Deriving genotypes from RAD-seq short-read data using Stacks

Nicolas C Rochette & Julian M Catchen

Department of Animal Biology, University of Illinois at Urbana–Champaign, Urbana, Illinois, USA. Correspondence should be addressed to J.M.C. (jcatchen@illinois.edu).

Published online 30 November 2017; doi:10.1038/nprot.2017.123

Restriction site-associated DNA sequencing (RAD-seq) allows for the genome-wide discovery and genotyping of single-nucleotide polymorphisms in hundreds of individuals at a time in model and nonmodel species alike. However, converting short-read sequencing data into reliable genotype data remains a nontrivial task, especially as RAD-seq is used in systems that have very diverse genomic properties. Here, we present a protocol to analyze RAD-seq data using the Stacks pipeline. This protocol will be of use in areas such as ecology and population genetics. It covers the assessment and demultiplexing of the sequencing data, read mapping, inference of RAD loci, genotype calling, and filtering of the output data, as well as providing two simple examples of downstream biological analyses. We place special emphasis on checking the soundness of the procedure and choosing the main parameters, given the properties of the data. The procedure can be completed in 1 week, but determining definitive methodological choices will typically take up to 1 month.

INTRODUCTION

RAD-seq refers to a set of restriction enzyme-based protocols used to prepare reduced-representation libraries of genomic DNA. These protocols have made it practical to generate genome-scale genotype data for hundreds of individuals at a time, even in species for which no reference genome is available. They have, as a result, become widely used in the fields of population genetics, phylogeography, demography, hybridization, phylogenetics, and genetic mapping (reviewed in refs. 1,2).

A RAD-seq project typically involves obtaining samples, performing genomic DNA extractions, preparing a library of restriction enzyme-associated DNA, sequencing this library, analyzing the resulting short-read data, and finally performing biological analyses based on the derived genotypes. The molecular steps have been explained in detail in the different RAD-seq protocols available[3–9], and the biological analyses ought to be within the field of expertise of RAD-seq users. However, the issues associated with the analysis of raw sequencing data have received comparatively less attention.

Here, we describe a state-of-the-art procedure to derive genotypes from raw short-read data in the RAD-seq context. We demonstrate both a reference-based analysis and a strictly *de novo* analysis of the same data, as particularly in the fields in which RAD-seq is used it may not be possible to rely on a reference genome (**Box 1**). The procedure primarily relies on Stacks (http://catchenlab.life.illinois.edu/stacks/)[10,11], a software suite that is widely used for such analyses[12–15].

We cover the quality assessment and demultiplexing of the raw data, mapping of reads (if a reference-based analysis is performed), assembly of RAD loci and calling of genotypes, and filtering and export of these genotypes for use in biological analyses. We point out the checks that should be performed during the analysis to ensure its soundness, and explain how to select the relevant approach and parameter values (**Boxes 1–3**). Lastly, we discuss the intrinsic limitations of sequence data analysis and provide recommendations applicable to the design of a RAD-seq study so as to maximize its effectiveness.

Applications

The PROCEDURE is demonstrated on a data set that stems from a single-digest, shearing-based RAD-seq library[3], which uses inline barcodes and was single-end sequenced on the Illumina platform for population genetics purposes. However, the PROCEDURE is generally indifferent to the actual flavor of RAD-seq used. It is applicable to, in particular, ddRAD[5], GBS[4], ezRAD[7], quaddRAD[8], and Rapture[6]. Furthermore, it can easily be adapted to a range of different biological contexts and sequencing techniques. Thus, the pre-processing and genotyping process that we describe can be used directly for most population genetics purposes. The PROCEDURE will also be useful for genetic mapping analyses (linkage analysis or mapping of quantitative traits), as the overall structure of the analysis is the same, although some steps will differ (we do not, however, detail the required changes). Finally, the PROCEDURE can be used with paired-end data, when a double-digest protocol has been employed, with any barcoding strategy, and with data generated with some sequencing technologies other than Illumina (e.g., IonTorrent, although this technology requires additional steps to trim sequence data to a uniform length).

The workflow is aimed at all biologists, but basic UNIX command-line skills are required. These skills are beyond the scope of the protocol, and we encourage readers to engage in a hands-on bioinformatics course in anticipation. The PROCEDURE also requires some simple skills in cluster computing, and we cannot provide specific help on this topic because each cluster is different. However, the PROCEDURE is written so that it should be easily adaptable to the reader's particular system, and we provide some general advice for such adaptation (**Box 4**). Basic skills in the statistics-oriented programming language R[16] are desirable but could be replaced by experience with other statistics tools (e.g., Excel).

Limitations

Stacks relies on the stacking property of reads drawn from RAD-seq libraries: by construction, all reads from a given locus start

Box 1 | Reference-aligned versus *de novo* analyses

One of the strengths of a RAD-based experiment is that it can be applied in model and non-model organisms. The number of reference genomes is growing, but there are only a few very high-quality references that are actively maintained (e.g., human and mouse, maintained by the Genome Reference Consortium), and substantially fewer references have been fully assembled into chromosomes[39]; many more exist as small collections of scaffolds, and there are a multitude of poorly assembled draft genomes containing collections of tens of thousands of scaffolds.

For non-model organisms, often a reference genome for a specific species does not exist, but perhaps a phylogenetically closely related genome does (e.g., dozens of researchers work in various salmonid fishes, but the Atlantic salmon is the highest-quality reference genome available[40]).

In the case of a *de novo* analysis, done without a reference genome, the strengths are that the assembly algorithm in Stacks conservatively assembles putative alleles, followed by putative loci. If proper filtering is applied when creating the catalog (**Fig. 3**) and when analyzing the population (**Box 3**), a reliable set of loci can be identified. The downside is that each *de novo* sample contributes a certain number of false-positive loci—derived from convergent sequencing error or errors in the early stages of PCR amplification. Of course, without the genome, loci cannot be visualized positionally; instead each locus is examined independently to check if it is an outlier.

A reference genome helps separate out the false-positive loci from the true loci and importantly provides collinearity to nearby variants, increasing the chance of detecting true outliers. However, it also plays the role of a filter. If the reference is of draft quality, it will be missing some fraction of the true genome. It may also have notable errors, such as indels (e.g., remnants of PacBio sequencing), and certain regions of the genome may be present in multiple copies as uncollapsed haplotypes. Short-read aligners process each read independently and may produce different configurations of gaps in particular alignments with the same alignment score. Uncollapsed haplotypes may lead to reads being excluded because they appear to align to more than one place. Just as in the *de novo* case, alignment parameters should be optimized and the number of reads that align—and more importantly that of those that do not—should be tracked. If a reference is not of high quality, a complementary *de novo* analysis should be performed to compare how many loci may be missing from the reference genome or to identify outlier loci that are not in the assembled reference.

When the reference genome is distant, or if its quality is questionable, it makes sense to take a hybrid approach: assembling loci *de novo*, and then aligning their consensus sequences and backporting the positional information into the *de novo* data set (Stacks provides this capability). This allows one to take advantage of the reference genome without compromising the consistency of genotype calls. In addition, this hybrid approach enables a direct comparison of the *de novo* and reference-based analyses. When the genome is of good quality and the genomic properties (e.g., repeat content) of the system are such that *de novo* assembly of RAD loci works well, the two approaches should yield very similar results (**Supplementary Fig. 2**).

at the same position, the restriction enzyme cutting site. Because of this design, the analysis procedure described here is specific to RAD-seq data and is not applicable to other sequencing strategies (e.g., randomly sheared DNA or cDNA libraries). In particular, it is not compatible with hyRAD[9], as this technique uses restriction enzymes for bait design, but then relies on whole-genome shotgun libraries. A second consequence is that for paired-end sequencing data derived from a single-digest RAD-seq library, Stacks cannot currently make use of paired-end reads that have been sheared, even when a reference-based approach is taken. In addition, although the pipeline is suitable for most types of RAD-seq, there are some exceptions. Specifically, it is unsuitable for sequencing data drawn from pooled samples, because Stacks assumes diploidy for locus assembly and genotyping.

The applicability of the protocol can also be limited by the intrinsic properties of the data set considered. A general limitation of all sequence-based genotyping methods is that error rates become considerably higher when the coverage is low[17], as illustrated in **Figure 1**. Recent methods have tried to mitigate or circumvent this issue by using reference panels and genotype imputation[18] and/or by working with genotype likelihoods[19]. However, these approaches are especially relevant in the context of whole-genome shotgun sequencing, and when extensive genomic resources are available, as in humans. In the context of a RAD-seq project, in comparison, available genomic resources are typically much more limited, and the sequencing cost is not as preeminent. We, therefore, recommend targeting a high coverage ($\geq 20\times$), rather than relying on more sophisticated statistical treatments to correct lower-quality genotype data.

Comparison with other methods

A variety of methods have been used to process raw RAD-seq data. There is a clear distinction between methods that were developed specifically for RAD-seq data, and methods designed primarily for whole-genome sequencing (WGS) data but that can also be used for RAD-seq data analysis if a reference genome is available.

Among methods specific to RAD-seq, PyRAD[20] was developed for the *de novo* analysis of data sets comprising divergent samples, such as phylogenetic data sets. It uses the global alignment clustering algorithm implemented in USEARCH[21] to assemble RAD loci, and thus supports indels. AftrRAD[22] is another stand-alone pipeline that supports indels by using a clustering strategy based on the ACANA[23] pairwise alignment algorithm, and claims to be faster than PyRAD. Support for alignment-based clustering and indels is also available in Stacks, although this feature must be explicitly activated (by specifying the '`--gapped`' parameter). TASSEL[24] is a suite of command-line and graphical programs that focus on the so-called genotype-by-sequencing (GBS) flavor of RAD-seq[4] that is especially used in species of agronomical interest. TASSEL previously allowed for *de novo* assembly and genotyping through its UNEAK subunit[25], but support for this feature has been discontinued in TASSEL v5, presumably as genomic resources are becoming widely available in organisms in which GBS is applied. Finally, dDocent[26] is a composite pipeline that

Box 2 | Parameter choice for assembly and alignment

After having decided whether to use a *de novo* approach or a reference-based one (**Box 1**), the experimenter must consider the parameters that control the behavior of the tools involved in this approach. The aim should be to ensure that a large enough number of reads are used, but also to reduce the proportion of artifactual loci, alignments, and calls. In practice, for both the *de novo* approach and the reference-based one, the most sensitive parameters are those that control the allowed level of sequence dissimilarity within a locus, as they ultimately determine how loci are defined and which reads become part of them. In general terms, these parameters should be relaxed enough to account for genetic variation and sequencing errors, but strict enough to discriminate between paralogous loci.

