Stacks: an analysis tool set for population genomics

JULIAN CATCHEN,* PAUL A. HOHENLOHE,*† SUSAN BASSHAM,* ANGEL AMORES‡ and WILLIAM A. CRESKO*

*Institute of Ecology and Evolution, University of Oregon, Eugene, OR 97403-5289, USA, †Biological Sciences, University of Idaho, Moscow, ID 83844-3051, USA, ‡Institute of Neuroscience, University of Oregon, Eugene, OR 97403-1254, USA

Abstract

Massively parallel short-read sequencing technologies, coupled with powerful software platforms, are enabling investigators to analyse tens of thousands of genetic markers. This wealth of data is rapidly expanding and allowing biological questions to be addressed with unprecedented scope and precision. The sizes of the data sets are now posing significant data processing and analysis challenges. Here we describe an extension of the Stacks software package to efficiently use genotype-by-sequencing data for studies of populations of organisms. Stacks now produces core population genomic summary statistics and SNP-by-SNP statistical tests. These statistics can be analysed across a reference genome using a smoothed sliding window. Stacks also now provides several output formats for several commonly used downstream analysis packages. The expanded population genomics functions in Stacks will make it a useful tool to harness the newest generation of massively parallel genotyping data for ecological and evolutionary genetics.

Keywords: GBS, genetics, next-generation sequencing, population genomics, RAD-seq

Received 19 November 2012; revision received 16 April 2013; accepted 16 April 2013

Introduction

The study of nearly complete genetic information in numerous individuals drawn from scores of populations is now rapidly becoming a reality (Storz 2005; Bonin 2008; Hohenlohe et al. 2010a, 2012a; Stapley et al. 2010). New molecular genetic techniques (Mardis 2008), enabled by massively parallel short-read sequencing technologies coupled with powerful software, have been critical to advances in this nascent field of population genomics. Investigators have employed these methods to move from painstakingly developing dozens of microsatellite markers to rapidly producing tens of thousands of single nucleotide polymorphism (SNP) markers (Davey et al. 2011; McCormack et al. 2013).

Several molecular approaches have been developed to focus the large number of short reads provided by modern sequencing platforms on specific, restriction enzyme–anchored positions in the genome (e.g. CRoPS, Van Orsouw et al. 2007; RAD-seq, Baird et al. 2008; Etter et al. 2011b; GBS, Elshire et al. 2011; double-digest RAD-seq, Peterson et al. 2012; and 2bRAD, Wang et al. 2012b). This family of reduced representation genotyping approaches, generically called genotype-by-sequencing (GBS) or restriction site-associated DNA sequencing (RAD-seq; Davey et al. 2011), subsamples the genome at homologous locations to identify and type SNPs evenly throughout the genome. Population genomics using GBS allows classic problems in ecological and evolutionary genetics, such as identification of parentage and relatedness, migration and gene flow, population structure and phylogeography, and phylogenetic reconstruction, to be addressed with unprecedented power and precision (Mitchell-Olds et al. 2008; Hohenlohe et al. 2010a; Stapley et al. 2010). More importantly, population genomic studies allow the simultaneous identification of a genome-wide average and outliers for any given statistic to help identify genomic regions contributing to local adaptation or even speciation (Lewontin & Krakauer 1973; Maynard Smith & Haigh 1974; Luikart et al. 2003; Beaumont & Balding 2004; Nielsen 2005; Storz 2005; Nielsen et al. 2007; Foll & Gaggiotti 2008; Gaggiotti et al. 2009; Hohenlohe et al. 2010b, 2012b; Strasburg et al. 2012).
The wealth of genetic data provided by massively parallel short-read sequencing brings serious challenges in data processing and analysis (Shendure & Ji 2008; Glenn 2011). Studies now commonly comprise billions of raw sequences used to genotype tens of thousands to millions of SNPs. The key to making such studies feasible is software that can efficiently assemble reads together, identify alleles and genotypes, and track those genotypes in hundreds of individuals in scores of populations using a statistically rigorous framework (Lynch 2009; Gompert et al. 2010; Hohenlohe et al. 2010b). To help minimize the challenges of using GBS methods for genetic studies, we developed Stacks (http://creskolab.uoregon.edu/stacks/), a computational pipeline designed to work with any restriction enzyme-based GBS data. Stacks is computationally robust, efficient and flexible and can assemble short reads de novo or use data aligned to a reference genome. The Stacks software can handle data from thousands of individuals and incorporates a MySQL database and web front end for efficient data visualization, management and modification. Stacks was initially designed for genetic mapping crosses (Catchen et al. 2011), and we have added significant functionality for ecological and evolutionary genomic analyses. Here, we describe and evaluate these new features of Stacks using RAD-seq data from Oregon threespine stickleback (Gasterosteus aculeatus) populations. A complete manual for Stacks is available (http://creskolab.uoregon.edu/stacks/stacks_manual.pdf), as are additional tutorials and other resources.

Experimental space and the central concept of Stacks

Analysing GBS data requires several steps such as acquiring raw sequence data, filtering out low-quality reads, assembling or aligning reads, and finally inferring SNPs and genotypes. Each step has its own associated challenges and uncertainties. These arise from genomic attributes such as the number of loci identified, the degree of repetitive sequences throughout the genome, and the level of polymorphism and divergence among populations. These biological factors also interact with sequencing characteristics such as the quality of DNA and degree of sample multiplexing, the total number and length of reads, and the sequencing error rate. Key decisions therefore need to be made at each step about such items as the required depth of coverage or allowable nucleotide distance between reads for assembly. Finally, because of biological and sequencing sampling variation, the use of statistical models will often be necessary.

