# A bioinformatics workflow for detecting signatures of selection in genomic data

Murray Cadzow[1], James Boocock[1], Hoang Tan Nguyen[1,2], Phillip Wilcox[1,3], Tony R Merriman[1] and Michael A Black[1]

[1]Department of Biochemistry, University of Otago
[2]Department of Mathematics and Statistics, University of Otago
[3]Scion Research, Rotorua, New Zealand

December 12, 2013

## Contents

# 1 Introduction

This selection analysis workflow utilizes genotype data derived from next-generation sequencing (NGS) or high-denisity microarray (e.g., "SNP chip") experiments to identify the presence of signatures of selection. The tools used to detect selection are dependent on the selection signature being investigated (Sabeti *et al.*, 2006). The pipeline presented here generates various output files containing within- and between-population selection signatures. The starting point for the analysis is a variant call format (VCF) file of the genotype data and populations of interest (Danecek *et al.*, 2011). Both $F_{ST}$ and Tajima's D can be calculated from standard genotype data (Weir and Cockerham, 1984; Tajima, 1989). To compute iHS, Rsb and Fay and Wu's H requires haplotype information, and thus the genotype data must be phased prior to calculation of these statistics (Voight *et al.*, 2006; Gautier and Vitalis, 2012; Fay and Wu, 2000). For phasing, shapeit2 is used, and for imputation impute2 is used (Howie *et al.*, 2009; Delaneau *et al.*, 2013). Furthermore these statistics also require ancestral allele information (Flicek *et al.*, 2012). The pipeline performs phasing if the VCF files do not contain phase information, and then performs ancestral allele annotation. Phasing can be performed on species other than humans, provided you have genetic maps in the format shapeit expects. Once complete, the rehh package for R provides a simple interface for implementing EHH-based analyses (Gautier and Vitalis, 2012). Here we have extended rehh to include penalties for gaps that match those used in the original iHS paper (Voight *et al.*, 2006). rehh is used to calculate iHH, iHS, iES and Rsb. To calculate Fay and Wu's H, a C program, variscan, was utilised (Vilella *et al.*, 2005). If a genetic map is present, the pipeline uses linear interpolation to calculate the map positions for each marker (Nievergelt *et al.*, 2004). To run the pipeline an ancestral FASTA file is required, and optionally a genetic map, if avaliable, is recommended. The pipeline is implemented in Python, and takes a VCF file as input. The output is a colection of files relating to selection statistics generated by the various software tools.

# 2 Getting Started

## 2.1 Prerequisites

The selection pipeline was developed on a 64-bit Ubuntu 13.04 system and has been tested on 64-bit Centos, Ubuntu 13.10 and Mac OSX (version 10.9 with some minor tweaking of the environment) installations. The pipeline should work on any 64-bit linux derivative assuming some basic libraries and tools are installed on the system. 8GB of RAM should be sufficient for all computation steps (imputation is the most RAM-intensive component of the pipeline).

- Python $>=$ 2.6
- Bourne-again Shell (Bash)
- Perl5
- R $>=$ 3.0.0 (with the ability to install packages into a library within your home directory structure)
- GNU Autotools
- GCC
- Git

The software is installed with the same permissions as the user than runs the script: if the user is not root then a local (i.e., user-writeable) R library is required. The program also installs the scripts to the user's /.local/bin directory. This directory should be added to the system PATH to give direct access to the programs from the command-line.

## 2.2 Python dependencies

The following Python packages are required for the pipeline:

- python-setuptools
- python-numpy
- python-scipy

  if using python $< 2.7$ the package argparse will need to be installed.

  It can be installed using the following command.

```
easy\_install-2.6 argparse
```

Most linux distributions provide these packages through the official package management repositories.

## 2.3 Downloading

The selection pipeline can be obtained at the url: https://github.com/smilefreak/selectionTools
To download run the following command in a terminal.

```
git clone https://github.com/smilefreak/selectionTools
```

## 2.4 Installation

To perform an automatic installation of the selection analysis pipeline, run the following command in the root of the directory in which the pipeline was installed (i.e., within the "selectionTools" directory).

```
./install.sh
```

The installation process creates a default configuration file located in the base directory of the pipeline. It also adds a program called selection_pipeline to the system path. To test that the program is installed correctly, run the following command at a terminal prompt.

```
selection_pipeline -h
```

If the above command does not work, make sure that /.local/bin/ is included in the PATH environment variable.

## 2.5 Genetic Maps and Impute Haplotypes

To use the phasing and imputation features of the pipeline requires both genetic map files and haplotype files. For humans, files that conform to the format required for shapeit and impute2 can be found here. For impute2, one reference is available here. Download and extract the archive to referencefiles/impute_ref and uncompress

the contents. For shapeit2, a genetic map can be found here. Download and extract the archive to reference-files/genetic_maps.

To use other reference files with the selection pipeline requires setting the following options in the config file. The question mark character "?" in the config is substituted by the chromosome number: this is used for reference files that are split on chromosomes.

```
...
genetic_map_prefix=genetic_map_chr?_combined_b37.txt
...
impute_map_prefix=genetic_map_chr?_combined_b37.txt
impute_reference_prefix=ALL_1000G_phase1integrated_v3_chr?_impute
...
```

If you decide to store the reference files in another location, further options require alteration in the config file:

```
...
genetic_map_dir= ${HOME}/selectionTools/referencefiles/genetic_maps
...
impute_map_dir= ${HOME}/selectionTools/referencefiles/impute_ref
impute_reference_dir= ${HOME}/selectionTools/referencefiles/impute_ref
...
```

The SHAPEIT and PLINK formats are accepted for genetic maps. The genetic map shown above is in SHAPEIT format and a genetic map in PLINK format can be located here.

To use genetic maps like this one will require changing both the settings mentioned above, although no specification of file format is necessary.

If your species of interest does not have a genetic map you can specify the –no-genetic-map, forcing the selection pipeline to use beagle phasing (no genetic map required) and calcule EHH statistics using the physical map.

## 2.6 Ancestral Fasta Files

The generation of results for iHS requires assigning the ancestral allele. The selection pipeline uses the ancestral alleles from the 6-way EPO (Enredo-Pecan-Ortheus) alignment pipeline. The files can be downloaded from here. Be sure to extract the contents of the archive after download. The default directory to store the ancestral reference files is

```
referencefiles/ancestral_ref/
```

If you downloaded your reference to a different location you can alter the following setting in your config file.

```
...
ancestral_fasta_dir = # directory you downloaded alignment to #
...
```

# 3 Tutorial

## 3.1 Selection Signatures at the Lactase Locus

### 3.1.1 Getting the Data

In humans, lactase is encoded by the LCT gene, which is located on Chromosome 2 at the coordinates: 136,545,410-136,594,750. For this example we will use a 10 megabase region containing the LCT, and genotype data from the CEU and YRI populations from the 1000 Genomes Project. In order to demonstrate the functionality of the pipeline we will use the chromosome 2 region 130,000,000-140,000,000. The lactase gene is an example of strong selection in the last 5,000-10,000 years in human populations, specifically those of European ancestory (Bersaglieri *et al.*, 2004). The tutorial commands will be given as examples using phasing and imputation or taking advantage of the fact that the 1000 genomes VCF files actually contain phasing information. If you choose to run the tutorial without using phasing and imputation you will only need to download the Ancestral allele information. To download the example dataset enter the following command:

```
git clone https://github.com/smilefreak/SelectionPipelineTestData
```

Navigate to the created SelectionPipelineTestData folder and extract *selection_pipeline_tutorial.tar.gz*.

```
cd SelectionPipelineTestData
tar xzf selection_pipeline_tutorial.tar.gz
```

## 3.2 Setting up the Pipeline Run

## 3.3 Population Files

Population files are required for any cross population comparisions. The commands below will initiate the data generation step. Population files are line seperated files, where the first line contains the population name, and every successive line contains an individual ID from that population.

```
<POPULATION_IDENTIFIER>
<INDIVIDUAL ID 1>
<INDIVIDUAL ID 2>
.......
<INDIVIDUAL ID N>
```

## 3.4 Run The Tutorial

The default configuration file is located in the base directory of the selection pipeline. To run the pipeline with phasing and imputation, execute the command below in the folder in which the example data were extracted, and change the –config-file parameter to match the location where the pipeline is installed.

```
multipop_selection_pipeline -p CEU_ids.txt -p YRI_ids.txt \
  -i CEU_YRI_lactase.vcf --config-file defaults.cfg \
  -a "--imputation" -c 2
```

As the 1000 genomes datasets are actually phased VCF to run the pipeline without phasing run the following command. The –phased-vcf option significantly speeds up the selection pipeline.

```
multipop_selection_pipeline -p CEU_id.txt -p YRI_ids.txt \
    -i CEU_YRI_lactase.vcf --config-file defaults.cfg \
    --a "--phased-vcf"  -c 2
```

The generated folders and current folder have all the data required to perform further selection analysis. Within each population folder four output files are generated. These contain Tajima's D, iHH, an updated VCF, and Fay and Wu's H statistic. The files are located in the results folder inside each population subfolder. $F_{ST}$ is calculated between each population and results are located in the fst folder. $F_{ST}$ results are calculated using the Weir and Cockerham estimator.