In a *de novo* approach, the problem is that of assembly: reads for the same locus must be regrouped across alleles and across samples based on sequence similarity. In Stacks, this is done in a two-step manner. The ustacks unit first assembles loci in each sample independently, then the cstacks unit merges the loci of all samples into a common pool (the 'catalog'). The main parameters that control this process are the *M* parameter of ustacks and the *n* parameter of cstacks, which specify the number of allowed mismatches between reads during the two steps. More specifically, *M* controls the number of mismatches allowed between the two alleles of a heterozygote sample, whereas *n* controls the number of mismatches allowed between any two alleles of the population.

In addition, the *m* parameter of ustacks controls the number of identical reads required to initiate a new putative allele. Whether or not to perform gapped alignments (`--gapped` option of ustacks and cstacks) should also be considered at this stage. Depending on the data set, gapped alignments may provide more loci for the analysis, with the danger of building more complex, and potentially error-containing, loci.

These parameters must be optimized for any particular data set, because their optimal value depends on the amount of genetic diversity that is present and on the characteristics of the raw sequencing data itself[41–44]. In addition, although some population statistics estimates appear stable across a range of assembly parameter values, others, and in particular those derived from coalescent modeling, are sensitive to the actual values chosen[42,43]. Here, we follow the optimization approach detailed in ref. 44, which involves running series of *de novo* analyses with different parameter values and monitoring the number of polymorphic RAD loci found in 80% of samples or more (the *r80* loci), until a stable set of values for *M, n,* and *m* is found. The series demonstrated in Step 15A and **Figure 2** is the most basic one: *m* is set to 3, as this value performs well for a broad range of data sets, and *M* and *n* are varied while kept equal. Other approaches are similarly based on running series of computations using a range of parameter values but monitor other summary statistics, such as the total number of loci and number of SNPs present in at least 75% of samples[43], the frequency at which loci have one, two, or more than three alleles (i.e., 'primary stacks') per sample[41,42], or the number of repeats from the organism's reference genome (assuming one is available) that are being collapsed[41].

In a reference-based approach, in contrast, the problem is that of alignment: reads are assigned to loci based on their similarity to a reference sequence. The first methodological choice is that of the short-read aligner itself. A range of methods have been described, of which at least several are competitive. In this procedure, we use the BWA short-read aligner[34], but good results can also be obtained with other aligners. Tunable parameters differ between aligners, as they rely on different algorithms, but in general one can expect one parameter to control how many mismatches to the reference are allowed, and one to control how many indels are allowed. In the BWA-MEM method, this can be done (indirectly) through the `-B` and `-O` parameters, which control the penalty assigned for mismatches and gaps, respectively.

When a reference genome of a good quality is available, and the populations in the data set are close to it, default alignment parameters are likely to yield good results and, therefore, optimization may be unnecessary. In this case, the experimenter should focus on validating the effectiveness of default parameters (Step 15B). If the reference genome is substantially more distant from the samples, however, it becomes important to closely monitor alignment statistics (such as the percentage of reads aligning multiply and uniquely) to exclude potential biases, and it may be appropriate to compare alignments obtained with different methods or parameters, or even to compare the resulting biological results with those given by a *de novo* approach.

Another important parameter to control is the tendency of the aligner to perform partial alignments (alignments in which one end of the sequence is 'masked' or 'clipped' and not actually aligned, as if it did not exist). In BWA-MEM, this can be controlled through the `-L` option. An excess of partial alignments can cause errant SNP calls in the 3 region of RAD loci, so it can be useful to reduce the default aligner's clipping behavior. By default, the 'pstacks' unit of Stacks, the entry point of the reference-based pipeline, discards alignments in which >15% of the total length has been clipped (controllable through the `--max_clipped` parameter).

combines Rainbow[27] (for assembling RAD-seq loci), BWA (for aligning reads to loci), and FreeBayes (for calling genotypes; see below). Although all these major pipelines will recover a core set of reliable loci, there are differences depending on the chosen pipeline and parameters, and Stacks has been shown to perform well[28]. Furthermore, Stacks excels by providing mechanisms to allow the experimenter to quickly gain an intuition for the dynamics of the underlying data and extract from them the best biological signal, owing to its 'populations' module (**Box 3**). And although most other pipelines are single-nucleotide polymorphism (SNP)-based, Stacks provides an additional, rich set of information by calculating haplotypes from each RAD locus.

Methods oriented toward whole-genome shotgun sequencing data include SAMtools' mpileup[29], HaplotypeCaller[30], FreeBayes[31], and ANGSD[19]. Those that are currently used the most with RAD-seq data are HaplotypeCaller and ANGSD. HaplotypeCaller is part of the Broad Institute Genome Analysis Toolkit framework, in which it has replaced UnifiedGenotyper as the genotyping unit. ANGSD is a more recent genotyping software that is designed for low-coverage data, and thus includes tools to compute

Box 3 | Applying a population genomics context with the populations program

The main Stacks pipeline views a set of RAD-seq data as a generic metapopulation; it could be the parents and progeny from a genetic cross, a set of interspecific samples in a phylogenetic analysis, or one or more related populations of a single species. With the populations program, Stacks applies a biological context to the generic data—allowing the set of samples to be viewed as an explicit set of populations, filtering biologically implausible (or statistically insignificant) nucleotide sites from the data, calculating summary statistics at the nucleotide and haplotype level, and finally exporting the population genotypes in one or more formats.

The population map is a two-column file that specifies each sample in the analysis and its respective population. Assignments in the population map can be based on geography, phenotype, species, or another variable. Population maps can also be used to process subsets of the data—either population subsets or a subset excluding outlier samples from an analysis.

There are four primary filters that should be used, particularly in *de novo* analyses, to control for false-positive loci. They control (i) the fraction of individuals of a single population a locus must be found in to be processed (`-r`), (ii) the number of populations that a locus must be found in to be processed (`-p`), (iii) the MAF a variable site must possess to be included (`--min_maf`), and (iv) the maximum level of heterozygosity a variable site can possess to be included (`--max_obs_het`). The `-r` and `-p` filters primarily provide biological control (e.g., a population genetic analysis (widely available loci) versus a phylogenetic study (maximum loci across any subset of species)). The MAF and maximum heterozygosity filters help prevent processing data derived from false-positive SNP calls and the erroneous merging of paralogous loci, respectively. To exclude error, the MAF filter need only be high enough to ensure an allele is found in three or more samples, so it may be set as low as 1–5%, depending on the number of samples in the data set. Some statistical models are designed with more frequent alleles in mind, and to export data for this purpose (not just to exclude error), values of 5–10% are common. The maximum heterozygosity is used to exclude error and can be set at any reasonably high level, >60 or 70%.

In Stacks, each RAD locus (the 100–500 bp adjacent to the restriction enzyme cut site that was sequenced) is considered an independent unit. The populations program computes two sets of summary statistics to describe each population given in the population map—nucleotide level statistics for each variable position within each RAD locus, and haplotype-level statistics, which consider all of the SNPs from a single RAD locus as a set of haplotypes (these SNPs are phased because they were sequenced together). The batch_X.sumstats.tsv file contains nucleotide-level summary statistics, such as observed and expected heterozygosity, π, and F_{IS}, for each variable site, whereas batch_X.sumstats_summary.tsv contains mean values for these statistics for each population. The batch_X.hapstats.tsv file contains haplotype level statistics such as gene diversity and haplotype diversity.

If requested, the populations program will calculate a nucleotide-level F_{ST} value, using an analysis of molecular variance framework[45], between each pair of populations (`--fstats`). A Fisher's exact test is computed for each variant site to test if the allele frequencies are different between the two populations for that variant site and provides a *P* value for each individual F_{ST} measure. If a reference genome is available, the F_{ST} statistics can be smoothed across each chromosome (`-k`; π and F_{IS} can also be smoothed); bootstrapping can be employed to provide a *P* value for each sliding window region (`--bootstrap`). These measures are provided in the batch_X.fst_Y-Z.tsv files.

At the haplotype level, the populations are again compared pairwise, providing several F_{ST} statistics, including ϕ_{ST} and F_{ST}'[46,47]. Haplotype-based F_{ST} methods provide a stronger signal of differentiation at each locus, as there may be more haplotypes (and their associated variance can be more richly partitioned) than the two allelic states available to a single SNP. Both ϕ_{ST} and F_{ST}' are normalized F_{ST} methods that account for situations in which alternative alleles are not fixed, but there are instead no shared alleles between a pair of populations[48]. These measures are provided in the batch_X.phistats_Y-Z.tsv files, and ϕ_{ST} and F_{ST}', as well as haplotype and gene diversity, can also be smoothed across a genome.

Analysis of Stacks-independent data

The populations program can import a set of SNPs, generated from any sequencing method (e.g., RNA-seq or WGS), in a VCF file. All of the population-level analyses can be applied to these SNPs and the same array of exports is available as well.

If a closely related reference genome is the only option, a final option is to conduct a *de novo* assembly and to then align the consensus sequences from the *de novo* assembled loci to the reference. Stacks includes a script (integrate_alignments.py) that will import this location data back into the *de novo* data set so it can be used by the populations program.

certain population statistics based on raw genotype likelihoods, rather than on definite genotype calls. We note that no direct comparisons exist between Stacks in reference-based mode and software designed for WGS data. However, the genotyping models used in Stacks[11,32] should perform well when coverage is high or intermediate, which is what we recommend for RAD-seq (see above).

Overview of the Procedure

The PROCEDURE comprises five main parts: (i) preparation of the environment and of the data; (ii) demultiplexing and filtering of raw reads; (iii) application of the pipeline to a small test subset to make sure that it works properly, and that the chosen approach and parameters are appropriate; (iv) application of the pipeline to the complete data set; and (v) export for downstream biological analyses. It demonstrates both a *de novo* analysis and a reference-based one, in parallel. The workflow differs slightly between the two, the most notable differences arising in regard to parameter selection.

Throughout the protocol, we use a three-spined stickleback (*Gasterosteus aculeatus*) data set derived from the original data of ref. 12. This demonstration data set comprises 78 samples from four populations along the coast of Oregon (USA), of which two are oceanic ('Cushman Slough' (CS) and 'South Jetty' (SJ)) and

Box 4 | Advice for managing parallel jobs and files

Many steps of the protocol will probably need to be run on a computer cluster (TIMING section). Although the specifics of cluster computing vary from one system to another, we try to provide general advice below.

