# DEVELOPING TOOLS

# FOR MAKING INFERENCES FROM GENOMIC DATA

---

A Thesis

presented to the Faculty of the Graduate School

at the University of Missouri-Columbia

---

In Partial Fulfillment of the Requirements

for the Degree Master of Arts

---

by

PAUL PETROWSKI

Dr. Elizabeth G. King, Thesis Supervisor

MAY 2019

The undersigned, appointed by the Associate Vice Chancellor of the Office of Research and Graduate Studies, have examined the thesis entitled:

DEVELOPING TOOLS FOR MAKING INFERENCES FROM GENOMIC DATA

presented by Paul Petrowski, a candidate for the degree of Master of Biology, and hereby certify that, in their opinion, it is worthy of acceptance.

_____

Elizabeth G. King, PhD

_____

Lori S. Eggert, PhD

_____

J. Chris Pires, PhD

_____

Jianlin J. Cheng, PhD

**For Carolyn and Luke**

**ACKNOWLEDGEMENTS**

I would have failed miserably at this long ago if not for the supernatural poise and patience of Libby King, my mentor and role model. She has gently inspired me to think and act purposefully. I am in constant awe of her skill and character.

Libby is the latest in a long line of incredible women who have guided me down the road of science. Brenda Parness was the first to spark a flame in me for biology, Roslyn Crowder destroyed and then rebuilt my perception of what it takes to be a scientist, and Alicia Slater gave me my first shot at doing research. It's safe to say that without her influence this document would never have come into existence.

There are those occasions in life when the way forward is uncertain, and we look to others to provide a belief in ourself that we cannot conjure up from within. John Davenport, Dan Gaffney, Joe Ferreira, and Don Askew have each in turn been that person.

I also have to give a big thanks to Joe, Taylor, Kyle, Sharjeel, Skylar, Sammi Jo, Zack, Austin, Troy, Harly, Sarah, Alex, and Abi for being a part of my life over the last six years. Without them, life would be much more empty.

Lastly, I thank my Mother, who for 23 years has been my unceasing champion, and my Father, who for 23 years has been my unshakable rock.

## TABLE OF CONTENTS

# DEVELOPING TOOLS FOR MAKING INFERENCES FROM GENOMIC DATA

Paul Petrowski

Dr. Elizabeth G. King, Thesis Supervisor

## ABSTRACT

The central question that the King lab seeks to answer is, “What happens on the genomic level when phenotypes evolve?” To this end, the core focus of my work has been on developing software and analytical methods that facilitate the ability of other researchers working in this space to answer their questions of interest.

I have long believed that writing code is like carpentry. The fundamental tools are simple, but extremely beautiful and practical products may emerge through sufficiently complex sequencing of these tools. What I hope I’ve done over these last two years is build products that are both useful and intuitive to use. In the coming pages, I’ll describe the two major projects that I’ve worked on during my time as a graduate student at the University of MIssouri, how they are useful, and what comes next.

In the first of these two chapters, we describe `ohtadstats`, an R package designed to facilitate the computation of Tomoka Ohta’s D statistics on large data datasets. Ohta’s D statistics measure linkage disequilibrium in

subdivided populations, and as she posited in her 1982 paper, they may provide insights into both the existence and mechanism of selection acting on the paired loci. Our implementation makes it straightforward to compute these statistics for both a single pair of loci, or massive genomic datasets that have become common as the difficulty and cost associated with whole genome sequencing has plummeted over the past decade.

In the second chapter, we describe the problem of inferring ancestral haplotype frequencies from pooled sequence data. We discuss five approaches to this problem that have already been developed, and we discuss a new genetic algorithm based approach that we designed. We perform a benchmarking test on a method developed by Burke *et al.* and we compare that method to the genetic algorithm by evaluating the performance of each on a simulated dataset. We find that the the new method outperforms the existing one.

# CHAPTER ONE: OHTA'S D STATISTICS

## Introduction

Pronounced signatures are left in the genomes of species undergoing selection. These telltale signals may reveal selected loci and details regarding the selection pressures that have been applied [1]. Notably, selection modifies the correlation of alleles at sites near the selected locus. This correlation between alleles is known as linkage disequilibrium (LD). As a rule of thumb, selection reduces genetic variability and increases LD [2]. Tomoka Ohta [3] derived a series of statistics (from here on referred to as Ohta's D statistics) designed to partition LD into between and within population components in a manner analogous to Sewall Wright's F statistics, a measure of inbreeding [4]. She posited that for a pair of loci, deviations from expected levels of LD in a given subpopulation may be indicative of an epistatic selection event. More recently, Beissinger et al (2016) demonstrated that Ohta's D Statistics, particularly the $D'^2_{IS}$ statistic, can identify traditional hard selective sweeps [2] in addition to epistatic selection, and they developed a null-distribution that enables the genome-wide identification of selection candidates.

Several studies have been conducted that utilize Ohta's D statistics to test for epistatic selection [5–7], which is selection acting on a favorable combination of loci, rather than a single locus independently. However, software suites for measuring Ohta's D statistics are limited. Programs that evaluate the statistics on

a locus-by-locus basis exist [8–10], but there is currently no framework available to facilitate the implementation of Ohta's D statistics on a genome-wide basis. Furthermore, the web-based platform supplied by [9] and its predecessor [8] have two significantly limiting attributes that are resolved by our implementation. First, they require that input data be limited to no more than 80 SNPs in 30 or fewer subpopulations. Second, they treat sample size as population size, an approach that will only occasionally reflect reality.

Ohta's D statistics are computed in a pairwise fashion between markers, so evaluating even a relatively small marker set of a few hundred or thousands of SNPs requires an efficient implementation. Therefore, we have developed ohtadstats, a freely available R package with convenient, flexible, and powerful tools to perform the computation of Ohta's D statistics in a variety of use cases. By leveraging the R statistical software platform [11], ohtadstats is fast, scalable, and most importantly adaptable to an endless array of system architectures and high-throughput computing systems. Here, we describe the capabilities of the ohtadstats package and demonstrate its applicability to datasets both small scale and genome wide.

