RNAz 1.0

Predicting structural noncoding RNAs

Stefan Washietl
Department for Theoretical Chemistry
University Vienna

wash@tbi.univie.ac.at
http://www.tbi.univie.ac.at/~wash/RNAz

$Id: manual.tex,v 1.2 2006/03/24 17:56:08 wash Exp $

Contents

Preface

The function of many noncoding RNAs (ncRNAs) depend on a defined secondary structure. RNAz detects evolutionary conserved and thermodynamically stable RNA secondary structures in multiple sequence alignments and thus efficiently filters for candidate ncRNAs.

There are two main goals of this document. First we want to give you detailed technical advice on how to use RNAz. Second, we want to help you to get a well-founded understanding of the results you get from RNAz. We want to assist in a sensible interpretation of RNAz predictions — leading, as we hope, to reasonable conclusions for your application.

This document is largely based on a draft for a book chapter and it is thus organized in an idiosyncratic way. Until there is more time to write a dedicated tutorial and manual we will keep this organization.

We start with a short introduction to the problem of *de novo* prediction of ncRNAs and the RNAz algorithm. In the next part, we explain how to install RNAz and all necessary helper programs on your system. Next, we demonstrate the basic usage of RNAz including the correct formatting of the input alignments. More advanced techniques which require pre-processing steps of the input alignments are discussed afterwards. In the last section, we show you how to conduct a RNAz screen of a large number of automatically generated alignments on the example of genome-wide screen of *Saccharomyces cerevisiae*.

1 Introduction

1.1 Prediction of noncoding RNAs

In contrast to protein-gene finders which are routinely used for genome annotation, noncoding RNA (ncRNA) gene finders are still in their infancy. The main reason that hinders systematic *de novo* prediction of ncRNAs is that there are no common statistically significant features in primary sequence (e.g. open reading frames or codon bias) which could be exploited for efficient algorithms.

It is even not clear what we define as "ncRNA". There is no doubt that independent "RNA genes" with a defined molecular function such as tRNAs, microRNAs, or snoRNAs should be called ncRNAs. But the situation is not always that clear. The transcriptional activity of at least mammalian genomes is much more complex than anticipated [3]. We see mRNA-like ncRNAs, non-polyadenylated RNAs from both intronic and intergenic regions, overlapping transcripts, extensive antisense transcription, and transcribed protein-pseudogenes. In addition, there is a recent example of a noncoding transcript that only is expressed to interfere with and downregulate the transcription of a neighboring gene but the produced RNA molecule itself does not have any obvious function [9]. There is even an example of a functional RNA encoding a protein [1]. The spectrum of ncRNAs and their mode of action is very heterogeneous. One can safely assume that the full spectrum of functions is not yet discovered and that a general ncRNA gene finder is an unrealistic goal even in the long term.

However, there is a subclass of ncRNAs which — with the help of comparative genomics — can be predicted with fair accuracy. *Structural* ncRNAs have a defined and evolutionary conserved secondary structure which is of functional importance. Most of the well known "classical" ncRNAs, as for example tRNA, rRNA, RNAse P, or SRP RNA, are of this class. Pioneering work in the prediction of structural ncRNAs by comparative genomics was done by Rivas & Eddy. QRNA predicts conserved RNA secondary structures on pairwise alignments using a probabilistic approach based on a stochastic context free grammar to model RNA structure [11, 12, 10]. RNAz [16] takes a different approach. It is based on minimum free energy (MFE) structure prediction algorithms [17, 7]. It relies on the fact that structural RNAs have two characteristic features: (i) unusual thermodynamic stability and (ii) conservation of secondary structure. The following section outlines the basic principles of RNAz.

1.2 The RNAz approach

1.2.1 Thermodynamic stability

It is easy to calculate the MFE as a measure of thermodynamic stability for a sequence using e.g. RNAfold [7]. However, the MFE depends on the length and the base composition of the sequence and is, therefore, difficult to interpret in absolute terms. RNAz calculates a normalized measure of thermodynamic stability by comparing the MFE m of a given (native) sequence to the MFEs of a large number of random sequences of the same length and base composition. A z-score is calculated as $z = (m-\mu)/\sigma$, where μ and σ are the mean and standard deviations, resp., of the MFEs of the random samples. Negative z-scores indicate that a sequence is more stable than expected by chance. RNAz does not actually sample random sequences but approximates z-scores, which is much faster but of the same accuracy.

1.2.2 Structural conservation

RNAz predicts a consensus secondary structure for an alignment by using the RNAalifold approach [6]. RNAalifold works almost exactly as single sequence folding algorithms (e.g. RNAfold), with the main difference that the energy model is augmented by covariance information. Compensatory mutations (e.g. a CG pair mutates to a UA pair) and consistent mutations (e.g. AU mutates to GU) give a "bonus" energy while inconsistent mutations (e.g. CG mutates to CA) yield a penalty. This results in a *consensus* MFE E_A. RNAz compares this consensus MFE to the *average* MFE of the individual sequences $\bar{E}$ and calculates a structure conservation index: SCI $= E_A/\bar{E}$. The SCI will be high if the sequences fold together equally well as if folded individually. On the other hand, SCI will be low if no consensus fold can be found.

1.2.3 Putting it together

The two independent diagnostic features of structural ncRNAs, z-score and SCI, are finally used to classify an alignment as "structural RNA" or "other". For this purpose, RNAz uses a support vector machine (SVM) learning algorithm which is trained an a large test set of well known ncRNAs.

Using RNAz, it is possible to efficiently screen alignments for functional RNA secondary structures. It is important to note that RNAz cannot distinguish functional RNA elements which are part of ncRNAs from elements which are *cis*-regulatory elements of mRNAs.

1.3 General remarks and typographical conventions

There is no graphical user interface for RNAz. All steps are carried out on a command-line (terminal). Lines starting with a "#" are commands and you should type them into your terminal window, followed by pressing return. The "#" sign stands for your command line prompt and may look different on your system. If a command is too long for one line in this book it is separated by a backslash "\" and continues on the next line. Do *not* input the backslash, simply type in the command on one line.

All programs are implemented as filters, i.e. they read from the standard input and write to the standard output. Therefore, we make use of the pipe ("`|`") and redirection operators ("<","[>]").

You can get a online documentation on the usage of each program by using the `--help` option, e.g.:

```
# RNAz --help
```

For the Perl programs you get more detailed manual pages by using the `--man` option. All manual pages are reproduced in Appendix A in this manual.

Most command line options have a long (e.g. `--help`) and a short (e.g. `-h`) form. For didactic reasons, we use long option names throughout this manual.

2 Materials

2.1 Hardware

RNAz is generally fast. Small to medium sized data sets, as for example the yeast screen in section 3.6, can be analyzed within reasonable time on a single modern desktop or even laptop computer.

2.2 Operating system

If available, we recommend to use a Linux/UNIX system for your analysis. Also Mac OS X, in principle a full featured UNIX system, is an adaequate platform.

Alternatively, you can run RNAz also on Microsoft Windows. Most of the methods described in this manual can be carried out on Windows without any modification.

2.3 Perl

The RNAz program is bundled with a variety of helper programs which are written in the Perl programming language. To run these programs you need to have installed Perl on your system, which is most likely the case on all Linux/UNIX systems and on Mac OS X.

Perl is not part of a standard Windows system. Windows users can download it from `www.activestate.com`. Choose the latest ActivePerl MSI installer package for Windows and simply follow the installation instructions. Make sure that you have selected the "Add Perl to the PATH environment variable" and "Create Perl file extension association" options during installation.

2.4 RNAz

The RNAz program can be downloaded from: `www.tbi.univie.ac.at/~wash/RNAz`. For the examples in this manual, RNAz version 1.0 was used. For Linux/UNIX and OS X, download the file `RNAz-1.0.tar.gz`. Windows user download the file `RNAz-1.0-win32.msi`.

2.5 Optional software

Some advanced analysis steps (sections 3.6.8 and 3.6.10) require additional software to be installed on your system.

To create HTML formatted output of the results as described in section 3.6.8 you will need to have installed the Vienna RNA package (`www.tbi.univie.ac.at/RNA`) and the postscript interpreter Ghostscript (`http://www.cs.wisc.edu/~ghost/`).

To perform automatic database searches of predicted ncRNA candidates you need NCBI Blast (`ftp://ftp.ncbi.nih.gov/blast`).

2.6 Example files

Most of the example files used in this manual are part of the RNAz package. If you want to reproduce the *S. cerevisiae* screen described in section 3.6 you can download the data file from: `www.tbi.univie.ac.at/papers/SUPPLEMENTS/MiMB/`.

3 Methods

3.1 Installation of RNAz

3.1.1 Linux/UNIX and OS X

In the simplest case you can run the following series of commands to build and install RNAz:

```
# tar -xzf RNAz-1.0.tar.gz
# cd RNAz-1.0
# ./configure
# make
# su
# make install
```

This requires root privileges and installs all files under the `/usr/local` tree. The `RNAz` executable is installed in `/usr/local/bin` and you should now be able to run the program (try `RNAz --version` on a terminal window). If you do not have root privileges or experience other problems (e.g. `gcc` compiler not found) see note 4.1.

The Perl programs are installed to `/usr/local/share/RNAz/perl`. To make these programs available from other locations you can either add this directory to your `PATH` of executables environment variable or copy the Perl programs to an existing directory already in your `PATH`. In case you are not familiar on how to run Perl programs refer to note 4.2.

