StructureFold3

Tools to facilitate rapid analysis of whole transcriptome chemical probing data

Introduction

StructureFold3 is the Python3 successor to Structurefold2 (Tack et al. (2018)), aiming to continue building functionality and ease of use.

Software Dependencies

- Python 3
- BioPython
- Numpy
- Cutadapt (Martin (2011))
- Bowtie2 (Langmead and Salzberg (2012))
- STAR (Dobin et al. (2013))

Recommended Software

- FastQC
- RNAStructure (Reuter and Mathews (2010))
- R (Team and others (2013))

StructureFold3 is a set of Python3 scripts designed to process Structure-seq (Ding et al. (2015), Ding et al. (2014), Ritchey et al. (2017)) or other high-throughput libraries generated to yield reverse transcription stops via chemically probing RNA structure into reactivity values at the single nucleotide level (structurome).These reactivity values may be used to guide folding with RNAstructure (Reuter and Mathews (2010)) or similar software, greatly enhancing structure prediction with *in-vivo* probing data. This package allows a user with a minimal computationalor statistical background to execute a number of mostly automated processes to assemble *in-vivo* reverse transcriptase stops collected by chemical probing and derive reactivity values indicative of the probability of base pair strandedness, thus generating a reasonable facsimile of the actual *in-vivo* RNA structurome.

Ultimately, the final design and analysis of the data is up to the experimenter; data may be piped into RNAstructure to obtain predicted folds of transcripts, or the experimenter may choose to directly analyze the derived chemical reactivity or raw RT stop values in any way that suits their experimental design with typical statistical packages, such as R (Team and others (2013)). StructureFold3 is amenable to a variety of chemical probes (e.g. DMS, glyoxal, SHAPE), provided a negative reagent control library is included. Structurefold3 is designed to be modularl; steps and programs can be modified to accomodate unforseen experimental, chemical, or biological experimental constraints.

Walkthrough

Before you begin. . .

An accurate StructureFold3 analysis hinges on a prudent selection of the transcriptome of interest. We highly recommend that the user take the time to carefully consider the version, quality of annotation, and size of the transcriptome to be used; the <.fasta> you initially choose that contains this information may not be changed during the course of an analysis. Some Eukaryotes contain an excessive number of transcript isoforms per gene, which may complicate downstream analyses, and it may be wise to prune to no more than two isoforms per gene to map against. Selecting the most abundant isoform based on other techniques or experiments is an alternative approach to reduce the number of included transcripts per gene and enhance the precision of reactivity values and the simplicity of all downstream analyses. Ensembl (yates2020ensembl) is a great place to look for transcriptome (cDNA) files if one has not already been selected to map against, although absolutely any <.fasta> file containing transcripts will work provided the transcripts are of sufficient length.

Ensembl Transcriptome Sources

- Ensembl
- Ensembl Plants
- Ensembl Protists
- Ensembl Bacteria

We recommend mapping to the whole transcripts before partitioning the resulting derived information into transcript features (e.g. 5'UTR, CDS, 3'UTR) in later steps so that reads overlapping two of these features will not encounter trouble mapping. To ensure that transcript annotation is sufficient to later subdivide features for individual analysis, check to ensure the start and stop coordinates of each of these features are included with the annotation or an associated <.gtf> or <.gff> file. At this time we are unable to provide an automated script to handle annotation and automatically perform this subdividison due to differences in annotation standards, but it could be included in a future version.

StructureFold3 is not a computationally demanding package by itself. Bowtie2, the recommended short read aligner, is likewise lightweight and not exceptionally demanding on most hardware, even on larger transcriptomes and data sets. During the course of data processing however, many intermediate files will be generated; thus, having at least ~4 TB of space available is advised to store these files until an analysis is completed. Using RNAStructure to interpret the restraints/constraints generated by StructureFold3 with the included scripts to batch-fold thousands of transcripts is computationally demanding; this can tie up multiple cores for several days. Thus, access to a server or a dedicated personal machine is advised if carrying out this particular task. We recommend running StructureFold3 on either any mainstream Linux system or MACOS. It is possible to run StructureFold3 on a Windows install within an emulated environment, but all features may not work properly and no additional support can be given.

