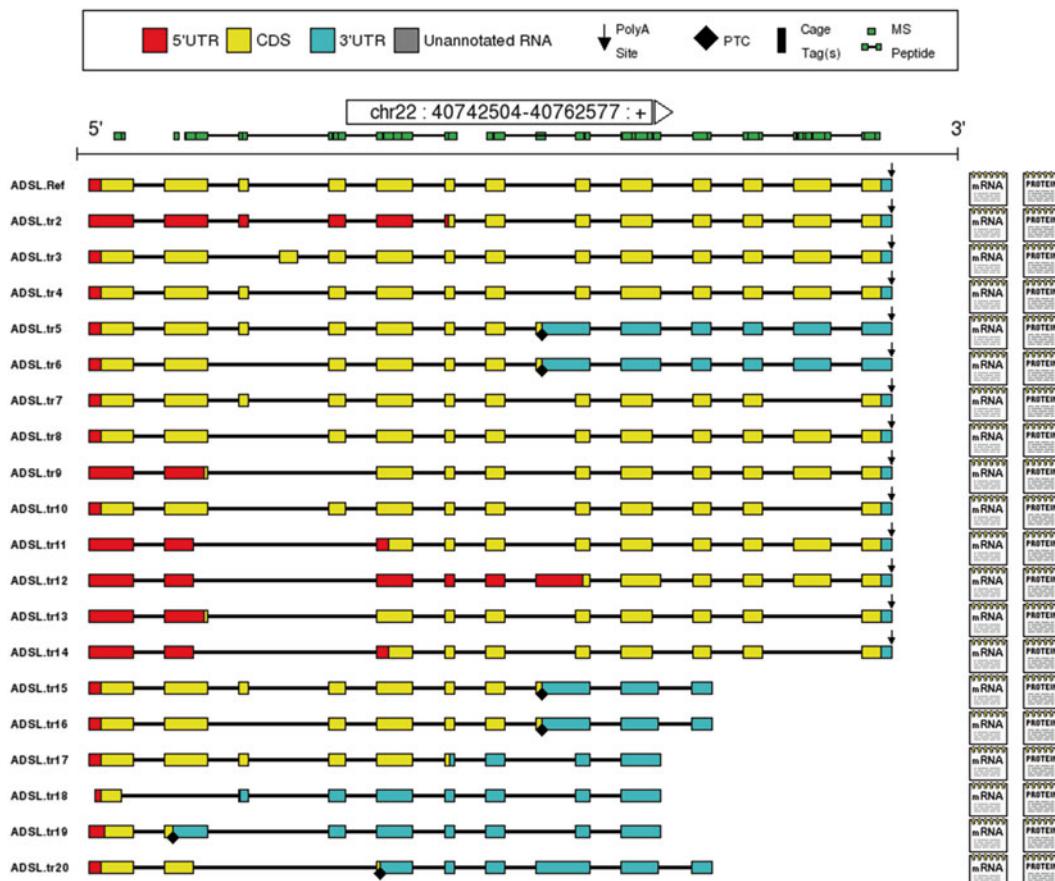


Fig. 5 Visualization of predicted transcripts for the adenylosuccinate lyase (ADSL) encoding gene

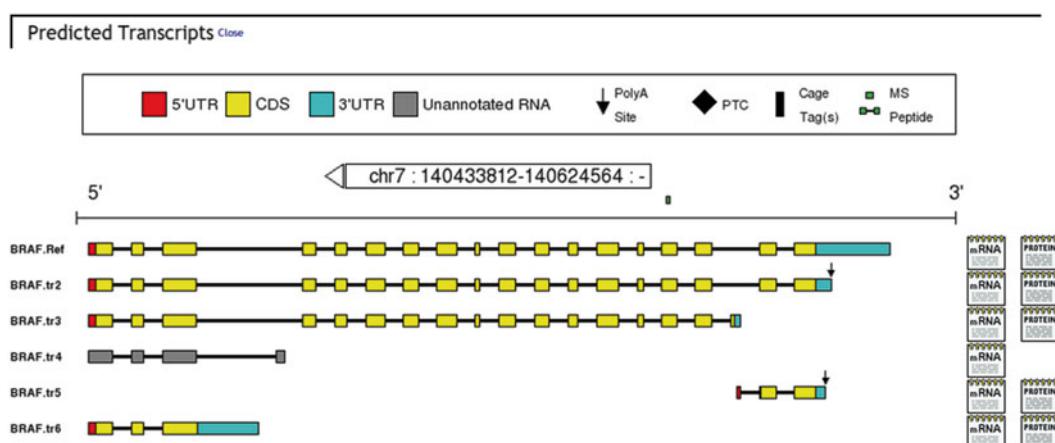


Fig. 6 Visualization of alternative transcripts predicted for B-Raf proto-oncogene serine/threonine kinase (BRAF)

(*unannotated RNA*), and obviously it's not possible to download the associated protein sequence. The figure also shows the presence of a poly(A) tail at the 3' end of the second and fifth transcript.

A CDS can be terminated with a Premature Termination Codon, represented with a black rhombus. Potential splicing regulatory element binding sites (fetched from SpliceAid-F [17] database) are displayed as red vertical bars.

The *Predicted Proteins* section shows an overview of all the functional domains of the alternative predicted proteins including Pfam's positions, transmembrane (TN), and coiled-coil (CC) domains. TN domains are represented by an orange stylized icon, while CC domains with a helix. Figure 7 shows an example of the ARAF protein structure. A mouseover tooltip displays additional details on the selected domain such as coordinates, length, and P value.

The *Predicted splice sites* section shows the multiple alignment of the genomic sequence against the mRNA and EST, specifically at the boundaries of all the predicted introns. This feature can be very useful when double-checking the predicted data, since it allows to verify which information is used for prediction, and how the alignments have been made.

In the *Transcript Table* a complete list of predicted alternative transcripts is provided as well as their variations with respect to a



Fig. 7 Visualization of predicted functional domains for A-Raf proto-oncogene, serine/threonine kinase (ARAF)

reference transcript (the first one in the table) usually corresponding to one of the transcripts collected in RefSeq. The purpose of this section is to integrate the *Predicted Transcripts* section with additional details. Each transcript is annotated with the length, the number of exons, and the coding sequence (CDS) coordinates. In case of truncated transcript coding sequences, coordinates are not completely reliable and it's possible to find a CDS longer than the one reported. In such cases coordinates are annotated with a sign < at the start point and/or a sign > at the end point.

The table also shows information on the predicted product and in particular if the CDS coordinates totally or partially match with the ones in the reference transcript and if its shares start/stop codons. The Variant Type column lists all predicted events for the transcript, encoded with a conventional code [23] to summarize the information.

Every splicing event is indicated with a keyword, such as *new* (new exon), *skip* (exon skip), *IR* (intron retention), *init* (alternative initiation), or *term* (alternative termination).

Every keyword is followed by the element affected by the event. For instance, E3 is the third exon or I9 the ninth intron in the reference RefSeq. If there is an additional element (e.g., exon) present with respect of the reference RefSeq element, a letter is added after the position of the last element in the reference, e.g., if a new exon is located downstream exon 1 in the reference, the variant is labeled “new (E1A).” Furthermore, the event localization is included: 5'UTR, CDS, 3'UTR, or in an intersection region 5'UTR_CDS and CDS_3'UTR.

In this example, *skip(E2),5'UTR*; *term(E6c),CDS_3'UTR*, two splicing events are described. The second exon (in 5'UTR) is skipped and the transcript terminates with the third (c) variant of the sixth exon, overlapping the CDS and 3'UTR.

The *Intron Table* lists all predicted introns and their main characteristics, in particular, the relative and absolute coordinates, the length, and the number of supporting ESTs. Donor-acceptor sites are reported with relative scores, encoded with the number of mismatching bases on the first and last 15 bases of, respectively, the downstream and upstream exon. The introns included in the reference transcript are printed in red.

The user can find in the *Protein Table* every predicted protein with annotation coming from different annotation software. This section lists additional details about the predicted alternative proteins, such as:

- Type: the protein class (i.e., globular or transmembrane) as predicted by Ensembl [30]
- Subloc: subcellular localization (i.e., nucleus, cytoplasm, mitochondria) from BaCelLo [31]
- PDB: link to the Protein Data Bank [14]
- UniProt: link to the UniProt Bank [13]

- SPEP: presence of signal peptides (predicted by SPEPlip software [32])
- GPI prediction from PredGPI [33]
- TMDDOM: coordinates of transmembrane domains (Ensembl [30])

3.5 Exporting Results

The possibility of exporting search results is crucial to allow further analysis and cross-validation with different tools. ASPicDB includes different download options and formats.

Search result tables contain a link to the download section, where it is possible to explore the element (i.e., gene, transcript, exon, intron, or protein) sequences. Every element has different download options, e.g., an intron can be extended with exonic sequences preceding and following the donor-acceptor sites (extension length is customizable by the user). For each transcript, exon sequences are downloadable, as well as its alternative protein.

Since the result sets can be considerably large, the extraction of the sequence data can be time-consuming. To avoid long waiting times, the user can just leave an email address, where a web link with the data will be sent, upon query completion.

A second layer of data export is available in the different *Result Visualization* section. In *Gene Information* all coding and transcript sequences are available for download, as well as the prediction result in a GTF file (Gene Transfer File). The GTF contains the list of predicted transcripts, exons, and introns. For each transcript all its exons and introns are listed, with a report containing the EST supporting each intron.

Moreover, in *Predicted Transcripts*, it's possible to download sequences for each exon, intron, transcript, and encoded proteins. Proteins can also be fetched in the *Protein Table*. In *Predicted Splice sites* an overall export process is available: users can download all the multiple EST alignments against genome splicing sites.

Acknowledgments

This work was supported by the Italian Ministero dell'Istruzione, Università e Ricerca (MIUR): PRIN 2009 and 2010 and Consiglio Nazionale delle Ricerche: Flagship Project Epigen, Medicina Personalizzata and Aging Program 2012–2014.

References

1. Wang ET, Sandberg R, Luo S, Khrebtukova I, Zhang L, Mayr C, Kingsmore SF, Schroth GP, Burge CB (2008) Alternative isoform regulation in human tissue transcriptomes. *Nature* 456:470–476
2. Pan Q, Shai O, Lee LJ, Frey BJ, Blencowe BJ (2008) Deep surveying of alternative splicing complexity in the human transcriptome by high-throughput sequencing. *Nat Genet* 40(12):1413–1415

3. Hallegger M, Llorian M, Smith CW (2010) Alternative splicing: global insights. *FEBS J* 277(4):856–866
4. Barash Y, Calarco JA, Gao W, Pan Q, Wang X, Shai O, Blencowe BJ, Frey BJ (2010) Deciphering the splicing code. *Nature* 465: 53–59
5. Castrignanò T, D’Antonio M, Anselmo A, Carrabino D, D’Onorio De Meo A, D’Erchia AM, Licciulli F, Mangiulli M, Mignone F, Pavese G, Picardi E, Riva A, Rizzi R, Bonizzoni P, Pesole G (2008) ASPicDB: a database resource for alternative splicing analysis. *Bioinformatics* 24(10):1300–1304
6. Martelli PL, D’Antonio M, Bonizzoni P, Castrignanò T, D’Erchia AM, D’Onorio De Meo P, Fariselli P, Finelli M, Licciulli F, Mangiulli M, Mignone F, Pavese G, Picardi E, Rizzi R, Rossi I, Valletti A, Zauli A, Zambelli F, Casadio R, Pesole G (2011) ASPicDB: a database of annotated transcript and protein variants generated by alternative splicing. *Nucleic Acids Res* 39(Database issue):D80–D85
7. Bonizzoni P, Mauri G, Pesole G, Picardi E, Pirola Y, Rizzi R (2009) Detecting alternative gene structures from spliced ESTs: a computational approach. *J Comput Biol* 16:43–66
8. Bonizzoni P, Rizzi R, Pesole G (2005) ASPIC: a novel method to predict the exon-intron structure of a gene that is optimally compatible to a set of transcript sequences. *BMC Bioinformatics* 6:244
9. Pirola Y, Rizzi R, Picardi E, Pesole G, Della Vedova G, Bonizzoni P (2012) PIintron: a fast method for detecting the gene structure due to alternative splicing via maximal pairings of a pattern and a text. *BMC Bioinformatics* 13(Suppl 5):S2. doi:[10.1186/1471-2105-13-S5-S2](https://doi.org/10.1186/1471-2105-13-S5-S2)
10. Pontius JU, Wagner L, Schuler GD (2003) UniGene: a unified view of the transcriptome. In: McEntyre J, Ostell J The NCBI Handbook. Bethesda, MD: National Center for Biotechnology Information
11. Benson DA, Cavanaugh M, Clark K, Karsch-Mizrachi I, Lipman DJ, Ostell J, Sayers EW (2013) Genbank. *Nucleic Acids Res* 41(Database issue):D36–D42
12. refseq. <http://www.ncbi.nlm.nih.gov/pubmed/24259432>
13. Boutet E, Lieberherr D, Tognoli M, Schneider M, Bairoch A (2007) UniProtKB/Swiss-Prot. *Methods Mol Biol* 406:89–112
14. Berman HM, Westbrook J, Feng Z, Gilliland G, Bhat TN, Weissig H, Shindyalov IN, Bourne PE (2000) The protein data bank. *Nucleic Acids Res* 28(1):235–242
15. Pfam database. <ftp://ftp.sanger.ac.uk/pub/databases/Pfam/Tools/>
16. Uniref90. <http://www.uniprot.org/help/uniref>
17. Giulietti M, Piva F, D’Antonio M, D’Onorio De Meo P, Paoletti D, Castrignanò T, D’Erchia AM, Picardi E, Zambelli F, Principato G, Pavese G, Pesole G (2013) SpliceAid-F: a database of human splicing factors and their RNA-binding sites. *Nucleic Acids Res* 41(Database issue):D125–D131
18. Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ (1990) Basic local alignment search tool. *J Mol Biol* 215(3):403–410
19. Gray KA, Daugherty LC, Gordon SM, Seal RL, Wright MW, Bruford EA (2013) Genenames.org: the HGNC resources in 2013. *Nucleic Acids Res* 41(Database issue):D545–D552. doi:[10.1093/nar/gks1066](https://doi.org/10.1093/nar/gks1066), Epub 2012 Nov 17
20. Maglott D, Ostell J, Pruitt KD, Tatusova T (2005) Entrez Gene: gene-centered information at NCBI. *Nucleic Acids Res* 33(Database issue):D54–D58
21. Hamosh A, Scott AF, Amberger JS, Bocchini CA, McKusick VA (2005) Online Mendelian Inheritance in Man (OMIM), a knowledge-base of human genes and genetic disorders. *Nucleic Acids Res* 33(Database issue):D514–D517
22. Ashburner M, Ball CA, Blake JA, Botstein D, Butler H, Cherry JM, Davis AP, Dolinski K, Dwight SS, Eppig JT, Harris MA, Hill DP, Issel-Tarver L, Kasarskis A, Lewis S, Matese JC, Richardson JE, Ringwald M, Rubin GM, Sherlock G (2000) Gene ontology: tool for the unification of biology. The Gene Ontology Consortium. *Nat Genet* 25(1):25–29
23. Riva A, Pesole G (2009) A unique, consistent identifier for alternatively spliced transcript variants. *PLoS One* 4(10):e7631
24. Zhang H, Hu J, Recce M, Tian B (2005) PolyA_DB: a database for mammalian mRNA polyadenylation. *Nucleic Acids Res* 33(Database issue):D116–D120
25. Kim N, Alekseyenko AV, Roy M, Lee C (2007) The ASAP II database: analysis and comparative genomics of alternative splicing in 15 animal species. *Nucleic Acids Res* 35(Database issue):D93–D98
26. Stamm S, Riethoven JJ, Le Texier V, Gopalakrishnan C, Kumanduri V, Tang Y, Barbosa-Moraes NL, Thanaraj TA (2006) ASD: a bioinformatics resource on alternative splicing. *Nucleic Acids Res* 34:D46–D55
27. Thierry-Mieg D, Thierry-Mieg J (2006) AceView: a comprehensive cDNA-supported

