Whole-genome population genomic analyses

Inaugural-Dissertation

zur Erlangung des Doktorgrades der
Mathematisch-Naturwissenschaftlichen Fakultät der
Heinrich-Heine-Universität Düsseldorf

vorgelegt von

Bastian Pfeifer
aus München

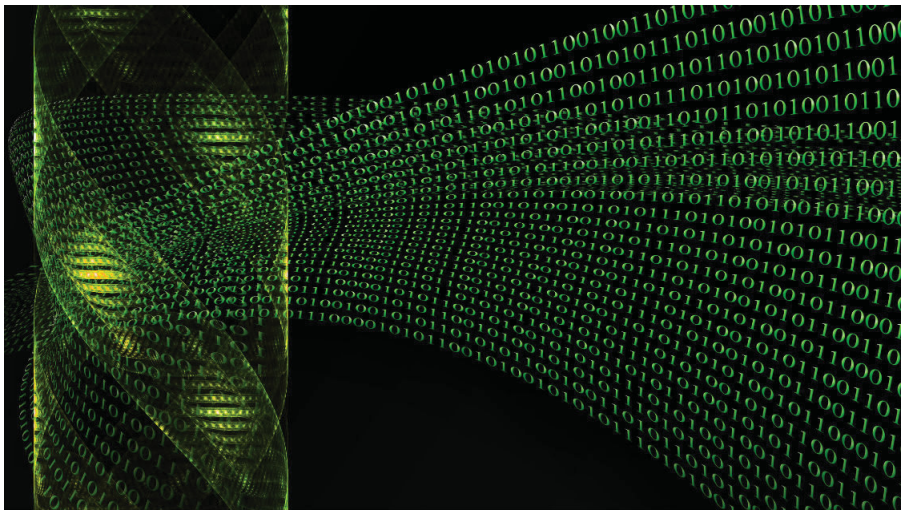
Düsseldorf, Juni 2014

aus dem Institut für Informatik
der Heinrich-Heine-Universität Düsseldorf

Gedruckt mit der Genehmigung der
Mathematisch-Naturwissenschaftlichen Fakultät der
Heinrich-Heine-Universität Düsseldorf

Referent: Prof. Dr. M.J. Lercher
Korreferent: Prof. Dr. L. Rose

Tag der mündlichen Prüfung:



My father would say: "*History repeats. Now, Bioinformatics is the same to Biology as Computer Science was a few years ago to Mathematics*". I suppose he does not really understand what I did in my thesis, but this sentence of him has some truth in it.

When I was forced to explain my friends what I was doing all the time and what Bioinformatics is about, I said: "*Bioinformatics is what physicists applied to other systems of nature long time ago*" . However, from the very beginning I was fascinated with this research area and the excellent reputation of Bioinformatics at the Heinrich-Heine-University was one of the main reasons to study there.

Acknowledgements

I would like to thank my girlfriend Olivia who really had some hard times with my neurotic fixation on this thesis during the last 3 years. Thank you for your abundance of patience.

I also want to thank my family Sarah, Peter and Anneliese for all the support. I deeply acknowledge my supervisor Martin for offering me this thesis and opening me a door to the world of science.

Summary

Massive re-sequencing projects such as the human 1000 genomes project are providing genetic variation data from thousands of individuals around the world. The production of such databases rapidly increased in the last decades and has shifted the bottleneck in population genetics from data acquisition to data analysis. Projects like this imply a substantial progress for the elucidation of important questions regarding human evolution and the genetic contribution to disease.

Widely used population genetic software packages like DnaSP do not have the capability to handle such whole genome data in an efficient way. This (cumulative) thesis introduces the new population genomic software package PopGenome, "*An efficient Swiss army knife for population genomic analyses in R*". PopGenome facilitates large-scale population genetic analyses and at the same time provides an ideal framework for the effortless integration of new population genetic & genomic methods.

An ultimate goal of theoretical population genetics is the design of methods which can tell as much as possible about the evolutionary history of populations on the basis of DNA sequence data observed in the present. Once the evolutionary forces are detected, future developments of biological systems become predictable. Genes which are subject to any kind of selection play a decisive role in the understanding of evolution as they reveal an important biological functionality.

The second project introduced here is a new genome scan method to detect genes which are under balancing or directional (population-specific) selection via Bayesian inference.

Zusammenfassung

High-throughput Technologien haben in den letzten Jahren einen rasante und kostengünstige Produktion genomischer Informationen von tausenden biologischen Organismen ermöglicht. Das 1000 Genomes Projekt z.B. sequenziert zur Zeit menschliche Genome von Populationen auf der ganzen Welt, um den größten Katalog menschlicher genetischer Variationen in der Geschichte zu erstellen. Es zielt darauf ab, bedeutende Fragen der Evolutionsbiologie zu beantworten und neue Erkenntnisse über die Rolle einzelner Variationen bei der Entstehung von Krankheiten zu erlangen.
Dieser Fortschritt geht einher mit der Notwendigkeit von Software Applikationen, mithilfe derer die Analyse dieser enormen Datenmengen problemlos zu realisieren ist. Etablierte populations-genetische Analyse Software, wie z.B. das Programm *DnaSP*, sind nicht dafür entwickelt worden Datenmengen solchen Ausmaßes effizient zu verarbeiten. Diese Dissertation präsentiert unter Anderem das neue Software Paket *PopGenome*. Zum einen beinhaltet *PopGenome* ein weites Spektrum populationsgenomischer Methoden, zum Anderen bietet die zugrunde liegende Architektur eine Umgebung, die die Integration neuer Methoden spielerisch einfach macht.

Das Ziel theoretischer Populationsgenetiker ist es, Methoden zu entwickeln, die auf der Basis heutiger Muster im Genom möglichst genaue Aussagen über die Evolutionsgeschichte machen können. Erst wenn diese Muster entschlüsselt sind können zukunftige Entwicklungen biologischer Systeme sicher vorhergesagt werden. Gene, die unter Selektionsdruck stehen und somit eine bedeutende biologische Funktion implizieren, sind ein wesentlicher Bestandteil dieser Entwicklungen. Ein zweiter Teil dieser Dissertation stellt eine neue bayesianische Methode vor, die anhand von genomweiten SNPs (Single nucleotide polymorhisms) Gene verifiziert, die sich entweder unter balancierender oder populations-spezifischer Selektion befinden.

Contents

List of Figures

Chapter 1

Introduction

In the last decades high throughput sequencing technologies boosted the production of genomic data containing DNA sequence informations from thousands of organisms and individuals around the world. The 1000 genomes project [5] aims to provide the biggest database of human genetic variation data in history, facilitating the understanding of genetic contribution to disease. Analogously the same is done for whole genome variation data of the reference plant Arabidopsis thaliana [3], which is the most popular model organism in plant genetics.

Projects like these imply a substantial progress for the elucidation of important questions regarding human evolution [6]. However, while many people are doing serious sequencing now on numerous non-model organisms there is no population genomics software package available to analyse such data in an effortless and efficient way. The widely used software program DnaSP [7], for instance, incorporates a wide range of established methods but only is suitable for small scale population genetic analyses.

Usually, biologists write their own scripts as an ad-hoc solution for the analysis needed for a specific task. Thus, the corresponding software is not designed for extensibility. In addition, the source code is not available to the research community, consequently identical processes are done over and over again. The paper "*Computer programs for population genetics data analysis: a survival guide*" [13] published in the journal *Nature Reviews Genetics* signifies this dysplasia by providing a guide through a wide range of population genetic & genomic software packages. Obviously a software environment is needed where most of those implementations can be easily integrated and which is simultaneously armed for the big data era.

My first project during my PHD encompassed the development of such a framework.

The corresponding paper is in press at the journal *MBE* with the title "*PopGenome: An efficient Swiss army knife for population genomic analyses in R*".

An ultimate goal of theoretical population genetics is the design of methods which can tell as much as possible about the evolutionary history of populations on the basis of DNA sequence data observed in the present. The knowledge about the history of genes or whole genomes strongly influences the predictability of future developments of biological systems. One of the main goals is to distinguish between neutrally evolving genes and genes which play a significant role in terms of revealing an important biological functionality.

The paper "*GeneFeST: Bayesian calculation of gene-specific F_{ST} from genomic SNP data*" introduces a new method to detect genes which are subject to selection on the basis of population differentiation measurements. We have submitted the paper to the journal *MBE*.

Chapter 2

PopGenome: An efficient Swiss army knife for population genomic analyses in R

PopGenome is a new R-package for population genomic & genetic analyses in R. R is a high-level open source interpreted programming language with particular strengths in statistical computing and graphics. The R-project enables scientists to contribute their software packages and guarantees system independence due to stringent restrictions on the source code. PopGenome is designed to handle a huge amount of individual loci as well as big data files containing whole genome variation data such as those contributed by the HapMap [1], the Human 1000 genomes [5] and the Arabidopsis 1001 genomes project [3].

The current version of PopGenome includes a wide range of neutrality statistics, linkage disequilibrium and population differentiation measurements like Hudson's F_{ST} [18] which all can be applied to individual loci, sets of loci and sliding windows. Neutrality statistics like the Tajima's D statistic [8] aims to distinguish between "neutrally" evolving DNA sequences and those which are influenced by evolutionary forces like selection or demographic factors (e.g population expansions or bottlenecks).

Neutrality statistics are based on the coalescent theory firstly introduced by Kingman (1982) [12]. It is the most popular concept of neutral evolution in the field of population genetics. The coalescent process attempts to trace all alleles of a single locus (gene) backwards in time until a most recent common ancestor (MRCA) for the whole sample of alleles is found. During this binomial sampling process an ancestral history of the alleles is produced. The sampling scheme can be affected by additional

evolutionary parameters and thus enables to test the hypothesis if the alleles observed in the present could have arisen by chance (selectively neutral) [11].

Simulation programs like Hudson's MS [17] utilize this theory to generate patterns of neutrally evolving genes under different model parameter settings. To analyse several types of selection the coalescent simulation program MSMS [9] can be used. PopGenome incorporates both programs and provides an effortless way to compare simulated "theoretical" data with observed data.

Beside a wide range of so called "moment estimators", PopGenome also provides an R version of Foll & Gaggiotti (2008) [15], a Bayesian approach to detect genes which are subject to selection. The corresponding *BayeScan* software calculates an explicit posterior probability for non-neutral evolution at each locus.

Moreover, PopGenome can automatically handle annotation files like those in the GFF/GTF format. Accordingly, the full range of methods can be applied to feature centered regions (e.g. genes and coding regions) of the genome. One of the major strengths of this work is that our framework is suitable for sequence data which contains full nucleotide information of the reference genome, as well as for SNP data which only contains information about the variable sites, which hampers the calculation e.g. of synonymous and non-synonymous codon substitutions.

PopGenome stores the genomic data in ff-objects provided by the R-package ff [2]. ff objects are structures that are stored on disk but behave (almost) as if they were in RAM by transparently mapping only a section (pagesize) in main memory. In fact, PopGenome only depends on this single R-package and; this dependence can be easily replaced by other mechanisms in the future, if necessary.

2.1 Accelerated computations on large vectors and matrices in R

During the work on PopGenome we discovered that R itself produces some extensive computational bottlenecks hindering efficient population genomic analysis of whole genome variation data. R is an interpreted vector-oriented programming language and is internally written in C and FORTRAN to ensure good computational performance.

However, R functions are designed for a general applicability to a wide range of data structures and thus cause unnecessarily large overheads in special cases where computation time is important. Numeric vectors, for instance, containing already sorted entries are rarely considered in the R environment. Moreover, vector-oriented programming languages usually apply calculations to every entry of a data structure, whereas in many cases a return value can already be found from a subset of the entries.

PopGenome uses as much as possible simple data structures like matrices and vectors to store genomic data. In the paper "*BASIX: Accelerated computation on large vectors and matrices in R*" we introduce the R package BASIX, which provides some useful functions applicable to large vectors and matrices. The work discusses the gained accelerations theoretically, based on O-notations, and through computer simulations. In this work we also suggest some R-packages contributed on CRAN which might be very useful for researchers intending to speed up their source code in R.

The BASIX paper is currently under review at the *R journal*.

2.2 High-speed access to whole genome variation and sequence data

Ulrich Wittelsbürger, now also a PHD candidate at the Bioinformatics group of Heinrich Heine University, started developing an R-package named WhopGenome in his Master thesis under my supervision. The idea was to offer comfortable access to whole genome data in the format published by the 1000 genomes project.

WhopGenome incorporates the software Tabix [10], providing selective access to VCF (Variant Call Format) [4] files, which is the main file format contributed on the 1000 genome project platform. In addition, for PopGenome it was vital to have a mechanism which enables to stream genomic data as SNP chunks in order to interpret the data step by step instead of reading everything into the RAM. PopGenome is build up for reading data chunks by interpreting those chunks step by step and concatenating the necessary information afterwards.

At the end WhopGenome became a stand-alone R-package which now also is available on CRAN. It provides additional functionalities which may be very useful for population geneticists worldwide.

The corresponding paper has been submitted to the journal *Bioinformatics*.

2.3 Application: The CNTNAP5 gene

PopGenome is the ideal framework for exploratory genomic data analysis. During some sliding window genomic scans on the human chromosome 2, we discovered a selective sweep region at the genomic region 123Mb-127Mb (see Figure 2.1). The only protein-coding gene annotated in this region is the CNTNAP5 gene. Its deletion appears to be related to autism (and possibly dyslexia) as well as other cognitive/psychological disorders (e.g. bipolar disorder). In a detailed population-specific analysis (see also Figure 2.1), we discovered the Asian populations as a sweep candidate for this specific region.

Africa (246 individuals)

Location	Number of individuals
African Ancestry in Southwest US	61
Lukya in Webuye, Kenya	97
Yoruba in Ibadan, Nigeria	88

Asia (286 individuals)

Location	Number of individuals
Han Chinese in Beijing, China	97
Han Chinese South	100
Japanese in Toyko, Japan	89

Europe (287 individuals)

Location	Number of individuals
Northern and Western European	87
British from England and Scotland	88
Iberian Populations in Spain	14
Toscani in Italia	98

Figure 2.1: CNTNAP5: Populations and number of individuals used for the genomic scan on Human chromosome 2 shown in Figure 2.2

The results observed from Figure 2.2 suggests a recent selective sweep in the Asian

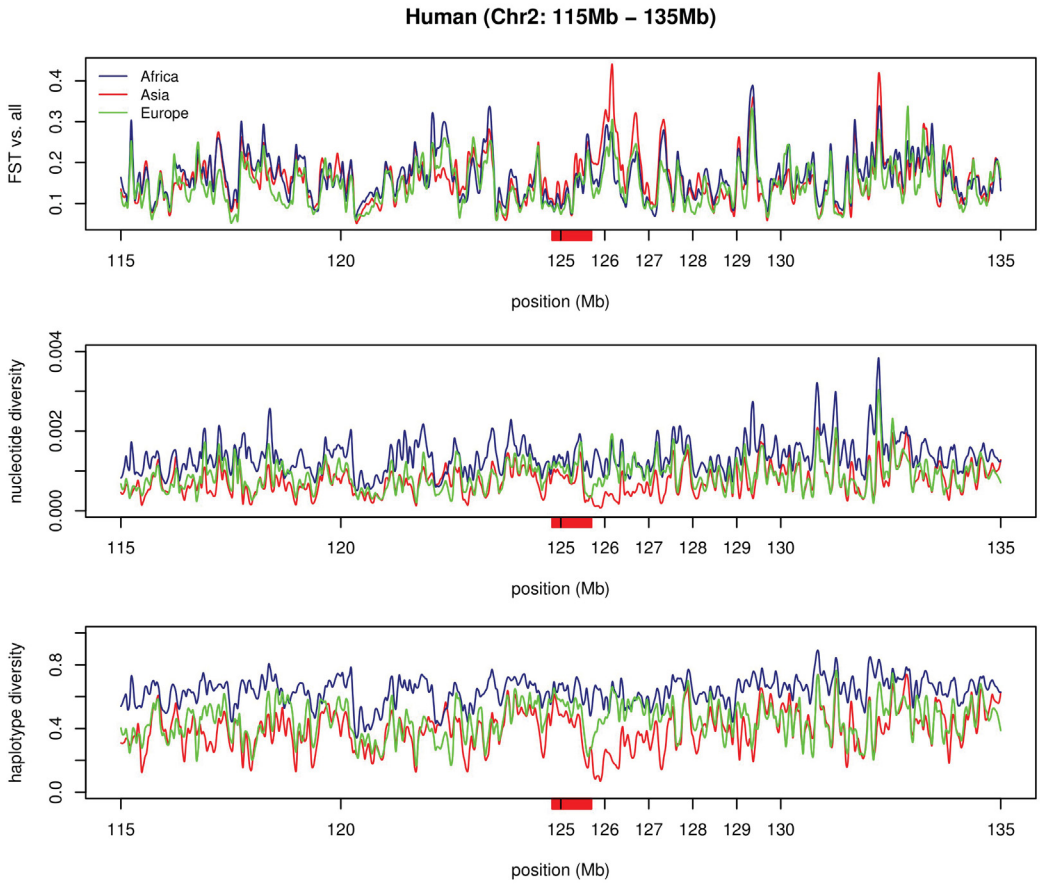


Figure 2.2: Measures of diversity in sliding windows (10kb) on Human chromosome 2 in 3 populations using the R-package PopGenome (LOESS-smoothed in R, span=0.01). The red bar on the x-axis indicate the position of the CNTNAP5 gene (the only protein-coding gene annotated 125Mb-127Mb, GRCh37.p10). F_{ST} vs. all means: nucleotide F_{ST} in sliding windows, contrasting one population with the union of the two others.

populations, possibly related to selection on cognitive/psychological traits. Interestingly, the same calculations applied to the data published by the *International HapMap Consortium* produced results where the peak is slightly more shifted into the CNTNAP5 region. Moreover, the diversity measurements on human chromosome 2 indicate the CNTNAP5 gene as a target of recent regional selection in Asians on cognitive traits.

Previous population genetics papers mentioned CNTNAP5, but always in passing or in tables and/or supplements, and always without reference to cognitive functions. However, the main issue of these observations is that the sweep area apparently lies

directly behind the target gene. We didn't find any annotated functional elements (e.g promoters) in this region. It would be very interesting to know if there are positions of disease-causing mutations inside the sweep area. Moreover, it would be interesting to examine if genes functionally linked to CNTNAP5 show similar signals.

For the calculations we have used data from variant calls in VCF-format published 2011 on the 1000 Genome project:`ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/release/20101123/`. We repeated the analyses for the following data published 2012 on the 1000 Genome project: `ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/release/20110521/` with very similar results.

Chapter 3

GeneFeST: Bayesian calculation of gene-specific F_{ST} from genomic SNP data

The fixation index F_{ST}, first introduced by Wright (1950) [19], measures the degree of population subdivision. On the account that signals of population differentiation are very similar to those caused by several types of selection (e.g balancing or directional or population-specific selection), F_{ST} was also established as a test for non-neutral evolution.