## A Review of Ohta's D Statistics

Ohta's D statistics are a set of five statistics, termed $D^2_{it}$, $D^2_{is}$, $D^2_{st}$, $D'^2_{is}$, and $D'^2_{st}$. The specific forms of these statistics have been covered in depth by Ohta [3] so we will not go in depth here. Briefly, from Beissinger et al. [6]:

- $D^2_{it}$ is the correlation of two alleles occurring on the same gamete in a subpopulation compared to the expectation of them occurring together in the total population
- $D^2_{is}$ is the expected variance of LD for subpopulations
- $D^2_{st}$ is the correlation of alleles in a subpopulation relative to their expected correlation in the total population
- $D'^2_{is}$ is the correlation of the appearance of two alleles on the same gamete in a subpopulation relative to that of the total population
- $D'^2_{st}$ is the variance of LD in the total population

Consider a comparison between two loci, A and B. Here, $x_{i,k}$ and $y_{j,k}$ are the frequencies of the $i^{th}$ and $j^{th}$ alleles at loci A and B in the $k^{th}$ subpopulation, $g_{ij,k}$ is the frequency of gametes $A_iB_j$ in the $k^{th}$ subpopulation. Averages of these values are denoted with bars. These statistics may be calculated as follows:

$$D^2_{IT} = E\{\sum_{i,j}(g_{ij,k} - \bar{x_i}\bar{y_j})^2\}$$

$$D^2_{IS} = E\{\sum_{i,j}(g_{ij,k} - x_{i,k}y_{j,k})^2\}$$

$$D^2_{ST} = E\{\sum_{i,j}(x_{i,k}y_{j,k} - \bar{x_i}\bar{y_j})^2\}$$

$$D^2_{IS} = E\{\sum_{i,j}(g_{ij,k} - \bar{g_{ij}})^2\}$$

$$D^2_{IT} = E\{\sum_{i,j}(\bar{g_{ij}} - \bar{x_i}\bar{y_j})^2\}$$

**Implementation and Architecture**

The ohtadtats package includes five functions: `dstat`, `dwrapper`, `dheatmap`, `dparallel`, and `dfilter`. Detailed descriptions and example code can be found at https://github.com/pfpetrowski/OhtaDStats, as well as in the documentation of the R package.

The first of these functions, `dstat`, is the workhorse of the package. This function computes each of Ohta's D statistics for a given pair of loci, and returns these results in a vector. The `dstat` function also returns the number of subpopulations included in the analysis. This number may be less than the total number of subpopulations as a result of filtering. To avoid spurious associations between alleles that are not truly in LD, `dstat` has an initial allele frequency filtering step designed to remove loci that fall below a specified minor allele frequency threshold. During the filtering step, `dstat` also removes any subpopulations which have a minor allele frequency below a given threshold. This feature prevents subpopulations which are fixed or nearly fixed at a given locus from appearing to be under selection when in reality the effect is due to small sample size. Both of these minor allele frequency thresholds are modifiable

arguments with can be changed by the user on demand. The `dstat` function returns a vector containing the number of populations included in the calculation and each of Ohta's five D statistics for the specified pair of markers.

It is important to note that the `dstat` function returns raw D statistic values. To assess statistical significance, `dstat` can be used to generate an empirical null distribution, as was used in [6]. Null distributions will vary by organism and model system, but the tools provided in ohtadstats can be used for their creation. This is achieved by implementing the function on a large number of pairs of physically unlinked SNPs.

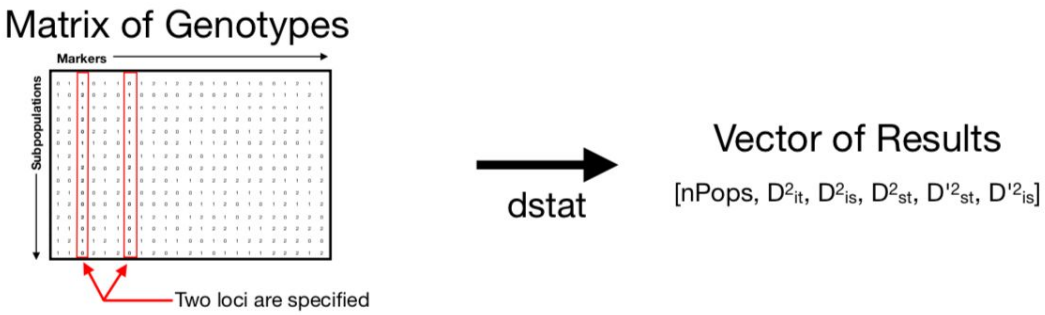


**Figure 1.** The `dstat` function calculates Ohta's D statistics for two specified loci.

The `dwrapper` function computes Ohta's D statistics for all possible pairs of loci in a matrix of genotypes. The result is returned as a list of matrices, with one matrix for each of Ohta's D statistics along with a matrix specifying the number of populations used for each comparison. This output format simplifies the process of looking up a D statistic for a specific pair of loci. It is important to note that because `dwrapper` evaluates all pairwise combinations of loci, it

scales on the order of $n^2$, where n is the number of genetic markers represented in the dataset. This means that the number of pairwise comparisons to be made scales exponentially with the number of markers being evaluated. This is not a problem for small datasets. Indeed, we successfully executed the `dwrapper` function on a dataset that included 100 SNPs from 1417 individuals using a 1.3GHz, 8GB RAM MacBook Air in only two minutes. This dataset is a subset of the data used by Beissinger et al. (2016) [6] and is included in the package as beissinger_data.rda. For large datasets (ie thousands of markers), we suggest parallelization via the `dparallel` function.

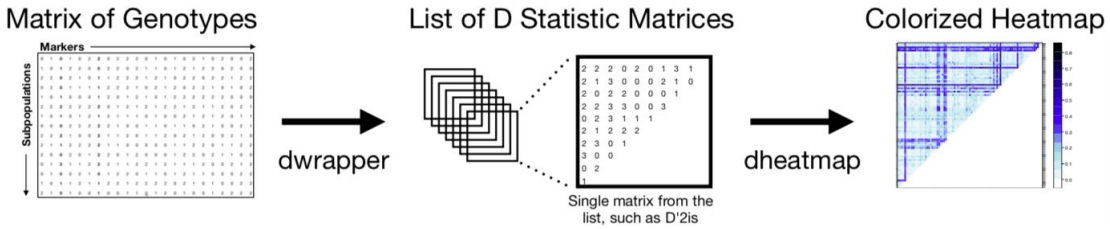


**Figure 2.** Given a single matrix of genotypes, `dwrapper` will produce a matrix for each of Ohta's D statistics with pairwise comparisons of each locus. Dheatmap will produce a colorized map for visualization of the values in a matrix.

The `dheatmap` function provides a convenient tool for data visualization. This function, which is based on the levelplot function from the lattice R package [12] takes any of the matrix outputs provided by the `dwrapper` function and returns a colorized heat map, making patterns of LD visible. Options are provided which allow the user to modify the colors used and how those colors are scaled. The `dheatmap` function provides three modes for the scaling of the colors. The

first mode, “linear”, is appropriate for most use cases. In this mode, the values in the matrix are distributed continuously across the color spectrum. The “truncated” mode is provided for use on the ratio matrices, where division by small numbers may cause certain values to be many orders of magnitude greater than others. In these cases, using “linear” is not ideal because large values will drive mid-magnitude values towards the extreme low end of the color spectrum. The “truncated” mode corrects for this issue by changing values greater than one to a value just higher than one. Colors are then scaled across this new spectrum of values. The final mode, “binned”, operates similarly to “truncated”, except that colors are not scaled across the new spectrum of values. Instead, values are placed into one of five bins, and colored accordingly.