The user will generally connect, using a secure shell protocol (SSH) client, to a master/head node (i.e., computer) on which no computationally intensive tasks should be performed. From this master node, the user is able to submit 'jobs', in the form of shell scripts, to a job management system that will execute these scripts on computation nodes.

The user should also have the option of connecting directly to a computation node by asking for an 'interactive job'. This is particularly useful to test and debug scripts without having to submit them as batch jobs, or to perform moderately intensive tasks (e.g., for which the overall execution time is less than an hour). Be sure to keep track of the commands used while working interactively.

Running a single script through a job submission system is very similar to running the script directly, except that a little extra information must be provided, such as the group the user belongs to, a time limit, or the number of processors that the script is going to use (note that the number of processors must always be specified in two places independently, once to the programs that will be executed by the script and once to the job submission system, so that the proper resources can be reserved).

A common pitfall is that it may not be obvious in which directory the script is going to be executed. For this reason, it is practical to use only absolute paths, or to move ('cd') into the correct directory at the very beginning of the script. Another frequent issue is that scripts executed in batch mode may fail to find programs installed in non-standard places (i.e., other than `/usr/bin` or `/usr/local/bin`). This typically indicates that the user's shell environment, and in particular the PATH shell variable, has not been initialized properly. The user should make sure that his/her login files (e.g., `~/.bashrc`) are actually sourced and that (for clusters that use a module system) the scripts load the appropriate modules.

Running multiple similar jobs at the same time is often necessary, for instance, to perform the same operation for each sample in the data set. We recommend writing a pair of companion scripts: one script that will run through the submission system and that will actually perform the operation for a given sample, and one script that will be executed directly by the user on the master node and that will submit the first script as many times as there are samples. The least obvious part in this setup is how to inform the first script of the name of the sample it should process. Depending on the actual system, there are many ways of doing so. The simplest and most portable one is to pass the name in an environment variable (the value of which will be specific to each job).

The following illustrates a typical 'companion scripts' setup for the QSUB submission system interface, implemented by the TORQUE and Sun Grid Engine software, among others (by convention, we capitalize environment variables):

Script '2' (to be executed directly, on the master node):

```
cd /path/to/tests.ref
for spl in $(cut -f1 ../info/popmap.tests_samples.tsv); do
qsub -v "SAMPLE=$spl" ./script1.sh
done
```

Script '1' (to be submitted as a job):

```
cd /path/to/tests.ref
echo "Working on '$SAMPLE'."
fq_file=./cleaned/$SAMPLE.fq.gz
#and so on to process the sample.
```

two are freshwater ('Winchester Creek' (WC) and 'Pony Creek Reservoir' (PCR)). The RAD-seq library was prepared following ref. 3, using inline barcodes, and it was sequenced on a HiSeq 2000 platform with a read length of 101 bp. This data set is available as described in the MATERIALS section.

The first part, preparation of the environment, consists of obtaining access to a UNIX computer (either a stand-alone workstation with sufficient resources or a computing cluster), installing the required software on this system, creating a directory for the project (Step 1), and gathering all the relevant files and information into this directory.

The actual analysis begins with the filtering and demultiplexing all lanes of the raw data to obtain cleaned, per-sample sequencing data (Steps 1–11). This allows one to check the quality of the raw sequencing data, and simultaneously identify and remove unusable samples, if any. Some sequencing centers will return files that are already demultiplexed (per sample); the PROCEDURE is slightly simpler in this case, as the user does not have to manage barcodes, but these files should still be quality-filtered. In the case of already-demultiplexed data, the protocol changes as follows: Step 3 (formatting the barcode information) can be ignored; at Step 6, barcodes should already have been removed; at Step 8, remove the barcodes-related arguments from the `process_radtags` command; and at Step 9, the number of reads per sample is given by the per-input-file statistics instead of the per-barcode statistics.

The next stage is to experiment with the pipeline on a representative subset of samples. This allows one to survey the general properties of the data set and to assess which approach and parameter values are most suitable (Steps 12–15). We recommend this initial downscaling because performing large-scale analyses poses substantial organizational and computational resource issues, even more so when the analysis is to be run multiple times with a range of values for all parameters. The PROCEDURE describes the basic recommended workflow for selecting parameters, and a more general discussion of the parameters that must be considered, their effect, and the criteria that can be used to choose appropriate values, is given in **Box 2**.

The pipeline is then applied to the full data set (Steps 16–20) using the parameters chosen in the previous stage.

Finally, we demonstrate the filters that can be applied to the genotype data, and discuss how to obtain the main population genetics statistics (Step 21). We conclude the workflow by exporting the data in standard genotype file formats and performing two common biological analyses: we compute and smooth the fixation index (F_{ST}) across the genome (Steps 22–24) and carry out a principal components analysis (PCA) using the R library ADEgenet[33] (Steps 25–28).

MATERIALS

EQUIPMENT

Starting data

- Illumina sequencing read files returned by a sequencing center, in FASTQ format
- Information on the protocol that was used to create the RAD-seq library, including the restriction enzyme(s) that were used, the sample barcodes and/or indexes, and any information on which the structure of the reads depends
- An archive comprising the 78-sample demonstration data set, including example 'barcodes' and 'population map' files, and demonstration shell scripts that follow the entire PROCEDURE, is available at http://catchenlab.life.illinois.edu/data/rochette2017_gac_or.tar.gz. This archive (except for the heavier lane-sequencing files) is also browsable at http://www.bitbucket.org/rochette/rad-seq-genotyping-demo/src/

Hardware and software

- Access to a computing cluster running under Linux, preferably with at least 8–16 cores and 64-Gb of memory
- Available disk space (depending on the analysis, several hundred gigabytes or more may be required)
- Generic scientific software, including R, Perl, and a recent C++ compiler
- SAMtools suite[29], available at http://www.htslib.org/
- BWA short-read aligner[34], available at http://bio-bwa.sourceforge.net/
- Stacks (v1.45 or later)[10,11], available at http://catchenlab.life.illinois.edu/stacks/
- ADEgenet R package[33], available from the CRAN repositories or at http://adegenet.r-forge.r-project.org/ (for PCA)

PROCEDURE

▲ **CRITICAL** We follow the convention that lines of code preceded by a percent sign (`%`) are shell commands that may be entered at the prompt (excluding the percent sign). In addition, a backslash ('\') indicates that a command continues to the next line (and should be omitted unless you are also continuing a command on the next line).

Preparing the input files ● TIMING 1 h

1| (On your cluster) Create a directory for the analysis. In principle, a meaningful name should be employed (for instance, `gac_or/` can stand for *G. aculeatus*, Oregon), but here we will call this directory `top/` so as to remain generic. Then create subdirectories to obtain the following hierarchy:

```
top/
    alignments/
    cleaned/
    genome/
    info/
    raw/
    stacks.denovo/
    stacks.ref/
    tests.denovo/
    tests.ref/
```

2| For each sequencing lane, create a subdirectory under the `raw/` directory. Place the read files of each lane in the corresponding subdirectory. Most commonly these files will have a `.fq.gz` extension (FASTQ format and gzip-compressed).

Our example data set comprises three lanes, and we thus start with three subdirectories in `raw/` (`lane1/`, `lane2/`, and `lane3/`), that each contain several read files.

3| For each lane, prepare the barcodes file—a file containing the list of samples present in the lane, along with their inline barcodes or indexes. These files should be written in tab-delimited text format (this format is also known as tab-separated values, or TSV, and is supported by all major spreadsheet applications). For data sets using a single barcode or index per sample, the actual format is:

```
<barcode>TAB<sample name>

<barcode>TAB<sample name>

(and so on)
```

As a general rule, we recommend that all text fields start with a letter and contain only letters, numbers, underscore ('_'), dot ('.'), or dash ('-') characters, and strictly exclude spaces. Thus, the barcode file for the first lane (`barcodes.lane1.tsv`) starts with:

```
CTCGCC sj_1819.35

GACTCT sj_1819.31

(and so on for 22 other samples in lane1)
```

Place all three barcodes files in the `info/` directory.
▲ **CRITICAL STEP** If the lanes have already been demultiplexed (your sequencing center returned per-sample read files), this step can be ignored.
? TROUBLESHOOTING

4| Prepare a second tab-delimited file—the population map, `popmap.tsv`, also in the `info/` directory, describing the populations to which the samples belong. The format is:

```
<sample name>TAB<population name>

<sample name>TAB<population name>

and so on
```

In our case, the population map contains:

```
cs_1335.01 cs

cs_1335.02 cs

(and so on for 76 samples in four populations)
```

Where '`cs_1335.01`' is the name of our first sample, and '`cs`' (Cushman Slough) the population it belongs to. Note that we only include the sample prefixes in both the barcodes file and the population map, we do not include any filename suffixes such as `.fq` or `.gz`.
▲ **CRITICAL STEP** The sample names in the barcodes and population map files should be identical.
? TROUBLESHOOTING

Visually assessing the sequencing data ● TIMING 30 min

5| Write down the read structure that your DNA library construction protocol should have led to. In particular, indicate the sequence that should be left from the restriction enzyme cut site and the presence (if any) and length(s) of inline barcodes. Illumina indexes are not part of the read itself—instead, they will be reported, for each read, at the end of the FASTQ header.

In our case, we expect all reads to start with a 6-nt barcode, followed by the six nucleotides (TGCAGG) that should remain of the SbfI restriction site (CCTGCAGG) in the DNA fragments after ligation, and the total read length should be 101. Thus, the expected read structure is:

```
<6-nt barcode>TGCAGG<unique 89-nt sequence>
```

6| Then, inspect a few reads in any one of the FASTQ raw sequence files to confirm that the data matches these expectations. Assuming the FASTQ files are compressed, this can be done with:

```
% zcat <file name>.fq.gz | head -n 20
```

For instance, the first read in our second FASTQ file (our first FASTQ file comprised all the low-quality, ambiguous reads) was:

```
GAGAGATGCAGGCCAGAACCGACAC...
```

This matches the expectation above (`GAGAGA` being a valid barcode in this lane).
▲ **CRITICAL STEP** If the lanes have already been demultiplexed, the barcodes should already have been removed.

