pysradb: A Python package to query next-generation sequencing metadata and data from NCBI Sequence Read Archive [version 1; peer review: 2 approved]

Saket Choudhary

Computational Biology and Bioinformatics, University of Southern California, Los Angeles, California, 90089, USA

Abstract
The NCBI Sequence Read Archive (SRA) is the primary archive of next-generation sequencing datasets. SRA makes metadata and raw sequencing data available to the research community to encourage reproducibility and to provide avenues for testing novel hypotheses on publicly available data. However, methods to programmatically access this data are limited. We introduce the Python package, pysradb, which provides a collection of command line methods to query and download metadata and data from SRA, utilizing the curated metadata database available through the SRAdb project. We demonstrate the utility of pysradb on multiple use cases for searching and downloading SRA datasets. It is available freely at https://github.com/saketkc/pysradb.

Keywords
bioinformatics, metadata, SRA, NGS, NCBI, GEO

This article is included in the Python collection.
Corresponding author: Saket Choudhary (saketkc@gmail.com)

Author roles: Choudhary S: Conceptualization, Formal Analysis, Methodology, Software, Writing – Original Draft Preparation, Writing – Review & Editing

Competing interests: No competing interests were disclosed.

Grant information: The author(s) declared that no grants were involved in supporting this work.

Copyright: © 2019 Choudhary S. This is an open access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

How to cite this article: Choudhary S. pysradb: A Python package to query next-generation sequencing metadata and data from NCBI Sequence Read Archive [version 1; peer review: 2 approved] F1000Research 2019, 8:532 https://doi.org/10.12688/f1000research.18676.1

First published: 23 Apr 2019, 8:532 https://doi.org/10.12688/f1000research.18676.1
**Introduction**

Several projects have made efforts to analyze and publish summaries of DNA-1 and RNA-seq2 datasets. Obtaining metadata and raw data from the NCBI Sequence Read Archive (SRA)3 is often the first step towards re-analyzing public next-generation sequencing datasets in order to compare them to private data or test a novel hypothesis. The NCBI SRA toolkit4 provides utility methods to download raw sequencing data, while the metadata can be obtained by querying the website or through the Entrez efetch command line utility6. Most workflows analyzing public data rely on first searching for relevant keywords in the metadata either through the command line utility or the website, gathering relevant sample(s) of interest and then downloading these. A more streamlined workflow can enable the performance of all these steps at once.

In order to make querying both metadata and data more precise and robust, the SRAdb7 project provides a frequently updated SQLite database containing all the metadata parsed from SRA. SRAdb tracks the five main data objects in SRA’s metadata: submission, study, sample, experiment and run. These are mapped to five different relational database tables that are made available in the SQLite file. The metadata semantics in the file remain as they are on SRA. The accompanying package, SRAdb8, made available in the R programming language9, provides a convenient framework to handle metadata queries and raw data downloads by utilizing the SQLite database. Though powerful, SRAdb requires the end user to be familiar with the R programming language and does not provide a command-line interface for querying or downloading operations.

The pysradb package10 builds upon the principles of SRAdb, providing a simple and user-friendly command-line interface for querying metadata and downloading datasets from SRA. It obviates the need for the user to be familiar with any programming language for querying and downloading datasets from SRA. Additionally, it provides utility functions that will further help a user perform more granular queries, which are often required when dealing with multiple datasets on a large scale. By enabling both metadata search and download operations at the command-line, pysradb aims to bridge the gap in seamlessly retrieving public sequencing datasets and the associated metadata.

pysradb10 is written in Python11 and is currently developed on GitHub under the open-source BSD 3-Clause License. To simplify the installation procedure for the end-user, it is also available for download through PyPI and bioconda12.

**Methods**

**Implementation**

pysradb10 is implemented in Python and uses pandas13 for data frame based operations. Since downloading datasets can often take a long time, pysradb displays progress for long haul tasks using tqdm14. The metadata information is read in the form of an SQLite15 database, made available by SRAdb7.