The `dparallel` function is designed to facilitate parallelization of `dstat` on commercial high throughput computing platforms. By pairing a simple R script with a scheduler such as slurm [13], massive datasets on the scale commonly seen today can be analyzed in a reasonable amount of time. This function works by generating a virtual table of locus pairs to compare and executing `dstat` on each. Using this method, a number of equally sized jobs limited only by time and computational resources can be performed. The `dparallel` function is not meant to be used directly in the R environment. Instead it is designed to be set up in an R script, multiple instances of which are then spawned using a slurm scheduler array, or equivalent (Figure 3). Example scripts of this setup are available in the OhtaDStats GitHub repository

(https://github.com/pfpetrowski/OhtaDStats).

Lastly, the `dfilter` function is provided to perform the basic data preprocessing step of removing any subpopulations from the dataset if that subpopulation is too small. The `dfilter` function works by taking a dataset and an integer value for the minimum number of samples, and returning that dataset with only subpopulations that meet the threshold. Similar to the minor allele frequency filtering that is performed within the `dstat` function, this mitigates the danger of small sample sizes leading to spuriously large values of D.

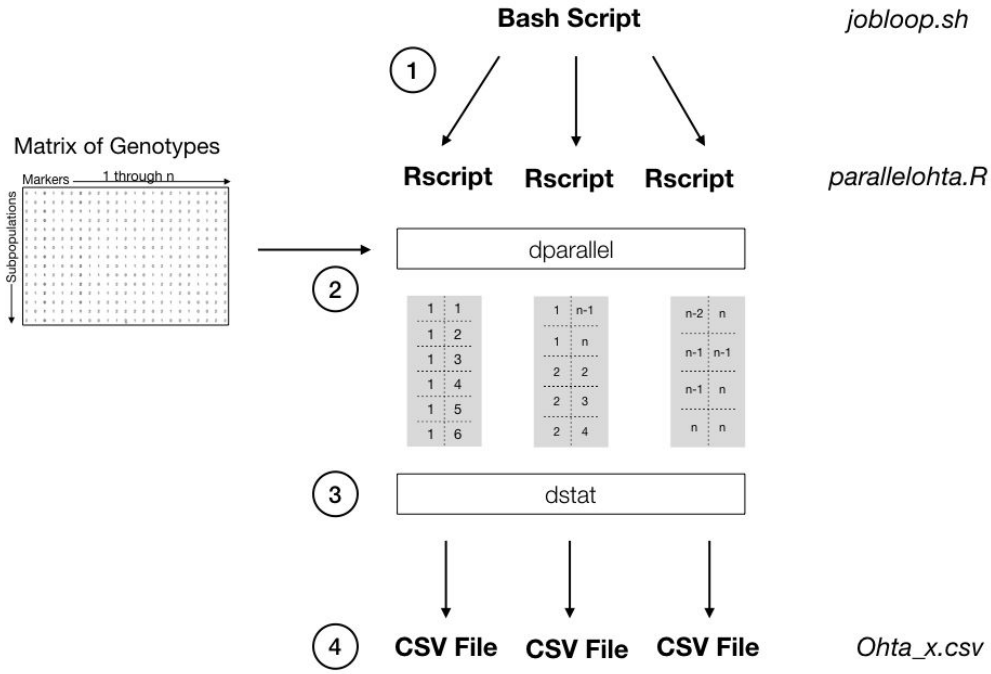


**Figure 3.** The `dparallel` workflow. Specific files associated with the general steps are in italics to the right of the diagram. 1) A bash script initializes a specified number of R processes, each of which executes the `dparallel` function on the specified dataset. 2) Each `dparallel` process infers a unique set of locus pairs for which to compute Ohta's D statistics. 3) Each R process calls `dstat` on each row (pair of loci) in its unique set. 4) Results are returned in csv files.

## Quality Control

To ensure that this package accurately calculates Ohta's D statistics, We simulated a small dataset containing 18 individuals across 3 subpopulations and three loci. We evaluated this data set using an implementation of LinkDOS [9] available at Genepop on the Web [14], and also using the ohtadstats package to ensure that results were equivalent. The sample dataset and code are available in the GitHub repository.In addition, examples included in this package are tested daily on the CRAN servers across Windows, MacOS, and Unix operating systems.

## Operating system

The ohtadstats package is designed for use with R versions 3.0.0 or later. R is supported on Windows, MacOS, and major Linux distributions. Minimum operating system versions are as follows:

Windows: Windows 7

MacOS: MacOS 10.9 (Mavericks)

Ubuntu: 14.04 (Trusty)

## List of contributors

Paul F. Petrowski, Timothy M. Beissinger, Elizabeth G. King

## Software location

### *Archive*

***Name:*** ohtadstats

***Persistent identifier:*** https://doi.org/10.5281/zenodo.1406484

***Licence:*** MIT

***Publisher:*** Paul Petrowski

***Version published:*** 2.1.1

***Date published:*** 20/03/19

### Code repository

***Name:*** OhtaDStats

***Identifier:*** https://github.com/pfpetrowski/OhtaDStats

***Licence:*** MIT

***Date published:*** 18/03/19

## Reuse potential

Ohta's D statistics are useful quantities for assessing linkage disequilibrium in genomic data sets. As such, this package may be useful to anyone looking to quantify linkage disequilibrium in their system of study. This includes any individual investigating the fields of population, quantitative, or evolutionary genetics. A typical use case may involve looking across a number of subpopulations of a species in an effort to detect evidence of selection. Other methods of using LD as a measurement of selection have been previously

described, including the integrated haplotype score (iHS)[15] and extended haplotype homozygosity (EHH) [16]. These commonly-applied methods are designed to identify hard sweeps in a single population, while Ohta's D statistics are best applied to data including multiple populations, and have the potential to additionally identify epistatic selection. Therefore, these approaches may be complementary -- researchers may find different selected loci based on Ohta's D stats than by applying iHS or EHH, and vice versa.

**Acknowledgements**

We would like to thank Jake Gotberg from the Mizzou Research Computing Support Service for contributing his time and expertise in setting up an efficient parallelization workflow. Computation for this work was performed on the high performance computing infrastructure provided by Research Computing Support Services and in part by the National Science Foundation under grant number CNS-1429294 at the University of Missouri, Columbia MO.

**Funding statement**

This research was supported by funding from the USDA Agricultural Research Service. PFP is funded by the University of Missouri Life Sciences Fellowship and a training grant from the National Institute of Health (T32GM008396).

## Competing interests

The authors declare that they have no competing interests.