Installation and Support

Everything is hosted on the github repository, including this manual. A typical install would be accomplished via git.

```
git clone https://github.com/StructureFold2/StructureFold3
```

Next, this directory would be added to the path, typically by adding information to your shell profile

```
PATH=$PATH:/home/path/to/new/folder/StructureFold3
```

Lastly, whatever permissions are most applicable must now be set.

```
chmod -R 755 StructureFold3/
```

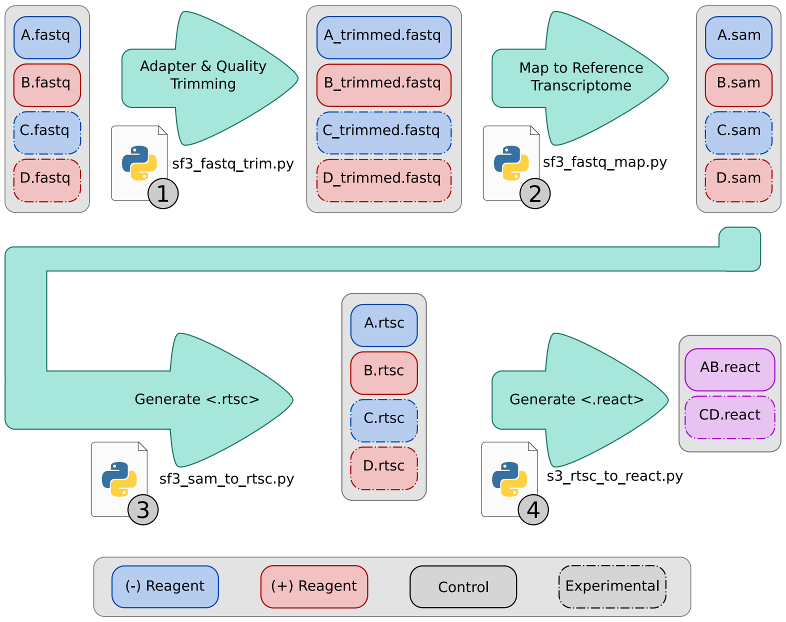
Make sure to check the dependencies (requirements) of StructureFold3 and install these according to any instructions provided on respective websites and documentation. Many of these packages are commonly available from package managers native to either Linux or MacOS systems. For a listing of these dependencies, check the top of page 1. Depending how these are installed, permissions and the path may need to be set up for each dependency. StructureFold3 will automatically call some of these programs. To check that you have successfully installed StructureFold3 and all of the core dependencies, try executing the test script, sf3_test_sf3, in a new shell. The script will notify the user of any missing dependencies that are not available in the path or for which the user does not have adequate permissions to run. If you plan to customize the StructureFold3 pipeline and forgo the use of suggested trimming software, these dependencies do not need to be installed and set up.

This manual makes the best attempt possible to detail the present StructureFold3 tool set and demonstrate its use. However, it is nearly impossible to plan for all contingencies. While the software is provided ‘as is’, we are eager to improve and further streamline our analysis pipeline. If you encounter a bug or require support, or have an idea for a feature that would improve your overall StructureFold3 experience, please do not hesitate to contact via GitHub. At some point in the future, SF3 may be available via PIP, etc.

Segment 1 Overview

Segment 1 will take raw reads and generate reactivity profiles. Reads are trimmed, mapped, then converted into a custom format (<.rtsc>, reverse transcriptase stop counts), before these are used to calculate reactivity (<.react>). Other trimming and/or mapping software can be subsituted readily, as the first two modules are of convenience; they run programs under the reccomended settings and all output is properly named and organized. Any other trimming/mapping protocols can be substituted, provided the end result is in <.sam> format.