Cleaning and demultiplexing the reads ● TIMING 1 d

7| Move into the `cleaned/` directory.

```
% cd cleaned/
```

8| For each lane, call the `process_radtags` unit of Stacks: use the corresponding lane directory and barcodes file (e.g., `../raw/lane1` and `../info/lane1.barcodes.tsv`) as input, and let the output directory be the current directory (`./`); provide the restriction enzyme (SbfI) and the barcoding strategy (in our case `--inline_null`, as we use a single inline barcode); finally, specify the operations to perform: filter ambiguous and low-quality reads (`-c` and `-q`), but rescue mutated barcodes when possible (`-r`). Thus, the command line is:

```
% raw_dir=../raw/lane1/

% barcodes_file=../info/barcodes.lane1.tsv

% process_radtags -p $raw_dir -b $barcodes_file -o ./ \

      -e sbfI --inline_null -c -q -r &> process_radtags.lane1.oe
```

▲ **CRITICAL STEP** Whenever running a program, it is important to save the 'standard output' and 'standard error' logs of the program in a file (using the `>`, `2>`, or `&>` shell operators). Throughout the protocol we save both of these logs in a single file, for which we use the `.oe` suffix (for 'output and error').
▲ **CRITICAL STEP** Do not decompress your raw files. These files are very large and we can save substantial space by keeping them compressed (the Stacks pipeline can work directly with compressed data).
▲ **CRITICAL STEP** This is the first step that is slow and computationally intensive (see TIMING section). In the present case, it is because of the amount of data that must be read from the disk and decompressed. Use your cluster's batch job submission system, if necessary (**Box 4**), and make sure, by inspecting the log files, that all lanes have been processed entirely.
▲ **CRITICAL STEP** If the lanes have already been demultiplexed, remove the barcodes-related arguments from the `process_radtags` command.
? TROUBLESHOOTING

9| For each lane, in the `.log` text file that `process_radtags` will have produced, check the overall proportion of retained reads and the main reasons why reads were discarded (no valid barcode, absence of restriction site, or low quality). Abnormally low numbers may indicate a mismatch between the actual read structure and the one that was specified (e.g., the wrong barcode type or restriction enzyme was supplied), or a problem with the sequencing itself.

In the same files, check the per-sample numbers of retained reads; these numbers indicate the relative coverage of the samples. It is practical to extract the relevant tables (one per lane) to a separate TSV file, or to copy/paste them into a spreadsheet application. Note the sample with the most reads (across all lanes) and the one with the least reads. Plotting the entire distribution is also useful (**Supplementary Fig. 1**). Abnormal numbers can stem from typos in the barcodes files or from issues with the multiplexed DNA library.

In our case, two samples ('sj_1483.05' and 'sj_1819.31') appear to have almost no reads (respectively, only 553 and 522), probably because they were not multiplexed properly during library construction. We therefore exclude them from

the analysis—simply by removing them from the primary population map `info/popmap.tsv` (or commenting them out by preceding their lines in the map file with a '#' character).
▲ CRITICAL STEP If the lanes have already been demultiplexed, the number of reads per sample is given by the per-input-file statistics instead of the per-barcode statistics at this step.

10| If your molecular protocol allows it, assess the number of PCR duplicates in the raw reads. This step will inform as to how many template molecules were in the unamplified molecular library versus the number of duplicates of those templates created during amplification. If your data set has an abnormally high number of duplicates (e.g., >50%), you can filter them out. For this operation to be possible, the data must either include random oligonucleotides (added for the specific purpose of filtering out PCR duplicates, such as with the RAPTURE protocol[6] or the RADCAP protocol[35]), or be derived from the shearing-based[3] protocol and paired-end sequenced. As this step is not possible for our demonstration data set, we will not detail it, but we recommend that one perform it using the '`clone_filter`' unit of Stacks, for which the command syntax is very similar to that of '`process_radtags`'.

11| If you are working with paired-end sequencing data, concatenate, for each sample, the four files of reads produced by `process_radtags` (`<sample name>.1.fq.gz, <sample name>.rem.1.fq.gz, <sample name>.2.fq.gz,` and `<sample name>.rem.2.fq.gz`).

```
% cp sample_name.1.fq.gz sample_name.fq.gz

% cat sample_name.2.fq.gz > sample_name.fq.gz

% cat sample_name.rem.1.fq.gz > sample_name.fq.gz

% cat sample_name.rem.2.fq.gz > sample_name.fq.gz
```

Creation of a reference genome database (reference-based analysis only) ● TIMING 1 h
12| Download the reference genome sequence of your organism. This is a FASTA (`.fa` or `.fa.gz`) file that can be obtained from the sequence or genome databases. In our case, we download the *G. aculeatus* reference genome from the Ensembl FTP site; the name of the file is `Gasterosteus_aculeatus.BROADS1.dna.toplevel.fa`, and for tractability we do not change it. We place this file in the `genome/` directory.

13| Create a BWA genome database:

```
% cd genome/

% mkdir bwa

% genome_fa=Gasterosteus_aculeatus.BROADS1.dna.toplevel.fa

% bwa index -p bwa/gac $genome_fa &> bwa/bwa_index.oe
```

If you are considering testing several alignment programs, create a database for each of them (each program works with its own, specific database files).

Working on a subset of samples for parameter testing ● TIMING 2 d
14| Choose a subset of a dozen representative samples, based on the read coverage numbers obtained above (Step 9) and the expected genetic similarities among samples (e.g., populations, if any). Write this list in a text file `popmap.test_samples.tsv`, formatted like `popmap.tsv` in Step 4, and place this file in the `info/` directory.

15| To apply the pipeline to this subset of samples to survey the properties of the data set and decide on an adequate parameter combination, follow option A if you are performing a *de novo* analysis or option B if you are performing a reference-based analysis.
(A) ***De novo* analysis**
(i) Decide on a series of combinations of *de novo* RAD locus assembly parameters to examine. The main parameters to con sider are *M*, *n*, and *m* (see **Box 2** for a discussion of these parameters and advice on how to choose reasonable values). We recommend investigating a range of parameter values, and a good point to start is a range of *M* and *n* values from 1 to 9 (fixing $M = n$) and $m = 3$.

(ii) Move into the `tests.denovo/` directory.

```
% cd tests.denovo/
```

(iii) For each parameter combination, create a subdirectory. Use tractable names; we call our directories, e.g., `stacks.M1/`, `stacks.M2/`, and so on.

(iv) For each parameter combination, run the Stacks *de novo* genotyping pipeline. The pipeline is most easily run using the `denovo_map.pl` wrapper, that will consecutively run the ustacks, cstacks, sstacks, and populations units.

```
% popmap=../info/popmap.test_samples.tsv
% reads_dir=../cleaned
% M=1
% out_dir=stacks.M$M
% log_file=$out_dir/denovo_map.oe
% denovo_map.pl --samples $reads_dir
  -O $popmap -o $out_dir \
-M $M -n $M -m 3 -b 1 -S &> $log_file
```

The wrapper will likely take several hours to complete; use your job submission system if you are working in a cluster environment to submit all pipeline runs at once (**Box 4**). The `-T` option can be used to make the pipeline run faster if multiple processors are available.

(v) Make sure that the wrapper has completed successfully for each parameter combination. A `denovo_map.log` text log file should have been created in the output directory. Check that it does not mention any error. The following command is a generic command to find errors (if any) in log files:

```
% grep -iE "\b(err|e:|warn|w:|fail|a
bort)" $log_file
```

? TROUBLESHOOTING

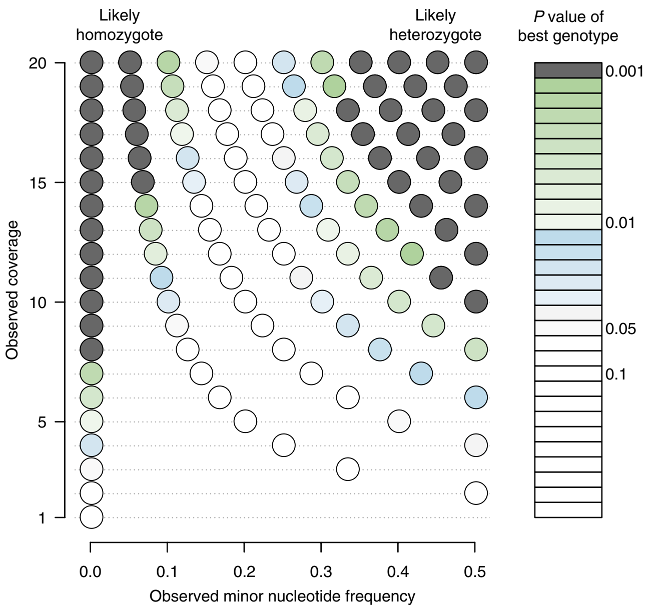

Figure 1 | Genotype calling greatly depends on coverage. The figure illustrates statistical certainty associated with genotype calls at different coverages (*y* axis). Each dot corresponds to a set of nucleotide observations that the genotype caller may face for a given SNP and given sample, and the quality of the associated call is indicated by color. *P* values are based on the multinomial model with maximum likelihood error rate estimation of ref. 32, assuming two possible alleles. Note that due to the randomness of the sequencing process, the two alleles of a heterozygote will most often not be sampled equally, even in the absence of error; consequently, for coverages <10×, heterozygotes cannot be called consistently.

(vi) Note the per-sample coverage values for $M = 1$. These values are reported in the same log file as above. They provide a lower bound for coverage—at $M = 1$, coverage is underestimated because many heterozygous loci will be counted as two separate loci and the number of loci is artifactually inflated.

▲ **CRITICAL STEP** Coverage strongly affects the quality of genotype calls (see INTRODUCTION and **Fig. 1**). In low-coverage data sets, the missing data and error rates can increase greatly; the experimenter should also be particularly cautious in his/her choice of model and filters for genotype calls. In the rest of the protocol we assume that the coverage is at least 10×. If your coverage is <10× for multiple samples, genotypes may be unreliable and can remain biased even after filtering.

(vii) For each parameter combination, filter the raw results of the pipeline using the populations unit, keeping only loci shared by at least 80% of samples (`-r 0.80`).

```
% M=1
% stacks_dir=stacks.M$M
% out_dir=$stacks_dir/populations.r80
% log_file=$out_dir/populations.oe
% populations -P $stacks_dir -O $out_dir -r 0.80 &> $log_file
```

(viii) By comparing the results obtained with different parameter combinations, decide on the one that will be applied to the full data set and be relied on for downstream analyses. Often, however, it is not possible to argue in an absolute way that a particular parameter combination is optimal. The goal is then to learn enough about the data set to be able to make a reasonable, informed choice.