Each sub-command of pysradb contains a self-contained help string that describes its purpose and usage example. The help text can be accessed by passing the ‘–help’ flag. There is also additional documentation available for the sub-commands on the project’s website. We also provide example Jupyter16 notebooks that demonstrate the functionality of the Python API.

pysradb’s development primarily occurred on GitHub and the code is tested continuously using Travis CI webhook. This monitors all incoming pull requests and commits to the master branch. The testing happens on Python version 3.5, 3.6, and 3.7 on an Ubuntu 16.04 LTS virtual machine, while testing webhooks on the bioconda channel provide additional testing on Mac-based systems. Nevertheless, pysradb should run on most Unix derivatives.

**Operation**

pysradb10 can be run on either Linux- or Mac-based operating systems. It supports Python 3.5, 3.6 and 3.7. Requiring just two additional dependencies, pysradb can be easily installed using either a pip- or conda-based package manager via the bioconda12 channel.

An earlier version of this article can be found on bioRxiv https://doi.org/10.1101/578500

**Use cases**

pysradb10 provides a chain of sub-commands for retrieving metadata, converting one accession to other and downloading. Each sub-command is designed to perform a single operation by default, while additional operations
can be performed by passing additional flags. In the following section we demonstrate some of the use cases of these sub-commands.

pysradb uses **SRAmetadb.sqlite**, a SQLite file produced and made available by SRAdb project. The file itself can be downloaded using pysradb as:

```bash
$ pysradb srametadb
```

The **SRAmetadb.sqlite** file is required for all other operations supported by pysradb. This file is required for all the sub-commands to function. By default, pysradb assumes that the file is located in the current working directory. Alternatively, it can supplied using the ‘--db path/to/SRAmetadb.sqlite’ argument. The **SRAmetadb.sqlite** is available at: [https://s3.amazonaws.com/starbuck1/sradb/SRAmetadb.sqlite.gz](https://s3.amazonaws.com/starbuck1/sradb/SRAmetadb.sqlite.gz) or alternatively at [https://gbnci-abcc.ncifcrf.gov/backup/SRAmetadb.sqlite.gz](https://gbnci-abcc.ncifcrf.gov/backup/SRAmetadb.sqlite.gz). The examples here were run using **SRAmetadb.sqlite** with schema version 1.0 and creation timestamp 2019-01-25 00:38:19.

**Search**

Consider a case where a user is looking for Ribo-seq public datasets on SRA. These datasets will often have ‘ribosome profiling’ appearing in the abstract or sample description. We can search for such projects using the ‘search’ sub-command:

```bash
$ pysradb search ""ribosome profiling"" | head
```

| study_accession | experiment_accession | sample_accession | run_accession |
|-----------------|----------------------|-----------------|---------------|
| DRP003075       | DRX019536            | DRS026974       | DRR021383     |
| DRP003075       | DRX019537            | DRS026982       | DRR021384     |
| DRP003075       | DRX019538            | DRS026979       | DRR021385     |
| DRP003075       | DRX019540            | DRS026984       | DRR021387     |
| DRP003075       | DRX019541            | DRS026978       | DRR021388     |
| DRP003075       | DRX019543            | DRS026980       | DRR021390     |
| DRP003075       | DRX019544            | DRS026981       | DRR021391     |
| ERP013565       | ERX1264364           | ERS1016056      | ERR1190989    |

The results here list all relevant ‘ribosome profiling’ projects.

**Getting metadata for a SRA project**

Each SRA project (accession prefix ‘SRP’) on SRA consists of single or multiple experiments (accession prefix ‘SRX’) which are sequenced as single or multiple runs (accession prefix ‘SRR’). Each experiment is carried out on an individual biological sample (accession prefix ‘SRS’).