Module	Function	In	Out
sf3_fastq_trim	Batch runs read trimming	fastq	fastq
sf3_fastq_map	Batch runs read mapping	fastq	sam
sf3_sam_to_rtsc	Extracts read stops	sam	rtsc
sf3_rtsc_specificity	Calculates Reagent Specificity	rtsc	csv
sf3_rtsc_combine	Combines <.rtsc> together	rtsc	rtsc
sf3_rtsc_coverage	Calculates transcript coverage	rtsc	csv,txt
sf3_rx_correlation.py	Formats stop correlation analysis	rtsc	csv
sf3_rtsc_to_react	Calculates reactivity	rtsc	react



1.1 Trim Reads

Trimming is accomplished via the sf3_fastq_trim module, which drives cutadapt (Martin (2011)). A typical analysis would simply run the module on the directory containing the raw <.fastq> files, generating a new directory with trimmed versions of each <.fastq> alongside a detailed log file generated from the output of cutadapt. output as a text file.

```
sf3_fastq_trim.py -dyr <fastq_directory> -outdir <trimmed_fastqs>
```

The actual run of reverse transcriptase is infered by extacting the insert between the StructureSeq adapters. Inspect the log as a diagnostic. When the 3' sequencing adapter is trimmed, everything downstream of it is also removed. Excessively short read length after 3' adapter trimming may suggest the RNA was low quality or partially degraded before the adapters were ligated. While trimming in general for most RNA based techniques is becoming sort of obselete given options like softcliping when mapping, it retains important for structure probing. Another trimming program/protocol may be subsituted for cutadapt and this step skipped entirely, so long as the user is confident all adapters have been removed, and trimmmed reads are available for use in mapping. This module and protocol are for use on non-paired end reads; this feature will be added once there are paired-end Structure-Seq or other analogous libraries to work with. Theoretically, unless the inserts are prohibitively small one would only need to trim the 5' adapter from the forward read and the 3' adapter from the reverse.

1.2 Map Reads

Mapping is accomplished via the sf3_fastq_map module, which drives Bowtie2 (Langmead and Salzberg (2012)) or STAR((Dobin et al. (2013))). A typical analysis would simply run the module on the directory containing the trimmed <.fastq> files, generating a new directory with a corresponding mapped <.sam> of each and writing a log detailing mapping statistics. Be sure to generate an appropriate index to map against for the program you intend to use, using the transcriptome fasta.

```
sf3_fastq_map.py -dyr <trimmed_fastqs> -outdir <mapped>
```

Be sure to check the mapping statistics are reasonable given your experiment. Excessive multimapping can be addressed either by choosing a truncated transcriptome to map against. Other mapping programs/protocols may be substituted and this step skipped entirely, so long as the user is confident the reads are properly mapped and in <.sam> format.

1.3 Extract Read Stops

Reverse transcriptase stops are extracted from mapped reads via sf3_sam_to_rtsc. This process can be executed on both directories and individual files. The data at this point are greatly condensed, as all read and mapping information other than the nucelotide where RT stopped do not persist onto future steps. Running the module on directory containing mapped <.sam> files and providing the index fasta file are used to generate a directory of corresponding .rtsc.

```
sf3_sam_to_rtsc.py <index> -dyr <sam files> -outdir <rtsc files>
```

Inspecting the log will file will indictate how many reads were incorporated into corresponding .rtsc(s), and how many reads were discrarded, and for what reasons. The current version of the module does not support paired-end reads; this feature will be added once there are paired-end Structure-Seq or other analogous libraries to work with. Theoretically, paired-end reads would have an easier time finding a unique mapping in a transcriptome, which is great, but only the RT stop, infered from the forward read, would persist past this step

1.4 Check Reagent Specificity

The specificity of libraries should be verified to match the predicted pattern of the reagent(s) used in the experiment. This can be done with the sf3_rtsc_specificity This will produce a detailed specificity report on each file analyzed; the process can be executed on both directories and individual files, and requires the index fasta file used to generate the <.rtsc>s.

```
sf3_rtsc_specificity.py <index> -dyr <rtsc files>
```

While there is bound to be species to species differences in patterns, check that the plus reagent libraries show appropriate increases on the nucleotides they interact with compared to untreated libraries.

1.5 Pool Replicates

Replicates of the same condition/sample type can pooled using sf3_rtsc_combine. This sums the stops on each nucleotide for each transcript, increasing depth of data.

```
sf3_rtsc_combine.py <rtsc_files>
```

The variability between component replicates will be examined in a future step.

1.6 Calculate Transcript Coverage

The per-transcript coverage for one or multiple files is calculated via sf3_rtsc_coverage. Typically this is done on pooled files of plus reagent.

```
sf3_rtsc_coverage.py
```

This will yield both a detailed coverage report, and an overlap file listing each transcript that passed coverage threshold in all samples analyzed. This overlap file will be used in later modules to truncate analyses to only those transcripts with coverage at or above this threshold.