# CHAPTER TWO: HAPLOTYPE INFERENCE

## INTRODUCTION

Pooled sequencing data is used broadly throughout many subfields of genetics. When population level parameters are of more interest than individual level parameters, pooled sequencing provides a low cost alternative to the independent sequencing of multiple individuals [17]. Sometimes the sample of interest is inherently pooled, such as when samples are drawn from the ambient environment. Ley *et al.* [18] used pooled sequencing to quantify the abundance of different types of microbes in the human gut. Hastings *et al.* [19], Hastings and Smith [20], and Takala *et al.* [21] developed methods to infer strains of malaria present in human blood samples. Other times samples are deliberately pooled to save time and money [22–24]. This is commonly the case in laboratory based evolution experiments, where the direct sequencing of every individual involved is much less practical than simply pooling them [22].

The convenience of pooling comes with the tradeoff of losing information at the individual level. When samples are pooled, the resulting sequence data consists of allele frequencies, rather than allele calls, as indicated in figure 4. It is not possible to say which individual had a particular allele. This tradeoff is acceptable when population level parameters are of primary interest, as is the case in laboratory based evolution experiments. Using frameworks such as

Evolve and Resequence (E&R) [25], scientists attempt to quantify how allele frequencies change in response to selective pressures.

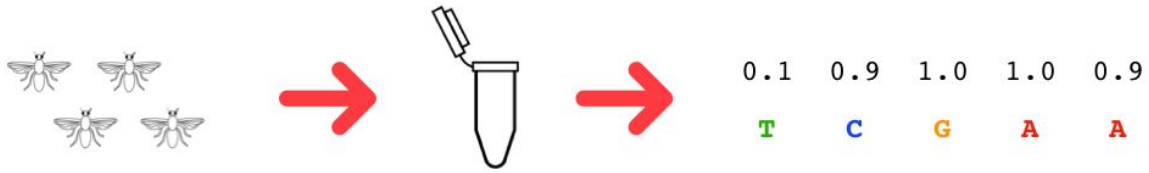


**Figure 4.** Flies from one treatment group are pooled and sequenced together. The resulting sequence reads represent allele frequencies rather than one individual's base calls.

In these experiments, a population of individuals are subjected to a particular environment or selective pressure, and bred under those conditions for some number of generations. At the end, a sample of the resulting population is sequenced and compared to the initial population. Quantifying which allele frequencies have changed should give insights into which loci are responsible for adaptation to the selective pressure. Usually the initial population is derived from a population with known ancestors such as the DGRP [26] or DSPR [27] in *D. melanogaster*, or MAGIC populations in plant systems [28]. For instance, all of the flies in the DSPR are descendents from the same set of founder flies. The population descends from eight founder flies whose progeny were bred randomly for 50 generations. These flies were inbred via a full sib mating scheme for 20 generations to generate 800 recombinant inbred lines (RILs).This scheme results in each individual's genome being a mosaic of haplotypes, each of which can be traced back to one of the eight original founders. By characterizing how

haplotype frequencies change over time, we can still get an idea of where potentially causative positions are located.

While this is a powerful framework, accurate measurement of allele frequencies poses significant challenges. Read coverage must be high in order for the measurement to be believable. This is for the simple reason that a 25% allele frequency is much more trustworthy when the allele is observed in 25 out of 100 reads than when it is observed in 1 out of 4 reads. Frequencies obtained at low read coverages are inherently more vulnerable to sampling error. Sequencing at the necessary levels of coverage gets expensive quickly. Fortunately it is possible to accurately assess allele frequencies from low coverage pooled sequence data by taking a more circuitous approach. When the population used in the experiment comes from founders with known sequences, it is possible to use the low coverage allele frequencies from multiple markers to infer the founder haplotype frequencies. A haplotype is a group of alleles that are co-inherited from a single parent. Haplotype frequency inference works with low coverage sequence data because it uses data from multiple genomic positions, rather than relying on a single position. This “breadth” of data means that haplotype frequencies can be reliably inferred. Once the founder haplotype frequencies are known, it is a straightforward process to calculate what the allele frequencies should be at each position.

A number of methods exist to infer founder haplotype frequencies from pooled sequence data. In one of the earliest attempts, Long *et al.* [26] used a

regression model to address the haplotype inference problem, but it does not estimate rare haplotypes well. Kessner *et al.* [27]used a maximum likelihood approach, which makes estimates based on mapping raw reads to a reference. It performs at its best when the distribution of haplotype frequencies is nonuniform. Cao and Sun [28]used a system of linear equations to approach the problem. It too struggles with low frequency haplotypes. Burke *et al.* [29] used a calculus based optimization approach. Franssen *et al.* [30] developed a method that can reconstruct founder haplotypes from pooled sequence data even if they are not known in advance by measuring allele frequency trajectories, but it requires data from multiple time points.

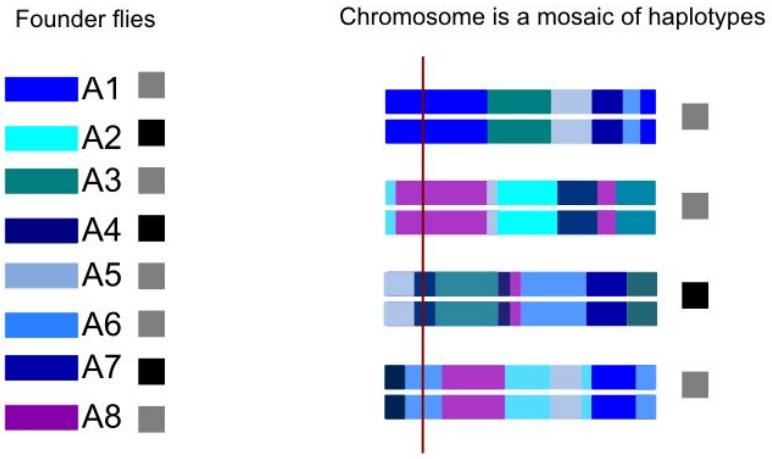


**Figure 5.** The DSPR. Genomes of descendent flies (right) are mosaics of haplotypes from the founder flies (left). Haplotypes will share SNPs (grey and black boxes) which can be derived from the haplotype frequencies once they are estimated.

In this chapter we perform benchmarking tests on the gradient method developed by Burke *et al*., and determine the optimal number of SNPs to pass to the algorithm in order to estimate haplotype frequencies. We also performed tests to determine if the number of individuals in the pool had any impact on the performance of the algorithm. Next, we introduce a new approach to the haplotype inference problem that is based on a genetic algorithm. Genetic algorithms are an appealing approach to this problem because they handle epistatic effects well, that is, optimizations in which one parameter's value is determined to some extent by the values of other parameters [31]. We compare this method to the gradient method by simulating pools from the known sequences of DSPR recombinant inbred lines.