- gene and transcripts annotation. *Genome Biol* 7 (Suppl 1), S12, 1–14
- 28. Desiere F, Deutsch EW, King NL, Nesvizhskii AI, Mallick P, Eng J, Chen S, Eddes J, Loevenich SN, Aebersold R (2006) The PeptideAtlas project. *Nucleic Acids Res* 34(Database issue):D655–D658
 - 29. Kawaji H, Kasukawa T, Fukuda S, Katayama S, Kai C, Kawai J, Carninci P, Hayashizaki Y (2006) CAGE basic/analysis databases: the CAGE resource for comprehensive promoter analysis. *Nucleic Acids Res* 34(Database issue):D632–D636
 - 30. Hubbard T, Barker D, Birney E, Cameron G, Chen Y, Clark L, Cox T, Cuff J, Curwen V, Down T, Durbin R, Eyras E, Gilbert J, Hammond M, Hummicki L, Kasprzyk A, Lehvaslaiho H, Lijnzaad P, Melsopp C, Mongin E, Pettett R, Pocock M, Potter S, Rust A, Schmidt E, Searle S, Slater G, Smith J, Spooner W, Stabenau A, Stalker J, Stupka E, Ureta-Vidal A, Vastrik I, Clamp M (2002) The Ensembl genome database project. *Nucleic Acids Res* 30(1):38–41
 - 31. Pierleoni A, Martelli PL, Fariselli P, Casadio R (2006) BaCeLo: a balanced subcellular localization predictor. *Bioinformatics* 22(14): e408–e416
 - 32. Fariselli P, Finocchiaro G, Casadio R (2003) SPEPlip: the detection of signal peptide and lipoprotein cleavage sites. *Bioinformatics* 19(18):2498–2499
 - 33. Pierleoni A, Martelli PL, Casadio R (2008) PredGPI: a GPI-anchor predictor. *BMC Bioinformatics* 9:392–395. doi:[10.1186/1471-2105-9-392](https://doi.org/10.1186/1471-2105-9-392)

Chapter 24

Analysis of Alternative Splicing Events in Custom Gene Datasets by AStalavista

Sylvain Foissac and Michael Sammeth

Abstract

Alternative splicing (AS) is a eukaryotic principle to derive more than one RNA product from transcribed genes by removing distinct subsets of introns from a premature polymer. We know today that this process is highly regulated and makes up a large part of the differences between species, cell types, and states. The key to compare AS across different genes or organisms is to tokenize the AS phenomenon into atomary units, so-called AS events. These events then usually are grouped by common patterns to investigate the underlying molecular mechanisms that drive their regulation. However, attempts to decompose loci with AS observations into events are often hampered by applying a limited set of a priori defined event patterns which are not capable to describe all AS configurations and therefore cannot decompose the phenomenon exhaustively.

In this chapter, we describe working scenarios of AStalavista, a computational method that reports all AS events reflected by transcript annotations. We show how to practically employ AStalavista to study AS variation in complex transcriptomes, as characterized by the human GENCODE annotation. Our examples demonstrate how the inherent and universal AStalavista paradigm allows for an automatic delineation of AS events in custom gene datasets. Additionally, we sketch an example of an AStalavista use case including next-generation sequencing data (RNA-Seq) to enrich the landscape of discovered AS events.

Key words Gene expression, RNA processing, Alternative splicing, AS event, Definition of alternative splicing, Transcriptome annotation, RNA-seq, Splicing nomenclature, AS code, Bioinformatics

1 Introduction

Alternative splicing (AS) is the mechanism enabling eukaryotic cells to define and to regulate the set of introns that is removed from nascent RNA molecules during transcription. AS plays a key role in gene expression and transcriptome diversity, and the phenomenon still remains to be fully characterized [1]. One way to investigate AS is to compare and to characterize the RNA content of cell samples (e.g., from different experimental conditions, tissues, treatments, etc.) in order to identify variations between transcripts and consequently the underlying AS events. On the way to achieve efficient and user-friendly tools, computational methods of

AS analysis are constantly coping with advances in transcriptomics, in particular, recent innovations in the field of so-called next-generation sequencing (NGS) technologies, which require high-throughput pipelines to analyze large data volumes.

AStalavista (alternative splicing and transcriptional landscape visualization tool altogether) provides a computational method and software application for the automatic characterization of AS landscapes in custom transcriptome data [2]. The method automatically identifies AS events by comparing all transcripts of a given annotation file and reporting an exhaustive yet nonredundant list of the resulting variations. To allow for the generic characterization of—possibly hitherto undiscovered—morphologies of AS events, so-called AS patterns, AStalavista employs a generic nomenclature, the alternative splicing code (the “AS code”).

In this chapter, we present the AStalavista method and its notation to describe AS events and their corresponding patterns. Employing publicly available data, we demonstrate the simplicity and universality of the approach for custom gene datasets: we first provide a concrete example of an AStalavista run on the transcript annotation of the complete human genome, and then we use the method to study transcriptome variation depicted by a NGS experiment of the ENCODE project.

2 Materials

2.1 Installing AStalavista

The AStalavista package can be accessed from the website
<http://astalavista.sammeth.net>

where also the source code is available from the “Download” section. To install, just unzip/untar the bundle once downloaded, for instance, by the command

```
tar xzf AStalavista-3.2.tgz
```

The command creates a folder named “AStalavista” (with the corresponding version number) that contains all the files from the tarball: documentation, license information, and program files. AStalavista is executed by wrapper scripts located in the “bin” subfolder within the installation directory, either “astalavista.bat” (for Windows platforms) or “astalavista” (for UNIX-based operating systems, including OSX).

The “--help” flag provides a brief description of the software, its version, and the available options: flags to handle verbosity, parallelization, output behavior, and available tools (“--list-tools”; see below).

```
AStalavista-3.2/bin/AStalavista--elp
AStalavista v3.2 (Flux Library: 1.22)
-----Documentation & Issue Tracker-----
```

```

Flux Wiki (Docs): http://sammeth.net/confluence
Flux JIRA (Bugs): http://sammeth.net/jira
Please feel free to create an account in the
public
JIRA and reports any bugs or feature requests.
-----
The Flux library comes with a set of tools.
You can switch tools with the -t option. The
general options
change the behaviour of all the packaged tools.
General Flux Options:
...
List of available tools
...

```

2.2 Obtaining the GENCODE Annotation

The full GENCODE transcriptome annotation dataset [3] can be downloaded from the project's ftp site by the command:

```

wget
ftp://ftp.sanger.ac.uk/pub/gencode/
release\_12/gencode.v12.annotation.gtf.gz ./

```

and subsequently, the downloaded gzip archive is decompressed by

```
gunzip Gencode.v17.annotation.gtf.gz
```

2.3 Retrieving RNA-Seq Alignments

In our NGS examples, we employed publicly available RNA-Seq data from the ENCODE project [4] as downloaded from the CRG's (Centre for Genomic Regulation, Barcelona) RNA-Seq Dashboard:

```
http://genome.crg.es/encode\_RNA\_dashboard/hg19/
```

The CRG dashboard allows navigating hundreds of RNA-Seq experiments produced by the ENCODE consortium to profile the human transcriptome of different cell types, subcellular compartments, and RNA populations. Employing the STAR mapper [5], reads of some of the datasets have been split-mapped to the human genome in order to identify the position of the introns. For our example, we picked one of these mapping files downloaded from the URL:

```
http://genome.crg.es/~jlagarde/encode/pre-DCC/wgEncodeCshlLongRnaSeq/20130218\_promocell\_batches1-2\_minus\_batch1sFASTQ/SID38226\_SID38227\_SJ.bed
```

The aligned RNA-Seq data has been obtained from long RNAs that have been extracted from HPAEC (Human Pulmonary Artery Endothelial Cells) and sequenced on an Illumina HiSeq 2000 machine using a strand-specific protocol. However, the ENCODE dashboard provides data from many other experimental conditions, which could be combined or compared with this example dataset.

3 The AStalavista Method

3.1 AS Event

Definition and Nomenclature

Computer-aided comparison of transcriptome data is a natural way to analyze AS, and many studies have followed that approach [6]. In order to make conclusions on the effects of AS, the idea is to identify in the first place common differences in the exon-intron structure of processed RNA molecules. This can be straightforwardly achieved by comparing the transcript sequences with each other, either directly or by using their localization within the context of the corresponding genomic sequence, if available. The “mapping” of transcripts onto a genomic reference allows to identify the position of all exons and introns, producing what we call a transcriptome annotation.

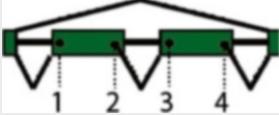
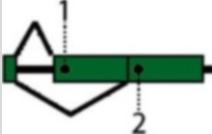
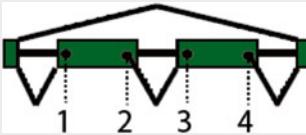
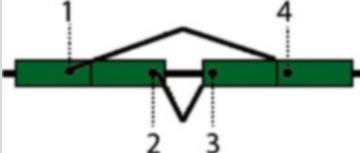
3.1.1 AS Events

Given an annotation, each transcript can be uniquely identified by the type and the genomic position of all of its exonic boundaries. Here we consider three types of exonic boundaries, we refer to as “sites”: the transcription start site (TSS) at the 5' extremity of the transcript, the cleavage site (CVS) at the 3'-end of the transcript—often coupled with the polyadenylation site—and splice sites that delineate introns in the premature transcript; splice sites are furthermore categorized as *donor* or *acceptor* sites according to their localization at the 5'- or at the 3'-end of an intron, respectively. Alternative splicing occurs when a splice site is used in the splicing process of one transcript but not of another transcript of the same locus. An AS event intuitively comprises one or multiple of such alternative splice sites; moreover, since transcription and splicing are known to co-occur and interact in eukaryotic cells [7, 8], some AS events also include alternative TSSs or CVSs in addition to alternative splice site(s).

More precisely, when comparing the exon boundaries of two or more transcripts from a genomic locus, the AStalavista approach first identifies *constitutive* (splice) sites as exon boundaries employed by all transcripts that overlap the corresponding genomic position; consequently, sites that are not used by all transcripts of the locus are considered as *variable*. In a further step, AStalavista delineates AS events as a maximal sequence of consecutive variable sites, with at least one alternative splice site. Based on this definition, AS events either contain exclusively alternative splice sites delimited by common sites at both ends (so-called internal AS events) or they extend until the transcript extremities and contain variable TSSs and/or CVSs.

By this generic event definition of AStalavista, resulting events can describe variations between two or more different transcript structures. The number of transcripts that are simultaneously compared naturally plays an important role in the process of assessing whether a certain site is variable or constitutive. For the sake of convenience, we will consider here pairwise AS events only, but we

Table 1
Examples of distinct AS events with the corresponding AS patterns

Traditional name	Exon-intron structure	AS code	Example
A Exon skipping		0, 1-2^	PTBP1 polypyrimidine tract-binding protein 1, ninth exon
B Alternative acceptor		1-, 2-	PTBP1 polypyrimidine tract-binding protein 1, ninth exon
C		0, 1-2^3-4^	FBLIM1 filamin-binding LIM protein 1, fourth + fifth exon
D		1^4-, 2^3-	ZWINT ZW10 interacting kinetochore protein, sixth – seventh exon

would like to point interested readers to the concept of complete alternative splicing events for a more advanced characterization of AS loci [9].

Table 1 shows some AS patterns along with examples as they are observed in the GENCODE annotation of the human transcriptome. The alternative exon 9 in the PTBP1 (polypyrimidine tract binding protein 1) gene has been reported to be entirely skipped (“exon skipping,” A) or included with different acceptor flanks (“alternative acceptor,” B). One of the multiple possibilities to process the pre-mRNA of the FBLIM1 (Filamin-binding LIM protein 1) gene (C) excludes simultaneously the subsequent exons 4 (188 nt) and 5 (103 nt), such that overall the reading frame is not disturbed (292 nt corresponds to 97aa). (D) Similarly, variations on either side of the sixth intron in the ZWINT locus regulate the optional inclusion of 47 amino acids into the resulting protein. As we will show in the next section, the complete set of AS events can be automatically identified and reported by the ASstalavista software [10].

3.1.2 The AS Code

AS events are commonly classified in different categories based on the observed differences in the exon-intron structure between the respective transcripts. Typically, most studies employ the terms

“exon skipping,” “alternative donor/acceptor,” “intron retention,” and (sometimes) “mutually exclusive exons” to describe AS patterns. However, a nomenclature built exclusively by this set of predefined terms allows to describe only a couple of AS configurations, which due to their simplicity are observed quite frequently but constitute only a small fraction of the spectrum of all event patterns found in complete transcriptome annotations. An exhaustive characterization of the AS landscape from arbitrary transcript structures requires a more flexible notation that permits to identify all events that share the same “topography.”