pysradb **metadata** can be used to obtain all the experiment, sample, and run accessions associated with a SRA project as:

```bash
$ pysradb metadata SRP010679 | head
```

| study_accession | experiment_accession | sample_accession | run_accession |
|-----------------|----------------------|-----------------|---------------|
| SRP010679       | SRX118285            | SRS290854       | SRR403882     |
| SRP010679       | SRX118286            | SRS290855       | SRR403883     |
| SRP010679       | SRX118287            | SRS290856       | SRR403884     |
| SRP010679       | SRX118288            | SRS290857       | SRR403885     |
| SRP010679       | SRX118289            | SRS290858       | SRR403886     |
| SRP010679       | SRX118290            | SRS290859       | SRR403887     |
| SRP010679       | SRX118291            | SRS290860       | SRR403888     |
| SRP010679       | SRX118292            | SRS290861       | SRR403889     |
| SRP010679       | SRX118293            | SRS290862       | SRR403890     |
| SRP010679       | SRX118294            | SRS290863       | SRR403891     |
| SRP010679       | SRX118295            | SRS290864       | SRR403892     |
| SRP010679       | SRX118296            | SRS290865       | SRR403893     |
However, this information by itself is often incomplete. We require detailed metadata associated with each sample to perform any downstream analysis. For example, the assays used for different samples and the corresponding treatment conditions. This can be done by supplying the ‘–desc’ flag:

```bash
$ pysradb metadata SRP010679 –desc | head -5
```

| study_accession | experiment_accession | sample_accession | run_accession | sample_attribute                      |
|-----------------|----------------------|------------------|--------------|----------------------------------------|
| SRP010679       | SRX118285            | SRS290854        | SRR403882    | source_name: PC3 human prostate cancer cells || cell line: PC3 || sample type: polyA RNA || treatment: vehicle |
| SRP010679       | SRX118286            | SRS290855        | SRR403883    | source_name: PC3 human prostate cancer cells || cell line: PC3 || sample type: ribosome protected RNA || treatment: vehicle |
| SRP010679       | SRX118287            | SRS290856        | SRR403884    | source_name: PC3 human prostate cancer cells || cell line: PC3 || sample type: polyA RNA || treatment: rapamycin |
| SRP010679       | SRX118288            | SRS290857        | SRR403885    | source_name: PC3 human prostate cancer cells || cell line: PC3 || sample type: ribosome protected RNA || treatment: rapamycin |

This can be further expanded to reveal the data in ‘sample_attribute’ column into separate columns via ‘–expand’ flag. This is most useful for samples that have associated treatment or cell type metadata available.

```bash
$ pysradb metadata SRP010679 –desc –expand
```

| run_accession | cell_line | sample_type                     | source_name                                  | treatment |
|--------------|-----------|---------------------------------|----------------------------------------------|-----------|
| SRR403882    | pc3       | polya rna                        | pc3 human prostate cancer cells              | vehicle   |
| SRR403883    | pc3       | ribosome protected rna           | pc3 human prostate cancer cells              | vehicle   |
| SRR403884    | pc3       | polya rna                        | pc3 human prostate cancer cells              | rapamycin |
| SRR403885    | pc3       | ribosome protected rna           | pc3 human prostate cancer cells              | rapamycin |
| SRR403886    | pc3       | polya rna                        | pc3 human prostate cancer cells              | pp242     |
| SRR403887    | pc3       | ribosome protected rna           | pc3 human prostate cancer cells              | pp242     |
| SRR403888    | pc3       | polya rna                        | pc3 human prostate cancer cells              | pp242     |
| SRR403889    | pc3       | ribosome protected rna           | pc3 human prostate cancer cells              | pp242     |
| SRR403890    | pc3       | polya rna                        | pc3 human prostate cancer cells              | pp242     |
| SRR403891    | pc3       | ribosome protected rna           | pc3 human prostate cancer cells              | pp242     |
| SRR403892    | pc3       | polya rna                        | pc3 human prostate cancer cells              | pp242     |
| SRR403893    | pc3       | ribosome protected rna           | pc3 human prostate cancer cells              | pp242     |

Any SRA project might consist of experiments involving multiple assay types. The assay associated with any project can be obtained by providing –assay flag:

```bash
$ pysradb metadata SRP000941 –assay | tr -s ' ' | cut -f5 -d ' ' | tail -n +2 | sort | uniq -c
```