**METHODS**

In this study we aimed to accomplish two main goals. First, to characterize the window size and pool size for which the gradient method achieves the best performance. We performed simulations to determine the ideal window size of genomic positions to pass into the algorithm, and also to determine if the algorithm performs better when more individuals are present in the pool or fewer. Upon establishing that the ideal window size is approximately 3200 SNPs, we used that as our window size in all later analyses. Next, we sought to compare the performance of the gradient based method to a new genetic algorithm that we developed.

The existing method is a gradient based method developed by Burke *et al.* [29]. and a genetic algorithm approach that we developed. To evaluate the frequency of haplotypes in a pool for a given position, both algorithms require information about the sequences at neighboring sites. This is because a single position will usually not be enough to discern haplotypes apart, especially when the number of founder haplotypes are large.

To assess the performance of the two methods, we used simulated data from the DSPR (figure 5). We took recombinant inbred lines with known sequences from the DSPR to simulate pooled sequencing datasets. We randomly added lines to *in silico* pools and computed observed reference allele frequencies for each position. These simulated minor allele frequencies are then fed to the algorithms. Because the original sequences are known, the founder haplotype for a given segment of the genome can be ascertained using a hidden markov model [32] (see [33] for a detailed explanation and tutorial on hidden markov models). To measure the accuracy of the algorithms we tested, we calculated the sum squared error between the estimated and known set of haplotype frequencies for a given position.

**Gradient Method**

A method developed by Burke et al. implements the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm [34, 35] to optimize a vector of founder haplotype frequencies. We will refer to this method as the

gradient based method, because the algorithm makes use of gradient information to determine how the estimate is updated. The algorithm attempts to minimize the sum squared error between the sequence of observed reference allele frequencies, and the sequence of predicted reference allele frequencies that would occur given a frequency estimate.

Let F be the set of founder haplotypes, M be a vector of haplotype frequencies (the quantity which we are aiming to optimize), and Y be the observed reference allele frequency that would be observed given haplotype frequencies M.

$$Y = F \cdot M$$

Now let X be the observed reference allele frequency, and n be the number of founder haplotypes. When M is an exact representation of the true haplotype frequency in the pool, the sum squared error between the two vectors will be zero.

$$Error = \sum_{i=1}^{n} (X_i - Y_i)^2$$

Additionally, a constraint term is imposed to ensure that the sum of haplotype frequencies does not exceed 1.

$$Constraint = (100 \cdot w \cdot ((\sum_{j=1}^{n} M_j) - 1)^2$$

Where w is the length of sequence of DNA being evaluated. Note that F is a matrix of size (n, w), and M is a vector of length n.

The measure of accuracy for a given vector of haplotype frequencies is given as:

$$Score = Error + Constraint$$

The BFGS algorithm is used to minimize this quantity. It was implemented in R via the `optim` function.

**Genetic Algorithm**

As a new approach to the haplotype inference problem, we designed a genetic algorithm that can accurately infer the founder haplotypes given a sequence of allele frequencies. Generally speaking, genetic algorithms work by iteratively making a set of guesses at the answer to a problem, evaluating those guesses according to some objective function, and then making a new set of guesses based on modifying the best guesses from the previous round. Over the course of many iterations of this procedure, the “best guess” is steadily refined

until it approximates or achieves optimality. Guesses are typically referred to as individuals, and each sets of individuals per iteration are the population. Each individual is a vector of parameters aiming to be optimized. The inspiration for this class of algorithms draws heavily on the Darwinian concept of natural selection, hence the name “genetic algorithm”. This procedure can be described as follows:

1. Generate a population of individuals, where each individual is a vector of parameter values.
2. Score each individual according to the fitness function.
3. Generate a new population of individuals, where each new individual is a modification of one of the top performing individuals from the previous generation.
4. Repeat until optimality is achieved or until a certain number of generations have been reached.

For an in depth tutorial on genetic algorithms, we recommend [36].

In our implementation of a genetic algorithm, the parameters we are attempting to optimize are the frequencies of each founder haplotype. To assess the fitness of each individual, we apply our fitness function as described below.

**Fitness function**

Let F be the set of founder haplotypes, M be an individual member of the GA population,

w be the length of sequence of DNA being evaluated, n be the number of founder haplotypes, and Y be the reference allele frequency that one would observe if M were

correct. F is a matrix of size (n, ws), and M is a vector of length n.

$$Y = F \cdot M$$

This is equivalent to asking, “for a given guess of haplotype frequencies, what would be the observed set of reference allele frequencies if it were correct?” The sum squared error between the “guessed” set of minor allele frequencies and the observed minor allele frequencies is then calculated, and assigned to be the fitness score for the guess.

Now, let X be the observed reference allele frequency from the pool-seq data. If M is

perfectly accurate, Y and X should be identical. We measure similarity by taking the sum squared error between X and Y.

$$Error = \sum_{i=1}^{n} (X_i - Y_i)^2$$

The error measurement is not the final measure of fitness. We impose additional constraints that penalize frequencies with negative values and sets of frequencies that sum to be greater than 1.

Penalty 1 is the sum of the frequencies in the vector minus 1. It will be equal to zero if the vector sums to 1.

$$pen1 = (\sum_{i=1}^{n} |M_i|) - 1$$

Penalty 2 assesses if there are any frequency values that are negative. It is simply the absolute value of the sum of negative values in the vector. The final fitness assignment is given as:

$$Fitness = error + pen1 + pen2$$

Note that in the biological sense high fitness is typically considered a good thing, but in

this fitness is a measure of error and we are seeking to minimize it.

**Mutation & Inversion**

Aside from the fitness evaluation, the other key aspect to GAs is the set of rules governing how new populations are created from the previous. It is essential that the new population be in some way different from the original, otherwise the top performing individual does not change from generation to generation, and no convergence to optimality can occur. To ensure that the populations change over time and converge on the optimal set of haplotype frequencies, we define a set of mutations that are applied to individuals of high fitness. These mutations make small changes to each individual that are passed on to the “offspring” in the next generation.

For our algorithm, we applied two different types of mutation to successful individuals in our populations. The first is a simple function which deducts small, normally distributed values from one element of the vector, and adds that same value to a different element. The second is referred to as an “inversion”, whereby two elements switch positions in the vector.

The first mutation ensures that the numerical values for each individual changes from generation to generation. The second, the inversion, is a concession to the fact that purely due to random chance, all individuals in a population may wind up with very similar values at a particular element. When this occurs, it is similar to a lack of genetic diversity in natural populations. Natural populations with low genetic diversity are slow to adapt; *in silico* populations are slow to converge, or may not produce an accurate prediction at

all. Inversion ensures that if this happens, a different value from a different element can be swapped in, thus injecting some variance into the population.