3.1.2 Microsoft Windows

To install RNAz on Windows simply double click on the `RNAz-1.0-win32.msi` and follow the instructions. Open a console window and type `RNAz --version` to test your installation.

3.2 Installation of optional Software

We cannot cover in detail the installation procedure of the optional software. We just give an outline how to install the Vienna RNA package and NCBI blast on a standard Linux system. Together with an existing Ghostscript installation, this will allow you to run the examples in sections 3.6.8 and 3.6.10. Windows and OS X users see Note 4.3.

To install the Vienna RNA package, get the latest `ViennaRNA-X.X.tar.gz` file from `www.tbi.univie.ac.at/RNA`. The package can be installed in exactly the same way as RNAz, using `./configure` and `make`. Please refer to the `INSTALL` document for detailed installation options. Make sure that the Perl programs in the `Utils` directory are in your `PATH` of executables.

To install NCBI Blast download the `blast-2.*.tar.gz`-package matching your platform from `ftp://ftp.ncbi.nih.gov/blast/executables/LATEST/`. Copy it to an installation directory of your choice and "untar" it. The executables are located in the `bin` subdirectory which you should add to your `PATH` variable.

3.3 Installation of example files

Move the example file `yeast-examples.tar.gz` to a directory of your choice and "untar" the file:

```
# tar -xzf yeast-examples.tar.gz
```

If you are using Windows, download the file `yeast-examples.zip` and unzip it in a directory of your choice.

3.4 Basic usage of RNAz

3.4.1 Input alignment

RNAz takes a multiple sequence alignment as input. RNAz does *not* align sequences, so you have to use other programs for creating your alignments. If you prepare your alignments manually (in contrast to automatic genome-wide alignments as in section 3.6) we recommend using Clustal W [14]. It is an easy-to-use and widely available tool which performs well on structural RNAs [4]. For hints on preparing the alignments see note 4.4.

RNAz can read two different alignment formats: Clustal W (Fig. 1A) and MAF (Fig. 1B). The Clustal W format is a concise format which is supported by many programs and thus suitable for every-day use.

For genomic screens, however, it is necessary to exactly store the genomic locations of aligned sequences. For this purpose, the MAF format was developed which requires six fields for each sequence entry:

1. a unique identifier of the source sequence,

2. the start position of the aligned subsequence with respect to this source sequence,
3. the length of the aligned subsequence without gaps,
4. "+" or "-" indicating if the sequence is in the same reading direction of the source sequence or the reverse complement,
5. the sequence length of the complete source sequence,
6. the aligned subsequence with gaps.

The full specification of the format can be found here: `http://genome.ucsc.edu/goldenPath/`
It should be noted that RNAz and all other helper programs do not make use of field 5 and also ignore the value of the "`score=`" field in the header line. So it is possible to simply fill these fields with 0 or any other arbitrary values, if the real values are not easily available.

The RNAz package contains several example files which are by default installed to `/usr/local/share/RNAz/examples`. To run the following examples change into this directory.

3.4.2 Running RNAz

As soon you have prepared your alignment you can immediately score it with RNAz. In the simplest case you type:

```
# RNAz tRNA.maf
```

The file `tRNA.maf` is that one shown in Fig. 1B and the command gives you the output shown in Fig. 1C.

3.4.3 Understanding the output

As described in the introduction, RNAz calculates various folding characteristics to classify the alignment. These are displayed in the header section of the RNAz output.

The mean single MFE is compared to the consensus MFE which results in the SCI, a measure for structural conservation (section 1.2.2). In this ideal example of a tRNA, we observe a very high SCI of 0.97. The SCI depends on the mean pairwise identity and the number of sequences in the alignment. So, it is not possible interpret the significance of a SCI-value in absolute terms. As a rule of thumb, a

A

```
CLUSTAL W (1.83) multiple sequence alignment


sacCer1         GCCTTGTTGGCGCAATCGGTAGCGCGTATGACTCTTAATCATAAGGTTAGGGGTTCGAGC
sacBay          GCCTTGTTGGCGCAATCGGTAGCGCGTATGACTCTTAATCATAAGGTTAGGGGTTCGAGC
sacKlu          GCCTTGTTGGCGCAATCGGTAGCGCGTATGACTCTTAATCATAAGGCTAGGGGTTCGAGC
sacCas          GCTTCAGTAGCTCAGTCGGAAGAGCGTCAGTCTCATAATCTGAAGGTCGAGAGTTCGAAC
                ** *   * ** ** **** ** ****  * *** *****  ****    * ****** *

sacCer1         CCCCTACAGGGCT
sacBay          CCCCTACAGGGCT
sacKlu          CCCCTACAGGGCT
sacCas          CTCCCCTGGAGCA
                * **    * **
```

B

```
##maf version=1
a score=119673.000000
s sacCer1.chr4      1352453 73 - 1531914 GCCTTGTTGGCGCAATCGGTAGCGCGTATGACTCTT...
s sacBay.contig_465   14962 73 -   57401 GCCTTGTTGGCGCAATCGGTAGCGCGTATGACTCTT...
s sacKlu.Contig1694     137 73 +    4878 GCCTTGTTGGCGCAATCGGTAGCGCGTATGACTCTT...
s sacCas.Contig128      258 73 +     663 GCTTCAGTAGCTCAGTCGGAAGAGCGTCAGTCTCAT...
```

C

```
##########################  RNAz 0.1.1  ############################

 Sequences: 4
 Columns: 73
 Reading direction: forward
 Mean pairwise identity:  80.82
 Mean single sequence MFE: -27.20
 Consensus MFE: -26.50
 Energy contribution: -23.62
 Covariance contribution:  -2.88
 Combinations/Pair:   1.43
 Mean z-score:  -2.18
 Structure conservation index:   0.97
 SVM decision value:   2.39
 SVM RNA-class probability: 0.993311
 Prediction: RNA

######################################################################

>sacCer1.chr4 1352453 73 - 1531914
GCCUUGUUGGCGCAAUCGGUAGCGCGUAUGACUCUUAAUCAUAAGGUUAGGGGUUCGAGCCCCCUACAGGGCU
(((((((.(((((........))))...((((.((((....))))))))))(((((....))))))).))))))). ( -29.20)
>sacBay.contig_465 14962 73 - 57401
GCCUUGUUGGCGCAAUCGGUAGCGCGUAUGACUCUUAAUCAUAAGGUUAGGGGUUCGAGCCCCCUACAGGGCU
(((((((.(((((........))))...((((.((((....))))))))))(((((....))))))).))))))). ( -29.20)
>sacKlu.Contig1694 137 73 + 4878
GCCUUGUUGGCGCAAUCGGUAGCGCGUAUGACUCUUAAUCAUAAGGCUAGGGGUUCGAGCCCCCUACAGGGCU
(((((((.(((((........)))).(((((.......))))).....(((((....))))))).))))))). ( -27.20)
>sacCas.Contig128 258 73 + 663
GCUUCAGUAGCUCAGUCGGAAGAGCGUCAGUCUCAUAAUCUGAAGGUCGAGAGUUCGAACCUCCCCUGGAGCA
(((((((..((((........)))).((((.........))))(((((((.....)).)))))...))))))). ( -23.20)
>consensus
GCCUUGUUGGCGCAAUCGGUAGCGCGUAUGACUCUUAAUCAUAAGGUUAGGGGUUCGAGCCCCCUACAGGGCU
(((((((..((((........)))).(((((.......))))).....(((((.......))))))))))))).
                                                    (-26.50 = -23.62 +  -2.88)
```

Fig. 1. Supported alignment formats and RNAz output. (A) Clustal W format, (B) MAF format (sequences have been shortened due to space restrictions), (C) Output of RNAz on the MAF file shown in (B).

SCI near or even above the mean pairwise identity is "good" and might indicate structural conservation. For example, given an alignment with five sequences and a mean pairwise identity of 60%, a SCI of 0.75 can be regarded as strong hint for a conserved fold. On the other hand, on a pairwise alignment with 90% identity, SCI=0.75 does not indicate a conserved fold at all.

The second characteristic is thermodynamic stability, which is expressed as the mean z-score of the sequences in the alignment (see section 1.2.1). z-scores of MFEs are not exactly normal distributed, so you cannot directly give a statistical significance for your z-score. However, mean z-scores below -3 or -4 generally indicate very stable structures, that should arise only in rare cases by chance. Also here, one has to consider the overall sequence divergence in the alignment. On a pairwise alignment with 90% identity a z-score of -4 is much more likely to occur by chance than on an alignment of six sequences with only 60% identity.

Apart from SCI and z-score, there are a few other values displayed in the RNAz output. If you are wondering what they mean, see note 4.5.

RNAz assists you in the final classification by providing an overall "RNA-class probability", or "P-value". It is important to know that this is *not* a P-value in a strict statistical sense, simply because there is no underlying statistical model. Instead, RNAz uses a rather *ad hoc* machine learning technique to calculate this value. If $P >$0.5, the alignment is classified as "RNA". The false positive rate at this cutoff was found to be $\approx$ 4%, i.e. we expect 4 positive hits in 100 random alignments. For many applications it is useful to set a more stringent cutoff of P=0.9 with an associated false positive rate of $\approx$ 1%. Reasons why estimations of false positives must always be taken with caution are given in Note 4.6.