1.7 Calculate Stop Correlation

Each reagent treated replicate within each sample should be then checked for repeatability in stop patterns against each other treated replicate of the same sample. This can be done with the sf3_rx_correlation.py module. It is reccomended only to do correlation analyses on replicates of reagent treated samples, for the nucelotides those reagents are specific for, on transcripts where the combined coverage of all replicates exceeded the default threshold of one.

```
sf3_rx_correlation.py <index> <.rtsc files> -restrict <overlapfile>
```

This produces data readily fed into R(Team and others (2013)) or any other statistical software to calculate reactivity correlation. A high correlation suggests good repeatability between replicates.

1.8 Calculate Reactivity

Finally, reactivity is calculated on one or more samples using sf3_rtsc_to_react. Only samples sharing the final 2-8% scale are comparable

```
sf3_rtsc_to_react.py
```

This module completes the first segment of SF3 analysis.

Segment 2 Overview

Segment 2 focuses on direct analysis of transcript reactivity patterns, both within and between conditions. These modules do not utilize folding software, though some may yield subsets of reactivity files which are then used with folding software. Most of the the output of the modules from this segment are in formats easily used with R or other statistical packages.

Module	Function	In	Out
sf3_react_statistics	Summarizes reactivity metrics	react	csv
sf3_rtsc_abundances	Estimates Transcript Abundance	rtsc	csv

2.1 Calculate Basic Metrics

2.2 Calcualte Transcript Abundance

Segment 3 Overview

Segment 3 will focus on Folding and Subsequent Analysis.

Module	Function	In	Out
sf3_batch_fold	Batch runs folding programs	react	ct
sf3_structure_statistics	Summarizes folded transcripts	ct	csv

Modules

sf3_rtsc_combine.py

Empty Placeholder

```
Combines <.rtsc> files, typically replicates of the same sample

optional arguments:
  -h, --help  show this help message and exit

Input:
  rtsc        Input <.rtsc> files

Output:
  -sort       Sort output by transcript name
  -name NAME  Specify output file name
```

sf3_fastq_trim.py

This module drives cutadapt (Martin (2011)) under the reccomended settings. By default, the Structure-Seq2 5' and 3' adapterstrimmed, but alternative adapters can be removed (-tp, -fp). Simularly, the minimum quality and minimum length parameters are set by default, but can be changed (-minlen, -minqual). The maximum length of a read to be accepted can also be set (-maxlen); filtering reads by the maximum length after adapter trimming is a circuitous way to only accept reads where adapters were both found and removed. For example, by setting the maximum allowed read length under the actual lenght of the raw reads, only reads where adapters have been both detected and removed are accepted, ensuring that data going forward was modified per the protcol.

```
Generates <.rtsc> files from <.sam> files

optional arguments:
  -h, --help            show this help message and exit

Input:
  fasta                 Index Fasta File
  -sam SAM [SAM ...]    Input SAM file(s)
  -dyr DYR [DYR ...]    Input Directories

Settings:
  -mismatches <number>  [default = 3] Maximum allowed mismatches/indels
  -firstmm              Accept alignments with first base mismatches
  -reverse              Accept alignments to the reverse strand
  -rm_secondary         Remove secondary alignments

Output:
  -logname LOGNAME      [default = filter_log.csv] Log file name
  -params PARAMS        [default = params_log.csv] Parameters file name
  -outdir PATH          [default = rtsc] Out directory
```

sf3_fastq_map.py

This module does not yet have an SF3 implementation.

sf3_sam_to_rtsc.py

Generates <.rtsc> from <.sam> files.