It can be misleading to consider only one gene and make predictions about selection on the basis of these observations, as the neutral F_{ST} value strongly depends on factors which contribute to heterogeneous genomic divergence. Thus, genome wide calculations should be preferred. In nowadays practice, fixation indices are routinely calculated for whole chromosomes to get an idea of the genome-wide neutral F_{ST} distribution. Accordingly, genes are determined as significant outliers when their F_{ST} value passes some user defined threshold (e.g the 0.05 quantile). To facilitate those approaches PopGenome is the ideal platform as one can easily scan genome-wide data based on annotations such as those specified in GFF/GTF files. However, it is a challenging task to verify such a threshold value.

To remedy these shortcomings, Beaumont & Balding (2004) [14] introduced a rigorous way to detect outlier loci that are subject to selection via Bayesian calculations. The method is based on a logistic regression model to distinguish between locus-specific effects (selection), and population-specific effects that reflect the common history of all loci. Accordingly, F_{ST} is split into a locus (e.g gene) specific compounded α and

a population specific parameter β shared by all loci. Monte-Carlo Markov-Chain (MCMC) is used to determine the corresponding parameters.

Foll & Gaggiotti (2008) [15] extended this work by introducing a reversible jump model conceived by Green (1995) [16] to explicit generate posterior probabilities for non-neutral population differentiation. In the paper "*GeneFeST: Bayesian calculation of gene-specific F_{ST} from genomic SNP data*" we introduce a modified version of these approaches and validate our method with Greg's MSMS [9] simulation program.

The simulations clearly show that our new approach has better performance to detect genes that are subject to selection, in particular, balancing selection genes were detected significantly better. Moreover, we discovered that the posterior probabilities of non-neutrality produced by Foll & Gaggiotti's approach contains much less information than the corresponding posterior F_{ST} values.

An R version of the method is available on CRAN and the corresponding paper has been submitted to the journal *MBE*.

Chapter 4

Manuscripts

PopGenome: An efficient Swiss army knife for population genomic analyses in R
Authors: *Bastian Pfeifer, Ulrich Wittelsbürger, Sebastian E. Ramos-Onsins and Martin J. Lercher*

BASIX: accelerated computations on large vectors and matrices in R
Authors: *Bastian Pfeifer and Martin J. Lercher*

WhopGenome: high-speed access to whole genome variation and sequence data in R
Authors: *Ulrich Wittelsbürger, Bastian Pfeifer and Martin J. Lercher*

GeneFeST: Bayesian calculation of gene-specific F_{ST} from genomic SNP data
Authors: *Bastian Pfeifer, Stefan Habenschuss and Martin J. Lercher*

4.1 Manuscript 1: (chapter 2)

PopGenome: An efficient Swiss army knife for population genomic analyses in R

Authors: *Bastian Pfeifer, Ulrich Wittelsbürger, Sebastian E. Ramos-Onsins and Martin J. Lercher*

Status quo: published
Journal: *Molecular Biology and Evolution (MBE).*
Impact factor: 10.353

Contributions: MJL conceived of the project in general terms. BP developed the details of the project, designed the program architecture, implemented the algorithms, and published the R-package PopGenome on CRAN. UW implemented the functionality for the selective access to tabixed VCF (Variant Call Format) files. SERO consulted BP on the implementation of statistical tests. BP and MJL wrote the paper.

MBE Advance Access published May 5, 2014

PopGenome: An Efficient Swiss Army Knife for Population Genomic Analyses in R

Bastian Pfeifer,[1] Ulrich Wittelsbürger,[1] Sebastian E. Ramos-Onsins,[2] and Martin J. Lercher*[,1,3]

[1]Institute for Computer Science, Heinrich Heine University, Düsseldorf, Germany

[2]Centre for Research in Agricultural Genomics, Bellaterra, Spain

[3]Cluster of Excellence on Plant Sciences, Düsseldorf, Germany

***Corresponding author:** E-mail: lercher@cs.uni-duesseldorf.de.

Associate editor: Juliette de Meaux

Abstract

Although many computer programs can perform population genetics calculations, they are typically limited in the analyses and data input formats they offer; few applications can process the large data sets produced by whole-genome resequencing projects. Furthermore, there is no coherent framework for the easy integration of new statistics into existing pipelines, hindering the development and application of new population genetics and genomics approaches. Here, we present PopGenome, a population genomics package for the R software environment (a de facto standard for statistical analyses). PopGenome can efficiently process genome-scale data as well as large sets of individual loci. It reads DNA alignments and single-nucleotide polymorphism (SNP) data sets in most common formats, including those used by the HapMap, 1000 human genomes, and 1001 Arabidopsis genomes projects. PopGenome also reads associated annotation files in GFF format, enabling users to easily define regions or classify SNPs based on their annotation; all analyses can also be applied to sliding windows. PopGenome offers a wide range of diverse population genetics analyses, including neutrality tests as well as statistics for population differentiation, linkage disequilibrium, and recombination. PopGenome is linked to Hudson's MS and Ewing's MSMS programs to assess statistical significance based on coalescent simulations. PopGenome's integration in R facilitates effortless and reproducible downstream analyses as well as the production of publication-quality graphics. Developers can easily incorporate new analyses methods into the PopGenome framework. PopGenome and R are freely available from CRAN (http://cran.r-project.org/) for all major operating systems under the GNU General Public License.

Key words: **population genomics, software, single-nucleotide polymorphisms.**

Introduction

Recent sequencing technologies allow to map genetic variation across hundreds of individual genomes (Harrison 2012); notable examples are the 1000 genomes project in humans (1000genomes.org) and the 1001 genomes project in *Arabidopsis thaliana* (1001genomes.org). These technological developments have shifted the bottleneck in population genetics from data acquisition to data analysis.

Different software packages for population genetics (or population genomics) analyses typically have limited overlap in implemented statistics and accepted input formats. This diversity hampers both efficient data analysis and the quick dispersion of new statistical approaches. Many widely used software packages, such as DnaSP (Rozas et al. 2003), cannot handle the data formats developed for massive resequencing projects. Only few programs support the use of genomic annotation, and thus users interested in specific regions have to preprocess the data using other tools.

To address these issues, a new software tool for population genomic data analysis should:

- read data in a variety of input formats, including both traditional formats and those used by the major resequencing projects;
- implement a comprehensive range of population genetics/genomics analyses and statistics;
- read associated annotation files and allow to systematically select regions of interest;
- be able to analyze individual loci, multiple loci, and sliding windows;
- be open source and be easily extendable by the scientific community to incorporate new types of analyses;
- be integrated with powerful numerical and graphical capabilities; and
- be platform independent.

We implemented these features in PopGenome, a package embedded in the freely available, platform-independent, statistical, and graphical computing environment R (http://cran.r-project.org/, last accessed April 30, 2014).

Description of PopGenome

Summary

To fully exploit the capabilities of the R statistical and graphical environment, and to allow the creation of stable workflows (scripts), all processes in PopGenome are executed as command line functions. However, we anticipate that a

Table 1. Times Required to Read Large Data Sets.

Data Set	Individuals	SNPs	Format	Time for Reading[a]
Arabidopsis (Chr 1)	80	1,200,000	SNP (1001 Genomes)	<1 min[b] ~3 min
Human (Chr2: 100–150 Mb)	1,094	660,000	VCF (1000 Genomes)	~5 min
3450 individual alignments	25	200,000	FASTA	~15 s

[a]Intel® Core™ i3-2130 CPU @ 3.40 GHz × 4, 8 GB RAM, with data stored in temporary files.
[b]Without temporary files, if sufficient RAM is available.

graphical user interface implementing the most important functionalities will be available in the near future.

PopGenome is designed to facilitate the easy integration of virtually all major types of population genetics and population genomics analyses, and, as outlined below, its current version includes a large array of different statistics (see overview in table 2). The emphasis on extensibility was inspired by the way Bioconductor (bioconductor.org) has come to dominate the analysis of microarray data, where newly developed methods for specific tasks are easily integrated into a preexisting larger framework, thereby obviating the need to recreate tasks shared with existing analysis pipelines. We hope that PopGenome may become the kernel of a similar paradigm in the analysis of population genomics data, enabling researchers to effortlessly implement and share new or modified statistics.

That PopGenome is embedded within the R framework not only facilitates the easy integration of extensions and stable workflows but also allows immediate and effortless postprocessing of analysis results with R's powerful numerical and graphical capabilities.

Data Organization

PopGenome can read data both as full alignments and in single-nucleotide polymorphism (SNP) formats such as those generated by large resequencing projects. PopGenome's ability to simultaneously manage large numbers of loci, which allows for variation in sequence and population coverage, provides a convenient framework for multilocus analyses.

After reading data, PopGenome first converts it into a biallelic matrix, that is, a matrix whose rows correspond to sequences and whose columns correspond to SNP positions in the alignment. Entries are either 0, indicating the major allele, or 1, indicating the minor allele (with entries corresponding to unknown variants labeled NA). In PopGenome, this matrix is stored as part of a *GENOME* object, which contains additional data needed for downstream analyses; this includes information on missing data, as well as annotations for individual SNPs (e.g., if these are transitions/transversions, coding/noncoding, or synonymous/nonsynonymous in the case of coding sequences).

Large input files are split automatically into smaller chunks, with the resulting partial biallelic matrices stored on the hard disk. For this temporary storage, we use ff objects (Adler et al. 2013). These data structures are stored on disk but behave (almost) as if they were in RAM, by transparently mapping only a section (pagesize) in main memory. An ff object needs about 3 kB of RAM to store SNP data from 1 million SNPs across 50 individuals. When analyzing larger data sets, PopGenome concatenates the partial biallelic matrices into a single temporary file.

This strategy allows to simultaneously process whole-chromosome or whole-genome SNP data from hundreds of individuals, as collected in the 1000 human genomes (1000genomes.org) and 1001 Arabidopsis genomes (1001genomes.org) projects. To further speed up the reading process, we employ the R-package parallel (Vera et al. 2008) that facilitates parallel computations on computers with multiple cores/CPUs.

Most functions in PopGenome are implemented in C or C++ to speed up computations and to limit memory requirements. This also applies to the reading of genome-scale alignments and SNP data. Supported alignment formats include FASTA, NEXUS, MEGA, MAF, and Phylip. An almost de facto standard for whole-genome variation data is the Variant Call Format (VCF), used, among others, by the 1000 genomes project (1000genomes.org) and the UK10k project (uk10k.org). PopGenome can read large SNP data sets stored in this format very efficiently, using indexes created with Tabix (Li 2011). SNPs in defined regions can be extracted directly from the corresponding file without time-consuming search operations over the entire file, with input speeds exceeding 6,000 variant positions per second even on older desktop computers.

The implementations of PopGenome's data access functions conform to a set of optimization guidelines to guarantee a minimal reading time. To the best of our knowledge, there is currently no general-purpose population genetics software capable of directly accessing VCF files with comparable speed. Typical times required to read large data sets are listed in table 1.

PopGenome sessions can be saved on hard disk; thus, the conversion to the biallelic matrix needs to be performed only once per data set. The function region.as.fasta() can be used to export data of a specific region, a group of subsites, or the entire data set (i.e., all SNPs or even complete genome alignments) as a FASTA file. A parameter include.unknown indicates if unknown positions should be included in the analyses.

Implemented Methods

To structure the rich landscape of population genetics and genomics analysis methods, PopGenome partitions the implemented methods into modules. Currently, PopGenome provides nine modules (table 2). All modules use the

Table 2. Population Genetics Statistics Implemented in PopGenome's Modules.

Module	Statistics
Neutrality statistics	Tajima's *D* (Tajima 1989), Fu and Li's *F** & *D** (Fu and Li 1993), Fay and Wu's *H* (Fay and Wu 2000), Zeng's *E* (Zeng et al. 2006), Strobeck's *S* (Strobeck 1987), Achaz's *Y* (Achaz 2009), Fu's F_S (Fu 1997), Ramos-Onsins' and Rozas' *R2* (Ramos-Onsins and Rozas 2002), as well as all corresponding theta values
Linkage disequilibrium	ZnS (Kelly 1997), B/Q (Wall 1999), ZA/ZZ (Rozas et al. 2001), and correlation coefficient r^2 for each pair of SNPs within or between windows/regions
Recombination statistics	Four-gamete test (Hudson and Kaplan 1985)
Diversities	Nucleotide and haplotype diversity (Hudson, Boos et al. 1992); (Nei 1979); see "Neutrality statistics" for a list of calculated Theta values
Selective sweeps	CL, CLR (Nielsen et al. 2005)
FST estimates	G_{ST} (Nei 1973); F_{ST} (Hudson, Slatkin et al. 1992); G_{ST}, H_{ST}, K_{ST} (Hudson, Boos et al. 1992); S_{nn} (Hudson 2000); Phi_{ST} (Excoffier and Smouse 1992)
MKT	McDonald–Kreitman test (McDonald and Kreitman 1991)
Mixed statistics	Site frequency spectrum; fixed and shared polymorphisms; biallelic structure
BayeScanR	Bayesian estimation of F_{ST} (Foll and Gaggiotti 2008)

GENOME object created when the input data were read, and store their results in the same GENOME object. For analyses that require to distinguish between ancestral and derived alleles, outgroups can be specified.

As a default, all analyses packaged into one module are performed simultaneously when the module is executed. To accelerate calculations on large data sets, individual methods can be switched off using additional arguments. We plan to integrate more methods in the future, and welcome requests for the implementation of specific statistics. In the next release of PopGenome, we aim to incorporate methods for detecting recent selective sweeps, such as the algorithm implemented in the software SweeD (Pavlidis et al. 2013).

Apart from many "standard" statistics (e.g., neutrality and linkage disequilibrium statistics), PopGenome offers several tests for the detection of nonneutral evolution. So far, we have implemented the McDonald–Kreitmann test (McDonald and Kreitman 1991) and a wide range of FST measurements, including an implementation of a previously published method based on Bayesian statistics (Foll and Gaggiotti 2008). PopGenome also includes a calculation of r^2 correlation coefficients and the corresponding *P* values (Fisher's exact test) for interregion calculations. By concatenating the corresponding GENOME objects, statistics that rely on comparisons between regions can be calculated even when these are located on different chromosomes. Details of the implemented methods are given in table 2 and in the PopGenome documentation.

PopGenome is fully integrated with two widely used coalescent simulation tools: Hudson's MS (Hudson 2002), as well as Ewing's MSMS (Ewing and Hermisson 2010), which incorporates selection. The PopGenome function MS() compares the statistics calculated for the observed data with corresponding data simulated by the coalescent method. PopGenome supports the full coalescent simulation capabilities of MS and MSMS. Parameters can be specified as vectors in the dedicated class "cs.stats," and thus different models (mutation rates, migration rates, etc.) can be applied to different windows or regions of the genome. PopGenome's MS() function stores the calculated statistics of coalescent simulations in a dedicated R object. Direct comparison to coalescent simulations is currently implemented for the modules Neutrality statistics, Linkage, and FST. If statistics from other modules need to be compared with coalescent simulations, PopGenome can directly read MS output files and then process these data using the method of interest (readMS()).

Table 3. Calculation Speed for Haplotype and Nucleotide Diversity in Sliding Windows.

Data	Sliding Window (nucleotides)	Running Time[a]
Arabidopsis (Chr 1)	Window size = 10,000	~30 s
80 individuals	Jump size = 10,000	
1,200,000 SNPs	Number of windows = 3,042	
Human (Chr 2: 100–150 Mb)	Window size = 1,000	~5 min
1,094 individuals	Jump size = 1,000	
660,000 SNPs	Number of windows = 50,000	
3,450 alignments	3,450 windows (alignments)	~7 s
25 individuals		
5,086,953 sites		
200,097 SNPs		

[a]Intel® Core™ i3-2130 CPU @ 3.40 GHz × 4, 8 GB RAM.

R is an efficient environment for large-scale computations. However, although most native R functions are implemented in C or Fortran, R itself is an interpreted and vector-oriented language. As a consequence, some types of calculations tend to be slow when applied to large objects. To avoid major bottlenecks, we implemented several specialized calculations in C ++ . PopGenome finishes the calculation of most statistics in minutes even for very large data sets (table 3).

Partitioning and Interpreting SNPs

Users can restrict analyses to subregions specified either by genomic coordinates or positions in SNP files. When an annotation file in GFF format is present (GFF v2 or v3), PopGenome will automatically label SNPs located in genes, exons, coding regions, and UTRs. Other annotations of interest in the GFF file can be read using the function getgffinfo().

The user can apply the full range of implemented methods to all SNPs observed in a specific class (e.g., all exonic SNPs), or to each region specified in the GFF file separately (e.g., all introns individually).

If a GFF file is present, PopGenome can also classify synonymous and nonsynonymous sites; for SNP data formats, this additionally requires a reference genome in FASTA format. PopGenome stores the codons internally as numerical values, coded from a polynomial function as in the PGE Toolbox (Cai 2008). Based on the GFF file, the function get.codons() will provide information about the nature of amino acid changes resulting from the observed SNPs (encoded amino acids, charges, hydrophobicities, size, and polarity changes). Per default, PopGenome assumes the standard genetic code, but alternative codes can be specified.

Most multipurpose population genetics software tools are geared toward the analysis of discrete loci (Rozas et al. 2003; Excoffier et al. 2005), restricting their utility for the analysis of whole-genome SNP data sets. One widely used approach to apply population genetics methods to whole-genome data is the analysis of sliding windows (Rozas et al. 2001). In PopGenome, users can freely choose window and jump sizes for sliding windows, measured either in nucleotides or in numbers of SNPs. The underlying algorithm copies the information (mostly pointer) stored in the GENOME object to another object of the same class, where the data are reorganized into the specified windows. The full spectrum of PopGenome methods is thus available both for arbitrarily large sets of individual loci and for systematic genomic scans.

Easy Integration of New Methods

To be used by the scientific community at large, a new algorithm ideally has to be implemented in a framework that allows its efficient application to data in the diverse file formats commonly used in different resequencing projects. PopGenome is geared toward making this task as easy as possible. All information required for the calculation of population genetics statistics, including the biallelic matrix as well as genomic annotation, is stored in a GENOME object, which new methods can directly access.

To further simplify the integration of new methods, we implemented the function create.PopGenome.method(), which generates a skeleton of a typical PopGenome function. New methods thus implemented are fully embedded in the PopGenome framework and can be applied to sliding windows or subsites in the same way as the existing modules. This approach frees developers of new population genetics or population genomics algorithms from the need to implement many auxiliary functions, such as efficient data input and output, data conversion, and region subsetting.

To enable PopGenome to work with additional data formats, users can write a simple parser that converts the data to a binary R object. The mechanism to integrate new methods or data formats is documented extensively in a tutorial, accessible by typing "vignette("Integration_of_new_Methods")" in R (see also supplementary file S1, Supplementary Material online).

Results

To illustrate the usage of PopGenome, we show two exemplary analyses for human and *A. thaliana* whole-genome SNP data. More details are found in the PopGenome documentation and in supplementary file S2, Supplementary Material online.