We have built the Stacks software platform to be modular and tunable to facilitate iterative exploration of the biological and sequencing parameter space for a particular study and to easily acquire and incorporate additional data. At the core of Stacks is the catalogue – a collection of all the loci and alleles identified in a population of individuals. In a mapping cross, the catalogue is simple and contains only loci found in the parents, enabling the identification of parental alleles present in the progeny. In the more general case of a set of individuals from one or more populations, the catalogue grows more complex and can often contain many more loci and segregating alleles. If a reference genome is available, those loci can be ordered, allowing them to be compared along the genome. Stacks uses a relational database and a web-based user interface. This interface allows for data visualization and user-directed modifications and corrections to the genetic hypotheses. Below we describe some of the major steps, decision points, statistical considerations and ways to specify the major parameters for Stacks.

Major steps of a Stacks analysis

The raw input data to Stacks are sequenced DNA fragments from any restriction enzyme-based GBS protocol. These protocols provide reads that will be anchored to homologous locations in the genome, which then appear as well arranged ‘stacks’ when visualized (see Davey et al. 2011 for details). Stacks can handle raw sequencing data in FASTA or FASTQ format to identify loci de novo and reads aligned against a reference genome in SAM (Li et al. 2009) format. Aligned reads may be gapped to allow for indels. Regardless of whether the data are assembled de novo, or aligned against a reference genome, many subsequent steps in Stacks are shared.

Stacks is a collection of several original C++ programs and Perl scripts. The components of Stacks can be run individually by hand or using one of two provided wrapper programs that will execute the entire pipeline (denovo_map.pl or ref_map.pl).

The pipeline is outlined in Fig. 1 and can be described as follows:

1. Raw sequence reads are demultiplexed and cleaned (process_radtags).
2. Data from each individual are grouped into loci, and polymorphic nucleotide sites are identified (ustacks or pstacks for unaligned or aligned data, respectively).
3. Loci are grouped together across individuals and a catalogue is written (cstacks).
4. Loci from each individual are matched against the catalogue to determine the allelic state at each locus in each individual (sstacks).
Allelic states are either converted into a set of mappable genotypes (for a genetic map) using genotypes or subjected to population genetic statistics via populations, with the results being written in one or several useful output files.

As described previously in Catchen et al. (2011), a web-based front end, backed by a MySQL database, is available to visualize the data. Both denovo_map.pl and ref_map.pl will automatically populate a MySQL database during execution.

De novo stack formation

Stacks will, through the program ustacks, use a k-mer search algorithm to merge alleles into loci. First, exactly matching reads are formed into stacks using a hashing algorithm. Stacks are subsequently decomposed into k-mers (subsequences of length k) that are compared among stacks to find matching alleles (see Catchen et al. 2011 for more detail). In the previous version of Stacks, this process was controlled by two parameters. The stack depth parameter (-m) controls the number of raw
reads required to form a stack, and the mismatch parameter (~-M) specifies the number of allowed nucleotide mismatches between two stacks to merge them into a locus.

We here add a third parameter. The maximum stacks allowed per locus can also now be modulated (~-max_locus_stacks). The expectation for nonrepetitive genomic regions is that a monomorphic locus will produce a single stack because the two sequences on the two homologous chromosomes are identical and thus indistinguishable. In contrast, a polymorphic locus will produce two stacks representing alternative alleles (Fig. 2A). More complex cases abound, however, from short, sequencing error-based stacks in addition to the true alleles, to repetitive sequences, where hundreds of loci in the genome may collapse to a single putative locus. Stacks can be used to identify and remove these confounding cases. For example, the maximum stacks per locus parameter allows the user to limit the number of stacks at any single locus (default 3). If the limit is exceeded, the locus is blacklisted, meaning it will not be available for insertion into, or matching against, the catalogue. These confusing loci can be ignored for all subsequent analyses. However, Stacks also contains a deleveraging algorithm in ustacks to help deconvolute some of these confounded loci. In previous versions of Stacks, if too many stacks were present at a single locus, the locus would be broken down using a hierarchical clustering algorithm. We have replaced this algorithm with a more sensitive heuristic that is based upon a minimum-spanning tree [See Appendix S1, 1.1, Supporting information for details of the algorithm].

Reference-guided stack formation

When a reference genome is available, Stacks relies on a set of aligned reads to assemble loci. Through the program pstacks, Stacks is able to use data from any alignment program that can produce SAM or BAM output files and has been extensively tested with Bowtie (Langmead et al. 2009), BWA (Li & Durbin 2009) and GSNAP (Wu & Nacu 2010). The pstacks program will read the CIGAR string (Li et al. 2009) from each alignment in the SAM file to determine whether the read contained an insertion, deletion or soft-masking [see Appendix S1, 1.2, Supporting information for information on CIGAR strings]. When a deletion has occurred in the read relative to the reference, pstacks will insert Ns to regain phase with the reference, and trim the end of the read to keep the length constant. Conversely, if an insertion has occurred in the read relative to the reference, pstacks will trim out the inserted bases and pad the end of the read with Ns. Both of these operations will allow bi-allelic loci to

Fig. 2 The ustacks deleveraging algorithm. (A) The simplest polymorphic locus is defined by a single SNP (A/T), and as the remainder of the locus is identical in both alleles, we can refer to the entire locus by the A/T haplotypes. A locus can be visualized as an undirected graph, with each allele or stack as a node, and with the nodes connected by an edge weighted according to the nucleotide distance between them. (B) This locus with three detected polymorphisms comprises six distinct stacks, which is not biologically possible and must be the result of either erroneous stacks or collapsed, repetitive loci. The deleveraging algorithm calculates a minimum-spanning tree from the locus (thick, black lines), calculates the minimum distance between any two nodes and breaks edges (separating loci) whenever they are connected by edges larger than the minimum edge. The result in this case is two loci, the first built from stacks 0, 1 and 2, and the second built from stacks 3, 4 and 5.
properly stack together when alleles vary due to an indel.

