Chapter 4

Profiling of RNA Structure at Single-Nucleotide Resolution Using nextPARS

Uciel Chorostecki, Jesse R. Willis, Ester Saus, and Toni Gabaldon

Abstract

RNA molecules play important roles in almost every cellular process, and their functions are mediated by their sequence and structure. Determining the secondary structure of RNAs is central to understanding RNA function and evolution. RNA structure probing techniques coupled to high-throughput sequencing allow determining structural features of RNA molecules at transcriptome-wide scales. Our group recently developed a novel Illumina-based implementation of in vitro parallel probing of RNA structures called nextPARS.

Here, we describe a protocol for the computation of the nextPARS scores and their use to obtain the structural profile (single- or double-stranded state) of an RNA sequence at single-nucleotide resolution.

Key words RNA secondary structure, Genome-wide enzymatic probing, RNA structurome, RNA folding

1 Introduction

Knowledge of the secondary structure of RNAs both in vivo and in vitro is crucial for understanding the regulatory roles that RNAs exert in most cell functions, via characterizing their intramolecular interactions, and how they can change depending on external conditions, including interactions with other molecules [1]. Among several approaches that have been described during the last years to interrogate RNA structure using high throughput sequencing technologies, nextPARS [2] is an enzymatic-based technique that allows probing the secondary structure of RNAs in vitro at a genome-wide scale. It is an adaptation of the previously developed PARS [3] strategy to the Illumina sequencing platform, which allows for sample multiplexing and higher throughput than the original technique.

The original version of this chapter was revised. The correction to this chapter is available at https://doi.org/10.1007/978-1-0716-1307-8_32

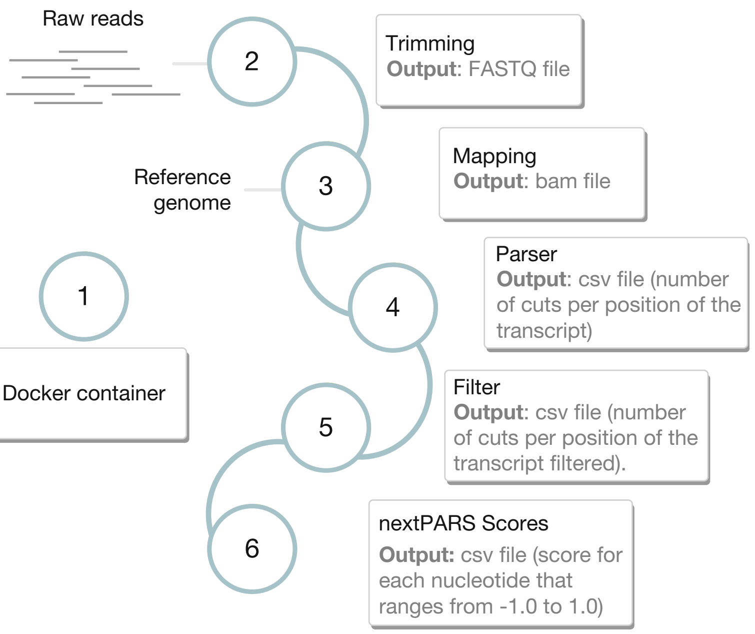
Raw reads
2
Trimming
Output: FASTQ file
Reference genome
3
Mapping
Output: bam file
1
Docker container
4
Parser
Output: csv file (number of cuts per position of the transcript)
5
Filter
Output: csv file (number of cuts per position of the transcript filtered).
6
nextPARS Scores
Output: csv file (score for each nucleotide that ranges from -1.0 to 1.0)

Fig. 1 nextPARS Pipeline. Flowchart showing the steps involved in the nextPARS protocol to go from sequencing reads to structural profiles of RNAs present in a sample

To obtain structural information of RNAs using nextPARS [2], two μg of total or polyA-selected RNA of the species of interest are first denatured, folded, and then enzymatically probed with RNase V1 to cleave double-stranded sites and with S1 nuclease in a separate sample to cut single-stranded conformations. Some additional RNA molecules of interest can be spiked in each of the samples, as possible positive controls with known structure, for example. Digested samples are library prepared using the TruSeq Small RNA Sample Preparation Kit (Illumina), reverse transcribed, PCR-amplified and finally sequenced in multiplex with Illumina platform. The reads obtained are mapped to the reference genome and the cutting points are determined at the 5′end of the reads. Further details on the experimental methodology can be found in the original nextPARS publication [2].

In this chapter, we briefly describe nextPARS methodology and report, using an illustrative example, how the nextPARS raw output data is analyzed in order to go from sequencing reads to structural profiles of RNAs present in a sample (Fig. 1).