It turned out to be a useful practice to use $P =$0.5 and $P =$0.9 as two main levels of significance. A more sophisticated interpretation of the P-value without considering the other values is generally not useful. In most cases you cannot say that, for example, a hit with $P =$0.97 is more reliable than a hit with $P =$0.95. See Note 4.7 on how to assess the reliability of a hit based on other criteria.

In the lower part of the RNAz output you explicitely see the predicted structures for your sequences. You get structure predictions for each single sequence and a consensus structure prediction for the whole alignment. The predicted structures are given below the sequences in a "dot-bracket" notation. Each base-pair in the secondary structure is indicated by a pair of brackets: "`(`" and "`)`". Unpaired bases are shown as dot: "`.`". Next to the structure you see the MFE in kcal/Mol. You can get a graphical output by using RNAalifold of the ViennaRNA package.

3.5 Advanced usage of RNAz

3.5.1 Analyzing forward and reverse strand

For a given alignment, a putative RNA can either be read in the forward direction or in the reverse complementary direction. Therefore, both reading directions should be scanned. By default, only the forward direction is scored, but you can use the `--forward`, `--reverse` and `--both-strands` flags to explicitely specify the reading direction.

If you have a strong RNA signal in one strand you can observe in many cases also a signal in the reverse complement. Usually the signals (SCI, z-score, consensus MFE) are stronger in the "correct" direction. In most cases this also goes along with a better P value. That is not always the case and, therefore, RNAz uses a separate SVM decision model to predict the correct strand. Please note, that in version 1.0 this is still an experimental feature. With the following command you can analyze both strands of the tRNA and, in addition, activate the strand prediction:

```
# RNAz --both-strands --predict-strand tRNA.maf
```

In this example, the signal from both strands are almost indistinguishable and also the P-values are almost the same (0.993 and 0.999). RNAz still suggests the correct (forward) strand and displays a "strand class probability":

```
# Strand winner: forward (0.88)
```

3.5.2 Scoring alignments with more than six sequences

RNAz is currently limited to alignments with not more than six sequences. If you have more than six sequences in your alignment, you have to reduce the number either manually or use the `rnazSelectSeqs.pl` program to filter your alignment before you put it into RNAz:

```
# rnazSelectSeqs.pl miRNA.maf | RNAz
```

The file `miRNA.maf` contains 12 aligned microRNAs. With default parameters, `rnazSelectSeqs.pl` selects a subset of six sequences trying to reach an optimal mean pairwise identity around 80%.

The default behaviour can be customized in various ways (use `--help` for details). The following command, for example, samples three different alignments with four sequences each.

```
# rnazSelectSeqs.pl --num-seqs=4 --num-samples=3 miRNA.maf | RNAz
```

By default, the first sequence in the alignment is always in the set of selected sequences. This is the desired behaviour for genomic screens, where one usually likes to retain a reference sequence.

3.5.3 Scoring long alignments

RNAz cannot score alignments longer than 400 columns. In practice, it is generally advisable that you score long alignments, say $>$200 columns, in shorter, overlapping windows. For general purpose screens we recommend a window size of 120. This window size appears large enough to detect local secondary structures within long ncRNAs and, on the other hand, small enough to find short secondary structures without loosing the signal in a much too long window.

The file `unknown.aln` contains a noncoding region conserved in vertebrates. You can scan it for RNA secondary structures by typing:

```
# rnazWindow.pl --window=120 --slide=40 unknown.aln \
                | RNAz --both
```

If you look trough the results you see that RNAz does not predict an RNA in this region. On UNIX like system you can add "`| grep Prediction`" to get a quick overview on the results. The `rnazWindow.pl` program has numerous additional functions and will be used again in section 3.6.

3.6 Large scale genomic screens

3.6.1 Overview

An analysis pipeline suitable for scanning a large number of genomic alignments is outlined in Fig. 2. In the following, we demonstrate the usage of this pipeline on the example of a genomic screen of *Saccharomyces cerevisiae*. We want to describe the method as general as possible and we will focus here mainly on technical details. A paper describing the results of a comprehensive RNAz screen in yeast is in preparation [13].

3.6.2 Choosing raw input alignments

Choosing a reasonable set of input alignments is one of the most important steps during the analysis. There are a variety of different programs available to gen-

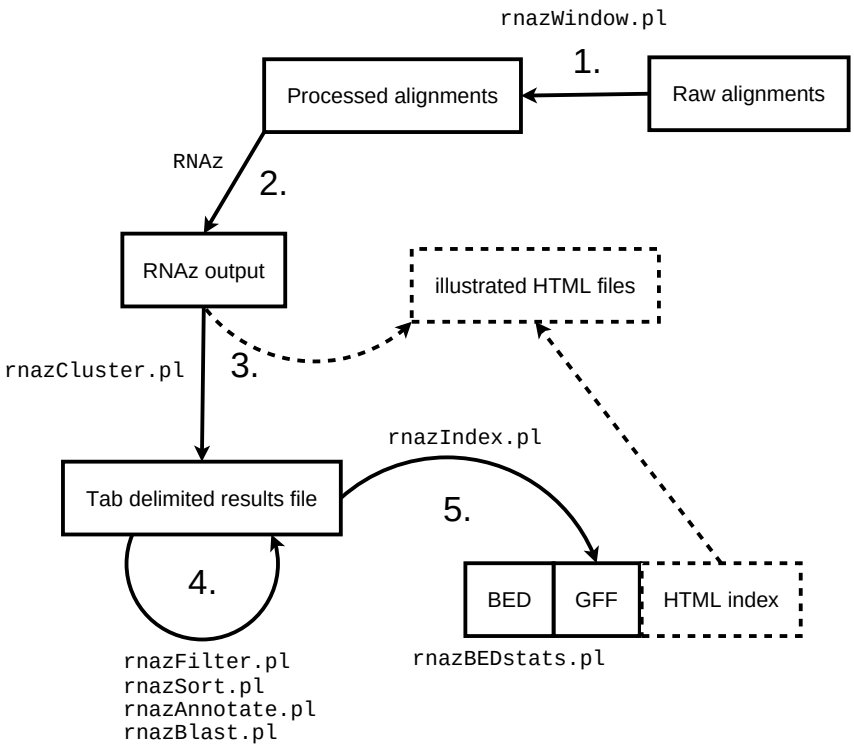

Fig. 2. Analyzing pipeline illustrating the use of RNAz and the helper programs. (1) `rnazWindow.pl` slices the input alignments in overlapping windows and performs a variety of filtering and pre-processing steps. (2) The processed alignments can be scored with the RNAz program (3) Overlapping hits are merged with `rnazCluster.pl`. In addition, all relevant data is extracted from the raw output and stored in a tabulator delimited data file. Using the `--html` option, `rnazCluster.pl` generates a tree of HTML pages with illustrations of the predicted structures. You need additional software for this step to work. (4) The results can be filtered, sorted and annotated in various ways. All programs read a tab-delimited file and write a tab-delimited file. (5) Using `rnazIndex.pl`, the tab-delimited data files can be exported to standard formats as GFF and BED. It is also possible to create a HTML formatted index file for the optional HTML output created in step 3.

erate genome-wide alignments. Here, we use Multiz alignments of up to seven *Saccharomyces* species which can be downloaded from the UCSC genome browser (`genome.ucsc.edu`). In principle, we could use *all* alignments covering the complete genome. The biggest problem in large genomic screens is probably specificity. We have a relatively constant background signal of false positives. The more sequences we put into the screen, the more false positives we get out. It is, therefore, a good idea to choose the input set as small as possible (trying not to discard any interesting regions of course). In our case, we only analyze the intergenic regions, i.e. we discard any coding regions and all other annotated features (pseudogenes, repeats, ARS elements, ...). We retain known ncRNAs as positive control in the set. The selection was easily accomplished using the "Table browser" feature of the genome browser. We finally obtained a MAF alignment (`input.maf`) with 10,822 alignment blocks, covering 983,947 bases of the genome (see section 3.6.11 how to get these numbers out of a MAF file.).

3.6.3 Pre-processing raw alignments

As described in section 3.5.3, it is necessary to score long alignments in overlapping windows. Given the partly poor quality of automatically generated genome-wide alignments additional pre-processing steps are required to filter out gap-rich regions, dubious aligned fragments or low complexity regions. All pre-processing is done by the `rnazWindow.pl` program which, per default, performs the following steps:

1. Slice alignments in overlapping windows of size 120 and slide 40.
2. Check each pairwise alignment of the reference sequence (= first sequence) to all other sequences and, after removing common gaps, discard sequences with more than 25% gaps in this pairwise alignment.
3. Discard any sequences which are outside the definition range of RNAz (e.g. $<$50 nucleotides, GC content $>$0.75).
4. Discard the complete alignment if either the reference sequence was discarded in a previous step or only the reference sequence is left (i.e. number of sequences $<$2)
5. If the number of sequences is $>$6, choose a subset of 6 sequences with mean pairwise identity optimized to a target value of 80%.
6. Remove all sequences which are 100% identical. Never remove the reference sequence and if all sequences are identical retain only a pairwise alignment.

All these steps can be customized with the appropriate command-line parameters. Here we use the default settings. We define, however, a minimum number of four

sequences in the alignment retaining only regions which are well conserved across several species:

```
# rnazWindow.pl --min-seqs=4 input.maf > windows.maf
```

This command will take a few minutes.