## 3.5 Data Visualisation

The purpose of the analysis pipeline is to generate standard signatures of selection from a VCF formatted input file. In order to assist with exploring/interpretting the results, visualization of the pipeline output can be extrenely useful. The next section describes some basic approaches to plotting these data using the R programming language. All of the following commands are run in an R session with the working directory set as the base directory from which the tutorial is being run. In each of the plots that follow, the vertical blue lines indicate the position of the lactase gene (LCT).

### 3.5.1 Visualizing $F_{ST}$ values

Plot the $F_{ST}$ results across the entire 10 megabase region for the tutorial. Mean $F_{ST}$ is being plotted in each case.

Plot mean Weir FST values.

```
weirFST = read.table('fst/2CEUYRI.weir.fst',header=T)
plot(weirFST[,6]~ weirFST[,2], pch=16, cex=.4,
type="p",ylab=expression(F[ST]),xlab="Chromosome position (bp)")
abline(h=mean(weirFST[,6]) + 3 * sd (weirFST[,6]),col='red')
rect(136545410,-1900,136594750,100,border="Blue")
```

Plot mean HapMap FST values.

```
hapmapFST = read.table('fst/2CEUYRI.hapmap.fst',header=T)
plot(hapmapFST[,6]~ hapmapFST[,2], pch=16, cex=.4,
type="p",ylab=expression(F[ST]),xlab="Chromosome position (bp)")
abline(h=mean(hapmapFST[,6]) + 3 * sd (hapmapFST[,6]),col='red')
rect(136545410,-1900,136594750,100,border="Blue")
```

Figures ( 1 and 2) show the weir and hapmap FST values for the tutorial region.