From our set of test runs, we will plot two statistics that are useful in making this choice: the number of polymorphic loci shared across most samples (the *r80* loci) and the distribution of the number of SNPs per locus (**Fig. 2**). Both of these statistics are reported in the `batch_1.populations.log` file (in the `populations.r80/` directory) created by the populations unit. Considering that, for our data set, the number of widely shared loci plateaus starting

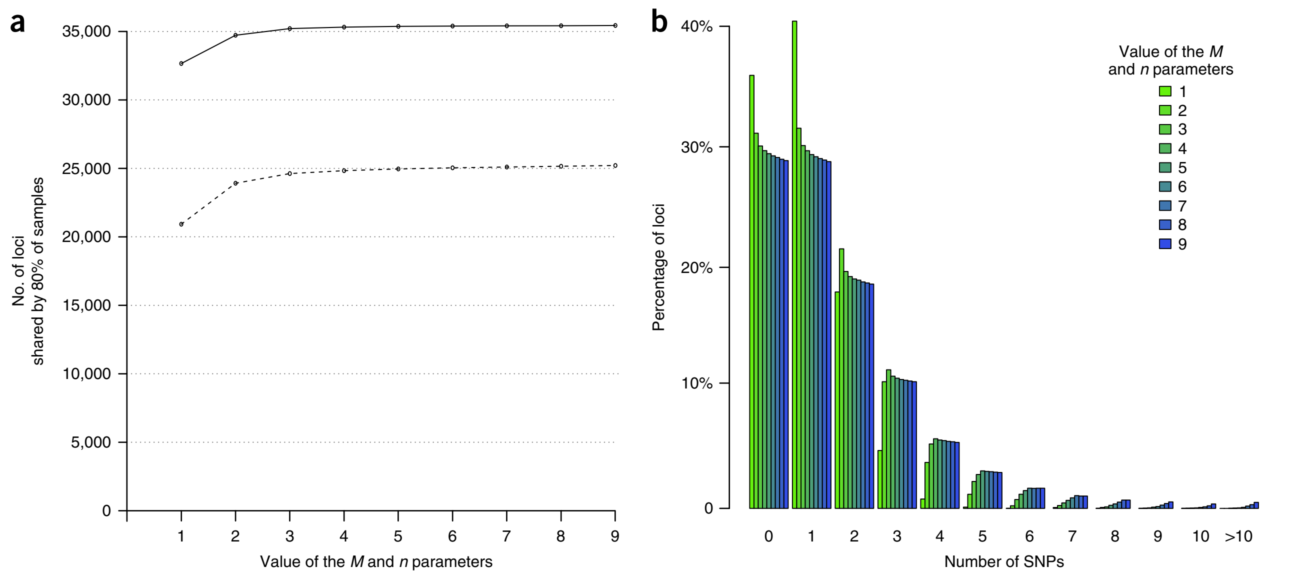

Figure 2 | Selection of assembly parameters in a *de novo* analysis. (**a**,**b**) Increasing the M and n assembly parameters affects the number of loci (solid line) and polymorphic loci (dashed line) shared by 80% of samples or more (**a**) and the distribution of the number of SNPs per locus (**b**). In both plots, the M and n parameters are kept equal, and the m parameter is fixed to 3. Note that for M and n values >3, the number of loci plateaus and the distribution is mostly unaffected—for instance the proportion of loci with two SNPs is 20% for $M = 3$ and 19% for $M = 9$. Nevertheless, with high values of M and n, a few loci with a high number of SNPs are assembled. Whether these highly divergent loci are artifactual or correspond to a real biological signal, such as long-term balancing selection, is an open question.

at about $M = 4$ (**Fig. 2a**), and that $M = 4$ is sufficient to stabilize the proportions of loci with 1–5 SNPs (**Fig. 2b**), we retained $M = 4$ for the main analysis. We recommend creating this plot with your own data using R or another plotting program (our plotting scripts are available along with our source scripts). If the *de novo* analysis is performed for comparison with a reference-based one, it is also useful to compare the two above statistics with the values derived from the reference-based analysis (**Supplementary Fig. 2**).

(B) Reference-based analysis

(i) Choose a putatively appropriate alignment method (see **Box 2** for further discussion on this topic). We start with BWA and default parameter values, but other short-read aligners or parameters are possible. Provided the reference genome is close to the considered data set, many parameter combinations will often yield equally good results, so testing alternative parameter combinations may not be necessary if the initial parameters lead to satisfactory results (Step 15B(ix–xii)).

(ii) Move into the `tests.ref/` directory:

```
% cd tests.ref/
```

(iii) Create two subdirectories: one to contain alignment files and the other one for Stacks files. Use names that indicate the parameters to which the directories correspond—in this case the BWA aligner and default parameters. Our test directory tree is then:

```
tests.ref/
alignments.bwa/
stacks.bwa/
```

(iv) For each sample of the subset, align the cleaned reads to the reference genome. The basic procedure is:

```
% sample=cs_1335.01
% fq_file=./cleaned/$sample.fq.gz
% bam_file=alignments.bwa/$sample.bam
% bwa_db=../genome/bwa/gac
% bwa mem -M $bwa_db $fq_file | samtools view -b > $bam_file
```

In the above command line, we align the reads to the reference genome using BWA (using `-M` for a simplified output), then we use the SAMtools suite to convert the output of BWA, which is in SAM format, to the equivalent, compressed BAM format.

(v) Perform quick checks to assess whether all the alignments have been completed successfully. Typically, this involves (i) checking that the expected number of output (BAM) files exist and are not empty, (ii) browsing through one or a few of the log files using a text editor/viewer, looking for errors, and (iii) systematically searching for potential errors or warnings in the log files using a command such as the one given previously (Step 15A(v)).

? TROUBLESHOOTING

(vi) Move into the Stacks directory.

```
% cd stacks.bwa
```

(vii) Run the pstacks unit of Stacks on each sample of the subset. Provide a unique identifier for each sample (e.g., set the sample index to '1' for the first sample file, '2' for the second, and so on).

```
% sample=cs_1335.01
% sample_index=1
% bam_file=../alignments.bwa/$sample.bam
% log_file=$sample.pstacks.oe
% pstacks -f $bam_file -i $sample_index -o ./ &> $log_file
```

(viii) Assess the successful completion of these jobs as described in Step 15B(v). pstacks should have created four files for each sample (`$sample.tags.tsv.gz`, `$sample.models.tsv.gz`, `$sample.snps.tsv.gz`, and `$sample.alleles.tsv.gz`), in addition to the log file.

(ix) From the log files of pstacks, extract, for all samples, the number of primary alignments (i.e., reads) that were kept for the analysis. Dividing this number by the initial number of reads of the sample (as reported by `process_radtags`, Step 9) gives the percentage of reads that remain after alignment and filtering. This percentage is one of the main statistics to consider in assessing the effectiveness of an alignment protocol. In our case, for 12 test samples, the mean percentage of used reads is 83.9% (min. 82.4%; max. 85.0%).

▲ CRITICAL STEP Percentages of aligned reads under 50% should be a trigger to reconsider the alignment protocol. Such low numbers may be caused, notably, by a misuse of the alignment program, a poor choice of parameters, or a very incomplete and/or distant reference genome.

(x) Also in the log files of pstacks, check, for all samples, the proportion of aligned reads that are discarded because of soft-clipping—in our case, on average, this is 2.2% (min. 2.0%; max. 2.4%).

▲ CRITICAL STEP If the value is high, it is likely that the reads just do not align properly to the reference, and the alignment protocol should be reconsidered.

(xi) Still in the log files of pstacks, check, for all samples, the average per-locus coverage. These numbers should provide an estimate of the actual sequencing coverage that was achieved. Per-locus coverage for our 12 test samples was 26.3 (min. 16.6; max. 59.1).

▲ CRITICAL STEP Coverage strongly affects the quality of genotype calls (see INTRODUCTION and **Fig. 1**). In low-coverage data sets, the missing data and error rates can increase greatly; in addition, the experimenter should be particularly cautious in his/her choice of model and filters for genotype calls. In the rest of the protocol, we assume that coverage is $\geq$10.

(xii) If the alignment results discussed above are satisfactory, make a note that they are and proceed with the analysis of the full data set. Otherwise, go back to Step 15B(i) and adjust the alignment procedure, or reconsider the reference-based approach.

Running the genotyping pipeline on the full data set ● TIMING 2 d

16| After a set of parameters has been chosen, move back to the top directory.

```
% cd top/
```

17| Apply the pipeline to the full data set. These steps are computationally intensive and should be performed on your computer cluster. Follow option A below if you are performing a *de novo* analysis or option B if you are performing a reference-based analysis.

(A) *De novo* analysis

(i) Move into the corresponding Stacks directory.

```
% cd stacks.denovo/
```

(ii) Run the ustacks unit on each sample separately. In principle, using the wrapper as in Step 15A(iv) above would be possible; however, in a cluster environment, we can schedule each ustacks run to a different node, saving time. In a single workstation environment, either method can be used. Pass the M and m parameter values chosen above to ustacks. Provide each sample with a unique identifier.

```
% sample=cs_1335.01
% sample_index=1
% fq_file=../cleaned/$sample.fq.gz
% log_file=$sample.ustacks.oe
```