Several alignment programs allow the ends of reads to be implicitly trimmed by soft-masking them (converting the bases to Ns). The pstacks program will convert nucleotides that were soft-masked during alignment (this operation is recorded in the CIGAR string) into literal Ns, so that they do not improperly contribute to SNP calling. Users should beware, however, because some seed-based aligners (BWA and GSNAP) perform *terminal alignments*, in which large fractions of either end of a read can be soft-masked (all but the matching seed). This can result in alignments where only a fraction of the read was truly aligned to the reference and can have strange effects, such as the inability to call haplotypes despite the successful inference of SNPs, when depth of coverage is low. This behaviour can be turned off in some aligners (GSNAP).

Although reference genome aligners report reads aligned to both the positive and negative strand by the left-most genomic coordinate, pstacks will utilize the CIGAR string in the SAM file to reorient all reads such that their genomic alignment position is determined by the location of the restriction enzyme cut site. This has no effect on positively aligned reads, but will change the alignment position of negatively aligned reads to the right-hand side. Without this strand modification, bi-allelic loci containing reads with indels aligned to the negative strand would appear to be aligned to different positions and would not ‘stack’. Finally, similar to the pstacks, a threshold can be set in pstacks (-m) to require a minimum number of reads before declaring a set of aligned reads a locus.

**Identifying SNPs using a bounded-error model**

A fundamental statistical decision with GBS data is whether the distribution of read variants that contain sequencing error supports the inference of a true SNP at a given locus (Lynch 2009; Hohenlohe et al. 2012a). Stacks employs a multinomial-based likelihood model for identifying SNPs for diploid organisms whether processing data de novo or with the aid of a reference genome (Hohenlohe et al. 2010b, 2012a; Catchen et al. 2011). In the case of a reference genome, SNPs are called irrespective of the reference sequence itself. This model, implemented in both ustacks and pstacks, works by estimating the maximum-likelihood value of the sequencing error rate \( \varepsilon \) at each nucleotide position, for each possible genotype, and then calculating the likelihood of the two most frequently observed genotypes (homozygous for the most observed nucleotide or heterozygous for the two most observed nucleotides) at each site. A standard likelihood ratio test of the two hypotheses is then performed using a chi-square distribution and one degree of freedom (Hohenlohe et al. 2010b, 2012a; Catchen et al. 2011).

We introduce a bounded-error SNP calling model in this version of Stacks (Fig. 3). Our previous model allowed the error parameter to vary freely, sometimes to unrealistically high values (above 10%). Now, if the maximum-likelihood value of \( \varepsilon \) exceeds a lower or upper bound, the boundary value is substituted, allowing prior information on sequencing error rate to be used in polymorphism detection. For instance, sequencing of control samples or known sequence, or known average error rates within a sequencing facility, can be used to directly estimate error rate distribution at positions across reads (e.g. 0.001 to 0.1). Calibration of the \( \varepsilon \) bounds can also be used to balance an investigator’s tolerance for false positive vs. false negative rates in calling genotypes. Reducing the upper bound on \( \varepsilon \) increases the chance of calling a heterozygous genotype (Fig. 3). Allowing high values for the error rate \( \varepsilon \) (e.g. greater than 10%) increases the likelihood that a locus with a number of alternative reads will be called a homozygous site with excessive error. Reducing the upper \( \varepsilon \) bound decreases the chance of calling a homozygote when the true genotype is heterozygous, but...
conversely increases the potential of falsely calling a heterozygote at a homozygous locus with sequencing error. Reducing the upper bound on $\varepsilon$ may be warranted in some circumstances, such as when the sequence data have been conservatively filtered for read quality or when the data stem from pooled samples or from a polyploid organism (see Appendix S1, 1.3, Supporting information for more details).

The bounded-error model can be selected in both ustacks and pstacks by specifying the --model_type bounded option. The bounds can be set by specifying the --bound_high and --bound_low options to ustacks and pstacks. Finally, the genotype likelihood ratio test critical value ($\alpha$) was hardcoded to a value of 0.05 in the previous release of Stacks. We now allow the user to set $\alpha$ (alpha) to 0.1, 0.05, 0.01 and 0.001.

**From SNPs to haplotypes in Stacks**

Using a standard Illumina HiSeq machine, an average RAD locus will be 80–150 bp in length and may contain more than one SNP that can be phased together at a locus to form a haplotype. Within a single diploid individual, there can of course be one or two haplotypes at a locus, but within and among populations, multiple haplotypes may be segregating at each locus. The genotypes program in Stacks assigns haplotypes from the two parents of a mapping cross a meaningful letter (e.g. 'a', 'n' or 'H') depending on the design of the cross and the linkage mapping software being used and then assigns progeny corresponding genotypes based upon parental haplotypes.

Stacks presently works primarily at the SNP level for population genomics data largely for computational tractability. Although haplotype information is useful for many genetic studies, the present information content of most haplotypes from GBS is low because the reads are so short (on average 100 bp). However, Stacks’s populations program still reports which haplotypes are present in each individual in the analysis by default (in a file called batch_X.haplotypes.tsv), and it is trivial to encode these haplotypes using letters or some other meaningful scheme to be utilized for haplotype-based analyses in other population genetic analysis programs (see below for other data output formats). As read lengths of common sequencing platforms increase, the utility of haplotype information will increase. Furthermore, paired-end sequencing of sheared RAD tags with sufficient depth allows one to produce longer haplotypes from the randomly sequenced paired ends (Catchen et al. 2011; Etter et al. 2011a), allowing for the possibility of longer (500 bp) haplotypes being inferred. We will add full support for haplotypes to the population genetics components of Stacks in future releases.

