Vignette for the package *rehh* (version 2+)

Mathieu Gautier, Alexander Klassmann and Renaud Vitalis

24/10/2016

Contents

This vignette is aimed at presenting additional information on the R package *rehh* by describing how to use it to perform whole genome scan for footprints of selection using statistics related to the Extended Haplotype Homozygosity (EHH) (Sabeti *et al.* 2002). Importantly, the current implementation of tests assumes markers are bi-allelic.

The *rehh* package is currently available for most platforms (Linux, MS Windows and MacOSX) from the CRAN repository (http://cran.r-project.org/) and may be installed using standard procedure. Once the package has been successfully installed on your system, it can be loaded using the following command:

```
library(rehh)
```

1 Input Files

The package *rehh* basically requires as input:

1. haplotype data file(s) (in three possible format) for each population of interest (see 1.1)
2. a SNP information file (see 1.2)

Important Note: For a given chromosome, SNPs are assumed to be ordered in the same way in both the haplotype file (columns) and the SNP information file.

For illustration purposes, example files that originate from a previously published study on the Creole cattle breed from Guadeloupe (CGU) (Gautier and Naves 2011) are provided in the package and can be copied in the working directory with the following command:

```
make.example.files()
```

In the following examples, the command `make.example.file()` is assumed to have been run (see above) so that example files are in the working directory.

1.1 Haplotype data file

Three haplotype input file formats are supported:

1. a standard haplotype format. Each line represents a haplotype (the first element being an haplotype identifier) with SNP genotype in columns as in the example file `bta12_cgu.hap` created when running the function `make.example.files()` and that consists of 280 haplotypes (identifier 1 to 280) of 1424 SNPs each. See 1.3.1 for a detailed example.
2. A (transposed) format with haplotype in columns and SNPs in row as in the example file `bta12_cgu.thap` created when running the function `make.example.files()`. Note that this format is similar to the one produced by the phasing program *SHAPEIT2* (O'Connell *et al.* 2014). See 1.3.2 for a detailed example.
3. the output file format from the phasing program *fastPHASE* (Scheet and Stephens 2006) as in the `bta12_hapguess_switch.out` example file created when running the function `make.example.files()`. Note that haplotypes might originate from several different populations (i.e., if the -u *fastPHASE* option was used). See 1.3.3 for a detailed example.

By default alleles are assumed to be coded as 0 (missing data), 1 (ancestral allele) or 2 (derived allele). Recoding of the alleles in this format, according to the SNP information data file (see 1.2) can be performed with the `recode.allele` option of the function `data2haplohh()` (see 1.3).

1.2 SNP information data file

This data file should contain SNP information as in the `map.inp` example file created when running the function `make.example.files()`. Each line contains five columns corresponding to:

1. the SNP name
2. the SNP chromosome (or scaffold) of origin
3. the SNP position on the chromosome (or scaffold). Note that it is up to the user to choose either physical or genetic map positions (if available).
4. the SNP ancestral allele (as coded in the haplotype input file)
5. the SNP derived alleles (as coded in the haplotype input file)

The fourth and fifth columns (allele coding) should be filled in but the corresponding information is only used when activating the `recode.allele` option of the function `data2haplohh()` (see 1.3). In that case, for each SNP, the allele specified in the fourth (respectively fifth) column will be recoded as 1 (respectively 2), any other allele name will be recoded as 0 (i.e., missing data). More importantly, it should be noticed that the ancestral or derived allele information associated to this coding are only relevant for within population tests (based on *iHS*). In other words, if one is only interested in across-population tests (based on *Rsb* or *xpEHH*), assignment of the two SNP alleles in the fourth and fifth column may be performed randomly.

As an illustration, the following R command displays the first five row of the `map.inp` example file created when running the function `make.example.files()`:

```r
head(read.table("map.inp"))
```

```
>         V1 V2     V3 V4 V5
> 1 F0100190  1 113642  T  A
> 2 F0100220  1 244699  C  G
> 3 F0100250  1 369419  G  C
> 4 F0100270  1 447278  A  T
> 5 F0100280  1 487654  T  A
> 6 F0100290  1 524507  C  G
```

1.3 Loading data files

The `data2haplohh()` function allows to convert data file into an R object of class `haplohh` subsequently used by the functions of the *rehh* package. The following main options are available to recode alleles or select SNPs (based on Minor Allele Frequency or percentage of missing data) and haplotypes (based on percentage of missing data):

1. **Allele recoding option**: This option is activated with `recode.allele=TRUE` and allows to recode haplotype according to the ancestral and derived allele definition available in the SNP information file (fourth and fifth columns) as: 0 (missing data), 1 (ancestral allele) or 2 (derived allele).
2. **Discard haplotypes with a high amount of missing data**. If haplotypes contain missing data (which is generally not the case since most phasing programs allow imputing missing genotypes), it is possible to discard those with less `min_perc_geno.hap` % of the SNPs genotyped. By default `min_perc_geno.hap=100` meaning that only completely phased haplotypes are retained.

3. **Discard SNPs with a high amount of missing data**. It is possible to discard SNPs genotyped on less than `min_perc_geno.snp` % of the haplotypes. By default `min_perc_geno.snp=100` meaning that only fully genotyped SNPs are retained.

4. **Discard SNPs with a low Minor Allele Frequency**. It is possible to discard SNPs with a MAF below `min_maf`. This is generally not recommended and by default `min_maf=0` meaning that all SNPs are retained.[1]

More details about the different arguments of the function are available in the documentation accessible using the command:

```
?data2haplohh
```

In the following we detail three different examples based on the example data files provided with the package (see 1).

1.3.1 Example 1: reading haplotype file in standard format

In this example, the example haplotype input file `bta12_cgu.hap` (standard format) and SNP information input files `map.inp` are converted into an `haplohh` object named `hap`. Because the map file contains information for SNPs mapping to other chromosomes than the one of interest (BTA12), we use the option `chr.name=12`. Allele recoding is activated (`recode.allele=TRUE`) to allow recoding alleles in the *rehh* format (0,1 or 2).

```
hap<-data2haplohh(hap_file="bta12_cgu.hap",map_file="map.inp",
                  recode.allele=TRUE,chr.name=12)
```

```
> Map file seems OK: 1424  SNPs declared for chromosome 12
> Standard rehh input file assumed
> Alleles are being recoded according to map file as:
>   0 (missing data), 1 (ancestral allele) or 2 (derived allele)
> Discard Haplotype with less than  100 % of genotyped SNPs
> No haplotype discarded
> Discard SNPs genotyped on less than  100 % of haplotypes
> No SNP discarded
> Data consists of 280 haplotypes and 1424 SNPs
```

If no value is specified for `chr.name` argument and more than one chromosome is detected in the map file, the function asks interactively which chromosome to choose:

```
hap<-data2haplohh(hap_file="bta12_cgu.hap",map_file="map.inp",
                  recode.allele=TRUE)
```

```
> More than one chromosome name in Map file: map.inp
> Which chromosome should be considered among:
> 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29
> 1:
```

```
12
```

```
> Map file seems OK: 1424  SNPs declared for chromosome 12
> Standard rehh input file assumed
> Alleles are being recoded according to map file as:
>   0 (missing data), 1 (ancestral allele) or 2 (derived allele)
```

[1]The arguments `min_perc_geno.hap`, `min_perc_geno.snp` and `min_maf` are evaluated in this order.

```
> Discard Haplotype with less than  100 % of genotyped SNPs
> No haplotype discarded
> Discard SNPs genotyped on less than  100 % of haplotypes
> No SNP discarded
> Data consists of 280 haplotypes and 1424 SNPs
```

Finally, as an example of error message, the following message is prompted if the number of SNPs in the chromosome (for instance when a wrong chromosome is declared) does not correspond to the one in the haplotype file:

```
hap<-data2haplohh(hap_file="bta12_cgu.hap",map_file="map.inp",
                  recode.allele=TRUE,chr.name=18)
```

```
> Map file seems OK: 1123  SNPs declared for chromosome 18
> Standard rehh input file assumed
> The number of snp in the haplotypes 1424  is not equal
> to the number of snps declared in the map file 1123

> Error in data2haplohh(hap_file = "bta12_cgu.hap", map_file = "map.inp", : Conversion stopped
```

1.3.2 Example 2: reading haplotype file in transposed format (*SHAPIT2*–like)

In this example, the example haplotype input file `bta12_cgu.thap` (transposed format) and SNP information input files `map.inp` are converted into an `haplohh` object named `hap`. Setting `haplotype.in.columns=TRUE` informs the function that the haplotype file is in transposed format:

```
hap<-data2haplohh(hap_file="bta12_cgu.thap",map_file="map.inp",haplotype.in.columns=TRUE,
                  recode.allele=TRUE,chr.name=12)
```

```
> Map file seems OK: 1424  SNPs declared for chromosome 12
> Haplotype are in columns with no header
> Alleles are being recoded according to map file as:
>   0 (missing data), 1 (ancestral allele) or 2 (derived allele)
> Discard Haplotype with less than  100 % of genotyped SNPs
> No haplotype discarded
> Discard SNPs genotyped on less than  100 % of haplotypes
> No SNP discarded
> Data consists of 280 haplotypes and 1424 SNPs
```

