Fantasio

Version 0.1

Isuru HAUPE & Marie MICHEL

2019-02-18

Contents

1 Introduction

Fantasio is composed of several functions. Its goals are:

- For population genetic studies: estimating and detecting inbreeding on individuals without known genealogy, estimating the population proportion of mating types and the individual probability to be offspring of different mating types
- For rare disease studies: performing homozygosity mapping with heterogeneity
- For multifactorial disease studies: HBD-GWAS strategy

Fantasio implements the creation of several random sparse submaps on genome-wide data (to remove linkage disequilibrium). It also provides graphical outputs to facilitate interpretations of homozygosity mapping results and plots.

In this vignette, we illustrate how to use the package, using the data set HGDP-CEPH, a ship which contains 1043 individuals and 660918 markers. In order to access the data, you will need to download the HGDP.CEPH package (see below how) and load it.

Not all options of the functions are described here, but rather their basic usage. The reader is advised to look at the manual page of the function for details.

1.1 Principal concepts

Fantasio implements a maximum likelihood method that uses a hidden Markov chain to model the dependencies along the genome between the (observed) marker genotypes of an individual, and its (unobserved) homozygous by descent (HBD) status. The emission probabilities of this hidden Markov model (HMM) depend on the allele frequencies. The transition probabilities depend on the genetic distance between two adjacent markers. This model allows estimating the inbreeding coefficient f of an individual, and a parameter a, where af is the instantaneous rate of change per unit map length (here cM) from no HBD to HBD. Both HBD and non-HBD segment lengths are assumed to be distributed exponentially with mean lengths $\frac{1}{a(1-f)}$ and $\frac{1}{af}$, respectively.

The method requires the markers to be in minimal linkage disequilibrium (LD). Otherwise biased estimations of f are produced. A strategy consisting of generating multiple random sparse genome maps (submaps) has been proposed to avoid this bias (Leutenegger et al. 2011). When several submaps are considered, f is estimated as the median value on all the f estimation obtained on the different maps after removing submaps with $a > 1$ (having an average HBD segment length of 1 cM is unlikely to be detected with a SNP density of 1 per 0.5 cM). This strategy has the advantage of not requiring any LD computation on the sample and of minimizing loss of information, as compared with a strategy that based on a single map of markers in minimal LD.

Fantasio statistical framework allows fixing HMM parameters to compute the likelihood of a mating type. These likelihoods can be used for:

- Inferring an individual as inbred by comparing the maximized likelihood with the one to be outbred with a likelihood ratio test
- Estimating the population proportion of mating types
- Estimating the individual probability to be into different mating types

When multiple submaps are used, the median p-values/probabilities are considered. See Leutenegger et al. 2011 for more details on the calculations.

Homozygosity mapping (Lander and Botstein 1987) consists in focusing on inbred affected individuals and searching for a region of the genome of shared homozygosity. Original homozygosity mapping requires that the genealogy of patients be known so that inbred patients can be identified and their respective f estimated. Leutenegger et al. (Leutenegger et al. 2006) proposed using the f estimated on genome-wide genetic data to compute a FLOD score, similar to Morton's LOD score (Morton 1955).

Genin et al. (Genin et al. 2012) adapted the FLOD formula for multiple submaps. FLOD(i)(m,s) is computed for each individual i, each marker m on each submap s, using the equation (1):

$$FLOD^{(i)}(m,s) = log_{10}\frac{P\left(Y^{(i)}_{m,s}|H_1\right)}{P\left(Y^{(i)}_{m,s}|H_0\right)} = log_{10}\frac{P\left(X^{(i)}_{m,s}=1|Y^{(i)}_{m,s}\right)+q.P\left(X^{(i)}_{m,s}=0|Y^{(i)}_{m,s}\right)}{\hat{f}^{(i)}_s+q.\left(1-\hat{f}^{(i)}_s\right)}$$

With the following parameters :

- $Y^{(i)}_{m,s}$ the observed genotype of individual i at marker m on submap s
- H_1 the hypothesis where marker m is linked to the disease, and H_0 the one where it is not
- $X^{(i)}_{m,s}$ the HBD status of individual i at marker m on submap s that is estimated together with the inbreeding coefficient using the HMM of the package
- $\hat{f}^{(i)}_s$ the estimated inbreeding coefficient of individual i on submap s
- q the assumed frequency of the mutation involved in the disease for this individual.