**Novel population genomics components of Stacks**

**The Populations program**

The populations program is a new addition to the Stacks package enabling the calculation of core population genetics statistics (Tables 1 and 2). The goal was not to provide an exhaustive set of population genetic and genomic analysis capabilities, which are available in other software packages. Rather, we have built in the ability to export SNP and genotype data in common formats for popular population genetic and phylogenetic programs.

The list of sampling populations are supplied to the populations program in a population map file, which contains the individual sample in one column and an integer representing the population in another column. Once the first four stages of Stacks have completed, populations can be run on these processed reads repeatedly using the same catalogue-matched data, but using different parameters or population maps. Researchers can thus evaluate the sensitivity of results on different parameters and divide samples in various ways geographically or by phenotype.

The populations program has a number of filtering parameters that allow one to control execution. For example, for each locus, a researcher can set a minimum percentage of individuals within a population (-r), a minimum number of populations (-p), a minimum depth of coverage for each individual (-m) and a minimum allele frequency (-a). The populations program also produces several core population genetic statistics including $\pi$, $F_{IS}$ and $F_{ST}$ among others (Tables 1 and 2). Because various forms of statistical estimators for many population genetic parameters have been produced, we present the specific formulae for each estimator in the Appendix S1, 1.4 (Supporting information).

**Kernel smoothing of reference aligned statistics**

If a reference genome is available, the populations program provides the option of using a sliding window (-k option). Because random biological or sequencing variation might occur at any particular SNP, this application makes it possible to more easily extract consistent signals of genomic regions such as signatures of increased or decreased diversity, nonrandom mating or directional selection (Hohenlohe et al. 2011a).
By applying a Gaussian weighting function, the program can generate a kernel-smoothed moving average across each contig, scaffold or chromosome. The sliding window is centred over each polymorphic locus on each chromosome (Fig. 4), and the weights generated by the Gaussian

Table 1 Summary statistics reported for each site in each population by the populations program in the batch_X.sumstats.tsv file

<table>
<thead>
<tr>
<th>Summary statistics output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batch ID</td>
</tr>
<tr>
<td>Locus ID</td>
</tr>
<tr>
<td>Chromosome</td>
</tr>
<tr>
<td>Base pair</td>
</tr>
<tr>
<td>Column</td>
</tr>
<tr>
<td>Population ID</td>
</tr>
<tr>
<td>P Nucleotide</td>
</tr>
<tr>
<td>Q Nucleotide</td>
</tr>
<tr>
<td>Number of Individuals</td>
</tr>
<tr>
<td>P</td>
</tr>
<tr>
<td>Observed Heterozygosity</td>
</tr>
<tr>
<td>Observed Homozygosity</td>
</tr>
<tr>
<td>Expected Heterozygosity</td>
</tr>
<tr>
<td>Expected Homozygosity</td>
</tr>
<tr>
<td>( \pi )</td>
</tr>
<tr>
<td>Smoothed ( \pi )</td>
</tr>
<tr>
<td>Smoothed ( \pi ) P-value</td>
</tr>
<tr>
<td>( F_{SB} )</td>
</tr>
<tr>
<td>Smoothed ( F_{SB} )</td>
</tr>
<tr>
<td>Smoothed ( F_{SB} ) P-value</td>
</tr>
<tr>
<td>Private allele</td>
</tr>
</tbody>
</table>

Table 2 \( F_{ST} \) values reported for each site in a pair of populations by the populations program, recorded in the batch_X.fst_Y-Z.tsv file, where Y and Z are population IDs

<table>
<thead>
<tr>
<th>Pairwise ( F_{ST} ) output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batch ID</td>
</tr>
<tr>
<td>Locus ID</td>
</tr>
<tr>
<td>Population ID 1</td>
</tr>
<tr>
<td>Population ID 2</td>
</tr>
<tr>
<td>Chromosome</td>
</tr>
<tr>
<td>Base pair</td>
</tr>
<tr>
<td>Column</td>
</tr>
<tr>
<td>Overall ( \pi )</td>
</tr>
<tr>
<td>( F_{ST} )</td>
</tr>
<tr>
<td>FET ( P )-value</td>
</tr>
<tr>
<td>Odds Ratio</td>
</tr>
<tr>
<td>CI High</td>
</tr>
<tr>
<td>CI Low</td>
</tr>
<tr>
<td>LOD Score</td>
</tr>
<tr>
<td>Expected Heterozygosity</td>
</tr>
<tr>
<td>Expected Homozygosity</td>
</tr>
<tr>
<td>Corrected ( F_{ST} )</td>
</tr>
<tr>
<td>Smoothed ( F_{ST} )</td>
</tr>
<tr>
<td>Smoothed ( F_{ST} ) P-value</td>
</tr>
</tbody>
</table>
function

\[ f(x) = e^{-\frac{(x-a)^2}{2\sigma^2}} \]

are applied to all the measures of either \( F_{IS} \) or \( \pi \) within the window, and the scaled values are then averaged to produce a smoothed statistic that is assigned to the location at the centre of window \( (c) \). For each window, \( x \) represents the location of each SNP in the window, and the weighting calculation is performed for all SNPs in each window. The size of the window is determined by \( \sigma \), and by default, the tail of each side of the window is truncated at 3\( \sigma \) base pairs. This algorithm is also applied to pairs of populations to generate a kernel-smoothed \( F_{ST} \) measure in the same manner.

Kernel smoothing can only be performed using ordered genetic markers. However, if one does not have access to a reference genome, identifying \( F_{ST} \) outlier loci is still possible. One way is to create an \( F_{ST} \) by heterozygosity plot (Beaumont & Balding 2004), which can be done by matching loci across the batch_X.sumstats.tsv and batch_X.fst_Y-Z.tsv files output by populations. Another approach to finding outlier loci is to generate an empirical \( F_{ST} \) distribution through resampling. Several programs exist that can do these calculations, such as Arlequin (Excoffier & Lischer 2010) and GenePop (Rousset 2008), and the populations program provides an export into GenePop format, which in turn can be converted to Arlequin’s native input with one of several free conversion utilities.