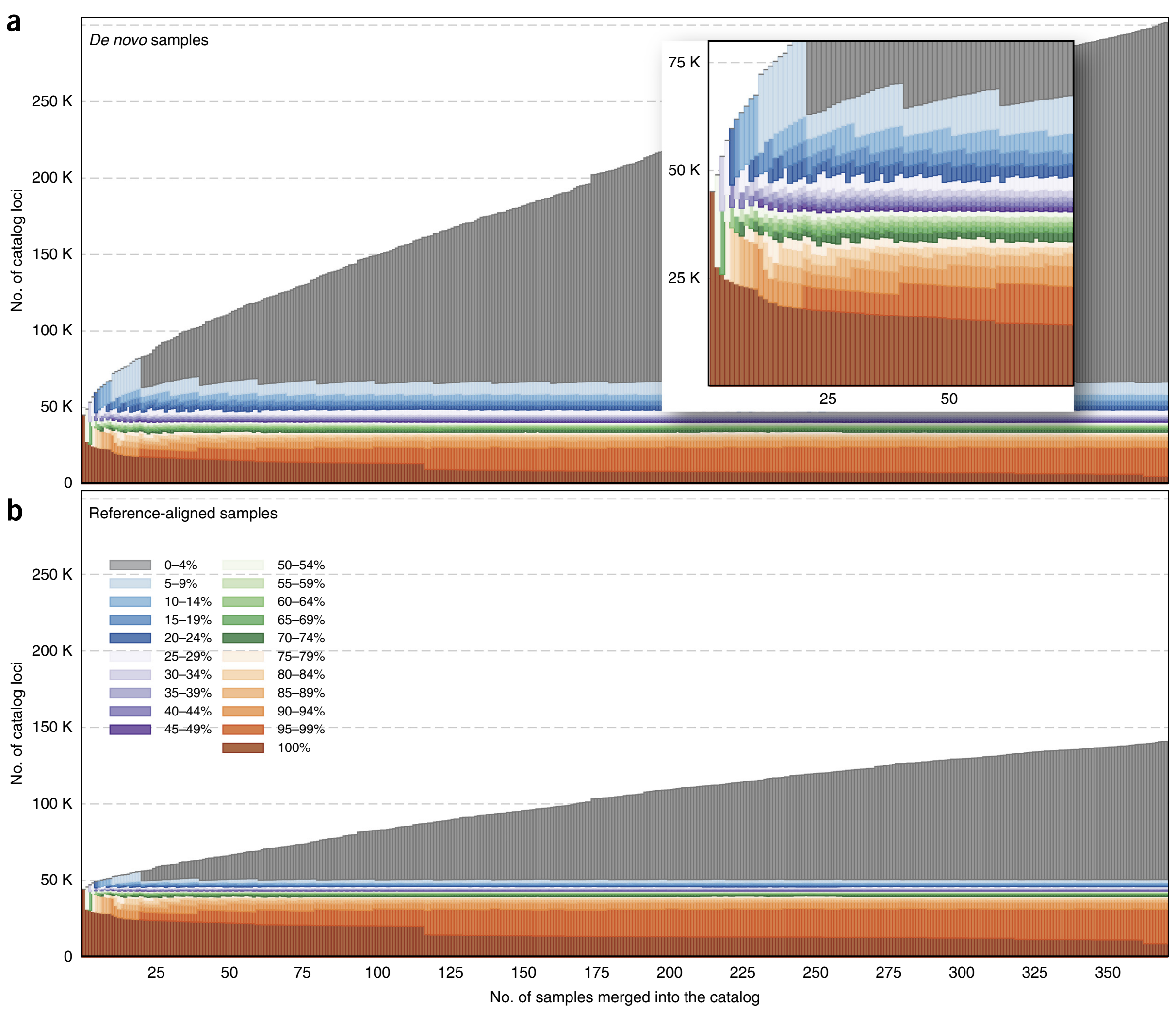

Figure 3 | Evolution of the catalog as more samples are added. (**a**,**b**) Both panels are bar plots of the number of loci in the catalog, broken up by the overall number of samples they can be found in, as a function of the number of samples that were used to build the catalog. (**a**) The *de novo* analysis. (**b**) The reference-based analysis. The number of loci in the catalog grows with the number of samples because each sample brings unique loci, but the core of widely shared loci remains stable. The inset shows the first 75 samples added to the catalog in the *de novo* case, for added detail.

```
% ustacks -f $fq_file -i $sample_index -o ./ -M 4 -m 3 &> $log_file
```

(iii) Make sure all jobs have completed successfully. Use the command described at Step 15A(v), and check that ustacks has created four files for each sample (`$sample.tags.tsv.gz`, `$sample.models.tsv.gz`, `$sample.snps.tsv.gz`, and `$sample.alleles.tsv.gz`) and that these files are not empty.

? TROUBLESHOOTING

(iv) From the log files of ustacks, extract the per-locus coverage of all samples. These values should be in line with their respective read coverages (Step 9) and with the average coverage found in the representative test subset.

(v) Select a list of ~40–200 samples to be included in the catalog. Pick samples that have high coverages and are representative of the genetic diversity in the data set (alleles that are not observed in the selected samples will subsequently be ignored). As with the test samples (Step 14), write this list as a population map `popmap.catalog.tsv` and place it in the `info/` directory. Including all samples in the catalog adds noise to it (**Fig. 3**), increases the computation time, and is unnecessary if low-minor allele frequency (MAF) variants are not of primary interest.
For our data set, we picked the ten samples with the highest coverage in each population (ignoring one sample with extremely high coverage), for a total of 40 samples.

(vi) Run the cstacks unit to create the catalog. Pass the `-n` parameter value chosen above. In the `stacks.denovo/` directory:

```
% cstacks -P ./ -M ../info/popmap.catalog.tsv -n 4 &> cstacks.oe
```

If several processors are available, the -p option can be used to improve the runtime. Make sure the program has completed.

(vii) Run the sstacks unit on each sample separately to match samples to the catalog:

```
% sample=cs_1335.01
% log_file=$sample.sstacks.oe
% sstacks -c ./batch_1 -s ./$sample -o ./ &> $log_file
```

On a cluster, each sstacks run can be scheduled to run on a different node; if running on a single machine instead of a cluster, all samples can be passed together, again using the population map:

```
% sstacks -c ./batch_1 -M ../info/popmap.catalog.tsv &> sstacks.oe
```

Make sure all jobs have completed.

(B) Reference-based analysis

(i) For each sample, align the cleaned reads to the reference genome. Use the alignment procedure that was chosen at Step 15B(xii). The commands are the same as those of Step 15B(iv), except for the file paths. Place these alignments in the top-level `alignments/` directory.
Although the samples of the test subset already have alignments, the simplest approach is to just compute those few alignments again, so as to not treat them differently from the other samples.

(ii) Perform quick checks, in the same way as at Step 15B(v), to assess whether all alignments have completed successfully.
? TROUBLESHOOTING

(iii) Move into the corresponding Stacks directory.

```
% cd stacks.ref/
```

(iv) For each sample, run the pstacks unit of Stacks in the same manner as at Step 15B(vii).

(v) Assess the successful completion of these jobs in the same way as at Step 15A(v).

(vi) Run the cstacks unit to create the catalog, specifying the `--aligned` option and the path to the list of all samples. In the `stacks.ref/` directory:

```
% cstacks --aligned -P ./ -M ../info/popmap.tsv
```

If several processors are available, the -p option can be used to improve the runtime. Make sure the program has completed.

(vii) Run the sstacks unit on each sample separately to match the samples to the catalog:

```
% sample=cs_1335.01
% log_file=$sample.sstacks.oe
% sstacks --aligned -c ./batch_1 -s ./$sample -o ./ &> $log_file
```

Make sure all the jobs have completed.

Filtering inferred SNPs and haplotypes based on population-level data ● TIMING 1 d

18| Use the rxstacks unit to improve SNP genotype calls, to filter out unlikely haplotypic combinations, and to remove loci having high error rates. Run this program after the catalog has been created to obtain a population-wide view of the data set. For each individual sample, genotypes are partially recomputed, as the software now knows the alleles present at each site across the population. rxstacks uses this information to constrain the genotype caller, improving calls and providing new calls that were previously statistically insignificant. Loci that are found to be confounded in multiple samples (i.e., possessing low locus likelihoods) are filtered out along with unlikely haplotypes resulting from error. The rxstacks unit rewrites the files for each sample, and afterward we will create a new catalog, excluding the identified, troublesome loci and including the improved genotype calls.

Start this part by creating a new subdirectory under the `stacks.ref/` or `stacks.denovo/` directory (for the reference-based or *de novo* analysis, respectively) and moving into this directory. For instance:

```
% mkdir stacks.denovo/rxstacks

% cd stacks.denovo/rxstacks
```

19| Execute the rxstacks unit. In addition to improving SNP calls, we choose to remove unlikely haplotypes, as well as loci that are assembled differently in >10% of the samples (the last filter is applicable only to *de novo* analyses).

```
% rxstacks -P ../ -o ./ --prune_haplo --conf_lim 0.10 &> rxstacks.oe
```

For each sample, rxstacks will write the same files as pstacks and ustacks in the new subdirectory.

20| Run the cstacks and sstacks units again, this time in the `rxstacks/` subdirectory, using the same commands as at Step 17A(vi–vii) (for a *de novo* analysis) or Step 17B(vi–vii) (for a reference-based analysis).

Filtering genotypes and exporting the data ● TIMING 1 h

21| Use the populations unit of Stacks to filter and export the genotype calls from the internal Stacks format to common biological formats and compute various population statistics (**Box 3**). The basic filters are the missing data filters (options `-r` and `-p`, minimum per-population percentage of samples and minimum number of populations to keep a locus), the minimum MAF filter (`--min_maf`), and the maximum accepted heterozygosity (`--max_obs_het`) (**Box 3**). Export formats are specified with options such as `--vcf`, `--genepop`, and so on. For the example, we use:

```
% min_samples=0.80

% min_maf=0.05

% max_obs_het=0.70

% populations -P ./ -r $min_samples --min_maf $min_maf \

      --max_obs_het $max_obs_het --genepop &> populations.oe
```

As we do not specify populations (i.e., we do not specify a population map, which we would do with `-M`), the samples are treated as a single pool, so the `-p` option is unnecessary.

Plotting raw and smoothed F_{ST} values over one chromosome (applicable only to reference-based analyses) ● TIMING 1 h

22| Create a new population map containing the samples to include in the analysis and the group in which they should be included. For example, to compare our two oceanic populations ('cs' and 'sj') with the two freshwater populations ('pcr' and 'wc'), we write the following population map:

```
cs_1335.01 oceanic

cs_1335.02 oceanic

(...)

pcr_1193.00 freshwater

(...)

sj_1483.01 oceanic

(...)

wc_1537.09 freshwater

(...)
```

One way to obtain this result is:

```
% cd top/info

% sed -r 's/\t(cs|sj)/\toceanic/; s/\t(pcr|wc)/\tfreshwater/;' popmap.tsv \

      > popmap.oceanic_freshwater.tsv
```

23| Run populations in the `stacks.ref/rxstacks/` directory. We use the same choice of filters as at Step 21, ask or F_{ST} statistics to be computed (`--fstats`) and smoothed with a characteristic distance of 100 kb (`-k --sigma 100000`).