Usage

```
Converts <.sam> into reverse transcriptase stop files <.rtsc>

optional arguments:
  -h, --help            show this help message and exit

Input:
  fasta                 Index Fasta File
  sam                   Input SAM file(s)

Settings:
  -mismatches <number>  [default = 3] Maximum allowed mismatches/indels
  -firstmm              Accept alignments with first base mismatches
  -reverse              Accept alignments to the reverse strand
  -rm_secondary         Remove secondary alignments

Output:
  -logname LOGNAME      [default = filter_log.csv] Log file name
```

sf3_rtsc_coverage.py

Calculates reagent coverage on one or more <.rtsc> files

Usage

```
Calculates RT stop coverage from <.rtsc> file(s)
```

```
optional arguments:
  -h, --help   show this help message and exit

Input:
  fasta        Reference Fasta
  f            Input <.rtsc> files

Settings:
  -bases ACGT  [default = AC] Coverage Specificity
  -ot OT       [default = 1.0] Overlap file threshold

Output:
  -name NAME   Output file name
  -ol          Create an overlap file
  -on          Overlap file name
```

sf3__fasta__statistics.py

This module takes any fasta and creates a per sequence composition report. It is intended to be used on the transcriptome file used for mapping, thus producing a <.csv> detailing transcript composition. This can be matched to other processed data from the same transcripts to investigate any potential relationships between composition and reactivity.

The module is agnostic to the actual characters used in the sequences so it could be used on a fasta containing amino acids to obtain similar per sequence composition frequencies. By default, the column containing the sequence names will be headed with 'transcript' but this can be specified by the user with the -description flag. By default, it will generate columns for any character in the sequences at or above 1% of the grand total; configurable via the -threshold flag.

Usage

```
Creates a composition report for a <.fasta> file

optional arguments:
  -h, --help           show this help message and exit

Input:
  fasta                Input Fasta File

Settings:
  -threshold <number>  [default = 1.0] Minimum percent to report
  -description <name>  [default = transcript] Sequence column heading

Output:
  -affix AFFIX         [default = composition] <.csv> file affix
```

sf3_react_heat_correct.py

Probing reagents can be more reactive at higher temperatures, thus one may wish to normalize this effect out from any two pairwise <.react>s derived from experiments at different temperatures. For any two such <.react> files generated against the same transcriptome, the sum of all base reactivties will be summed for both files, and used to scale the the lower/higher temperature file up/down respectively, such that the total amount of reactivity on all bases in both files is the same.

Usage

```
Corrects two <.react>s for differential temperature

optional arguments:
  -h, --help      show this help message and exit

Input:
  lower           lower temp <.react> file
  higher          higher temp <.react> file

Output:
  -suffix SUFFIX  [default = corrected] Suffix for out files
```

sf3_react_to_csv.py

For close or custom analysis, researchers may find the <.react> format clunky or just not particularly amenable to individual analysis. This module converts <.react> files into <.csv> files. A single transcript may be extracted to a <.csv> file or the entire file may be batch converted into a directory of <.csv> files. The corresponding <.fasta> used to generate the <.react> files must be provided so nucelotides can be matched to the reactivity values in the output <.csv>.

Usage

```
Reformats reactivity values to <.csv> format

optional arguments:
  -h, --help        show this help message and exit

Input:
  react             File to pull values from
  fasta             File used to generate the <.react>
```

```
Settings:
  -mode {S,M}       Single or Multi-Transcript mode
  -transcript NAME  [S] Specific Transcript to reformat
  -restrict <.txt>  [M] Limit analysis to these specific transcripts
  -outdir OUTDIR    [M] Name of the out directory, overrides default
```

sf3_rtsc_abundances.py

It is possible to infer releative transcript abundance from <.rtsc> files. This module will take one or more <.rtsc> files and report a selected abundance metric (RPKM/TPM) for each transcript based on the total number of mapped RT stops. All <.rtsc> being calculated together should be generated from mapping against the same reference files, and thus should contain the exact same set of transcript entries. However, by default transcripts without an entry in a file will result in an 'NA', unless changed by the user to zeros (-zero).

Usage

```
Determines approximate transcript abundance using <.rtsc> files

optional arguments:
  -h, --help        show this help message and exit

Input:
  rtsc              Input <.rtsc> file(s)

Settings:
  -mode {RPKM,TPM}  Abundance Metric to Calculate
  -zero             Set missing values to zero

Output:
  -name NAME        Specify output file name
```

sf3_rtsc_specificity.py

This module calculates the reverse transcriptase stop specificity using <.rtsc> files and the fasta they were generated against. Hence, comparing treated and untreated samples will detail the