**Genome-level tests of statistical significance using bootstrap resampling**

The smoothed values of each statistic generated by the sliding window algorithm are themselves point estimates, and confidence in these estimates requires a statistical test. For genome-wide statistics such as \( F_{IS} \), \( \pi \) or \( F_{ST} \), a common hypothesis is whether the particular value in a window is significantly different from the genome-wide average. Because of the uncertainty of distributional assumptions for genomic data, a common approach for hypothesis testing is through the use of permutation or resampling. Although conceptually simple, testing this null hypothesis using resampling or permutation can be computationally difficult. The variable number of SNPs and their locations within a window, and the evolutionary history of the genomes in the sample, make an analytical calculation of the null probability distribution very difficult and necessitate a numerical approach to generate the null distribution through resampling across the genome.

We have implemented bootstrap resampling in the populations program (\(--\)bootstrap). The sliding window is again centred on each variable site in each population. New values of \( F_{IS} \) and \( \pi \) are sampled...
with replacement from across the genome within the population and placed at the locations of the original SNPs of the focal window to calculate the smoothed statistic for that replicate. After being replicated a large number of times (as defined with the \( -b \) or \( -\text{bootstrap} \_\text{reps} \) parameter), an empirical null distribution for the test statistic is produced, against which the original \( F_\text{ST} \) and \( r \) values are compared to determine a \( P \)-value. This process is repeated for every variable site in the genome. A similar algorithm is available for re-sampling \( F_\text{ST} \) values across the genome for pairs of populations. Bootstrap resampling is computationally challenging because the calculations scale according to the number of populations multiplied by the number of variable sites multiplied by the number of bootstrap repetitions and is further complicated for \( F_\text{ST} \) calculations by scaling also with the number of pairs of populations. This algorithm is parallelized in the populations program to decrease computational time, but is still intensive and the computational challenges will increase as data sets grow.

Exporting data for use in other common evolutionary genomic programs

The populations program can output data in several additional formats. Raw haplotype calls for each catalogue locus are output into a file, \( \text{batch}_X\_\text{haplotypes}.tsv \). The populations program can export raw variable sites to a variant call format (VCF) (http://www.1000genomes.org/node/101). This file was standardized by the 1000 Genomes Project and outputs the state of each SNP in every individual in the analysis along with allele frequencies and other descriptive information. VCF files can be imported into a number of tools (e.g. VCFTools; http://vcftools.sourceforge.net). The populations program also provides a raw export, the \( \text{genomic} \) format, of every nucleotide site encoded as a number from one to ten, representing all bi-allelic combinations of nucleotides. Data can also be exported for use in common evolutionary genetic analysis programs. The SNP calls for each catalogue locus can be output in a format for the GenePop package (Rousset 2008), which can be translated for use in common packages such as Arlequin (Excoffier & Lischer 2010), FSTAT (Goudet 1995) and DnaSP (Librado & Rozas 2009) using freely available converters. The populations program can also export SNP data directly for use in the program STRUCTURE (Pritchard et al. 2000; Falush et al. 2003, 2007; Hubisz et al. 2009).

Another powerful application of restriction site-based sequencing is for phylogeographic and phylogenetic studies (Lemmon & Lemmon 2012; McCormack et al. 2013). Stacks’s \( \text{populations} \) program provides a special direct export to Phylip format for analysis in packages such as PhyML, MEGA or PAUP of a type of potentially phylogenetically informative loci, those that are fixed within all populations, but vary among at least two populations (\( -\text{phylip} \) option). Data generated from pools of DNA can be used (e.g. Emerson et al. 2010; Merz et al. 2013), but the fixed model must be specified to \( \text{ustacks} \) or \( \text{pstacks} \) manually. Instead of identifying polymorphisms, this model identifies fixed sites and masks out all other sites. Because the use of GBS data for phylogenetic studies is very recent (Rubin et al. 2012), researchers should bear in mind several caveats. While branching relationships are meaningful in these phylogenetic trees, branch lengths are not because of the concatenation of sites across the genome. Additionally, well-differentiated populations may contain a significant number of phylogenetically informative RAD loci, but as populations (or species) become even more divergent, the number of RAD loci will actually decrease as mutations accumulate in RAD restriction sites differentially across lineages. Conversely, for recently diverged populations, or those still exchanging alleles via gene flow, overall population level phylogenetic approaches that combine data from many loci may be inappropriate as different genomic regions will often exhibit incongruent genealogical patterns. Furthermore, loci may more likely be fixed among closely related populations due to diversifying selection and not neutral processes, leading to a biased concatenated tree. Congruence approaches that integrate over independent phylogenetic reconstructions at each RAD locus are still difficult because the short reads often have few variable sites. Variable sites within populations can also be included in the Phylip file by specifying the \( -\text{phylip} \_\text{var} \) flag for phylogenetic methods that can take advantage of polymorphism data.