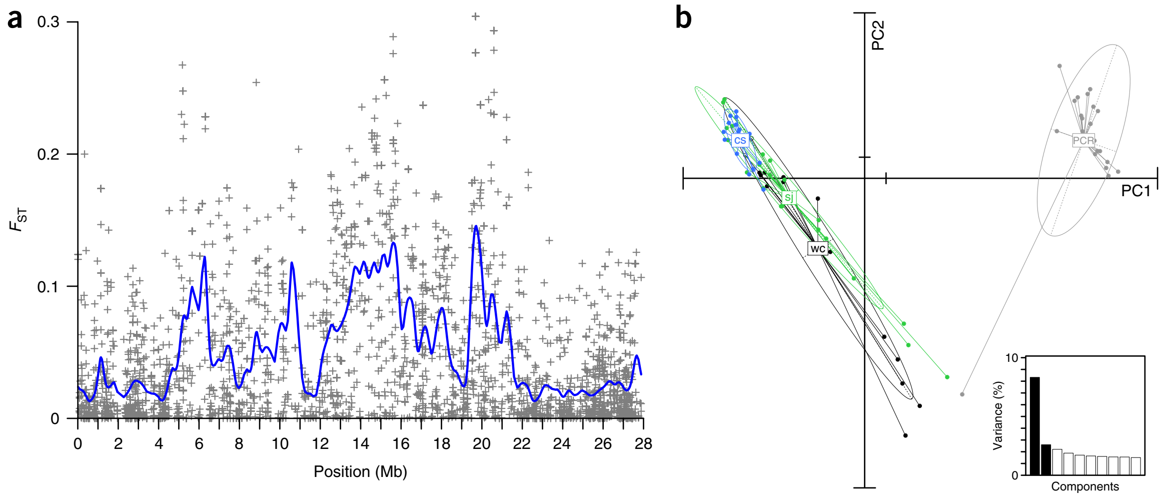

Figure 4 | Transitioning to populations genetics. (**a**) Distribution over *G. aculeatus* chromosome 7 of smoothed (blue curve) and raw (gray crosses) F_{ST} values. Values were computed using Stacks, then plotted with R (Steps 22–24). (**b**) PCA of 76 individuals in four populations based on SNPs present in >80% samples, computed using ADEgenet (Steps 25–28). Inset: percentage of the variance explained by the first ten components. The analysis separates the PCR ('Pony Creek Reservoir', freshwater, upstream) population from the WC ('Winchester Creek', freshwater, downstream), and the SJ and CS ('South Jetty' and 'Cushman Slough', oceanic) populations.

```
% cd stacks.ref/rxstacks

% popmap=../../info/popmap.oceanic_freshwater.tsv

% out_dir=pops.oceanic_freshwater

% log_file=pops.oceanic_freshwater.oe

% populations -P ./ -M $popmap -O $out_dir \
      -p 2 -r 0.80 --min_maf 0.05 --max_obs_het 0.70 \
      --fstats -k --sigma 100000 \
      &> $log_file
```

24| In R, load the contents of the output file containing the F_{ST} statistics (`batch_1.fst_oceanic-freshwater.tsv`) and plot the desired values.

```
#x = read.delim('pops.oceanic_freshwater/batch_1.fst_oceanic-freshwater.tsv')

#x.vii = subset(x, Chr=='groupVII')

#plot(x.vii$BP, x.iv$AMOVA.Fst, pch=3)

#lines(x.vii$BP, x.iv$Smoothed.AMOVA.Fst)
```

This creates a plot similar to **Figure 4a**.
? TROUBLESHOOTING

Performing a PCA using the R package ADEgenet ● TIMING 1 h
25| Export the genotypes in the GENEPOP format using populations. PCA can be performed after both the reference-based and *de novo* analyses—for this example, we use the results of the *de novo* analysis. In addition, we use the same SNP filters as at Step 21.

```
% cd stacks.denovo/rxstacks

% popmap=../../info/popmap.tsv

% out_dir=pops.r80-maf05-het70

% log_file=pops.r80-maf05-het70.oe
% populations -P ./ -M $popmap -O $out_dir \
-p 4 -r 0.80 --min_maf 0.05 --max_obs_het 0.70 \
--genepop \&> $log_file
```

26| Copy the GENEPOP file so that it has the `.gen` suffix that ADEgenet expects.

```
% cp pops.r80-maf05-het70/batch_1.genepop pops.r80-maf05-het70/batch_1.gen
```

27| In R, load the ADEgenet package, read the genotypes, and place samples in their respective populations.

```
# library(adegenet)

# x = read.genepop('pops.r80-maf05-het70/batch_1.gen')

# pop(x) = sapply(strsplit(indNames(x), '_'), function(x){x[1]})
```

28| Then, perform the PCA and plot the results.

```
# x.pca = dudi.pca(x, scannf=FALSE)

# s.class(x.pca$li, pop(x), col=rainbow(nPop(x)))

# add.scatter.eig(x.pca$eig[1:10], xax=1, yax=2)
```

This creates a plot similar to **Figure 4b**.
? TROUBLESHOOTING

? TROUBLESHOOTING

Troubleshooting advice can be found in **Table 1**.

TABLE 1 | Troubleshooting table.

Step	Problem	Possible reason	Solution
8	Samples are identified by multiple barcodes and/or indexes	Depends on experimental design	The online Stacks manual (http://catchenlab.life.illinois.edu/stacks/manual/) and the online documentation for the 'process_radtags' unit (http://catchenlab.life.illinois.edu/stacks/comp/process_radtags.php) explain how to write barcodes files and use 'process_radtags' for combinatorial barcoding strategies
3, 4	Problems with text files	The format is wrong; the file mixes tabs and spaces; the file contains invisible characters; the file is actually compressed	Use a specialized text file editor (such as Notepad++, GEdit, Emacs, or Nano). Make sure your files do not mix tabs and space characters. The carriage return character, present in text files created in Windows, will not create issues with the programs of the Stacks suite. As a last resort, if you suspect that a file or a copy/pasted piece of text might contain unwanted invisible characters, the shell command 'cat $file | od -c' is useful to display every single character (printable or not)
All	Program errors	Variable	Read the log files and pay attention to the explanations that accompany the error. Stay open-minded, and progressively narrow down the cause. Eliminate causes that are the easiest to test first

(continued)

TABLE 1 | Troubleshooting table (continued).

Step	Problem	Possible reason	Solution
All	'File cannot be opened'	There is a typo in the command line; an earlier step was skipped or failed	For files that should be read (input files), this error means that the file does not exist at the path that was specified. Common causes are (i) there is a typo in the file name, (ii) the file is actually in a directory other than the one that was given, and (iii), for relative paths, the program is not called from the right directory. If the file actually does not exist, it is likely that an earlier step failed or was skipped. For files that should be written (output files), this error means that the directory where the file was supposed to be created does not exist, or that you do not have the 'write permission' in this directory
15A(v), 15B(v), 17A(iii), 17B(ii)	Jobs not completed successfully, and you must re-run a program for specific samples only	Often necessary at some point in analyses with many samples	A practical solution is to remove the results files (if any) for the failed samples, then perform a 'for' loop on all samples, conditioning the actual processing on the inexistence of the results file (i.e., 'for sample in <all samples>; do results_file=<expected path>; if [[! -e "$results_file"]]; then <run programs>; fi; done')
24, 27	No plot is produced	Connection to the cluster is not configured to allow graphics	Plotting interactively requires a graphical session. If graphics are not available, R will create a 'Rplots.pdf' file instead of opening a new window. If your computer and cluster support it, using the 'ssh -X' option will enable graphics. Alternatively, you can download the PDF file to your computer

● TIMING

The overall times listed below include computation times and human labor for script writing and validation (details on computation times are given below). All estimates must be adjusted for the size of the data set and increased relative to (inversely proportional to) the user's proficiency in bioinformatics and cluster computing.
Steps 1–4, preparing the input files: 1 h
Steps 5 and 6, visually assessing the sequencing data: 30 min
Steps 7–11, cleaning and demultiplexing the reads: 1 d
Steps 12 and 13, creating a reference genome database (reference-based analysis only): 1 h
Steps 14 and 15, working on a subset of samples for parameter testing: 2 d
Steps 16 and 17, running the genotyping pipeline on the full data set: 2 d
Steps 18–20, improving SNP and haplotypes inferences based on population-level data: 1 d
Step 21, filtering genotypes and exporting the data: 1 h
Steps 22–24, plotting raw and smoothed F_{ST} values over one chromosome (applicable only to reference-based analyses): 1 h
Steps 25–28, performing a PCA using the R package ADEgenet: 1 h
The computational times reported below correspond to a data set of 560 M raw reads (57 Gbp of sequence) covering 382 samples[12]. This amounts to slightly less than one current single-end Illumina HiSeq 4000 lane. Steps that do not include computations, or for which computations are fast, are not reported. An asterisk ('*') indicates that the computation was run using 16 CPUs. Times will vary depending on data set size.
Step 8, cleaning and demultiplexing the reads: 10 h
Step 13, creating a BWA reference genome database (for a 460 Mb genome): 15 min
Step 15A, assessing parameters for a *de novo* analysis: 5 h*
Step 15B, assessing parameters for a reference-based analysis: 15 min*
Step 17A(ii), running 'ustacks' on all samples: 4 h*
Step 17A(vi), running 'cstacks' (using 100 samples): 1 h*
Step 17A(vii), running 'sstacks' on all samples: 15 min*
Step 17B(i), aligning all samples using BWA: 3 h*
Step 17B(iv), running 'pstacks' for all samples: 30 min*

Step 17B(vi), running 'cstacks' for all samples: 2 h*
Step 17B(vii), running 'sstacks' for all samples: 15 min*
Step 19, running 'rxstacks': 4h*
Step 20, running 'cstacks' and 'sstacks': 3 h 15 min*
Step 21, filtering and exporting the data: 15 min
Step 23, computing and smoothing F_{ST}: 1 h*
Step 25, exporting the data: 15 min
Steps 27 and 28, loading the data in ADEgenet and computing a PCA: 15 min

ANTICIPATED RESULTS

Upon completion of the protocol, the experimenter should have obtained per-sample genotype data in the form of files in a format suitable for genetic analyses, such as VCF, Genepop, and/or Structure (as specified by the user). However, before moving on to biological analyses, it is advisable to take a step back and interrogate the results as a whole. There is no universal way to guarantee that inferences have not been affected by unexpected artifacts, especially as many expectations are contingent on the specific biological system in which the analysis is performed. Nevertheless, there are general checks that should be applicable in most cases.

First of all, the order of magnitude of the total number of loci that were identified should match the expectations based on the genome size and composition, and the choices of restriction enzyme and molecular protocol[36]. Furthermore, and especially for a *de novo* analysis, the overlap between the loci found in different samples should be checked. A core set of loci shared by most samples and present in all populations should exist. More generally, the rates of missing data should be within reasonable bounds. Coverage should also be sufficiently high. We have explained before how low-coverage data requires special handling. These issues have been discussed elsewhere[17], but here we instead would like to caution our readers to ensure that they will get adequate coverage when they design their RAD-seq libraries. Finally, checks can be performed using the genotype data itself. Such tests include tracking the number of markers with implausibly high observed heterozygosities, checking that the value of the transition/transversion ratio is not strongly distorted[37], or checking that nucleotide diversity is plausible with regard to the system's biology[38].