1.3.3 Example 3: reading haplotype file in *fastPHASE* output format

In this example, the example *fastPHASE* output file `bta12_hapguess_switch.out` and SNP information input files `map.inp` are converted into a `haplohh` object named `haplo`. As explained above we use the option `chr.name=12`. Because, haplotypes originate from several populations (the -u *fastPHASE* option was used), we specify the population of interest (in our example the 280 haplotypes from the CGU population, see above) using the option `popsel=7` (7 corresponding to the code of CGU in the example *fastPHASE* input files).

```
hap<-data2haplohh(hap_file="bta12_hapguess_switch.out",map_file="map.inp",
                  recode.allele=TRUE,popsel=7,chr.name=12)
```

```
> Map file seems OK: 1424  SNPs declared for chromosome 12
> Looks like a FastPHASE haplotype file
> Haplotypes originate from  8  different populations in the fastPhase output file
> Alleles are being recoded according to map file as:
>   0 (missing data), 1 (ancestral allele) or 2 (derived allele)
> Discard Haplotype with less than  100 % of genotyped SNPs
> No haplotype discarded
> Discard SNPs genotyped on less than  100 % of haplotypes
> No SNP discarded
> Data consists of 280 haplotypes and 1424 SNPs
```

If no value is specified for the `popsel` argument and more than one population is detected in the *fastPHASE* output file, the function asks interactively which population to chose:

```
hap<-data2haplohh(hap_file="bta12_hapguess_switch.out",map_file="map.inp",
                  recode.allele=TRUE,chr.name=12)
```

```
> Map file seems OK: 1424  SNPs declared for chromosome 12
> Looks like a FastPHASE haplotype file
> Haplotypes originate from  8  different populations in the fastPhase output file
> Chosen pop. is not in the list of pop. number: 1 2 3 4 5 6 7 8
> Which population should be considered among: 1 2 3 4 5 6 7 8
> 1:
```

```
7
```

```
> Map file seems OK: 1424  SNPs declared for chromosome 12
> Looks like a FastPHASE haplotype file
> Haplotypes originate from  8  different populations in the fastPhase output file
> Alleles are being recoded according to map file as:
>   0 (missing data), 1 (ancestral allele) or 2 (derived allele)
> Discard Haplotype with less than  100 % of genotyped SNPs
> No haplotype discarded
> Discard SNPs genotyped on less than  100 % of haplotypes
> No SNP discarded
> Data consists of 280 haplotypes and 1424 SNPs
```

2 Computing *EHH*-based statistics for individual markers with the `calc_ehh()`, `calc_ehhs()` and `scan_hh()` functions

2.1 Definition and Computation

2.1.1 The (allele-specific) Extended Haplotype Homozygosity (*EHH*)

For a given core allele (i.e., the ancestral or derived allele) at a focal SNP, the (allele–specific) extended haplotype homozygosity (*EHH*) is defined as the probability that two randomly chosen chromosomes (carrying the core allele considered) are identical by descent (as assayed by homozygosity at all SNPs) over a given surrounding chromosome region (Sabeti *et al.* 2002). The *EHH* thus aims at measuring to which extent an extended haplotype is transmitted without recombination. In practice, the *EHH* ($\text{EHH}_{s,t}$) of a tested core allele a_s ($a_s = 1$ or $a_s = 2$) for a focal SNP s over the chromosome interval extending to SNP t is computed as:

$$\mathrm{EHH}_{s,t} = \frac{1}{n_{a_s}(n_{a_s}-1)} \sum_{k=1}^{K_{a_s,t}} n_k(n_k-1) \tag{1}$$

where n_{a_s} represents the number of haplotype carrying the core allele a_s, $K_{a_s,t}$ represents the number of different extended haplotypes (from SNP s to SNP t) carrying a_s and n_k is the number of the extended haplotype k (i.e., $n_{a_s} = \sum_{k=1}^{K_{a_s,t}} n_k$).

2.1.2 The integrated (allele-specific) *EHH* (*iHH*)

By definition, irrespective of the allele considered, *EHH* starts at 1, and decays monotonically to 0 with increasing distance from the focal SNP. For a given core allele, the integrated *EHH* (*iHH*) is defined as the area under the *EHH* curve with respect to map position (Voight *et al.* 2006)[2]. In *rehh*, *iHH* is computed using the trapezoid method. In practice, the integral might only be computed for the regions of the curve above an arbitrarily small *EHH* value (e.g., *EHH*>0.05).

2.1.3 The site-specific Extended Haplotype Homozygosity (*EHHS*)

For a given core SNP, the (site–specific) extended haplotype homozygosity (*EHHS*) is defined as the probability that two randomly chosen chromosomes are identical by descent (as assayed by homozygosity at all SNPs) over a given surrounding chromosome region. *EHHS* might roughly be viewed as linear combination of the *EHH*'s for the two alternative alleles with weights function of the corresponding allele frequencies. Two different *EHHS* estimators further denoted as $\mathrm{EHHS}^{\mathrm{Sab}}$ (Sabeti *et al.* 2007) and $\mathrm{EHHS}^{\mathrm{Tang}}$ (Tang *et al.* 2007) have been proposed. For a focal SNP s over a chromosome interval extending to SNP t, these are computed as (following the same notation as above):

$$\mathrm{EHHS}^{\mathrm{Sab}}_{s,t} = \frac{1}{n_s(n_s-1)} \sum_{a_s=1}^{a_s=2} \left(\sum_{k=1}^{K_{a_s,t}} n_k(n_k-1) \right) \tag{2}$$

where $n_s = \sum_{a_s=1}^{a_s=2} n_{a_s}$ and

$$\mathrm{EHHS}^{\mathrm{Tang}}_{s,t} = \frac{1-h^{(s,t)}_{hap}}{1-h^{(s)}_{all}} \tag{3}$$

where:

1. $h^{(s)}_{all} = \frac{n_s}{n_s-1}\left(1 - \frac{1}{n_s^2}\sum_{a_s=1}^{a_s=2} n^2_{a_s}\right)$ is an estimator of the focal SNP heterozygosity
2. $h^{(s,t)}_{hap} = \frac{n_s}{n_s-1}\left(1 - \frac{1}{n_s^2}\sum_{a_s=1}^{a_s=2}\left(\sum_{k=1}^{K_{a_s,t}} n^2_k\right)\right)$ is an estimator of haplotype heterozygosity across the chromosome region extending from SNP s to SNP t.

[2]In their seminal paper, Voight et al. considered genetic distances and apply a penalty (proportional to physical distances) for successive SNPs separated by more than 20 kb. In addition, they did not compute *iHH* if any physical distance between a pair of neighboring SNPs was above 200 kb. We did not implement such an approach in *rehh* although this might easily be done (providing relevant information is available for the genome of interest) by modifying the positions of the markers in SNP information input file.

2.1.4 The integrated *EHHS* (*iES*)

As for the *EHH* (see 2.1.2), *EHHS* starts at 1 and decays monotonically to 0 with increasing distance from the focal SNP. For a given focal SNP, and in a similar fashion as the *iHH*, *iES* is defined as the integrated *EHHS* (Tang *et al.* 2007). Depending on the *EHHS* estimator considered (respectively, EHHS^{Sab} and $\text{EHHS}^{\text{Tang}}$), two different *iES* estimators, that we further denoted as iES^{Sab} and iES^{Tang} respectively, can be computed. As for *iHH*, the *iES* integral is computed using the trapezoid method and might only be computed for the regions of the curve above an arbitrarily small *EHHS* value (e.g., *EHHS*>0.05).

2.1.5 Dealing with missing data

In the computation of both *EHH* and *EHHS* from a focal SNP s to a SNP t, only extended haplotypes with no missing data are considered. As a consequence, the number of extended haplotypes retained to compute these two statistics might decrease with distance from the focal SNP. However if the number of available extended haplotypes falls below a threshold (e.g., `limhaplo`=5), *EHH* and *EHHS* are not computed further. Note however that most phasing programs (such as *fastPHASE* or *SHAPEIT2*) allow to impute missing genotypes resulting in phased haplotypes with no missing data.