For this reason, AStalavista uses a generic notation system that can univocally describe any AS event. This nomenclature assigns an “AS code” to every sequence of alternative splice sites, such that events with qualitatively identical variations in their exon-intron structure receive the same AS code. This AS code represents each variable site by a number and a symbol, according to its relative position within the event and to its type. More precisely, the code of a certain AS event is generated as follows: first, all variable sites that are included in the event are sorted according to their position in the directionality of transcription, from 5' to 3'. Each site is assigned a number according to this order (1, 2, 3, etc.) and a symbol based on its type: “[“for TSS, ”]” for CVS, “^” for donor, and “-” for acceptor. Then, sites from the same transcript are represented next to each other by consecutive pairs of such number-symbol tuples (in the corresponding 5' → 3' order), forming a string that represents the AS event. In the absence of variable sites in one of the transcripts of the event (e.g., to describe the string for a transcript skipping an exon), the digit “0” is employed as a corresponding string. The AS code then is a comma-separated concatenation of thus obtained strings, one from each transcript variant, ordered by their numbers. Some examples for AS events along with their corresponding AS code are shown in Table 1.

3.2 Processing AS Events from Transcriptome Annotations: The GENCODE Example

The ENCODE project intends to establish a repertoire of functional elements in the human genome [11]. To this aim, a tremendous amount of experimental data has been (is being) produced and analyzed by hundreds of scientists worldwide. As part of the results from this effort, the GENCODE annotation provides the community with an expertized and thoroughly curated set of genes and transcript positions in the genomic sequence. The GENCODE annotation describes the exon-intron structure of thousands of alternative transcripts and therefore allows us to profile AS extensively and accurately. Therefore, as a first example for the application, we will employ AStalavista to characterize the AS landscape of the entire human transcriptome as described by the GENCODE annotation.

3.2.1 Running AStalavista on the GENCODE Annotation

The AStalavista program package is subdivided in different tools; a list of available tools can be obtained by requesting help via the corresponding command option

```
./bin/AStalavista -list-tools
```

which by the current version (AStalavista-3.2) outputs the summary:

```
List of available tools
scorer - Splice site scorer
sortBED - Sort a BED file. If no output
file is specified, result is printed to standard out
sortGTF - Sort a GTF file. If no output file
is specified, result is printed to standard out
asta - AStalavista event retriever
subsetter - Extract a random subset of lines
from a file
```

The tool “asta” performs the extraction of alternative splicing, for which additional parameters and their default values in turn are obtained by

```
./bin/AStalavista -t asta -h
```

Parameters can either be provided on the command line or in a separate parameter file (specified by flag “-p”). A mandatory parameter is the path to the file with the annotation that is to be analyzed (parameter “IN_FILE”, or command line parameter “-i”). As an example, an AStalavista run on the v12 of the GENCODE transcript collection is executed by

```
./bin/AStalavista -t asta -i Gencode.
v12. annotation.gtf
```

AStalavista employs all exons (i.e., lines identified by the feature “exon” in the third column) from the gtf file, which are to be sorted by their genomic positions and require a “transcript_id” identifier for correct gene clustering. If the file is not sorted, AStalavista will automatically sort the file before processing.

As a default setting, AStalavista reports only alternative splicing events [10] that are not linked to variable transcription initiation and/or cleavage sites, so-called internal AS events (ASI). External AS events (ASE) that include TSSs and/or CVSs can also be included in the list of reported event types (value ASE for the parameter EVENTS or command line flag “-e”):

```
./bin/AStalavista -t asta -i Gencode.
v12. annotation.gtf -e ASI,ASE
```

Per default, AStalavista further on shows exclusively pairwise AS events, i.e., events with exactly two variants (dimension 2); events of a higher dimension are projected to events between all

pairs of their variants. To change the order of dimension for the reported AS events, parameter EVENTS_DIMENSION (flag -d) has to be set to a corresponding integer value or “-1” for outputting the so-called complete AS events [9]:

```
./bin/ASTalavista -t asta -i Gencode.v12.annotation.gtf -d -1
```

3.2.2 Interpreting the Results

ASTalavista produces from the input annotation in gtf format a file with events that are equally stored as gff features: start (column 4) and end (column 5) of the event are the genomic coordinates of the delimiting common sites [10], respectively, the first/last variable site in the case of ASE events; the nature of these delimiting sites is detailed in the “flanks” attribute, which adopts the same “AS code” notation as used for “splice chain” and “structure” (see below):

```
chr1      Gencode_v12      as_event
12227    12721    .      +      .      transcript_id "ENST00000518655.2,ENST00000456328.2/ENST00000515242.2"; gene_id "chr1:11869-14412W"; flanks "12227^,12721^"; structure "1-,2-"; splice_chain "12595-,12613-"; dimension "2_2"; degree "4";
```

The attribute “degree” summarizes the number of variable sites in the event, which is naturally increasing with longer events or events of more variants. Additionally, the attribute “dimension” provides information about the reported event with respect to the underlying complete event: the dimensionality is denoted as a string of the form “ X_Y ” where X is the number of variants in the reported event and Y is the number of events in the corresponding complete event. All events with $X=Y$ are complete, and those with $X < Y$ are not ($X > Y$ is not possible by definition).

The “transcript_id” attribute describes a comma-separated list of the transcript identifiers for each of the variant structures of the event; if there is more than one transcript supporting a certain variant of the event, the corresponding identifiers are furthermore separated by a “/” separator. The attribute “splice_chain” describes the genomic coordinates of each variable site in the event, with the variants in the same comma-separated order as the identifiers of “transcript_id”.

As introduced previously, the AS code nomenclature specifies after the genomic coordinate of each site also the site type by a certain symbol: “^” denotes a splice donor, “-” a splice acceptor, and “[“a transcription start and”]” a cleavage site. Together with the name of the chromosome/contig, the value of “splice_chain” can be considered as a unique identifier of the event, because it describes every event univocally by the morphology of its sites as localized by corresponding genomic coordinates. Therefore, also events delimited by the same flanks (start/end tuples) can still differ by their splice chains.

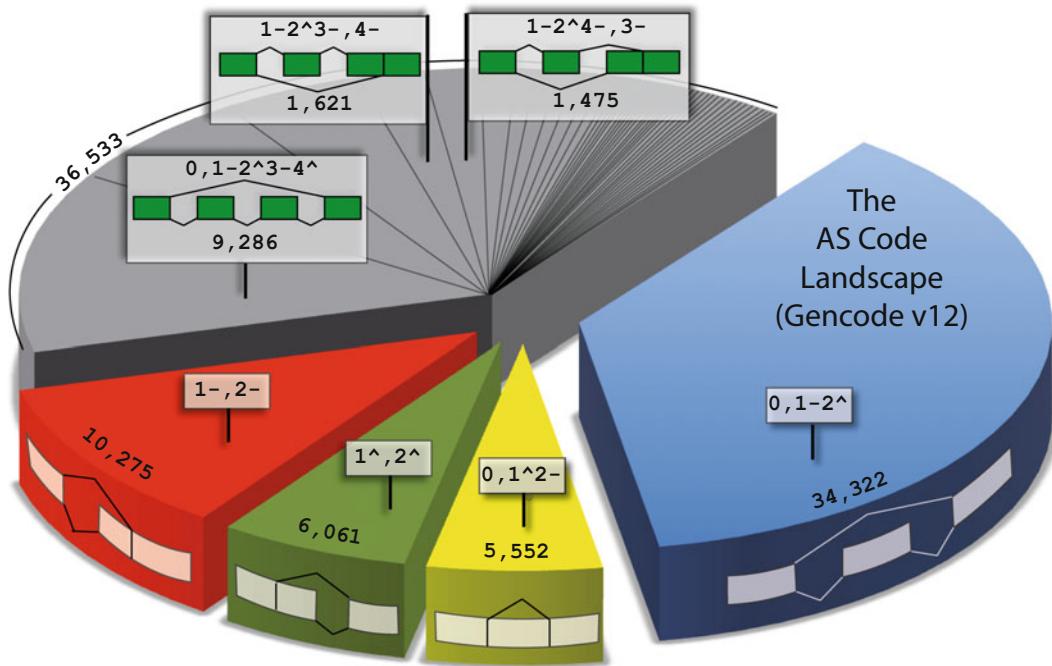


Fig. 1 AS landscape of internal events of the Gencode (v12) annotation

By replacing genomic coordinates reported for the “splice_chain” by relative positions of the corresponding sites within the event, the attribute “structure” abstracts events to their respective patterns that are named by the corresponding AS code; examples for AS codes of common events are: “0,1-2[^]” for exon skipping, “1[^],2[^]” for alternative donors, “1-,2-” for alternative acceptors, “0,1[^]2-” for intron retention, and “1-2[^],3-4[^]” for mutually exclusive exons.

Figure 1 depicts all *internal* AS events found in the Gencode (v12) transcriptome annotation, grouped by their respective AS code. Altogether, the landscape of different AS morphologies reveals that traditional events (blue, yellow, green, and red slices) in our days hardly make up 37 % of the spectrum spanned by all AS patterns—the majority of events (gray slice) cluster into 813 different, more complex events. Note that, depending on the position of the event, it might further be possible to introduce alternative TSSs or CVSs in several AS shapes to further amplify the landscape (*external* AS events).

The result of a default AStalavista runs as shown in Fig. 1, for example, GENCODE annotation retrieves 92,743 internal AS events (ASI) that cluster in 817 different patterns (respectively 670,104 events clustering in 17,741 patterns when including ASE events). Expectedly, the most abundant event patterns are of degree 2 which corresponds to the minimum number of variable

sites required for an AS event: “0,1-2[^]” (exon skipping) ranks first, “1-,2-” (alt. acceptors) ranks second, “1[^],2[^]” (alternate donors) ranks third, and “0,1[^]2-” (intron retention) ranks fourth most abundant pattern. The reasons for the differences between these patterns lie in splicing mechanism and constraints of coding sequences, which constitute the main part of messenger RNA transcripts [10].

Although the event complexity specified by “degree” usually anticorrelates with the observed abundance, there are exceptions. For instance, the skipping of multiple exons is mechanistically facilitated and therefore more often observed than other events of the corresponding degree: “0,1-2[^]3-4[^]” ranks before “1[^],2[^]” and “0,1[^]2-”, “0,1-2[^]3-4[^]5-6[^]” and also before many other patterns of degree 4, etc. Moreover, only ~47 % of pairwise events found on the GENCODE annotation are complete; from another point of view, that means that the true complexity of more than half of the events is underestimated when looking at them exclusively by pairwise variant comparisons.

3.3 Discovery of Novel AS Events by RNA-Seq Experiments

3.3.1 Employing AStalavista for NGS Data

AStalavista can also analyze NGS data from RNA sequencing experiments (RNA-Seq). The main conceptual difference between transcriptome annotations, like GENCODE and RNA-Seq data, is that the latter only provide partial information on the transcriptomes: RNA-Seq reads are limited to parts of expressed genes, because current NGS technologies still cannot reproduce sequences of entire RNA molecules and transcripts thus are fragmented before sequencing. Although RNA-Seq reads correspond to fragments and cannot be considered as full-length transcripts, they can provide valuable information about AS by potentially novel splice junctions they harbor: if the exon-exon junction of a spliced transcript is captured in a read, the position of the corresponding intron can be revealed by aligning the read sequence on the genomic reference.

Dedicated NGS alignment methods—as implemented in software like GEM [12] or STAR [5]—allow to generate such spliced alignments where introns correspond to gaps flanked by exonic regions. These experimentally derived evidences about splicing events can be analyzed by AStalavista, either separately or in combination with a reference annotation, to identify novel alternative splicing events that are not captured by the reference gene set. An interesting and original aspect of the latter approach is that no transcript assembly is required for the analysis; indeed, AStalavista considers reads as independent pieces of information and does not take any assumptions which of them might originate from the same and which of them have been obtained from independent RNA molecules, respectively. In the next section, we employ this strategy to illustrate how AStalavista can enrich a repertoire of AS events by processing RNA-Seq data.

3.3.2 Processing RNA-Seq Data

We converted the format of the bed file with mapped reads downloaded earlier from the CRG dashboard, and produced an ASTalavista-compatible gtf file by the commands:

```
bed=SID38226_SID38227_SJ.bed; gtf=${bed%.
bed}.gtf; cat $bed | awk -v bed=$bed -v OFS="\t"
'{print $1,bed,"exon",$2,$2,$5,$6,".","gene_id
\"SJ"NR"\\"; transcript_id \"SJ"NR"\\";";print
$1,bed,"exon",$3,$3,$5,$6,".","gene_id
\"SJ"NR"\\"; transcript_id \"SJ"NR"\\";"}'>$gtf
```

Finally, we concatenated thus obtained gtf representation of the NGS mappings with the GENCODE annotation from our previous example:

```
cat Gencode.v12.annotation.gtf $gtf>
Gencode_rnaseq.gtf
```

The joint dataset then was processed with ASTalavista in order to compare correspondingly obtained AS events with the results derived from the GENCODE annotation alone:

```
./bin/ASTalavista -t asta -i Gencode_
rnaseq.gtf
```

3.3.3 Identifying Novel AS Events

After processing the NGS-enriched GENCODE annotation with ASTalavista, we characterized events involving RNA-Seq reads that were not detected using the annotation alone. We found 342 such novel internal AS events (ASI). Figure 2 shows that the distribution of patterns among these additionally found events follows closely the one described by GENCODE events, albeit some rank positions are permuted: evidence by RNA-Seq split mappings indicates that among the simple AS patterns, alternative exon boundaries (89 donors and 79 acceptors events) are most underestimated by the current GENCODE annotation, followed by alternatively skipped exons (53 novel alternative exons); furthermore, RNA-Seq data pinpoints 177 novel complex events of 34 distinct patterns (Fig. 2).