Diversity on *A. thaliana* Chromosome 1

The 1001 genomes project (1000genomes.org) stores all SNP calls from one individual *A. thaliana* plant in one (.SNP) file. We downloaded .SNP files for 80 individuals (Cao et al. 2011) into a subdirectory named "Arabidopsis." After starting R and loading the PopGenome library, we read in the data for chromosome 1 to analyze diversity and population differentiation:

```
> library(PopGenome)
> genome <- readSNP("Arabidopsis," CHR=1)
```

We define the populations as a list of character vectors containing the individuals of each population:

```
> Central_Asia <- c("ICE127," "ICE130," "ICE134,"
"ICE138," "ICE150," "ICE152," "ICE153" , "Sha")
> . . . (analogous for the other populations) . . .
> populations <- list(Central_Asia, Caucasus,
N_Europe, N_Africa, S_Italy, S_Russia, S_Tyrol,
Swabia)
```

The population definitions are now added to the GENOME object:

```
> genome <- set.populations(genome, populations)
```

We then transform these data into consecutive 10-kb-sliding windows:

```
> genome.slide <- sliding.window.transform(genome, width=10000, jump=10000)
```

The nucleotide and haplotype diversities for each window are calculated in the module *diversity.stats*. We store the results also in the GENOME object:

```
> genome.slide <- diversity.stats(genome.slide)
```

The "slot" of the *GENOME* object that stores the nucleotide diversities of the individual populations is called nuc.diversity.within. Slots of such objects are accessed by appending the slot name to the object name, separated by an @ symbol: genome.slide@nuc.diversity.within. These data can be analyzed further using the built-in statistical and graphical capabilities of R. Here, we plot the sliding window nucleotide diversities (fig. 1*A*) with a specialized function:

```
> PopGplot(genome.slide@nuc.diversity.within,
colours)
```

where colors is a vector listing the R names of the colors used for the eight populations in the figure. The haplotype diversity per 10-kb window (fig. 1*B*), as well as Hudson's fixation index, F_{ST} (fig. 1*C*), is produced with similar function calls.

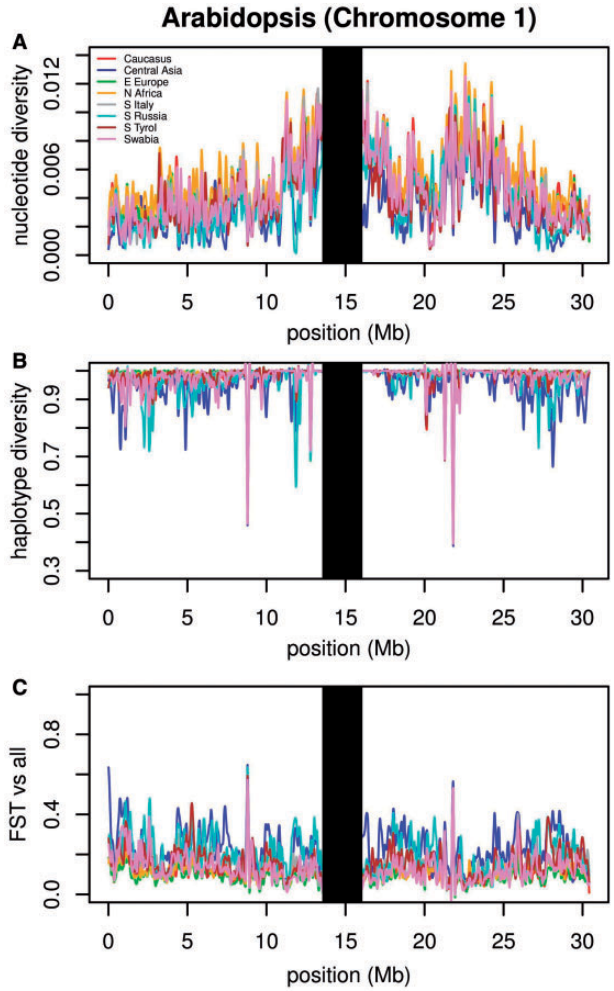


FIG. 1. Diversity statistics for *Arabidopsis thaliana* chromosome 1. Data from the 1001 genomes project website (1001genomes.org) was analyzed in consecutive 10-kb windows. (*A*) Nucleotide diversity, (*B*) haplotype diversity, (*C*) fixation index (Hudson's F_{ST}), contrasting one population against all other individuals. Each line corresponds to one population (see legend in panel [*A*]). Lines were smoothed using spline interpolation. The black bars around 15-Mb mask the centromere.

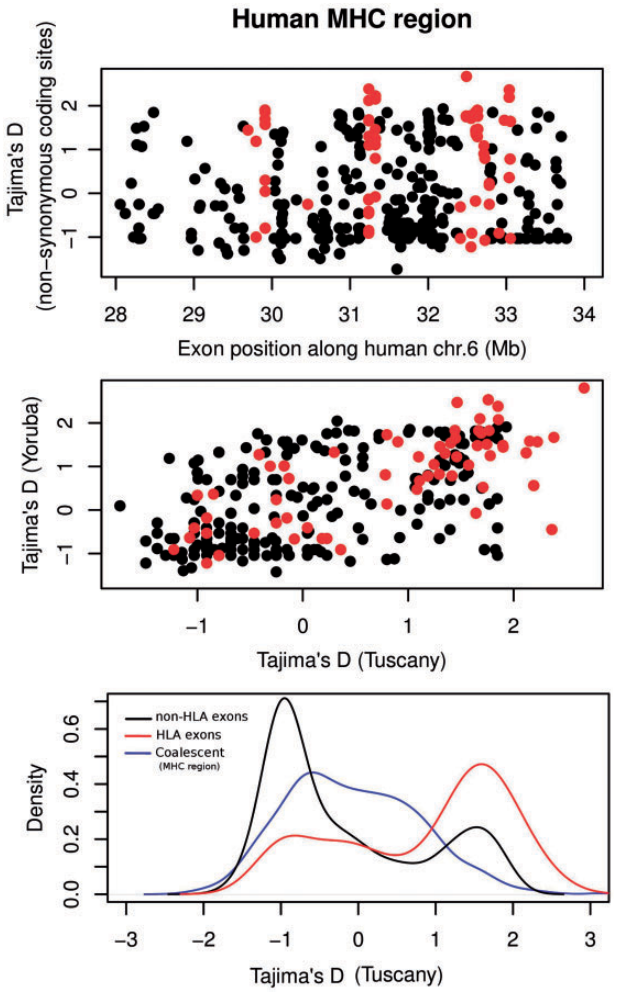


FIG. 2. Tajima's *D* calculated across nonsynonymous coding sites of exons in the human MHC region on chromosome 6. Each data point in (*A*) and (*B*) represents one exon; HLA type I and type II exons are shown in red. (*A*) Tajima's *D* of a Tuscan population (117 individuals), plotted along chr. 6. (*B*) Comparison of Tajima's *D* between a Tuscan (117 individuals) and a Yoruba (229 individuals) population. (*C*) Distribution (density curves) of the Tajima's *D* values in (*A*) for MHC (red) and non-MHC exons (black). The blue curve displays the distribution of neutral values from coalescent simulations with Hudson's MS based on all SNPs in the MHC region. Data from 1000genomes.org.

The data thus displayed in figure 1 indicate a lower level of diversity in the Central Asia population of *A. thaliana* (dark blue lines) along the whole chromosome. As reported earlier (Cao et al. 2011), we observe a dip in diversity in the region around 20 Mb in all populations, suggesting a recent selective sweep in this region.

We find two additional, even stronger candidate regions for selective sweeps around position 8 and 22 Mb. Closer inspection of these two regions shows that they are devoid of polymorphisms between positions 8765643 and 8831390, and between positions 21766562 and 21823063. This observation suggests additional recent, species-wide selective sweeps in these two regions. Furthermore, we find a strong decrease in diversity in plants from Asia and Southern Russia between genomic positions 11870001 and 11900000, suggesting a population-specific selective sweep in this region.

Tajima's *D* across Human MHC Exons

As a second illustration of the PopGenome usage, we calculated Tajima's *D* around the human MHC (Major Histocompatibility Complex) region. The 1000 genomes project stores SNP calls from all examined individuals in gzipped VCF format, together with a Tabix (Li 2011) index (.tbi) file. To read data for the MHC region on human chromosome 6 (including the corresponding annotation in the GFF file), we use the following line:

```
> genome <- readVCF("chr6.vcf.gz," numcols=
10000, tid="6," from=28000000, to=34000000,
gffpath="chr6.gff")
```

where "numcols" is the number of SNPs read in simultaneously, "tid" is a chromosome identifier, "from/to" delimit

Programs	PopGenome	adegenet & pegas (Jombar 2008; Paradis 2010)	DnaSP (Rozas et al. 2003)	Arlequin (Excoffier et al. 2005)	Plink (Purcell et al. 2007)	VariScan (Vilella et al. 2005)
Supported alignment formats	FASTA NEXUS MEGA MAF PHYLIP RData (own)	(own) FASTA NEXUS PHYLIP	FASTA MEGA NBR/PIR NEXUS PHYLIP	(own)	-	MAF MGA XMFA Phylip
Supported SNP data formats	VCF SNP HapMap MS, MSMS	(own) PED	HapMap	(own)	PED(own) VCF HapMap	HapMap
Whole genome data	++	+	-	-	++	+
Neutrality statistics	++	+	++	+	-	++
Linkage disequilibrium	+	-	+	+	+	+
Recombination statistics	+	-	+	+	-	-
FST	++ inc. Bayesian simulation	+	++	+	-	-
MKT	+	-	+	-	-	-
Sweep statistics	+	-	-	-	+	-
Diversity statistics	++	+	+	+	+	+
Sliding windows	++	-	+	-	-	++
Analysis of annotation-derived subsites	++	-	-	-	+	+
Flexible graphical output	++	++	-	-	-	+
Easy integration of new methods	++	+	-	-	-	-
Coalescent Simulation	++	+	+	-	-	-

Fig. 3. Comparison of PopGenome with existing software for population genetics and population genomics analyses. Symbols reflect the breadth of the implemented functionalities: ++, broad; +, limited; −, nonexistent. Details on the criteria used for assignment to the breadth classes are given in supplementary table S1, Supplementary Material online.

the nucleotide positions of the SNPs read in, and "gffpath" specifies the position of the GFF annotation file. Based on the annotations read from the GFF file and the chromosomal reference sequence in FASTA format, the next command labels positions in protein-coding regions as either synonymous or nonsynonymous:

```
> genome <- set.synnonsyn(genome, ref.chr= "chr6.fas")
```

As in the Arabidopsis example, we define the populations via the vectors "Africa" and "Europe," which contain the identifiers of the corresponding individuals:

```
> genome <- set.populations(genome, list(Africa, Europe))
```

Here, we want to calculate statistics for each exon individually, considering only nonsynonymous SNPs. To do this, we first split the region into individual loci, where each locus stores the SNP information from the coding sequence part contained in one exon, as annotated in the GFF file:

```
> genome.exons <- splitting.data(genome, subsites="coding")
```

Next, we calculate Tajima's *D* across all nonsynonymous positions of each exon. This calculation is performed in the module "neutrality.stats":

```
> genome.exons <- neutrality.stats(genome.exons, subsites="nonsyn")
```

We thus obtain the data displayed in figure 2. The high Tajima's *D* values in some loci likely reflect balancing selection (Hedrick 1998), for example, due to frequency-dependent selection in pathogen recognition.

We can use coalescent simulations to derive the expected neutral distribution of Tajima's *D* values across the MHC region. For this, we first calculate Theta across all SNPs in the MHC region:

```
> genome <- neutrality.stats(genome)
```

We then use the genome object as input for a call to Hudson's MS program:

```
> ms <- MS(genome, thetaID="Tajima," neutrality=TRUE)
```

If no additional parameters are specified for the MS simulations, PopGenome will use the standard neutral model (SNM).

The simulated Tajima's *D* values are then extracted:

```
> MS.get.stats(ms)
```

Figure 2*C* compares the distributions of expected Tajima's *D* values under the SNM with values observed for Human Leukocyte Antigen (HLA, red) and non-HLA (black) exons. The distribution of Tajima's *D* values is strongly shifted toward higher values in HLA exons compared with non-HLA exons, indicating strong balancing selection; a deviation from neutral expectations for HLA exons is supported by a comparison to the simulated data.

The highest Tajima's *D* values—suggesting strong balancing selection—are seen in HLA type I and type II genes (marked in red in fig. 2). Tajima's *D* values are correlated between the African (Yoruba) and European (Tuscany) populations (fig. 2*B*; Spearman's $R^2 = 0.33$). One notable outlier is the coding exon 2 of the HLA-DPA1 gene, which shows evidence of purifying selection in Yoruba, but strong evidence of balancing selection in Tuscans.

Discussion

Several computer programs for population genetics or population genomics analyses are publicly available. However, these tend to specialize on specific subsets of analyses (e.g., Vilella 2005; Rozas et al. 2001; Purcell et al. 2007; Jombart 2008; Paradis 2010) or cannot process whole-genome data (e.g., Rozas et al. 2003; Excoffier et al. 2005). Figure 3 compares major features of PopGenome with five other widely used software packages.

PopGenome can not only read data in several common alignment formats but also understands the widest choice of SNP data formats; this includes data from the HapMap as well as the 1000 and 1001 genomes projects. PopGenome's ability to work efficiently with temporary files allows calculations on very large SNP data sets. No other available program offers a similar combination of flexible data input options with a broad toolbox of population genetics and genomics statistics, including the ability to perform analyses in sliding windows.

PopGenome can read GFF annotation files, which permits high plasticity in the variability analysis of different regions of the genome. PopGenome can link this annotation automatically to the SNP data, which can thus be restricted to specific annotated features or feature groups (e.g., all introns vs. all exons). This feature also allows to discriminate synonymous and nonsynonymous codon positions in whole-genome SNP data sets, which is necessary, for example, for McDonald–Kreitman tests. Like adegenet/pegas (which are more limited in scope), PopGenome is fully integrated with the powerful graphical and data analysis capabilities of the R environment (http://cran.r-project.org/, last accessed April 30, 2014), thus simplifying downstream analyses and graphics as well as the development of stable work flows.

Supplementary Material

Supplementary files S1–S3 are available at *Molecular Biology and Evolution* online (http://www.mbe.oxfordjournals.org/).

Acknowledgments

The authors thank Juliette de Meaux, Laura Rose, Kristian Ulrich, Thorsten Klösges, and William Martin for helpful discussions. This work was supported by the Spanish Ministerio de Ciencia e Innovación Grant CGL2009-09346 to S.E.R.O. and by the German Research Foundation DFG grants EXC 1028 and CRC 680 to M.J.L.

References

Achaz G. 2009. Frequency spectrum neutrality tests: one for all and all for one. *Genetics* 183(1):249–258.

Adler D, Gläser C, Nenadic O, Oehlschlägel J, Zucchini W. 2013. ff: memory-efficient storage of large data on disk and fast access functions [R package version 2.2-11]. [cited 2013 Dec]. Available from: http://CRAN.R-project.org/package=ff.

Cai J. 2008. PGEToolbox: a Matlab toolbox for population genetics and evolution. *J Hered.* 99(4):438–440.

Cao J, Schneeberger K, Ossowski S, Günther T, Bender S, Fitz J, Koenig D, Lanz C, Stegle O, Lippert C, et al. 2011. Whole-genome sequencing

of multiple *Arabidopsis thaliana* populations. *Nat Genet.* 43(10):956–963.
Ewing G, Hermisson J. 2010. MSMS: a coalescent simulation program including recombination, demographic structure and selection at a single locus. *Bioinformatics* 26(16):2064–2065.
Excoffier L, Laval G, Schneider S. 2005. Arlequin (version 3.0): an integrated software package for population genetics data analysis. *Evol Bioinform Online.* 1:47–50.
Excoffier L, Smouse PE. 1992. Analysis of molecular variance inferred from metric distances among DNA haplotypes: application to human mitochondrial DNA restriction data. *Genetics* 131(2):479–491.
Fay J, Wu C. 2000. Hitchhiking under positive Darwinian selection. *Genetics* 155(3):1405–1413.
Foll M, Gaggiotti O. 2008. A genome-scan method to identify selected loci appropriate for both dominant and codominant markers: a Bayesian perspective. *Genetics* 180(2):977–993.
Fu Y. 1997. Statistical tests of neutrality of mutations against population growth, hitchhiking and background selection. *Genetics* 147(2):915–925.
Fu Y, Li W. 1993. Statistical tests of neutrality of mutations. *Genetics* 133(3):693–709.
Harrison R. 2012. Understanding genetic variation and function—the applications of next generation sequencing. *Semin Cell Dev Biol.* 23(2):230–236.
Hedrick. 1998. Balancing selection and MHC. *Genetica* 104(3):207–214.
Hudson R. 2000. A new statistic for detecting genetic differentiation. *Genetics* 155(4):2011–2014.
Hudson R, Boos D, Kaplan N. 1992. A statistical test for detecting geographic subdivision. *Mol Biol Evol.* 9(1):138–151.
Hudson R, Kaplan N. 1985. Statistical properties of the number of recombination events in the history of a sample of DNA sequences. *Genetics* 111(1):147–164.
Hudson R. 2002. Generating samples under a Wright–Fisher neutral model of genetic variation. *Bioinformatics Appl Note.* 18(2):337–338.
Hudson R, Slatkin M, Maddison W. 1992. Estimation of levels of gene flow from DNA sequence data. *Genetics* 132(2):583–589.
Jombart T. 2008. adegenet: a R package for the multivariate analysis of genetic markers. *Bioinformatics* 24(11):1403–1405.
Kelly J. 1997. A test of neutrality based on interlocus associations. *Genetics* 146:1197–1206.
Li H. 2011. Tabix: fast retrieval of sequence features from generic TAB-delimited files. *Bioinformatics* 27(5):718–719.
McDonald J, Kreitman M. 1991. Adaptive protein evolution at the Adh locus in *Drosophila*. *Nature* 351(6328):652–654.
Nei M. 1973. Analysis of gene diversity in subdivided populations. *Proc Natl Acad Sci U S A.* 70(12):3321–3323.
Nei M. 1979. Mathematical model for studying genetic variation in terms of restriction endonucleases. *Proc Natl Acad Sci U S A.* 76(10):5269–5273.
Nielsen R, Williamson S, Kim Y, Hubisz M, Clark A, Bustamante C. 2005. Genomic scans for selective sweeps using SNP data. *Genome Res.* 15: 1566–1575.
Paradis E. 2010. pegas: an R package for population genetics with an integrated-modular approach. *Bioinformatics* 26(3):419–420.
Pavlidis P, Zivkovic D, Stamatakis A, Alachiotis N. 2013. SweeD: likelihood-based detection of selective sweeps in thousands of genomes. *Mol Biol Evol.* 30:2224–2234.
Purcell S, Neale B, Todd-Brown K, Thomas L, Ferreira M, Bender D, Maller J, Sklar P, de Bakker PI, Daly MJ, et al. 2007. PLINK: a tool set for whole-genome association and population-based linkage analyses. *Am J Hum Genet.* 81(3):559–575.
Ramos-Onsins S, Rozas J. 2002. Statistical properties of new neutrality tests against population growth. *Mol Biol Evol.* 19(12):2092–2100.
Rozas J, Gullaud M, Blandin G, Aguade M. 2001. DNA variation at the rp49 gene region of Drosophila simulans: evolutionary inferences from an unusual haplotype structure. *Genetics* 158(3):1147–1155.
Rozas J, Sanchez-DelBarrio J, Messeguer X, Rozas R. 2003. DnaSP, DNA polymorphism analyses by the coalescent and other methods. *Bioinformatics* 19(18):2496–2497.
Strobeck C. 1987. Average number of nucleotide differences in a sample from a single subpopulation: a test for population subdivision. *Genetics* 117(1):149–153.
Tajima F. 1989. Statistical method for testing the neutral mutation hypothesis by DNA polymorphism. *Genetics* 123(3):585–595.
Vera G, Jansen R, Suppi R. 2008. R/parallel—speeding up bioinformatics analysis with R. *BMC Bioinformatics* 9:390.
Vilella J, Blanco-Garcia A, Hutter S, Rozas J. 2005. VariScan: analysis of evolutionary patterns from large-scale DNA sequence polymorphism data. *Bioinformatics* 21(11):2791–2793.
Wall J. 1999. Recombination and the power of statistical tests of neutrality. *Genet Res.* 74(1):65–79.
Zeng K, Fu Y, Shi S, Wu C-I. 2006. Statistical tests for detecting positive selection by utilizing high-frequency variants. *Genetics* 174(3):1431–1439.