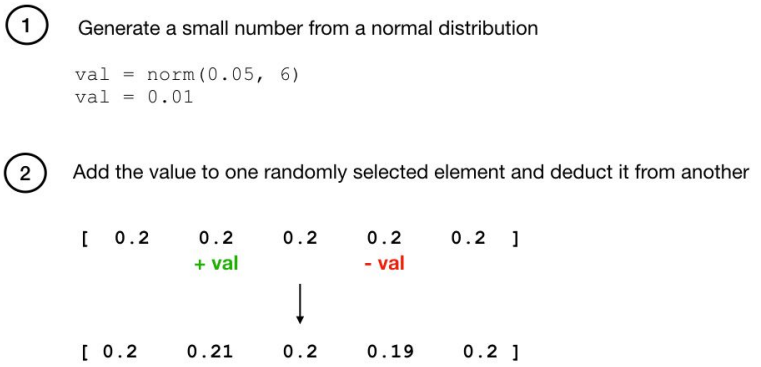


**Figure 6.** Illustration of how mutation works in the genetic algorithm. Small randomly distributed numbers are added to one element and deducted from another.


**Figure 7.** Illustration of inversion. The order of two elements in the vector are switched.

## RESULTS

We first sought to determine the window size of neighboring positions that leads to the best performance with the gradient based method. We measured the performance of the gradient based method on a simulated dataset, using window

sizes of 50, 100, 200, 400, 800, 1600, 3200, 6400, 12800, and 25600, and pool sizes of 100, 400, and 800. Performance was measured as the sum squared error between the estimated haplotype frequencies and the haplotype frequencies known to be true. We found that the window size that yielded the lowest error was 3200 bases. For this reason we used this as our window size in remaining analyses.

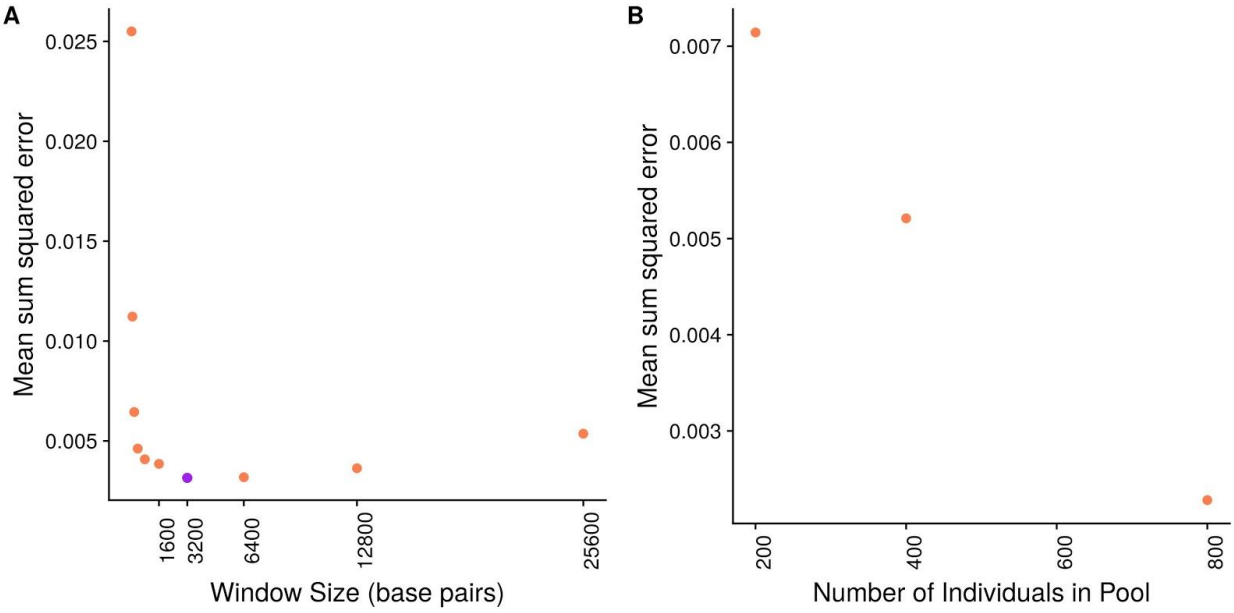


**Figure 8.** A) Small window sizes do not provide the algorithm with enough information, and predictions are poor. The optimal window size is ~3200 base pairs. As the window size increases beyond that, the probability of spanning multiple recombination breakpoints increases, and error slowly goes up again. Window sizes between 50 and 800 are not labelled. Average error over all pool sizes are shown. B) Error declines linearly as the number of individuals in the pool goes up.

**Comparison of Genetic Algorithm to Gradient Method**

To compare the performance of the genetic algorithm with the performance of the gradient based optimization method, we simulated a pool-seq dataset by adding flies with known sequences to an *in silico* pool and computing observed reference allele frequencies for each position. Each algorithm inferred haplotype frequencies at every 50th position, using a window of 1600 SNPs on either side of the position of interest.

