Restriction site-associated DNA from Python-implemented Digestion Simulations (RApyDS) Technical Documentation

Last updated: February 2019

Title Restriction site-associated DNA from Python-implemented Digestion Simulations (RApyDS)

Description RApyDS, a simulation tool that provides users with evaluation metrics to aid in choosing suitable REs based on their target RADseq design. RApyDS can perform simulations for single- or double-digest RADseq, preferably with a supplied reference genome. The tool outputs an overview page, electrophoresis visualization, mapping of restriction cut sites, and RAD loci density across the genome. If supplied with an annotation file, the program can also output evaluation metrics for a specified genomic feature.

Language Python, BASH

Dependencies BWA

License GNU General Public License v3.0

Contents

1 Technical Specifications

1.1 Software Requirements

- Linux OS
- Python 3+
- Python Pip3
- BWA (>0.7.12) (http://bio-bwa.sourceforge.net/)
- Firefox (for viewing html files)

1.2 Package Files

- **database/** - contains the default restriction enzyme database file
- **docs/** - contains the documentation files and sample output images
- **enzyme/** - contains the enzyme configuration files for the program run
- **src/** - contains the CSS and Javascript source code files for the output report files
- **templates/** - contains the html source code files for the output report files
- **LICENSE** - license file
- **README.md** - quick start guide
- **bwa_aln.sh** - shell script to run `BWA` alignment of the *reads*
- **bwa_index.sh** - shell script to run `BWA` indexing on the input `FASTA` file
- **create_histogram.py** - python script that plots the RAD Loci density per input sequence, per restriction enzyme
- **create_html.py** - python script that creates the html reports
- **rapyds.py** - main python script
- **remove_repeat.py** - python script that parses the `SAM` file output of the `BWA` program
- **requirements.txt** - list of python dependencies to be installed
- **tojson.py** - python script that converts the `csv` files to `json` files

2 Using RApyDS

2.1 Installation

Download the source from the Github repository https://github.com/pgcbioinfo/rapyds and extract it. Or clone using the command:
`git clone https://github.com/pgcbioinfo/rapyds.git`

Enter the directory and run `pip install -r requirement.txt` or
`pip3 install -r requirements.txt` whichever is applicable.

2.2 Arguments

`-h, --help` - show this help message and exit
`-gc [GC]` - input GC frequency. Value must be between 0 and 1
`-dna [DNA]` - input estimated DNA length
`-i [I]` - directory containing the input files
`-pre [PRE]` - prefix of the input files (must match the file name of the sequence, annotation, and/or index files)
`-at [AT]` - target feature in annotation file (ex. gene region, exon, intron, etc) (default: gene)
`-db [DB]` - resriction enzyme dabatase file.
Format per line: SbfI,CCTGCA|GG
`-re [RE]` - file containing list of restriction enzyme to be tested
`-min [MIN]` - minimum fragment size (default: 200)
`-max [MAX]` - maximum fragment size (default: 300)
`-bp [BP]` - base pair read length for mapping (default: 100)
`-p [P]` - RADSeq protocol: use `ddrad` for double digestion (default: `orig`)
`-o [O]` - output file name (default: report)
`-t [T]` - number of threads (default: 16)

Optional Flags:
`--skip_bwa` - skip BWA indexing and alignment
`--skip_graph` - skip cut site location histogram graphing
`--skip_clean` - skip cleaning intermediate files after running
`--verbose` - print verbose output including debug information

2.3 Input Requirements

- User can only choose between having a directory input (using `-i` with `-pre`) or generating a sequence based on GC frequency (using `-gc` with `-dna`)
  With valid FASTA file (directory input):
  `./rapyds.py -i <input_dir> -pre <prefix> [other arguments]`

Without any FASTA file (generation of sequence based on GC frequency):

```
./rapyds.py -gc <value from 0-1> -dna <int:length> [other arguments]
```

- The input directory must contain at least sequence file in the standard FASTA format having the extension of either `.fasta`, `.fna`, or `.fa` and the prefix or file name as specified in the `-pre` argument.
- The annotation and index files **with filenames same as the prefix** (ex. `ecoli.fasta` and `ecoli.gff3`), must be inside the speicified input directory.
- The annotation file must be in the standard GFF3 format with the extension either be `.gff`, `.gff3`, or `.gtf`. While the index files must have the extensions `.amb`, `.ann`, `.bwt`, `.pac`, and `.sa`.
- User can only choose between `orig` (default) and `ddrad` as protocol (`-p`)
- Using the ddRAD protocol requires a `-re` argument or a list of restriction enzymes to be tested on.

  ```
  ./rapyds.py -i <input_dir> -pre <prefix> -p ddrad -re <path/to/RE_list.txt>
  ```

- In original RADSeq protocol, if no `-re` argument is given, by default the program uses all the restriction enzymes in the database
- Formats for the list of REs and the database are found in the next section.