999 Bisulfite-Seq
768 ChIP-Seq
121 OTHER
353 RNA-Seq
28 WGS

**Getting SRPs from GSE**
The Gene Expression Omnibus database (GEO) is the NCBI data repository for functional genomics data.
It accepts array and sequence-based data from gene profiling experiments. For sequence-based data, the corresponding raw files are deposited to the SRA. GEO assigns a dataset accession (accession prefix ‘GSE’) that is linked to the corresponding accession on the SRA (accession prefix ‘SRP’). It is often necessary to interpolate between the two accessions. `gse-to-srp` sub-command allows converting GSE to SRP:

```
$ pysradb gse-to-srp GSE24355 GSE25842
```

| study_alias | study_accession |
|-------------|----------------|
| GSE24355    | SRP003870       |
| GSE25842    | SRP005378       |

It can be further expanded to obtain the corresponding experiment and run accessions:

```
$ pysradb gse-to-srp --detailed --expand GSE100007 | head
```

| study_alias | experiment_accession | sample_accession | experiment_alias | sample_alias |
|-------------|----------------------|------------------|-----------------|--------------|
| GSE100007   | SRX2916198           | SRS2282390       | GSM2667747      | GSM2667747   |
| GSE100007   | SRX2916199           | SRS2282391       | GSM2667748      | GSM2667748   |
| GSE100007   | SRX2916200           | SRS2282392       | GSM2667749      | GSM2667749   |
| GSE100007   | SRX2916201           | SRS2282393       | GSM2667750      | GSM2667750   |
| GSE100007   | SRX2916202           | SRS2282394       | GSM2667751      | GSM2667751   |
| GSE100007   | SRX2916203           | SRS2282395       | GSM2667752      | GSM2667752   |
| GSE100007   | SRX2916204           | SRS2282396       | GSM2667753      | GSM2667753   |
| GSE100007   | SRX2916205           | SRS2282397       | GSM2667754      | GSM2667754   |
| GSE100007   | SRX2916206           | SRS2282400       | GSM2667755      | GSM2667755   |

**Getting a list of GEO experiments for a GEO study**

Any GEO study (accession prefix ‘GSE’) will involve a collection of experiments (accession prefix ‘GSM’). We can obtain an entire list of experiments corresponding to the study using the `gse-to-gsm` sub-command from `pysradb`:

```
$ pysradb gse-to-gsm GSE41637 | head
```

| study_alias | experiment_alias |
|-------------|------------------|
| GSE41637    | GSM1020640_1     |
| GSE41637    | GSM1020641_1     |
| GSE41637    | GSM1020642_1     |
| GSE41637    | GSM1020643_1     |
| GSE41637    | GSM1020644_1     |
| GSE41637    | GSM1020645_1     |
| GSE41637    | GSM1020646_1     |
| GSE41637    | GSM1020647_1     |
| GSE41637    | GSM1020648_1     |

However, a list of GSM accessions is not useful if one is performing any downstream analysis, which essentially requires more detailed information about the metadata associated with each experiment. This relevant metadata associated with each sample can be obtained by providing `gse-to-gsm` additional flags:

```
$ pysradb gse-to-gsm --desc GSE41637 | head
```

| study_alias | experiment_alias | sample_attribute |
|-------------|------------------|-----------------|
| GSE41637    | GSM1020640_1     | source_name: mouse_brain || strain: DBA/2J || tissue: brain |
| GSE41637    | GSM1020641_1     | source_name: mouse_colon || strain: DBA/2J || tissue: colon |
| GSE41637    | GSM1020642_1     | source_name: mouse_heart || strain: DBA/2J || tissue: heart |
| GSE41637    | GSM1020643_1     | source_name: mouse_kidney || strain: DBA/2J || tissue: kidney |
| GSE41637    | GSM1020644_1     | source_name: mouse_liver || strain: DBA/2J || tissue: liver |
| GSE41637    | GSM1020645_1     | source_name: mouse_lung || strain: DBA/2J || tissue: lung |
| GSE41637    | GSM1020646_1     | source_name: mouse_skm || strain: DBA/2J || tissue: skeletal muscle |
| GSE41637    | GSM1020647_1     | source_name: mouse_spleen || strain: DBA/2J || tissue: spleen |
| GSE41637    | GSM1020648_1     | source_name: mouse_testes || strain: DBA/2J || tissue: testes |
The metadata information can then be parsed from the sample_attribute column. To obtain more structured metadata, we can use an additional flag ‘--expand’:

```bash
$ pysradb gse-to-gsm -desc --expand GSE41637 | head
```