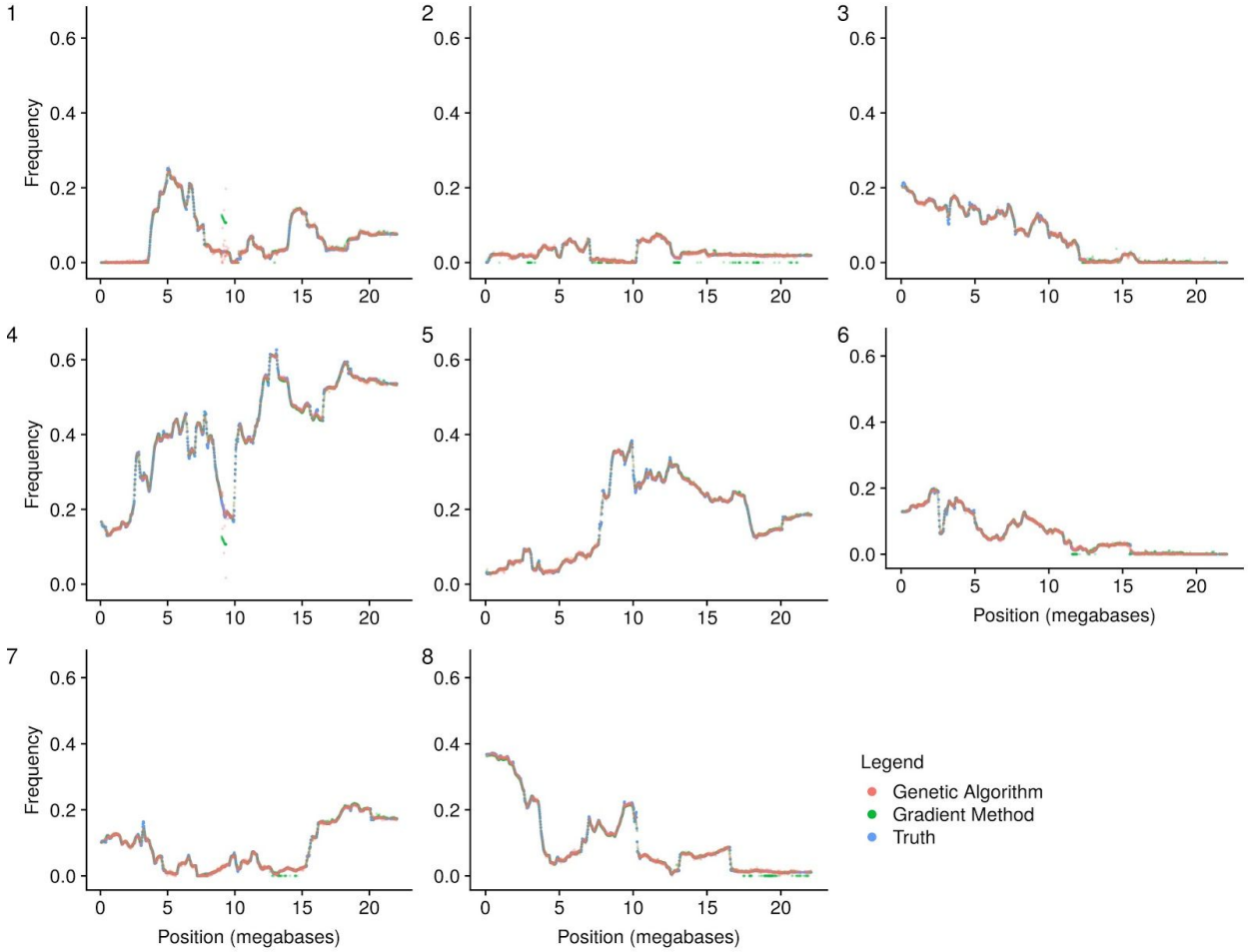


**Figure 9.** Estimates of haplotype frequencies given by the genetic algorithm (in red) and the gradient method (in green) compared to the ground truth (in blue). Each plot is labeled in the upper left corner with the founder haplotype it corresponds to.

We calculated an overall mean squared error 0.000485 for the gradient based method and 0.00190 for the genetic algorithm.

## DISCUSSION

In this chapter, we explored the uses, merits, weaknesses, and applications of pooled sequence data. We gave special attention to how pool-seq has potent implications for laboratory evolution experiments, namely that haplotype derived allele frequencies are more trustworthy than what comes directly off the machine because they make use of the breadth of sequence information in addition to the depth. Thus, methods that can accurately infer haplotype frequencies from the pooled sequence data are an important component of these studies. We performed benchmarking on a method that was previously developed to solve that problem, and show that a window size of approximately 3200 base pairs total is ideal in our system. We also show that in simulations, more individuals in a pool leads to better performance.

Next, we introduced a new approach to the problem based on genetic programming, a paradigm in which a series of iteratively more accurate "guesses" are made at the solution to the problem. We show that this new approach outperforms the first on a simulated dataset. Genetic algorithms are an appealing approach to haplotype inference due to the "epistatic" nature of the problem [31]. That is, the parameters may not be solved independently of one another because a change to one necessitates a change to another.

Another (though unverified) possibility is that the genetic algorithm is more capable of handling situations in which founder haplotypes along a given genomic segment have similar sequence identity. When this occurs, it can be

difficult or impossible for any algorithm to correctly infer the correct frequencies. In the worst case scenario, two or more founder sequences are identical at a given genomic segment. In this case, it should not be possible to tell which founder reads along this position actually came from. In these situations, it is possible that the gradient method favors one haplotype over the other as a result of gradient descent. In contrast, the genetic algorithm makes no use of gradient information, and is blind to the state of the previous iteration.This question may be answered more definitively by taking a closer look at the genomic positions where the gradient method performs poorly in comparison to the genetic algorithm. We also note that the genetic algorithm seems to perform well even in situations where founder haplotype frequencies are low (see figure 9).

Methods that can accurately infer haplotype frequencies from pooled sequencing data greatly aid our ability to conduct laboratory based evolution experiments. It is key to note that our genetic algorithm is suitable only for situations in which the founder haplotypes are known in advance. It is not capable of inferring unknown haplotypes from the pooled data. Franssen *et al.* [30] have proposed a method by which unknown haplotypes can be reconstructed. In conjunction, these two methods may be capable of inferring unknown haplotype frequencies from pooled sequence data.

## FUTURE DIRECTIONS

Although the genetic algorithm performs well, it is not without room for improvement. The current mutation scheme is effective but simple. One potential improvement would be to make it so that the mean of the normal distribution for from which the mutation value is drawn decays as the number of generations goes on. This change would allow larger mutations to take place in the early generations when the population is farthest from optimality, but finer adjustments would take place later on when the population is closer to optimality.

Additionally, we note that all computations described in this chapter deal with simulated data, where certain real world variables are not considered. For instance, sequencing error is a factor that would certainly lead to reduced performance, but in simulations we assume that all reads are accurate. Future work in this space should consider the use of an empirical dataset.

There are a number of other parameters whose impact on the accuracy of haplotype inference algorithms could be assessed via simulation. Read coverage, marker density, and window size based on genetic distance rather than the number of SNPs are all possibilities that could be explored.

## REFERENCES


[1] Vitti JJ, Grossman SR, Sabeti PC 2013 Detecting Natural Selection in Genomic Data. *Annual Review of Genetics* 47(1): 97–120. doi: 10.1146/annurev-genet-111212-133526.

[2] Maynard J, Haigh J 1974 The hitch-hiking effect of a favourable gene. *Genetics Research* 89(5–6): 391–403. doi: 10.1017/S0016672308009579.

[3] Ohta T 1982 Linkage disequilibrium due to random genetic drift in finite subdivided populations. *Proceedings of the National Academy of Sciences of the United States of America* 79(6): 1940–1944.

[4] Wright S 1922 Coefficients of Inbreeding and Relationship. *The American Naturalist* 56(645): 330–338. doi: 10.1086/279872.

[5] Miyashita NT, Aguadé M, Langley CH 1993 Linkage disequilibrium in the white locus region of Drosophila melanogaster. *Genetical Research* 62(02): 101. doi: 10.1017/S0016672300031694.

[6] Beissinger TM, Gholami M, Erbe M et al. 2016 Using the variability of linkage disequilibrium between subpopulations to infer sweeps and epistatic selection in a diverse panel of chickens. *Heredity* 116(2): 158–166. doi: 10.1038/hdy.2015.81.

