Bioinformatics, 36(8), 2020, 2592–2594
doi: 10.1093/bioinformatics/btz966
Advance Access Publication Date: 6 January 2020
Applications Note

Sequence analysis

ipyrad: Interactive assembly and analysis of RADseq datasets

Deren A.R. Eaton[1,*] and Isaac Overcast[2]

[1]Department of Ecology, Evolution, and Environmental Biology, Columbia University, New York, NY 10027, USA and [2]Department of Biology, Graduate School, University Center of the City University of New York, New York, NY 10016, USA

*To whom correspondence should be addressed.

Associate Editor: Russell Schwartz

Received on September 17, 2019; revised on December 9, 2019; editorial decision on December 23, 2019; accepted on December 31, 2019

Abstract

Summary: *ipyrad* is a free and open source tool for assembling and analyzing restriction site-associated DNA sequence datasets using *de novo* and/or reference-based approaches. It is designed to be massively scalable to hundreds of taxa and thousands of samples, and can be efficiently parallelized on high performance computing clusters. It is available both as a command line interface and as a Python package with an application programming interface, the latter of which can be used interactively to write complex, reproducible scripts and implement a suite of downstream analysis tools.

Availability and implementation: *ipyrad* is a free and open source program written in Python. Source code is available from the GitHub repository (https://github.com/dereneaton/ipyrad/), and Linux and MacOS installs are distributed through the conda package manager. Complete documentation, including numerous tutorials, and Jupyter notebooks demonstrating example assemblies and applications of downstream analysis tools are available online: https://ipyrad.readthedocs.io/.

Contact: de2356@columbia.edu

1 Introduction

Over the last decade molecular systematics has increasingly transitioned from investigating phylogenetic and phylogeographic patterns using datasets composed of one or only a handful of markers to massive datasets containing thousands or tens of thousands of loci. Among several methods that have been developed for subsampling loci from the genome (McKain *et al.*, 2018), restriction site-associated DNA sequencing (RADseq) and related methods have become a popular choice for their flexibility and affordability (Andrews *et al.*, 2016; Baird *et al.*, 2008; Davey *et al.*, 2011; Elshire *et al.*, 2011; Miller *et al.*, 2007; Peterson *et al.*, 2012). RADseq (and RADseq-like) protocols use restriction enzymes to digest (fragment) a genome such that regions proximal to restriction enzyme recognition sequences can be consistently selected for short-read sequencing. In contrast to whole genome sequencing or re-sequencing (Stratton, 2008), RADseq provides a more efficient way to gather high depth comparative sequence data shared across large numbers of samples, especially when genome sizes are large (Clugston *et al.*, 2019). For this reason, RADseq methods have been employed for diverse questions ranging from population genetics (García-Olivares et al., 2019), and phylogenetics (Eaton and Ree, 2013; Hipp *et al.*, 2014; Wagner *et al.*, 2013), to constructing linkage maps (Amores *et al.*, 2011; Rubin and Moreau, 2016), QTL-mapping (Palaiokostas *et al.*, 2013) and investigating DNA methylation (Schield *et al.*, 2016; Trucchi *et al.*, 2016). Even as future technological improvements reduce the per read cost of sequencing, reduced-representation methods will continue to offer advantages to studies that benefit from sequencing many populations or individuals (e.g. phylogeography), or that do not require sampling the entire genome (e.g. linkage mapping). Similarly, RADseq methods are likely to continue to improve in ways that promote these benefits, as with recent advances that reduce the cost of indexing and allow for PCR duplicate removal (Glenn *et al.*, 2019), and methods for enriching libraries to reduce missing data and increase multiplexing efficiency (Hoffberg *et al.*, 2016).