Sensitivity analysis of Stacks parameters

To highlight the dynamics of the algorithms involved in these new \( \text{Stacks} \) features, we used \( \text{Stacks} \) to process RAD-seq data from 578 threespine stickleback fish from nine different coastal and inland populations in Oregon (Catchen et al. 2013). The data set comprised more than 820 million raw reads, of which nearly 600 million passed stringent initial quality thresholds (see Table 2, Supporting information Fig. 1 in Catchen et al. 2013). We identified 25 679 RAD loci that were present in all nine populations, nearly all of which contained one or more SNPs. We used the stickleback reference genome to align reads, infer genotypes and produce population genetic statistics. Across all individuals, an average of 84% of the reads were aligned to the genome, and those that failed to do so were from RAD sites that existed in
regions of highly repetitive sequences. A significant proportion of reads also fell into gaps in the present reference threespine stickleback genome assembly, as evidenced by the many quality stacks that were formed when we analysed the data de novo (see below). Of the reads that were aligned to the reference genome, nearly 99% of them were used in subsequent analyses. Over 110 000 SNPs were identified that produced strong phyleogeographic and population genomic inferences (Catchen et al. 2013). The very high proportion of utilized reads retained, and clear phyleogeographic and population genomic results that we obtained, support the efficiency with which Stacks can process and extract information from GBS data such as RAD-seq, despite statements to the contrary (Peterson et al. 2012).

We also ran ustacks on these same data to produce de novo stacks. This allowed us to explore the dynamics of the de novo algorithm’s parameters through a comparison with the reference genome results. In the de novo data set, ustacks was run with the lumberjack stacks (-r) and deleveraging (-d) algorithms turned on, a mismatch distance of four nucleotides between stacks (-M) and a minimum stack depth (-m) of three. On average, 37 634.8 loci per individual were discovered using the reference genome, while 42 284.5 loci per individual were found de novo. On average, an

![Fig. 5](image)

Fig. 5 De novo stack formation. (A) We ran ustacks against 590 threespine stickleback fish and compared these de novo results against the same data set aligned against the threespine stickleback reference genome. On average, 37 184.6 de novo loci aligned to a single location in the reference indicating they were correctly constructed. A small number of loci align to multiple places in the genome indicating incorrect de novo construction. (B–D) We explored how three key parameters affect the formation of de novo loci. (B) Allowing two mismatches between stacks (equivalent to nucleotide distance) results in 990 de novo loci that should be merged into 492 loci according to the reference genome. Increasing -M reduces these undermerged loci, although the rate of reduction decreases after -M 4. (C) As we increase the minimum number of raw reads required to form a stack, we see a trade-off between the number of false loci removed from our data set (blue line) vs. the number of true loci lost due to low coverage of the locus (green line). (D) As we increase the number of stacks allowed to exist at a single locus, we see a trade-off between the number of true loci added to the data set (green line) vs. the number of collapsed, false loci we add to the data set (blue line).
additional 337.3 confounded loci, or loci containing too many stacks to be biologically real, were identified and blacklisted. On average, 37 184.6 de novo loci were found to exist in unique locations in the reference genome (Fig. 5A), supporting the conclusion that they were correctly assembled in the de novo analysis. Only 233.1 loci on average were found to align to two reference genome positions, and a small number to more than two, most likely due to overmerging into a single locus in the de novo data set. Surprisingly, we found 4 839.7 de novo loci that are not present in the reference genome, indicating that the present assembly of the stickleback genome is incomplete. A number of the loci that did not align to the reference genome represent loci containing insertion/deletions (indels) in one or both of the alleles that could not be merged together using the de novo algorithm.

To explore how variation in the main usm stacks parameters affects under- and overmerging, we ran a number of trials on a single fish from which 1 053 649 raw 95-bp reads were generated. Increasing the nucleotide distance allowed between stacks (the mismatch parameter \( m \)), some undermerged alleles correctly join other alleles at a locus (Fig. 5B). Undermerged loci were identified by sets of discrete de novo loci that all align to the same reference genome locus. When the mismatch parameter is two (Fig. 5B, ‘M2’), 990 de novo loci were formed that belong in 492 reference genome locations. Increasing the mismatch parameter to three (Fig. 5B, ‘M3’) resulted in a significant drop in the number of undermerged loci, a trend that continued to a parameter value of four. However, this rate of capturing undermerged loci dropped significantly beyond four, and the number of overmerged loci steadily increased (data not shown). A trade-off between under- and overmerged loci therefore depends on the mismatch parameter, the effects of which should be explored for any particular data set.

A similar sensitivity analysis of the minimum stack depth parameter (\( m \)) revealed an analogous trade-off (Fig. 5C). We varied the minimum stack depth (starting at 2) and examined the number of erroneously formed de novo loci (Fig. 5C, green line) vs. correctly assembled and found in the reference data set (Fig. 5C, blue line). Moving from a minimum stack depth of two to three (Fig. 5C, ‘m2 > m3’, blue line) resulted in pruning 2200 erroneously formed de novo loci from the data set. These results indicated that at a minimum stack depth of two, many reads with errors existed in duplicate and were labelled as stacks in the initial hashing stage of the algorithm. Increasing the minimum stack depth parameter to three prevented these reads from forming stacks on their own and they were merged into other loci. A small number of stacks that truly have a depth of only two were lost (Fig. 5C, ‘m2 > m3’, green), but these few short stacks were unlikely to contribute to subsequent analyses because SNPs would be difficult to infer from such few reads. As the minimum stack depth was increased, the rate at which stacks absent from the reference were removed slowed and remained constant, while the number of true stacks that were discarded also slowed to a constant rate. This change occurred until a stack depth minimum of six and then continued increasing again as large numbers of true allelic stacks began to be dismantled. The exact dynamic of these transition points is contingent on the mean depth of coverage in a data set, and in general, a larger number of reads will allow for greater stack depth and thus increased sensitivity and accuracy in determining correct stacks. Similar to the mismatch parameter, researchers should perform a sensitivity analysis of the minimum stack depth parameter for each new data set.