2.2 The function `calc_ehh()`

The `calc_ehh()` function allows to compute *EHH* for both the ancestral ($a_s = 1$) and derived ($a_s = 2$) alleles at the s^{th} SNP relative to each SNP (t) upstream and downstream and corresponding *iHH*. The two options `limehh` and `limhaplo` allow to specify condition to stop computing *EHH* (see 2.1.1). By default `limehh`=0.05 and `limhaplo`=2. Finally, if `plotehh=TRUE`, the decay of *EHH* for both the ancestral and derived alleles is plotted against SNP map position (`main_leg` allows to change the plot legend). More details are available in the R documentation by using the command:

```
?calc_ehh
```

In the following example,the *EHH* statistics are computed for both the ancestral and derived allele of the 456th focal SNP. Note that the `haplohh_cgu_bta12` object was generated using the `data2haplohh()` function with the example input files (1.3.1). For convenience, it is stored as an example object (accessible with the R function `data`) as shown below:

```
#example haplohh object (280 haplotypes, 1424 SNPs) see ?haplohh_cgu_bta12 for details
data(haplohh_cgu_bta12)
#computing EHH statistics for the focal SNP at position 456
#which display a strong signal of selection
res.ehh<-calc_ehh(haplohh_cgu_bta12,mrk=456)
```

The five different elements of the resulting `res.ehh` object are as follows:

```
res.ehh$ehh[1:2,454:458]
```

```
>                    F1205380  F1205390 F1205400  F1205420  F1205440
> Ancestral allele 0.2764706 0.5529412        1 0.8879552 0.6422969
> Derived allele   1.0000000 1.0000000        1 1.0000000 1.0000000
```

```
res.ehh$nhaplo_eval[1:2,454:458]
```

```
>                   F1205380 F1205390 F1205400 F1205420 F1205440
> Ancestral allele        85       85       85       85       85
> Derived allele         195      195      195      195      195
```

```
res.ehh$freq_all1
```

```
> [1] 0.3035714
```

```
res.ehh$ihh
```

```
> Ancestral allele   Derived allele
>           284633          2057152
```

In addition, as `plotehh=TRUE` by default, we obtain the following plot (Figure 1):

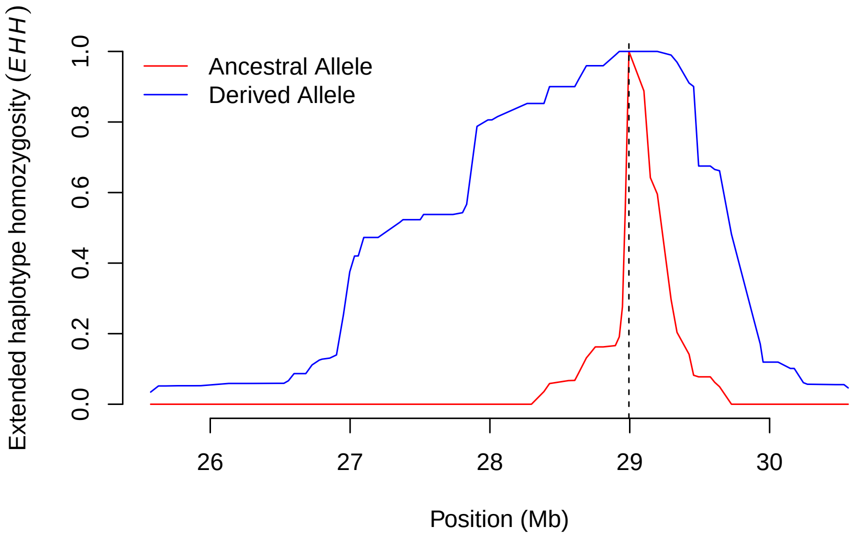


Figure 1: Graphical output for the `calc_ehh()` function

2.3 The function `calc_ehhs()`

The `calc_ehhs()` function allows to compute the *EHHS* (both the EHHS$^{\mathrm{Sab}}$ and EHHS$^{\mathrm{Tang}}$ estimators) at the s^{th} SNP relative to each SNP (t) upstream and downstream. This function also compute the corresponding *iES* (iES$^{\mathrm{Sab}}$ and iES$^{\mathrm{Tang}}$ estimators respectively). The two options `limehhs` and `limhaplo` allow to specify condition to stop computing *EHHS* (see 2.1.3). By default `limehhs`=0.05 and `limhaplo`=2. Finally, if `plotehhs=TRUE`, the decay of *EHHS* is plotted against SNP map position (`main_leg` allows to change the plot legend). More details are available in the R documentation by using the command:

```
?calc_ehhs
```

In the following example, the *EHHS* statistics are computed for the 456th focal SNP on the `haplohh_cgu_bta12` object defined above (see 2.2) was generated using the `data2haplohh()` function with the example input files (see 1.3.1) described above. For convenience, it is stored as an example object (accessible with the R function `data`).

```
#example haplohh object (280 haplotypes, 1424 SNPs) see ?haplohh_cgu_bta12 for details
data(haplohh_cgu_bta12)
#computing EHH statistics for the focal SNP at position 456
#which display a strong signal of selection
res.ehhs<-calc_ehhs(haplohh_cgu_bta12,mrk=456)
```

The five different elements of the resulting `res.ehhs` object are as follows:

```
res.ehhs$EHHS_Sabeti_et_al_2007[453:459]
```

```
>  F1205370  F1205380  F1205390  F1205400  F1205420  F1205440  F1205450
> 0.5017153 0.5095238 0.5347926 1.0000000 0.5654122 0.5429595 0.5386841
```

```
res.ehhs$EHHS_Tang_et_al_2007[453:459]
```

```
>  F1205370  F1205380  F1205390  F1205400  F1205420  F1205440  F1205450
> 0.8715588 0.8851234 0.9290193 1.0000000 0.9822104 0.9432066 0.9357794
```

```
res.ehhs$nhaplo_eval[453:459]
```

```
> F1205370 F1205380 F1205390 F1205400 F1205420 F1205440 F1205450
>      280      280      280      280      280      280      280
```

```
res.ehhs$IES_Tang_et_al_2007
```

```
> [1] 1760565
```

```
res.ehhs$IES_Sabeti_et_al_2007
```

```
> [1] 964698
```

In addition, as `plotehh=TRUE` by default, we obtain the following plot (Figure 2):

2.4 The function `scan_hh()`

The `scan_hh()` function allows to efficiently compute *IHH* (for both the ancestral and derived alleles) and *IES* (both the iES^{Sab} and iES^{Tang} estimators) for all the SNPs in the `haplohh` object considered. The options `limehh`, `limehhs` and `limhaplo` specify conditions to stop computing *EHH* and *EHHS*. By default `limehh`=`limehhs`=0.05 and `limhaplo`=2. Finally, the option `threads`, set by dafault to `threads`=1, allows to specify the number of available threads to parallelize computation (parallelization being carried out over SNPs). For instance to scan the `haplohh_cgu_bta12` object (containing data on 1424 SNPs for 280 haplotypes), one may use the following command:

```
data(haplohh_cgu_bta12)
res.scan<-scan_hh(haplohh_cgu_bta12)
```

The resulting object `res.scan` is a data frame with `haplohh_cgu_bta12nsnp` (number of SNPs declared in the `haplohh` object) and seven columns giving for each SNP:

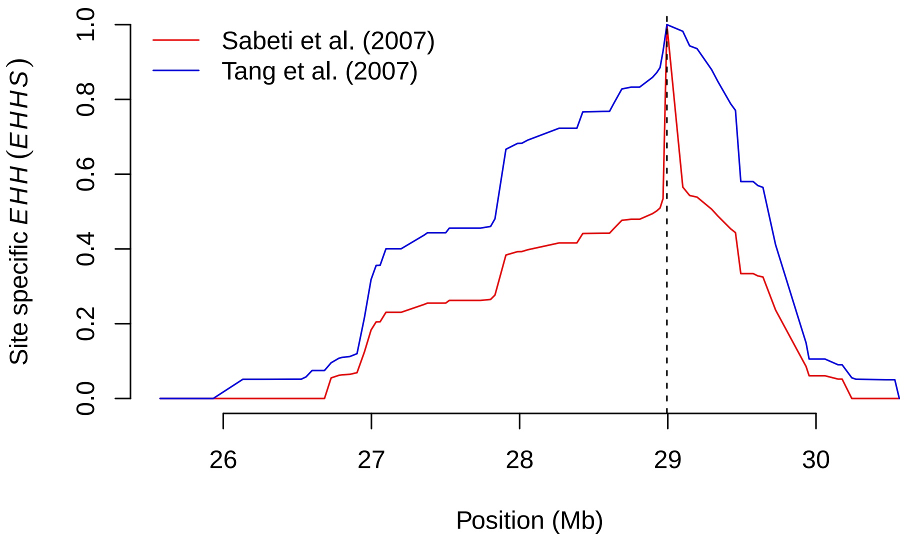


Figure 2: Graphical output for the `calc_ehhs()` function

1. The SNP chromosome of origin
2. The SNP position
3. The SNP ancestral allele frequency
4. The estimated *IHH* for the ancestral allele (*IHH_A*)
5. The estimated *IHH* for the derived allele (*IHH_D*)
6. The estimated iES$^{\text{Tang}}$
7. The estimated iES$^{\text{Sab}}$.