Figure 2 also summarizes the number and nature of AS events that are revealed by RNA-Seq introns; many describe the skipping of (multiple) exons: 53 “0,1-2^” (1 exon), 21 “0,1-2^3-4^” (2 exons), 11 “0,1-2^3-4^5-6^” (3 exons), etc. Albeit our analysis focuses on a small set of high-confidence events, we observe nearly half of the exons that predicted to be skipped (20 out of 53) exhibit a frame-preserving length, i.e., the size of the exon is a multiple of 3. These events include, for instance, the third exon in the TMCO1 (transmembrane domain protein) locus, where also additional EST (Expressed Sequenced Tags) evidence (BP308568) and complementary computational gene models (ENST00000476173) support the skipping of the corresponding 60 nt exon; as another example, we find NGS evidence for the skipping of the 36 nt long exon number 10 in the TRIP (thyroid hormone receptor interacting protein) gene, which so far has not been discovered by EST evidence.

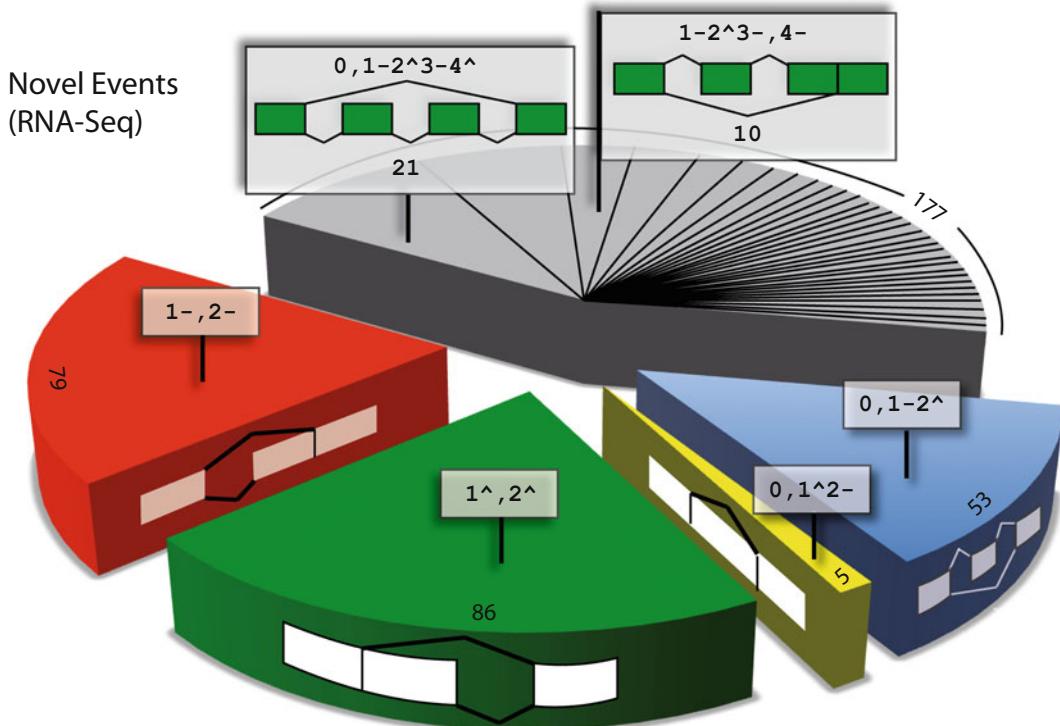


Fig. 2 Distribution of novel AS events found by additional RNA-Seq evidence

As in other analyses that employ NGS data, potential artifacts might impair the results, and some of the events might originate from errors in the sequencing or the alignment process. As in the case of reference annotations, ASTALAVISTA processes the exons and introns provided by NGS data and all variations in the exon-intron structure are reported. Reassuringly, however, we find among these novel events several traces of constraints created by coding sequences, for instance, a nonnegligible proportion of exons that are predicted to be skipped by RNA-Seq evidence would not break the coding frame. Also, there is independent evidence from cDNA and EST data that supports some of the NGS-derived events. Finally, it is worth noting that most of the NGS evidence for exon skipping predicts alternative exons longer than 100 bp, which are unlikely to originate from misalignments or gaps in the alignment process.

Certainly, those AS events can be subject to further ranking and/or filtering procedures, for instance, by considering their overlap with the annotated coding sequence, by taking the number of reads supporting each of the splice junctions in the event and their corresponding alignment quality (e.g., gaps) into account, etc. However, the example demonstrates that already a straightforward application of ASTALAVISTA to different RNA-Seq datasets

allows to analyze cell-type or condition-specific transcriptomes, as by adding the split-mapped reads to a reference annotation and comparing the correspondingly obtained AS events for novel configurations.

An interesting specificity of the approach is that—unlike most of other RNA-Seq data processing tools—AStalavista does not attempt to assemble the reads into longer contigs nor even into transcribed fragments of the genomic space (the so-called transfrags). Although transfrag-based methods have been producing better results in terms of accuracy along the years, transcriptome reconstruction from RNA-Seq data is still a highly challenging task in the field [3]. Whether methods use a genomic reference, as in our example, or whether they rather compare reads with each other (“de novo” assembly), the risk of merging some unrelated reads to a common contig cannot be completely avoided [13], generating some artificial hybrid that does not exist *in vivo*. Moreover, transcriptome assembly methods are prone to errors when dealing with repeated sequences and/or incomplete data, thus processing RNA-Seq data by AStalavista right after the read mapping step improves the general reliability of the analysis. Under the assumption that genomic read alignments are usually a more reliable source of information than transcript structures reconstructed upon them, AStalavista is not concluding that reads with a common subsequence originate from identical transcripts but rather focuses on splicing differences they (might) harbor: by the atomary definition of delineating variations strictly within the common genomic space ensures that any of the reported AS events is supported by two real sequences.

Another advantage of the AStalavista approach is its efficiency in terms of processing time. The example shown in this chapter took about 5 min on a 2 GHz Intel system, occupying <1 GB of RAM, which makes it suitable for high-throughput processing. The rather simplistic examples we provide here can straightforwardly be extended to the application of AStalavista for processing additional datasets, e.g., for comparing different reference annotations, transcriptomes from different cell compartments, and/or different experimental conditions or data preprocessing strategies.

References

- Kornblihtt AR, Schor IE, Alló M (2013) Alternative splicing: a pivotal step between eukaryotic transcription and translation. *Nat Rev Mol Cell Biol* 14:153–165
- Foissac S, Sammeth M (2007) ASTALAVISTA: dynamic and flexible analysis of alternative splicing events in custom gene datasets. *Nucleic Acids Res* 35:W297
- Harrow J, Frankish A, Gonzalez JM (2012) GENCODE: the reference human genome annotation for The ENCODE Project. *Genome Res* 22:1760
- Dunham I, Birney E, Lajoie BR, Sanyal A (2012) An integrated encyclopedia of DNA elements in the human genome. *Nature* 489:57

5. Dobin A, Davis CA, Schlesinger F (2013) STAR: ultrafast universal RNA-seq aligner. *Bioinformatics* 29:15
6. Mironov AA, Fickett JW, Gelfand MS (1999) Frequent alternative splicing of human genes. *Genome Res* 9:1288–1293. doi:[10.1101/gr.9.12.1288](https://doi.org/10.1101/gr.9.12.1288)
7. Tilgner H, Knowles DG, Johnson R (2012) Deep sequencing of subcellular RNA fractions shows splicing to be predominantly co-transcriptional in the human genome but inefficient for lncRNAs. *Genome Res* 22:1616
8. Boireau S, Maiuri P, Basyuk E, de la Mata M, Knezevich A, Pradet-Balade B, Bäcker V, Kornblihtt A, Marcello A, Bertrand E (2007) The transcriptional cycle of HIV-1 in real-time and live cells. *J Cell Biol* 179:291–304. doi:[10.1083/jcb.200706018](https://doi.org/10.1083/jcb.200706018)
9. Sammeth M (2009) Complete alternative splicing events are bubbles in splicing graphs. *J Comput Biol* 16:1117
10. Sammeth M, Foissac S, Guigó R (2008) A general definition and nomenclature for alternative splicing events. *PLoS Comput Biol* 4:e1000147. doi:[10.1371/journal.pcbi.1000147](https://doi.org/10.1371/journal.pcbi.1000147)
11. Djebali S, Davis CA, Merkel A, Dobin A, Lassmann T, Mortazavi A, Tanzer A et al (2012) Landscape of transcription in human cells. *Nature* 489:101–108. doi:[10.1038/nature11233](https://doi.org/10.1038/nature11233)
12. Marco-Sola S, Sammeth M, Guigó R, Ribeca P (2012) The GEM mapper: fast, accurate and versatile alignment by filtration. *Nat Methods* 9:1185. doi:[10.1038/nmeth.2221](https://doi.org/10.1038/nmeth.2221)
13. Mundry M, Bornberg-Bauer E (2012) Evaluating characteristics of de novo assembly software on 454 transcriptome data: a simulation approach. *PLoS One* 7:e31410

Chapter 25

Computational Design of Artificial RNA Molecules for Gene Regulation

**Alessandro Laganà, Dario Veneziano, Francesco Russo,
Alfredo Pulvirenti, Rosalba Giugno, Carlo Maria Croce, and Alfredo Ferro**

Abstract

RNA interference (RNAi) is a powerful tool for the regulation of gene expression. Small exogenous noncoding RNAs (ncRNAs) such as siRNA and shRNA are the active silencing agents, intended to target and cleave complementary mRNAs in a specific way. They are widely and successfully employed in functional studies, and several ongoing and already completed siRNA-based clinical trials suggest encouraging results in the regulation of overexpressed genes in disease.

siRNAs share many aspects of their biogenesis and function with miRNAs, small ncRNA molecules transcribed from endogenous genes which are able to repress the expression of target mRNAs by either inhibiting their translation or promoting their degradation. Although siRNA and artificial miRNA molecules can significantly reduce the expression of overexpressed target genes, cancer and other diseases can also be triggered or sustained by upregulated miRNAs.

Thus, in the past recent years, molecular tools for miRNA silencing, such as antagomiRs and miRNA sponges, have been developed. These molecules have shown their efficacy in the derepression of genes downregulated by overexpressed miRNAs. In particular, while a single antagomiR is able to inhibit a single complementary miRNA, an artificial sponge construct usually contains one or more binding sites for one or more miRNAs and functions by competing with the natural targets of these miRNAs. As a consequence, natural miRNA targets are reexpressed at their physiological level.

In this chapter we review the most successful methods for the computational design of siRNAs, antagomiRs, and miRNA sponges and describe the most popular tools that implement them.

Key words RNAi, siRNA, shRNA, antagomiR, miRNA, Sponge, Gene expression

1 Introduction

The discovery of the first short ncRNA capable of acting as endogenous regulator of gene expression was made by Ambros et al. in 1993, while investigating the role of lin-4 in the developmental timing of *C. elegans* [1]. That accomplishment revealed only a glimpse of the much broader reality uncovered by Fire and Mello 5 years later when they reported the capability of exogenous double-stranded RNAs to silence genes in a specific manner, disclosing the

mechanism we know today as RNA interference (RNAi) [2]. In 1999, a similar process was discovered to take place in plants, as short RNA sequences (~20–25 nt) were found to be able to bind their mRNA targets through perfect base complementarity [3]. Since then, a revolution has occurred in the field of RNA biology, thanks to the characterization of RNAi as an innate biological process through which the expression of specifically targeted genes can be modulated and/or silenced, ushering in a new world of research and potential therapeutic applications. Specifically, RNAi is a naturally occurring mechanism resulting in a sequence-specific, post-transcriptional downregulation of gene expression induced by double-stranded RNA (dsRNA) homologous to the target gene. This regulation can occur at different levels of the gene expression process, including transcription, mRNA processing, and translation. The small RNA molecules responsible for RNAi originate from endogenous and exogenous double-stranded precursors and play a specific role in guiding effector protein complexes toward their nucleic acid targets by partial or full complementarity bonds [4].

Although several classes of regulatory ncRNAs have been identified, the most relevant ones can essentially be represented by two groups, according to their origin, structure, associated effector proteins, and function: small interfering RNAs (siRNAs), principally of exogenous origin in animals, and microRNAs (miRNAs), which are endogenous genome products. These molecules, apparently present only in eukaryotes and in some viruses, are in fact the most abundant regulatory molecules in terms of both phylogeny and physiology.

siRNA is a class of double-stranded RNA molecules, 20–25 base pairs in length, whose role appears to be involved in the defense of the cell and maintenance of genome integrity through the silencing of exogenous nucleic acids and undesired transcripts (such as transposons and repetitive elements) [4, 5]. Although their posttranscriptional gene-silencing role in plants and in some simple animal species was shown to be of endogenous origin, siRNAs in animals, instead, are mainly exogenous molecules obtained from perfectly complementary long double-stranded precursors coming from viruses and transposons [3]. As it was first demonstrated in nematodes [2] and later in mammals [6], siRNA-mediated RNAi can be obtained either by simple dsRNAs of about 21 nucleotides (nt) with two-nucleotide 3' overhang or by stably expressed short hairpin RNAs (shRNAs), which are processed into siRNAs [7, 8]. Compared to unprocessed siRNA, this last approach undoubtedly presents many advantages, such as long-lasting silencing effects, cost-effectiveness, as well as easy delivery methods. Generally, shRNA is transcribed in cells from a DNA template as a single-stranded RNA molecule (about 50–100 bases). The complementary regions are spaced by a hairpin loop, therefore the name “short hairpin” RNA [9]. Intracellular presence of siRNAs

which are perfectly complementary to their target mRNAs is of crucial importance for the induction of RNAi and leads to mRNA degradation.