2.4 File Formats

2.4.1 Database of Restriction Enzymes (-db)

This file will serve as the database of restriction enzymes. Each restriction enzyme must be a line in the file and should follow the format below: `Enzyme,CUT|SITE`

```
AccI,GT|MKAC
AciI,C|CGC
AclI,AA|CGTT
AfeI,AGC|GCT
AflII,C|TTAAGA
...
```

The program can accept the standard IUPAC nucleotide code except for the gap represented by `.` or `-`.

2.4.2 List of Restriction Enzymes (-re)

By default, if no `-re` argument is suppplied, the program uses the entire restriction enzyme database on the original RADSeq protocol. If the user wishes to run the program using a custom list of restriction enzyme, the input file for the `-re` argument must list down the enzymes separated by a line break.

When using the ddRAD protocol, the `-re` argument is required and the input file list its pair of restriction enzyme separated by a space in between. A sample is provided below:

```
EcoRI ApeKI
EcoRI MspI
ApeKI BfaI
...
```

2.4.3 Sequence File

The input genome file must follow the standard FASTA format. Below is an example for the first few lines:

```
>U00096.3 Escherichia coli str. K-12 substr. MG1655, complete
     genome
AGCTTTTCATTCTGACTGCAACGGGCAATATGTCTCTGTGTGGATTAAAAAAAGAGTGTCTGATAGCAGC

TTCTGAACTGGTTACCTGCCGTGAGTAAATTAAAATTTTATTGACTTAGGTCACTAAATACTTTAACCAA

TATAGGCATAGCGCACAGACAGATAAAAATTACAGAGTACACAACATCCATGAAACGCATTAGCACCACC

ATTACCACCACCATCACCATTACCACAGGTAACGGTGCGGGCTGACGCGTACAGGAAACACAGAAAAAAG

CCCGCACCTGACAGTGCGGGCTTTTTTTTTCGACCAAAGGTAACGAGGTAACAACCATGCGAGTGTTGAA
```

The program can accomodate multiple `FASTA` in one single file as long as each of the `FASTA` starts with a `>` symbol immediately followed by its *identifier*.

2.4.4 Feature / Annotation File

The feature or annotation file must follow the General Feature Format 3 (GFF3). Below is an example for the first few lines:

```
##sequence-region U00096.3 1 4641652
##species https://www.ncbi.nlm.nih.gov/Taxonomy/Browser/wwwtax.
     cgi?id=511145
U00096.3 Genbank region 1 4641652 . + . ID=id-1;Dbxref=taxon
     :511145;Is_circular=true;Name=ANONYMOUS;gbkey=Src;genome=
     chromosome;mol_type=genomic DNA;strain=K-12;substrain=MG1655
U00096.3 Genbank gene 190 255 . + . ID=gene-b0001;Dbxref=EcoGene:
     EG11277;Name=thrL;gbkey=Gene;gene=thrL;gene_biotype=
     protein_coding;gene_synonym=ECK0001,JW4367;locus_tag=b0001
U00096.3 Genbank CDS 190 255 . + 0 ID=cds-AAC73112.1;Parent=gene-
     b0001;Dbxref=ASAP:ABE-0000006,UniProtKB/
```

Note: The *identifiers* in the FASTA file must match the values in the first column of the corresponding GFF file to map the annotation entries to its respective source sequence.

2.5 Common Usage

Original RADSeq with:

- known or given input sequence file `ecoli.fasta` inside a directory `ecoli_dir`
  `./rapyds.py -i ecoli_dir -pre ecoli [other args]`
- known or given input sequence file with custom restriction enzyme list
  `./rapyds.py -i ecoli_dir -pre ecoli -re <path/to/RE_file.txt> [other args]`
- unknown sequence but with GC content/frequency
  `./rapyds.py -gc <GC frequency> -dna <sequence length> [other arguments]`

ddRADSeq with known genome:

```
./rapyds.py -i ecoli -pre ecoli -p ddrad -re <re_file.txt> [other args]
```

Note: In case `./rapyds.py` didn't work, an alternative is to run using `python rapyds.py`.