3.6.4 Running RNAz

The file `windows.maf` is now ready for being scored with RNAz. We use the `--both-strands` parameter to score both the forward and the reverse complement strand. We also set `--show-gaps` which means that the output is shown including the gaps. With this option it is possible to recover the complete alignment from the RNAz output file which is useful in later steps of the pipeline. Finally, we set a P value cutoff of 0.5, meaning that only positive predictions are stored resulting in a much smaller output file.

```
# RNAz --both-strands --show-gaps --cutoff=0.5 windows.maf > rnaz.out
```

This will take approximately one hour on a modern desktop computer but may vary depending on your system.

3.6.5 Clustering the results

The file `rnaz.out` now holds all windows that have a positive RNAz signal with $P > 0.5$. It is possible that several windows cover the same genomic region. Overlapping windows are therefore clustered in *loci*:

```
# rnazCluster.pl rnaz.out > results.dat
```

This command assigns each window a consecutively numbered "window ID" and each group of overlapping windows a "locus ID". For each window and each locus all relevant data (use `--help` for details) is stored in a tabulator separated text file.

Inspecting the file `results.dat`, we see that we have 1104 windows which can be grouped in 454 loci.

It is important to note that the term "locus" must *not* be understood in the sense of a genetic unit. It is, of course, possible that several loci of our procedure cover one long ncRNA gene.

At this point we also want to add that we are painfully aware of the fact that the process of first slicing the alignments and the re-cluster them is not optimal. Ideally one would like to predict conserved RNA structures *locally* without sliding windows. Although this should be possible [8] and we are working on a local version of RNAz, the sliding window approach is currently the only reasonable protocol.

3.6.6 Filtering and sorting the results

The data file now contains the raw data of all hits. In the following analysis steps, one usually wants to filter and sort candidates by various criteria. For this purpose you can use the programs `rnazFilter.pl` and `rnazSort.pl`. For example,

```
# rnazFilter.pl "P>0.9" results.dat
```

lists all windows that have a P-value higher than 0.9. For hints on how to formulate more complex filtering expressions see Note 4.8. With the `--count` option you can count the hits. We have 670 Windows in 303 loci on the P¿0.9 significance level. In addition, we can sort the hits:

```
# rnazFilter.pl "P>0.9" results.dat | rnazSort.pl combPerPair
```

This sorts the output by the “Combinations/Pair” value, i.e. by compensatory mutations supporting the structure (explained in Note 4.5).

3.6.7 Exporting the results to standard annotation formats

Using different combinations of `rnazFilter.pl` and `rnazSort.pl` you can create various sub-selections of the complete data from `results.dat`. You always get a tabulator delimited data-file. The program `rnazIndex.pl` helps you to convert these kind of data files into the standard annotation formats GFF (`--gff`) or BED (`--bed`). GFF (`http://www.sanger.ac.uk/Software/formats/GFF/`) is a widely used format supported by many programs. BED (`http://genome.ucsc.edu/FAQ/FAQ`) is the native annotation format for the UCSC genome browser but is generally useful because of its simplicity (in its simplest form it is a list of genomic locations: `sequenceID start stop`).

The following command creates a GFF file from all results:

```
# rnazIndex.pl --gff results.dat > results.gff
```

3.6.8 Visualizing the results on a website

It is often insightful to manually check individual predictions, for example by analyzing different illustrations of consensus structures (see Note 4.7). The creation of the necessary files is a tedious task which, however, can easily be automatized. If you run the cluster command from section 3.6.5 with the option `--html`,

```
# rnazCluster.pl --html rnaz.out > results.dat
```

the program generates image files for all hits. For the `--html` option to work, you need to have installed the Vienna RNA package (including the Perl programs of the `Utils` directory) and the program Ghostscript, see section 2.5. `rnazCluster.pl` creates a subdirectory called `results`, which, in turn, has a subdirectory `locusN` for each locus. In the `locusN` directories you find the image files together with an `index.html` which arranges the images for each locus on a web-page. You can open the index files using your favorite web-browser.

To get an HTML formatted table of all hits linking to the sub-pages for each locus, you can use `rnazIndex.pl` with the `--html` option:

```
# rnazIndex.pl --html results.dat > results/results.html
```

3.6.9 Comparing hits to known annotation

Once you have a list of predicted RNAs, you may want to add additional annotations to your predictions. You can simply add additional fields to the tabulator separated data file at your convenience. Here we demonstrate this by comparing our prediction with the known ncRNA annotation from the *Saccharomyces* genome database. The program `rnazAnnotate.pl` checks each predicted locus for overlap with an annotation file in BED format:

```
# rnazAnnotate.pl --bed ../sgdRNA.bed results.dat > annotated.dat
```

We find that out of 454 predicted loci, 280 overlap with known ncRNAs (of the 303 loci with $P > 0.9$, 215 are known ncRNAs). We detect all sorts of different ncRNA classes (tRNAs, rRNA, snRNAs, snoRNAs, RUFs [10], and other ncRNAs like telomerase RNA or RNAseP,...) Most of the known 373 ncRNAs in yeast are tRNAs (275), which are partly difficult to detect in this screen because most of them are $\approx$ 100% conserved (i.e. no covariance information).

Without providing a detailed sensitivity analysis for this specific yeast screen, we want to add that sensitivity highly depends on the ncRNA class. MicroRNAs, for

example are easy to detect because of the high thermodynamic stability of the hairpin precursor. On the other hand, C/D type snoRNAs for example are generally difficult to detect because they lack a pronounced secondary structure. We miss completely ncRNAs which do not depend on a secondary structure for their function, as for example the yeast *SER3* regulating RNA [9] which, as expected, does not show up in this screen.

3.6.10 Annotating hits with database search

Another possibility to annotate predicted ncRNAs is to compare the sequences to databases of known ncRNAs. In the following we match the predicted loci against the Rfam database [5] using a simple Blast sequence search. Alternatively, one could use more sensitive methods which also incorporate secondary structural information (e.g. Infernal [2]). To run this example, you need the *S.cerevisiae* sequence files, the Rfam database file and a working NCBI Blast installation.

First change into the directory `rfam` and run:

```
# formatdb -t rfam -i rfam -p F
```

This command creates the index files for the file `rfam`, which is a Fasta formatted file with all entries of the database. You now can run:

```
# rnazBlast.pl --database rfam --seq-dir=seq \
               --blast-dir=rfam results.dat >annotated.dat
```

This program takes the *S. cerevisiae* reference sequence for each locus and runs a Blast search against the Rfam database. If there is a hit with an expectation value below some cutoff (default: $E < 10^{-6}$), the name of the matching database query is added as a new field to the data file. Please note that you have to specify the locations of the sequence data files and the blast index files on the command line.

3.6.11 Estimating false positives and gathering statistics

To get an impression of the false-positive rate of a specific screen it is useful to do a control screen on randomized alignments. The command

```
# rnazRandomizeAln.pl input.maf > random-input.maf
```

will produce a randomized version of the input alignments by shuffling the positions in the alignments. The program aims to remove any correlations arising from

Tab. 1. Statistics of the yeast example screen

	P¿0.5	P¿0.9
Predicted loci	454	303
Known ncRNAs	280	215
Loci without annotation	174	88
Predicted bases	60,834	44,082
Fraction of input alignments (%)	10.6	7.7
Predicted loci random	102	39
Predicted bases random	12,823	6,017
Fraction of input alignments random (%)	2.2	1.0

a natural secondary structure while preserving important alignment and sequence characteristics as for example mean pairwise identity or base composition [15].

We repeated the complete analysis with the randomized alignments and we get 102 and 39 loci, on the $P >$0.5 and $P >$0.9 level, respectively.

Table 1 summarizes all results of this example screen. There are a few programs which help you to gather statistics on your data. For example,

```
# rnazIndex.pl --bed results.dat \
               | rnazBEDsort.pl | rnazBEDstats.pl
```

gives you detailed information on the predicted loci, including the covered genomic region in nucleotides. This command first exports the results as BED file, sorts the results by the genomic location and, finally, evaluates the coordinates in the BED file. If you want to get statistics on your input alignments, you can use a command like this:

```
# rnazMAF2BED.pl --seq-id=sacCer windows.maf \
                 | rnazBEDsort.pl | rnazBEDstats.pl
```

`rnazMAF2BED.pl` converts a MAF formatted alignment file to coordinates in BED format. With `--seq-id` you specify which sequence is used as reference.

Using these tools, you find for example that in the random control 1.0% of the input sequences are predicted as RNA on the $P >$0.9 level. This is exactly the false positive rate as expected (section 3.4.3). The absolute number of false positives, however, strongly depends on your specific screen. In this example we have 88 hits $P >$0.9 without RNA annotation and find that 39 hits should be expected by

chance. So we must expect that roughly half of our predictions are false positives. On the other hand, this implies that the other half of the predicted loci should be real functional RNA structures, either as part of a ncRNA or as regulatory element of a mRNA. However, one always have to bear in mind possible shortcomings of this kind of random control, see Note 4.6.

4 Notes

4.1 Custom installation of RNAz

The installation process using `./configure` and `make` should work on all UNIX-like systems. If you get error messages it may be necessary that you install additional "developer packages". On some Linux distributions, for example, there is no C-compiler installed by default. Also on OS X it is necessary that you have installed the "XCode" tools.