2 Materials

2.1 Dataset and Overview

Let us briefly describe the samples and formats for this work (Fig. 1).

– *Raw sequencing data*: we obtained the sequencing data (raw reads) from the SRA database (accession number

PRJNA380612). The raw sequencing data is in standard Sanger FASTQ format. Details on the experimental methodology can be found in nextPARS publication [2].

- *Preprocessing*: we removed adapter sequences from sequencing reads. The input and output data is in FASTQ format.
- *Mapping of Illumina reads and determination of enzymatic cleavage points*. Illumina reads were aligned to the *S. cerevisiae* reference genome and we concatenated the sequences of spike-ins control molecules. We use S288C full chromosomes version R64-2-1, released 18 Nov 2014 (fasta file) and features file from the 16 nuclear chromosomes plus the mitochondrial genome (gff file). The output is in .bam format.
- *Parsing*. For each read alignment, we retrieved the 5′-end position in the reference genome and compared this to the genome annotation. The resulting digestion profile is stored as the number of cuts per position of the transcript. This is stored in csv, comma-separated format (values for each position are separated by semicolons).
- *nextPARS score*: To obtain the scores from nextPARS experiments we used as an input the csv files described above (number of cuts per position of the transcript), and we obtained a new csv file. This file contains a structural profile of an RNA transcript (single- or double-stranded state), with a score for each nucleotide that ranges from −1.0 (highest preference for single strand) to 1.0 (highest preference for double-strand).

2.2 Computational Hardware

We tested the pipeline on a computer with Intel(R) Xeon(R) CPU 3.30GHz, with four cores and 32 GB of RAM. We use Linux (Debian) on x86_64 Architecture. The size of the raw data, intermediate files and final results are about 30 GB for this example. Therefore, 100 GB of hard disk space is required for computing the example data.

2.3 Computational Software

Reproducibility facilitates peer review and ensures that the model, application, and analysis you build can run without worrying about details and software installation. Also, it facilitates collaboration and sharing. Therefore, we built a Docker container based on Ubuntu 18 for nextPARS software**.** Docker is a tool designed to make it easier to create and run applications by using containers. Containers allow developers to package up an application without the need to make custom builds for different environments. As a result, the application will run on any other Linux machine regardless of any custom settings the machine may have that could differ from the machine used to write and test the code.

This protocol assumes users have a Unix-like operating system (i.e., Linux or macOS X), with a bash shell or similar, thus all commands have to be run in a terminal. Although it is possible to follow this protocol with a Microsoft Windows machine (e.g., using the Unix-like Cygwin), the additional steps required are not discussed here.

3 Methods

To illustrate this example, we will use six samples from *S. cerevisiae* and additional RNA molecules from V1 and S1 nextPARS experiment. The data can be downloaded from the publicly SRA database (accession number PRJNA380612) (*see* **Note 1**). Most of the examples are shown using one of the samples (1_S1), so you have to repeat each step on the other five samples. In step 7, the scripts were adapted to process all samples together.

In order to go from the fastq outputs of the nextPARS experiments to a format that allows us to calculate scores, first map the reads in the fastq files to a reference genome using the program of your choice [4]. If you already did that you can omit steps 2 and 3. But, If you want to download the data from SRA with SRA toolkit, trim the sequences with cutadapt and map to the genome with STAR (see **Note 2** and Subheadings 3.2 and 3.3).

This table shows the correlation of each sample run from the SRA project and the sample name (V1's and S1's experiments).

```
# RUN        SAMPLE_NAME
# SRR5422921 1_V1.fastq
# SRR5422920 2_V1.fastq
# SRR5422919 3_V1.fastq
# SRR5422926 1_S1.fastq
# SRR5422925 2_S1.fastq
# SRR5422924 3_S1.fastq
```

You can download the source code and the sample data from GitHub (https://github.com/Gabaldonlab/nextPARS_docker) and the docker container from Docker Hub (https://hub.docker.com/nextpars).

3.1 Setup the Pipeline: TIMING <5 min

3.1.1 Clone the Git Repository

```
git clone https://github.com/Gabaldonlab/nextPARS_docker.git
```

Download the last image version from Docker Hub.

```
docker pull cgenomics/nextpars
```

3.1.2 Download and Run Docker Container

Start a docker interactive session using the `nextpars` container.

```
 docker run -it -v /path/to/nextpars/github/:/home/nextPARS
cgenomics/nextpars:latest bash
```

The `/path/to/nextpars/github` refers to the path from your local machine where you have the nextPARS directory (clone from github). Warning: Anything you do within the docker will be reflected in the data folder!