3 Output Files

After running RApyDS, it will produce a `.zip` file containing the generated report. The contents of zipped file has the following structure:

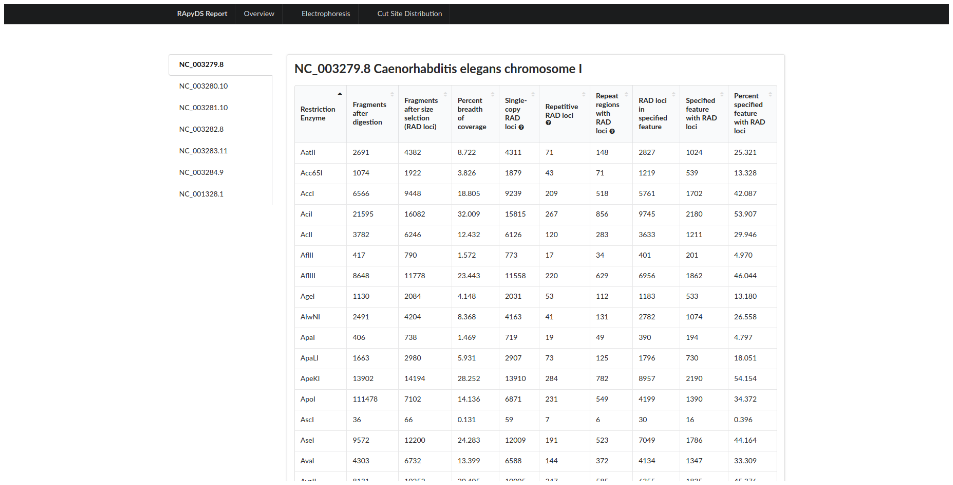
```
report/
├── output/
│   └── ...data files generated
├── src/
│   └── ...source files
├── cutsite.html
├── gel.html
└── index.html
```

There will be 3 html files:

- `index.html` contains the tabular information about the fragments
- `gel.html` is the electrophoresis simulation
- `cutsite.html` shows the cut site locations and distribution

3.1 Overview (index.html)

This html file shows information about the fragments after running with the original RADSeq simulation. Column headers can be clicked to sort the table according to the column's value in increasing or decreasing order. And on the left is a side bar with list of sequence identifier.



NC_003279.8
NC_003280.10
NC_003281.10
NC_003282.8
NC_003283.11
NC_003284.9
NC_001328.1

NC_003279.8 Caenorhabditis elegans chromosome I

Restriction Enzyme	Fragments after digestion	Fragments after size selction (RAD loci)	Percent breadth of coverage	Single-copy RAD loci	Repetitive RAD loci	Repeat regions with RAD loci	RAD loci in specified feature	Specified feature with RAD loci	Percent specified feature with RAD loci
AatII	2691	4382	8.722	4311	71	148	2827	1024	25.321
Acc65I	1074	1922	3.826	1879	43	71	1219	539	13.328
AccI	6566	9448	18.805	9239	209	518	5761	1702	42.087
AciI	21595	16082	32.009	15815	267	856	9745	2180	53.907
AclI	3782	6246	12.432	6126	120	283	3633	1211	29.946
AflII	417	790	1.572	773	17	34	401	201	4.970
AflIII	8648	11778	23.443	11558	220	629	6956	1862	46.044
AgeI	1130	2084	4.148	2031	53	112	1183	533	13.180
AlwNI	2491	4204	8.368	4163	41	131	2782	1074	26.558
ApaI	406	738	1.469	719	19	49	390	194	4.797
ApaLI	1663	2980	5.931	2907	73	125	1796	730	18.051
ApeKI	13902	14194	28.252	13910	284	782	8957	2190	54.154
ApoI	111478	7102	14.136	6871	231	549	4199	1390	34.372
AscI	36	66	0.131	59	7	6	30	16	0.396
AseI	9572	12200	24.283	12009	191	523	7049	1786	44.164
AvaI	4303	6732	13.399	6588	144	372	4134	1347	33.309

Figure 1: Overview table of the in silico digestion (original RADSeq) for C. elegans.