If you do not have root privileges or want to install RNAz into a different location than `/usr/local/` (e.g. your home directory) you can use the following command:

```
# ./configure --prefix=/home/stefan --datadir=/home/stefan/share
```

This installs the executable to `/home/stefan/bin` and the example files, Perl programs and other data to `/home/stefan/share/RNAz`. Please note that the `bin` directory must be in your `PATH` of executables if you want to call the `RNAz` executable without specifying the complete path.

4.2 Running the Perl programs

Since different people usually like to have their scripts in different locations, the Perl programs are *not* installed to `/usr/local/bin` by default. They are installed to
`/usr/local/share/RNAz/perl`. To make them available from other locations, copy all files from this directory to a directory which is included in your `PATH` of executables, e.g.:

```
# cp /usr/local/share/RNAz/perl/* /usr/local/bin
```

Alternatively, you can add the directory with the Perl programs to your `PATH` variable by editing your `.bashrc` or `.cshrc` file in your home directory.

In any case, it is important that the Perl module file `RNAz.pm` resides in the same directory as the Perl programs (`*.pl`). All the Perl programs depend on this module file.

Another important point is, that the Perl programs expect that the path of the Perl executable is `/usr/bin/perl`. This is the standard location on almost all Linux/UNIX systems and OS X. If your Perl installation is different you have to customize the first line of all the Perl programs according to the location of your `perl` executable.

On a Windows system the Perl programs should work if you have installed Perl as described in section 2.3 and set the `Path` variable as described in section 3.1.2.

4.3 Optional software on Windows on OS X

To install the necessary software for the programs in sections 3.6.8 and 3.6.10 might be a bit tricky on Windows and OS X.

You can install the Vienna RNA package and NCBI Blast without problems on OS X by following the instructions in section 3.2. However, unlike on a Linux system, Ghostscript is not installed per default. You can try to get a pre-compiled package from `fink.sourceforge.net` or `darwinports.opendarwin.org`. Alternatively, you can download the source from `http://www.ghostscript.com/` and build the package with `./configure` and `make`.

On Windows you can install Ghostscript through a simple installer file which you can download from `http://www.ghostscript.com/`. Follow the installation instructions. Locate the newly installed file `gswin32c.exe` and copy it to a folder which is in your `Path` (e.g. the folder, where the `RNAz.exe` executable resides). Rename the file to `gs.exe`.

Windows users do not have to install the Vienna RNA package. The relevant programs are part of the RNAz windows installer.

To install NCBI blast on windows, create a new folder (e.g. `c:\Program Files\blast`) and download the `blast-2.*-win32.exe` file from `ftp://ftp.ncbi.nih.gov/blast`. Within the new folder, double click on the `blast-2.*-win32.exe` file which extracts the programs and data. Add the `bin` subdirectory to your `Path`: Right-click *My Computer*, then click *Properties*. Select *Advanced*/*Environment variables*/*New*. Add the complete path of the blast `bin` directory to the variable *Path*, use "`;`" as separator.

4.4 Creating the input alignments

RNAz can only detect a conserved structure if this structure is accurately reflected in the alignment. Therefore, the quality of the alignment is crucial for the success of the analysis. In practice, we found that if your alignment has a mean pairwise identity above appr. 60% simple sequence based progressive, global alignment methods yield reasonable results and there is not much difference between methods. One of the best programs for aligning RNAs is Clustal W. For genome-wide alignments we have only experience with Multiz alignments. Also these alignments are of reasonable quality and there is generally no need for re-alignment. We suppose that also other genome-wide alignment methods produce suitable alignments as long the aligned regions are of sufficient similarity (mean pairwise identity somewhere around 60% or above). In cases with sequences below 60% identity, simple sequence based methods usually do not find an optimal *structural* alignment. Altough in principle structural enhanced alignments could help here, this alternative is not relevant in practice. First, there are hardly any structural multiple sequence alignment programs available. Second, current approaches are much too slow to use them for every-day analysis. Third, RNAz is not trained on structural alignments. In contrast to pure sequence based alignment, you would get unusual high SCIs. This could confuse the decision model and you would get unpredictable results.

4.5 Additional output values

The consensus MFE which is calculated by the RNAalifold algorithm (see section 1.2.2) can be split in two terms. One is the "energy contribution", which is the folding energy from the standard energy model. The "covariance contribution" is the part which comes from the additional "bonus" or "penalty" energies for compensatory/consistent and inconsistent mutations, respectively. If the covariance term is negative, there are more compensatory mutations than inconsistent mutations.

RNAz also calculates another value quantifying compensatory/consistent mutations: "Combinations/Pair". This is the number of *different* base pair combinations in the consensus structure divided by the number of pairs in the consensus structure. Both the covariance contribution of the consensus MFE and the "Combinations/Pair" are mainly useful for final sorting a set of equally good predictions with have been filtered using other criteria (e.g. P or z-scores).

RNAz uses a SVM algorithm for classification. The raw output of the SVM is the so-called "decision-value". This real-valued number is positive if the prediction is "RNA" and negative otherwise. From this value we calculate the more intuitive "RNA class probability" or "P-value" which is 0.5 for a decision value of 0. In some cases, the raw decision value can be more convenient than the P value (e.g. if you want to plot the distribution of RNAz results).

4.6 Estimating false positives

The RNAz classification model is trained on a test set consisting of natural RNAs as positive examples and randomly shuffled alignments as negative examples. Thus, any signal reported by RNAz is relative to an *artificial* background. Although this null model of shuffled sequences is probably the most sensible choice possible, one cannot assume that it behaves exactly like the *natural* background of real sequence data. Also the estimation of false positive rates is based on shuffled sequences. We want to stress that, therefore, such an estimation of false positives must be regarded as a lower bond since one cannot rule out the possibility that non-random patterns in natural sequences cause a higher rate of false positives than one observes in synthetic random sequences. In particular, the z-score calculation might be affected by such effects. For example di-nucleotide content could bias the MFE structure prediction. As an opposite effect one must consider the possibility that the shuffling procedure cannot remove all secondary structure signals and thus *over*estimates the real false positive rate. If you shuffle an alignment with many compensatory mutations, the number of "compatible columns" stays the same, allowing for compensatory mutations also in the shuffled alignment.

4.7 Manual inspection of candidates

If you have a hit with $P >$0.9, you have approximately a chance of 1 in 100, that this arises through pure chance (but see als Note 4.6). It makes sense to critically look at a hit. Sometimes the signal only comes from a low z-score of borderline significance and there is no evidence for structural conservation. Sometimes the complete alignment looks pathological (weird gap patterns, low complexity regions etc.) which suggests that this is not a relevant structure. It is useful to analyze a predicted structure with RNAalifold and its visualization methods. Visual inspection of a color coded alignment and the consensus structure gives you an idea about compensatory mutations supporting the structure and inconsistent mutations which do not support the structure. It must be noted that many ncRNAs in real life-data are not supported by compensatory mutations, still they can be detected based on the stability and/or the SCI. The SCI implicitely also considers the mutational pattern outside of stems. To conclude, the P value efficiently filters your data for candidates, but only the complete picture can help you in your decision on the relevance of a hit.

4.8 Advanced filtering

Filtering the tab-delimited data files using standard UNIX tools like `grep` or `awk` is difficult because of the special window/locus grouping of the data. You can use the `rnazFilter.pl` program. The filter statement uses the field names (e.g.

`z,SCI,combPerPair`, see `--help` for a complete list) and standard logical operators as used in the Perl language: `>` (greater than), `<` (smaller than),`==` (equals numerically), `eq` (equals string), `not`, `and`, `or`, `=~/regex/` (pattern match). In addition you can use brackets to group and combine statements. For example the following statement gives you all windows with $P > 0.9$ and $z < -3$ on chromosome 13:

```
# rnazFilter.pl "P>0.9 and z<-3 and seqID=~/chr13/" results.dat
```

It is important that *everything* you put in the filter statement is evaluated by the Perl interpreter. This can be potentially harmful, so take care.