As an example, the following R codes provide the dimension and the first five rows of the `res.scan` data frame obtained above:

```
dim(res.scan)
```

```
> [1] 1424    7
```

```
head(res.scan)
```

```
>          CHR POSITION    freq_A    iHH_A     iHH_D iES_Tang_et_al_2007
> F1200140  12    79823 0.1500000 135102.2  68522.91            69776.85
> F1200150  12   125974 0.4071429 161680.3 107183.15           123607.13
> F1200170  12   175087 0.3571429 157333.1 155777.56           156021.90
> F1200180  12   219152 0.2214286 250037.4 159839.73           166214.75
> F1200190  12   256896 0.1750000 466071.8 173269.33           184453.42
> F1200210  12   316254 0.3892857 292077.5 228681.21           246572.65
>          iES_Sabeti_et_al_2007
> F1200140              53669.39
> F1200150              76287.51
> F1200170              92770.96
> F1200180             110712.37
> F1200190             134092.34
> F1200210             130156.22
```

Note that running `scan_hh()` is more efficient than running `calc_ehh()` and `calc_ehhs()` in turn as shown in the example below (`scan_hh()`:.

```
system.time(res.scan<-scan_hh(haplohh_cgu_bta12))
```

```
>    user  system elapsed
>   0.280   0.000   0.281
```

```
foo<-function(haplo){
res.ihh=res.ies=matrix(0,haplo@nsnp,2)
for(i in 1:length(haplo@position)){
  res.ihh[i,]=calc_ehh(haplo,mrk=i,plotehh=FALSE)$ihh
  tmp=calc_ehhs(haplo,mrk=i,plotehhs=FALSE)
  res.ies[i,1]=tmp$IES_Tang_et_al_2007
  res.ies[i,2]=tmp$IES_Sabeti_et_al_2007
}
list(res.ies=res.ies,res.ihh=res.ihh)
}
system.time(res.scan2<-foo(haplohh_cgu_bta12))
```

```
>    user  system elapsed
>  13.296   0.000  13.321
```

Note however that the same results are obtained (since the same options were used) as illustrated by the following R code:

```
sum(res.scan2$res.ihh[,1]!=res.scan[,4]) + sum(res.scan2$res.ihh[,2]!=res.scan[,5]) +
sum(res.scan2$res.ies[,1]!=res.scan[,6]) + sum(res.scan2$res.ies[,2]!=res.scan[,7])
```

```
> [1] 0
```

3 Computing *iHS*, *Rsb* and *xpEHH*: the `ihh2ihs()`,`ies2rsb()` and `ies2xpehh()` functions

3.1 Within population test: the *iHS*

3.1.1 Definition

Let UniHS represent the log-ratio of the *iHH* for its ancestral (iHH_a) and derived (iHH_d) alleles:

$$\text{UniHS} = \log\left(\frac{\text{iHH}_a}{\text{iHH}_d}\right)$$

The *iHS* of a given focal SNP s (iHS(s)) is then defined as its standardized UniHS (UniHS(s)) following (Voight *et al.* 2006):

$$\text{iHS}(s) = \frac{\text{UniHS}(s) - \mu^{p_s}_{\text{UniHS}}}{\sigma^{p_s}_{\text{UniHS}}}$$

where $\mu^{p_s}_{\text{UniHS}}$ and $\sigma^{p_s}_{\text{UniHS}}$ represent the average and standard deviation of the UniHS computed over all the SNPs with a derived allele frequency p_s similar to that of the core SNP s. In practice, the derived allele frequencies are generally binned so that each bin are large enough (e.g., >10 SNPs) to obtain reliable estimate of $\mu^{p_s}_{\text{UniHS}}$ and $\sigma^{p_s}_{\text{UniHS}}$.

Note that the *iHS* is constructed to have an approximately standard Gaussian distribution and to be comparable across SNPs regardless of their underlying allele frequencies. Hence, one may further transform *iHS* into p_{iHS} (Gautier and Naves 2011):

$$p_{\text{iHS}} = -\log_{10}\left(1 - 2|\Phi\left(\text{iHS}\right) - 0.5|\right)$$

where $\Phi\left(x\right)$ represents the Gaussian cumulative distribution function. Assuming most of the genotyped SNPs behave neutrally (i.e., the genome-wide empirical *iHS* distribution is a fair approximation of the neutral distribution), p_{iHS} might thus be interpreted as a two-sided P-value (on a $-\log_{10}$ scale) associated to the neutral hypothesis of no selection.

3.1.2 The function `ihh2ihs()`

The `ihh2ihs()` function allows to compute *iHS* using a matrix of *iHH* statistics (for both the ancestral and derived alleles) in the same format as obtained from the `scan_hh()` function (see 2.4). The argument `minmaf` allows to remove SNPs according to their MAF (by default SNPs with a MAF<`minmaf`=0.05 are discarded from the standardization). The argument `freqbin` controls the size of the allele frequency bins used to perform standardization (see 3.1.1). More precisely allele frequency bins vary from `minmaf` to 1-`minmaf` per step of size `freqbin` (by default `freqbin`=0.025). Note that if `freqbin` is set to 0 (e.g., with a large number of SNPs and few haplotypes), standardization is performed considering each observed frequency as a frequency class.

For instance, to perform a whole genome scan one might run `scan_hh()` in turn on haplotype data from each chromosome and concatenate the resulting matrices before standardization. In the following example, we assume that the haplotype files are named as `hap_chr_i.pop1` where the chromosome number i goes from 1 to 29 and the SNP information file is named `snp.info`. The R code below then generates a matrix `wg.res` with iHH_a and iHH_d estimates for all SNPs in an appropriate format to perform standardization with the `ihh2ihs` function:

```
for(i in 1:29){
 hap_file=paste("hap_chr_",i,".pop1",sep="")
 data<-data2haplohh(hap_file="hap_file","snp.info",chr.name=i)
 res<-scan_hh(data)
 if(i==1){wg.res<-res}else{wg.res<-rbind(wg.res,res)}
   }
wg.ihs<-ihh2ihs(wg.res)
```

As a matter of illustration, results of a similar genome scan (Gautier and Naves 2011) are provided as example data sets. The following R code allows to compute the *iHS* for the CGU population:

```
data(wgscan.cgu)
## results from a genome scan (44,057 SNPs) see ?wgscan.eut and ?wgscan.cgu for details
ihs.cgu<-ihh2ihs(wgscan.cgu)
```

The corresponding object `ihs.cgu` is a list with two elements corresponding to

1. a matrix of SNP *iHS* and the corresponding p_{iHS} (P-values in a $-\log_1 0$ scale assuming the *iHS* are normally distributed under the neutral hypothesis). For instance, the five first rows of the `ihs.cgu$iHS` data frame are displayed below using the following R command:

```
head(ihs.cgu$iHS)
```

```
>          CHR POSITION        iHS -log10(p-value)
> F0100190   1   113642 -0.5582992       0.2390952
> F0100220   1   244699  0.2723337       0.1049282
> F0100250   1   369419  0.4810736       0.2003396
> F0100270   1   447278  1.0618710       0.5401640
> F0100280   1   487654  0.8184060       0.3839181
> F0100290   1   524507 -0.3897024       0.1569189
```

2.a matrix summarizing the allele frequency bins. For instance, the five first rows of the `ihs.cgu$frequency.class` data frame are displayed below using the following R command:

```
head(ihs.cgu$frequency.class)
```

```
>              Number of SNPs mean of the log(iHHA/iHHD) ratio
> 0.05 - 0.075           1635                        0.7286087
> 0.075 - 0.1            1316                        0.5804760
> 0.1 - 0.125            1478                        0.4710504
> 0.125 - 0.15           1593                        0.3720585
> 0.15 - 0.175           1078                        0.3263215
> 0.175 - 0.2            1325                        0.2721166
>              sd of the log(iHHA/iHHD) ratio
> 0.05 - 0.075                      0.6457742
> 0.075 - 0.1                       0.5556798
> 0.1 - 0.125                       0.5079392
> 0.125 - 0.15                      0.4708235
> 0.15 - 0.175                      0.4524270
> 0.175 - 0.2                       0.4533404
```

3.1.3 Manhattan plot of the results: the function `ihsplot()`

The `ihsplot()` function allows to draw a Manhattan plot of the Whole Genome scan results as stored in the list object produced by the function `ihh2ies()`. Various options are available to modify the graphical display (see `?ihsplot`).

```
ihsplot(ihs.cgu,plot.pval=TRUE,ylim.scan=2,main="iHS (CGU cattle breed)")
```

3.2 The *Rsb*–based pairwise population test

3.2.1 Definition

For a given SNP s, let

$$\text{LRiES}(s)^{\text{Tang}} = \log\left(\frac{\text{iES}_{\text{pop1}}(s)^{\text{Tang}}}{\text{iES}_{\text{pop2}}(s)^{\text{Tang}}}\right)$$

represent the log-ratio of the $\text{iES}_{\text{pop1}}(s)^{\text{Tang}}$ and $\text{iES}_{\text{pop2}}(s)^{\text{Tang}}$ computed in the pop1 and pop2 populations (see 2.1.4).