While siRNAs are mainly exogenous molecules, miRNAs instead are a class of RNAi inducers which derive from partially complementary double-stranded hairpin precursors of endogenous origin. Once processed, they are small single-stranded RNAs (20–22 nt long) able to modulate posttranscriptional gene silencing through repression, and at times degradation, of specific mRNA target molecules [10]. It has been estimated that miRNA-coding genes represent 1 % of the total gene population, being the biggest class of regulatory molecules. They are present in plants, higher eukaryotes, and in some viruses. miRNAs are often encoded in clusters by genes usually located in introns and, more rarely, in exons of protein-coding genes, as well as in intergenic regions [10]. RNA polymerases produce primary miRNA transcripts (pri-miRNAs) from microRNA genes [11–13]. pri-miRNAs are approximately >100 nt long and are subsequently processed into ~70-nt precursor miRNAs (pre-miRNAs) by a microprocessor complex comprised of the enzyme Drosha and a subunit DCR8 [14–16]. Pre-miRNAs are then exported into the cytoplasm by the exportin 5 protein [17, 18]. From here on, in spite of their originating difference, common features, such as the length of their mature products and their sequence-specific inhibitory functions, suggest that siRNA and miRNA have similar processing and common mechanisms. Their double-stranded precursors (shRNAs and pre-miRNAs) are indeed both cleaved by Dicer, a ribonuclease III-type protein [9, 19–21], into short 19–21 duplexes having two symmetric nucleotide overhangs at the 3' end and a 5' phosphate along with a 3' hydroxyl group. After cleavage, Dicer, together with protein complexes TRBP and PACT, loads the obtained duplexes into a nuclease-containing multiprotein complex referred to as the RNA-induced silencing complex (RISC). Once the duplex is loaded, the Ago2 protein component of the RISC then cleaves one of the two strands of the duplex, which is thus, by convention, considered the sense “passenger” strand. The antisense strand, which remains loaded into the thus activated RISC, is instead called “guide” since it acts as an adapter for the complex to mRNA targets and allows it to carry out the RNAi mechanism [22–26]. In miRNAs, the strand which most commonly plays the role of the guide is called “mature miRNA,” while the other one is called “miRNA*” [27]. Animal 3' UTR sequences often present miRNA binding sites in multiple copies, while, conversely, most miRNAs in plants, as well as siRNAs, bind their targets in their coding regions with perfect complementarity. Another fundamental difference between miRNAs and siRNAs consists, in fact, in the type of binding, which is considered a key factor in their regulatory function: miRNAs bind their target with partial complementarity, allowing

bulges and loops in duplexes. However, a key feature in their target recognition is represented by the perfect base pairing with the target in positions 2–8 of the miRNA guide, which is known as the seed region. The presence of mismatches in the central part of the duplex is usually associated to translational repression, which seems to be the default mechanism of miRNA-mediated RNAi. The cleavage of perfectly paired duplexes, which is the default RNAi mechanism in the case of siRNAs, is instead considered for miRNAs an additional feature leading to the same effect on the protein level.

The biological role of these molecules is currently being intensively elucidated, and their involvement in fundamental processes, such as apoptosis, metabolism, cell proliferation, and organism development, has been widely demonstrated.

Due to the fast and cost-effective way of disrupting genes' functions provided by RNAi-based gene knockdown techniques, great and rapid progress has been made in recent years, and siRNAs have become a standard tool routinely used in molecular genetics and functional genomics laboratory [28]. Moreover, a large number of RNAi-based potential therapeutic agents are actively being explored, while several RNAi-based therapies against several diseases such as viral infections, inflammatory diseases, and cancers have already reached preclinical and even clinical stage in development [29–31].

In light of all this, great interest has thus eventually arisen in loss-of-function studies *in vivo* in order to further investigate the precise molecular function of miRNAs in mammals, which is still unknown on the greater part. Thus in 2005, Krutzfeldt et al. devised a novel class of chemically engineered oligonucleotides able to silence endogenous miRNAs *in vivo*, which they termed “antagomirRs” [32]. These new molecules were also shown to perform efficient and stable loss-of-function phenotypes for specific miRNAs by lentivirus-mediated delivery in cultured cells [33]. In 2007, other miRNA inhibitors, called “miRNA sponges,” were devised [34]. They consisted in transcripts expressed from strong promoters and essentially containing multiple, tandem binding sites to an miRNA of interest. Once vectors encoding these competitive miRNA inhibitors were transiently transfected into cultured cells, miRNA targets were shown to be derepressed just as strongly as previously accomplished, thus suggesting a valid alternative to antagomirRs.

These results clearly show that these molecular constructs may represent a valid strategy for silencing miRNA in diseases, such as cancer, and further investigate potential therapeutic applications.

In this chapter we will give an overview of the most successful approaches and algorithms regarding RNAi design techniques and briefly describe the most popular tools that implement them.

2 Materials

The methods described in this chapter are implemented in tools publicly available online. They can be executed on any personal computer equipped with Internet connection and a browser and don't require any particular resource.

3 Methods

RNAi represents today a well-established approach for gene silencing in loss-of-function studies and genetic screens. Since its discovery in 1998, the main focuses of RNAi computational research have been the discovery of siRNA design rules and the development of siRNA efficacy and specificity prediction models. Nevertheless, a considerable number of siRNA design tools and siRNA databases are freely available online and widely employed for gene knockdown experiments.

The other side of artificial RNAi is represented by molecular tools for miRNA silencing, such as antagomiRs and sponges.

In this section we summarize the main rules for the design of siRNA, antagomiRs, and sponges and provide a brief introduction to several tools and databases which are freely available online.

3.1 Design of Functional siRNA

From a computational point of view, siRNA design is the process of choosing a functional binding site on a target mRNA sequence, which will correspond to the sense strand of the siRNA under design (typically 21–23 nt long) [35]. The siRNA antisense sequence is obtained as the complement to the sense strand. Symmetric 3' overhangs, usually dTdT, are added to improve stability of the duplex and to facilitate RISC loading, ensuring equal ratios of sense and antisense strands incorporation [6, 36, 37]. Other overhang sequences are acceptable, but some combinations, such as GG, should be avoided. The efficacy of RNAi is mostly determined by sequence-specific factors which affect the stability of the duplex ends. siRNA duplexes often have asymmetric loading of the antisense versus sense strands [38, 39]. The strand whose 5' end is thermodynamically less stable is preferentially incorporated into the RISC.

Elbashir et al. suggest to choose the 23-nt sequence motif AA(N19)TT as binding site (N19 means any combination of 19 nucleotides), where (N19)TT corresponds to the sense strand of the siRNA, while the complement to AA(N19) corresponds to the antisense strand (*see* Fig. 1a) [6].

Many different features associated to functional siRNAs have been identified in the past years and some of them are now widely accepted as standard rules in siRNA design and are implemented in

a	Antisense siRNA												Sense siRNA													
	3'						5'						3'						5'							
	dTdT			GAUGUAGCUGUUCCACCGC			CUACAUCGACAAGGUGC			GdTdT			AACTACATCGACAAGGTGCGCTT			3'			3'			3'				
	Targeted Region																									
b	Ref	73 21 36	73 67 70 71 71	67 70 71	67 70 71	73 70 71	73 71	73 71	77 73 66 73 67 70 71	5'																
	AS	3'	21 20	19 18	17 16	15 14	13 12	11 10	9 8	7 6	5 4	3 2	1	5'												
		G/T		G/C No A No G	C No A No G	No A		U No G	U No G	U	A			U No C							No C	U	A/U			
			G/C No U	A No C	No C	A/U	No C	U		A/U No C	No G		G/C No G	No G		A	No G		A/U No G No C	No C	G/T					
	S	5'	1 2 3 4	5 6 7 8	9 10	11 12	13 14	14 15	15 16	16 17	17 18	18 19	19 20	20 21	21 3'											
	Ref	66 68 74	41 65	65 65	68 49	65	41 74 65	41 21	41 49	65 65	65 74	68 41 64 21 49 36														

Fig. 1 siRNA design rules. (a) An example of target region in an mRNA sequence and the corresponding siRNA duplex with 3' overhangs. (b) Specific positional rules for siRNA design. The darker cells represent positions on the siRNA antisense (AS) and sense (S) strands. The light gray cells contain specific rules for the corresponding positions. For each set of rules, references are given (Ref). The striped gray background indicates inconsistencies of the rules, due to the different experiments that they come from

the majority of design tools. They can be classified into four different categories: (1) general binding rules, (2) nucleotide composition rules, (3) specific positional rules, and (4) thermodynamics rules. Table 1 summarizes categories (1), (2), and (4), while rules in category (3) are represented in Fig. 1b.

General binding rules refer to factors such as the position of the binding sites in the target transcript. For example, the target region should preferably be between 50 and 100 nt downstream of the start codon, and the middle of the coding sequence should be avoided. Another rule suggests pooling of four or five siRNA duplexes per target gene, in order to ensure a stronger repression.

Another class of rules concerns the siRNA nucleotide composition. A major feature, implemented by every design tool, is the G/C content, which should typically be in the range of 30–55 %, although values as low as 25 % or as high as 79 % are still associated to functional siRNAs.

Other features in this category include the presence/absence of particular motifs in the antisense strand and the absence of internal repeats.

Specific positional rules are the most numerous and regard the selection of nucleotides to prefer or avoid in specific positions of either the sense or the antisense strand of the duplex. For example,

Table 1
Rules for siRNA design

Design rule	References
<i>General targeting rules</i>	
Select the target region preferably 50–100-nt downstream of the start codon	[21]
Avoid to target the middle of the coding sequence of the target gene	[64]
Pooling of four or five siRNA duplexes per gene	[64]
<i>Nucleotide composition rules</i>	
Antisense strand with higher information content	[65]
Absence of any GC stretch >9 nt long	[66]
At least five A/U residues in the 5' terminal one-third of the antisense strand	[49, 66, 67]
A higher “A/U” content in the 3' end than that in the 5' end (sense strand)	[68]
G/C content ranges: 32–58 %, 30–52 %, 32–79 %, 36–53 %, 35–73 %, 25–55 %	[21, 41, 68–71]
Absence of internal repeats	[41]
Presence of motifs “AAC,” “UC,” “UG,” “AAG,” “AGC,” “UCU,” “UCCG,” “CUU,” “CU,” “GUU,” “UCC,” “CG,” “AUC,” “GCG,” “UUU,” “ACA,” “UUC,” “CAA” in antisense strand	[49, 70–72]
Avoid motifs “CUU,” “CUA,” “GUU,” “GU,” “GAU,” “ACGA,” “GCC,” “GUGG,” “CCC,” “GGC,” “CCG,” “GGG,” “CAG,” “GAG,” “GCA,” “AUA,” “CUG,” “AG,” “GG,” “GGA” in antisense strand	[49, 70–72]
High content of “U” in antisense strand	[72]
Low content of “G” in antisense strand	[72]
<i>Thermodynamics rules</i>	
Total hairpin energy <1	[69]
Antisense 5' end binding energy <9	[69]
Sense 5' end binding energy in range 5–9	[69]
Middle binding energy <13	[69]
Energy difference <0	[69]
Energy difference within –1 and 0	[69]
Significant ΔG difference between positions 1 and 18	[67]
High ΔG in positions 1–4, 5–8, and 13–14 in the antisense strand	[70]
Low ΔG in positions 18–19 in the antisense strand	[70]
Avoid folding of siRNA	[70]

The rules are classified in four different categories, three of which are summarized in this table (see also Fig. 1b). For each rule, or group of rules, references are given

the antisense sequence should always have an A/U base at its 5' end. This is associated to a weaker thermodynamic stability which facilitates incorporation of the strand into the RISC, as already discussed above. Other rules suggest to avoid G/C nucleotides at the 5' end of the sense strand or to have either a G or a C in position 4 of the sense strand.

Finally, thermodynamics rules refer to the global or local energy of the duplex and are partly related to the choice of nucleotides in specific positions such as the 5' end of the antisense strand, as already mentioned, or in other regions such as the middle of the duplex. Other thermodynamic features associated to siRNA efficacy include the structural accessibility of the target site and the fact that folding of siRNAs should be avoided.

These and other rules are implemented in a considerable number of siRNA design tools available online, which will be described in the next subsection. One thing that is worth mention is that all the rules were obtained from distinct experiments performed in different conditions by different labs; thus inconsistencies and contradictions are inevitable, such as the one highlighted in Fig. 1b. This is one major drawback as discussed at the end of next section.

3.2 siRNA Design

Online Resources

Table 2 provides a list of tools for the automated design of siRNA and shRNA sequences. Most tools have user-friendly interfaces which don't require any additional specifications aside from the target sequence.

OptiRNAi 2.0 is a fast tool which predicts 21–23-nt RNAi target sites on a user-provided sequence using the criteria described by Elbashir et al. in 2001 and Reynolds et al. in 2004 [36, 40, 41]. The program generates a list of up to ten siRNA target sites for each of which a score indicates how well it matches the considered features. The tool doesn't return the actual siRNA antisense sequence, which has to be manually derived as the reverse complement of the binding sites.

As for *OptiRNAi*, *siDirect 2* also accepts just the target sequence as input. It returns a table with a list of potential binding sites (including the 2-nt overhang) [42]. For each site, the corresponding siRNA duplex strands are given, together with the melting temperature (Tm) of the seed-target duplex, as a measure of thermodynamic stability. The seed-target duplex is formed between the region 2–8 of the siRNA guide strand (from the 5' end) and its target mRNA site.

Other details provided include the list of potential off-target genes for the guide and passenger strands and a graphical view of the siRNA binding sites in the target sequence.

siRNA scales are another design tool which accepts as input a target sequence and returns a list with all possible 19-nt long siRNA sequences and the predicted percentage of target mRNA copies present in the cells after siRNA-directed cleavage as a

Table 2
Tools for siRNA design

Tool	Param	sh	as	Therm	Off	Link
OptiRNAi 2.0	×	×	×	×	×	http://rnai.nci.nih.gov
siDirect 2	×	×	✓	✓	✓	http://sidirect2.rnai.jp
siRNA scales	×	×	✓	×	×	http://gesteland.genetics.utah.edu/siRNA_scales
siExplorer	×	×	✓	×	✓	http://rna.chem.t.u-tokyo.ac.jp/cgi/siexplorer.htm
RFRCDB-siRNA	×	×	✓	×	✓	http://www.bioinf.seu.edu.cn/siRNA/index.htm
OligoWalk	×	×	✓	✓	×	http://rna.urmc.rochester.edu/cgi-bin/server_exe/oligowalk/oligowalk_form.cgi
Sfold	✓	×	✓	✓	×	http://sfold.wadsworth.org
siMAX	✓	×	✓	✓	✓	http://www.operon.com/products/siRNA/sirna-overview.aspx
DSIR	✓	✓	✓	×	✓	http://biodev.cea.fr/DSIR/
siRNA scan	✓	×	✓	×	✓	http://bioinfo2.noble.org/RNAiScan.htm
RNAxs	✓	×	✓	✓	✓	http://rna.tbi.univie.ac.at/cgi-bin/RNAxs
i-Score	✓	×	✓	✓	×	http://www.med.nagoya-u.ac.jp/neurogenetics/i_Score/i_score.html
siVirus	✓	×	×	×	✓	http://sivirus.rnai.jp

Details and links are given for each tool. *Param*: it is possible to provide parameters and constraints as input. *sh*: the tool also returns the complete shRNA sequence. *as*: both antisense and sense siRNA sequences are returned. *Therm*: the tool takes into account thermodynamic features. *Off*: the tool performs or has the option for the computation of off-target sequences for the designed siRNAs

measure of efficiency [43]. Both sense and antisense strands are returned. Users can also specify to show only siRNAs with high predicted efficiency (%mRNA \leq 30).

siExplorer and *RFRCDB-siRNA* are other siRNA design tools which accept a target sequence as input and returns a list of potential binding sites and siRNAs ranked by their experimental or predicted efficiency [44, 45]. In particular, *siExplorer* also returns GC% content and, for each sequence, provides a link to perform a BLAST search for off-target evaluation. Moreover, charts with the distribution of prediction scores and the distribution of the top 10 binding sites on the target sequences are visualized. Users can choose to show the top 10, 20, or 50 results that can also be downloaded in Excel format. *RFRCDB-siRNA* also provides tested sequences in addition to the predicted ones. The tool, indeed, performs a database search on an experimentally validated siRNA database in order to find possible matches with the user-provided sequence.