The process of organizing and making sense of the vast quantities of data that come off a modern sequencing instrument is nontrivial, and of great consequence. Simple parameter misspecification during the assembly process can have considerable impact on downstream analysis, potentially influencing the interpretation of the genetic patterns in the data (Linck and Battey, 2019; Shafer *et al.*, 2017). Prior to the availability of unified assembly tools (Catchen *et al.*, 2013; Eaton, 2014; Rochette *et al.*, 2019), these datasets were typically assembled in an *ad hoc* fashion using scripts developed in-house, leading to wide variability in the quality of assemblies being performed by the community. Additionally, downstream analyses typically involve writing complicated scripts to manage running multiple iterations of statistical inference software, organizing and post-processing the output and generating publication-ready plots. This proliferation of methods and lack of community standards has two significant consequences: (i) unnecessary complexity in assembly and analysis workflows which increases the potential for errors and (ii) a lack of reproducibility or oversight when *ad hoc* scripts are rarely reused or evaluated. What is needed is a user friendly, computationally robust and scalable method for both assembling and analyzing large-scale genomic datasets.

ipyrad was developed to address this need, and provides a simple, reproducible and well-documented RADseq assembly and analysis framework that is computationally efficient, massively scalable across large computing clusters, flexible to accommodate all variants of RADseq data types and suitable for population genetic scale as well as phylogenetic scale datasets. The *ipyrad* application programming interface (API) enables and encourages the creation of reproducible scientific workflows by providing a uniform, well-documented interface to several popular downstream phylogenetic and population genetics programs. *ipyrad* is a ground-up reimplementation of the RADseq assembly workflow implemented in pyRAD (Eaton, 2014), and includes numerous new capabilities which greatly extend the power, speed and utility of the original program.

2 *ipyrad* assembly process

The *ipyrad* assembly workflow is fully self-contained, capable of taking raw Illumina data from a sequencing facility and producing assembled output files without the need for pre- or post-processing by other software. The general workflow consists of seven steps: (i) demultiplexing raw reads to samples (based on single or combinatorial inline barcodes or indexed adapters) or alternatively importing data which has already been demultiplexed; (ii) quality control, filtering and trimming for adapter contamination; (iii) identifying read copies from the same locus within samples using *de novo* clustering or reference mapping. For paired-end data, the *de novo* method first merges read pairs with VSEARCH (Rognes *et al.*, 2016) before clustering, and indels are then imputed during a gapped alignment process which is performed by MUSCLE (Edgar, 2004). For reference assemblies, paired-end reads are mapped to the reference to produce gapped alignments, and mate pairs that map with incorrect orientation or to multiple locations (i.e. as paralogs) are discarded; (iv) joint estimation of sequencing error rate and sample heterozygosity; (v) making consensus basecalls and haplotype calls within samples; (vi) identifying orthology across samples by *de novo* clustering or reference mapped positions; and (vii) applying a final round of filtering and trimming to assembled loci, generating informative assembly statistics and writing output files in numerous useful formats for downstream analysis.

While *ipyrad* retains the general workflow of the original pyRAD method, the codebase has been completely refactored and rewritten with emphasis on performance and scalability. Even on a comparatively small dataset the performance gains are substantial. For example, using the original pyRAD (v.1.0) on a computer with 12 cores and 48 GB of RAM, the 13 *Pedicularis* samples from Eaton and Ree (2013) assembled in ~20 h. Using the same hardware, *ipyrad* assembles the same data in <30 min. The *Pedicularis* dataset has few samples, and is high-quality, single-end RAD data (Baird *et al.*, 2008). Paired-end data, very large datasets, low quality data and reference assembly methods obtain even greater performance improvements in the new implementation.

3 New capabilities implemented in *ipyrad*

3.1 Massive parallelization

Multi-process and multi-node computing (MPI; Gropp *et al.*, 1996) allows for efficient distribution of work across massive-scale computing clusters. *ipyrad* utilizes the ipyparallel Python library to distribute jobs across cores of a single computer, and can leverage MPI to distribute jobs across compute nodes on high performance computing clusters, even while working interactively. By default, *ipyrad* uses a load-balanced scheduler to distribute jobs among cores (including across different host nodes), and efficiently distributes threaded functions (e.g. VSEARCH clustering; Rognes *et al.*, 2016) across physical cores within compute nodes. Built on top of ipyparallel and MPI, *ipyrad* parallelization can easily and efficiently scale to hundreds of cores. Although the codebase of ipyrad is written in Python, it retains high performance through the use of just-in-time compilation (Lam *et al.*, 2015), and incorporation of industry standard compiled software into the assembly pipeline (Edgar, 2004; Li *et al.*, 2009; Li and Durbin, 2009; Martin, 2011; Quinlan, 2014; Rognes *et al.*, 2016).