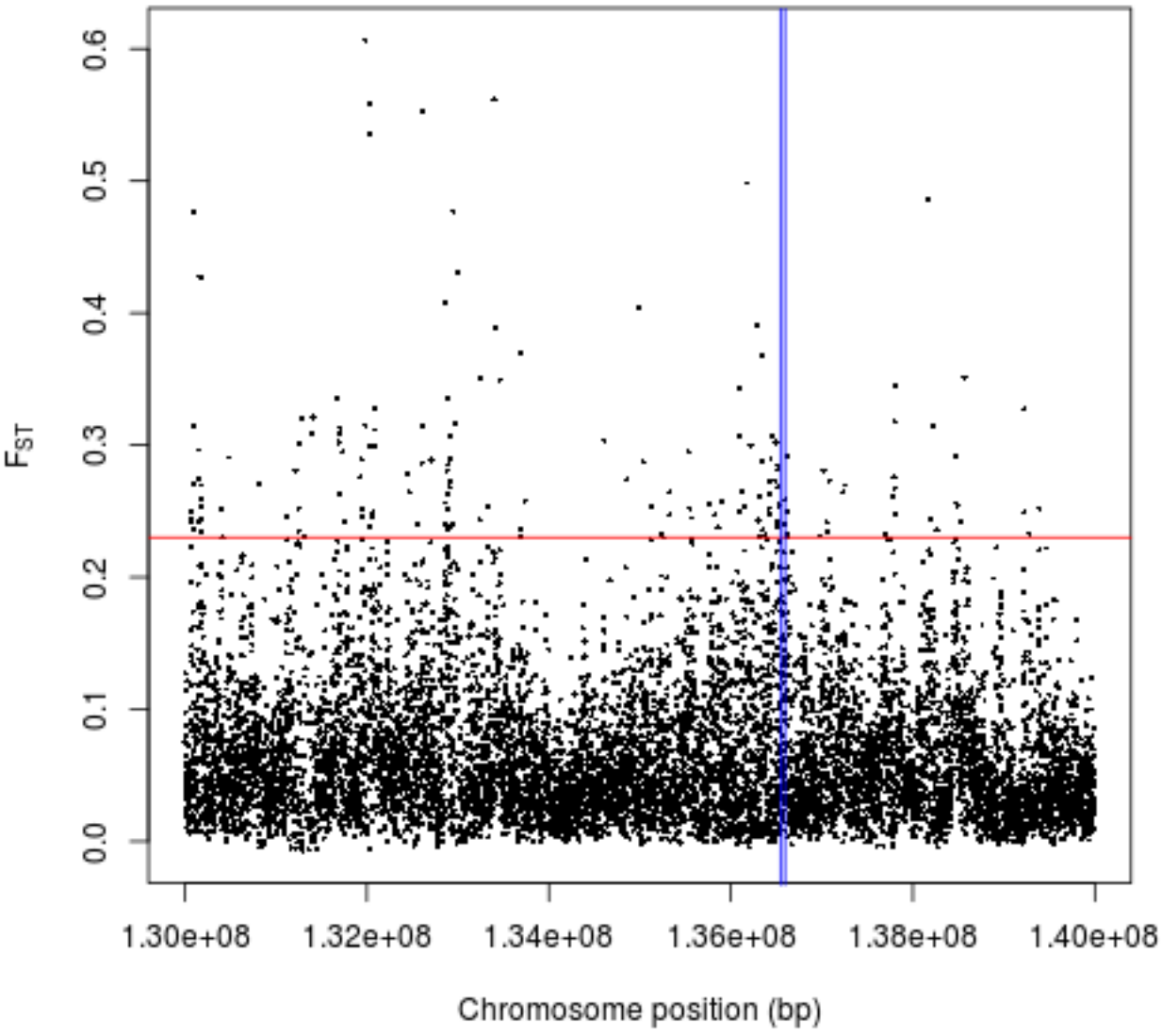


Figure 1: Values of Weir $F_{ST}$ mean FST between the CEU and YRI populations

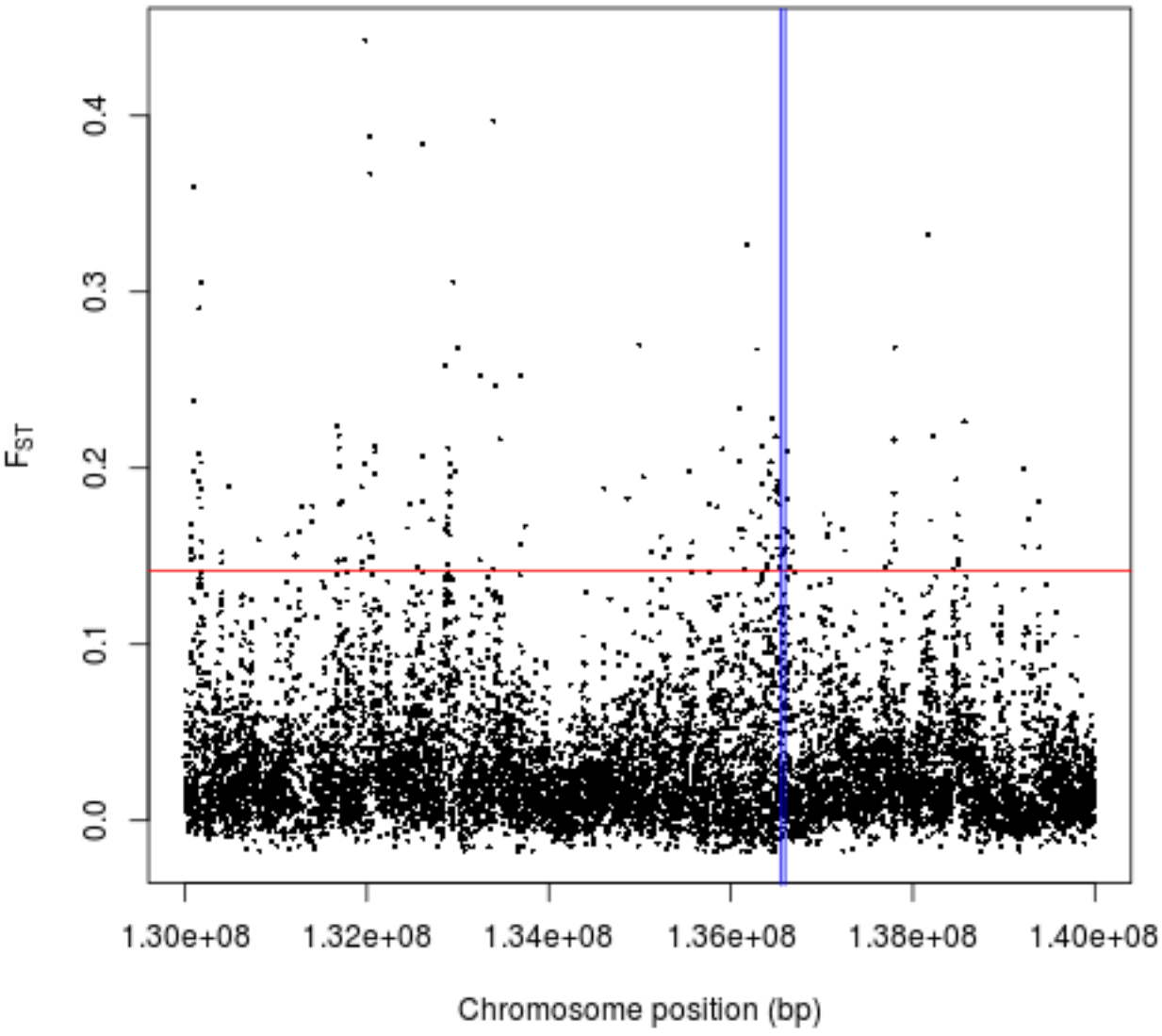


Figure 2: Values of HapMap $F_{ST}$ mean FST between the CEU and YRI populations

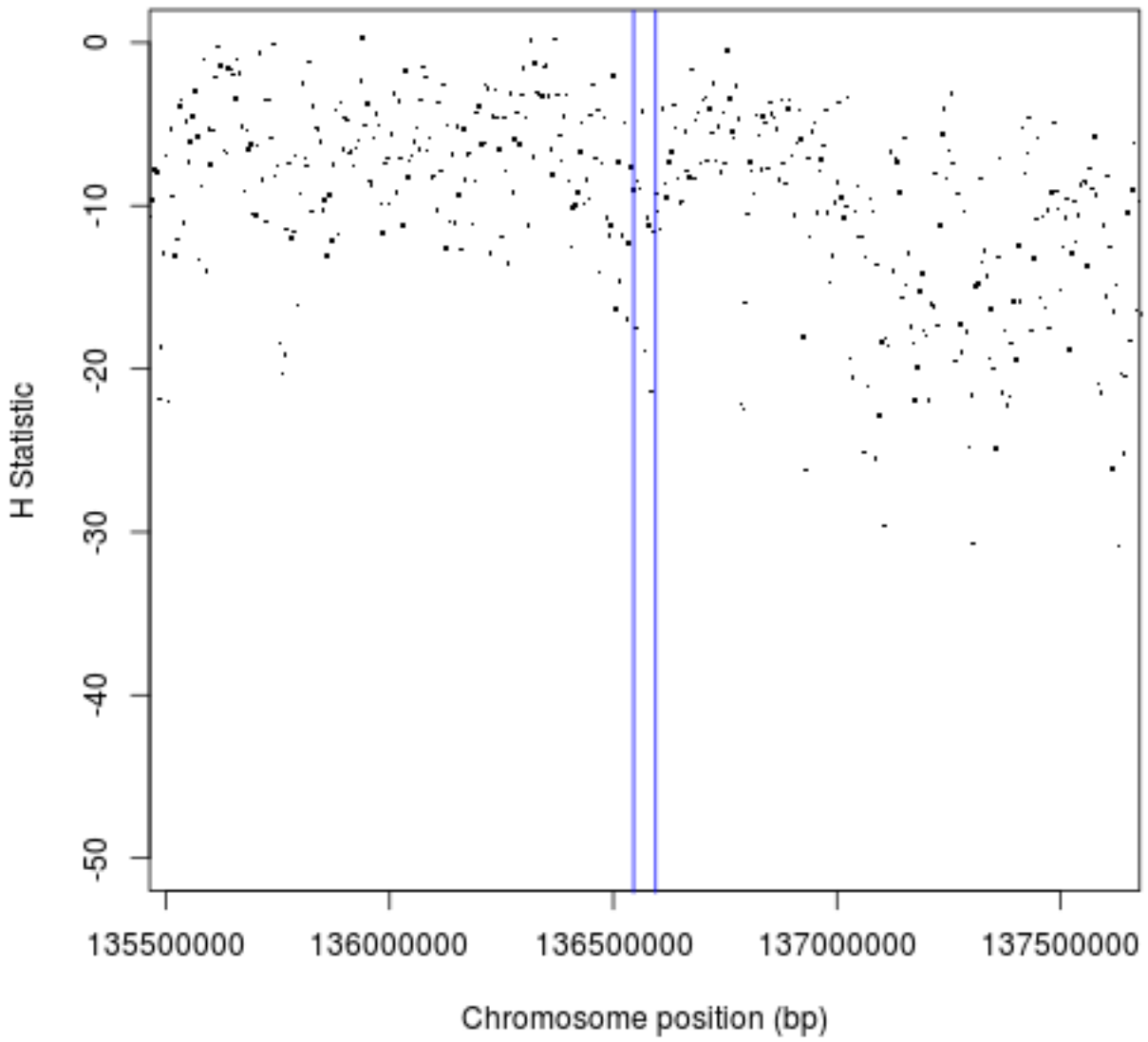


Figure 3: Fay and Wu's H statistic in the CEU population, across a two megabase region around the LCT gene.

### 3.5.2 Fay and Wu's H

Plot the Fay and Wu's H values for the CEU population:

```
CEUFay=read.table('CEU/results/CEU2.faw',comment.char="#")
#Plot Fay and Wu's H
plot(CEUFay[,15] ~ CEUFay[,1],xlim=c(136545410-1e6,136594750+1e6),
pch='.',cex=2,ylim=c(-50,0),
xlab='Chromosome position (bp)',ylab="H Statistic")
rect(136545410,-1000,136594750,100,border="Blue")
```

Figure 3 shows the Fay and Wu's H statistic one megabase downstream and upstream of the lactase gene. Plot the Fay and Wu's H values for the YRI population:

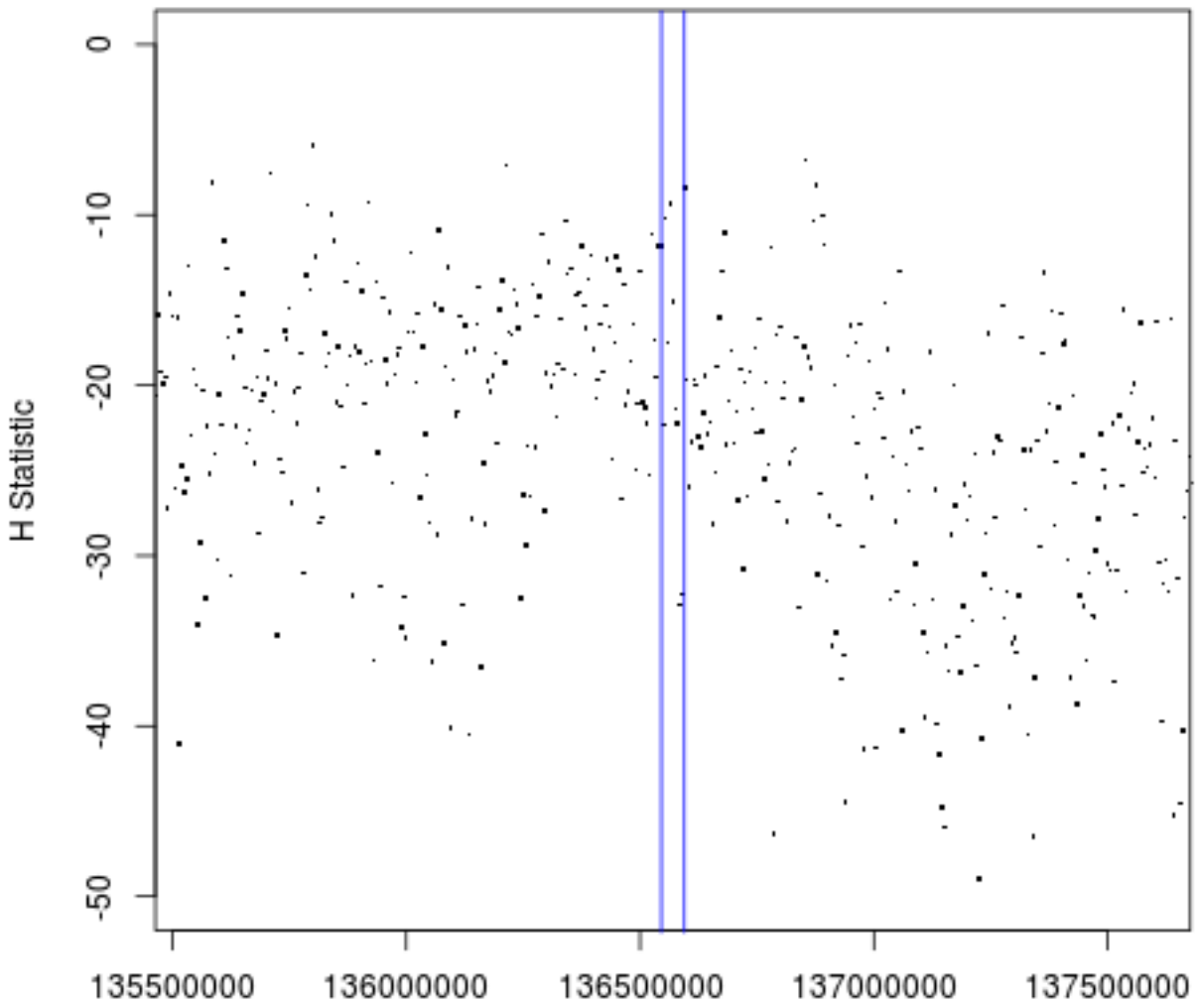


Figure 4: Fay and Wu's H statistic in the YRI population, across a two megabase region around the LCT gene.

```
YRIFay=read.table('YRI/results/YRI2.faw',comment.char="#")
#Plot Fay and Wu's H
plot(YRIFay[,15] ~ YRIFay[,1],xlim=c(136545410-1e6,136594750+1e6),
pch='.',cex=2,ylim=c(-50,0),
xlab='Chromosome position (bp)',ylab="H Statistic")
rect(136545410,-1900,136594750,100,border="Blue")
```

Figure 4 shows Fay and Wu's H statistic one megabase downstream and upstream of the lactase gene.