Now you are working inside the docker container. First, cd into `bin/scripts` directory and execute the configure bash script.

```
cd /home/nextPARS/bin/scripts
source configure.sh
```

3.1.3 Build Genome Indexes (Optional)

Step 1.3, 2, and 3 are optional. If you want to trim and map the raw reads with different software than cutadapt [5] and STAR [6], proceed directly to Step 4 (Fig. 1).

STAR requires a reference genome index for mapping. We use *S. cerevisiae* S288C as the reference genome that was downloaded from Saccharomyces Genome Database (SGD) [7]. We build the index by using the following command.

```
 cd $DATAPATH/DB/saccharomyces_cerevisiae
 STAR --runThreadN 4 \
 --runMode genomeGenerate \
 --genomeDir . \
 --genomeFastaFiles saccharomyces_cerevisiae.fasta \
 --sjdbGTFfile saccharomyces_cerevisiae.gff \
 --sjdbGTFfeatureExon gene \
 --sjdbGTFtagExonParentTranscript Parent
  # --runThreadN Number of threads to be used for genome
generation
 # --runMode genomeGenerate option directs STAR to run genome
indices generation job
 # --genomeDir specifies the path to the directory where the
genome indices are stored
 # genomeFastaFiles specified one or more FASTA files with the
genome reference sequences
 # specifies the path to the file with annotated transcripts in
the standard GTF format
 # --sjdbGTFfeatureExon tag name to be used as exons' parents
for building transcripts
 # --sjdbOverhang specifies the length of the genomic sequence
around the annotated junction to be used in constructing the
splice junctions database.
```

3.2 Trimming Sequence Adapters: TIMING <3 min per sample (Optional)

Raw sequencing reads likely to contain parts of the adapter sequence. Therefore, these sequences must be identified and trimmed. These adapters can be removed using a specialized adapter removal tool and there is a more than large choice for the appropriate published adapter trimming tools. Here, we use cutadapt for this purpose. In `fastq` directory you should have the 6 samples from *S. cerevisiae* nextPARS experiments.

```
 cd $DATAPATH
 cutadapt -a TGGAATTCTCGGGTGCCAAGGAACTCCAGTCAC -m 18 -j 4 -o
$DATAPATH/trimming/1_S1.fastq.qz $DATAPATH/fastq/1_S1.fastq
 # -a ADAPTER Regular 3′ adapter
 # -m LENGTH Discard processed reads that are shorter than
LENGTH.
 # -j N, where N is the number of cores to use.
 # -o output file (output file formats are FASTA and FASTQ,
with optional compression and the output file format is
recognized from the file name extension).
```

3.3 Aligning the Reads Against the Genome using STAR: TIMING <5 min per Sample (Optional)

Mapping reads to the genome: TIMING <5 min per sample

The trimmed reads should be aligned to the reference genome. Mapping results are in BAM format.

```
 cd $DATAPATH
 STAR --runThreadN 4 \
 --genomeDir $DATAPATH/DB/saccharomyces_cerevisiae/ \
 --readFilesIn $DATAPATH/trimming/1_S1.fastq.qz \
 --outFileNamePrefix $DATAPATH/mapping/1_S1.fastq \
 --outSAMtype BAM SortedByCoordinate \
 --outTmpDir $TMP_DIR/1_S1.fastq
 # --runThreadN Number of threads to use for mapping
 # --genomeDir path of the STAR index
 # ---readFilesIn name(s) (with path) of the files containing
the sequences to be mapped
 # --outFileNamePrefix Output files name prefix (full or
relative path).
 # --outSAMtype type of SAM/BAM output
 # --readFilesCommand Command line to execute for each of the
input files.
 # --outTmpDir path to a directory that will be used as
temporary by STAR.
```

Once you obtain the bam file, use nextPARSParser.py to count the number of reads at each position (which indicates a cut site for the enzyme in the file name).

3.4 Parser: Number of Reads Beginning at Each Position: TIMING <8 min per Sample

If you skipped steps 1.3, 2 and 3, we assume that you have the bam files in the `data/mapping` directory. As an alternative, you can use already generated bam files (subsets from the originals), that are in `data/mapping_subset` directory. To do so, you have to change `-b` argument by modifying `mapping` by `mapping_subset` on the following command.

```
 cd $BINPATH

 python nextPARSParser.py \
 -b $DATAPATH/mapping/1_S1.fastqAligned.sortedByCoord.out.bam
\
 -g $DATAPATH/DB/saccharomyces_cerevisiae/saccharomyces_cere-
visiae.gff \
 -o $DATAPATH/tab/1_S1.tab \
 -t gene

 # -b Path to the SAM/BAM file containing the mapped reads
 # -g Path to the GTF file containing the features
 # -o The name given to the output file in csv format (.tab
extension)
 # -t Feature type (3rd column in GTF file) to be used
(default, suitable for Ensembl GTF files: exon)
 # -a Skip all reads with MAPQ alignment quality lower than the
given minimum value (default: 10). MAPQ is the 5th column of a
SAM/BAM file and its usage depends on the software used to map
the reads.
```

The output of this script is a csv file containing the name of the molecule and the count values (number of inferred enzyme cuts) for each position, separated by semicolons.