Table S1.

breadth	++ (broad)	+ (limited)	- (non-existent)
Whole genome data	Whole genomes or whole chromosomes can be efficiently read in without loading everything into RAM, by splitting data into chunks and using temporary files stored on disk. Data needs to be read only once. Whole genome SNP data as well as very large sets of alignments can be processed.	Whole genomes or whole chromosomes can be read in. The software provides memory efficient storage of the data, but data size is limited by the computer's RAM. Reading and calculation times are not optimized.	The data will completely loaded into the RAM before it is analyzed. The software is thus incapable to process whole chromosomes from many individuals as provided, e.g., by the 1000 and 1001 Genomes projects.
Neutrality statistics	The software can perform a wide range of neutrality statistics including those needing an outgroup. Significance tests can be directly performed via simulations in *MS/MSMS*.	Only a few statistics are provided. No flexible framework for significance tests.	No neutrality statistics available.
Linkage disequilibrium	Specialized model based functionalities are provided to improve significance estimates of correlations between sites within the whole genome.	Standard methods like D', r^2 and Kelly's ZnS are implemented.	No linkage disequilibrium based methods are included.
Recombination statistics	Specialized model based functionalities are provided.	Some basic functions to calculate the recombination rate are included.	Not available.
FST	A wide range of FST methods can be applied including specialized model based methods, such as Bayesian approaches. Fixation indices can be calculated also in sliding windows.	The standard FST methods are included.	No possibility to perform population differentiation based methods such as FST.
MKT	The Mcdonald-Kreitman test (MKT) and/or *Ka/Ks* can be calculated for whole genome SNP data as well as sequence alignments with different codon substitution models.	The MKT can be performed for either alignments or SNP data.	The MKT cannot be performed.
Sweep statistics	Specialized (model based) functionalities are provided, such as the full framework implemented in SweepFinder {Sweepfinder}.	Standard methods that are widely used for detecting selective sweeps are implemented. The data can be scanned over user-defined subregions or flexible sliding windows.	No flexible framework to efficiently scan the data for selective sweeps exists.
Diversity statistics	Haplotype and nucleotide based diversity (pi, theta, ...) calculations are provided, which can be calculated within and between the defined populations.	Haplotype and nucleotide based calculations (pi, theta, …) are implemented to calculate the diversities within a population.	No diversity estimates are possible.
Sliding windows	Sliding windows can be defined based on nucleotide counts or based on SNP counts; jump and width sizes are flexible. All methods implemented in the software can be applied to these windows.	Sliding window analysis is possible but only a subset of methods can be applied.	No sliding windows can be analyzed.
Analysis of annotation-derived subsites	The framework is linked to annotation files, e.g., in GFF or GTF format. Information of interest can be extracted from these files. Synonymous and nonsynonymous positions can be verified for sequence data as well as SNP data.	The software can handle additional annotation files, e.g., GFF or GTF, but has limited flexibility to use this information.	No annotation files are supported.
Flexible graphical output	The software is completely linked to highly specialized graphical output environments, and thus allows flexible graphical representations. The framework does not depend on pre-defined graphical functions provided by the software itself.	The software provides a wide range of graphical output functions, but is limited to pre-defined plots.	No graphical output capabilities.
Easy integration of new methods	The software provides a specialized workflow for the effortless implementation of new methods. The internal program structure (e.g., classes and objects) is designed to be easily accessible to users/developers.	It is possible to write own functionalities by linking the software to custom-made scripts.	The program is not written for extensibility.
Coalescent Simulation	Coalescent simulations can be performed with and without selection and can be directly compared with the observed data (also over multiple subregions/windows/alignments).	Coalescent simulations can be performed but are much more limited. The user cannot define the model parameters directly in the framework. The simulations cannot be applied to user-defined subregions or windows of whole genome SNP data.	No coalescent simulations are provided.

4.2 Manuscript 2: (section 2.1)

BASIX: accelerated computations on large vectors and matrices in R
Authors: *Bastian Pfeifer and Martin J. Lercher*

History:
- re-submitted after first round of peer review
- rejected
Journal: *BMC Bioinformatics*
Impact factor: 3.02

Status quo: submitted
Journal: *R journal*
Impact factor: 0.825

Contributions: BP conceived of the project, developed its details, analysed and implemented the algorithms, ran all tests and simulations, and published the R-package BASIX on CRAN. BP and MJL wrote the paper.

BASIX: accelerated computations on large vectors and matrices in R

by Bastian Pfeifer and Martin J. Lercher

Abstract GNU R is a high-level open source interpreted programming language, with particular strengths in statistical computing and graphics. To accelerate computations, most of the embedded functions are internally written in C/C++ and FORTRAN. To ensure their general applicability, these functions are designed for a wide range of different inputs, causing excessive running times for special cases that would lend themselves to faster, specialized algorithms. Due to the data deluge following recent advances in nucleotide sequencing technology, inefficient analysis of large structured vectors and arrays has become an increasingly important bottleneck. This provides a strong incentive to replace slow generic native R functions with faster algorithms that exploit specific data structures. Here, we introduce the R-package **BASIX**, which provides several R functions to accelerate calculations on large structured vectors and matrices. These functions are internally written in C/C++, and either replace generic R functions for specific types of input or provide new functionalities that solve frequently occurring subproblems fast and efficiently. We demonstrate theoretically and through simulations that the **BASIX** functions can accelerate specific, frequently occurring tasks in R. They thus can aid in developing fast methods to process large structured datasets as those produced in re-sequencing projects.

Background

The high-level programming language R (`cran.r-project.org`) is the de facto standard for statistical analyses, and has become an important platform in the analysis of microarray data, as well as in evolutionary genomics, population genomics, and systems biology. R is a functional object-oriented language and is available under the GNU public license for all major operating systems. R functions are based on efficient, compiled C/C++ and FORTRAN code. R also has extensive graphics capabilities to facilitate data interpretation and publication. Since R version 3.0, numeric index values exceeding 2^{31} are possible (`http://cran.r-project.org/src/base/NEWS.html`), allowing the generation and processing of huge objects. These features make R appealing for bioinformaticians and theoretical biologists interested in analysing large datasets fast and efficiently. However, the applicability of R to large structured data such as that produced in re-sequencing projects is limited in two aspects: there is no memory-efficient storage mechanism for large vectors and matrices; and the generic R functions are designed for a wide range of input types, and are thus unnecessarily slow when applied to some types of structured data. Specific R packages exist to partially circumvent these problems (e.g., **ff** (Adler et al., 2013), **bigmemory** (Kane and Emerson, 2013) , **foreach** (Analytics and Weston, 2013) , **parallel** (Vera et al., 2008)). In this article, we introduce the package **BASIX** (Pfeifer, 2013), which is written in C/C++ and provides functions specialized to the acceleration of computationally intensive calculations on large matrices and vectors. As one example, native R functions rarely take into account that specific vectors may already be sorted, and **BASIX** implements accelerations for this case.

Implementation

R provides elegant mechanisms for the integration of compiled code. We use the interface function `.Call` to integrate compiled C code, as it allows C functions to directly allocate memory and enables the use of complex data types (see also the R-package **Rcpp** (Eddelbüttel and Francois, 2011)). A major problem of vector-based languages such as R is that most algorithms are designed to apply calculations to every entry of a vector or matrix, even though this may not be necessary in special cases where a return value can be found from a subset of the entries. In the following, we shortly describe the main functions of the **BASIX** package and compare their runtime complexity to that of the corresponding native R solutions.

BASIX.match(x,y)

`BASIX.match(x,y)` implements a fast version of the native `match()` function for sorted vectors `x,y`. `BASIX.match(x,y)` takes a value from vector `x` and verifies the first match position `i` in the target vector `y`. The next value from `x` will than be tested against the vector `y[i:length(y)]`. The runtime complexities of `BASIX.match()` and native R `match()` are compared in Table 1 for input vectors of lengths `m`, `n`. The best case occurs when the first element of `x` is the last entry in `y`. Our runtime

complexity analysis assumes that the vectors are already sorted. If vectors need to be sorted first, the runtime for sorting ($O(n * log(n))$) must be added, diminishing the gain in speed accordingly.

	Worst case	**Average case**	**Best case**
`BASIX.match(x,y)`	$max\{m,n\} = O(c \cdot n)$ $= O(n)$	$O(c \cdot n) = O(n)$ $c < m/n$	$O(n)$
native R `match(x,y)`	$O(m \cdot n)$	$O(m \cdot n)$ *	$O(m)$

* see Figure 2

Table 1: Runtime complexity of `BASIX.match()` compared to native R `match()` for input vectors of lengths m, n.

BASIX.equal(x,y)

`BASIX.equal(x,y)` provides a fast test for the identity of two vectors `x`, `y`. For this task in R, users may use the function `all(x==y)` (see also Figure 1), where all entries of the vectors will be compared and a vector with `TRUE` or `FALSE`, if the values are equal or not, will be returned to the function `all()`, which then checks if all entries are `TRUE`. This is highly inefficient for the comparison of vectors with more than one mismatch, as the inequality is established after encountering the first occurrence of `FALSE`. Table 2 compares the runtime complexity of `BASIX.equal()` to that of the native R solution for two vectors `x,y` of length n. The `BASIX.equal()` function is a case of so-called "shortcut functions" that are often used in Perl programming; the same strategy is likely useful also for the acceleration of other tasks in R.

	Worst case	**Average case**	**Best case**
`BASIX.equal(x,y)`	$O(n)$	$c \cdot (n/2) \in O(n)$ *	$O(1)$
native R `all(x==y)`	$2 \cdot n = O(n)$	$2 \cdot n = O(n)$	$2 \cdot n = O(n)$

* $\sum_{i=1}^{n} \frac{1}{x^2} \cdot i \leq c \cdot \frac{n}{2}$, i.e., $c = \frac{1}{x^2} < 1$, where x is the number of different numeric values that can be observed. For the average case we assume that the values are independent.

Table 2: Runtime complexity of `BASIX.equal()` compared to the native R solution for input vectors of lengths n.

BASIX.combnapply(x,op)

`BASIX.combnapply(x,op)` accelerates the implementation of nested for loops, which are generically slow in R. `BASIX.combnapply()` extends the class of apply functions, which perform calculations on combinations or permutations of indices of a vector. `BASIX.combnapply()` applies an arithmetical operation provided by the user to all pairs of entries of a vector. For example, `BASIX.combnapply(x,'==')` computes all pairwise differences of entries in vector `x`, a task often performed in population genetics and genomics to measure the nucleotide diversity of DNA sequences. The runtime complexity is the same for **BASIX** and for the native R solution which would combine nested for loops ($(n * (n-1))/2 = O(n^2)$) for input vectors of length n. However, nested for-loops carry a large overhead because R is an interpreted and vector-oriented language, and as a consequence vectorized code will be perform much better in most cases. Our analogous implementation of pairwise comparisons in compiled C-code speeds up the calculations enormously (see Results).

BASIX.unique(x) and BASIX.table(x)

`BASIX.unique(x)` and `BASIX.table(x)` provide fast alternatives/extensions for the native R function `unique()`, which is fast for vectors but slow on matrices. `BASIX.unique()` returns a matrix with duplicated rows removed, saving the corresponding row numbers as row names. `BASIX.table` extends the native R function `table()`, which works only on vectors, by calculating the counts of the unique rows of a matrix. `BASIX.table()` and `BASIX.unique()` can, for example, be used to efficiently calculate haplotype diversities from matrices storing DNA information (see Figure 3 and Table 3).

	Worst case	Average case*	Best case
`BASIX.unique(x)`	$O(n^3)$	$O(n^2 \cdot O(n)) \in O(n^3)$	$O(n^2)$
native R `unique(x)`	$O(n^3)$	$O(n^3)$	$O(n^3)$

* Average case: Rows that have been already visited and classified as non-unique rows will be ignored in the next loop iteration (see also `BASIX.equal()`).

Table 3: Runtime complexity of `BASIX.unique()` (and equivalently, `BASIX.table()`) compared to native R `unique()` for quadratic input matrices `x` with n rows and columns.

BASIX.find.interval(x,from,to)

`BASIX.find.interval(x,from,to)` implements a fast version of the native `findInterval()` function for sorted vectors. `BASIX.find.interval()` identifies the entries of a vector that lie in a user-defined interval faster than the native R function `findInterval()`. The asymptotic runtime complexity is the same for both functions ($O(n)$), but the implementation for sorted vectors is slightly more efficient than the native R version (see Results). `BASIX.find.interval()` can be used, e.g., to scan genomic diversity data with sliding windows; such data is commonly already sorted by position. If vectors need to be sorted first, the runtime for sorting ($O(n * log(n))$) must be added; however, this does not change the overall runtime complexity of `BASIX.find.interval()`.

Results and discussion

To illustrate the run time improvement in comparison with native R functions, we generated vectors and measured running times with the R function `system.time()` (total elapsed). The calculations were performed on a standard PC (Intel® Core™ i3 CPU M 350 @ 2.27GHz $\times$ 4 with 3.8 GB RAM). The analysis of large datasets is frequently done on large computer clusters. However, such clusters are combinations of many processors not unlike those in common desktop computers. Thus, our results obtained on a standard PC scale linearly to such clusters. As seen from Figure 1, `BASIX.equal()` is substantially faster than native R solutions for large vectors and matrices especially when the vectors exceed a size of 2^{20}(1 million) elements. The running times shown on the y-axis are for repeated applications of the functions (1000 equally sized vectors in each case); such repeated applications occur, e.g., when analyzing all genes of a genome, or when performing randomizations to assess statistical significance. As seen from our simulations (Figure 2), the computational acceleration of the `BASIX.match()` function becomes significant when applied to vectors with more than about 2^{15} elements. When the vector size increases at constant sample size (< length of vector) and both vectors `x,y` have the same length, `BASIX.match(x,y)` and `match(x,y)` have the same asymptotical runtime. However, in real world applications such as genome-wide population genetic analyses, where the sample size often represents the number of polymorphic positions, the sample size will always be bigger than the length of the vectors. The `BASIX.unique()` function can be used, for example, to identify

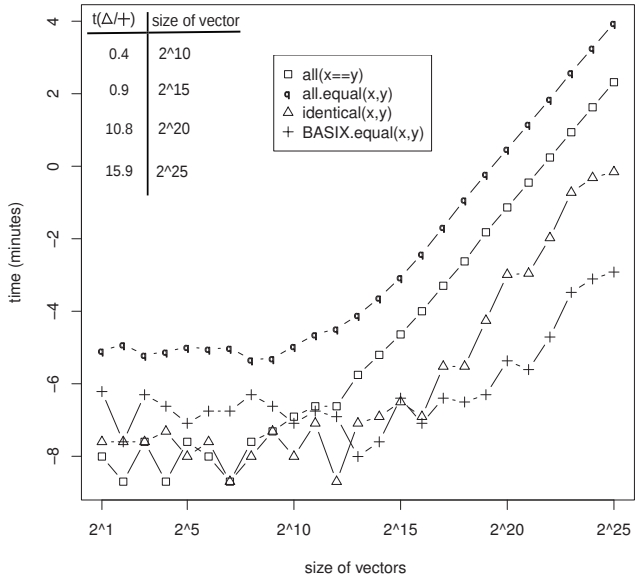


Figure 1: Runtimes for `BASIX.equal()` and native R alternatives. We randomly sampled values from $\{0,1\}$ for two binary vectors `x` and `y` of equal length n. We then sorted these vectors to ensure that the mismatch will not occur at the very beginning (in which case the average asymptotic running time of `BASIX.equal()` would be constant, and its advantage correspondingly trivial; however, this type of situation may often occur in practice). We show runtimes for repeated application (1,000 times in each case) for exponentially increasing vector lengths n, comparing the performance of `BASIX.equal()` with the native R functions `all(...==...)`, `all.equal()` and `identical()`. The table in the inset shows the ratio of runtimes for `all(...==...)` vs. `BASIX.equal()`. We use a logarithmic scale for the y-axis.

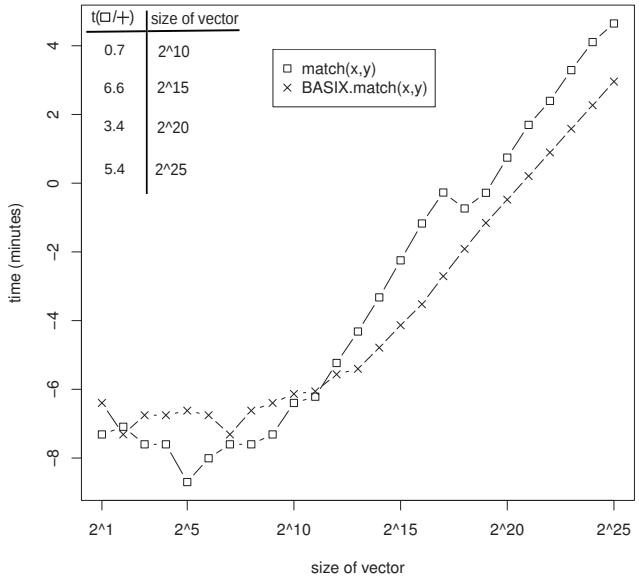


Figure 2: Runtimes for `BASIX.match()` and the native R alternative `match()`. We produced vectors `x` and `y` of exponentially increasing lengths n, each time filled by randomly sampling $2n$ values out of $\{1...n\}$ with replacement and then sorting `x` and `y`. We show runtimes for repeated application (1,000 times in each case). The table in the inset shows the ratio of runtimes for `all(...==...)` vs. `BASIX.equal()`. We use a logarithmic scale for the y-axis.

identical haplotypes in population genomics analyses. In times of high-throughput re-sequencing projects covering thousands of individuals, such as the 1000 genomes project (McVean, 2012), this type of acceleration becomes an important issue in data analysis. We integrated `BASIX.unique()` (along with several other **BASIX** functions) with the R-package **PopGenome** (Pfeifer et al., 2014) and used it to calculate haplotype diversities for sliding windows of human chromosome 6 across 350 European individuals resequenced in the HapMap project (`www.hapmap.org`) (HapMap, 2005). The calculation with `BASIX.unique()` was 8 times faster ($\sim$ 5 minutes) than that using the native R version ($\sim$ 40 minutes). As seen in Figure 2, the haplotype counts decreases especially in the MHC-region, signifying lower diversity in this region. To validate the accelerated computation of `BASIX.combnapply()`, we

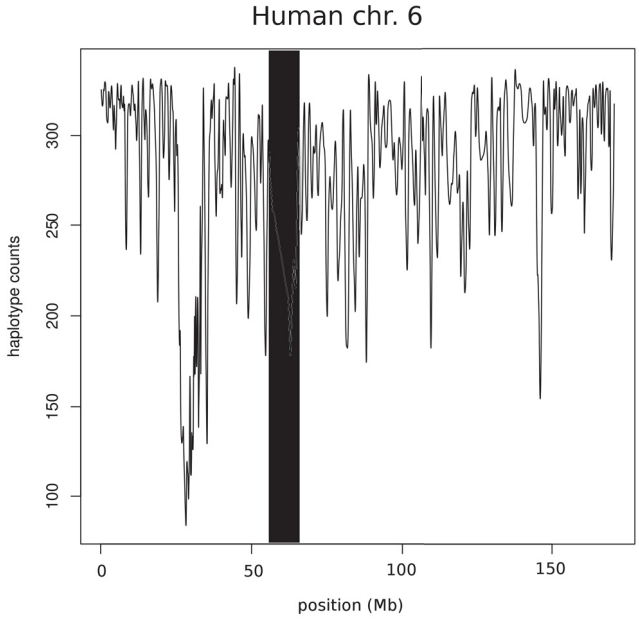


Figure 3: Haplotype counts along human chromosome 6, calculated using `BASIX.unique()` integrated with the R package **PopGenome**. We scanned the HapMap data for 350 European individuals with a window size of 100 SNPs and a jump size of one SNP, producing about 64,000 windows. To each window, we applied `BASIX.unique()` to verify the total number of unique sequences in each window. Using `BASIX.unique()`, the calculation takes 5 minutes, while the native R-function needed about 40 minutes.

generated a vector of length 10,000 which leads to a total number of pairwise comparisons of $\sim$ 50,000,000. We than applied a function written in native R written with nested for-loops, which took about 5 minutes of computer time. After that we used the function `cmpfun()` from the R-package **compiler**, which provides an interface to a byte-code compiler for R. The computation needed about one minute. `BASIX.combnapply()` completed the same computation after 0.84 seconds. To test the improvement in runtimes obtained with `BASIX.find.interval()`, we generated a vector of size 2^{20} by sampling out of $\{1, ..., 2^{20}\}$ with replacement, and sorted the vector afterwards (note that in the intended applications, the vectors will already be sorted). We than sampled randomly two values out of this vector to define an interval. We repeated the simulation 1000 times. `BASIX.find.interval()` was on average three times faster (in total 4 seconds) than the native R version `findInterval()`.