A similar trade-off existed for the maximum stacks allowed per locus (\(-max\_locus\_stacks\); Fig. 5D). An additional 66 loci (Fig. 5D, ‘2 > 3’, green line) appeared when three as compared to two stacks are allowed, while in 23 cases, the result was to overmerge a locus that subsequently aligned to multiple places in the reference genome (Fig. 5D, ‘2 > 3’, blue line). The most likely explanation for the additional 66 loci is that a small error stack occurred along with the two true alleles in the data set. The rate of single reference alignment gain stayed well above overmerged loci gain until a maximum stacks value of four to five (Fig. 5D, ‘4 > 5’, green line). An asymptote was reached at this point, but not before the number of overmerged stacks began to outpace the gain in valid reference alignments.

The specific values of the mismatch distance (\( m \)), minimum stack depth (\( m \)) and maximum stacks allowed per locus (\(-max\_locus\_stacks\)) chosen by the researcher represent a trade-off between leaving undermerged loci in the data set and confounding loci in the data by overmerging them. The optimal values for these parameters depend on the rate of polymorphism, the amount of sequencing error and the depth of sequencing performed. We therefore strongly encourage researchers to test a range of values for each parameter when approaching a data set for the first time.

**The efficacy of kernel-smoothed \( F_{ST} \) analysis**

Marine stickleback populations are thought to be large, old, genetically diverse and well-mixed (Wootton 1976; Bell & Foster 1994; Cresko et al. 2007; Hohenlohe et al. 2010b). By contrast, the freshwater populations to which they give rise are thought to be smaller and younger and more genetically rarified (Cresko et al. 2007). Using Stacks’s populations program (\(-k\) option to turn on kernel smoothing), we compared
pairwise $F_{ST}$ values for a subset of four of our nine Oregon populations (Fig. 6) spanning a range of geographical distances. The results demonstrated the effect of degree and kind of genetic divergence on the ability to detect signatures of selection using $F_{ST}$ scans. The two marine populations, which are phenotypically similar and geographically close, showed no significant divergence along linkage group I. (B) The coastal fresh and marine populations show clusters of highly significantly diverged SNPs from 7 to 12 Mb of group I. (C) In the marine by Willamette basin comparison, a series of divergent SNPs raise the overall level of $F_{ST}$ along the entire chromosome and probably represent neutral differences accumulated during the long separation of the populations. (D) The large number of fixed differences SNPs with an $F_{ST}$ of 1.0 in comparison between marine and central Oregon fish is the result of genetic subsampling during recent founding of the inland populations.
Discussion

Many new genomic studies can now be performed using GBS techniques such as RAD-seq, particularly in organisms for which few genomic resources presently exist (Barchi et al. 2011, 2012; Baxter et al. 2011; Rowe et al. 2011; Bus et al. 2012; Everett et al. 2012; Houston et al. 2012; Lemmon & Lemmon 2012; Scaglione et al. 2012; Wang et al. 2012a; Yang et al. 2012a). For example, many closely related populations and species have evolutionary histories that are obscured by incomplete lineage sorting during rapid cladogenesis, significant gene flow after lineage splitting or independent evolutionary histories of genomic regions. RAD-seq data can increase the resolution of population structure and phylogenetic relationships significantly (Rubin et al. 2012).

Phylogeographic studies using GBS markers have recently been completed in the pitcher plant mosquito Wyeomyia smithii (Emerson et al. 2010; Merz et al. 2013), carnivorous plant Sarracenia alata (Zellmer et al. 2012), nine-spine stickleback Pungitius pungitius in Scandinavia (Bruneaux et al. 2013) and recently diverged species of birds (McCormack et al. 2012). Similarly RAD-seq is well suited to identify genomic regions under selection because of the uniform high density of markers across genomes. This approach has been successful in global isolates of C. elegans (Andersen et al. 2012), sunflowers of the genus Helianthus (Andrew et al. 2013), Heliconius butterflies (Nadeau et al. 2013), trees in the genus Populus (Stolting et al. 2013), cichlid species (Keller et al. 2013; Wagner et al. 2013), different lineages of trout (Hohenlohe et al. 2011; Amish et al. 2012; Everett et al. 2012; Hecht et al. 2012, 2013; Miller et al. 2012) and three-spine stickleback (Hohenlohe et al. 2010b, 2012b).

GBS approaches are also producing key insights into genomic divergence during speciation in a variety of organisms (Gompert et al. 2012; Nosil et al. 2012; Nice et al. 2013; Parchman et al. 2013). RAD-seq data can also be used to link genotype to phenotype through QTL Mapping (Barchi et al. 2011, 2012; Chutimanitsakun et al. 2011; Pfender et al. 2011; Houston et al. 2012; King et al. 2012), and a promising avenue for the use of GBS studies may be genome-wide association studies (GWAS) in natural populations (Rosenberg et al. 2010; Balding 2006; Luo et al. 2011). GBS-based GWAS approaches have been used to identify loci associated with migration propensity in steelhead salmon (Hecht et al. 2012, 2013), genomic regions in lodgepole pine important for cone opening during fires (Parchman et al. 2012), the sex determination region in zebrafish (Anderson et al. 2012) and a locus responsible for resistance to stem blight disease in lupin (Lupinus angustifolius L.; Yang et al. 2012a,b).

The massive amounts of data in the studies listed above are truly revolutionizing the fields of ecological and evolutionary genomics, but this increasing volume poses serious challenges for data processing and analysis. We wrote Stacks as an integrated and focused platform to help speed GBS analyses. Stacks is primarily written in the computationally efficient C++ programming language and includes Perl scripts for common tasks. Much of the pipeline is parallelized to take advantage of shared memory multicore computers. Stacks can take as input any restriction digest-based data (Davey et al. 2011; Peterson et al. 2012; Wang et al. 2012b) and now produces core population genomic summary statistics such as diversity indices (π and private alleles) and inbreeding coefficients (FIS and FST), and SNP-by-SNP statistical tests (Fisher’s exact test, P-value cut-offs and multiple test corrections). When performed in conjunction with a reference genome, the software synthesizes these statistics together across the genome using a sliding window algorithm that generates bootstrap resampling statistics. Stacks now provides several common output formats to mesh Stacks-generated genotype data with downstream analysis packages.