3.2 Application programming interface

ipyrad provides a command line interface that is easy to use and which inherits interaction logic from its predecessor (Eaton, 2014). Additionally, *ipyrad* provides an API mode, which can be accessed programmatically to run interactive assemblies in Jupyter notebooks (Kluyver *et al.*, 2016). The API mode allows researchers to document, share and publish their assembly workflows, promoting reproducibility in science. The API mode also serves as a starting point for analyses using the downstream tools available through the *ipyrad-analysis* toolkit.

3.3 *De novo* and reference-based assemblies

To assess orthology of sequenced reads *ipyrad* implements two core assembly methods: *de novo*, in which a sequence similarity threshold is applied during a seed and extend clustering algorithm; and *reference*, in which reads are mapped to a reference genome. In addition, *ipyrad* can implement aspects of these methods in conjunction. For example, if a reference genome is quite distant from sample taxa then the *de novo + reference* method can recover more data by applying the reference workflow to mapped reads, and the *de novo* workflow to non-mapping reads, with the final dataset compiled of the two datasets together. Two alternative methods make use of a reference genome in a contrasting way, as a filter. In *de novo–reference* and *reference–reference* any reads that map to a 'filter-reference' file are removed from the dataset, prior to *de novo* or *reference* assembly of the remaining reads, which provides a useful means for removing sequences from contaminants or symbionts.

3.4 Branching architecture

Methods for assembling RADseq data are sensitive to the parameter settings used during filtering, mapping/clustering and base calling. For example, Linck and Battey (2019) showed that different minor allele frequency thresholds can produce significantly different inference of population structure. It is therefore critical to generate multiple datasets under a range of parameter settings for comparison (Crotti *et al.*, 2019; Paris *et al.*, 2017). *ipyrad* implements an iterative branching design that reduces redundancy and facilitates the generation of multiple datasets exploring a range of parameters settings, without the need to re-run the entire assembly. By saving intermediate files, different named assemblies can be restarted from intermediate steps of the assembly workflow. This vastly reduces redundancy in computation; enforces a reproducible workflow in which new branches do not overwrite earlier results; provides a convenient step in which to remove individuals from assemblies (failed samples, outgroups, etc.), or to merge samples from different libraries into a shared assembly.

3.5 Analysis tools

ipyrad includes an 'analysis' module which provides a powerful, simple and reproducible interface to several widely used methods for inferring phylogenetic relationships (RAxML; Stamatakis, 2014), population structure (STRUCTURE; Pritchard *et al.*, 2000) and admixture (TreeMix; Pickrell and Pritchard, 2012), among many others. In typical usage the analysis API will use an internal data structure generated by the *ipyrad* assembly process, but it is also flexible enough to import genotypes (e.g. VCF files) generated by other RADSeq assembly programs. Various population genetic and phylogenetic methods can be differentially impacted by missing data, therefore the analysis API provides simple options for filtering, imputing, consensus sampling and/or running replicate analyses to effectively quantify uncertainty around missing data. The analysis API leverages the massive parallelization provided by the *ipyrad* backend, manages organization of intermediate files and provides a simple interface for generating publication-ready plots of results (Eaton, 2019), contributing benefits of both usability and reproducibility.