3.5 Filtered Out Transcripts with Low Counts: TIMING <1 min per Sample

The reformat_PARSparser_output.py script filters out transcripts with low counts and produces output in csv format.

```
 cd $BINPATH
 python reformat_PARSparser_output.py -t $DATAPATH/tab/1_S1.
tab -m 20

 # -m Min average counts for a given transcript
 # -tab csv file to be reformatted
```

3.6 nextPARS Scores: TIMING <1 min per Sample

To obtain the final scores from nextPARS experiments (from .tab files), use the following command. For details on how to calculate the scores, *see* **Note 3**.

```
cd $BINPATH
python get_combined_score.py \
 -i U1 \
 -inDir ../data/tab \
 -f ../data/fasta/U1.fa \
 -o ../data/score/U1_score.RNN.tab

 # -i to indicate the molecule for which you want scores (all
available data files will be included in the calculations --
molecule name must match that in the data file names)
 # -inDir to indicate the directory containing the .tab files
with read counts for each V1 and S1 enzyme cuts
 # -f to indicate the path to the fasta file for the input
molecule
 # -s to produce an output Structure Preference Profile (SPP)
file. Values for each position are separated by semi-colons.
Here 0 = paired position, 1 = unpaired position, and NA =
position with a score too low to determine its configuration.
 # -o to output the calculated scores, again with values for
each position separated by semi-colons.
 # --nP_only to output the calculated nextPARS scores before
incorporating the RNN classifier, again with values for each
position separated by semi-colons.
 # {-V nextPARS} to produce an output with the scores that is
compatible with the structure visualization program VARNA1
 # {-V spp} to produce an output with the SPP values that is
compatible with VARNA.
 # -t to change the threshold value for scores when determining
SPP values [default = 0.8, or -0.8 for negative scores]
 # -c to change the percentile cap for raw values at the
beginning of calculations [default = 95]
 # -v to print some statistics in the case that there is a
reference CT file available. If not, will still print nextPARS
scores and info about the enzyme .tab files included in the
calculations.
```

3.7 Automatization of Steps

In order to simplify steps 2–5 (Fig. 1), there are different bash scripts in the bin/scripts directory inside the container. These scripts will help to automate the pipeline and to use a set of V1 and S1 samples. We assume that you have the .fastq files in the `data/fastq` directory.

```
cd ~/bin/scripts
```

Trimming

```
./trimming.sh
```

Mapping reads. Use the following script to do the mapping. First, you have to generate the genome index (step 1.3).

```
./mapping.sh
```

If you want to start with the Parser step (count number of reads beginning at each position), we assumed that you have the bam files in the `data/mapping` directory. As an alternative, you can use already generated bam files (subsets from the originals), that are inside the `data/mapping_subset` directory. To do so, you have to modify the FILE variable (changing `mapping` by `mapping_-subset`) on the following script.

Parser

```
./nextPARSParser.sh
```

Filter out transcripts

```
./reformat_PARSParser.sh
```

4 Notes

1. *Download SRA sequences.*

   How to download sequence data files using SRA Toolkit is explained in detail here: https://www.ncbi.nlm.nih.gov/sra/docs/sradownload/.

2. *Trimming the data.*

   If you are not sure if you need to trim your data, fastp [8] is a tool that has implemented methods that automatically detect 5’ or 3’ adapters for both paired and single-end data.

3. *Computation of nextPARS scores.*

   We present here a summary of the main step of the next-PARS methodology, the computation of nextPARS scores from csv (.tab) files (number of reads at each position, which indicates a cut site for the enzyme). This is implemented in get_-combined_score.py script. A more detailed description can be found in [2].

   Phase I: scores from raw experimental data (Sprofile).

   There are five parts to the profile score (S_{profile}) calculation:

   (a) First, the digestion profiles are read from the .tab files. This gives the number of cuts at each position, and it should include one or more .tab file for both the V1 and S1 enzymes.