Eventually, the experimenter will settle on one methodology. The final results of the genotyping part of the analysis take the form of a single directory containing the genotypes called for all samples in the Stacks internal format. These results can then be exported to standard formats using the 'populations' unit for direct analysis in biological formats or for further processing. Multiple such exports will typically be performed to account for requirements of the target biological analysis programs (constraints may exist on file format, data set size, or presence of missing data), to study specific sample groups, or to compare biological results obtained using stringent or more relaxed SNP and genotype filters.

Note: Any Supplementary Information and Source Data files are available in the online version of the paper.

ACKNOWLEDGMENTS We thank J. Paris and N. Rayamajhi for their help in testing the procedure and for discussion of the manuscript.

AUTHOR CONTRIBUTIONS N.C.R. and J.M.C. designed the protocol, performed experiments, and wrote the manuscript.

COMPETING FINANCIAL INTERESTS The authors declare no competing financial interests.

1. Narum, S.R., Buerkle, C.A., Davey, J.W., Miller, M.R. & Hohenlohe, P.A. Genotyping-by-sequencing in ecological and conservation genomics. *Mol. Ecol.* **22**, 2841–2847 (2013).
2. Andrews, K.R., Good, J.M., Miller, M.R., Luikart, G. & Hohenlohe, P.A. Harnessing the power of RADseq for ecological and evolutionary genomics. *Nat. Rev. Genet.* **17**, 81–92 (2016).
3. Baird, N.A. *et al.* Rapid SNP discovery and genetic mapping using sequenced RAD markers. *PLoS One* **3**, e3376 (2008).
4. Elshire, R.J. *et al.* A robust, simple genotyping-by-sequencing (GBS) approach for high diversity species. *PLoS One* **6**, e19379 (2011).
5. Peterson, B.K., Weber, J.N., Kay, E.H., Fisher, H.S. & Hoekstra, H.E. Double digest RADseq: an inexpensive method for *de novo* SNP discovery and genotyping in model and non-model species. *PLoS One* **7**, e37135 (2012).
6. Ali, O.A. *et al.* RAD capture (Rapture): flexible and efficient sequence-based genotyping. *Genetics* **202**, 389–400 (2016).
7. Toonen, R.J. *et al.* ezRAD: a simplified method for genomic genotyping in non-model organisms. *PeerJ* **1**, e203 (2013).
8. Franchini, P., Monné Parera, D., Kautt, A.F. & Meyer, A. quaddRAD: a new high-multiplexing and PCR duplicate removal ddRAD protocol produces novel evolutionary insights in a nonradiating cichlid lineage. *Mol. Ecol.* **26**, 2783–2795 (2017).
9. Suchan, T. *et al.* Hybridization capture using RAD probes (hyRAD), a new tool for performing genomic analyses on collection specimens. *PLoS One* **11**, e0151651 (2016).
10. Catchen, J.M., Amores, A., Hohenlohe, P., Cresko, W. & Postlethwait, J.H. Stacks: building and genotyping loci *de novo* from short-read sequences. *G3* **1**, 171–182 (2011).
11. Catchen, J., Hohenlohe, P.A., Bassham, S., Amores, A. & Cresko, W.A. Stacks: an analysis tool set for population genomics. *Mol. Ecol.* **22**, 3124–3140 (2013).
12. Catchen, J. *et al.* The population structure and recent colonization history of Oregon threespine stickleback determined using restriction-site associated DNA-sequencing. *Mol. Ecol.* **22**, 2864–2883 (2013).
13. Lescak, E.A. *et al.* Evolution of stickleback in 50 years on earthquake-uplifted islands. *Proc. Natl. Acad. Sci. USA* **112**, E7204–E7212 (2015).
14. Kautt, A.F., Machado-Schiaffino, G. & Meyer, A. Multispecies outcomes of sympatric speciation after admixture with the source population in two radiations of Nicaraguan Crater Lake cichlids. *PLoS Genet.* **12**, e1006157 (2016).
15. Malinsky, M. *et al.* Genomic islands of speciation separate cichlid ecomorphs in an East African crater lake. *Science* **350**, 1493–1498 (2015).
16. R Core Team. *R: A Language and Environment for Statistical Computing* (R Foundation for Statistical Computing, 2015).

17. Nielsen, R., Paul, J.S., Albrechtsen, A. & Song, Y.S. Genotype and SNP calling from next-generation sequencing data. *Nat. Rev. Genet.* **12**, 443–451 (2011).
18. Browning, S.R. & Browning, B.L. Haplotype phasing: existing methods and new developments. *Nat. Rev. Genet.* **12**, 703–714 (2011).
19. Korneliussen, T.S., Albrechtsen, A. & Nielsen, R. ANGSD: analysis of next generation sequencing data. *BMC Bioinformatics* **15**, 356 (2014).
20. Eaton, D.A.R. PyRAD: assembly of *de novo* RADseq loci for phylogenetic analyses. *Bioinforma. Oxf. Engl.* **30**, 1844–1849 (2014).
21. Edgar, R.C. Search and clustering orders of magnitude faster than BLAST. *Bioinforma. Oxf. Engl.* **26**, 2460–2461 (2010).
22. Sovic, M.G., Fries, A.C. & Gibbs, H.L. AftrRAD: a pipeline for accurate and efficient *de novo* assembly of RADseq data. *Mol. Ecol. Resour.* **15**, 1163–1171 (2015).
23. Huang, W., Umbach, D.M. & Li, L. Accurate anchoring alignment of divergent sequences. *Bioinforma. Oxf. Engl.* **22**, 29–34 (2006).
24. Glaubitz, J.C. *et al.* TASSEL-GBS: a high capacity genotyping by sequencing analysis pipeline. *PLoS One* **9**, e90346 (2014).
25. Lu, F. *et al.* Switchgrass genomic diversity, ploidy, and evolution: novel insights from a network-based SNP discovery protocol. *PLoS Genet.* **9**, e1003215 (2013).
26. Puritz, J.B., Hollenbeck, C.M. & Gold, J.R. dDocent: a RADseq, variant-calling pipeline designed for population genomics of non-model organisms. *PeerJ* **2**, e431 (2014).
27. Chong, Z., Ruan, J. & Wu, C.-I. Rainbow: an integrated tool for efficient clustering and assembling RAD-seq reads. *Bioinforma. Oxf. Engl.* **28**, 2732–2737 (2012).
28. Shafer, A.B.A. *et al.* Bioinformatic processing of RAD-seq data dramatically impacts downstream population genetic inference. *Methods Ecol. Evol.* http://dx.doi.org/10.1111/2041-210X.12700 (2016).
29. Li, H. A statistical framework for SNP calling, mutation discovery, association mapping and population genetical parameter estimation from sequencing data. *Bioinforma. Oxf. Engl.* **27**, 2987–2993 (2011).
30. McKenna, A. *et al.* The Genome Analysis Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data. *Genome Res.* **20**, 1297–1303 (2010).
31. Garrison, E. & Marth, G. Haplotype-based variant detection from short-read sequencing. Preprint at https://arxiv.org/abs/1207.3907 (2012).
32. Hohenlohe, P.A. *et al.* Population genomics of parallel adaptation in threespine stickleback using sequenced RAD tags. *PLoS Genet.* **6**, e1000862 (2010).
33. Jombart, T. & Ahmed, I. adegenet 1.3-1: new tools for the analysis of genome-wide SNP data. *Bioinforma. Oxf. Engl.* **27**, 3070–3071 (2011).
34. Li, H. & Durbin, R. Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinforma. Oxf. Engl.* **25**, 1754–1760 (2009).
35. Hoffberg, S.L. *et al.* RADcap: sequence capture of dual-digest RADseq libraries with identifiable duplicates and reduced missing data. *Mol. Ecol. Resour.* http://dx.doi.org/10.1111/1755-0998.12566 (2016).
36. Herrera, S., Reyes-Herrera, P.H. & Shank, T.M. Predicting RAD-seq marker numbers across the eukaryotic tree of life. *Genome Biol. Evol.* http://dx.doi.org/10.1093/gbe/evv210 (2015).
37. DePristo, M.A. *et al.* A framework for variation discovery and genotyping using next-generation DNA sequencing data. *Nat. Genet.* **43**, 491–498 (2011).
38. Romiguier, J. *et al.* Comparative population genomics in animals uncovers the determinants of genetic diversity. *Nature* **515**, 261–263 (2014).
39. Braasch, I. *et al.* A new model army: emerging fish models to study the genomics of vertebrate Evo-Devo. *J. Exp. Zool. B Mol. Dev. Evol.* **324**, 316–341 (2015).
40. Lien, S. *et al.* The Atlantic salmon genome provides insights into rediploidization. *Nature* **533**, 200–205 (2016).
41. Ilut, D.C., Nydam, M.L. & Hare, M.P. Defining loci in restriction-based reduced representation genomic data from nonmodel species: sources of bias and diagnostics for optimal clustering. *Biomed. Res. Int.* **2014**, 675158 (2014).
42. Harvey, M.G. *et al.* Similarity thresholds used in DNA sequence assembly from short reads can reduce the comparability of population histories across species. *PeerJ* **3**, e895 (2015).
43. Rodríguez-Ezpeleta, N. *et al.* Population structure of Atlantic mackerel inferred from RAD-seq-derived SNP markers: effects of sequence clustering parameters and hierarchical SNP selection. *Mol. Ecol. Resour.* **16**, 991–1001 (2016).
44. Paris, J.R., Stevens, J.R. & Catchen, J.M. Lost in parameter space: a road map for stacks. *Methods Ecol. Evol.* **8**, 1360–1373 (2017).
45. Weir, B.S. *Genetic Data Analysis II: Methods for Discrete Population Genetic Data* (Sinauer Associates, 1996).
46. Excoffier, L., Smouse, P.E. & Quattro, J.M. Analysis of molecular variance inferred from metric distances among DNA haplotypes: application to human mitochondrial DNA restriction data. *Genetics* **131**, 479–491 (1992).
47. Meirmans, P.G. Using the AMOVA framework to estimate a standardized genetic differentiation measure. *Evol. Int. J. Org. Evol.* **60**, 2399–2402 (2006).
48. Bird, C.E., Karl, S.A., Mouse, P.E. & Toonen, R.J. in *Phylogeography and Population Genetics in Crustacea* 31–55 (CRC Press, 2011).