References

1. Chooniedass-Kothari S, Emberley E, Hamedani MK, Troup S, Wang X, Czosnek A, Hube F, Mutawe M, Watson PH, and Leygue E. **The steroid receptor RNA activator is the first functional RNA encoding a protein.** *FEBS Lett*, 2004. **566**:43–7.
2. Eddy SR. **A memory-efficient dynamic programming algorithm for optimal alignment of a sequence to an RNA secondary structure**. *BMC Bioinformatics*, 2002. **3**:18.
3. Frith MC, Pheasant M, and Mattick JS. **The amazing complexity of the human transcriptome.** *Eur J Hum Genet*, 2005. **13**:894–7.
4. Gardner PP, Wilm A, and Washietl S. **A benchmark of multiple sequence alignment programs upon structural RNAs.** *Nucleic Acids Res*, 2005. **33**:2433–9.
5. Griffiths-Jones S, Moxon S, Marshall M, Khanna A, Eddy SR, and Bateman A. **Rfam: annotating non-coding RNAs in complete genomes**. *Nucleic Acids Res*, 2005. **33**:D121–D124.
6. Hofacker IL, Fekete M, and Stadler PF. **Secondary structure prediction for aligned RNA sequences.** *J Mol Biol*, 2002. **319**:1059–1066.
7. Hofacker IL, Fontana W, Stadler PF, Bonhoeffer LS, Tacker M, and Schuster P. **Fast folding and comparison of RNA secondary structures**. *Monatsh Chem*, 1994. **125**:167–188.
8. Hofacker IL, Priwitzer B, and Stadler PF. **Prediction of locally stable RNA secondary structures for genome-wide surveys.** *Bioinformatics*, 2004. **20**:186–190.
9. Martens JA, Laprade L, and Winston F. **Intergenic transcription is required to repress the *Saccheromyces cerevisiae SER3* gene**. *Nature*, 2004. **429**:571–574.
10. McCutcheon JP and Eddy SR. **Computational identification of non-coding RNAs in *Saccharomyces cerevisiae* by comparative genomics.** *Nucleic Acids Res*, 2003. **31**:4119–4128.
11. Rivas E and Eddy SR. **Noncoding RNA gene detection using comparative sequence analysis.** *BMC Bioinformatics*, 2001. **2**:8.
12. Rivas E, Klein RJ, Jones TA, and Eddy SR. **Computational identification of noncoding RNAs in *E. coli* by comparative genomics.** *Curr Biol*, 2001. **11**:1369–1373.
13. Steigele S, Stadler PF, and Nieselt K. **Structured RNA motifs in the yeast genome**. In preparation.
14. Thompson JD, Higgins DG, and Gibson TJ. **CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice.** *Nucleic Acids Res*, 1994. **22**:4673–4680.
15. Washietl S and Hofacker IL. **Consensus folding of aligned sequences as a new measure for the detection of functional RNAs by comparative genomics.** *J Mol Biol*, 2004. **342**:19–30.
16. Washietl S, Hofacker IL, and Stadler PF. **Fast and reliable prediction of noncoding RNAs**. *Proc Natl Acad Sci USA*, 2005. **102**:2454–2459.
17. Zuker M and Stiegler P. **Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information.** *Nucleic Acids Res*, 1981. **9**:133–148.

A Manual pages

A.1 RNAz

Detecting stable and conserved RNA secondary structures in multiple sequence alignments.

SYNOPSIS

```
RNAz [options] [file]
```

OPTIONS

-f, –forward

-r, –reverse

-b, –both-strands

Scores the forward direction, the reverse complement or both. By default only the forward direction as given in the input alignment is scored.

-o NAME, –outfile=NAME

Stores the output in a file. By default the output is printed to the standard output.

-w START-STOP, –window=START-STOP

Scores a sub-region of the alignment starting with column START up to column STOP.

-p X, –cutoff=X

Only show results with RNA class probability P $>$ X. (Default: 0.5)

-g, –show-gaps

Print the output including gaps. Useful if the alignment wants to be recovered from the RNAz output. (Default: off)

-s X, –predict-strand=X

Experimental feature which scores both strands and gives you a guess which direction is more likely to encode a structural RNA. (Default: off)

-V, –version

Prints version information and exits.

-h, –help

Prints a brief help message and exits.

DESCRIPTION

RNAz detects stable and conserved RNA secondary structures in multiple sequence alignments. It calculates two independent scores for structural conservation (the structure conservation index SCI) and for thermodynamical stability (the z-score). High structural conservation (high SCI) and thermodynamical stability (negative z-scores) are typical features of functional RNAs (e.g. noncoding RNAs or cis-acting regulatory elements). RNAz uses both scores to classify a given alignment as functional RNA or not. It uses a support vector machine classification procedure which estimates a class-probability which can be used as convenient overall-score.

RNAz reads one or more alignments in CLUSTAL W or MAF format from a file or standard input and prints the results to the standard output.

Please refer to the files README and manual.pdf for full documentation.

AUTHORS

Stefan Washietl <wash@tbi.univie.ac.at>

Ivo Hofacker <ivo@tbi.univie.ac.at>

Kristin Missal <missal@izbi.uni-leipzig.de>

A.2 `rnazWindow.pl`

Slice alignments in overlapping windows and process/filter alignment windows in various ways.

SYNOPSIS

```
rnazWindow.pl [options] [file]
```

OPTIONS

-w, –window

Size of the window (Default: **120**)

-s, –slide

Step size (Default: **40**)

–max-gap=X

Maximum fraction of gaps. If a reference sequence is used (i.e. `--no-reference` is not set), each sequence is compared to the reference sequence and if in the pairwise comparison the fraction of columns with gaps is higher than X the sequence is discarded. If no reference sequence is used, all sequences with a fraction of gaps higher than X are discarded. (Default: **0.25**)

–max-masked=X

Maximum fraction of masked (=lowercase letters) in a sequence. All sequences with a fraction of more than X lowercase letters are discarded. This is usually used for excluding repeat sequences marked by `RepeatMasker` but any other information can be encoded by using lowercase letters. (Default: **0.1**)

–min-id=X

Discard alignment windows with an overall mean pairwise identity smaller than X%. (Default: **50**)

–min-seqs=N

Minimum number of sequences in an alignment. Discard any windows with less than N sequences (Default:**2**).

–max-seqs=N

Maximum number of sequences in an alignment. If the number of sequences in a window is higher than N, a subset of sequences is used with exactly N sequences. The greedy algorithm of the program `rnazSelectSeqs.pl` is used which optimizes for a user specified mean pairwise identity (see `--opt-id`). (Default: **6**)

–min-length=N

Minimum number of columns of an alignment slice. After removing sequences from the alignment, “all-gap” columns are removed. If the resulting alignment has fewer than N columns, the complete alignment is discarded.

–opt-id=X

If the number of sequences has to be reduced (see `--max-seqs`) a subset of sequences is chosen which is optimized for this value of mean pairwise identity. (In percent, default: **80**)

–max-id=X

One sequence from pairs with pairwise identity higher than X % this is removed (default: **99**, i.e. only almost identical sequences are removed) **NOT IMPLEMENTED**

–forward

–reverse

–both-strands

Output forward, reverse complement or both of the sequences in the windows. Please note: `RNAz` has the same options, so if you use `rnazWindow.pl` for an RNAz screen, we recommend to set the option directly in `RNAz` and leave the default here. (Default: **–forward**)

–no-reference

By default the first sequence is interpreted as reference sequence. This means, for example, that if the reference sequence is removed during filtering steps the complete alignment is discarded. Also, if there are too many sequences in the alignment, the reference sequence is never removed when choosing an appropriate subset. Having a reference sequence is crucial if you are doing screens of genomic regions. For some other applications it might not be necessary and in such cases you can change the default behaviour by setting this option.

-v, –version

Prints version information and exits.

-h, –help

Prints a short help message and exits.

–man

Prints a detailed manual page and exits.

DESCRIPTION

In many cases it is necessary to slice, pre-process and filter alignments to get the optimal input for RNAz. This can be a tedious task if you have a large number of alignments to analyze. This program performs the most common pre-processing and filtering steps.

Basically it slices the input alignments (`CLUSTAL W` or `MAF` format) in overlapping windows. The resulting alignments windows are further processed and only “reasonable” alignment windows are finally printed out, i.e. not too much gaps/repeats, not too few or too many sequences...

EXAMPLES

```
# rnazWindow.pl --min-seqs=4 some.aln
```

Slices the alignment -`some.aln` in overlapping windows of size 120, slide 40 and filters the windows for an optimal input to RNAz (=default behaviour). Only alignments with at least four sequences are printed.

AUTHORS

Stefan Washietl <wash@tbi.univie.ac.at>

A.3 `rnazCluster.pl`

Cluster RNAz hits and print a summary of the results.

SYNOPSIS

```
rnazCluster.pl [options] [file]
```

OPTIONS

-c X, –cutoff=X

Only consider hits with RNAz class probablility P>X (Default:**0.5**)

-w, –windows

-l, –loci

Set these flags to print information for “windows” and/or “loci” in the output. By default, both single windows and combined loci are printed.

-d, –header

Print a header explaining the fields of the output (see below for a detailed description of the fields).

–html

Generates HTML formatted output of the results in the subdirectory `results`. For this option to work you need to have installed ghostscript and a few programs from the ViennaRNA package. More precisely you need the following executables in your PATH: `gs`, `RNAalifold`, `colorrna.pl`, `coloraln.pl`. Alternatively you can adjust the locations of these programs directly in the `rnazCluster.pl` script. Please note that if you use this option the program will get **very slow** because the figures have to be generated. It is also important that you have run RNAz with the **`--show-gaps`** option!

-v, –version

Prints version information and exits.

-h, –help

Prints a short help message and exits.

–man

Prints a detailed manual page and exits.

DESCRIPTION

`rnazCluster.pl` reads RNAz output files and combines hits in overlapping windows to "loci". It prints a summary of the windows and/or loci as a tabulator delimited text to the standard output. An explanation of the fields can be found below. See the user manual for a more detailed meaning of these values.

To work properly, your RNAz output file needs to contain position information. This means there must have been genomic locations in your original alignments you scored with RNAz (i.e. MAF files with a reference sequence). Moreover, the original input alignments have to be **ordered by the genomic location of the reference sequence**.

If you want HTML output please see the notes for the `--html` option above.