OligoWalk is another siRNA design tool [46]. It returns a list of siRNA sequences with their probability of being efficient.

For each siRNA, a thermodynamics analysis is performed. In particular, the target structure is computed before and after oligo binding, and details about the energy and the Tm of the duplex, together with other energy values, are provided.

Sfold is a suite of RNA folding prediction tools, which include a program for the design of siRNA sequences based on thermodynamics features [47]. The tool allows interactive computation for up to 250-nt-long target sequences and batch processing for longer sequences. In the latter case, a notification is sent by e-mail when results are available. In addition to the target sequence, users can also provide further structural constraint information, i.e., force/prevent pairing of specific base pairs.

Several files are returned as output, containing detailed information on the predicted siRNAs and their interaction with the target, such as the siRNA duplex GC content and thermodynamics score, the total stability of the duplex and the differential stability of its ends, the target accessibility score, the average internal stability and disruption energy of the binding site, and the sum of probabilities of unpaired target bases. Other details regarding secondary structure probabilities are also given and various filters are available. Finally, the complete probability profile, the regional probability profile, and the siRNA internal stability profile are provided as graph charts.

Other tools with more sophisticated interfaces allow the specification of parameters and constraints about the siRNA to be designed and its target sequence.

siMAX is a design resource whose input consists of the target sequence together with a limited number of parameters such as GC content range, minimum and maximum distance from start and stop codons, and the mRNA motif to look for (e.g., AA(N19)TT or AA(N19)NN) [48]. A filter can be enabled in order to avoid stretches of bases of the same kind. Users can also select a species among human, mouse, or rat, as a BLAST parameter for the off-target analysis. Results include the siRNA sense and antisense strands, the distance of the binding sites from start and stop codons, the GC content, the results of the BLAST analysis, and the details about the secondary structure of the siRNA.

The tool *DSIR* allows the specification of a few prediction parameters as well, such as the siRNA length and the score threshold [49]. Filters for nucleotide stretches and immunostimulatory motifs can also be enabled. Users can choose to design either simple double-stranded siRNAs or shRNA sequences. The tool returns a list of candidate siRNAs (both strands) together with their scores, the complete shRNA sequences (if chosen), and the option to perform a BLAST search on some or all of the predicted siRNAs for the evaluation of off-target effects. Results are exportable in different formats.

siRNA scan is another tool that allows users to specify several design options, other than the length and GC content of the

siRNA, such as the 5' terminal base of the antisense strand, the minimum number of A/U base pairs in seven terminal bases of the antisense strand, and the 5' terminal base of the sense strand [50]. Stretches of 4 or 9 G/C nucleotides in a row can be avoided, and the number of mismatches in the BLAST similarity search can also be specified.

RNAxs is another tool based on thermodynamics features, concerning local target accessibility in particular [51]. Users can specify design parameters such as the accessibility thresholds, the folding energy, the sequence and energy asymmetry, and the custom sequence constraints. However, default values which have shown to give an optimal separation of functional and nonfunctional siRNAs are already pre-chosen. The output consists of the candidate siRNA sequences, together with accessibility, asymmetry, and self-folding scores. Accessibility plots of the binding sites are also shown, and a BLAST search for similarity can be easily performed by clicking on the provided links.

The tool *i-Score*, instead, is a sort of “consensus-based tool,” since, in addition to its original score, it also provides nine different designing scores based on different rule sources or other design tools, such as *DSIR* [52]. For a given target sequence, it returns the complete list of possible siRNAs together with the ten scores, duplex energy, GC content, and the length of the longest GC stretch, highlighting the top ten miRNAs according to *i-Score*, *s-BiopredSi* score, and *DSIR* score.

Finally, we introduce *siVirus*, a tool for antiviral siRNA design [53]. The system allows the design of siRNA for HIV-1, HCV, SARS, and influenza virus. For each virus, users can select multiple viral subtypes and the target regions in the viral genome. The output consists of a list of siRNA target sites (with the 2-nt overhang), mapped to one or more of the selected genomic regions. For each siRNA, the predicted efficacy according to three different set of rules is given, together with predicted off-target hits and conservation percentage in the selected sequences.

Unfortunately, as mentioned at the end of the previous section, a major issue concerning tools such as the ones described above is represented by the inconsistencies among the current siRNA design rules, mostly due to the heterogeneity of the siRNA data [35]. The Max-Planck Institute devised a principle aimed at identifying all key features relevant to miRNA design. Nevertheless, this effort has shown to yield many noneffective siRNAs which have shown to have a high false-positive rate [54].

A recent meta-rules ensemble strategy which integrates several factors, meta-design rules, and filter criteria has shown to report a 98 % rejection of false-positive miRNAs, showing great improvement over traditional state-of-the-art siRNA design programs [55]. In addition to such strategy, the integration of heterogeneous data sources can greatly alleviate inconsistency issues among siRNA design tools.

Table 3
siRNA databases

Database	Data	Link
NCBI RNAi	siRNA/shRNA sequences	http://www.ncbi.nlm.nih.gov/projects/genome/rnai/
MIT/ICBP siRNA DB	Human and mouse siRNA	http://web.mit.edu/sirna/
HuSiDa	Human siRNA	http://itb.biologie.hu-berlin.de/~nebulus/sirna/
siRecords	Mammalian siRNA	http://sirecords.biolead.org/sirecords
VIRsiRNADB	Antiviral siRNA/shRNA	http://crdd.osdd.net/servers/virsirnadb/

For each resource, the kind of data and the link are given

3.3 siRNA Databases

Several sources of siRNA sequences are publicly available online. Here we provide a brief overview of some of them, which are listed in Table 3.

NCBI's RNAi resource page allows easy access to the RNAi probes (siRNA/shRNA), stored in the NCBI database. For each probe, details about the sequence, the targets, and the hairpin, in case of an shRNA, are given. Queries are submitted through the standard NCBI interface, which allows results filtering by automatically adding the keywords “gene silencing” to the query.

The *MIT/ICBP siRNA Database* is a comprehensive database which stores and distributes information on validated siRNAs and shRNAs.

Currently the database contains siRNA and shRNA sequences against over 100 genes from three different sources: (1) sequences designed and tested by MIT researchers, (2) sequences designed by Qiagen and tested by Natasha Caplen’s group at the NCI, and (3) sequences designed by Greg Hannon and Steve Elledge and tested by the ICBP and CGAP programs at the NCI. The database can be searched by keywords (e.g., target gene name) or browsed by gene name and siRNA ID. The results include links to NCBI probe pages. The website also has a section for the submission of new validated reagents. Sequences are available for human and mouse.

HuSiDa is a database that contains sequences of published functional siRNA molecules targeting human genes and important technical details of the corresponding gene silencing experiments [56]. The database is searchable by different terms, such as gene name, cell line, transfection methods, siRNA source, and siRNA sequence.

siRecords archives experimentally tested siRNA inferred from literature [57]. Different data are available for each siRNA, such as its sequence and the alignment with the target gene, the cell types or tissues in which it was tested, the forms of the siRNA agents

(e.g., chemically synthesized oligos, vector-transfected shRNA, etc.), and the methods applied to test its efficacy (e.g., Western blot, RT-PCR, etc.). A 4-level efficacy score is assigned to each RNAi experiment based on the data provided by the authors in the original papers. In particular, an experiment is rated as “very high” if the gene product is reduced by more than 90 %, “high” if the gene product is reduced by 70–90 %, “medium” if 50–70 % repression is achieved, and “low” if less than 50 % of the gene product is reduced.

Finally, *VIRsiRNAdb* is a curated database of experimentally validated viral siRNA and shRNA-targeting genes of 42 human viruses including influenza, SARS, and hepatitis viruses [58]. Currently, the database provides detailed experimental information about 1,358 siRNAs/shRNAs and can be browsed by virus, virus family, gene, and Pubmed ID. It is also searchable by different keywords. For each siRNA, detailed information are shown, including sequence, virus subtype, target gene, GenBank accession, design algorithm, cell type, test object, test method, and efficacy. A section of the database, called *EscapeDb*, provides information about siRNAs for which viral escape is known, such as the target site mutations.

As previously mentioned, a fundamental issue in siRNA design efficacy and efficiency is represented by the lack of an effective means which would merge all heterogeneous data into the same framework, allowing results based on different data to be comparable.

Different solutions to this nontrivial problem have been devised, but their description is beyond the scope of this chapter. One solution worth mention, though, is the one proposed by Liu et al., which consists in considering each siRNA data source as a “task” and aiming at the development of a joint efficacy model for all the siRNA data sets simultaneously rather than focusing on single data sets in order to derive design rules and efficacy prediction, combining the different results at the end [35].

In conclusion, the main issues with siRNA design are essentially represented by inconsistency among the design rules and improper integration of the cross-platform siRNA data. In a more detailed analysis, aspects such as the lack of a complete feature set and an inadequate consideration of the specificity of target mRNAs could also very well be considered as major causes of the aforementioned problems.

3.4 AntagomiRs: Composition and Resources

A better understanding of the precise molecular function of miRNAs in mammals requires loss-of-function studies *in vivo* to shed light on a landscape of processes and mechanisms which are to date still largely unknown. To this end, specific and efficient silencers of endogenous miRNAs are necessary tools in aiding research. In 2005, the Krutzfeldt et al. studied the biological significance of

silencing miRNAs *in vivo* with chemically modified, cholesterol-conjugated oligoribonucleotides which they termed “antagomiRs,” disclosing a landscape of therapeutical applications regarding miRNA involved in a disease [32].

The composition of these synthetic RNA analogues basically consists in a hydroxyprolinol-linked cholesterol solid support and 2'-OMe phosphoramidites to make them more resistant to degradation. They essentially reproduce the antisense strand of the endogenous miRNA they inhibit; thus there are no actual design rules applied.

To our knowledge, due to the simple nature of their composition, there are no tools available online for antagomiR design; rather biomedical companies provide researchers with the possibility to chemically synthesize single-stranded, modified RNAs which specifically inhibit endogenous miRNA function after transfection into cells by binding to them and causing the miRNA to be subsequently degraded.

Of relevant importance as a resource for antagomiRs, the database antagomiRBase presents a collection of 53 putative antagomiR sequences for a set of 22 human miRNAs which were used as template in the design of the putative antisense sequences, using GC content and secondary structures of the stem-loop sequences of the miRNAs as parameters, along with the prediction of the free energy of the unbound antagomiRs [59]. The database presents the following information for each miRNA-antagomiR pair: the position of the guide or passenger strand of the miRNA to be targeted in its stem-loop sequence, the actual target and antagomiR sequences, respectively, and the binding energy of the hybrid duplex. A tool is also provided to allow the user to specify a 20–25-nt sequence which is used to query the database, and in case a match in the antagomiR sequences is found, it returns its secondary structure along with all its miRNA targets.

Generally, antagomiRs are efficient as a means to control the expression of specific miRNA molecules. Nevertheless, a valid alternative that could also provide the great advantage of silencing an entire family of miRNAs simultaneously is represented by a group of longer ncRNAs called “sponges,” which we will discuss in the next paragraph.

3.5 Design of Effective miRNA Sponges

Ebert et al. first introduced miRNA sponges in 2007, as an alternative to chemically modified antisense oligonucleotides for miRNA inhibition [34]. Sponges contain multiple binding sites for endogenous miRNAs and function by “absorbing” and distracting them from their natural targets, thus representing a useful tool to probe miRNA functions in a variety of experimental systems.

Sponges can be easily cloned into expression vectors and transiently transfected into cultured cells in order to efficiently derepress miRNA targets. They can also be delivered by virus-based vectors,

in order to ensure their stable expression and create continuous miRNA loss of function in cell lines and transgenic organisms [60].

MiRNA binding sites in sponges are usually specific to the miRNA seed region, allowing inhibition of a whole miRNA family. This can represent an advantage over the use of antagomirs which are highly specific for a single miRNA, being their function dependent upon full complementary match to the miRNA. A single sponge can thus efficiently replace many antagomirs, with the consequent reduction of potential off-target effects.

Sponges can also be designed to inhibit multiple miRNAs at once. This powerful feature makes them an efficient solution for loss-of-function studies over the traditional knockout model based on miRNA gene deletion, allowing the inhibition of entire genomic and/or functional miRNA clusters, in addition to families. Moreover, the deletion of a single miRNA gene which is part of a cluster could affect the other miRNAs of the cluster, while a sponge represents an efficient and easy way to avoid this side effect and still assure selective inhibition.

Up to date, there are no tools available for the automatic design of single or multiple miRNA sponges; thus we are going to describe some of the design methodologies employed so far in a few successful application.

Ebert et al. constructed PolII- and PolIII-generated sponges. PolIII sponges were constructed by inserting multiple miRNA binding sites into the 3' UTR of a destabilized GFP reporter gene driven by the CMV promoter. PolIII sponges were constructed by sub-cloning the miRNA binding region from the GFP construct into a vector containing a U6 snRNA promoter with 5' and 3' stem-loop elements. MiRNA binding sites were either bulged or perfectly complementary to the miRNAs. In the first case, a bulge at positions 9–12 of the binding site was introduced in order to prevent cleavage and degradation of the sponge. These sites were separated by 4-nt spacers, while perfect sites had no spacers (Fig. 2b). Both CMV and U6 sponges with 4–7 bulged binding sites produced stronger derepressive effects than sponges with two perfect binding sites. Fluorescence *in situ* hybridization showed that U6 sponges mainly localized to the nucleus, thus making CMV constructs a better choice. Experiments showed that sponges could selectively inhibit different miRNAs and that a sponge designed for a certain miRNA could also derepress targets of the other miRNAs of the same family. Moreover, the authors suggested 6 as the highest number of functional binding sites, as sponges with more than 6 sites showed a marginal increase in activity above 6 sites. However, they argued that sponges expressed at lower levels could benefit the presence of additional sites.