The *Rsb* for a given focal SNP is then defined as the standardized $\text{LRiES}(s)^{\text{Tang}}$ (Tang *et al.* 2007):

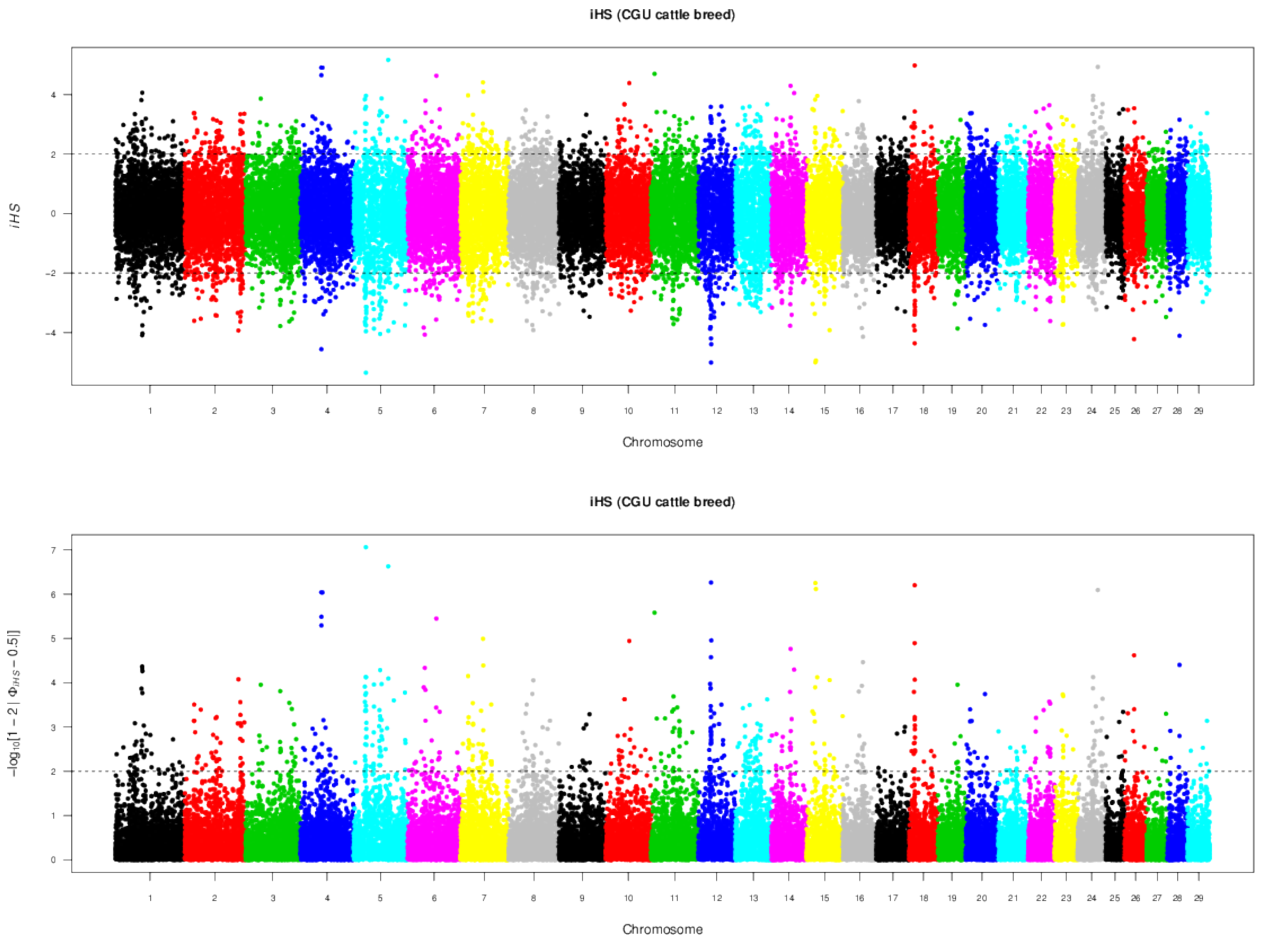


Figure 3: Graphical output for the `ihsplot()` function

$$\mathrm{rSB}(s) = \frac{\mathrm{LRiES}(s)^{\mathrm{Tang}} - \mathrm{med}_{\mathrm{LRiES}^{\mathrm{Tang}}}}{\sigma_{\mathrm{LRiES}^{\mathrm{Tang}}}} \tag{4}$$

where $\mathrm{med}_{\mathrm{LRiES}^{\mathrm{Tang}}}$ and $\sigma_{\mathrm{LRiES}^{\mathrm{Tang}}}$ represent the median and standard deviation of the $\mathrm{LRiES}(s)^{\mathrm{Tang}}$ computed over all the analyzed SNPs. Note that the median is used instead of the mean because it is less sensitive to extreme data points (Tang *et al.* 2007). More importantly, it should be noticed that the information about the ancestral and derived status of alleles at the focal SNP is not needed.

As for the *iHS* (see 3.1.1), *Rsb* is constructed to have an approximately standard Gaussian distribution and may further be transformed into p_{rSB}:

$$p_{\mathrm{rSB}} = -\log_{10}\left(1 - 2|\Phi\left(\mathrm{rSB}\right) - 0.5|\right) \tag{5}$$

where $\Phi\left(x\right)$ represents the Gaussian cumulative distribution function. Assuming most of the genotyped SNPs behave neutrally (i.e., the genome-wide empirical *Rsb* distribution is a fair approximation of their corresponding neutral distributions), p_{rSB} might thus be interpreted as a two-sided P-value (in a $-\log_{10}$ scale) associated to the neutral hypothesis of no selection. Alternatively, p_{rSB} might also be computed (Gautier and Naves 2011):

$$p\prime_{\mathrm{rSB}} = -\log_{10}\left(|\Phi\left(\mathrm{rSB}\right)|\right) \tag{6}$$

$p\prime_{\mathrm{rSB}}$ and $p\prime_{\mathrm{rSB}}$ might then be interpreted as a one-sided P-value (in a $-\log_{10}$ scale) allowing the identification of those sites displaying a significantly high extended haplotype homozygosity in population *pop*2 (represented in the denominator of the corresponding LRiES) relatively to the *pop*1 reference population.

3.2.2 The function `ies2rsb()`

The `ies2rsb()` function allows to compute *Rsb* using two data frames containing the *iES* statistics for each of the two populations considered in the same format as the one obtained after running the `scan_hh()` function (see 2.4). For instance, to perform a genome scan one might first run for each population `scan_hh()` in turn on haplotype data from each chromosome and concatenate the resulting matrices. In the following example, we assume that the haplotype files are named as `hap_chr_i.pop1` and `hap_chr_i.pop2` where *i* is the chromosome number (going from 1 to 29), the suffixes pop1 and pop2 indicate the population of origin and the SNP information file is named `snp.info`. The R code below then generates two data frames (`wg.res.pop1` and `wg.res.pop2`) containing the results from all SNPs in the appropriate format to compute *Rsb* with the `ies2rsb()` function:

```
for(i in 1:29){
 hap_file=paste("hap_chr_",i,".pop1",sep="")
 data<-data2haplohh(hap_file="hap_file","snp.info",chr.name=i)
 res<-scan_hh(data)
 if(i==1){wg.res.pop1<-res}else{wg.res.pop1<-rbind(wg.res.pop1,res)}
 hap_file=paste("hap_chr_",i,".pop2",sep="")
 data<-data2haplohh(hap_file="hap_file","snp.info",chr.name=i)
 res<-scan_hh(data)
 if(i==1){wg.res.pop2<-res}else{wg.res.pop2<-rbind(wg.res.pop2,res)}
 }
wg.rsb<-ies2rsb(wg.res.pop1,wg.res.pop2)
```

As a matter of illustration, one may consider results from a similar genome scan (Gautier and Naves 2011) provided as example data sets and compute for each SNP the *Rsb* between the CGU and EUT populations as follows:

```r
data(wgscan.cgu) ; data(wgscan.eut)
## results from a genome scan (44,057 SNPs) see ?wgscan.eut and ?wgscan.cgu for details
cguVSeut.rsb<-ies2rsb(wgscan.cgu,wgscan.eut,"CGU","EUT")
```