Usage

To illustrate the use of **BASIX** functions, we list short examples for the three functions `BASIX.combnapply()`, `BASIX.table()` and `BASIX.unique()`. Almost all functions in **BASIX** can be viewed as extensions or replacements of existing R functions, and their usage will thus be familiar to R users.

```
> install.packages("BASIX")
> library(BASIX)

# BASIX.combnapply
> vec <- 1:5
```

```
> BASIX.combnapply(vec,"+")
 [1] 3 4 5 6 5 6 7 7 8 9

# BASIX.table
> mat <- matrix(c(1,2,1,2,2,2), ncol=2)
> mat
        [,1]        [,2]
[1,]        1        2
[2,]        2        2
[3,]        1        2

> BASIX.table(mat)
rows         1 2
counts        2 1

# BASIX.unique
> mat <- matrix(sample(c(0,1),10000000,replace=TRUE),nrow=10000,ncol=1000)
> system.time(unique(mat))
15.537(user) 0.144(system) 15.721(elapsed)
> system.time(BASIX.unique(mat))
1.600(user) 0.088(system) 1.691(elapsed)
```

Discussion

R is designed to handle a wide range of different data types organized in objects such as lists and data frames. This design ensures flexible programming and enables fast software development but at the same time produces computational overheads, e.g., caused by the need to check data types prior to performing calculations. However, data structures storing only one type of data, such as matrices or vectors, can be processed much faster. Accordingly, to accelerate computations, genomic data should be organized as much as possible in simple data structures, and we have thus concentrated here on matrices and vectors, which can store numeric as well as character values. Bioinformaticians who are interested in using large data tables should also have a look at the CRAN package **data.table** (Dowle et al., 2013). In this paper, we presented several new functions contributed by the R-package **BASIX** to speed up specific calculations in R. The computational overhead of related native R-functions has diverse reasons. `BASIX.combnapply()`, for example, which also provides a new function for a widely used functionality, was introduced as a wrapper to compiled C code because the computational overhead in R here preliminary comes from the R's interpreted language design. In addition, R is optimized for vector-based programming, and it is sometimes hard to vectorize such structures. Other **BASIX** functions like `BASIX.unique()` are introduced to directly solve the computational costs arising in vector-oriented languages. Finally, we integrated functions like `BASIX.match()` optimized on already sorted vectors; this important special case has rarely received attention in R.

Conclusions

BASIX includes a set of functions that seamlessly integrate with R and that accelerate some commonly used computations substantially when applied to very large vectors and matrices. These functions will be particularly useful, e.g., in population genomics studies that need to process large vectors and matrices of polymorphism (SNP) data. They thus can aid in developing fast methods for the processing of large structured datasets as those produced in re-sequencing projects in R. R has the potential to become widely used for large-scale genomic data analysis projects, aided by the R tradition of publicly available libraries contributed by academic scientists; **BASIX** may accelerate the development and runtime of some of such future contributions.

Availability and requirements

BASIX (version 1.1) is freely available from CRAN (`www.r-project.org`), which provides binary versions for all major computer platforms (Linux, Mac OS X, Windows, and Solaris). A manual describing usage and providing detailed accounts of the underlying algorithms can also be downloaded from CRAN. We encourage any users to share their experiences with the authors to contribute to the extension of **BASIX**.

Acknowledgements

We thank Gabriel Gelius-Dietrich and Ulrich Wittelsbürger for helpful discussions. MJL acknowledges financial support from the German Research Foundation (DFG grants EXC 1028 and CRC 680).

Bibliography

D. Adler, C. Gläser, O. Nenadic, J. Oehlschlägel, and W. Zucchini. *ff: memory-efficient storage of large data on disk and fast access functions*, 2013. URL http://CRAN.R-project.org/package=ff. R package version 2.2-11. [p1]

R. Analytics and S. Weston. *foreach: Foreach looping construct for R*, 2013. URL http://CRAN.R-project.org/package=foreach. R package version 1.4.1. [p1]

M. Dowle, T. Short, S. L. with contributions from A Srinivasan, and R. Saporta. *data.table: Extension of data.frame for fast indexing, fast ordered joins, fast assignment, fast grouping and list columns.*, 2013. URL http://CRAN.R-project.org/package=data.table. R package version 1.8.10. [p6]

D. Eddelbüttel and R. Francois. Rcpp: Seamless R and C++ Integration. *Journal of Statistical software*, 40(8), 2011. [p1]

HapMap. International Human Genome Sequencing and International Hapmap Consortium: A haplotype map of the human genome. *Nature*, 437, 2005. [p5]

M. J. Kane and J. W. Emerson. *bigmemory: Manage massive matrices with shared memory and memory-mapped files*, 2013. URL http://CRAN.R-project.org/package=bigmemory. R package version 4.4.3. [p1]

McVean. An integrated map of genetic variation from 1,092 human genomes. *Nature*, 491, 2012. [p5]

B. Pfeifer. *BASIX: An efficient C/C++ toolset for R.*, 2013. URL http://CRAN.R-project.org/package=BASIX. R package version 1.1. [p1]

B. Pfeifer, U. Wittelsbürger, S. E. Ramos-Onsins, and M. J. Lercher. PopGenome: A swiss army knife for population genomic analyses in R. *Molecular Biology and Evolution [epub ahead of print]*, 2014. [p5]

G. Vera, R. Jansen, and R. Suppi. R/parallel–speeding up bioinformatics analysis with R. *BMC Bioinformatics*, 9, 2008. [p1]

Bastian Pfeifer
Institute for Computer Science, Heinrich Heine University
Universitätsstr. 1, 40225 Düsseldorf
Germany Bastian.Pfeifer@uni-duesseldorf.de

Martin J. Lercher
Institute for Computer Science, Heinrich Heine University
Universitätsstr. 1, 40225 Düsseldorf
Germany lercher@cs.uni-duesseldorf.de

4.3 Manuscript 3: (section 2.2)

WhopGenome: high-speed access to whole genome variation and sequence data in R

Authors: *Ulrich Wittelsbürger, Bastian Pfeifer and Martin J. Lercher*

Status quo: submitted
Journal: *Bioinformatics*
Impact factor: 5.323

Contributions: BP conceived of the project. BP and UW developed the project details. UW implemented the algorithms, ran all tests and simulations, and published the R-package WhopGenome on CRAN. UW and BP incorporated the core functionality of WhopGenome into the population genomic software PopGenome. UW and MJL wrote the paper.

WhopGenome: high-speed access to whole genome variation and sequence data in R

Journal:	*Bioinformatics*
Manuscript ID:	BIOINF-2014-0840
Category:	Applications Note
Date Submitted by the Author:	13-May-2014
Complete List of Authors:	Wittelsbürger, Ulrich; Heinrich Heine University, Department for Computer Science Pfeifer, Bastian; Heinrich Heine University, Department for Computer Science Lercher, Martin; Heinrich Heine University, Department for Computer Science
Keywords:	Population genetics, Genome analysis, Genome annotation, Sequence analysis, SNPs

BIOINFORMATICS

Vol. 00 no. 00 20XX
Pages 1–3

WhopGenome: high-speed access to whole genome variation and sequence data in R

Ulrich Wittelsbürger [1], Bastian Pfeifer [1] and Martin J. Lercher [1,2*]

[1]Institute for Computer Science, Heinrich Heine University, D-40255 Düsseldorf, Germany
[2]Cluster of Excellence on Plant Sciences CEPLAS, Düsseldorf, Germany

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Associate Editor: XXXXXXX

ABSTRACT

Summary: The statistical programming language R has become a *de* facto standard for the analysis of many types of biological data, and is well suited for the rapid development of new algorithms. However, variant call data from population-scale resequencing projects is typically too large to be read and processed efficiently with R's built-in I/O capabilities. WhopGenome can efficiently read whole genome variation data stored in the widely used VCF file format into several R data types. VCF files can be accessed either on local hard drives or on remote servers. WhopGenome can associate variants with annotations such as those available from the UCSC genome browser, and can streamline the reading process by filtering loci according to user-defined criteria. WhopGenome can also read other Tabix-indexed files, and can create indices to allow fast selective access to FASTA-formatted sequence files.

Availability: The WhopGenome R package is available on CRAN at http://cran.r-project.org/web/packages/WhopGenome/

Contact: Martin J. Lercher, lercher@cs.uni-duesseldorf.de

1 INTRODUCTION

Population-scale whole genome sequencing projects produce information on SNPs, InDels, and structural variations across thousands of individuals. These projects commonly use the Variant Call Format (VCF) (1000 Genomes Project Analysis Group, 2011) text files for data storage. The resulting files often contain millions of variant sites and may fill tens of gigabytes. The environment for statistical computing R (R Core Team, 2013) has established itself as a *de*-facto standard for general statistics and for the analysis of different types of sequencing data, and has efficient functions to process large vectorized data. However, routinely reading gigabyte-sized VCF files into R is not realistic with R's built-in I/O capabilities.

VCF files are typically compressed and indexed with Tabix (Li, 2011). Tabix compresses appropriately formatted data files and produces an accompanying index file. The index can be used to quickly locate, decompress, and extract selected portions of the data. While several R packages are capable of reading VCF files (VariantAnnotation (Obenchain *et al.*, 2014), seqminer[1], Rplinkseq[2]), these either lack the desirable speed, ease of use, or completeness of support for Tabix files. Further, these implementations post-process the text returned by Tabix in R, which incurs a sizable overhead especially for repeated and large-scale processing.

Here, we present WhopGenome, an R package for fast, straightforward, and flexible processing of genomic variation data in VCF format. WhopGenome is also capable of compressing files with BGZF and indexing suitably formatted files with Tabix, allowing efficient selective access to the data stored in these files. This data needs to be organized into entries identifiable via unique index pairs: a group name (e.g., a chromosome) and a number (e.g., a chromosomal position). With WhopGenome's generic Tabix interface, users can process, for example, GFF or BED files in order to access them efficiently from within R.

The same selective access functionality exists also for FASTA files through WhopGenome's interface to FaIdx (the indexing solution included in samtools) (1000 Genome Project Data Processing Subgroup, 2009). Using this interface to pre-process FASTA files facilitates quick selective retrieval of DNA or amino acid sequence regions. WhopGenome can thus efficiently integrate information from associated sequence, genome annotation, and population-scale variation files for a given chromosomal region for joint processing.

2 FEATURES & IMPLEMENTATION

All indexed data files can reside either on local hard disks or on remote HTTP or FTP servers. Thus, WhopGenome can, for example, selectively read data from the 1000 Genomes Project directly from the NCBI servers [3].

WhopGenome provides functionality to relate genomic loci to annotation data. A wide range of different annotation data types is accessible through the UCSC Genome Browser (Kent *et al.*, 2002), the AmiGO Gene Ontology database (Carbon *et al.*, 2009), and through BioConductor's (Gentleman *et al.*, 2004) org.XX.eg.db annotation packages. WhopGenome includes user-friendly interfaces to the UCSC Genome Browser and the AmiGO servers, vastly simplifying the construction of the necessary SQL queries and the communication with the remote servers. WhopGenome also provides a comfortable way to select and download the BioConductor annotation packages.

*to whom correspondence should be addressed

[1] http://cran.r-project.org/web/packages/seqminer/

[2] https://atgu.mgh.harvard.edu/plinkseq/

[3] ftp://ftp-trace.ncbi.nih.gov/1000genomes/ftp/release

Table 1. Comparison overview

	WhopGenome	SeqMiner	VariantAnnotation
ms/SNP[1]			
- single SNP	4.88	9.46	330.39
- list of 1000	0.16	0.62	1.15
- matrix of 1000	0.07	0.60	1.27
Est. total time for entire file			
- one by one	4h 1m	7h 47m	11d 7h 51m
- list of 1000	7m 53s	30m 36s	56m 51s
- matrix of 1000	3m 27s	29m 37s	1h 2m
Pre-filtering	Y	N	N
Result formats[2] R,V,M,L	Y,Y,Y,N	Y,N,N,Y	N,N,Y[3],Y
Read via HTTP/FTP	Y	N	Y
Create indices	Y	Y[4]	N[5]

[1] Average time in milliseconds per SNP when reading 100,000 SNPs
[2] Directly supported result formats (without additional calls)
[3] An extra R function call extracts the genotype matrix from the result
[4] SeqMiner can create indices on compressed files; WhopGenome compresses, too
[5] Not specifically for VCF files

To link genomic variation to pedigree data, WhopGenome includes support for .PED files (a simple text-based table format used, e.g., by PLINK). Users can load this data into a matrix, modify it, save it back, and locate individuals with certain family relationships. This is mainly useful for selecting samples or correlating them with phenotypes, populations, or other information.

When reading from VCF files with WhopGenome, a typical workflow would be as follows. The function VCF_open() creates a handle to the VCF file. This handle is required to select samples (individuals), genomic regions, and filtering steps, as well as for reading data.

Users can choose to get their results in a variety of R data types. Besides reading each data field independently, it is also possible to read only information on single nucleotide polymorphisms (SNPs) and store the data fields for each SNP in a vector. Especially useful for sliding window analyses are the matrix variants, which can return SNP genotypes in 4 different representations, either numeric or textual. In order to maximize speed gains, we wrote a dedicated read function for each result format. If specific areas of research would benefit from additional data representations in R, we will implement these in future versions of WhopGenome.

After setting a region by specifying a chromosome (or contig) and start and end positions, the next read call will return data for the first variant within that region. Active filters will exclude lines depending on a list of user-defined rules. Rules are specified with simple function calls in R, but are run in compiled C++.

The .PED file support for pedigrees, Gene ontology queries, UCSC Genome Browser database queries, and Bioconductor genome annotation support is implemented in R. All time-critical code is written in C/C++. To avoid losing time by allocating memory, many read functions expect an R variable as a parameter in which to store the results. This improves speed dramatically especially if the data is read into matrices.

3 EVALUATION

WhopGenome is able to parse about 9,000 variant sites per second from a typical VCF file on a current PC (Intel Core i7, 32GB RAM, Linux Mint 16 64-bit, Kernel 3.11.0-12). We compared WhopGenome in terms of speed, features, and ease of use to two other R packages that make use of Tabix: SeqMiner and VariantAnnotation (Table 1). We did not make a comparison to Rplinkseq, as Rplinkseq could not be compiled without manual code changes, and because using Rplinkseq requires extensive manual interaction with the external PLINK software (Purcell *et al.*, 2007).

As reference file, we chose the 1000 Genomes Project's chromosome 1 consensus VCF file, describing over 2.9 million variants in 1094 individuals, stored in 49 gigabytes of text, compressed down to 1.4 gigabytes. SeqMiner expects special non-standard annotation in the VCF files, which is not present in the 1000 Genomes Project files. We thus ran all benchmarks on the same, pre-processed file for better comparability (to pre-process the input file for SeqMiner, we needed to uncompress the original file and install additional software).

Although all three packages rely on the Tabix library, their usage and feature sets differ substantially. VariantAnnotation was the slowest program by a large margin, while WhopGenome was the fastest (Table 1). All programs provide matrix representations of genotypes, but only WhopGenome offers four alternative forms of matrices. WhopGenome's pre-filtering capabilities have no equivalent in the other packages. With regards to the learning curve, we consider our solution to be easier to understand than VariantAnnotation, while SeqMiner's limited feature set makes its usage somewhat simpler, but also much less powerful.

The core functionality of WhopGenome has been successfully used by the population genomics software PopGenome (Pfeifer *et al.*, 2014), which implements a broad range of population genetics analyses for individual loci, sliding windows, and genomic feature sets such as exons.

Besides its ability to efficiently read VCF and other Tabix-indexed files, WhopGenome can also index and access FASTA-formatted sequence files efficiently. With its auxiliary feature set covering pedigree, genome annotation, and fast pre-filtering, we expect WhopGenome to substantially accelerate the development and application of genomic and population genomic analyses in R.

ACKNOWLEDGEMENT

WhopGenome makes use of Tabix and FaIdx by Heng Li, of bgzf written by Bob Handsaker and modified by Heng Li, and of zlib by Jean-loup Gailly and Mark Adler.

Funding: This work was supported by the German Research Foundation [DFG grants EXC 1028 and CRC 680 to M.J.L.].

Conflict of Interest: none declared.

REFERENCES

1000 Genome Project Data Processing Subgroup (2009). The sequence alignment/map format and samtools. *Bioinformatics*, **25**(16), 2078–2079.

1000 Genomes Project Analysis Group (2011). The variant call format and vcftools. *Bioinformatics*, **27**(15), 2156–2158.

Carbon, S., Ireland, A., Mungall, C. J., Shu, S., Marshall, B., Lewis, S., the AmiGO Hub, and the Web Presence Working Group (2009). Amigo: online access to ontology and annotation data. *Bioinformatics*, **25**(2), 288–289.