| study_alias | experiment_alias | source_name | strain | tissue    |
|------------|------------------|-------------|--------|-----------|
| GSE41637   | GSM1020640_1     | mouse_brain | dba/2j | brain     |
| GSE41637   | GSM1020641_1     | mouse_colon | dba/2j | colon     |
| GSE41637   | GSM1020642_1     | mouse_heart | dba/2j | heart     |
| GSE41637   | GSM1020643_1     | mouse_kidney| dba/2j | kidney    |
| GSE41637   | GSM1020644_1     | mouse_liver | dba/2j | liver     |
| GSE41637   | GSM1020645_1     | mouse_lung  | dba/2j | lung      |
| GSE41637   | GSM1020646_1     | mouse_skm   | dba/2j | skeletal muscle |

**Getting SRR from GSM**

gsm-to-srr allows conversion from GEO experiments (accession prefix ‘GSM’) to SRA runs (accession prefix ‘SRR’):

```bash
$ pysradb gsm-to-srr GSM1020640 GSM1020646
```

| experiment_alias | run_accession |
|------------------|---------------|
| GSM1020640_1     | SRR594393     |
| GSM1020646_1     | SRR594399     |

**Downloading SRA datasets**
pysradb enables seamless downloads from SRA. It organizes the downloaded data following the NCBI hierarchy: ‘SRP’ => ‘SRX’ => ‘SRR’ of storing data. Each ‘SRP’ (project) has multiple ‘SRX’ (experiments) and each ‘SRX’ in turn has multiple ‘SRR’ (runs). Multiple projects can be downloaded at once using the download sub-command:

```bash
$ pysradb download -p SRP000941 -p SRP010679
```

download also allows Unix pipes-based inputs. Consider our previous example of the project SRP000941 with different assays. However, we want to be able to download only ‘RNA-seq’ samples. We can do this by subsetting the metadata output for only ‘RNA-seq’ samples:

```bash
$ pysradb metadata SRP000941 --assay | grep 'study|RNA-Seq' | pysradb download
```

This will only download the ‘RNA-seq’ samples from the project.

**Summary**
pysradb provides a command-line interface to query metadata and download sequencing datasets from the SRA. It enables seamless retrieval of metadata and conversion between different accessions. pysradb is written in Python 3 and is available on Linux and Mac OS. The source code is hosted on GitHub and licensed under BSD 3-clause license. It is available for installation through PyPI and bioconda.

**Data availability**

**Underlying data**

- Dataset from DDBJ Sequence Read Archive, Accession number DRP003075: [https://identifiers.org/insdc.sra/DRP003075](https://identifiers.org/insdc.sra/DRP003075)
- Dataset from EMBL-EBI Sequence Read Archive, Accession number ERP013565: [https://identifiers.org/insdc.sra/ERP013565](https://identifiers.org/insdc.sra/ERP013565)
- Dataset from Gene Expression Omnibus, Accession number GSE24355: [https://identifiers.org/geo/GSE24355](https://identifiers.org/geo/GSE24355)
- Dataset from Gene Expression Omnibus, Accession number GSE25842: [https://identifiers.org/geo/GSE25842](https://identifiers.org/geo/GSE25842)
- Dataset from Gene Expression Omnibus, Accession number GSE100007: [https://identifiers.org/geo/GSE100007](https://identifiers.org/geo/GSE100007)
- Dataset from Gene Expression Omnibus, Accession number GSE41637: [https://identifiers.org/geo/GSE41637](https://identifiers.org/geo/GSE41637)
Discussions and comments on the software and manuscript. The author acknowledges support from the USC University of Southern California (USC) and Shweta Ramdas at the University of Pennsylvania for helpful

The author thanks Amal Thomas, Meng Zhou, Rishvanth Prabakar, Wenzheng Li, and Xiaojing Ji at the

The author(s) declared that no grants were involved in supporting this work.