In subsequent works different types of constructs have been proposed for the expression of miRNA sponges, but from now on we will focus only on the design rules, that being the purpose of this review.

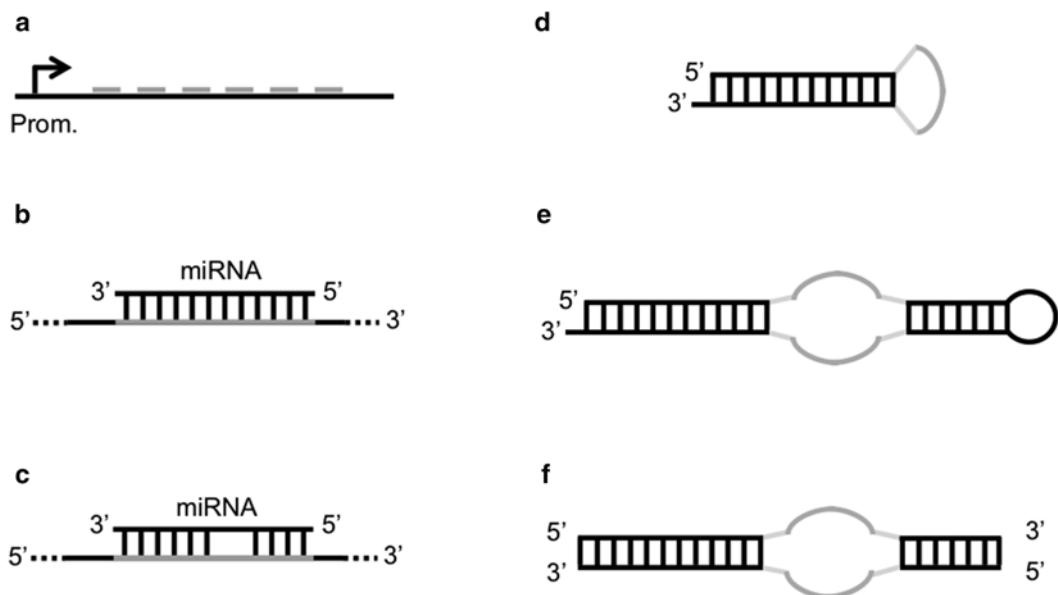


Fig. 2 miRNA sponge constructs. (a) Basic sponge with six miRNA binding sites separated by 4-nt spacers. (b) Perfect miRNA binding site on a sponge. (c) Bulged miRNA binding site on a sponge. (d) Prototype decoy consisting of a short hairpin molecule where the loop exposes a binding site for an miRNA. (e) Tough decoy (TuD) with two exposed miRNA binding sites. (f) Synthetic tough decoy (S-TuD) consisting of two fully 2'-*O*-methylated RNA strands exposing an miRNA binding site each

Haraguchi et al. reported optimal conditions for the design of TuD RNAs (tough decoy RNAs), efficient sponges with structurally accessible and indigestible miRNA binding sites [61]. The prototype decoy consisted of a short hairpin molecule where the loop exposed a binding site for an miRNA (Fig. 2c). The length of the stem is critical for the efficient transport of stem-loop structures into the cytoplasm by Exp-5. Experiments determined that the optimal stem length, associated to higher inhibitory effects, was 18 bp. Indeed, stems longer than 18 bp had a reduced binding affinity to Exp-5, while longer stems could be easily digested by Dicer in the cytoplasm. Starting from this prototype, the authors investigated several structural modifications in order to optimize the inhibitory potency of decoys. Experiments showed that the optimal TuD RNA consisted of a bulged stem-loop structure where both sides of the bulge were miRNA binding sites flanked by 3-nt linkers and the two stems separated by the bulge were 18 nt and 8 nt long, respectively (Fig. 2d). The optimal binding sites had a 4-nt insert between nucleotides 10 and 11, in order to avoid cleavage of the decoy. In a subsequent work, the same authors introduced S-TuD (synthetic TuD), a modification of TuD which consists of two fully 2'-*O*-methylated RNA strands exposing an miRNA binding site each [62]. Following the hybridization of the

two paired strands, the resultant S-TuD forms a secondary structure which resembles the corresponding TuD RNA molecule (Fig. 2e). In this work, the authors found that internal base pairing between the two miRNA binding sites on the two strands of a S-TuD can negatively affect the structural accessibility for the miRNA and reduce the inhibitory effect. In light of this, they refined the design rules and suggested that an optimal S-TuD molecule would feature two miRNA binding sites which are perfectly complementary to the target miRNA sequence and which don't form any base pairing regions longer than 9 nt. If this is not possible, the introduction of a single mutation or a 4-nt insertion in the middle region of the binding sites, as for the TuD previously described, is sufficient in many cases to abolish the base pairing without significantly affecting the affinity to the target miRNA.

These design rules proposed by Haraguchi et al. are the most sophisticated rules described so far for the design of effective miRNA sponges. However, the construction of simple RNA molecules featuring several binding sites for one or more miRNAs, separated by 2–4-nt spacers, is still the most widely used approach. This was confirmed by a recent work of Kluiver et al., in which the authors developed a methodology for the rapid generation of miRNA sponges by making use of simple constructs with up to 20 perfect or bulged miRNA binding sites, as described in the earlier work by Ebert et al. [63].

Nevertheless, it must be noted that despite the optimization of the sponge construct, different application contexts could yield different degrees of inhibition, making the verification of the success of a sponge treatment more challenging than that of genetic miRNA deletion. Thus, it is still under investigation whether *in vivo* sponge expression can effectively provide a valid alternative to genetic knockouts of miRNA families [60].

References

- Lee RC, Feinbaum RL, Ambros V (1993) The *C. elegans* heterochronic gene lin-4 encodes small RNAs with antisense complementarity to lin-14. *Cell* 75:843–854
- Fire A, Xu S, Montgomery M, Kostas S, Driver S, Mello C (1998) Potent and specific genetic interference by double-stranded RNA in *Caenorhabditis elegans*. *Nature* 391:806
- Tomari Y, Zamore PD (2005) Perspective: machines for RNAi. *Genes Dev* 19: 517–529
- Chu C-Y, Rana TM (2007) Small RNAs: regulators and guardians of the genome. *J Cell Physiol* 213:412–419
- Kutter C, Svoboda P (2008) miRNA, siRNA, piRNA: knowns of the unknown. *RNA Biol* 5:181–188
- Elbashir SM, Harborth J, Lendeckel W, Yalcin A, Weber K, Tuschl T (2001) Duplexes of 21-nucleotide RNAs mediate RNA interference in cultured mammalian cells. *Nature* 411:494–498
- Brummelkamp TR, Bernards R, Agami R (2002) A system for stable expression of short interfering RNAs in mammalian cells. *Science* 296:550–553
- Paddison PJ, Caudy AA, Bernstein E, Hannon GJ, Conklin DS (2002) Short hairpin RNAs (shRNAs) induce sequence-specific silencing in mammalian cells. *Genes Dev* 16:948–958
- Bernstein E, Caudy AA, Hammond SM, Hannon GJ (2001) Role for a bidentate ribonuclease in the initiation step of RNA interference. *Nature* 409:363–366

10. Bartel DP (2004) MicroRNAs: genomics, biogenesis, mechanism, and function. *Cell* 116: 281–297
11. Borchert GM, Lanier W, Davidson BL (2006) RNA polymerase III transcribes human microRNAs. *Nat Struct Mol Biol* 13: 1097–1101
12. Monteys AM, Spengler RM, Wan J, Tecedor L, Lennox KA, Xing Y, Davidson BL (2010) Structure and activity of putative intronic miRNAs promoters. *RNA* 16:495. doi:[10.1261/rna.1731910](https://doi.org/10.1261/rna.1731910)
13. Ozsolak F, Poling LL, Wang Z, Liu H, Liu XS, Roeder RG, Zhang X, Song JS, Fisher DE (2008) Chromatin structure analyses identify miRNA promoters. *Genes Dev* 22:3172–3183
14. Zeng Y, Yi R, Cullen BR (2005) Recognition and cleavage of primary microRNA precursors by the nuclear processing enzyme Drosha. *EMBO J* 24:138–148
15. Lee Y, Ahn C, Han J, Choi H, Kim J, Yim J, Lee J, Provost P, Rådmark O, Kim S et al (2003) The nuclear RNase III Drosha initiates microRNA processing. *Nature* 425:415–419
16. Gregory RI, Yan K-P, Amuthan G, Chendrimada T, Doratotaj B, Cooch N, Shiekhattar R (2004) The Microprocessor complex mediates the genesis of microRNAs. *Nature* 432:235–240
17. Lund E, Güttinger S, Calado A, Dahlberg JE, Kutay U (2004) Nuclear export of microRNA precursors. *Science* 303:95–98
18. Yi R, Qin Y, Macara IG, Cullen BR (2003) Exportin-5 mediates the nuclear export of pre-microRNAs and short hairpin RNAs. *Genes Dev* 17:3011–3016
19. Hamilton AJ, Baulcombe DC (1999) A species of small antisense RNA in posttranscriptional gene silencing in plants. *Science* 286:950–952
20. Zamore PD, Tuschl T, Sharp PA, Bartel DP (2000) RNAi: double-stranded RNA directs the ATP-dependent cleavage of mRNA at 21 to 23 nucleotide intervals. *Cell* 101:25–33
21. Elbashir SM, Lendeckel W, Tuschl T (2001) RNA interference is mediated by 21- and 22-nucleotide RNAs. *Genes Dev* 15:188–200
22. Matranga C, Tomari Y, Shin C, Bartel DP, Zamore PD (2005) Passenger-strand cleavage facilitates assembly of siRNA into Ago2-containing RNAi enzyme complexes. *Cell* 123:607–620
23. Miyoshi K, Tsukumo H, Nagami T, Siomi H, Siomi MC (2005) Slicer function of Drosophila Argonautes and its involvement in RISC formation. *Genes Dev* 19:2837–2848
24. Rand TA, Petersen S, Du F, Wang X (2005) Argonaute2 cleaves the anti-guide strand of siRNA during RISC activation. *Cell* 123: 621–629
25. Gregory RI, Chendrimada TP, Cooch N, Shiekhattar R (2005) Human RISC couples MicroRNA biogenesis and posttranscriptional gene silencing. *Cell* 123:631–640
26. Maniataki E, Mourelatos Z (2005) A human, ATP-independent, RISC assembly machine fueled by pre-miRNA. *Genes Dev* 19: 2979–2990
27. Okamura K, Phillips MD, Tyler DM, Duan H, Chou Y-T, Lai EC (2008) The regulatory activity of microRNA* species has substantial influence on microRNA and 3' UTR evolution. *Nat Struct Mol Biol* 15:354–363
28. Gunsalus KC, Piano F (2005) RNAi as a tool to study cell biology: building the genome–phenome bridge. *Curr Opin Cell Biol* 17:3–8
29. Kim DH, Rossi JJ (2007) Strategies for silencing human disease using RNA interference. *Nat Rev Genet* 8:173–184
30. Hadj-Slimane R, Lepelletier Y, Lopez N, Garbay C, Raynaud F (2007) Short interfering RNA (siRNA), a novel therapeutic tool acting on angiogenesis. *Biochimie* 89:1234–1244
31. de Fougerolles A, Vornlocher H-P, Maraganore J, Lieberman J (2007) Interfering with disease: a progress report on siRNA-based therapeutics. *Nat Rev Drug Discov* 6:443–453
32. Krützfeldt J, Rajewsky N, Braich R, Rajeev KG, Tuschl T, Manoharan M, Stoffel M (2005) Silencing of microRNAs in vivo with ‘antagomirs’. *Nature* 438:685–689
33. Scherr M, Eder M (2007) Gene silencing by small regulatory RNAs in mammalian cells. *Cell Cycle* 6:444
34. Ebert MS, Neilson JR, Sharp PA (2007) MicroRNA sponges: competitive inhibitors of small RNAs in mammalian cells. *Nat Methods* 4:721–726
35. Liu Q, Zhou H, Zhu R, Xu Y, Cao Z (2014) Reconsideration of in silico siRNA design from a perspective of heterogeneous data integration: problems and solutions. *Brief Bioinform* 15:292. doi:[10.1093/bib/bbs073](https://doi.org/10.1093/bib/bbs073)
36. Elbashir SM, Martinez J, Patkaniowska A, Lendeckel W, Tuschl T (2001) Functional anatomy of siRNAs for mediating efficient RNAi in *Drosophila melanogaster* embryo lysate. *EMBO J* 20:6877–6888
37. Strapps WR, Pickering V, Muir GT, Rice J, Orsborn S, Polisky BA, Sachs A, Bartz SR (2010) The siRNA sequence and guide strand overhangs are determinants of in vivo duration of silencing. *Nucleic Acids Res* 38:4788–4797
38. Khvorova A, Reynolds A, Jayasena SD (2003) Functional siRNAs and miRNAs exhibit strand bias. *Cell* 115:209–216