Gentleman, R. C., Carey, V. J., Bates, D. M., Bolstad, B., Dettling, M., Dudoit, S., Ellis, B., Gautier, L., Ge, Y., Gentry, J., *et al.* (2004). Bioconductor: open software development for computational biology and bioinformatics. *Genome biology*, **5**(10), R80.

Kent, W. J., Sugnet, C. W., Furey, T. S., Roskin, K. M., Pringle, T. H., Zahler, A. M., and Haussler, D. (2002). The human genome browser at ucsc. *Genome Research*, **12**(6), 996–1006.

Li, H. (2011). Tabix: fast retrieval of sequence features from generic tab-delimited files. *Bioinformatics*, **27**(5), 718–719.

Obenchain, V., Lawrence, M., Carey, V., Gogarten, S., Shannon, P., and Morgan, M. (2014). Variantannotation: a bioconductor package for exploration and annotation of genetic variants. *Bioinformatics*, page doi:10.1093/bioinformatics/btu168 (epub ahead of print).

Pfeifer, B., Wittelsbürger, U., Ramos-Onsins, S. E., and Lercher, M. J. (2014). Popgenome: An efficient swiss army knife for population genomic analyses in r. *Molecular biology and evolution*, page doi:10.1093/molbev/msu136 (epub ahead of print).

Purcell, S., Neale, B., Todd-Brown, K., Thomas, L., Ferreira, M. A., Bender, D., Maller, J., Sklar, P., De Bakker, P. I., Daly, M. J., *et al.* (2007). Plink: a tool set for whole-genome association and population-based linkage analyses. *The American Journal of Human Genetics*, **81**(3), 559–575.

R Core Team (2013). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

4.4 Manuscript 4: (chapter 3)

GeneFeST: Bayesian calculation of gene-specific F_{ST} from genomic SNP data

Authors: *Bastian Pfeifer, Stefan Habenschuss and Martin J. Lercher*

Status quo: under review
Journal: *Molecular Biology and Evolution (MBE)*
Impact factor: 10.353

Contributions: BP and MJL conceived of the study and developed the project. BP designed and implemented the algorithms, ran all tests and simulations, and published the R-package GeneFeST on CRAN. SH contributed to the section "Improving discrimination performance by controlling average P-values". BP and MJL wrote the paper.

GeneFeST: Bayesian calculation of gene-specific FST from genomic SNP data

Journal:	*Molecular Biology and Evolution*
Manuscript ID:	MBE-14-0444
Manuscript Type:	Article
Date Submitted by the Author:	29-Apr-2014
Complete List of Authors:	Pfeifer, Bastian; Institute for Computer Science, Heinrich Heine University Düsseldorf Habenschuss, Stefan; Institute for Theoretical Computer Science, University of Technology Lercher, Martin; Heinrich Heine University, Institute for Computer Science
Key Words:	Fixation index, Genome scan method, Bayesian inference

GeneFeST: Bayesian calculation of gene-specific F_{ST} from genomic SNP data

Bastian Pfeifer[1], Stefan Habenschuss[2] & Martin J. Lercher[1,3]*

[1] Institute for Computer Science, Heinrich Heine University Düsseldorf, Germany

[2] Institute for Theoretical Computer Science, Graz University of Technology, Austria

[3] Cluster of Excellence on Plant Sciences CEPLAS, Düsseldorf, Germany

*corresponding author:

Martin J. Lercher

Phone: +49 211 81-10546

Email: lercher@cs.uni-duesseldorf.de

Abstract

The fixation index F_{ST}, which measures population differentiation, can be used to identify non-neutrally evolving loci from genome-scale SNP data across two or more populations. Recent Bayesian approaches to estimate F_{ST} based on Markov-Chain Monte-Carlo simulations were not originally developed for SNP data, and cannot account reliably for the co-evolution of linked SNP positions. Here, we present a new Bayesian approach that assigns identical locus-specific effects to all SNPs located in predefined blocks; a typical application is the identification of genes under balancing or directional selection. Through extensive simulations, we show that the new method is superior to previous moment-based and Bayesian approaches. We also show that haplotype-based F_{ST} estimates show systematic biases, and that the posterior probabilities of non-neutral evolution provided by some Bayesian approaches are inaccurate indicators of selection. An R implementation of our method, which builds on the powerful population genetics and genomics software PopGenome, is available freely from CRAN.

Introduction

Wright (1950) introduced the fixation index F_{ST} as a method to measure the inbreeding effect of population subdivision. In coalescent theory, F_{ST} is now more often discussed as $1 - F$, where *F* is the probability that two randomly chosen individuals are identical by descent. Accordingly, F_{ST} can be considered as an index of genetic differentiation, where small values indicate low genetic differentiation between populations, while high values indicate larger genetic differentiation. F_{ST} is commonly defined as $F_{ST} = (H_T - H_S) / H_T$, where H_T is total heterozygosity, and H_S is subpopulation heterozygosity. F_{ST} has a theoretical minimum of 0 and a theoretical maximum of 1. In addition to F_{ST}, a wide range of related *F*-statistics has been developed over the past decades to better account for the effects of sampling only limited numbers of subpopulations and individuals (Hartl and Clark 2007; Hudson, et al. 1992; Weir 1996; Weir and Cockerham 1984).

Different types of F_{ST} measurements can be broadly classified into methods based on haplotypes and sequence-based methods that consider individual nucleotide differences. At small sample sizes, many or most haplotypes will occur only once in a given dataset except when haplotype diversity is very low, and hence haplotype-based methods have low power to detect non-neutral evolution (Hudson 2000). Sequencing errors may artificially increase the number of haplotypes, exacerbating this effect. Furthermore, in genome-scale F_{ST} analyses, the number of haplotypes per gene will depend on haploblock length, causing systematic biases in haplotype-based F_{ST} measurements. Thus, in many situations, nucleotide-based F_{ST} methods should be preferred (Riebler, et al. 2008).

Analyses that calculate F_{ST} values frequently aim to separate neutral contributions to F_{ST} caused by population histories from gene-specific contributions due to non-neutral evolution. However, different populations have different histories and are influenced by different factors, such as mutations, genetic drift, and migration (Kronholm, et al. 2010). As a consequence, the neutral distribution of F_{ST} values strongly depends on the organisms and populations considered. The separation between non-neutral and neutral (population history-based) F_{ST} contributions can be made explicit in Bayesian approaches that employ Markov-Chain Monte-Carlo (MCMC) methods to estimate F_{ST} (Beaumont and Balding 2004; Foll and Gaggiotti 2008; Riebler, et al. 2008).

The new method introduced in this paper is based on the work of Beaumont & Balding (2004), which establishes an F_{ST}-based hierarchical Bayesian model to detect loci that evolve non-neutrally. This Bayesian approach uses a logistic regression model to distinguish

between locus-specific effects (selection) and population-specific effects (reflecting the common history of all loci). Foll & Gaggiotti (2008) and, independently, Riebler *et al.* (2008) extended this work through a reversible jump MCMC method (Green 1995), which calculates an explicit posterior probability for non-neutral evolution at each locus. The software BayeScan (http://cmpg.unibe.ch/software/BayeScan) implements this approach (Foll and Gaggiotti 2008).

The new method introduced here is an extension of BayeScan and related Bayesian methods for the analysis of population differentiation. Whereas previous algorithms allowed the analysis of either haplotypes or of individual single nucleotide polymorphisms (SNPs), we propose to group neighboring SNPs that likely experienced the same evolutionary pressures into blocks, *e.g.*, by gene or by sliding window. This joint analysis of co-evolving SNPs increases the statistical power of our approach, allowing us to more reliably detect directional and in particular balancing selection.

A logistic regression model for F_{ST}

At the heart of the Bayesian estimate of F_{ST} lies a logistic regression, first formulated by Beaumont & Balding (Beaumont and Balding 2004; Foll and Gaggiotti 2008):

(1) $$log(\frac{F_{ST}^{ij}}{1-F_{ST}^{ij}}) = log(\frac{1}{\theta_{ij}}) = \alpha_i + \beta_j$$

(2) $$\theta_{ij} = exp(-(\alpha_i + \beta_j))$$

Here, F_{ST}^{ij} measures the differentiation at locus i between subpopulation j and the ancestral population. In the Bayesian approach, F_{ST}^{ij} and θ_{ij} are not calculated for each individual locus using moment estimators, but are instead defined in terms of two independent parameters, α_i and β_j. α_i is a parameter specific to locus i (capturing locus-specific deviations from neutral evolution), while β_j is a parameter specific to population j (capturing locus-independent effects, i.e., consequences of population history).

The β_j are determined such that the average α_i (assumed to be neutral) is zero in each population. Thus, negative α_i correspond to F_{ST}^{ij} values below the genomic average (indicating that locus i is under balancing selection), while positive α_i correspond to above-average F_{ST}^{ij} (indicating population-specific directional selection at this locus).

Estimation of gene- and population-specific parameters via MCMC

BayeScan (Foll and Gaggiotti 2008) uses an MCMC approach directly based on the ideas in (Balding and Nichols 1995) and (Beaumont and Balding 2004). However, these simulations do not provide the final output of BayeScan. Instead, they represent "pilot runs" (or burn-in simulations) that determine parameters for the second phase of the algorithm, the application of the reversible jump model. For every locus i, BayeScan first calculates the total counts of each allele k over all populations j, and generates a corresponding set of allele frequencies of the ancestral population $p_i = (p_{ik})$ out of a Dirichlet distribution.

Given a specific parameterization of the model (*i.e.,* ancestral allele frequencies p_i, locus-specific effect α_i, and population-specific effect β_j), the likelihood of the observed allele distribution at locus i and population j, a_{ij}, can be calculated as (Foll and Gaggiotti 2008):

(3) $$P(a_{ij}|p_i, \alpha_i, \beta_j) = \frac{n_{ij}!\Gamma(\theta_{ij})}{\Gamma(n_{ij}+\theta_{ij})} \prod_{k=1}^{K_i} \frac{\Gamma(a_{ijk}+\theta_{ij}p_{ik})}{a_{ijk}!\Gamma(\theta_{ij}p_{ik})}$$

Here, n_{ij} is the sample size at locus i of population j, and a_{ijk} is the number of alleles of type k observed at locus i in the sample from subpopulation j ($k = 1,2$ in case of bi-allelic SNP data). The likelihood to observe the overall allele frequencies of one locus is then calculated as the product over the populations

$$\prod_{populations j} P(a_{ij}|p_i, \alpha_i, \beta_j),$$

and the overall likelihood of observing the data given the model is the product over all populations j and loci i,

$$L = \prod_{populations j; loci\ i} P(a_{ij}|p_i, \alpha_i, \beta_j).$$

In each pilot run iteration r, the algorithm proposes new values for α_i and β_j, each drawn from a normal distribution $N(x, sd_i)$, where x denotes the value at the previous iteration; this creates a random walk through parameter space. The proposed values are accepted with a probability equal to the ratio in likelihood between the proposed and the previous parameter values (and are always accepted if the likelihood increases); in other words, the new values are accepted with probability $\min(1, \frac{L(r)}{L(r-1)})$.

If the acceptance rate is high (e.g., >0.45), the parameters are close to an optimum, and the algorithm will decrease sd_i. In case of low acceptance rates (e.g., <0.25), sd_i will be increased instead to allow faster movement to the optimum. The pilot runs are used to adjust

the acceptance rates of p_i, α_i, and β_j. This pre-processing step hence facilitates good mixing of the MCMC and thus rapid convergence.

Finding optimal acceptance rates is a challenging task, and the rates are often tuned by hand to achieve the best results for a specific task. Generally, acceptance rates should be neither too high (resulting in a long time to explore the full Likelihood landscape) nor too low (reducing the probability to converge to global maxima). Our method uses the same limiting values [0.25,0.45] as suggested by Foll and Gaggiotti (2008). [NB: In case of the normal distribution, it was found that a value of 0.234 was optimal under certain conditions (Roberts, et al. 1997; Rosenthal 2009).]

Reversible jump model

BayeScan aims to determine the posterior probability of non-neutral population differentiation for each gene through a reversible jump MCMC approach. Reversible jump algorithms are widely used for model determination problems (Hastie and Green 2012). In the application to *F*-statistics, jumps occur between neutral states ($\alpha_i = 0$) and non-neutral states ($\alpha_i \neq 0$).

The α_i distributions estimated in the pilot runs should ensure good mixing of the Markov-chain in the jump-model phase (Hastie and Green 2012). After the pilot runs, the α_i are first set to zero, corresponding to an initial model of only neutral evolution. The means (m_i) and standard deviations (sd_i) of the α_i distributions sampled in the pilot runs are then used in the proposal scheme for each α_i, which is drawn from a normal distribution ($q(\alpha_i) := N(m_i, sd_i)$); this corresponds to a saturated space model (Brooks, et al. 2003). A proposed value is accepted with probability $min(1, A)$, where the likelihood ratio A is (Foll and Gaggiotti 2008):

(4) $$A = \frac{\pi(\tilde{p}|p,\alpha,\beta)\pi(\alpha_i)}{\pi(\tilde{p}|p,\alpha \; with \; \alpha_i=0,\beta)q(\alpha_i)} \quad .$$

Here, $\pi(\tilde{p}|p,\alpha,\beta)$ denotes the likelihood to observe the given data under the current ancestral frequencies p and the α and β values. $\pi(\alpha_i)$ is the prior probability distribution over α_i (a normal distribution $N(0,1)$).

BayeScan then iteratively proposes a new $\alpha_i \neq 0$ if currently $\alpha_i = 0$ and vice versa. If the proposed value is accepted, this α_i is included in the model, i.e., a corresponding jump between $\alpha_i = 0$ and $\alpha_i \neq 0$ is made. The posterior probability that a locus is subject to

selection $P(\alpha_i \neq 0)$ is simply the fraction of iterations in which $\alpha_i \neq 0$ is included in the model.

New Approaches

Considering SNPs in co-evolving groups (type1): α_i as a block effect

To identify non-neutrally evolving blocks of DNA (such as genes) from SNP data, BayeScan allows two possible strategies: either to consider each SNP as an independently evolving unit, or to consider haplotypes that combine all SNPs for one block of DNA (the same applies to the very similar algorithm published by Riebler, et al. (2008)). Considering each SNP separately may lead to statistical problems due to non-independence, and ignores prior knowledge (the strong linkage of SNPs within one co-evolving block). Conversely, using haplotypes can produce misleading results at moderate to high mutation rates, as each mutation introduces a new haplotype (Hudson 2000; Riebler, et al. 2008); this disadvantage of considering haplotypes was strongly confirmed by our simulations (see below).

To avoid the problems of using either individual SNPs or full haplotypes, we propose to combine the strengths of both approaches. We consider the likelihood contribution of each SNP separately, but assume that the locus-specific effect α_i is identical for all SNPs located in the same block of DNA (*e.g.*, the same gene). Thus, in Eqs. (1)-(4), we use one α_i for all SNPs located in the same DNA block. This means that also in the jump model, all SNPs in one block are either classified as evolving neutrally ($\alpha_i = 0$) or non-neutrally ($\alpha_i \neq 0$). This strategy is *type1* of our approach.

We envisage that typical applications of our algorithm might consider genes as coherent units of evolution, and we hence use the terms “block of DNA” and “gene” interchangeably in the remainder of this paper. However, the proposed algorithm can of course be equally applied to other block definitions, such as sliding windows or pre-defined haploblocks.

Improving discrimination performance by controlling average P-values

The likelihood that all SNPs in a block can be explained well by $\alpha_i = 0$ often becomes vanishingly small, especially for large group sizes. Thus, in many cases, a single uniform α_i for all SNPs in one block may be an overly restrictive assumption. Accordingly, we noticed in simulations that the *type1* method successfully distinguishes balancing from directional selection, but produces an overall shift towards 1 of posterior P-values for non-neutral evolution even for many sequences simulated under a neutral scenario. This is not

problematic as long as one considers the posterior distribution of α_i values as the main output of the method. However, the discrimination between neutrally and non-neutrally evolving loci based on the posterior probability *P* becomes inefficient when *P* values are close to 1.

To obtain a more realistic distribution of posterior P-values, we propose a simple and theoretically well-founded modification to the sampling scheme. First, the user specifies an expected mean P-value based on prior knowledge. A user-specified average *P* value of 0.1, for example, means that 10% of all blocks are expected to evolve non-neutrally. During the MCMC sampling, this expectation is treated as an additional constraint, enforced by iteratively adapting the model prior probability $\pi(model\ \alpha \neq 0)$ until the desired mean *P* value is attained (variational inference with posterior constraints, see (Graca, et al. 2007)). In our implementation, the model prior probability is adapted every 100 iterations. As long as the average posterior *P* value measured during the last 100 iterations is above the user-specified value, $log(\pi(model\ \alpha \neq 0))$ is decreased by a fixed step size. The step size can be tuned by the user. Use of this posterior constraint can be deactivated by the user if desired, and the algorithm then uses a fixed prior probability $\pi(\alpha \neq 0)$.

Considering SNPs in co-evolving groups (type2): selection as a block effect

Linkage between the SNPs in one block may not be perfect, and we might expect that deviations from neutral evolution ($\alpha_i = 0$) become weaker with increasing distance from the site of selection. Furthermore, evolution is a stochastic process, and thus individual SNPs may be best described with different α_i even if they evolved under the same selection pressures. To allow for these effects, we propose a second approach that sets no restrictions on the α_i of the individual SNPs in one block. Instead, we use a block-specific jump model (Eq. (4)). Thus, the SNPs in one block are assumed to evolve either neutrally ($\alpha_i = 0$ for all SNPs) or non-neutrally (α_i chosen from the pilot-run distribution for each SNP) together; the decision to include a block into the non-neutrally evolving part of the genome is based on the joint likelihood of all SNPs in the block. To characterize the amount of non-neutral evolution of the block, we use the most extreme α_i value of all SNPs (i.e., the maximal $|\alpha_i|$).

GeneFeST: an R implementation of our algorithm

We implemented our *type1* and *type2* algorithm in a freely available software, GeneFeST, available free of charge from CRAN (http://cran.r-project.org/). GeneFeST runs in the R environment for statistical computing (R Core Team 2014), the *de facto* standard in many

areas of statistics and biological data analysis. Our GeneFeST implementation uses the powerful data processing capabilities of PopGenome, a recently published comprehensive R software package for population genetic and population genomic analyses (Pfeifer, et al. 2014); PopGenome also includes an implementation of the original BayeScan approach (Foll and Gaggiotti 2008). GeneFeST utilizes PopGenome's ability to work with a wide range of input file formats, and comfortable methods exist to define blocks in genomic SNP data based on sliding windows or annotated features such as genes, coding sequences, or exons. Alternatively, the user can generate input text files as illustrated in the documentation, which is also available on CRAN. To ensure fast computations, the GeneFeST source code is highly vectorized, and we have implemented some remaining bottlenecks in C/C++.

Results & Discussion

Simulation Model

To validate our approach and to compare it to alternative F_{ST} measurements, we performed extensive simulations using the program MSMS (Ewing and Hermisson 2010), an extension of Hudson's ms (Hudson 2002) that allows coalescent simulations for a structured population under selection. We chose model and parameters similar to a previously published simplified model of human population history (Gutenkunst, et al. 2009). We assume a population with an effective population size of N_e=10,000 that split into two subpopulation 4,000 generations ago (Figure 1). After the splitting event there is no migration between the two populations (island model). The exact MSMS calls are listed in the Supplementary material.