Grant information

The author(s) declared that no grants were involved in supporting this work.

Acknowledgments

The author thanks Amal Thomas, Meng Zhou, Rishvanth Prabakar, Wenzheng Li, and Xiaojing Ji at the University of Southern California (USC) and Shweta Ramdas at the University of Pennsylvania for helpful discussions and comments on the software and manuscript. The author acknowledges support from the USC Provost Graduate Research Fellowship.

References

1. MacArthur DG, Balasubramanian S, Frankish A, et al.: A systematic survey of loss-of-function variants in human protein-coding genes. Science. 2012; 335(6070): 823–828. PubMed Abstract | Publisher Full Text | Free Full Text

2. Lachmann A, Torre D, Keenan AB, et al.: Massive mining of publicly available RNA-seq data from human and mouse. Nat Commun. 2018; 9(1): 1366. PubMed Abstract | Publisher Full Text | Free Full Text

3. Collado-Torres L, Nellore A, Kammers K, et al.: Reproducible RNA-seq analysis using recount2. Nat Biotechnol. 2017; 35(4): 319–321. PubMed Abstract | Publisher Full Text

4. Leinonen R, Sugawara H, Shumway M, et al.: The sequence read archive. Nucleic Acids Res. 2011; 39(Database issue): D19-D21. PubMed Abstract | Publisher Full Text | Free Full Text

5. SRA Toolkit Development Team: Sra toolkit. 2018; [Online; accessed 10-December-2018]. Reference Source

6. Kans J: Entrez direct: E-utilities on the unix command line. 2018. Reference Source

7. Zhu Y, Stephens RM, Melzer PS, et al.: SRAdb: query and use public next-generation sequencing data from within R. BMC Bioinformatics. 2013; 14(1): 19. PubMed Abstract | Publisher Full Text | Free Full Text

8. Zhu J, Davis S: Bioconductor:sradb. 2018. Publisher Full Text

9. R Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria. Reference Source

10. Choudhary S: saketkc/pysradb v0.9.0. 2019. http://www.doii.org/10.5281/zenodo.2579446

11. van Rossum G, Drake FL: The Python Language Reference Manual. Network Theory Ltd., 2011, ISBN 19069660141, 97819069660140. Reference Source

12. Grünig B, Dale R, Sjödin A, et al.: Bioconda: sustainable and comprehensive software distribution for the life sciences. Nat Methods. 2018; 15(7): 475–476. PubMed Abstract | Publisher Full Text

13. McKinney W: Data structures for statistical computing in python. In Stefan van der Walt and Jarrod Millman, editors, Proceedings of the 9th Python in Science Conference. 2010; 51–56. Reference Source

14. da Costa-Luis C, Stephen L, Mary H, et al.: tqdm/tqdm: tqdm v4.20.0 stable. 2018. Publisher Full Text

15. Sqlite home page. 2018; [Online; accessed 10-December-2018]. Reference Source

16. Kluver T, Ragan-Kelley B, Pérez F, et al.: Jupyter notebooks - a publishing format for reproducible computational workflows. In F. Loizides and B. Schmidt, editors, Positioning and Power in Academic Publishing: Players, Agents and Agendas. IOS Press, 2016; 87–90. Publisher Full Text

17. Ingolia NT, Ghaemmaghami S, Newman JR, et al.: Genome-wide analysis in vivo of translation with nucleotide resolution using ribosome profiling. Science. 2009; 324(5924): 218–223. PubMed Abstract | Publisher Full Text | Free Full Text

18. Barrett T, Willhite SE, Ledoux P, et al.: NCBI GEO: archive for functional genomics data sets--update. Nucleic Acids Res. 2013; 41(Database issue): D991-D995. PubMed Abstract | Publisher Full Text | Free Full Text

19. Blair JD, Hockemeyer D, Doudna JA, et al.: Widespread Translational Remodeling during Human Neuronal Differentiation. Cell Rep. 2017; 21(7): 2005–2016. PubMed Abstract | Publisher Full Text | Free Full Text

Dataset from NCBI Sequence Read Archive, Accession number SRP010679: https://identifiers.org/insdc.sra/SRP0106792

Dataset from NCBI Sequence Read Archive, Accession number SRP000941: https://identifiers.org/insdc.sra/SRP0009412

Software availability

Software available from: https://pypi.org/project/pysradb/.