Other pipelines are available to produce genotype information in groups of individuals. Two of the most widely used are SAMtools/BCFtools (Li et al. 2009) and the Genome Analysis Toolkit (GATK, McKenna et al. 2010). These tools are meant to operate on top of a genome, for example by detecting nucleotide variants through matches to the reference sequence. GATK, in particular, is highly optimized to work on the human genome. In contrast, Stacks was developed to have at its core a catalogue that works as an internal reference for each project regardless of the presence of a genome. Even when a reference genome is used to stack reads, nucleotide variants are still identified de novo. The catalogue approach is particularly useful for the majority of organisms for which a reference genome does not exist or is in a draft state. Furthermore, SAMtools/BCFtools and GATK can call SNPs in multiple samples and can generate allele frequencies, but there is no built-in concept of populations. Instead, populations are managed by hand as collections of BAM and VCF files, as compared to the integrated way that this occurs in Stacks. Finally, for all of these tools, the analysis ends with lists of SNPs (‘analysis ready variants’) that can be used in subsequent analyses but with some difficulty. In contrast, a Stacks analysis is highly integrated so as to start with raw sequencing reads and then progress through all stages of an analysis to produce allele and genotype calls, a number of core population genetics statistics and formatted output files.

The analysis of short-read sequence data for population genomics is advancing quickly, and Stacks has been...
built to grow in concert. Areas of rapid development are
the use of hidden Markov model (HMM; Boitard et al. 2013) and Bayesian approaches in population genomic analyses (Futschik & Schlötterer 2010; Gompert et al. 2010; Buerkle & Gompert 2012). Just as sampling individuals from populations leads to uncertainty in inferring population genomics statistics, short-read sequencing adds new levels of sampling variation during all aspects of the library preparation and sequencing process, notably at the very beginning of population genomic analyses (Hohenlohe et al. 2011; Davey et al. 2012; Gautier et al. 2012). A conceptually simple approach would be to directly integrate this sequencing uncertainty into hierarchical population genetic models using assumptions about parametric distributions (Kofler et al. 2011a,b, 2012; Buerkle & Gompert 2012). However, the causes of sequencing depth variation, and their effects on distributional assumptions, are poorly understood for short-read sequencing in general and GBS approaches in particular (Davey et al. 2012; Gautier et al. 2012). We have therefore decided for the time being to maintain a likelihood model hypothesis testing approach in Stacks to generate SNP and genotype calls. This approach is more conservative in that assumptions about the expected read depth distributions are not carried through to downstream analyses. In addition, the output from Stacks can be used with existing hierarchical Bayesian models once genotypes are inferred (Buerkle & Gompert 2012), but also allows a wider range of downstream analyses using other software, such as genetic mapping or estimates of linkage disequilibrium in natural populations, where the correct identification of SNPs and haplotypes is critical. In the future, we plan to incorporate more comprehensive Bayesian approaches into Stacks given an appropriate understanding of sequencing variation.

In summary, we have built Stacks to be a key resource to empower researchers to efficiently perform ecological and evolutionary genomic studies in model organisms and particularly in organisms with minimal or no genetic resources. Stacks now produces core population genomic summary statistics and SNP-by-SNP statistical tests. These statistics can be analysed across a reference genome using a smoothed sliding window. Stacks also now provides output formats for several commonly used downstream analysis packages. Stacks will be expanded and improved in concert with additional analytical developments in the field of population genomics such as model-based inferential statistics as the understanding of sequencing increases. Thus, the expanded population genomics functions in Stacks make it a useful tool to harness the newest generation of massively parallel genotyping data for ecological and evolutionary genetics studies now and into the future.

Acknowledgements

We thank Matt Streisfeld and Jay Sobel for insights into RAD-seq data for evolutionary genomic analyses and Janna Fierst and Patrick Phillips for discussions about statistical approaches. We also thank the Stacks user community for providing numerous helpful comments via the mailing list and the students we have taught during short courses who have provided important feedback on the functionality of Stacks. We are grateful to members of the Cresko Lab who have provided comments and encouragement at the University of Oregon. This work has been generously supported by NSF grants IOS-1027283 and DEB-0949053 (WAC), NIH grant 1R24GM079486-01A1 (WAC), the M. J. Murdock Charitable Trust (WAC) and NIH NRSA Ruth L. Kirschstein fellowship F32GM095213-01 (JMC). PAH also received support from NIH NCRR grant P20RR16448 (L. Forney, PI).

References


Yang H, Tao Y, Zheng Z et al. (2012b) Rapid development of molecular markers by next-generation sequencing linked to a gene conferring phomopsis stem blight disease resistance for marker-assisted selection in lupin (Lupinus angustifolius L.) breeding. TAG, Theoretical and Applied Genetics, 126, 511–522.

J.C. designed and wrote Stacks. P.H. and W.C. developed or adapted statistical methodologies that are implemented into Stacks. S.B., A.A., and W.C. provided input that improved the design of Stacks. J.C. performed the analysis of the Oregon data. J.C., S.B., P.H. and W.C. wrote the manuscript.

Data accessibility
All raw RAD-seq data, and inferred genotypes, utilized in this manuscript are the same as those presented in a companion manuscript in Molecular Ecology (Catchen et al. 2013). For information on accessing these data, see the information provided in this companion paper (Catchen et al. 2013).

Supporting information
Additional supporting information may be found in the online version of this article.
Appendix S1 Details of the minimum spanning tree deleveraging algorithm in ustacks, the bounded error SNP calling model, and implemented core population genetics statistics used in Stacks.