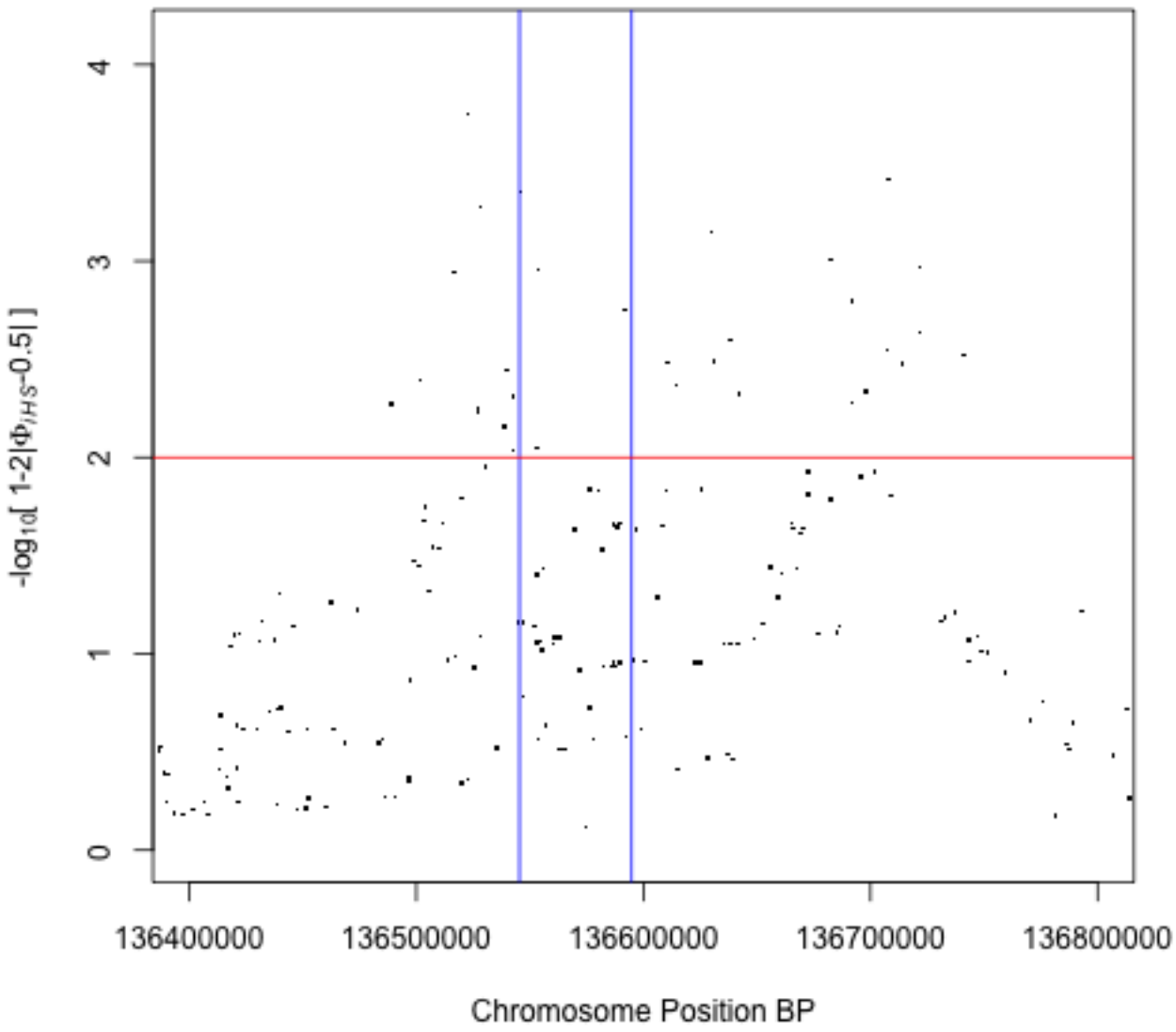


Figure 5: iHS statistic in the CEU population, across a 400 kilobase region around the LCT gene.

### 3.5.3 iHS

Plot the iHS values around the lactase gene for the CEU population:

```
CEUihs = read.table('CEU/results/CEUchr2.ihs')
#plot IHS pvalues
plot(CEUihs[,4] ~ CEUihs[,2],xlim=c(1.364e8,1.368e8),pch='.',cex=2,
ylab=expression("-" * log[10] * "[" ~ "1-2|" * Phi[scriptstyle(italic(iHS))] * "-0.5|" ~ "]"),
xlab="Chromosome Position BP")
rect(136545410,-10,136594750,10,border="Blue")
abline(h=2,col="red")
```

Figure 5 shows iHS pvalues around the lactase gene in the CEU population.

Plot the iHS values around the lactase gene for the YRI population:

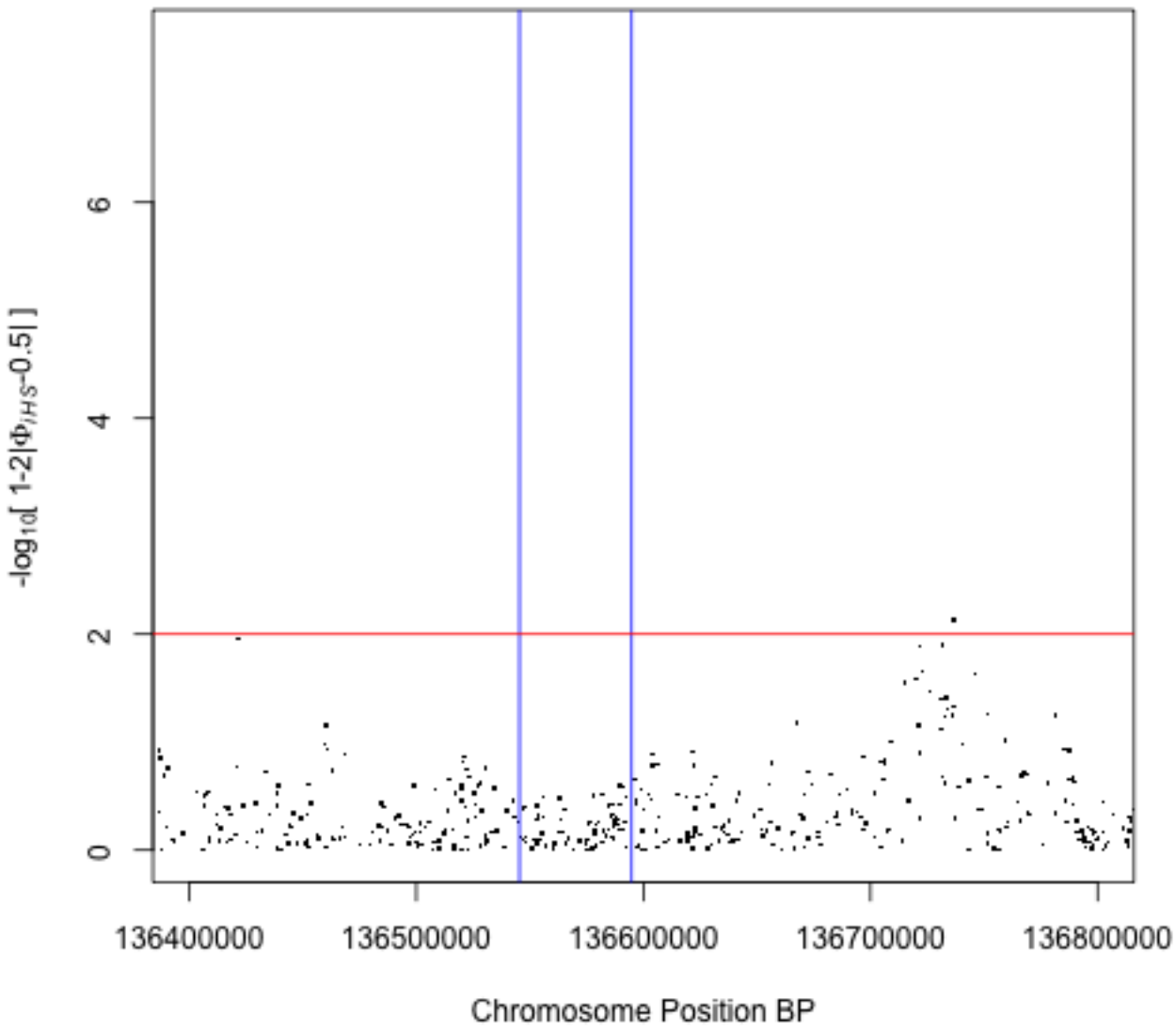


Figure 6: iHS statistic in the YRI population, across a 400 kilobase region around the LCT gene.

```
YRIihs = read.table('YRI/results/YRIchr2.ihs')
#plot IHS pvalues
plot(YRIihs[,4] ~ YRIihs[,2],xlim=c(1.364e8,1.368e8),pch='.',cex=2,
ylab=expression("-" * log[10] * "[" ~ "1-2|" * Phi[scriptstyle(italic(iHS))] * "-0.5|" ~ "]"),
xlab="Chromosome Position BP")
rect(136545410,-10,136594750,10,border="Blue")
abline(h=2,col="red")
```

Figure 6 shows iHS pvalues around the lactase.