Based on this population history, we used MSMS to sample 500 genes under a neutral scenario, 50 genes under balancing selection, and 50 genes under population-specific directional selection. We used the same selection coefficient *s* for both balancing and directional selection. We repeated this simulation for different per-gene mutation rates (θ=4$N_e u$), selection coefficients (s), numbers of sampled individuals (N_{ind}), and numbers of populations (N_{pop}).

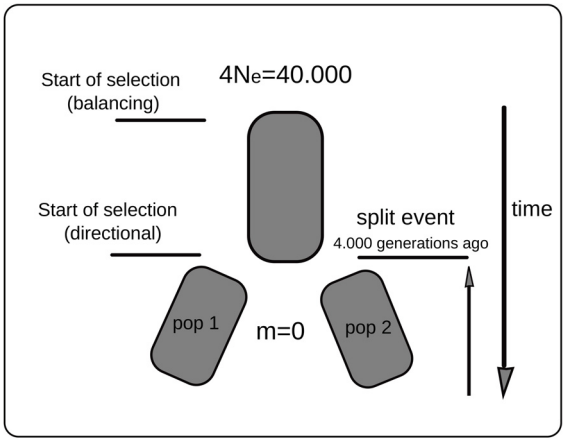


Figure 1. The evolutionary model used to generate samples under neutral, balancing, and directional (population-specific) selection. We start with one population of effective population size N_e=10,000, *i.e.,* an expected gene history of 40,000 generations. 4,000 generations ago, the population split into subpopulations. In the case of balancing selection, selection starts at the very beginning of the simulation; population-specific directional selection acts only after the split event. We assume that no migrations occur between the populations (m=0).

GeneFeST can store the MCMC simulation results to allow further diagnostics. We confirmed adequate convergence speed of the MCMC sampling using the Gelman Rubin convergence diagnostic (Gelman and Rubin 1992), provided in the R package coda (Plummer, et al. 2006). When processing simulated sampling data of 20 individuals each from 2 populations that evolved at a mutation rate of θ=1, this lead to a multivariate potential scale reduction factor (*mpsrf*) of 2.28 for 10,000 pilot run iterations. The *mpsrf* was reduced to 1.63 when we simulated with 20,000 runs. The 'point scale reduction factor' (*psrf*) of the α_i was on average 1.03, with a standard deviation of 0.01, for 10,000 pilot run iterations. As the α_i are interpreted independently from each other to make jump model decisions, we mainly concentrated on the *psrf* outcomes and set the default values of our method accordingly (10,000 pilot run iterations, with updates of sd_i every 500 iterations).

Benchmarking

To benchmark the ability of different F_{ST} measurements to detect non-neutral evolution at the gene level, we scored Receiver-Operator-Characteristic (ROC) curves, which plot sensitivity (fraction of true positives) versus specificity (1 - fraction of false positives) at different cutoff values for the parameter used for discrimination. The area under this curve (*AUC*) can be

interpreted as the overall accuracy of discrimination provided by the parameter (Riebler, et al. 2008). *AUC* was calculated with the R package pROC (Robin, et al. 2011).

We assessed the performance of BayeScan (Foll and Gaggiotti 2008) applied to either full gene haplotypes or individual SNPs, and Hudson's F_{ST} (Hudson 2000) calculated for haplotypes or for nucleotides. The accuracy of these methods was compared to the performance of the GeneFeST *type1* and *type2* models introduced here.

To obtain a rough estimate of the statistical variation expected for *AUC* values based purely on stochastic fluctuations in the coalescent process, we simulated for every parameter combination (see below) 10,000 loci under a neutral, balancing selection, and directional selection scenario, and compared the *AUC* values for Hudson's F_{ST} with those based on the smaller sample sizes. We found that *AUC* values typically differed by at most 0.03. Below, we thus consider *AUC* differences >0.06 as noteworthy.

F_{ST} measurements across mutation rates

We performed simulations for three different values of the scaled mutation rate θ (1, 5, and 10), corresponding to mutation rates *u* of 0.000025, 0.000125, and 0.00025 per gene and generation. Table 1 shows that with the simulated population history we expect on average 5.44θ SNPs per gene.

Table 2 and Figure 2 show the accuracy (*AUC* of the ROC curve) of the examined methods when aiming to detect non-neutral evolution from patterns of population differentiation under different mutation rates. Both Hudson's F_{ST} and BayeScan made more accurate predictions when applied to individual SNPs than when applied to haplotypes, indicating that summarizing SNP data into haplotypes is often associated with a substantial loss in information.

Table 1: *Relationship between mutation rate θ and the number of SNPs per gene*

Mutation rate $\theta=4N_e u$	Mean	Standard deviation
1	5.44	2.50
5	27.22	8.72
10	53.72	15.17

The jump model introduced by Foll & Gaggiotti (2008) and independently by Riebler, et al. (2008) aims to give a direct estimate of the probability that a locus evolves non-neutrally, expressed as a posterior probability *P*. However, Table 2 shows that predictions of non-neutrality based on the posterior α_i values (which describe the locus-specific component of the population differentiation) are much more accurate than those based on *P*.

At low mutation rate (θ=1, corresponding to on average 5.4 SNPs per gene), the methods proposed here (GeneFeST *type1* and *type2*) are similar in accuracy to BayeScan. At higher mutation rates, this is also the case for directional selection, while GeneFeST is more accurate for balancing selection. Hudson's nucleotide F_{ST} performed surprisingly well at intermediate and high mutation rates, where it showed an accuracy similar to BayeScan.

Table 2: *Accuracies (AUC) for F_{ST} measurements across mutation rates*

Number of pops	Sample size	Mutation rate $\theta=4N_eu$	*AUC*	BayeScan haplotype	BayeScan (SNPs)[1]	Hudson's nucleotide F_{ST}	Hudson's haplotype F_{ST}	GeneFeST (*type1*)	GeneFeST (*type2*)
2	20	1		pilot	pilot			pilot	pilot
			bal [2]	0.6755	0.732	0.6962	0.6171	0.7298	0.7298
			pos [3]	0.7281	0.7378	0.658	0.625	0.7179	0.7292
			P.norm [4]	0.5696	0.6099	0.6222	0.5919	0.611	0.614
				posterior	posterior			posterior	posterior
			bal [2]	0.6808	0.742			0.731	0.7535
			pos [3]	0.7424	0.7511			0.7257	0.7485
			P [5]	0.5834	0.5647			0.6243	0.6558
2	20	5		pilot	pilot			pilot	pilot
			bal [2]	0.5614	0.732	0.7738	0.6068	0.8215	0.7227
			pos [3]	0.7544	0.8256	0.7909	0.7647	0.8548	0.8333
			P.norm [4]	0.594	0.6888	0.7192	0.6405	0.7482	0.6863
				posterior	posterior			posterior	posterior
			bal [2]	0.5589	0.7368			0.8248	0.7419
			pos [3]	0.7571	0.8464			0.8546	0.8309
			P [5]	0.6083	0.734			0.7529	0.7414
2	20	10		pilot	pilot			pilot	pilot
			bal [2]	0.6067	0.7722	0.809	0.5911	0.8626	0.7696
			pos [3]	0.6508	0.8059	0.8006	0.6509	0.8233	0.8028
			P.norm [4]	0.5715	0.6805	0.6993	0.5614	0.6948	0.677
				posterior	posterior			posterior	posterior
			bal [2]	0.6045	0.7883			0.8633	0.7803
			pos [3]	0.6513	0.7923			0.8212	0.7972
			P [5]	0.571	0.6818			0.6867	0.7402

[1] *BayeScan performed on individual SNPs and interpreting the most extreme α_i in a group (identical to our type2 method except for the jump model).*

[2] *Detection of balancing selection*

[3] *Detection of population-specific directional selection*

[4] *Probability of non-neutral evolution (balancing and directional selection combined), estimated from fits of the parameter distribution (α_i^0 or F_{ST}) to a normal distribution.*

[5] *Probability of non-neutral evolution (balancing and directional selection combined), estimated via the jump model.*

We noticed in our simulations that the locus-specific effects α_i^0 estimated during the pilot runs were almost as useful for the detection of non-neutrality as the posterior α_i, and we hence also report *AUC* values based on these α_i^0. The detection of natural selection based on α_i^0 was originally proposed in (Beaumont and Balding 2004). To allow a direct statistical interpretation in this case, we also fitted the α_i^0 distributions from the pilot runs to normal distributions, and estimated a probability P_0 for non-neutrality from the position of a locus's α_i^0 relative to this distribution. In most simulations, these P_0 values were similarly accurate for the classification of non-neutrality as the computationally much more expensive posterior probabilities P. To allow a statistical interpretation of Hudson's F_{ST} values, we also fitted these to a normal distribution and calculated corresponding P_H values.

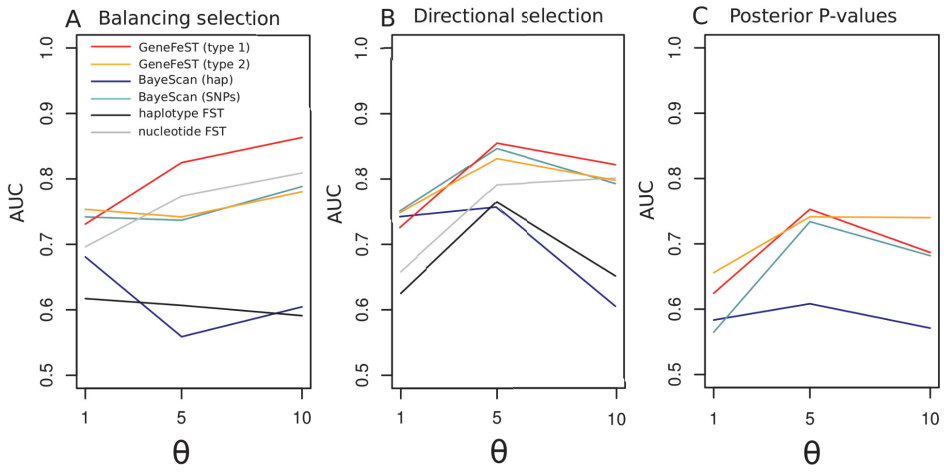


Figure 2. The accuracy of F_{ST} measurements across mutation rates. *AUC* values are plotted against the scaled mutation rate $\theta=4N_e u$. Panels (A), balancing selection, and (B), directional selection, show *AUC* calculated for posterior α_i from BayeScan and GeneFeST, as well as for Hudson's F_{ST}. (C) *AUC* calculated for posterior probabilities of non-neutral evolution *P* calculated with BayeScan and GeneFeST. The *AUC* values are also listed in Table 2.

F_{ST} measurements across sample sizes and numbers of populations

We also performed simulations across different sample sizes and numbers of populations. Consistent with previous observations (see Table 4 in (Foll and Gaggiotti 2008)), we found that all tested methods gain substantially in power when samples from more populations are available (Supplementary Table S1). In this situation of generous data availability, the choice of method becomes less important. Conversely, increasing sample sizes above 20 within the

same two populations does not systematically improve the accuracy of predictions (Supplementary Table S2).

Both GeneFeST *type2* and BayeScan performed on SNPs (as employed here) classify genes based on the most extreme α_i value. As seen in Supplementary Table S2, this strategy becomes more powerful with increasing sample sizes, presumably because increasingly more information is available for each individual SNPs.

F_{ST} measurements on genes of different lengths and across selection coefficients

When applying F_{ST} based methods to genome-scale data, it is important to recognize that gene lengths - and hence per-gene mutation rates θ - can vary widely between genes. We thus performed benchmark tests on data simulated on a continuous distribution of θ values. We estimated θ for all genes on human chromosome 21 (Genomes Project, et al. 2012) with Watterson's estimator (Watterson 1975) as implemented in the R package PopGenome (Pfeifer, et al. 2014). We then randomly sampled θ values out of this distribution to generate 500 neutrally evolving genes, 50 genes under balancing selection, and 50 genes under directional selection. These mixed θ values modeled on the human genome lead to on average 138 SNPs per gene, with a standard deviation of 136. We performed these simulations with three different selection coefficients s (0.01, 0.05, and 0.1), sampling 20 individuals each from two populations evolving as above.

Figure 3 shows ROC curves for the classifications based on Bayesian posterior parameters at strong selection (s=0.1). The number of haplotypes per gene increases strongly with increasing θ, and hence haplotype-based predictions become heavily biased by gene length in realistic situations (Figure 4). The ROC curves for BayeScan applied to haplotypes in Figure 3 are close to the diagonals; *i.e.*, haplotype-based classifications are hardly more accurate than random guess in realistic situations. Accordingly, the *AUC* values for haplotype-based methods are barely above the random expectation of 0.5 (Table 3 and Figure 5). In contrast, GeneFeST *type1* successfully separates the three simulated scenarios (neutral evolution, balancing selection, and directional selection); the separation is evident already from the locus-specific effects estimated in the pilot runs, α_i^0.

Comparing the different F_{ST}-based methods across selection coefficients (Table 3 and Figure 5), we again find that the accuracy of GeneFeST *type1* and BayeScan applied to SNPs is comparable for classifying genes under directional selection, while GeneFeST *type1* is substantially more accurate for the detection of balancing selection. In contrast, the

model of GeneFeST *type2* seems not to be appropriate for the realistic situation that sequences differ widely in length.

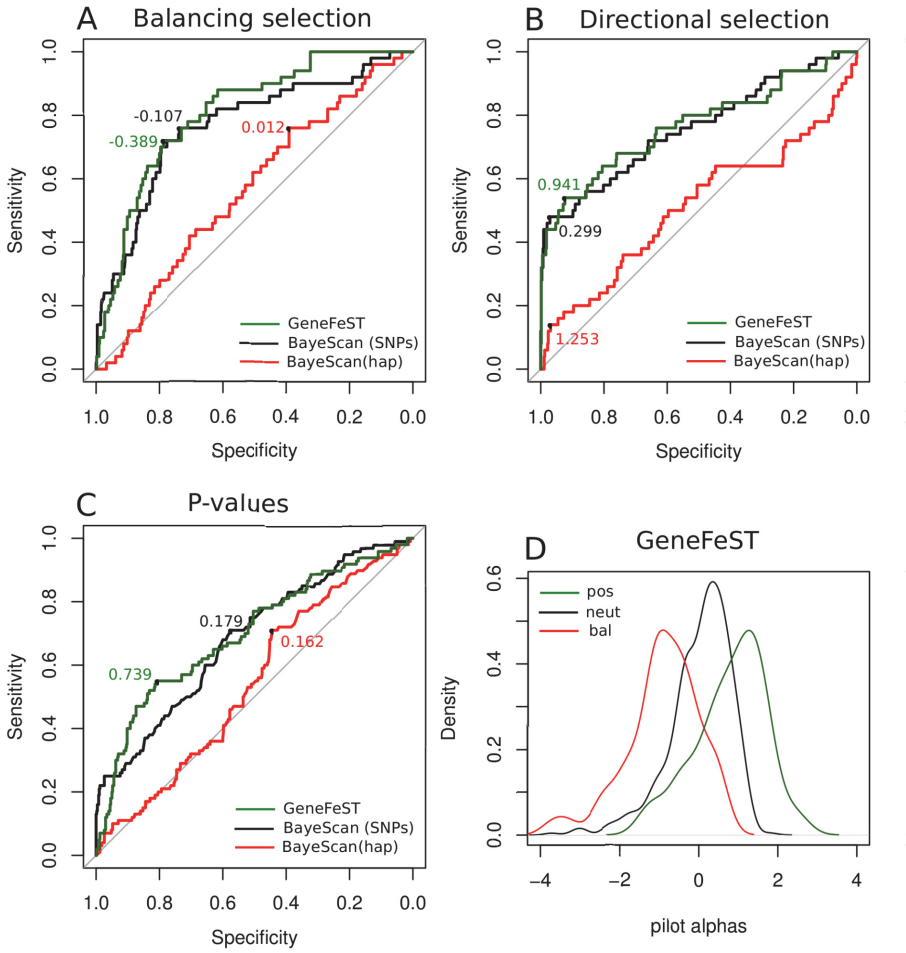


Figure 3. ROC curves and distributions of α_i^0 for simulated data corresponding to genes of varying lengths. Panels (A) and (B) show ROC curves for α_i computed by BayeScan and α_i^0 computed by GeneFeST *type1*. The numbers next to the ROC curves report the best threshold values (those with the highest distance from the diagonal). (C) The α_i^0 from the GeneFeST pilot runs were fitted to a normal distribution to estimate P_0 values for non-neutral evolution, which were then used to generate the ROC curve; this is compared to the posterior probabilities given by BayeScan. (D) Distribution of the pilot run α_i^0 from GeneFeST for genes under balancing selection (bal), directional selection (pos), and neutral evolution (neut). The plots were generated based on simulations under strong selection, *s*=0.1.

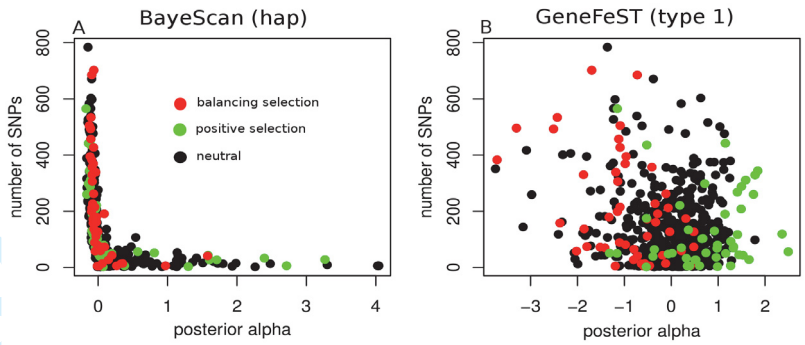


Figure 4: Correlation between α_i and the number of SNPs per gene (reflecting per-gene mutation rate or gene length). The total number of SNPs observed in a simulated gene is plotted against the α_i estimated by BayeScan applied to haplotype data (A) and by GeneFeST *type1* (B). Haplotype calculations are strongly biased by gene length, while this is not the case for the GeneFeST approach. Calculations were performed on 20 simulated individuals each from 2 populations at strong selection (*s*=0.1).