The resulting object `cguVSeut.rsb` is a data frame with of SNP *Rsb* (and corresponding P-Values assuming *Rsb* are normally distributed under the neutral hypothesis). Note that either bilateral (default) or unilateral might be performed (`method` argument). The five first rows of the `cguVSeut.rsb` data frame are displayed below using the following R command:

```r
head(cguVSeut.rsb)
```

```
>          CHR POSITION Rsb (CGU vs. EUT) -log10(p-value) [bilateral]
> F0100190   1   113642        -0.3398574                  0.13432529
> F0100220   1   244699        -1.0566283                  0.53658299
> F0100250   1   369419        -0.1468326                  0.05390941
> F0100270   1   447278        -1.8191608                  1.16186336
> F0100280   1   487654        -0.2193069                  0.08280392
> F0100290   1   524507        -0.7941300                  0.36945032
```

3.2.3 Manhattan plot of the results: the function `rsbplot()`

The `rsbplot()` function allows to draw a Manhattan plot of the Whole Genome scan results as stored in the data frame produced by the function `ies2rsb()`. Various options are available to modify the graphical display (see `?rsbplot`). As an example, the Figure 4 below provides the output of the function `rsbplot` for the *xpEHH* computed above across the CGU and EUT populations (see 3.2.2). Figure 4 was drawn using the following R code:

```r
rsbplot(cguVSeut.rsb,plot.pval=TRUE)
```

3.3 The *xpEHH*–based pairwise population test

3.3.1 Definition

The *xpEHH* statistics (Sabeti *et al.* 2007) is similar to the *Rsb* except that it is based on the $\text{iES}_{\text{pop2}}(s)^{\text{Sab}}$ (instead of $\text{iES}_{\text{pop2}}(s)^{\text{Tang}}$) estimator of the *iES* (see 2.1.4). Hence, for or a given SNP s, let

$$\text{LRiES}(s)^{\text{Sab}} = \log\left(\frac{\text{iES}_{\text{pop1}}(s)^{\text{Sab}}}{\text{iES}_{\text{pop2}}(s)^{\text{Sab}}}\right)$$

represent the log-ratio of the $\text{iES}_{\text{pop1}}(s)^{\text{Sab}}$ and $\text{iES}_{\text{pop2}}(s)^{\text{Sab}}$ computed in the pop1 and pop2 populations (see 2.1.4).

The *xpEHH* for a given focal SNP is then defined as the standardized $\text{LRiES}(s)^{\text{Sab}}$ (Sabeti *et al.* 2007):

$$\text{rSB}(s) = \frac{\text{LRiES}(s)^{\text{Sab}} - \text{med}_{\text{LRiES}^{\text{Sab}}}}{\sigma_{\text{LRiES}^{\text{Sab}}}} \qquad (7)$$

where $\text{med}_{\text{LRiES}^{\text{Sab}}}$ and $\sigma_{\text{LRiES}^{\text{Sab}}}$ represent the median and standard deviation of the $\text{LRiES}(s)^{\text{Sab}}$ computed over all the analyzed SNPs. More importantly, it should be noticed that the information about the ancestral and derived status of alleles at the focal SNP is not needed.

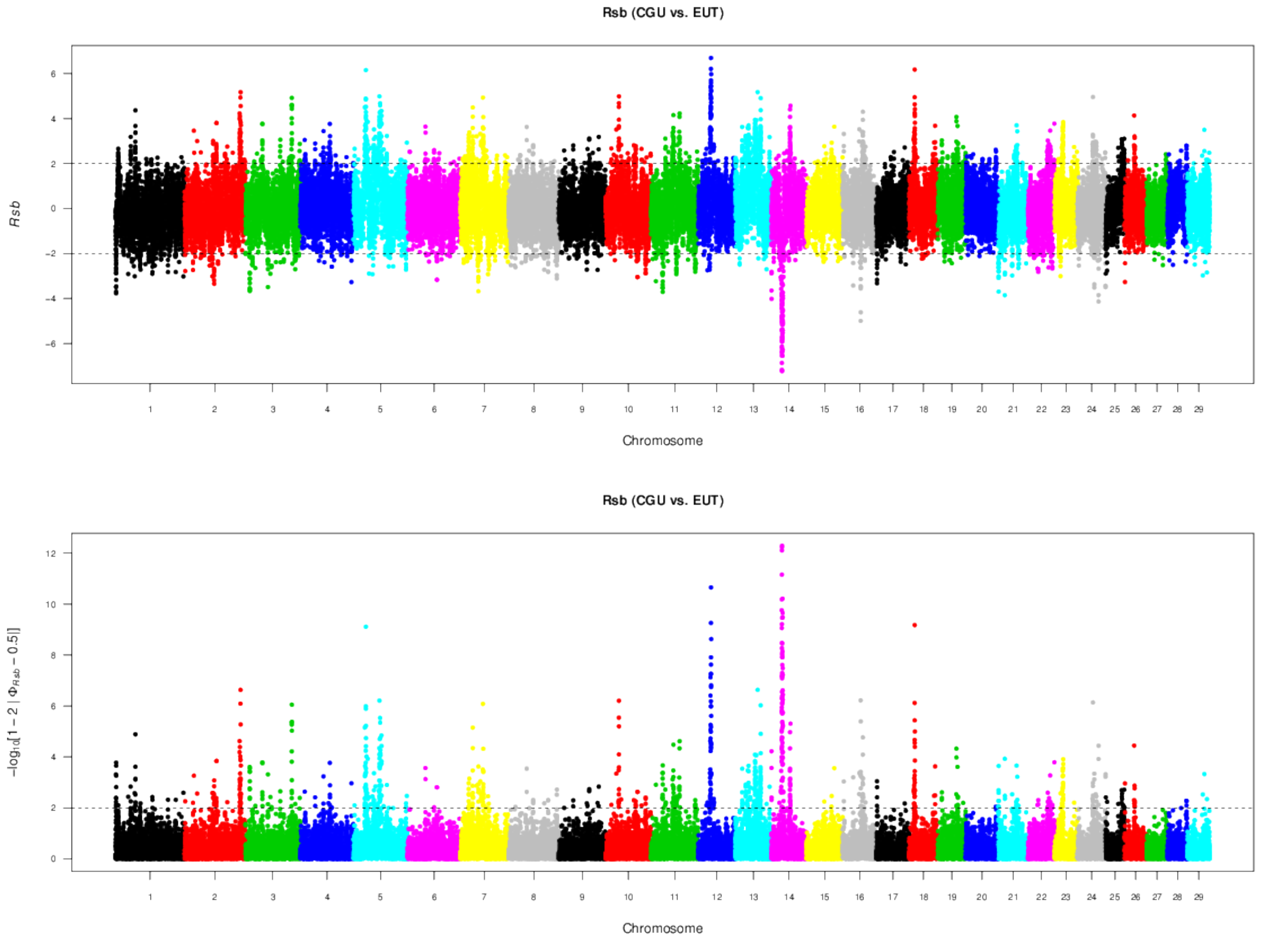


Figure 4: Graphical output for the `rsbplot()` function

As for the *iHS* (see 3.1.1) and *Rsb*, *xpEHH* is constructed to have an approximately standard Gaussian distribution and may further be transformed into p_{xpEHH}:

$$p_{\text{xpEHH}} = -\log_{10}\left(1 - 2|\Phi\left(\text{xpEHH}\right) - 0.5|\right) \tag{8}$$

where $\Phi\left(x\right)$ represents the Gaussian cumulative distribution function. Assuming most of the genotyped SNPs behave neutrally (i.e., the genome-wide empirical *xpEHH* distribution is a fair approximation of their corresponding neutral distributions), p_{xpEHH} might thus be interpreted as a two-sided P-value (in a $-\log_{10}$ scale) associated to the neutral hypothesis of no selection. Alternatively, p_{xpEHH} might also be computed (Gautier and Naves 2011):

$$p\prime_{\text{xpEHH}} = -\log_{10}\left(|\Phi\left(\text{xpEHH}\right)|\right) \tag{9}$$

$p\prime_{\text{xpEHH}}$ and $p\prime_{\text{xpEHH}}$ might then be interpreted as a one-sided P-value (in a $-\log_{10}$ scale) allowing the identification of those sites displaying a significantly high extended haplotype homozygosity in population *pop*2 (represented in the denominator of the corresponding LRiES) relatively to the *pop*1 reference population.