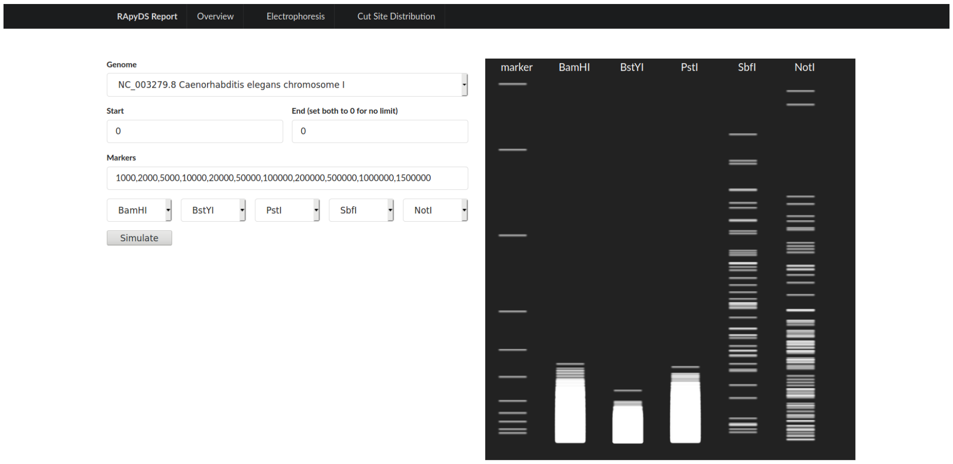
3.2 Electrophoresis (gel.html)

One of the two visualisation in the report is the electrophoresis simulation of the fragments after digestion. This makes use of `d3.js` and `d3-electrophoresis` javascript plug-ins.

User first selects a genome from the list, input the desired marker sizes, and selects up to five restriction enzymes to be compared. The webpage will read from the `json` output of RApyDS program. Loading the webpage might slow down from this part especially if the number of fragments is large. There is another option to further size select the digested fragments using the `start` and `end` input boxes. The page will only display the fragments within the desired size select range.

3.3 Cut Sites (cutsite.html)

The second and last visualisation is the cut site distribution. This also uses `d3.js` javascript plugin.



Genome: NC_003279.8 Caenorhabditis elegans chromosome I

Start: 0

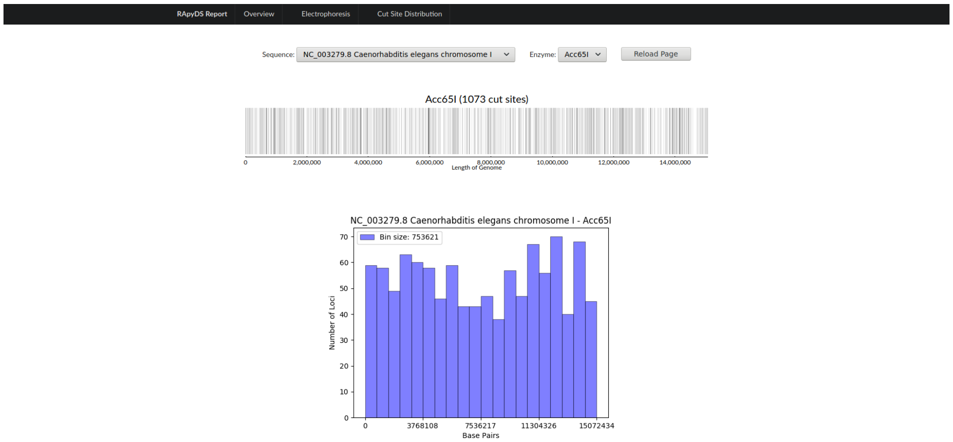
End (set both to 0 for no limit): 0

Markers: 1000,2000,5000,10000,20000,50000,100000,200000,500000,1000000,1500000

BamHI | BstYI | PstI | SbfI | NotI

Figure 2: Electrophoresis simulation for C. elegans with enzymes BamHI, BstYI, PstI, SbfI, and NotI.

For this visualisation, the data is from another `json` file that is also generated by the program which contains the cut site locations of the sequence. It is then rendered as a black line in perpendicular to the location in the sequence (in bp) as the x-axis.



Sequence: NC_003279.8 Caenorhabditis elegans chromosome I

Enzyme: Acc65I

Figure 3: Cut site markers location and distribution histogram for C. elegans chromosome I. with enzyme Acc65I