To visualize individual SNPs, a haplotype bifurication diagram can be used (Gautier and Vitalis, 2012). The SNP rs28453840 displayed a strong signal of selection using iHS in the CEU population. The following commands construct a bifurcation diagram for both alleles at this SNP using rehh.

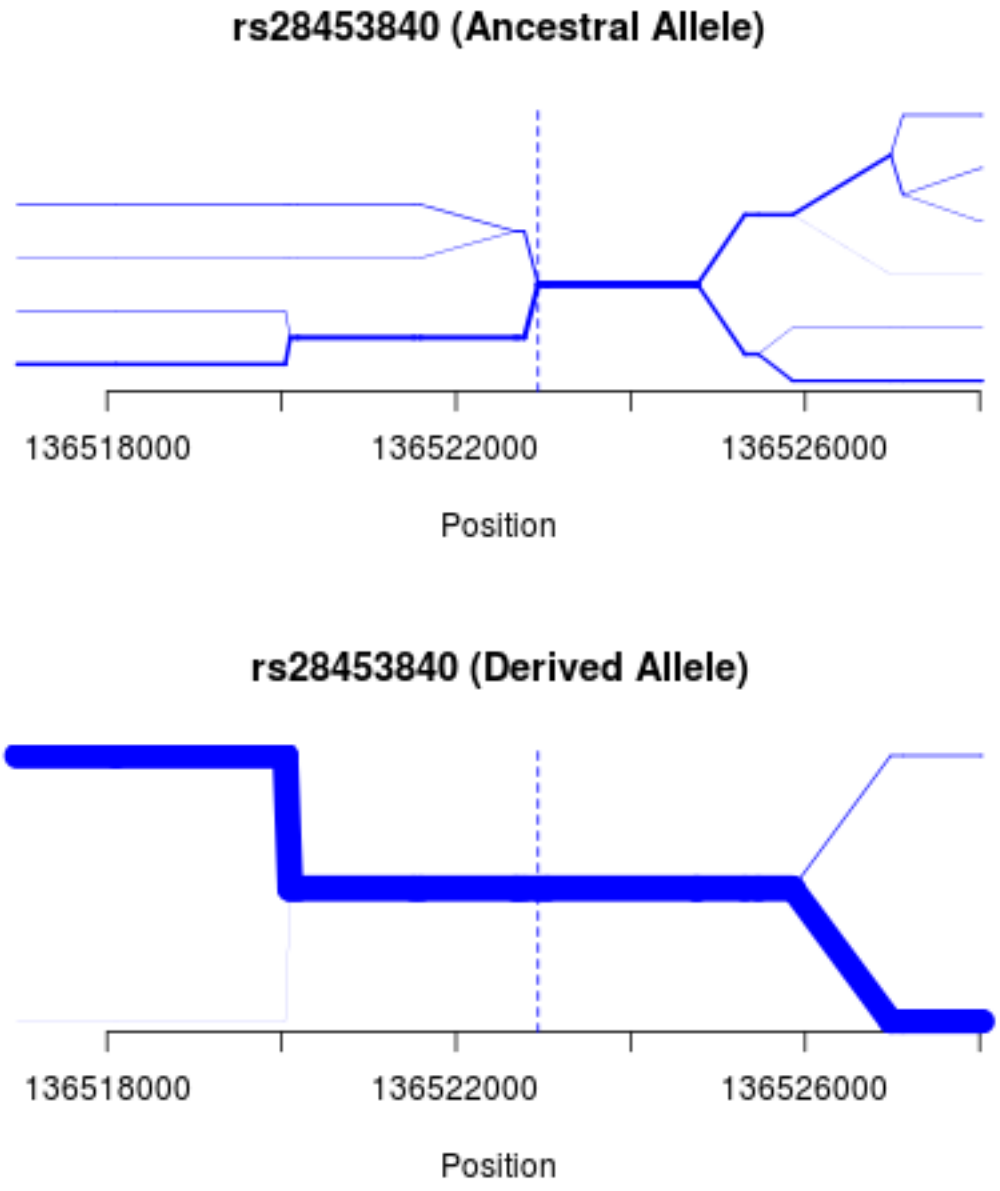


Figure 7: Bifurcation Diagram for rs28453840

```
# Plot  bifurcation diagram.
# Import rehh
library(rehh)
load('CEU/results/CEUchr2.RData')
par(mfrow=c(2,1))
bifurcation.diagram(haplohh=d,mrk_foc=13750,all_foc=1)
bifurcation.diagram(haplohh=d,mrk_foc=13750,all_foc=2)
```

Figure 7 shows the bifurcation plot for rs28453840 a snp that had a iHS score greater that 3 at position 136522941 on chromosome 2 just upstream of lactase.

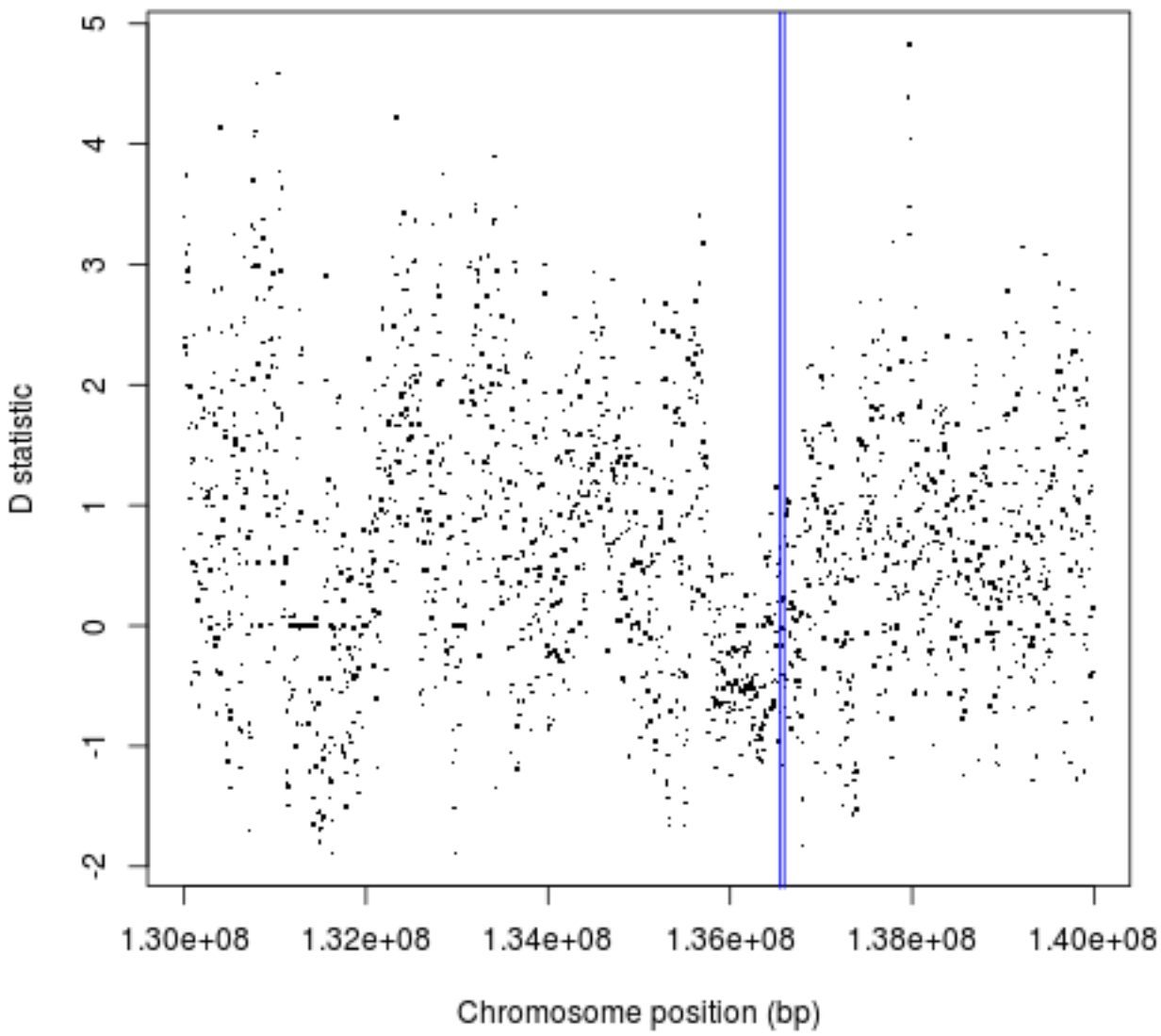


Figure 8: Tajima's D statistic in the CEU population, across the full 10 megabase pipeline region.

### 3.5.4 Tajima's D

```
tajimaD=read.table(file="CEU/results/CEU2.taj_d", header=TRUE)
plot(tajimaD[,4] ~ tajimaD[,2],pch='.',cex=2,xlab="Chromosome position (bp)", ylab="D statistic")
rect(136545410,-10,136594750,10,border="Blue")
```

Figure 8 show the Tajima's D statistic across the full tutorial region.

```
tajimaD=read.table(file="YRI/results/YRI2.taj_d", header=TRUE)
plot(tajimaD[,4] ~ tajimaD[,2],pch='.',cex=2,xlab="Chromosome position (bp)", ylab="D statistic")
rect(136545410,-10,136594750,10,border="Blue")
```

Figure 9 show the Tajima's D statistic across the full tutorial region.