Table 3: *Accuracies (AUC) for F_{ST} measurements across selection strengths*

Number of pops	Sample size	Selection coefficients	*AUC*	BayeScan haplotype	BayeScan (SNPs)[1]	Hudson's nucleotide F_{ST}	Hudson's haplotype F_{ST}	GeneFeST (*type1*)	GeneFeST (*type2*)
2	20	0.1		pilot	pilot			pilot	pilot
			bal [2]	0.5686	0.6714	0.7695	0.5685	0.8102	0.7384
			pos [3]	0.5147	0.7459	0.7435	0.5287	0.7664	0.7549
			P.norm [4]	0.5348	0.6687	0.688	0.5166	0.6978	0.7017
				posterior	posterior			posterior	posterior
			bal [2]	0.5652	0.7689			0.8124	0.6987
			pos [3]	0.5129	0.7592			0.7691	0.7561
			P [5]	0.5419	0.6834			0.6859	0.6621
2	20	0.05		pilot	pilot			pilot	pilot
			bal [2]	0.5543	0.6668	0.7115	0.5322	0.8059	0.7011
			pos [3]	0.5465	0.7598	0.7345	0.537	0.7647	0.7661
			P.norm [4]	0.5432	0.6811	0.6519	0.5429	0.6647	0.6968
				posterior	posterior			posterior	posterior
			bal [2]	0.5524	0.6751			0.807	0.6884
			pos [3]	0.5438	0.7951			0.7661	0.7712
			P [5]	0.5361	0.6611			0.6501	0.62
2	20	0.01		pilot	pilot			pilot	pilot
			bal [2]	0.5631	0.66	0.7249	0.5608	0.8084	0.6946
			pos [3]	0.5391	0.7486	0.6125	0.5523	0.7695	0.7433
			P.norm [4]	0.5198	0.6661	0.5871	0.5522	0.6993	0.6929
				posterior	posterior			posterior	posterior
			bal [2]	0.5586	0.7648			0.8132	0.6596
			pos [3]	0.5563	0.7687			0.7686	0.7427
			P [5]	0.5313	0.6562			0.6696	0.6294

[1] *BayeScan performed on individual SNPs and interpreting the most extreme α_i in a group (identical to our type2 method except for the jump model).*

[2] *Detection of balancing selection*

[3] *Detection of population-specific directional selection*

[4] *Probability of non-neutral evolution (balancing and directional selection combined), estimated from fits of the parameter distribution (α_i^0 or F_{ST}) to a normal distribution.*

[5] *Probability of non-neutral evolution (balancing and directional selection combined), estimated via the jump model.*

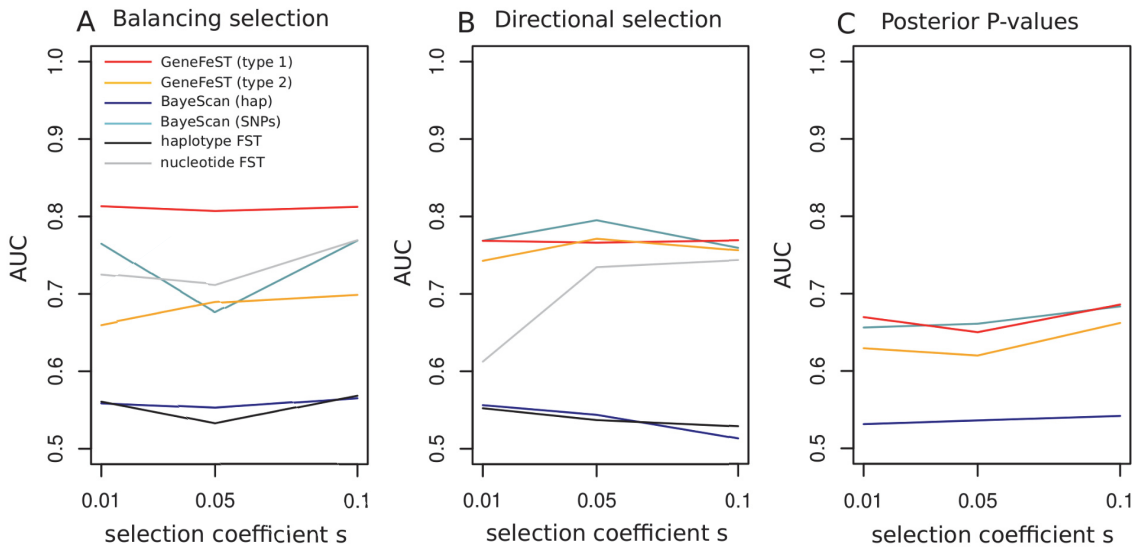


Figure 5. The accuracy of F_{ST} measurements across selection coefficients. *AUC* values are plotted against selection coefficients *s*. Panels (A), balancing selection, and (B), directional selection, show *AUC* calculated for posterior α_i from BayeScan and GeneFeST, as well as for Hudson's F_{ST}. (C) *AUC* calculated for posterior probabilities of non-neutral evolution *P* calculated with BayeScan and GeneFeST. The *AUC* values are also listed in Table 3.

Conclusions

F_{ST} statistics to assess the extent of population differentiation can detect both balancing selection and population-specific directional selection from SNP data. Neighboring SNPs, such as those co-localized in the same gene or exon, are often expected to be under similar selection pressures. However, summarizing SNP data into haplotypes results in a substantial loss of information, and haplotype-based F_{ST} methods were inferior in the detection of non-neutral evolution in all simulated situations.

Surprisingly, we found that the addition of a jump model to the original Bayesian approach of (Beaumont and Balding 2004), while requiring much longer computation times, leads to only a small improvement in accuracy. Moreover, in both BayeScan and GeneFeST, detection of non-neutral evolution is much less accurate when based on the jump model posterior probabilities *P* than when based on the locus-specific effects α_i. The apparently straightforward interpretation of the posterior *P* values is an illusion; fitting the α_i (or even the pilot run α_i^0) distribution to a Gaussian provides a more reliable reference frame.

While prediction of non-neutral evolution with the GeneFeST *type1* method was at least as accurate as the other methods tested and was superior in many simulations, we observed one shortcoming of this method. A single α_i for a whole block of SNPs may sometimes be an overly restrictive assumption, resulting in a global shift of posterior *P* values towards 1. We proposed a simple approach to mitigate this issue, by balancing posterior *P* values using the framework of variational inference with posterior constraints (Graca, et al. 2007). Future work should address how to improve the Bayesian model itself to ensure that posterior probabilities are more directly interpretable and representative of a locus's true selection history. One promising avenue of research may be the introduction of an additional layer in the hierarchical Bayesian model that mediates between the unspecific global prior distribution, $\pi(\alpha_i)$, and the SNP-level α_i. Linkage of SNPs within one gene could be accounted for in such a model by introducing group priors that bind together all α_i within each gene. Such a model would also allow to distinguish between gene-wide and more local locus-specific effects.

Several recommendations for the population differentiation analysis of SNP data can be drawn from our simulation study:

- When sequencing additional individuals, strong preference should be given to sampling additional populations;
- Haplotype-based methods are substantially less reliable than methods that examine individual SNPs, and are strongly biased when blocks with different lengths or mutation rates are compared;
- The strategy of assigning the same locus-specific coefficient to all SNPs within one co-evolving block (GeneFeST *type1*) leads to substantially improved predictions especially in the case of balancing selection compared to previous Bayesian approaches;
- The predictions derived from pilot runs (as originally proposed by Beaumont & Balding (2004)) are almost as good as those derived from the jump model results, but can be computed much faster.
- To detect non-neutrally evolving sequence blocks (genes) with Bayesian methods, one should not use the posterior probabilities provided by a jump model, but should use the locus effects α_i instead.
- Hudson's nucleotide F_{ST} provides a fast and relatively good measurement in the case of strong directional selection, but is inferior to the Bayesian approach implemented in GeneFeST (*type1*) especially for the detection of balancing selection.

Acknowledgments

We thank Pablo E. Verde, Gregory Ewing, Ernst Stadlober, Laura Rose, and Juliette de Meaux for helpful discussions. MJL acknowledges financial support from the German Research Foundation (DFG grants EXC 1028 and CRC 680). SH acknowledges partial financial support by the Human Brain Project.

References

Balding DJ, Nichols RA 1995. A method for quantifying differentiation between populations at multi-allelic loci and its implications for investigating identity and paternity. Genetica 96: 3-12.

Beaumont MA, Balding DJ 2004. Identifying adaptive genetic divergence among populations from genome scans. Mol Ecol 13: 969-980.

Brooks SP, Giudici P, Roberts GO 2003. Efficient construction of reversible jump Markov chain Monte Carlo proposal distributions. Journal of the Royal Statistical Society Series B-Statistical Methodology 65: 3-39.

Ewing G, Hermisson J 2010. MSMS: a coalescent simulation program including recombination, demographic structure and selection at a single locus. Bioinformatics 26: 2064-2065. doi: 10.1093/bioinformatics/btq322

Foll M, Gaggiotti O 2008. A genome-scan method to identify selected loci appropriate for both dominant and codominant markers: a Bayesian perspective. Genetics 180: 977-993. doi: 10.1534/genetics.108.092221

Gelman A, Rubin D 1992. Inference from iterative simulation using multiple sequences. Statistical Science 7: 457-511.

Genomes Project C, Abecasis GR, Auton A, Brooks LD, DePristo MA, Durbin RM, Handsaker RE, Kang HM, Marth GT, McVean GA 2012. An integrated map of genetic variation from 1,092 human genomes. Nature 491: 56-65. doi: 10.1038/nature11632

Graca JV, Ganchev K, Taskar B. 2007. Expectation Maximization and Posterior Contraints. Advances in Neural Information Processing Systems 20 (NIPS); Vancouver, Canada.

Green PJ 1995. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. Biometrika 82: 711-732. doi: DOI 10.1093/biomet/82.4.711

Gutenkunst RN, Hernandez RD, Williamson SH, Bustamante CD 2009. Inferring the joint demographic history of multiple populations from multidimensional SNP frequency data. PLoS Genet 5: e1000695. doi: 10.1371/journal.pgen.1000695

Hartl DL, Clark AG. 2007. Principles of population genetics. Sunderland, Mass.: Sinauer Associates.

Hastie DI, Green PJ 2012. Model choice using reversible jump Markov chain Monte Carlo. Statistica Neerlandica 66: 309-338.

Hudson RR 2002. Generating samples under a Wright-Fisher neutral model of genetic variation. Bioinformatics 18: 337-338.

Hudson RR 2000. A new statistic for detecting genetic differentiation. Genetics 155: 2011-2014.

Hudson RR, Slatkin M, Maddison WP 1992. Estimation of levels of gene flow from DNA sequence data. Genetics 132: 583-589.

Kronholm I, Loudet O, de Meaux J 2010. Influence of mutation rate on estimators of genetic differentiation - lessons from Arabidopsis thaliana (vol 11, pg 33, 2010). Bmc Genetics 11. doi: Artn 88

Doi 10.1186/1471-2156-11-88

Pfeifer B, Wittelsbürger U, Ramos-Onsins SE, Lercher MJ 2014. PopGenome: A swiss army knife for population genetic & genomic analysis. Molecular Biology and Evolution (in press).

Plummer M, Best NG, Cowles K, Vines K 2006. CODA: Convergence Diagnosis and Output Analysis for MCMC. R. News 6 (1): 7-11.

R: A Language and Environment for Statistical Computing [Internet]. Vienna, Austria: R Foundation for Statistical Computing; 2014.

Riebler A, Held L, Stephan W 2008. Bayesian variable selection for detecting adaptive genomic differences among populations. Genetics 178: 1817-1829. doi: 10.1534/genetics.107.081281

Roberts G, Gelman A, Gilks WR 1997. Weak convergence and optimal scaling of random walk Metropolis algorithms. The Annals of Applied Probability 7: 1-279.

Robin X, Turck N, Hainard A, Tiberti N, Lisacek F, Sanchez JC, Muller M 2011. pROC: an open-source package for R and S+ to analyze and compare ROC curves. BMC Bioinformatics 12: 77. doi: 10.1186/1471-2105-12-77

Rosenthal J. 2009. Optimal proposal distributions and adaptive MCMC. In. Handbook of Markov chain Monte Carlo: Methods and Applications. Chapman Hall/CRC Press, Florida-USA: Gelman, A., Jones, G., Meng, X., Brooks, S. (eds.).

Watterson GA 1975. On the number of segregating sites in genetical models without recombination. Theor Popul Biol 7: 256-276.

Weir BS. 1996. Genetic data analysis II : methods for discrete population genetic data. Sunderland, Mass.: Sinauer Associates.

Weir BS, Cockerham CC 1984. Estimating F-Statistics for the Analysis of Population-Structure. Evolution 38: 1358-1370. doi: Doi 10.2307/2408641

Wright S 1950. Genetical structure of populations. Nature 166: 247-249.

Supplemental Tables

Table S1: *F_{ST} measurements across population numbers*

Number of pops	Sample size	Mutation rate $\theta=4N_eu$	*AUC*	BayeScan haplotype	BayeScan (SNPs)[1]	Hudson's nucleotide F_{ST}	Hudson's haplotype F_{ST}	GeneFeST (*type1*)	GeneFeST (*type2*)
5	20	1		pilot	pilot			pilot	pilot
			bal [2]	0.7906	0.9073	0.8345	0.7548	0.8508	0.9081
			pos [3]	0.8828	0.8732	0.7828	0.7937	0.8568	0.8765
			P.norm [4]	0.6775	0.7884	0.7308	0.6852	0.7466	0.7937
				posterior	posterior			posterior	posterior
			bal [2]	0.7856	0.9064			0.8504	0.9094
			pos [3]	0.8831	0.8613			0.8599	0.8685
			P [5]	0.6802	0.7314			0.7392	0.7294
5	20	5		pilot	pilot			pilot	pilot
			bal [2]	0.6135	0.9159	0.9287	0.6661	0.9329	0.9187
			pos [3]	0.8586	0.9348	0.9374	0.9	0.92	0.9371
			P.norm [4]	0.5853	0.7988	0.8483	0.643	0.8497	0.799
				posterior	posterior			posterior	posterior
			bal [2]	0.6217	0.9184			0.9316	0.9209
			pos [3]	0.8559	0.937			0.9209	0.9328
			P [5]	0.5831	0.79			0.8498	0.7782

[1] *BayeScan performed on individual SNPs and interpreting the most extreme α_i in a group (identical to our type2 method except for the jump model).*

[2] *Detection of balancing selection*

[3] *Detection of population-specific directional selection*

[4] *Probability of non-neutral evolution (balancing and directional selection combined), estimated from fits of the parameter distribution (α_i^0 or F_{ST}) to a normal distribution.*

[5] *Probability of non-neutral evolution (balancing and directional selection combined), estimated via the jump model.*

Table S2: *F_{ST} measurements across sample sizes*

Number of pops	Sample size	Mutation rate $\theta=4N_eu$	*AUC*	BayeScan haplotype	BayeScan (SNPs)[1]	Hudson's nucleotide F_{ST}	Hudson's haplotype F_{ST}	GeneFeST (*type1*)	GeneFeST (*type2*)
2	100	1	bal [2] pos [3] P.norm [4] bal [2] pos [3] P [5]	pilot 0.7151 0.7434 0.636 posterior 0.7185 0.7482 0.6282	pilot 0.8459 0.7118 0.6886 posterior 0.8378 0.7129 0.6393	0.7517 0.5857 0.6722	0.6956 0.6052 0.615	pilot 0.7197 0.7053 0.6209 posterior 0.7201 0.7095 0.6178	pilot 0.8192 0.7175 0.7047 posterior 0.8328 0.7093 0.6523
2	100	5	bal [2] pos [3] P.norm [4] bal [2] pos [3] P [5]	pilot 0.59 0.7652 0.6153 posterior 0.5949 0.7616 0.6299	pilot 0.9318 0.7742 0.806 posterior 0.9146 0.7691 0.6622	0.8563 0.7329 0.7245	0.6395 0.7557 0.6548	pilot 0.8305 0.8067 0.7086 posterior 0.8289 0.807 0.7202	pilot 0.9328 0.7761 0.8035 posterior 0.9266 0.7747 0.673

[1] *BayeScan performed on individual SNPs and interpreting the most extreme α_i in a group (identical to our type2 method except for the jump model).*

[2] *Detection of balancing selection*

[3] *Detection of population-specific directional selection*

[4] *Probability of non-neutral evolution (balancing and directional selection combined), estimated from fits of the parameter distribution (α_i^0 or F_{ST}) to a normal distribution.*

[5] *Probability of non-neutral evolution (balancing and directional selection combined), estimated via the jump model.*

MSMS calls

Model:

Theta (-t) = 5

Number of populations (-I)=2

Sample size (-I) = 20

Effective population size (-N) = 10,000

Split event (-ej) = 0.1 (4,000 generations ago)

Selection coefficient (-SAA & -SaA) = 0.1

Start of selection (-SI) = 0.9 (balancing selection), 0.1 (positive selection)

Initial frequencies of the beneficial allele = 0.01

Neutral:

msms 40 500 -t 5 -N 10000 -I 2 20 20 0 -ej 0.1 1 2

Balancing selection:

msms 40 50 -t 5 -N 10000 -I 2 20 20 0 -ej 0.1 1 2 -SAA 1 -SaA 2000 -SI 0.9 2 0.01 0.01

Directional selection:

msms 40 50 -t 5 -N 10000 -I 2 20 20 0 -ej 0.1 1 2 -SAA 2000 -SaA 1 -SI 0.1 2 0.01 0.01

Bibliography

[1] The International HapMap Consortium. The international hapmap project. *Nature*, 426:789–796, 2003.

[2] Adler et. al. ff: memory-efficient storage of large data on disk and fast access functions. *R-project*, 2013.

[3] Cao et al. Whole-genome sequencing of multiple arabidopsis thaliana populations. *Nature Genetics*, 43:956–963, 2011.

[4] Danecek et al. The variant call format and vcftools. *Bioinformatics*, 27:2156–2158, 2011.

[5] McVean et al. An integrated map of genetic variation from 1,092 human genomes. *Nature*, 491:5665, 2012.

[6] Nielsen et al. Genomic scans for selective sweeps using SNP data. *Genome Research*, 15:1566–1575, 2005.

[7] Rozas et. al. DnaSP, DNA polymorphism analyses by the coalescent and other methods. *Bioinformatics*, 19(18):2496–2497, 2003.

[8] Tajima F. Statistical method for testing the neutral mutation hypothesis by dna polymorphism. *Genetics*, 123(3):585–95, 1989.

[9] Ewing G. and Hermisson J. msms: a coalescent simulation program including recombination, demographic structure and selection at a single locus. *Bioinformatics*, 26(16):2064–2065, 2010.

[10] Li H. Tabix: fast retrieval of sequence features from generic tab-delimited files. *Bioinformatics*, 27(5):718–719, 2011.

[11] Daniel L. Hartl and Andrew G. Clark. *Principles of population genetics*. Sinauer Associates, Inc. Publishers, Sunderland, Massachusetts, 2007.

[12] Kingman J.F.J. On the genealogy of large populations. *Journal of Applied Probability*, 19A:27–43, 1982.

[13] Excoffier L. and Heckel G. Computer programs for population genetics data analysis: a survival guide. *Nature Reviews Genetics*, 7:745–758, 2006.

[14] Beaumont M. and Balding D. Identifying adaptive genetic divergence among populations from genome scans. *Molecular Ecology*, 13:969–980, 2004.

[15] Foll M. and Gaggiotti O. A genome scan method to identify selected loci appropriate for both dominant and codominant markers: A bayesian perspective. *Genetics*, 180:977–998, 2008.

[16] Green PJ. Reversible jump markov chain monte carlo computation and bayesian model determination. *Biometrika*, 82:711–732, 1995.

[17] Hudson R. Generating samples under a wright-fisher neutral model of genetic variation. *Bioinformatics*, 18:337–338, 2002.

[18] Hudson R., Slatkin M., and Maddison W. Estimating of levels of gene flow from dna sequence data. *Genetics*, 13(2):583–589, 1992.

[19] Wright S. Genetical structure of populations. *Nature*, 166(4215):247–9, 1950.

[20] Cormen T., Leiserson C., Rivest R., and Stein C. *Section 24.3: Dijkstra's algorithm. Introduction to Algorithms (Second ed.).* MIT Press and McGrawHill. pp. 595–601, 2001.