39. Schwarz DS, Hutvágner G, Du T, Xu Z, Aronin N, Zamore PD (2003) Asymmetry in the assembly of the RNAi enzyme complex. *Cell* 115:199–208
40. Cui W, Ning J, Naik UP, Duncan MK (2004) OptiRNAi, an RNAi design tool. *Comput Methods Programs Biomed* 75:67–73
41. Reynolds A, Leake D, Boese Q, Scaringe S, Marshall WS, Khvorova A (2004) Rational siRNA design for RNA interference. *Nat Biotechnol* 22:326–330
42. Naito Y, Yoshimura J, Morishita S, Ui-Tei K (2009) siDirect 2.0: updated software for designing functional siRNA with reduced seed-dependent off-target effect. *BMC Bioinform* 10:392
43. Matveeva O, Nechipurenko Y, Rossi L, Moore B, Saetrom P, Ogurtsov AY, Atkins JF, Shabalina SA (2007) Comparison of approaches for rational siRNA design leading to a new efficient and transparent method. *Nucleic Acids Res* 35:e63
44. Katoh T, Suzuki T (2007) Specific residues at every third position of siRNA shape its efficient RNAi activity. *Nucleic Acids Res* 35:e27
45. Jiang P, Wu H, Da Y, Sang F, Wei J, Sun X, Lu Z (2007) RFRCDB-siRNA: improved design of siRNAs by random forest regression model coupled with database searching. *Comput Methods Programs Biomed* 87:230–238
46. Lu ZJ, Mathews DH (2008) OligoWalk: an online siRNA design tool utilizing hybridization thermodynamics. *Nucleic Acids Res* 36:W104–W108
47. Ding Y, Chan CY, Lawrence CE (2004) Sfold web server for statistical folding and rational design of nucleic acids. *Nucleic Acids Res* 32:W135–W141
48. Schramm G, Ramey R (2005) siRNA design including secondary structure target site prediction. *Nat Methods* 2
49. Vert JP, Foveau N, Lajaunie C, Vandenbrouck Y (2006) An accurate and interpretable model for siRNA efficacy prediction. *BMC Bioinform* 7:520
50. Xu P, Zhang Y, Kang L, Roossinck MJ, Mysore KS (2006) Computational estimation and experimental verification of off-target silencing during posttranscriptional gene silencing in plants. *Plant Physiol* 142:429–440
51. Tafer H, Ameres SL, Obernosterer G, Gebeshuber CA, Schroeder R, Martinez J, Hofacker IL (2008) The impact of target site accessibility on the design of effective siRNAs. *Nat Biotechnol* 26:578–583
52. Ichihara M, Murakumo Y, Masuda A, Matsuura T, Asai N, Jijiwa M, Ishida M, Shinmi J, Yatsuya H, Qiao S et al (2007) Thermodynamic instability of siRNA duplex is a prerequisite for dependable prediction of siRNA activities. *Nucleic Acids Res* 35:e123
53. Naito Y, Ui-Tei K, Nishikawa T, Takebe Y, Saigo K (2006) siVirus: web-based antiviral siRNA design software for highly divergent viral sequences. *Nucleic Acids Res* 34: W448–W450
54. Gong W, Ren Y, Xu Q, Wang Y, Lin D, Zhou H, Li T (2006) Integrated siRNA design based on surveying of features associated with high RNAi effectiveness. *BMC Bioinform* 7:516
55. Mysara M, Garibaldi JM, Elhefnawi M (2011) MysirNA-designer: a workflow for efficient siRNA design. *PLoS One* 6:e25642
56. Truss M, Swat M, Kielbasa SM, Schäfer R, Herzl H, Haghemeier C (2005) HuSiDa—the human siRNA database: an open-access database for published functional siRNA sequences and technical details of efficient transfer into recipient cells. *Nucleic Acids Res* 33:D108–D111
57. Ren Y, Gong W, Zhou H, Wang Y, Xiao F, Li T (2009) siRecords: a database of mammalian RNAi experiments and efficacies. *Nucleic Acids Res* 37:D146–D149
58. Thakur N, Qureshi A, Kumar M (2011) VIRsiRNAdb: a curated database of experimentally validated viral siRNA/shRNA. *Nucleic Acids Res* 40:D230–D236
59. Ganguli S, Mitra S, Datta A (2011) Antagomirbase: a putative antagomir database. *Bioinformation* 7:41
60. Ebert MS, Sharp PA (2010) MicroRNA sponges: progress and possibilities. *RNA* 16:2043. doi:[10.1261/rna.2414110](https://doi.org/10.1261/rna.2414110)
61. Haraguchi T, Ozaki Y, Iba H (2009) Vectors expressing efficient RNA decoys achieve the long-term suppression of specific microRNA activity in mammalian cells. *Nucleic Acids Res* 37:e43
62. Haraguchi T, Nakano H, Tagawa T, Ohki T, Ueno Y, Yoshida T, Iba H (2012) A potent 2'-O-methylated RNA-based microRNA inhibitor with unique secondary structures. *Nucleic Acids Res* 40:e58
63. Kluiver J, Gibcus JH, Hettinga C, Adema A, Richter MKS, Halsema N, Slezak-Prochazka I, Ding Y, Kroesen B-J, van den Berg A (2012) Rapid generation of MicroRNA sponges for MicroRNA inhibition. *PLoS One* 7:e29275
64. Hsieh A, Bo R, Manola J, Vazquez F, Bare O, Khvorova A, Scaringe S, Sellers W (2004) A library of siRNA duplexes targeting the phosphoinositide 3-kinase pathway: determinants of gene silencing for use in cell-based screens. *Nucleic Acids Res* 32(3):893

65. Peek AS (2007) Improving model predictions for RNA interference activities that use support vector machine regression by combining and filtering features. *BMC Bioinform* 8:182
66. Ui-Tei K, Naito Y, Takahashi F, Haraguchi T, Ohki-Hamazaki H, Juni A, Ueda R, Saigo K (2004) Guidelines for the selection of highly effective siRNA sequences for mammalian and chick RNA interference. *Nucleic Acids Res* 32:936–948
67. Shabalina SA, Spiridonov AN, Ogurtsov AY (2006) Computational models with thermodynamic and composition features improve siRNA design. *BMC Bioinform* 7:65
68. Amarzguioui M, Prydz H (2004) An algorithm for selection of functional siRNA sequences. *Biochem Biophys Res Commun* 316:1050–1058
69. Chalk AM, Wahlestedt C, Sonnhammer ELL (2004) Improved and automated prediction of effective siRNA. *Biochem Biophys Res Commun* 319:264–274
70. Klingelhoefer JW, Moutsianas L, Holmes C (2009) Approximate Bayesian feature selection on a large meta-dataset offers novel insights on factors that effect siRNA potency. *Bioinformatics* 25:1594–1601
71. Liu Q, Zhou H, Cui J, Cao Z, Xu Y (2012) Reconsideration of in-silico siRNA design based on feature selection: a cross-platform data integration perspective. *PLoS One* 7:e37879
72. Wang L, Huang C, Yang JY (2010) Predicting siRNA potency with random forests and support vector machines. *BMC Genomics* 11:S2

INDEX

A

- Adenosine to inosine (A-to-I) editing 189, 231, 232, 237
Alternative
 isoforms 173, 366, 369
 splicing 137, 141, 142, 164, 170, 173–188, 287, 293, 299, 365–376, 379–391
Alternative splicing (AS)
 code 380, 383–384, 386, 387
 event 379, 380, 382–391
AntagomiR 396, 397, 405–407

B

- Base pair classification 102, 105, 108
Bioinformatics 17, 29, 32, 33, 69, 80, 123, 163, 164, 166, 180, 244, 263, 275, 280, 327–330

C

- Cis*-regulatory elements 349
Clans 349–352, 356–360, 362
CLIP-seq 295, 296
Common/consensus secondary structures 17–34
Comparative methods 4
Computational
 biology 3, 67, 262
 tools 209, 219, 307, 350
Covariance model 322, 346, 350, 354, 356
Covariation 21, 22, 24–27, 87, 90

D

- Database 4, 5, 34, 41, 43, 50, 67, 102, 103, 107–109, 111, 120, 124, 166, 167, 169, 192, 194, 199, 201, 202, 204, 210, 213–215, 217, 218, 220–223, 224, 226, 234, 243, 260, 272, 280–285, 287–289, 323, 327, 330–333, 339–342
Deep sequencing 141, 165, 214, 220, 221, 222, 231–241, 328
Differential expression 143, 164, 168–171, 244, 245, 253, 254, 295, 300
DNA-seq 138, 148, 190, 191, 193–196, 199, 201, 203, 288, 323

E

- Energy model 21, 23, 67, 80, 124, 323, 324
Environmental DNA (e-DNA) 257–277
Expressed isoforms 178
Extended secondary structure 64, 66, 68, 78, 86, 110, 111

F

- Families 65, 86, 87, 107, 214, 280, 349–362, 407, 409
Free energy minimization 3–15, 314

G

- Gene
 annotated 144, 147, 247
 coding 208, 373
 expression 50, 123, 137, 141, 142, 145, 164, 189, 209, 219, 255, 339, 341, 379, 393, 394
 profiling 137, 141
 structure 138, 174, 177, 178, 180, 182, 371
Genomics 4, 39, 74, 164, 173–175, 178, 179, 182, 183, 185, 188, 190, 192–194, 199, 201, 203, 208, 214, 238, 240, 254, 255, 259, 287, 293, 298, 327, 330, 332, 334, 339, 366, 368, 371, 372, 374, 381, 382, 384–388, 391, 396, 403, 407
Graph drawing 64

H

- Half-read seeding 149, 153
Heuristic 21, 32, 33, 53, 61, 123–132, 216, 317
High-throughput
 methods 219
 sequencing 23, 149, 219, 220, 244, 245, 279, 295

I

- Immunoprecipitation 81, 219, 220, 294, 295, 300
Infernal 323, 346, 350, 354, 361

J

- Junction 113, 114, 117, 142, 147–150, 153–159, 294, 297, 372, 388
reads 147–150, 154, 155, 157–159

M

- Maximum expected gain (MEG) estimators 20, 21, 26–32, 34
 Meta-barcoding 257–277
 Metagenomics 257, 258, 260, 262, 287
 Metatranscriptomics 279–289
 Microbiome 258, 259, 261–262, 266
 Micro RNA (miRNA)
 modeling 207, 208, 211, 217
 prediction 207–215
 target
 prediction 208–217, 219, 221–223
 recognition 210, 231, 232, 396
 Minimum free energy 4, 15, 110, 125, 213, 308–311, 314, 317, 318, 320, 324
 Modtools 49
 Motif pattern 343–346
 Multiple sequence alignment 4, 18, 19, 26, 28, 32, 34, 65, 68, 80, 85, 87, 88, 91, 313, 315, 320, 322, 340, 350
 Mutual information 21, 22, 24

N

- ncRNAs. *See* Non-coding RNAs (ncRNAs)
 Next generation sequencing (NGS) 138, 209, 219, 221, 225, 243, 258, 293, 380
 Next-generation sequencing (NGS)-Trex 243–255
 Non-canonical motifs 72
 Non-coding RNAs (ncRNAs) 17, 24–26, 34, 39, 67, 87, 123, 137, 394, 406
 detection 322

P

- Perl 6, 93, 97, 166, 174, 180, 232–234, 236, 238–240, 265, 270, 271, 340, 342, 343
 Photoactivatable-Ribonucleoside-Enhanced
 Cross-linking and Immunoprecipitation
 (PAR-CLIP) 214, 215, 220, 221, 222, 226, 295, 296, 301
 Phylogenetic tree 21, 27, 260, 288, 289, 313, 356
 Post-transcriptional regulation 50, 123, 341
 Probabilistic model 21–23, 25–33, 35, 350
 Pseudo-knots 21, 31, 35, 64–66, 69, 71, 74, 78, 79, 80, 85–91, 110, 314, 317
 Pyrosequencing 259, 261, 264–266, 268, 269, 271, 287, 289
 Python 74, 80, 84, 88, 92, 111, 169, 174, 180, 190, 191, 202, 263, 264, 273–276, 330

Q

- Quality control 137–145, 155, 282, 329, 334

R

- REDItools 190–192, 194, 199–204
 Rfam 4, 5, 41, 67, 68, 76, 77, 82, 83, 87–91, 97, 98, 214, 281, 349–362
 Ribonucleic acid (RNA)
 alignment 40–42, 87
 binding protein 294–296, 298, 300, 302, 341
 bioinformatics 17, 33, 69, 80
 editing 189–204, 232, 237, 327–337
 folding prediction 10–11, 14, 402
 free energy parameters 6, 7
 functional annotation 39
 motifs 11, 13, 61, 83, 103, 107–109, 114, 115
 processing 95
 secondary structure
 prediction 3–15, 17–35, 43, 50, 52, 61, 69, 103, 109, 126–127, 307, 308, 312, 317, 325, 340, 346
 relationship 3, 9
 structure comparison 40
 untranslated sequences 49
 visualization 64–65
 RIP-seq 293–302
 RNA interference (RNAi) 214, 394–397, 400, 404, 405
 RNAProfile 49–61
 RNA–RNA
 interaction 33, 85–87, 97, 123–132
 prediction 124–126, 128, 130
 RNA-seq 137–145, 147–161, 163–171, 190–199, 201, 203, 213, 214, 221, 223–226, 243–255, 279, 280, 294–301, 327–337, 381, 388–391
 workflow 243–255

S

- Secondary structure 3–15, 17–35, 39–43, 45, 49–61, 63–99, 101–104, 107, 109–115, 123–127, 209, 220, 225, 233, 239, 307–315, 317, 318, 322, 324, 325, 340, 341, 343, 345, 346, 349, 350, 354, 356, 357, 402, 406, 409
 Seeding extension 149, 150, 155–156
 Sequence
 alignments 4, 18, 19, 21, 26, 28, 32, 34, 41, 43, 44, 65, 68, 80, 85, 87, 91, 137, 157, 166–167, 209, 287, 313, 315, 320, 322, 340, 350, 352
 design 6, 12–14, 308, 404
 Short hairpin RNAs (shRNAs) 394, 395, 400–402, 404, 405
 16S rRNA 87, 259, 261, 282, 283
 phylogenetic analysis 280
 Small interfering RNAs (siRNA) 123, 309, 323, 394–405
 Software package 4, 6, 68, 173, 244, 295

- Spliced alignment 173, 174, 176–178,
187, 188, 252, 330, 388
- SpliceMap 148–153, 155, 157–161
- Splicing nomenclature 380, 382–384
- Structure-informed multiple sequence
alignments 87
- T**
- Taxonomy 265, 275, 288, 358–359
- Tertiary structure 39, 72, 101, 103,
104, 107–109, 114–119
- Transcription 17, 280, 300, 301, 318,
330, 332, 339, 340, 372, 379, 382, 384–386, 394
- Transcriptome
annotation 148, 295, 381, 382, 384–388
profile 244, 245
- Transcriptomics 209, 214, 219, 380
- Transcript prediction 367–368, 372–376
- U**
- Untranslated region (UTR) 50, 54, 58, 143–145,
189, 216, 217, 219, 254, 323, 330, 339–347
- W**
- Web tool 365–376