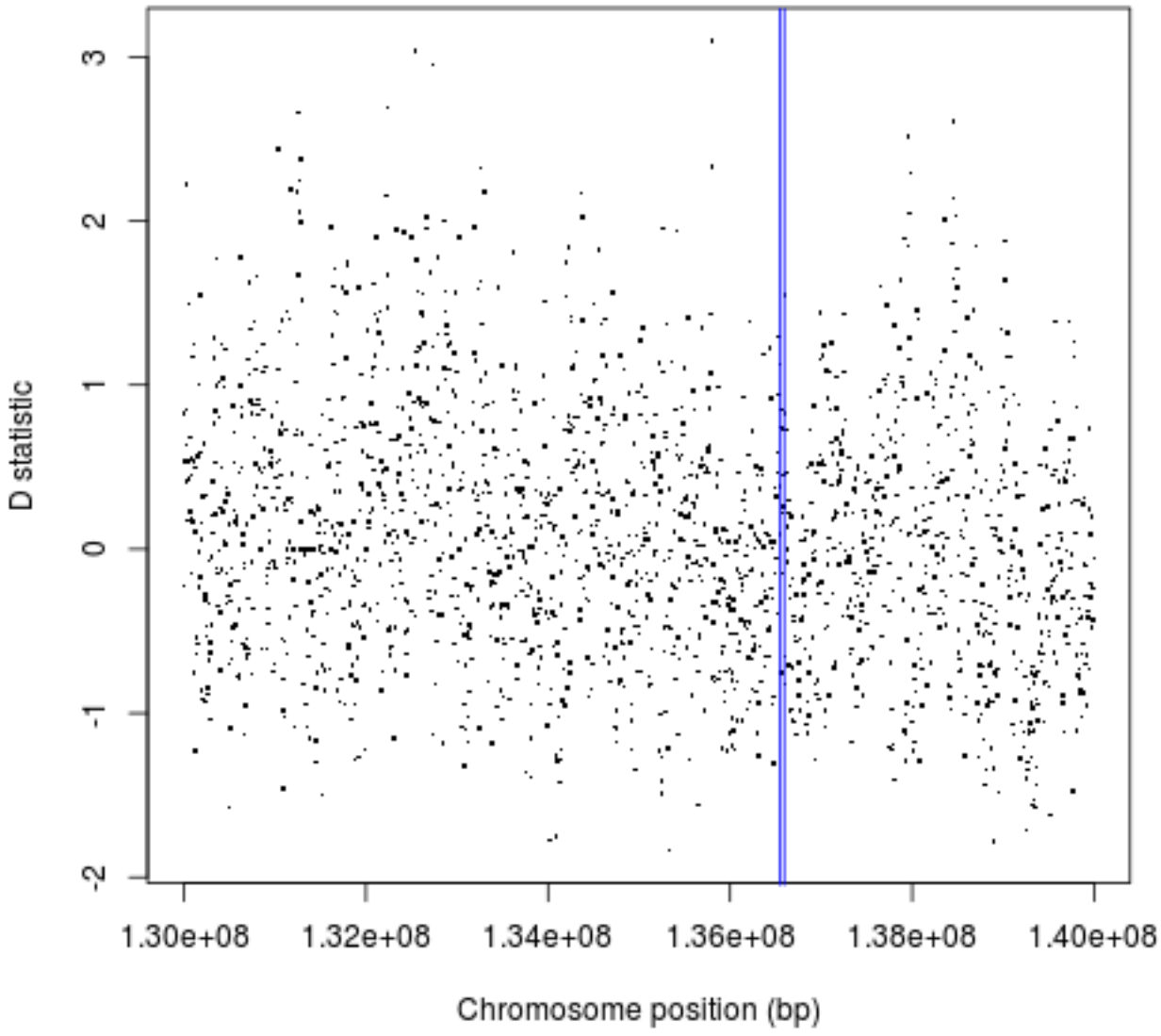


Figure 9: Tajima's D statistic in the YRI population for the tutorial region

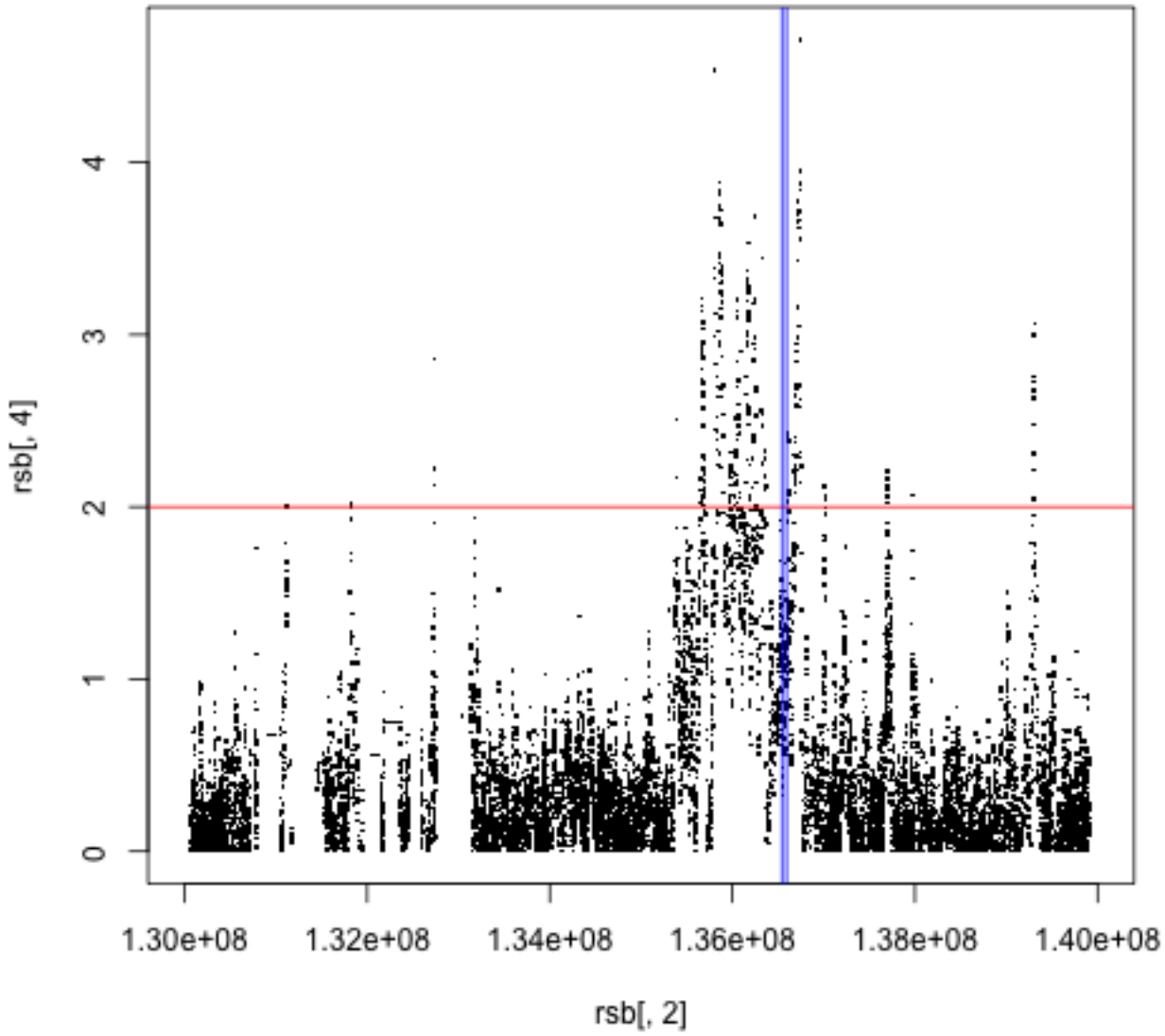


Figure 10: RSB statistic between the CEU and YRI population for the tutorial region

### 3.5.5 Rsb

Rsb makes the differeneces between the two populations very clear. The following code shows how to plot the Rsb bilateral P-values for the 10 megabase tutorial region.

```
rsb = read.table('rsb/2CEUYRI.rsb',header=T)
plot(rsb[,4] ~ rsb[,2],pch='.',cex=2)
abline(h=2,col='red')
# Plotting the lactase gene.
rect(136545410,-10,136594750,10,border="Blue")
```

Figure 10 shows the Rsb statistic for the entire region for the tutorial, it is visually striking the difference at the LCT locus.

# 4 Output Files

The output files are preserved in the same state as the original output from the program used to generate the data.

## 4.1 multipop_selection_pipeline

### 4.1.1 Fst

Located in the fst folder. Tab-delimited data file containing 1 header line followed data on each subsequent line

```
CHROM   BIN_START       BIN_END N_VARIANTS      WEIGHTED_FST    MEAN_FST
2       130000001       130010000       94      0.133102        0.0680276
```

## 4.2 selection_pipeline

All the outputs for each population are contained in the results folder. If you ran the tool using *multipop_selection_pipeline* the outputs are located in <pop name>/results.