Usage

```
Analyzes native/reagent nucleotide RT stop specificity

optional arguments:
  -h, --help          show this help message and exit

Input:
  index               Fasta used to generate the <.rtsc>
  -rtc RTC [RTC ...]  Individual <.rtsc>
  -dyr DYR [DYR ...]  Input <.rtsc> containting Directories

Settings:
  -report REPORT      Include these nucelotides in report
  -round DIGITS       [default = 5] Decimal places to report

Output:
  -name NAME          Specify output file name
```

sf3_sam_to_rtsc.py

This module filters mapped reads in sam format, extracting the implied reverse transcriptase (RT) stops that pass the default and/or user defined criteria. These stops are then written to a Reverse Transcriptase Stop Count <.rtsc> file to represent these stops in subsequent analysis steps. Longer reads may call for allowing more mismatches, as the window for error is greater with more bases sequenced. Input can be a mixture of individual files and directories, although generally it is best to work in a directory-wise fashion, completing this step for all samples at once, both for consistency and organization.

Usage

```
Generates <.rtsc> files from <.sam> files

optional arguments:
  -h, --help            show this help message and exit

Input:
  fasta                 Index Fasta File
  -sam SAM [SAM ...]    Input SAM file(s)
  -dyr DYR [DYR ...]    Input Directories

Settings:
  -mismatches <number>  [default = 3] Maximum allowed mismatches/indels
  -firstmm              Accept alignments with first base mismatches
  -reverse              Accept alignments to the reverse strand
  -rm_secondary         Remove secondary alignments

Output:
  -logname LOGNAME      [default = filter_log.csv] Log file name
  -outdir PATH          [default = rtsc] Out directory
```

sf3_fasta_stepwise.py

This module is designed to break up large sequences into smaller 'sliding windows', so that small, granular segments can be folded and further studied.

Usage

```
Splits sequences into sliding window segments

optional arguments:
  -h, --help  show this help message and exit

Input:
  fasta       Input Fasta File

Settings:
  -size SIZE  [default = 10] Size of the Windows
  -step STEP  [default = 5] Step of the Windows

Output:
  -name NAME  Specify output file name
  -sort_file  Sort the output by name
```

sf3_batch_fold.py

This module batch runs RNAStructure's Fold or partition on transcript or oligo sets, with the option to use <.react> files as restraints/constraints. Thus, RNAStructure must be installed, configured, and accesssible in the path. Input sequences are prescreened to prevent errors when folding; the min and max length tolerated can be assigned (-minlen, -maxlen), and any discrepancies between a sequence and any restraints/constraints will be recorded (-errorname to change log name) as a desynch error.

Usage

```
Batch runs folding programs

optional arguments:
  -h, --help            show this help message and exit

Input:
  fasta                 Reference Fasta File

Settings:
  -mode {R,V}           RNAStructure or Vienna
  -react <.react>       React file to use as restraints/constraints
  -restrict <.txt>      Limit folding to these specific transcripts
  -temp TEMP            [default = 310.15] Temperature to use for folding
  -cores CORES          [default = 4] Number of cores to use
  -distance DISTANCE    [default = 99999] Maximum pairing distance
  -minlen MINLEN        [default = 10] Minimum length to fold
  -maxlen MAXLEN        [default = 5000] Maximum length to fold
  -truncate TRUNCATE    [default = 0] Ignore <.react> for last N nucleotides
  -threshold TH         Apply hard constraints using this threshold

RNAStructure Settings:
  -slope SLOPE          [default = 1.8] Parameter for RNAstructure
  -intercept INTERCEPT  [default = -0.6] Parameter for RNAstructure
  -partition            Use the partition function
  -multiple             Output all structures rather than just MFE

Output:
  -errorname ERRORNAME  Name the error log
  -paraname PARANAME    Name the parameter log
  -outdir OUTDIR        Name the out directory
  -ct CT                Name the ct folder
  -ps PS                Name the ps folder
  -bp BP                Name the bp folder (Partition Only)
  -pfs PFS              Name the pfs folder (Partition Only)
```

sf3__rtsc__to__react.py

This module uses two <.rtsc> files (reagent -/+) and the sequences they were generated with (<.fasta>) to calculate per base reactivity. By default, the sample will be normalized to its own 2-8% scale; invoking the -scale flag followed by a <.scale> file will apply the 2-8% scale generated by another sample, thus calibrating both to a common normalization scale between samples. The specificity of the reactivity caluculation is determined by the -bases flag (default AC).