4 Conclusion

ipyrad is a user friendly, robust, efficient, scalable and flexible program for assembling and analyzing RADseq datasets. The parallelization backend allows *ipyrad* to scale up to the limit of computational resources it is provided, facilitating the assembly of very large-scale datasets encompassing hundreds of taxa and thousands of samples. The API mode facilitates the creation of documented and shareable assembly workflows, promoting reproducibility. The analysis module provides a unified and coherent interface to many common downstream phylogenetic and population genetic inference methods, reducing the friction and overhead generated by file format conversion, configuration file creation and execution which are typically associated with implementation of these methods. This combination of API mode, parallelized backend and analysis tools allows researchers to efficiently perform, document and publish their full RADseq assembly and analysis workflows within a single computational framework, thus greatly reducing cognitive overhead and promoting reproducibility.

Acknowledgement

We thank Laura Bertola and Ed Myers for useful comments on an early draft of the manuscript.

Funding

This work was supported by grants from the National Science Foundation [DEB-1253710; DEB 1745562; DEB-1557059], the São Paulo Research Foundation [BIOTA, 2013/50297-0] and the National Aeronautics and Space Administration through the Dimensions of Biodiversity Program [DOB 1343578]. I.O. was supported by the Mina Rees Dissertation Fellowship in the Sciences provided by the Graduate Center of the City University of New York.

Conflict of Interest: none declared.

References

Amores,A. *et al.* (2011) Genome evolution and meiotic maps by massively parallel DNA sequencing: spotted gar, an outgroup for the teleost genome duplication. *Genetics*, **188**, 799–808.

Andrews,K.R. *et al.* (2016) Harnessing the power of RADseq for ecological and evolutionary genomics. *Nat. Rev. Genet.*, **17**, 81–92.

Baird,N.A. *et al.* (2008) Rapid SNP discovery and genetic mapping using sequenced RAD markers. *PLoS One*, **3**, e3376.

Catchen,J. *et al.* (2013) Stacks: an analysis tool set for population genomics. *Mol. Ecol.*, **22**, 3124–3140.

Clugston,J.A.R. *et al.* (2019) RADseq as a valuable tool for plants with large genomes—a case study in cycads. *Mol. Ecol. Resour.*, **19**, 1610–1622.

Crotti,M. *et al.* (2019) Causes and analytical impacts of missing data in RADseq phylogenetics: insights from an African frog (Afrixalus). *Zool. Scr.*, **48**, 157–167.

Davey,J.W. *et al.* (2011) Genome-wide genetic marker discovery and genotyping using next-generation sequencing. *Nat. Rev. Genet.*, **12**, 499–510.

Eaton,D.A.R. (2014) PyRAD: assembly of de novo RADseq loci for phylogenetic analyses. *Bioinformatics*, **30**, 1844–1849.

Eaton,D.A.R. (2019) Toytree; a minimalist tree visualization and manipulation library for Python. *Methods Ecol. Evol.*

Eaton,D.A.R. and Ree,R.H. (2013) Inferring phylogeny and introgression using RADseq data: an example from flowering plants (Pedicularis: Orobanchaceae). *Syst. Biol.*, **62**, 689–706.

Edgar,R.C. (2004) MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Res.*, **32**, 1792–1797.

Elshire,R.J. *et al.* (2011) A robust, simple genotyping-by-sequencing (GBS) approach for high diversity species. *PLoS One*, **6**, e19379.

García-olivares,V. *et al.* (2019) A topoclimate model for Quaternary insular speciation. *J. Biogeogr.*, **46**, 2769–2786.

Glenn,T.C. *et al.* (2019) Adapterama I: universal stubs and primers for 384 unique dual-indexed or 147,456 combinatorially-indexed Illumina libraries (iTru & iNext). *Peer J*, **7**, e7755.

Gropp,W. *et al.* (1996) A high-performance, portable implementation of the MPI message passing interface standard. *Parallel Comput.*, **22**, 789–828.

Hipp,A.L. *et al.* (2014) A framework phylogeny of the American oak clade based on sequenced RAD data. *PLoS One*, **9**, e93975.

Hoffberg,S.L. *et al.* (2016) RAD cap: sequence capture of dual-digest RAD seq libraries with identifiable duplicates and reduced missing data. *Mol. Ecol. Resour.*, **16**, 1264–1278.