### 4.2.1 Fay and Wu's H

Space-delimited data file containing header line that start with a hash character (#). Contains lots of columns. If you are only interested in Fay and Wu's H, column 1 provides the position and column 15 provides the H statistic.

```
# RefStart   Refend ... FayWu_H
130000040 130005039 ... -22.2438460
```

### 4.2.2 iHS

Space-delimited data file containing one header line followed by data on each subsequent line.

```
"CHR" "POSITION" "iHS" "Pvalue"
"rs4662641" 2 130000272 0.0644902912148128 0.0229261107107533
```

### 4.2.3 iHH

Space-delimited data file containing one header line followed by data on each subsequent line.

```
"CHR" "POSITION" "FREQ_a" "IHHa" "IHHd" "IES"
"rs1251176" 2 130000040 0.9823 11558.89 83915.49 11571.13
```

### 4.2.4 Tajima's D

Space-delimited data file containing one header line followed by data on each subsequent line.

```
CHROM   BIN_START       N_SNPS  TajimaD
2       130000000       22      0.775224
```

# 5 Command line Arguments

The selection pipeline contains three programs: *selection_ pipeline*, *aa_ annotate* and *multipop_ selection_ pipeline*. The selection pipeline does all the intra-population statistics calculations. The multipop_selection_pipeline program calculates all the inter-population statistics and calls the selection pipeline. The aa_annotate program annotates a haplotype file or a phased vcf file with the ancestral allele from the 6-way EPO alignment, for other species or alternative ancestral annotation, the feauture will be added in the future.

## 5.1 Multipopulation

### 5.1.1 Input Files

- -i <vcf input file> VCF file containing all the populations you want to analyse from one chromosome or a part of a chromosome only.

### 5.1.2 Output Files

- FST

  Fst results are stored in the fst folder with the chromosome number followed by the two populations. e.g 2CEUYRI.fst

- Selection Pipeline Results

  All single population pipeline results are stored in the subdirectory of the population in a folder named results. These contain the iHH, Tajima's D and a population VCF file.

### 5.1.3 Other parameters (Compulsory)

- -c <Chromosome>

  Integer for the chromosome being used.

- –config-file <path to config file>

  Path to the selection pipeline config file an example config file is located in the base directory of the extracted package.

### 5.1.4 Other parameters (Optional)

- -l <log_file>

  Name for the log file. Moved into the logs folder at the end of program run.

- –no-clean-up

- -a <Arguments to the selection pipeline>

  Quoted string containing any extra arguments to the selection_pipeline program. e.g "–imputation" Do not clean up intermediate data files.

- –fst-window-size <FST window size>

  Fst calculation sliding window size in kilobases (default = 1kb).

- –fst-window-step <FST window step>

  Fst calculation windoow jump in kilobases, if window size is equal to the jump size non-overlapping windows are used (default = 1kb).

- –cores

  Number of cores avaliable for the pipeline overrides setting in the config file.

## 5.2 Selection Pipeline

### 5.2.1 Input Files

- -i <VCF input file>

  Single population single chromosome VCF input file. VCf should be unzipped

### 5.2.2 Output Files

The Results directory contains all the output files.

- .ihh file

  The outputted iHH data for each SNP

- .taj_d file

  Tajima's D output

- .vcf file

  Single population VCF updated by the pipeline, can contain.

### 5.2.3 Other parameters(Compulsory)

- –config-file <Config File path>

  Path to the selection pipeline config file an example config file is located in the base directory of the extracted package.

- -c <Chromosome>

  The chromosome that the VCF file represents.

- –population <Population Code>

  Code to represent the population of interest e.g CEU,YRI. Used for the results file prefixes.

### 5.2.4 Other parameters(Optional)

- -l <log_file>

  Name for the log file. Moved into the logs folder at the end of program run.

- –maf <minimum MAF>

  Minor allele frequency filter threshold any SNPs below this threshold will be discarded from the analysis (default = 0.01).

- –hwe <hardy-weinberg minimum p-value>

  A hardy weinberg test is performed on every snp any snps failing the test will be discarded (default = 0.001).

- –remove-missing <Inclusion threshold for missing genotypes>

  Inclusion criteria for SNPs with missing data. SNPs with less than this value will be removed from analysis (default = 0.99).

- –fay-Window-Width <window width>

  Sliding window width for Fay and Wu's H calculation in kb (default = 5kb).

- –fay-Window-Jump <Window Jump>

  Window jump for Fay and Wu's H calculation, if equal to fay-Window-Width non-overlapping windows will be used (default = 5kb).

- –TajimaD <tajimas D bin size>

  Tajima's D statistic bin size in kb (default = 5kb).

- –no-clean-up

  Do not clean up intermediate data files

- –ehh-window-size

  Window size for multicore rehh calculations in megabases (Mb) (default = 5Mb).

- –ehh-overlap

  Window overlap for multicore rehh calculations in megabases (Mb) (defaut = 2Mb).

- –imputation

  Perform imputation using impute2 on the genotypes produced by shapeit. Option only works on a unphased VCF file.

- –beagle

  Perform phasing using beagle. Option removes shapeit from the analysis and subsequent imputation step.

- –no-genetic-map

  Does not use a genetic map for the analysis when using REHH.

- –daf <Minimum derived allele frequency>

  Derived allele frequencies below this minimum will be discarded (default = 0.0).

- –big-gap

  Gap size in kb for not calculating iHH if the gap is too large. If set to zero the big-gap rule is not applied (default = 0kb).

- –small-gap

  Gap size in kb for applying a penalty to the area calculated by iHH. If set to zero the small-gap rule is not applied (default = 0kb).

- –small-gap-penalty

  Penalty multiplier for intergration step in iHH calculation. $multiplier/gap\_size * area$ is the formula we use. Setting the multiplier to the same value as the small gap threshold is recommended (default = 0kb).

- –cores

  Number of cores avaliable for the pipeline overrides setting in the config file.

## 5.3 Ancestral Annotation

The progam *ancestral_ annotation* is installed in the program path. The program annotates .haps and .vcf files with ancestral allele annotation from the 6-way IPO alignment or the human reference genome.

### 5.3.1 Input Files

- -i or –haps <HAPS File>

  Haplotype File (.haps)

- -v <Phased VCF file>

  Phased VCF file (.vcf), phased VCF genotypes denoted by a bar ( | ) for each sample.

- -a or -aa <Ancestral allele fasta>

  Ancestral allele annotation file. Currently only works on a the full 1000 Genomes Project GRCh37ḋ4 reference file or the single chromosome fasta files from the 6-way EPO alignment.

### 5.3.2 Output Files

- -o or –output <Output file name>

  Output file name optional argument by default output is sent to the stdout stream.

- -s or –sample-file <Sample file output>

  Sample file output name ( currently only works with phased vcf option)

### 5.3.3 Other parameters

- –header-regex

  Compulsory argument: it is a regex (regular expression) with a question mark denoting substitution for the chromosome number. The regex should match the header in the fasta file and when the question mark is replaced selectively return only the chromosome of interest. Optional for single chromosome fasta files.

- –single-chromosome

  Single chromosome pipeline run option.

- -c <chromosome number>

  The number or symbol of the chromosome being used.

- -f or –format <format>

  The 6-way EPO alignment denotes ancestral alleles with both high and low confidence. To use only ancestral alleles with high confidence use –format upper. To use both high and low confident alleles use –format lower. By default the program will use only highly confident alleles. The highly confident alleles are in uppercase.

  By default it uses all the valid bases in the file that are one of either A, T, C, G, a, t, c or g

## 5.4 Configuration File

The selection pipeline requires a configuration file. By default the program looks in the current working directory for a file named defaults.cfg but you can point the program to any file using command line argument –config-file <config_file_location>. There are two main programs in the selection pipeline namely *selection_pipeline* and *multipop_selection_pipeline*. These programs share a config file but certain configuration parameters can be ommitted when using the *selection_pipeline* program exclusively. A clean install of the pipeline generates an example configuration file containing default arguments for all the compulsory parameters. The default config file contains an example of the format.

### 5.4.1 system

- cores_avaliable

  Certain programs in the pipeline can take advantage of multicore computers. This option instructs the pipeline about the maximum number of concurrent processes it is allowed to use.

### 5.4.2 environment

- LD_LIBRARY_PATH

  Set the library path when running the pipeline, this enables the pipeline to use the shared libraries that are used for some programs in the pipeline (alter this option with caution!)

- PERL5LIB

  Sets the PERL5LIB environment variable, this enables the pipeline to use the perl libraries required by VCFTOOLS (alter this option with caution!)

### 5.4.3 selection_pipeline

- selection_pipeline_executable

  Points to the location of the selection_pipeline_executable.

### 5.4.4 vcf_tools

- vcf_tools_executable

  Points to the vcftools executable, by default it points to the vcftools executable installed with the pipeline.

- vcf_subset_executable

  Points to the vcf-subset executable, by default pointing to the vcf-subset installed with the pipeline.

- vcf_merge_executable

  Points to the vcf-merge executable, by default pointing to the vcf-subset installed with the pipeline.