Usage

```
Generates a <.react> file from two <.rtsc> files

optional arguments:
  -h, --help            show this help message and exit

Input:
  control               Control <.rtsc> file
  treatment             Reagent <.rtsc> file
  fasta                 Transcript <.fasta> file
```

```
Settings:
  -restrict <.txt>      Limit analysis to these specific transcripts
  -scale <.scale>       Provide a normalizaiton file for calculation
  -threshold THRESHOLD  [default = 7.0] Maximum Reactivity Cap
  -ln_off               Do not take the natural log of the stop counts
  -nrm_off              Do not perform final 2-8% reactivity normalization
  -bases ACGT           [default = AC] Reaction Specificity

Output:
  -save_fails           Log transcripts with zero or missing scales
  -name NAME            Specify output file name
```

sf3__rx__correlation.py

Placeholder

```
Reformats <.rtsc>/<.react> for easy correlation analysis

optional arguments:
  -h, --help        show this help message and exit

Input:
  fasta             Reference Fasta
  rx                Input <.rx> files

Settings:
  -verbose          Display metrics
  -bases ACGT       [default = ACGT] Nucleotide specifictiy
  -restrict <.txt>  Filter to these transcripts via coverage file

Output:
  -name NAME        Specify output file name
```

sf3__structure__statistics.py

In PPV mode, each potential pair of input directories is fed transcript-wise through scorer, reporting the PPV of accepted~predicted.

```
Summarizes or compares MFE <.ct> files

optional arguments:
  -h, --help          show this help message and exit
```

```
Input:
  -dyr DYR [DYR ...]  CT directories

Settings:
  -mode {R,F,P}       Raw/Fused/PPV statistics
  -offset OFFSET      Number of Underscores in Transcript Names
  -na NA              [default = NA] Null DeltaG value
  -digits <number>    [default = 5] Decimal spots to report

Output:
  -name NAME          Output file name
```

References

Ding, Yiliang, Chun Kit Kwok, Yin Tang, Philip C Bevilacqua, and Sarah M Assmann. 2015. "Genome-Wide Profiling of in Vivo RNA Structure at Single-Nucleotide Resolution Using Structure-Seq." *Nature Protocols* 10 (7): 1050.

Ding, Yiliang, Yin Tang, Chun Kit Kwok, Yu Zhang, Philip C Bevilacqua, and Sarah M Assmann. 2014. "In Vivo Genome-Wide Profiling of RNA Secondary Structure Reveals Novel Regulatory Features." *Nature* 505 (7485): 696–700.

Dobin, Alexander, Carrie A Davis, Felix Schlesinger, Jorg Drenkow, Chris Zaleski, Sonali Jha, Philippe Batut, Mark Chaisson, and Thomas R Gingeras. 2013. "STAR: ultrafast Universal RNA-Seq Aligner." *Bioinformatics* 29 (1): 15–21.

Langmead, Ben, and Steven L Salzberg. 2012. "Fast Gapped-Read Alignment with Bowtie 2." *Nature Methods* 9 (4): 357.

Martin, Marcel. 2011. "Cutadapt Removes Adapter Sequences from High-Throughput Sequencing Reads." *EMBnet. Journal* 17 (1): 10–12.

Reuter, Jessica S, and David H Mathews. 2010. "RNAstructure: software for RNA Secondary Structure Prediction and Analysis." *BMC Bioinformatics* 11 (1): 1–9.

Ritchey, Laura E, Zhao Su, Yin Tang, David C Tack, Sarah M Assmann, and Philip C Bevilacqua. 2017. "Structure-Seq2: sensitive and Accurate Genome-Wide Profiling of RNA Structure in Vivo." *Nucleic Acids Research* 45 (14): e135–e135.

Tack, David C, Yin Tang, Laura E Ritchey, Sarah M Assmann, and Philip C Bevilacqua. 2018. "StructureFold2: Bringing Chemical Probing Data into the Computational Fold of RNA Structural Analysis." *Methods* 143: 12–15.

Team, R Core, and others. 2013. "R: A Language and Environment for Statistical Computing."