Source code available from: https://github.com/saketkc/pysradb.

Archived source code at time of publication: https://doi.org/10.5281/zenodo.257944610.

License: BSD 3-Clause
20. Merkin J, Russell C, Chen P, et al.: Evolutionary dynamics of gene and isoform regulation in Mammalian tissues. Science. 2012; 338(6114): 1593–1599. PubMed Abstract | Publisher Full Text | Free Full Text

21. Hsieh AC, Liu Y, Edlind MP, et al.: The translational landscape of mTOR signalling steers cancer initiation and metastasis. Nature. 2012; 485(7396): 55–61. PubMed Abstract | Publisher Full Text | Free Full Text

22. Schultz MD, He Y, Whitaker JW, et al.: Human body epigenome maps reveal noncanonical DNA methylation variation. Nature. 2015; 523(7559): 212–6. PubMed Abstract | Publisher Full Text | Free Full Text
Open Peer Review

Current Peer Review Status:

Version 1

Reviewer Report 07 May 2019

https://doi.org/10.5256/f1000research.20450.r47516

© 2019 Dale R. This is an open access peer review report distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Ryan K. Dale
Bioinformatics and Scientific Programming Core, National Institute of Child Health and Human Development, National Institutes of Health, Bethesda, MD, USA

Pysradb is a Python package that extends the existing SRAdb R package by exposing command-line functionality such that metadata queries can be piped to other command-line tools.

Overall this is a well-written tool with good documentation and tests. As-is, it is very useful and my comments here are only aimed at making the tool more useful.

Downloading:
It would be really nice to be able to download fastq's, perhaps using SRAdb's strategy of downloading them from EBI.

When trying to run the example from the docs, "pysradb download -p SRP000941", I got a 550 HTTP error. Upon closer inspection, some of the SRAs in this entry are no longer available, but it causes the whole download to fail. I'm not quite sure of the underlying cause—this seems to be a mismatch between what's in SRA and the metadata database even though I just downloaded a fresh database with a timestamp of a week ago.

Two suggestions for making the download more robust. The first is to move on if there are any failures, reporting any failures at the end to stderr. The second is to provide a --list option that only prints the table with URLs such that the user can extract the URLs as needed (for example, to allocate to separate nodes on a cluster where failures can be handled by other mechanisms). With the current set of arguments, the only way to do the latter is to interactively answer "N" to the question of whether to continue downloading.

It could be useful to just download individual SRRs; I do not see that functionality available.

Searching:
An explanation of the search syntax would be helpful (e.g., SQL syntax, which allows wildcards but
not regular expressions). More complex examples would be good here. Given the heterogeneity of user-entered metadata in SRA, it would be convenient to use regular expressions when searching (which appears to be possible with Python and sqlite3 using a callback function).

I would prefer a nicer-formatted "no results found" printed to stderr rather than RuntimeWarning and UserWarning.

**Other:**
An interface to check the timestamp from the metaInfo table of the database would be helpful.

This may be a personal preference, but I would rather have output be tab-separated instead of requiring additional `tr -s " " | cut -d " "` commands (or awk) to do any sort of manipulation on the command line.

**Is the rationale for developing the new software tool clearly explained?**
Yes

**Is the description of the software tool technically sound?**
Yes

**Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?**
Yes

**Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?**
Yes

**Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?**
Yes

**Competing Interests:** No competing interests were disclosed.

**Reviewer Expertise:** bioinformatics, genomics, chromatin, gene expression throughout development

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Reviewer Report 26 April 2019  
https://doi.org/10.5256/f1000research.20450.r47560
Pysradb is a command line utility, written in python, which provides an easy scriptable interface for querying metadata and datasets from the SRA database.

The authors correctly point out that interacting with GEO/SRA through the official site and APIs can be slow and frustrating, and thus having a tool which can make this process more streamlined is of great value.