- extra_args

  A quoted string containing extra arguments to send to the vcf_tools executable.

#### 5.4.5 shapeit

- shapeit_executable

  Location of the shapeit executable.

- genetic_map_dir

  Directory containing the genetic map for shapeit.

- genetic_map_prefix

  The full file for the genetic map files with a "?" character representing the changing chromosome number.

- extra_args extra arguments to send to shapeit (Warning: Certain options could potentially break the pipeline - use with caution).

#### 5.4.6 impute2

- impute_executable

  Location of the impute2 executable

- impute_map_dir

  Directory containing the genetic map for impute2

- impute_reference_dir

  Directory containing the reference panel ( .legend and .hap) files for impute2.

- chromosome_split_size

  Window size for imputation calculation.

- impute_map_prefix

  The full file name for the genetic map files with a "?" character representing the changing chromosome number

- impute_reference_prefix

  The full file name for the reference panels minus the extension with a "?" character representing the changing chromosome number.

- extra_args

  extra arguments to send to impute2 (Warning: Certain options could potentially break to pipeline use with caution).

#### 5.4.7 plink

- plink_executable

  Location of the plink executable

#### 5.4.8 Rscript

- rscript_executable

  Location of the rscript executable (Program usually in path so just Rscript is the default).

- indel_filter

  Location of the rscript indel_filter (hap_indel_and_maf_filter.R)i

#### 5.4.9 haps_scripts

- haps_to_hapmap_script

  Location of the haps_to_hapmap script

- haps_filter_script

  Location of the haps_filter script

#### 5.4.10 ancestral_allele

- split_by_chromosome

  Option determines whether the ancestral fasta file is split by chromosome or not. If the ancestral fasta is split by chromosome the ancestral_fasta_dir and ancestral_prefix arguments are required. If the ancestral fasta is just one flat file the ancestral_fasta_file options is required which is just merely the location of the ancestral fasta option.

- ancestral_fasta_header_regex

  This options specifies a regular expression that will extract each chromosome from the fasta file. The '?' denotes the chromosome number and will be replaced in the regular expression for the chromosome number of interest. Even when using one fasta per chromosome this is required as the python library used to extract the sequence uses the fasta header for extraction. Only required if your fasta file is not split by chromosome.

- ancestral_fasta_file

  Ancestral fasta file used if the split_by_chromosome is set to false.

- ancestral_allele_script

  Location of the ancestral_annotation script (ancestral_annotation)

- ancestral_fasta_dir

  Directory containing the ancestral reference files

- ancestral_prefix

  Full file name for ancestral fasta files containing a "?" character

#### 5.4.11 qctool

- qctool_executable

  Location of the qctool executable.

#### 5.4.12 multicore_ihh

- multicore_ihh

  Location of the multicore_iHH.R script

# 6 Log Files

The log files contain all the information for a pipeline run including any errors if for some reason some part of the pipeline does not complete.

## 6.1 multipop_selection_pipeline

The location of the log file for *multipop_selection_pipeline* defaults is located in the log directory. It contains all the logging information for the between population selection signature calculations.

## 6.2 selection_pipeline

The location of the log file for *selection_pipeline* is located in the log directory. The log file contains all the logging information for the within population selection signature calculations.

# 7 F.A.Q

1. How do I run *multipop_selection_pipeline* with a phased VCF?

   In the -a argument for multipop_selection_pipeline merely add –phased-vcf between the quotes. This will ensure phasing and imputation will be skipped when *selection_pipeline* is called.

2. My populations are in seperate VCF-Files: how do I run *multipop_selection_pipeline*?

   To run the pipeline you will need to merge the VCF-files into one large multipopulation VCF file and generate the appropriate population files.

   To merge your vcfs you can use the vcf-merge program. For this to work correctly outside the selection pipeline you will need to add the following to your .bashrc file or equivalent dot-file for another shell.

```
export PERL5LIB=${PERL5LIB}:<path to selection pipeline>/lib/perl5
```

   The command to run vcf-merge is as follows.

```
vcf-merge <vcf1.vcf> <vcf2.vcf> ..... > big_vcf.vcf
```

3. My VCF file is not split by chromosome how do I get my VCF into a single chromosome?

   The vcftools program can be used to extract each chromosome from your full vcf file. If you do not have the vcftools program installed the bin/ directory contains exactly what you need. For example, for human 1000 genomes data to extract chromosome 2 from your VCF file use the following command.

```
vcf-tools --vcf big_vcf.vcf --chr 2 --out chr2 --recode
```

   The command will generate a vcf file named chr2.recode.vcf containing only data from chromosome 2.

4. How can I run the *selection pipeline* or *multipop_selection_pipeline* genome wide?

   A simple bash loop should do the trick, although here I am assuming you have multiple population VCF all in one directory, the file name before the chromosome number is prefixed by the string c̈hränd that the chromosomes include 1 through 22, X and Y.

   The following commands run the selection_pipeline on a 4 core machine with phasing and imputation on the CEU and YRI populations.

```
for chromosome in {1..22} X and Y
do
    mkdir -p $chromosome
    (cd ${chromosome}; multipop_selection_pipeline -p CEU_ids.txt -p YRI_ids.txt \
      -i chr${chromosome}.vcf --config-file defaults.cfg \
      -a "--imputation" -c ${chromosome} --cores 4)
done
```

5. Rehh alterations

   The rehh package source included with the pipeline has been altered to match the output filters used in Voight's paper. If the EHH $>$ 0.05 reaches the end of a chromosome or the start of a gap $>$ big_gap , then no value is returded for the core snp. The small_gap specifies the gap distance to reduce the distance spanned by the gap by a multiplicative factor specified by small_gap_multiplier. The formula for the penalty is $\frac{small\_gap\_multiplier}{gap\_size}$. To match the parameters used by Voight, a value of 200 should be used for big_gap_threshold, 20 for small_gap_threshold and 20 for small_gap_multiplier. All gap distances are in kb. (Voight *et al.*, 2006).

# References


Bersaglieri T, Sabeti PC, Patterson N, Vanderploeg T, Schaffner SF, Drake JA, Rhodes M, Reich DE, Hirschhorn JN (2004). "Genetic signatures of strong recent positive selection at the lactase gene." *Am. J. Hum. Genet.*, **74**(6), 1111–1120.

Danecek P, Auton A, Abecasis G, Albers CA, Banks E, DePristo MA, Handsaker RE, Lunter G, Marth GT, Sherry ST, McVean G, Durbin R, 1000 Genomes Project Analysis Group (2011). "The variant call format and VCFtools." *Bioinformatics (Oxford, England)*, **27**(15), 2156–2158.

Delaneau O, Zagury JF, Marchini J (2013). "Improved whole-chromosome phasing for disease and population genetic studies." *Nature Methods*, **10**(1), 5–6.

Fay JC, Wu CI (2000). "Hitchhiking under positive Darwinian selection." *Genetics*, **155**(3), 1405–1413.

Flicek P, Amode MR, Barrell D, Beal K, Brent S, Carvalho-Silva D, Clapham P, Coates G, Fairley S, Fitzgerald S, Gil L, Gordon L, Hendrix M, Hourlier T, Johnson N, Kähäri AK, Keefe D, Keenan S, Kinsella R, Komorowska M, Koscielny G, Kulesha E, Larsson P, Longden I, McLaren W, Muffato M, Overduin B, Pignatelli M, Pritchard B, Riat HS, Ritchie GRS, Ruffier M, Schuster M, Sobral D, Tang YA, Taylor K, Trevanion S, Vandrovcova J, White S, Wilson M, Wilder SP, Aken BL, Birney E, Cunningham F, Dunham I, Durbin R, Fernández-Suárez XM, Harrow J, Herrero J, Hubbard TJP, Parker A, Proctor G, Spudich G, Vogel J, Yates A, Zadissa A, Searle SMJ (2012). "Ensembl 2012." *Nucleic acids Research*.

Gautier M, Vitalis R (2012). "rehh: an R package to detect footprints of selection in genome-wide SNP data from haplotype structure." *Bioinformatics (Oxford, England)*, **28**(8), 1176–1177.

Howie BN, Donnelly P, Marchini J (2009). "A flexible and accurate genotype imputation method for the next generation of genome-wide association studies." *PLoS Genet.*, **5**(6), e1000529.

Nievergelt CM, Smith DW, Kohlenberg JB, Schork NJ (2004). "Large-scale integration of human genetic and physical maps." *Genome Research*, **14**(6), 1199–1205.

Sabeti PC, Schaffner SF, Fry B, Lohmueller J, Varilly P, Shamovsky O, Palma A, Mikkelsen TS, Altshuler D, Lander ES (2006). "Positive natural selection in the human lineage." *Science (New York, NY)*, **312**(5780), 1614–1620.

Tajima F (1989). "Statistical method for testing the neutral mutation hypothesis by DNA polymorphism." *Genetics*, **123**(3), 585–595.

Vilella AJ, 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.

Voight BF, Kudaravalli S, Wen X, Pritchard JK (2006). "A map of recent positive selection in the human genome." *PLoS biology*, **4**(3), e72.

Weir BS, Cockerham CC (1984). "Estimating F-statistics for the analysis of population structure." *evolution*.