FIELDS

"Window" lines

1. **windowID**

   Consecutive numbered ID for each window

2. **locusID**

   The locus which this window belongs to

3. **sequenceID**

   Identifier of the sequence (e.g. human.chr1 or contig42)

4. **start**

   Start position of the reference sequence in the window

5. **end**

   End position of the reference sequence in the window

6. **strand**

   Indicates if the reference sequence is from the positive or negative strand

7. **N**

   Number of sequences in the alignment

8. **columns**

   Number of columns in the alignment

9. **identity**

   Mean pairwise identity of the alignment

10. **meanMFE**

    Mean minimum free energy of the single sequences as calculated by the RNAfold algorithm

11. **consensusMFE**

    “consensus MFE” for the alignment as calculated by the RNAalifold algorithm

12. **energyTerm**

    Contribution to the consensus MFE which comes from the energy part of the RNAalifold algorithm

13. **covarianceTerm**

    Contribution to the consensus MFE which comes from the covariance part of the RNAalifold algorithm

14. **combPerPair**

    Number of different base combinations per predicted pair in the consensus seconary structure

15. **z**

    Mean z-score of the sequences in the alignment

16. **SCI**

    Structure conservation index for the alignment

17. **decValue**

    Support vector machine decision value

18. **P**

    RNA class probability as calculated by the SVM

”Loci” lines

1. **locusID**

   Consecutive numbered ID for each locus

2. **sequenceID**

   Identifier of the sequence (e.g. human.chr1 or contig42)

3. **start**

   Start position of the reference sequence in the window

4. **end**

   End position of the reference sequence in the window

5. **strand**

   Indicates if the reference sequence is from the positive or negative strand

6. **maxN**

   Maximum number of sequences in the alignments of this locus

7. **maxIdentity**

   Maximum mean pairwise indentity in the alignments of this locus

8. **maxP**

   Maximum RNA class probability in the alignments of this locus

9. **minZ**

   Minimum z-score in the alignments of this locus.

EXAMPLES

```
# rnazCluster.pl rnaz.out
```

Parses and clusters the hits in the file `rnaz.out` and prints loci and cluster information to the standard output.

```
# rnazCluster.pl -c 0.9 --html rnaz.out > results90.out
```

Clusters all hits from the file `rnaz.out` with P>0.9, writes the tab-delimited output to the file `results90.out` and, at the same time, generates a website in a subdirectory called `results`.

AUTHORS

Stefan Washietl <wash@tbi.univie.ac.at>

A.4 rnazSelectSeqs.pl

Select subsets of sequences from an alignment.

SYNOPSIS

```
rnazSelectSeqs.pl [options] [file]
```

OPTIONS

-n N, –num-seqs=N

Number of sequences in the output alignment(s). (Default:**6**)

-a N, –num-samples=N

Number of output alignments (Default: **1**)

-i X, –opt-id=X

The resulting alignment(s) is (are) optimized for this value of mean pairwise identity (in percent, default: **80**)

–max-id=X

Sequences from pairs with pairwise identity higher than X% are removed (default: **99**, i.e. only almost identical sequences are removed)

-x, –no-reference

By default the first sequence (=reference sequence) is always present in the output alignment(s). If you do not care having it removed, set this flag.

-v, –version

Prints version information and exits.

-h, –help

Prints a short help message and exits.

–man

Prints a detailed manual page and exits.

DESCRIPTION

`rnazSelectSeqs.pl` reads a multiple sequence alignment in `Clustal W` or `MAF` format and returns an alignment in the same format with a user specified number of sequences. The subset is greedily optimized for a user specified mean pairwise identity. There are options to removes sequences which are too similar. It is also possible to sample more than one alignment. The program uses a simple heuristic to accomplish that.

EXAMPLES

```
# rnazSelectSeqs.pl -n 4 -a 3 miRNA.maf
```

Samples three subsets of four sequences from the alignment `miRNA.maf`.

```
# rnazSelectSeqs.pl -n 5 -i 70 miRNA.maf
```

Selects a subset of five sequences optimized to a mean pairwise identity of 70%.

AUTHORS

Stefan Washietl <wash@tbi.univie.ac.at>

Ivo Hofacker <ivo@tbi.univie.ac.at>

A.5 rnazFilter.pl

Filter output files from `rnazCluster.pl` by different criteria.

SYNOPSIS

```
rnazFilter.pl [options] "filter" [file]
```

OPTIONS

-c, –count

Count the windows/loci instead of printing them.

-v, –version

Prints version information and exits.

-h, –help

Print a short help message and exits.

–man

Prints a detailed manual page and exits.

DESCRIPTION

`rnazFilter.pl` reads tab-delimited data files as generated by `rnazCluster.pl`. For each window a filter is applied and if the filter is passed the window and the corresponding locus are printed out. Thus, you get all loci with at least one window that fulfills your filter criteria.

The mandatory filter statement is given within double quotes (" ") and can contain comparison/logical statements and field identifiers as listed below.

Technically, the statement is directly interpreted by Perl, so you can use anything which works in Perl. The same caveats apply, for example: If you want compare numbers you must use `==`, if you compare strings you have to use `eq`.

Please note: *everything* you put in the filter statement is evaluated by Perl. This can be potentially harmful, so take care.

FIELDS

1. **windowID**

Consecutive numbered ID for each window

2. **locusID**

   The locus which this window belongs to

3. **seqID**

   Identifier of the sequence (e.g. human.chr1 or contig42)

4. **start**

   Start position of the reference sequence in the window

5. **end**

   End position of the reference sequence in the window

6. **strand**

   Indicates if the reference sequence is from the positive or negative strand

7. **N**

   Number of sequences in the alignment

8. **columns**

   Number of columns in the alignment

9. **identity**

   Mean pairwise identity of the alignment

10. **meanMFE**

    Mean minimum free energy of the single sequences as calculated by the RNAfold algorithm

11. **consensusMFE**

    “Consensus MFE” for the alignment as calculated by the RNAalifold algorithm

12. **energyTerm**

    Contribution to the consensus MFE which comes from the energy part of the RNAalifold algorithm

13. **covarianceTerm**

    Contribution to the consensus MFE which comes from the covariance part of the RNAalifold algorithm

14. **combPerPair**

    Number of different base combinations per predicted pair in the consensus seconary structure

15. **z**

    Mean z-score of the sequences in the alignment

16. **SCI**

    Structure conservation index for the alignment

17. **decValue**

    Support vector machine decision value

18. **P**

    RNA class probability as calculated by the SVM

OPERATORS

<,>

Less than, greater than

==

Equals numerically

eq

Equals (strings)

=~/regex/

Matches regular expression.

(,), and, or, not

Logical operators and grouping

EXAMPLES

```
# rnazFilter.pl "P>0.9 and z<-3 and seqID~=/chr13/" results.dat
```

Gives you all clusters with windows with P>0.9 and z<-3 on chromosome 13.

```
# rnazFilter.pl -c "P>0.9" results.dat
```

Counts all windows/loci with P>0.9.

AUTHOR

Stefan Washietl <wash@tbi.univie.ac.at>

A.6 `rnazSort.pl`

Sorts output files from `rnazCluster.pl` by different criteria

SYNOPSIS

```
rnazSort.pl [options] key [file]
```

OPTIONS

-r, –reverse

Sort in reverse order.

–no-loci

Do not preserve the locus grouping but simply sort the windows.

-v, –version

Prints version information and exits.

-h, –help

Prints a short help message and exits.

–man

Prints a detailed manual page and exits.

DESCRIPTION

`rnazSort.pl` reads tab-delimited data files as generated by `rnazCluster.pl`. The files are sorted according to a key which is given at the command line as a mandatory argument. See below for a list of possible keys. By default “better” hits are listed first (e.g. lower z-score or higher P). This can be changed by using the `--reverse` option. By default, the grouping in loci is preserved during sorting. For example if you sort by z-score, you get first the locus first whicht contains the window with the lowest z-score. If you simply want all windows sorted without considering the grouping use the `--no-loci` option.

FIELDS

1. **windowID**

   Consecutive numbered ID for each window. BUG: currently window10 comes before window9 because it is sorted alphabetically.

2. **locusID**

   The locus which this window belongs to. BUG: currently locus10 comes before locus9 because it is sorted alphabetically.

3. **seqID**

   Identifier of the sequence (e.g. human.chr1 or contig42)

4. **start**

   Start position of the reference sequence in the window

5. **end**

   End position of the reference sequence in the window

6. **strand**

   Indicates if the reference seqeunce is from the positive or negative strand

7. **N**

   Number of sequences in the alignment

8. **columns**

   Number of columns in the alignment

9. **identity**

   Mean pairwise identity of the alignment

10. **meanMFE**

    Mean minimum free energy of the single sequences as calculated by the RNAfold algorithm

11. **consensusMFE**

    “Consensus MFE” for the alignment as calculated by RNAalifold algorithm

12. **energyTerm**

    Contribution to the consensus MFE which comes from the energy part of the RNAalifold algorithm

13. **covarianceTerm**

    Contribution to the consensus MFE which comes from the covariance part of the RNAalifold algorithm

14. **combPerPair**

    Number of different base combinations per predicted pair in the consensus seconary structure

15. **z**

    Mean z-score of the sequences in the alignment

16. **SCI**

    Structure conservation index for the alignment

17. **decValue**

    Support vector machine decision value

18. **P**

    RNA class probability as calculated by the SVM

EXAMPLES

```
# rnazSort.pl combPerPair results.dat
```

Sort by “combinations per pair” value, i.e. gives you the hits with the most compensatory mutations.

AUTHOR

Stefan Washietl <wash@tbi.univie.ac.at>

A.7 `rnazAnnotate.pl`

Compare tab-delimited data file as generated by `rnazCluster` to a BED annotation file.

SYNOPSIS

```
rnazAnnotate.pl [options] [file]
```

OPTIONS

-b, –bed

Set the annotation BED file with this option.