3.3.2 The function `ies2xpehh()`

The `ies2xpehh()` function allows to compute *xpEHH* using two data frames containing the *iES* statistics for each of the two populations considered in the same format as the one obtained after running the `scan_hh()`

function (see 2.4). For instance, to perform a genome scan one might first run for each population `scan_hh()` in turn on haplotype data from each chromosome and concatenate the resulting matrices. In the following example, we assume that the haplotype files are named as `hap_chr_i.pop1` and `hap_chr_i.pop2` where *i* is the chromosome number (going from 1 to 29), the suffixes pop1 and pop2 indicate the population of origin and the SNP information file is named `snp.info`. The R code below then generates two data frames (`wg.res.pop1` and `wg.res.pop2`) containing the results from all SNPs in the appropriate format to compute *Rsb* with the `ies2rsb()` function:

```
for(i in 1:29){
 hap_file=paste("hap_chr_",i,".pop1",sep="")
 data<-data2haplohh(hap_file="hap_file","snp.info",chr.name=i)
 res<-scan_hh(data)
 if(i==1){wg.res.pop1<-res}else{wg.res.pop1<-rbind(wg.res.pop1,res)}
 hap_file=paste("hap_chr_",i,".pop2",sep="")
 data<-data2haplohh(hap_file="hap_file","snp.info",chr.name=i)
 res<-scan_hh(data)
 if(i==1){wg.res.pop2<-res}else{wg.res.pop2<-rbind(wg.res.pop2,res)}
 }
wg.xpehh<-ies2xpehh(wg.res.pop1,wg.res.pop2)
```

As a matter of illustration, one may consider results from a similar genome scan (Gautier and Naves 2011) provided as example data sets and compute for each SNP the *xpEHH* between the CGU and EUT populations as follows:

```
data(wgscan.cgu) ; data(wgscan.eut)
## results from a genome scan (44,057 SNPs) see ?wgscan.eut and ?wgscan.cgu for details
cguVSeut.xpehh<-ies2xpehh(wgscan.cgu,wgscan.eut,"CGU","EUT")
```

The resulting object `cguVSeut.xpehh` is a data frame with of SNP *xpEHH* (and corresponding P-values assuming *xpEHH* are normally distributed under the neutral hypothesis). Note that either bilateral (default) or unilateral might be performed (`method` argument). The five first rows of this data frame are displayed below using the following R command:

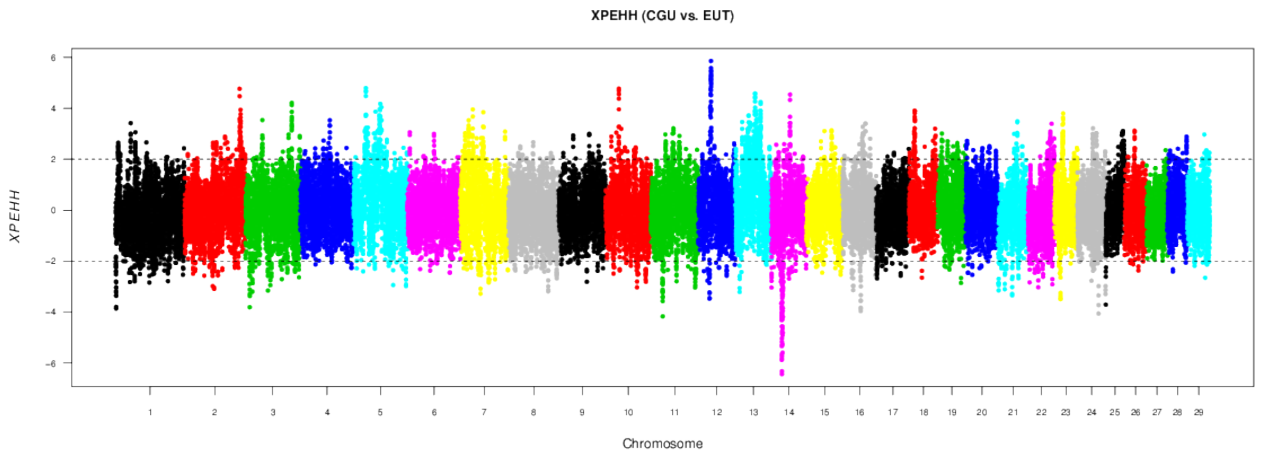
```
head(cguVSeut.xpehh)
```

```
>           CHR POSITION XPEHH (CGU vs. EUT) -log10(p-value) [bilateral]
> F0100190   1   113642          -0.5555841                   0.2377002
> F0100220   1   244699          -0.7516166                   0.3445910
> F0100250   1   369419          -0.8885736                   0.4268588
> F0100270   1   447278          -0.3470522                   0.1375394
> F0100280   1   487654          -0.9182772                   0.4455426
> F0100290   1   524507          -0.7521031                   0.3448721
```

3.3.3 Manhattan plot of the results: the function `xpehhplot()`

The `xpehhplot()` function allows to draw a Manhattan plot of the Whole Genome scan results as stored in the data frame produced by the function `ies2xpehh()`. Various options are available to modify the graphical display (see `?xpehhplot`). As an example, the Figure 5 below provides the output of the function `xpehhplot` for the *xpEHH* computed above across the CGU and EUT populations (see 3.3.2). Figure 5 was drawn using the following R code:

```r
xpehhplot(cguVSeut.xpehh,plot.pval=TRUE)
```



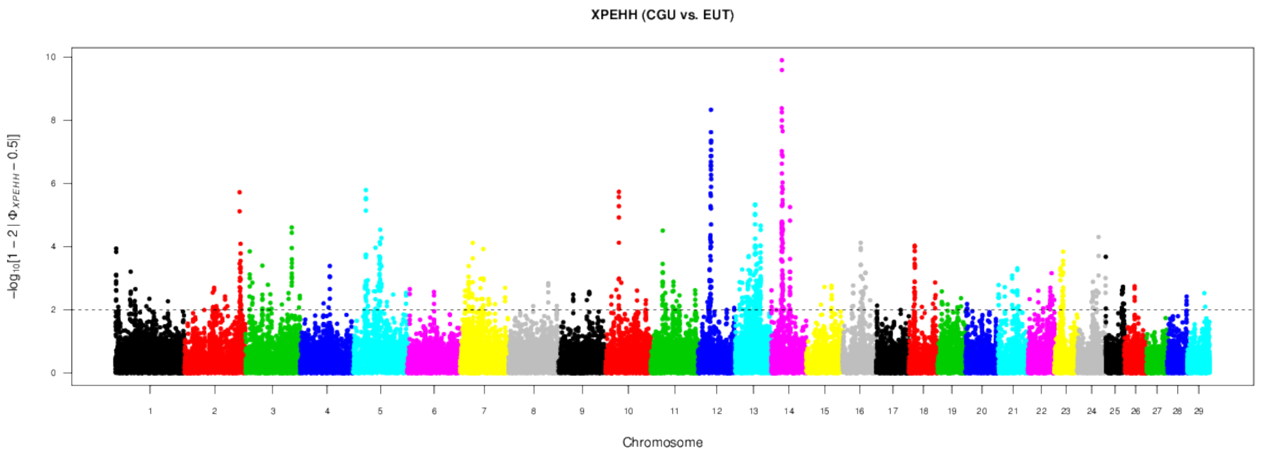


Figure 5: Graphical output for the `xpehhplot()` function

3.3.4 ***xpEHH* vs. *Rsb* comparison:**

A plot of the *xpEHH* against *Rsb* estimates across the CGU and EUT populations (see 3.3.2 and 3.2.2 respectively) is represented in the Figure 6 below. This figure was generated using the following R code:

```r
plot(cguVSeut.rsb[,3],cguVSeut.xpehh[,3],xlab="Rsb",ylab="XPEHH",pch=16,
    cex=0.5,cex.lab=0.75,cex.axis=0.75)
abline(a=0,b=1,lty=2)
```

3.4 Visual inspection of the standardized scores distribution: the function `distribplot()`

The `distribplot` function allows to easily visualize the distributions of the standardized scores (either *iHS*, *Rsb* or *xpEHH*) and compare them to the standard Gaussian distribution. As an example, the Figure 7 below provides the output the function `distribplot` when considering the *iHS* estimates obtained for the CGU population (see 3.1.2) using the following R code:

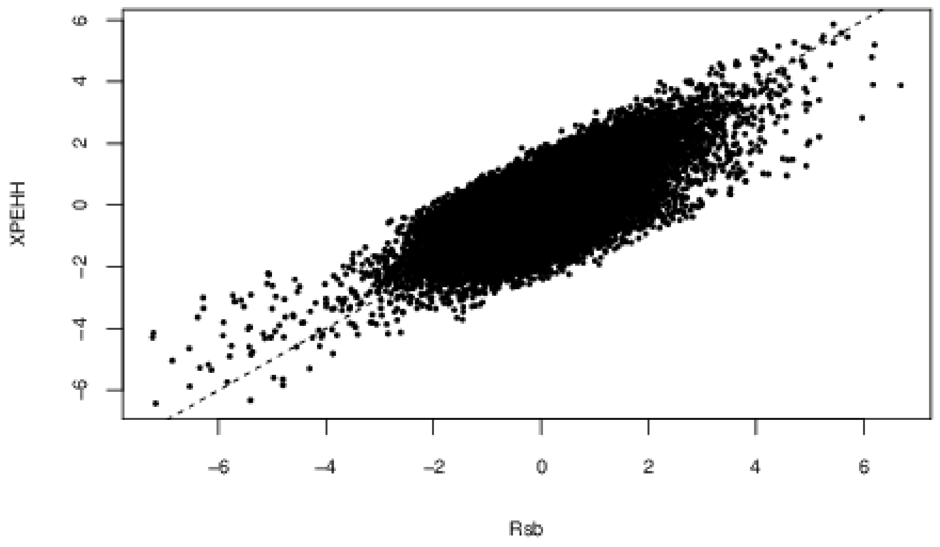


Figure 6: Comparison of the XPEHH and Rsb estimates across the CGU and EUT populations

```
distribplot(ihs.cgu$iHS[,3],xlab="iHS")
```

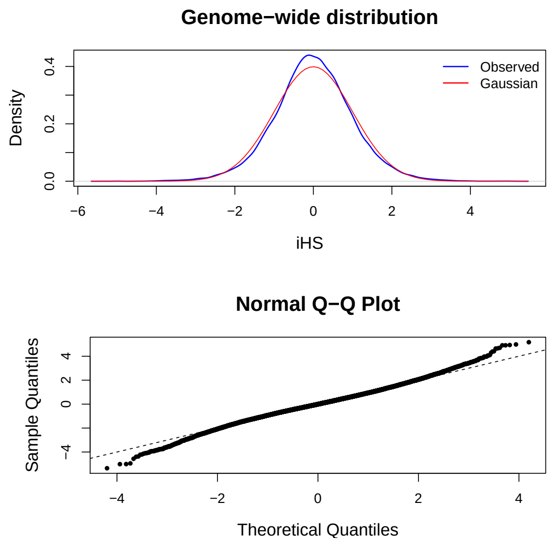


Figure 7: Graphical output for the function `distribplot`

4 Visualizing haplotype structure around a core allele: the function `bifurcation.diagram()`

The function `bifurcation.diagram()` function draws haplotype bifurcation diagrams (Sabeti *et al.* 2002) that allow to better understand the origin of an observed footprints of selection. Such diagrams indeed consist in plotting the breakdown of LD at increasing distances from the core allele at the selected focal SNPs. The root (focal SNP) of each diagram is the core allele and is here identified by a vertical dashed line. The diagram is bi-directional, portraying both centromere-proximal and centromere-distal LD. Moving in one direction, each marker is an opportunity for a node; the diagram either divides or not based on whether both or only one allele is present. Thus the breakdown of LD on the core haplotype background is portrayed at progressively longer distances. The thickness of the lines corresponds to the number of samples with the indicated long-distance haplotype. Several options are available to modify the aspect of the plots (see command `?bifurcation.diagram`) As a matter of illustration, Figure 8 shows the bifurcation diagrams for both the derived and ancestral alleles at the 456th SNP on BTA12 CGU haplotypes. This SNP displayed a strong signal of selection (using both *iHS* and *Rsb* statistics) and is located closed (<5kb) to a strong candidate genes involved in horn development (Gautier and Naves 2011). Figure 8 was obtained with the following R code:

```r
data(haplohh_cgu_bta12)
layout(matrix(1:2,2,1))
bifurcation.diagram(haplohh_cgu_bta12,mrk_foc=456,all_foc=1,nmrk_l=20,nmrk_r=20,
                    main="Bifurcation diagram (RXFP2 SNP on BTA12): Ancestral Allele")
bifurcation.diagram(haplohh_cgu_bta12,mrk_foc=456,all_foc=2,nmrk_l=20,nmrk_r=20,
                    main="Bifurcation diagram (RXFP2 SNP on BTA12): Derived Allele")
```

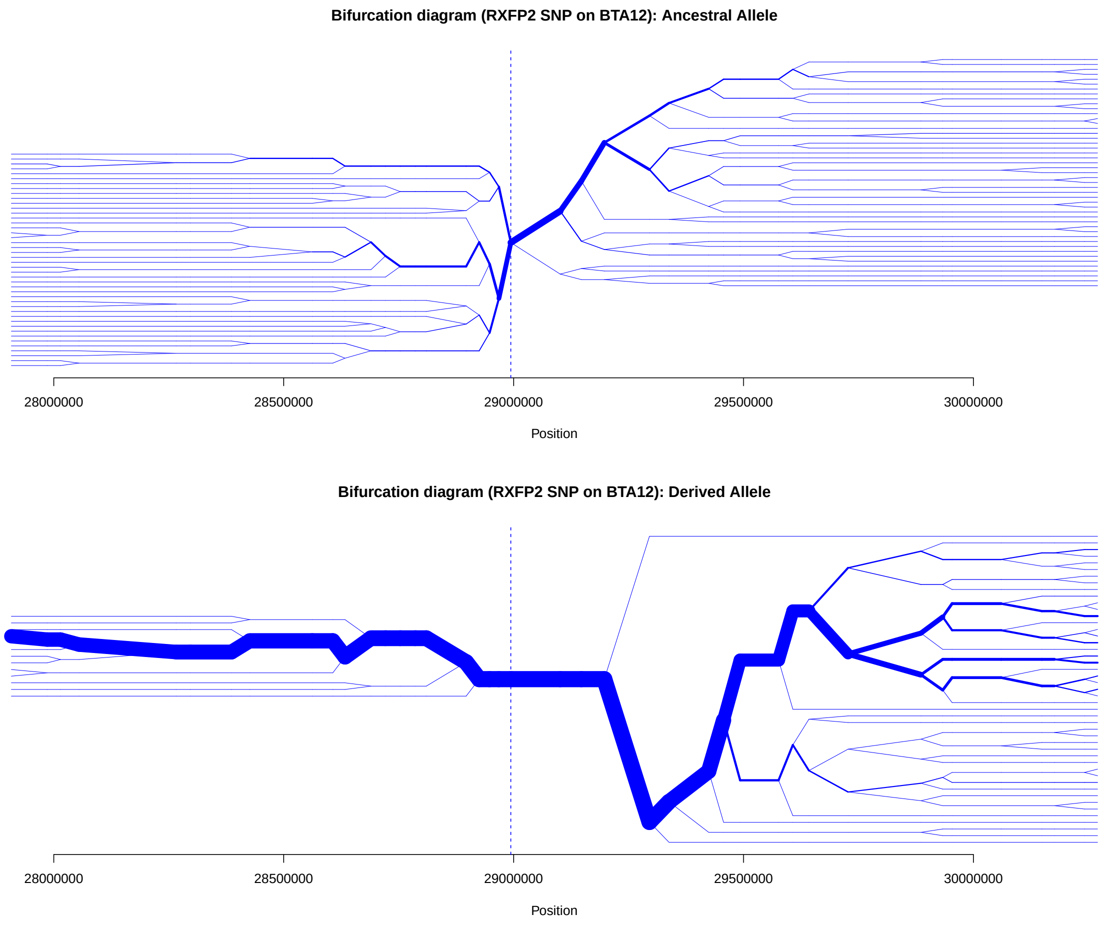


Figure 8: Graphical output for the function `bifurcation.diagram()`

References

Gautier M., Naves M., 2011 Footprints of selection in the ancestral admixture of a New World Creole cattle breed. Mol Ecol **20**: 3128–3143.

O'Connell J., Gurdasani D., Delaneau O., Pirastu N., Ulivi S., others, 2014 A general approach for haplotype phasing across the full spectrum of relatedness. PLoS Genet **10**: e1004234.

Sabeti P. C., Reich D. E., Higgins J. M., Levine H. Z. P., Richter D. J., others, 2002 Detecting recent positive selection in the human genome from haplotype structure. Nature **419**: 832–837.

Sabeti P. C., Varilly P., Fry B., Lohmueller J., Hostetter E., others, 2007 Genome-wide detection and characterization of positive selection in human populations. Nature **449**: 913–918.

Scheet P., Stephens M., 2006 A fast and flexible statistical model for large-scale population genotype data: applications to inferring missing genotypes and haplotypic phase. Am J Hum Genet **78**: 629–644.

Tang K., Thornton K. R., Stoneking M., 2007 A new approach for using genome scans to detect recent positive selection in the human genome. PLoS Biol **5**: e171.

Voight B. F., Kudaravalli S., Wen X., Pritchard J. K., 2006 A map of recent positive selection in the human genome. PLoS Biol **4**: e72.