Both the paper and the documentation for the tool are well written and easy to understand. The tool provides a series of individual operations performing queries, translations or downloads and these can be run individually or chained together through pipes (which is really nice!).

The tool requires an initial download of the sqlite database from the SRAdb project. Whilst I can see how this then makes all subsequent operations quick, it does mean that you have to download a >2GB file (which expands to >30GB), taking 30+ mins before you can do anything with the program. It presumably also means that you need to re-download this file every time there is an update to the data in GEO otherwise your searches are likely to be out of date. On our site at least, people are often getting data for papers which have just been released so this is going to entail a lot of waiting for this file to download. It would be great if there was a way to point to a publicly accessible SQL server to do queries without having to do the local download, and then providing the option of pulling a local copy if you need greater performance. Also having a way to do incremental updates to this file instead of re-downloading the whole thing would be nice. Neither of these is a deal breaker, but they mightn't be too hard to implement?

The individual tools all worked as described, with the exception of the issues listed at the bottom, and the experience was generally very good with the tool.

One frustrating limitation is that the piping support is not universal throughout the tool. You can pipe into the download command, but not, for example, into the metadata command. Being able to chain operations such as:

```
pysradb gse-to-srp GSE24355 | pysradb metadata | pysra download
```

..or

```
pysradb search "oocyte development" | head | pysradb metadata
```

..would be really nice and presumably not too hard to support?

The downloading side of the tool is very useful and probably the part which is hardest to achieve in the main SRA site. Whilst this worked as described there are some aspects of the way it works
which make it a little frustrating. Firstly, it downloads SRA files, which hardly anyone wants - having a way to get the fastq files directly would be a really useful addition rather than having to run fastq-dump manually afterwards. It also downloads into a structured set of folders, which makes sense, but for large downloads means your files are scattered through multiple folders which makes life harder when you want to process them. Even the --out-dir option doesn't mean the files are in that directory, but just that it's used as a basename. For the names of the files it would be nicer to have a name which incorporated the relevant SRR/SRX ids and maybe the user submitted sample name so that you can actually have a meaningful and complete name from the file. For example, the types of filenames generated by SRA explorer (https://ewels.github.io/sra-explorer/) are a nice compromise between being predictable, unique and yet informative at the same time.

If I'm being really picky I'd also quibble a bit at the choice of some of the defaults in the API. For example, I can't see why the --desc and --expand options aren't the default for the metadata sub-program - give me everything in a nice format and let me cut that down if I don't need everything.

Overall this tool is really nice and will be useful for a lot of people. With a small amount of refinement this is likely to become part of our standard toolbox.

---------------

Minor points to address:

1) The API seems to have changed since the paper was written. The option to download the metadata is now pysradb metadb, and not pysradb srametadb. This is wrong in both the paper and the Jupyter notebook example on the github page and should be changed. It might be nice to allow srametadb as a fallback if people have been using the old name?

2) Some of the documentation is incomplete. For example the quickstart documentation at https://saket-choudhary.me/pysradb/quickstart.html#the-full-list-of-possible-pysradb-operations doesn't list the search operation so I couldn't look up the options I had for that.

3) The piping option is great, but on my system generates a crash if there is too much output (possibly more than the pipe buffer can hold?). Submitted as bug #7

4) The metadata command line in the paper is broken. You need double dashes before the options, so --desc rather than -desc. This seems to happen elsewhere as well and might just be an auto-format problem.

5) If using wget to download the progress bar for downloading doesn't work. The data comes down but the progress stays at 0%.

Is the rationale for developing the new software tool clearly explained?  
Yes

Is the description of the software tool technically sound?
Yes

**Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?**
Yes

**Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?**
Yes

**Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?**
Yes

**Competing Interests:** No competing interests were disclosed.

**Reviewer Expertise:** bioinformatics

**I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.**

---

The benefits of publishing with F1000Research:

- Your article is published within days, with no editorial bias
- You can publish traditional articles, null/negative results, case reports, data notes and more
- The peer review process is transparent and collaborative
- Your article is indexed in PubMed after passing peer review
- Dedicated customer support at every stage

For pre-submission enquiries, contact research@f1000.com