DESCRIPTION

This simple programs reads a tab-delimited data file as generated by `rnazCluster.pl`. It compares the genomic region of each predicted locus to the annotations of a BED file. If there is some overlap, the description field of the annotation line in the BED file is added in double quotes as the last field to the locus line.

EXAMPLES

```
# rnazAnnotate.pl -b annotation.bed results.dat
```

Annotates the loci in `results.dat` with annotations in `annotation.bed`.

AUTHORS

Stefan Washietl <wash@tbi.univie.ac.at>

A.8 rnazBlast.pl

Compares predicted loci from data files as generated by `rnazCluster.pl` to a sequence database using BLAST.

SYNOPSIS

```
rnazBlast.pl [options] [file]
```

OPTIONS

-b name, –blast-dir=name

The directory with your BLAST database. If not set, the value from the `BLASTDB` environment variable is used.

-d name, –database=name

Name of the BLAST database to compare with. Must exist in the directory set with `--blast-dir` or in the directory set by `BLASTDB`.

-s name, –seq-dir=name

Directory with sequence files. For each sequence identifier in your input file you need to have a corresponding FASTA formatted file. The files should be named with the sequence identifier and the extension `.fa` or `.fasta`. If your identifier in your input file is for example `contig100` then you should have a file named `contig100.fa`. (If your identifier is of the form "assembly.chromosome" as for example used by UCSC alignments, it is also possible to name the file `chr22.fa` for a sequence identifier `hg17.chr22`).

-e X, –e-value=X

E-value cutoff. All hits with E $<$ X are reported. (Default: 1e-06)

-v, –version

Prints version information and exits.

-h –help

Prints a brief help message and exits.

–man

Prints the manual page and exits.

DESCRIPTION

`rnazBlast.pl` is a simple program to compare your hits to a sequence database using BLAST. To use it you need (i) a sequence database (ii) the sequence files to which the coordinates in your results file refer (iii) a NCBI BLAST installation, i.e. a `blastall` executable somewhere.

First you have to create a BLAST index file for your sequence database. You should have a FASTA formatted file of your database. Assume for example that the file `rfam` contains all sequences of the Rfam database. Run the following command

```
# formatdb -t rfam -i rfam -p F
```

Make sure that you have the sequence files available and named correctly (see notes for the `--seq-dir` option). In this example we assume that the files are in the subdirectory `seq`

You can run the following command to compare each locus in the file `results.dat` with the newly created `rfam` database (which is in the subdirectory `rfam`):

```
# rnazBlast.pl --database=rfam --seq-dir=seq \
               --blast-dir=rfam --e-value=1e-06 \
                 results.dat > annotated.dat
```

If there is a hit better than E=1e-06 the name of the matching sequence and the E-value is added in double quotes as additional field to the locus line.

AUTHORS

Stefan Washietl <wash@tbi.univie.ac.at>

A.9 rnazIndex.pl

Convert data files as generated by `rnazCluster.pl` to different formats.

SYNOPSIS

```
rnazIndex.pl [options] [file]
```

OPTIONS

-g, –gff

Generate GFF formatted output.

-b, –bed

Generate BED formatted output.

–ucsc

In UCSC MAF alignment files it is common to use sequence identifiers like for example “hg17.chr22”. However, in BED are usually specific for a given assembly and therefore only “chr22” is used in the BED files. With this option you change any identifier of the form “X.Y” into “Y”. Moreover, the scores are multiplied by 1000 and rounded to integers since the UCSC genome browser expects scores between 0 and 1000.

-l, –loci

Use the locus information to generate the lines for the GFF and BED files. This is the default.

-w, –windows

Print the ”windows” and not the ”loci”. Probably, rarely used function.

–html

With this option you get a HTML table which links to the the HTML pages which you can create by using the `--html` option in `rnazCluster.pl`. Redirect the output to some file which resides in the `results` directory created by `rnazCluster.pl` and open the file with your favourite web-browser.

-h, –help

Prints a short help message and exits.

–man

Prints a detailed manual page and exits.

DESCRIPTION

`rnazIndex.pl` reads tab-delimited data files as generated by `rnazCluster.pl` and converts them to GFF, BED or HTML formatted files.

GFF is the most widely used annotation file format and supported by many programs and systems (http://www.sanger.ac.uk/Software/formats/GFF).

BED is the native annotation file format used by the UCSC genome browser (http://genome.ucsc.edu).

EXAMPLES

```
# rnazIndex.pl --gff results.dat > results.gff
```

Converts the `results.dat` file to GFF format.

```
# rnazIndex.pl --ucsc --bed results.dat > results.bed
```

Create UCSC style BED format.

```
# rnazIndex.pl --html results.dat > results/index.html
```

Generates HTML formatted table.

AUTHOR

Stefan Washietl <wash@tbi.univie.ac.at>

A.10 rnazBEDsort.pl

Sorts a BED annotation file.

SYNOPSIS

```
rnazBEDsort.pl [options] [file]
```

OPTIONS

-v, –version

Prints version information and exits.

-h, –help

Prints a short help message and exits.

–man

Prints a detailed manual page and exits.

DESCRIPTION

`rnazBEDsort.pl` reads a BED formatted annotation file and sorts the lines by sequence identifier and genomic location. Note: this simple script is not very memory efficient so you could run into problems if you try to sort really large BEDs.

EXAMPLES

```
# rnazBEDsort.pl some.bed
```

Sorts the file `some.bed` and prints the results to standard out.

AUTHORS

Stefan Washietl <wash@tbi.univie.ac.at>

A.11 rnazBEDstats.pl

Reports some statistics on a BED annotation file.

SYNOPSIS

```
rnazBEDstats.pl [options] [file]
```

OPTIONS

-v, –version

Prints version information and exits.

-h, –help

Prints a short help message and exits.

–man

Prints a detailed manual page and exits.

DESCRIPTION

`rnazBEDstats.pl` reads a BED formatted annotation file and prints some basic statistics. It counts the single annotations (“items”) but also the bases covered by these items. “Item bases” means the number of bases that are covered by the items (overlapping regions are not counted). “Item total” is simply the sum of all items (overlapping regions are counted). Important: The BED file **must be sorted** for this program to work. You can use `rnazBEDsort.pl` for this task.

EXAMPLES

```
# rnazBEDstats.pl some.bed
```

Sorts the file `some.bed` and prints statistics for it.

AUTHORS

Stefan Washietl <wash@tbi.univie.ac.at>

A.12 `rnazMAF2BED.pl`

Convert sequence information from MAF formatted multiple sequence alignment to a BED style annotation format.

SYNOPSIS

```
rnazMAF2BED.pl [options] [file]
```

OPTIONS

-s, –seq-id

Specify the sequence identifier of the sequence which should be used as a reference to create the output. Use for example `hg17` if you want to get all sequences containing `hg17` in the idenitfier (e.g. `hg17.chr10`, `hg17.chr22`,...). This option is mandatory.

-v, –version

Prints version information and exits.

-h, –help

Prints a short help message and exits.

–man

Prints a detailed manual page and exits.

DESCRIPTION

This simple programs extracts the position information for a given sequence out of a MAF alignment and outputs it in a BED style annotation format.

EXAMPLES

```
# rnazMAF2BED.pl -s hg17 some.maf
```

Get the regions of the hg17 sequences in the alignment `some.maf`.

AUTHORS

Stefan Washietl <wash@tbi.univie.ac.at>

A.13 rnazRandomizeAln.pl

Randomize alignments by shuffling the columns.

SYNOPSIS

```
rnazRandomizeAln.pl [options] [file]
```

OPTIONS

-w N, –window=N

-s N, –slide=N

Long alignment blocks should be shuffled locally in order to maintain local characteristics of the alignment. Therefore alignments can be shuffled in windows. You can specify here the size of a window and the offset. Defaults are window=120 and slide=120, i.e. the alignments are shuffled in non-overlapping windows of 120 columns.

-l N, –level=N

The shuffling algorithm tries to mantain local conservation patterns, i.e. it shuffles only columns of the same degree of conservation. This becomes limiting if you have many sequences in your alignment. Therfore you can choose the level of “coarse graining” with this option.

To decide which columns have the same degree of conservation, the mean pairwise identity (MPI) of each column is calculated and finally only columns of the same value are shuffled. You can adjust the rounding of the MPI and thus the “coarse graining” level with this option. If you have two columns with say 0.52 and 0.48 MPI you get:

level 0: 1 and 0

level 1: 50 and 50

level 2: 52 and 48

So on level 0 you only have “conserved” (MPI $>$ 0.5) and “non-conserved” (MPI $<$ 0.5) columns while on level 2 you need almost exactly the same MPI to shuffle two columns.

Default value is 2.

-v, –version

Prints version information and exits.

-h –help

Prints a brief help message and exits.

–man

Prints the manual page and exits.

DESCRIPTION

`rnazRandomizeAln.pl` reads a multiple sequence alignment in Clustal W or MAF format and returns a randomized version in the same format. The program uses the algorithm described in Washietl & Hofacker, J. Mol. Biol. 342(1):19 (2004). It generates alignments of the same length, the same base composition, the same gap pattern, the same overall conservation and the same local conservation patterns (see also option **–level**).

EXAMPLES

```
# rnazRandomizeAln.pl -l 1 some.maf > random.maf
```

Randomizes the file `some.maf` using a less stringent parameter for maintaining conservation patterns.

AUTHORS

Stefan Washietl <wash@tbi.univie.ac.at>