Kluyver,T. *et al.* (2016) *Jupyter Notebooks—A Publishing Format for Reproducible Computational Workflows*. ELPUB, pp. 87–90.

Lam,S.K. *et al.* (2015) Numba: A LLVM-based Python JIT Compiler. In: *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*. ACM, New York, NY, USA.

Li,H. *et al.*; 1000 Genome Project Data Processing Subgroup. (2009) The sequence alignment/map format and SAMtools. *Bioinformatics*, **25**, 2078–2079.

Li,H. and Durbin,R. (2009) Fast and accurate short read alignment with Burrows–Wheeler transform. *Bioinformatics*, **25**, 1754–1760.

Linck,E. and Battey,C.J. (2019) Minor allele frequency thresholds strongly affect population structure inference with genomic data sets. *Mol. Ecol. Resour.*, **19**, 639–647.

Martin,M. (2011) Cutadapt removes adapter sequences from high-throughput sequencing reads. *EMBnet J.*, **17**, 10–12.

McKain,M.R. *et al.* (2018) Practical considerations for plant phylogenomics. *Appl. Plant Sci.*, **6**, e1038.

Miller,M.R. *et al.* (2007) Rapid and cost-effective polymorphism identification and genotyping using restriction site associated DNA (RAD) markers. *Genome Res.*, **17**, 240–248.

Palaiokostas,C. *et al.* (2013) Mapping the sex determination locus in the Atlantic halibut (*Hippoglossus hippoglossus*) using RAD sequencing. *BMC Genomics*, **14**, 566.

Paris,J. *et al.* (2017) Lost in parameter space: a road map for stacks. *Methods Ecol. Evol.*, **8**, 1360–1373.

Peterson,B.K. *et al.* (2012) Double digest RADseq: an inexpensive method for de novo SNP discovery and genotyping in model and non-model species. *PLoS One*, **7**, e37135.

Pickrell,J.K. and Pritchard,J.K. (2012) Inference of population splits and mixtures from genome-wide allele frequency data. *PLoS Genet*, **8**, e1002967.

Pritchard,J.K. *et al.* (2000) Inference of population structure using multilocus genotype data. *Genetics*, **155**, 945–959.

Quinlan,A.R. (2014) BEDTools: the Swiss-army tool for genome feature analysis. *Curr. Protoc. Bioinformatics*, **47**, 11–12.

Rochette,N.C. *et al.* (2019) Stacks 2: analytical methods for paired-end sequencing improve RADseq-based population genomics. *Mol. Ecol. Resour.*, **28**, 4737–4754.

Rognes,T. *et al.* (2016) VSEARCH: a versatile open source tool for metagenomics. *PeerJ*, **4**, e2584.

Rubin,B.E.R. and Moreau,C.S. (2016) Comparative genomics reveals convergent rates of evolution in ant–plant mutualisms. *Nat. Commun.*, **7**, 12679.

Schield,D.R. *et al.* (2016) Epi RAD seq: scalable analysis of genomewide patterns of methylation using next-generation sequencing. *Methods Ecol. Evol.*, **7**, 60–69.

Shafer,A.B.A. *et al.* (2017) Bioinformatic processing of RAD-seq data dramatically impacts downstream population genetic inference. *Methods Ecol. Evol.*, **8**, 907–917.

Stamatakis,A. (2014) RAxML version 8: a tool for phylogenetic analysis and post-analysis of large phylogenies. *Bioinformatics*, **30**, 1312–1313.

Stratton,M. (2008) Genome resequencing and genetic variation. *Nat. Biotechnol.*, **26**, 65–66.

Trucchi,E. *et al.* (2016) BsRADseq: screening DNA methylation in natural populations of non-model species. *Mol. Ecol.*, **25**, 1697–1713.

Wagner,C.E. *et al.* (2013) Genome-wide RAD sequence data provide unprecedented resolution of species boundaries and relationships in the Lake Victoria cichlid adaptive radiation. *Mol. Ecol.*, **22**, 787–798.