[7] Song B-H, Windsor AJ, Schmid KJ et al. 2009 Multilocus Patterns of Nucleotide Diversity, Population Structure and Linkage Disequilibrium in Boechera stricta, a Wild Relative of Arabidopsis. *Genetics* 181(3): 1021–1033. doi: 10.1534/genetics.108.095364.

[8] Black WC, Krafsur ES 1985 A FORTRAN program for the calculation and analysis of two-locus linkage disequilibrium coefficients. *Theoretical and Applied Genetics* 70(5): 491–496. doi: 10.1007/BF00305981.

[9] Garnier-Gere P, Dillmann C 1992 A Computer Program for Testing Pairwise Linkage Disequilibria in Subdivided Populations. *Journal of Heredity* 83(3): 239–239. doi: 10.1093/oxfordjournals.jhered.a111204.

[10] YEH F 1997 Population genetic analysis of co-dominant and dominant marker and quantitative traits. *Belgian J Bot* 130129–157.

[11] R Core Team 2018 R: A language and environment for statistical computing. Vienna, Austria, R Foundation for Statistical Computing.

[12] Sarkar D 2008 Lattice: Multivariate Data Visualization with R. New York, Springer.

[13] Jette MA, Yoo AB, Grondona M 2002 SLURM: Simple Linux Utility for Resource Management. In: Lect. Notes Comput. Sci. Proc. Job Sched. Strateg. Parallel Process. JSSPP 2003. Springer-Verlag, pp 44–60.

[14] Raymond M, Rousset F 1995 GENEPOP (Version 1.2): Population Genetics Software for Exact Tests and Ecumenicism. *Journal of Heredity* 86(3): 248–249. doi: 10.1093/oxfordjournals.jhered.a111573.
[15] Voight BF, Kudaravalli S, Wen X, Pritchard JK 2006 A Map of Recent Positive Selection in the Human Genome. *PLOS Biology* 4(3): e72. doi: 10.1371/journal.pbio.0040072.
[16] Sabeti PC, Reich DE, Higgins JM et al. 2002 Detecting recent positive selection in the human genome from haplotype structure. *Nature* 419(6909): 832–837. doi: 10.1038/nature01140.
[17] Schlötterer C, Tobler R, Kofler R, Nolte V 2014 Sequencing pools of individuals - mining genome-wide polymorphism data without big funding. *Nature Reviews Genetics* 15(11): 749–763. doi: 10.1038/nrg3803.
[18] Ley RE, Turnbaugh PJ, Klein S, Gordon JI 2006 Human gut microbes associated with obesity. *Nature* 444(7122): 1022. doi: 10.1038/4441022a.
[19] Hastings IM, Nsanzabana C, Smith TA 2010 A Comparison of Methods to Detect and Quantify the Markers of Antimalarial Drug Resistance. *The American Journal of Tropical Medicine and Hygiene* 83(3): 489–495. doi: 10.4269/ajtmh.2010.10-0072.
[20] Hastings IM, Smith TA 2008 MalHaploFreq: A computer programme for estimating malaria haplotype frequencies from blood samples. *Malaria Journal* 7(1): 130. doi: 10.1186/1475-2875-7-130.
[21] Takala SL, Smith DL, Stine OC et al. 2006 A high-throughput method for quantifying alleles and haplotypes of the malaria vaccine candidate Plasmodium falciparum merozoite surface protein-1 19 kDa. *Malaria Journal* 5(1): 31. doi: 10.1186/1475-2875-5-31.
[22] Futschik A, Schlötterer C 2010 The Next Generation of Molecular Markers From Massively Parallel Sequencing of Pooled DNA Samples. *Genetics* 186(1): 207–218. doi: 10.1534/genetics.110.114397.
[23] Huang W, Richards S, Carbone MA et al. 2012 Epistasis dominates the genetic architecture of Drosophila quantitative traits. *Proceedings of the National Academy of Sciences* 109(39): 15553–15559. doi: 10.1073/pnas.1213423109.
[24] OROZCO-terWENGEL P, Kapun M, Nolte V et al. 2012 Adaptation of Drosophila to a novel laboratory environment reveals temporally heterogeneous trajectories of selected alleles. *Molecular Ecology* 21(20): 4931–4941. doi: 10.1111/j.1365-294X.2012.05673.x.
[25] Long A, Liti G, Luptak A, Tenaillon O 2015 Elucidating the molecular architecture of adaptation via evolve and resequence experiments. *Nature*

*reviews Genetics* 16(10): 567–582. doi: 10.1038/nrg3937.

[26] Long Q, Jeffares DC, Zhang Q et al. 2011 PoolHap: Inferring Haplotype Frequencies from Pooled Samples by Next Generation Sequencing. *PLOS ONE* 6(1): e15292. doi: 10.1371/journal.pone.0015292.

[27] Kessner D, Turner TL, Novembre J 2013 Maximum Likelihood Estimation of Frequencies of Known Haplotypes from Pooled Sequence Data. *Molecular Biology and Evolution* 30(5): 1145–1158. doi: 10.1093/molbev/mst016.

[28] Cao C-C, Sun X 2015 Accurate estimation of haplotype frequency from pooled sequencing data and cost-effective identification of rare haplotype carriers by overlapping pool sequencing. *Bioinformatics* 31(4): 515–522. doi: 10.1093/bioinformatics/btu670.

[29] Burke MK, King EG, Shahrestani P et al. 2014 Genome-Wide Association Study of Extreme Longevity in Drosophila melanogaster. *Genome Biology and Evolution* 6(1): 1–11. doi: 10.1093/gbe/evt180.

[30] Franssen SU, Barton NH, Schlötterer C 2017 Reconstruction of Haplotype-Blocks Selected during Experimental Evolution. *Molecular Biology and Evolution* 34(1): 174–184. doi: 10.1093/molbev/msw210.

[31] Davidor Y 1990 Epistasis Variance: Suitability of a Representation to Genetic Algorithms. 16.

[32] King EG, Macdonald SJ, Long AD 2012 Properties and Power of the Drosophila Synthetic Population Resource for the Routine Dissection of Complex Traits. *Genetics* 191(3): 935–949. doi: 10.1534/genetics.112.138537.

[33] Rabiner LR 1989 A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE* 77(2): 257–286. doi: 10.1109/5.18626.

[34] Goldfarb D 1970 A family of variable-metric methods derived by variational means. *Mathematics of Computation* 24(109): 23–26. doi: 10.1090/S0025-5718-1970-0258249-6.

[35] Fletcher R 1970 A new approach to variable metric algorithms. *The Computer Journal* 13(3): 317–322. doi: 10.1093/comjnl/13.3.317.

[36] Whitley D 1994 A genetic algorithm tutorial. *Statistics and Computing* 4(2): 65–85. doi: 10.1007/BF00175354.