(b) These raw counts are capped at the maximum percentile of the value. By default, this is 95%, so that any position with more cuts than 95% of the other positions are set to this 95th percentile value. This helps to dampen the skew in cuts due to the few positions that may be preferentially cleaved by either V1 or S1 at rates orders of magnitudes greater than most other positions.

(c) The capped counts from each .tab file are then normalized to its average so that each .tab now has a mean of 1 cut per position. This corrects for (I) the different expression levels for each molecule, (II) different sequencing depth between runs of the nextPARS experiment, and (III) the different rates of cleavage between the V1 and S1 enzymes, the latter of which cuts more frequently. Since the final 50 positions do not have counts in a nextPARS experiment of this example (50 bases long reads were used), these are not included in the normalization.

(d) In the case that multiple replicates of the experiment were performed, a single list of cuts is generated separately for V1 and S1 by taking the average cuts at each position.

(e) Finally, a single combined *S* score is calculated to determine whether a position is likely to be paired or unpaired, using the following steps. (I) The lists of average normalized V1 and S1 cuts are each then normalized to a maximum of 1. (II) S1 values are subtracted from V1 values to know if a position tends to be cut more by one enzyme or the other. (III) The resulting positive values are then normalized to a maximum of +1 while the negative values are normalized to a minimum of -1, such that the range of values is always from -1 to +1. In this way, we have a fixed range to which we can apply threshold values for cuts per position to determine those which can be confidently called paired or unpaired.

Phase II: scores from a recurrent neural network (RNN) classifier (SRNN).

The $S_{profile}$ calculation can then be complemented by recurrent neural network (RNN) classifier score (S_{RNN}) which calculates the probability that a position is paired or unpaired by considering that nucleotide and its neighbors. The model is trained on a database of known RNA secondary structures and is constructed with a long term short memory (LTSM) layer and a dense neural network layer [9, 10]. For details on the training and implementation of the classifier model [2].

Use the RNN classifier separately.

The RNN classifier that already incorporated into the nextPARS scores can run separately, using a different experimental score input (in .tab format), it can be run like so: `python predict2.py -f molecule.fasta -p scoreFile.tab -o output.tab`

Acknowledgments

We thank Manuel Molina for his help with testing the pipeline. This work was supported by a grant from the Marie Skłodowska-Curie Actions [H2020-MSCA-IF-2017-793699] to U.C., and grants from the Spanish Ministry of Economy, Industry, and Competitiveness (MEIC) [PGC2018-099921-B-I00] cofounded by European Regional Development Fund (ERDF); from the CERCA Programme/Generalitat de Catalunya; from the Catalan Research Agency (AGAUR) SGR423, from the European Union's Horizon 2020 research and innovation program under the grant agreement ERC-2016-724173; and from INB (PT17/0009/0023—ISCIII-SGEFI/ERDF) to T.G.

References

1. Wan Y, Kertesz M, Spitale RC, Segal E, Chang HY (2011) Understanding the transcriptome through RNA structure. Nat Rev Genet 12 (9):641–655
2. Saus E, Willis JR, Pryszcz LP, Hafez A, Llorens C, Himmelbauer H, Gabaldón T (2018) nextPARS: parallel probing of RNA structures in Illumina. RNA 24(4):609–619
3. 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
4. Costa-Silva J, Domingues D, Lopes FM (2017) RNA-Seq differential expression analysis: an extended review and a software tool. PLoS One 12(12):e0190152
5. Martin M (2011) Cutadapt removes adapter sequences from high-throughput sequencing reads. EMBnet J 17:1
6. Dobin A, Davis C, Schlesinger F, Drenkow F, Zaleski C, Jha S, Batut S, Chaisson M, Gingeras TR (2013) STAR: ultrafast universal RNA-seq aligner. Bioinformatics 29(1):15–21
7. Cherry JM, Hong EL, Amundsen C, Balakrishnan R, Binkley G, Chan ET, Christie KR, Costanzo MC, Dwight SS, Engel SR, Fisk DG, Hirschman JE, Hitz BC, Karra K, Krieger CJ, Miyasato SR, Nash RS, Park J, Skrzypek MS, Simison M, Weng S, Wong ED (2012) Saccharomyces Genome Database: the genomics resource of budding yeast. Nucleic Acids Res 40(Database issue):D700–D705
8. Chen S, Zhou Y, Chen Y, Gu J (2018) fastp: an ultra-fast all-in-one FASTQ preprocessor. Bioinformatics 34-17:i884–i890
9. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9:1735–1780
10. Sutskever I, Vinyals O, Le Q (2014) NIPS '14: proceedings of the 27th international conference on neural information processing systems, vol 2. Neural Information Processing Systems (NIPS), Montreal, QC