Results are then averaged over the different submaps to obtain a single $FLOD^{(i)}(m)$ at each marker m.

Genin et al. (Genin et al. 2012) proposed to detect fully penetrant rare recessive variants by performing homozygosity mapping on inbred cases from Genome-Wide Association Study (GWAS) data. Linkage evidence is then evaluated over the entire set I of inbred cases by computing a FLOD score, HFLOD(m,α), at each marker m, in presence of genetic heterogeneity using a parameter α (2):

$$HFLOD(m,\alpha) = \sum log_{10}\left[\alpha.\frac{P\left(Y_{m,s}^{(i)}|H_1\right)}{P\left(Y_{m,s}^{(i)}|H_0\right)} + (1-\alpha)\right] = \sum log_{10}\left[\alpha.exp\left(FLOD^{(i)}(m) * log(10)\right) + (1-\alpha)\right]$$

This heterogeneity score is then maximized over α to evaluate the evidence of linkage at marker m where α is the estimate of the proportion of cases linked to this locus (3):

$$HFLOD(m) = max_{\alpha}(HFLOD(m,\alpha))$$

2 Getting started

The package `Fantasio` depends on `gaston`. Hereafter the functions of this package are used for data manipulation. If you are not familiar with `gaston`, please refer to the vignette of this package for more information.

2.1 Installing Fantasio

First and foremost install the package with the following command:

```
install.packages("Fantasio")
```

After that we can load the package.

```
require(Fantasio)
```

```
## Loading required package: Fantasio

## Loading required package: parallel

## Loading required package: gaston

## Loading required package: Rcpp

## Loading required package: RcppParallel

##
## Attaching package: 'RcppParallel'

## The following object is masked from 'package:Rcpp':
##
##     LdFlags

## Gaston set number of threads to 2. Use setThreadOptions() to modify this.

##
## Attaching package: 'gaston'

## The following object is masked from 'package:stats':
##
##     sigma

## The following objects are masked from 'package:base':
##
##     cbind, rbind
```

2.2 Installing example data: HGDP-CEPH

We illustrate the usage of the package with the data provided in the data package HGDP-CEPH. First, we need to install it:

```r
install.packages("HGDP.CEPH", repos="https://genostats.github.io/R/")
```

We can load the HGDP-CEPH data as follows:

```r
require(HGDP.CEPH)
```

```
## Loading required package: HGDP.CEPH
```

```r
filepath <- system.file("extdata", "hgdp_ceph.bed", package="HGDP.CEPH")
x <- read.bed.matrix(filepath)
```

```
## Reading /home/rv/R/x86_64-pc-linux-gnu-library/3.5/HGDP.CEPH/extdata/hgdp_ceph.rds
## Reading /home/rv/R/x86_64-pc-linux-gnu-library/3.5/HGDP.CEPH/extdata/hgdp_ceph.bed
```

The object `x` is a bed.matrix object (see gaston package for details).

We are going to work on the Bedouin population, so after updating the stats in `x` (this adds several informations such as allele frequencies, etc), we select this population:

```r
x <- set.stats(x)
```

```
## ped stats and snps stats have been set.
## 'p' has been set.
## 'mu' and 'sigma' have been set.
```

```r
x.be <- select.inds(x, population == "Bedouin")
```

3 Running Fantasio

Two differents methods for submaps creation are implemented in the package:

- "By Hotspots": with this method we use recombination hotspots to segment the genome. The package contains recombination hotspots for the Human Genome in builds 35 (hg17), 36 (hg18) and 37 (hg19) in datasets `hotspots_hg17`, `hotspots_hg18`,`hotspots_hg19`. The default recombination hotspots dataset used is `hotspots_hg19`. Segments should contain at least a user defined number of markers. Markers are then randomly selected within each segment along the genome. By doing this process we obtain a submap (a list of marker).
- "By Distance" or "by SNPs": with this method we use a fixed step based on genetic or physiscal distance (0.5 cM by default) to pick a marker randomly along the genome. More technically, segments are created whenever there is a gap larger than the step (0.5 cM) between adjacent markers. Each segment is then subdivided in several mini-segments. By default we create 20 mini-segments, each containing at least 50 markers. If this is not possible (not enough markers), we do not create mini-segments. After this process is done, we loop over the mini-segments, pick a random marker and walk through the mini-segments by picking the nearest marker after taking a step (default 0.5 cM) downstream and upstream the mini-segments.

The function `Fantasio` calls two different functions for submaps creation, based on the value of `segments`: `segmentsListByHotspots` and `segmentsListByDistance`. The first function `segmentsListByDistance` is used to create a list of segments though the genome. The second function `makeAtlasByDistance` or `makeAllSubsmapsbyHotspots` is used to create the submaps.

3.1 Hotspots (Default)

By default, the submaps are created using the file of recombination hotspots and summarizing the results for each snp that appears in a submap.

```
F1 <- Fantasio(bedmatrix=x.be, segments="Hotspots", n=5, verbose = FALSE)
```

```
## Warning in setSummary(h, probs = run.proba, recap.by.segments =
## recap.by.segments, : No cases (pheno = 2). HBD, FLOD and HFLOD are computed
## on all individuals
```

Among the slots containg results of interest, the slot `F1@marker_summary` gives **to be completed**

```
F1@marker_summary
```

```
##   markers picked
## 1  609560      0
## 2   45832      1
## 3    4364      2
## 4     742      3
## 5     131      4
## 6     289      5
```

Also **to be completed**

```
head(markerRepresentation(F1))
```

```
##           id chr     pos      dist Freq
## 1 rs11807848   1 1051029 0.5583714    1
## 2  rs6603811   1 1695996 1.1445260    1
## 3 rs10910030   1 2025544 1.7886153    1
## 4 rs11588312   1 2188917 1.8323414    1
## 5   rs903919   1 2212443 1.9805611    1
## 6  rs2173049   1 2229866 1.9948376    1
```

The slot `F1@submap_summary` contains a summmary of results accross submaps for all individuals:

```
head(F1@submap_summary, 10)
```

```
##          FID       IID STATUS SUBMAPS QUALITY       F_MIN       F_MAX
## 1  HGDP00607 HGDP00607      1   5 / 5     100 0.014775645 0.026694093
## 2  HGDP00608 HGDP00608      1   5 / 5     100 0.035932756 0.038941187
## 3  HGDP00609 HGDP00609      1   5 / 5     100 0.038533147 0.046427030
## 4  HGDP00610 HGDP00610      1   5 / 5     100 0.049026951 0.057987338
## 5  HGDP00611 HGDP00611      1   1 / 5      20 0.002630466 0.002630466
## 6  HGDP00612 HGDP00612      1   5 / 5     100 0.053518471 0.078518831
## 7  HGDP00613 HGDP00613      1   5 / 5     100 0.000000000 0.000000000
## 8  HGDP00614 HGDP00614      1   5 / 5     100 0.029900171 0.035733472
## 9  HGDP00615 HGDP00615      1   5 / 5     100 0.085438103 0.093103359
## 10 HGDP00616 HGDP00616      1   5 / 5     100 0.070925600 0.076893159
##         F_MEAN    F_MEDIAN   A_MEDIAN   pLRT_MEDIAN INBRED pLRT_inf_0.05
## 1  0.022403133 0.023963765 0.12288828  2.751027e-26   TRUE             5
## 2  0.037667178 0.038470080 0.05701012  8.372995e-57   TRUE             5
## 3  0.043258910 0.043881562 0.14093995  8.997578e-48   TRUE             5
## 4  0.053112575 0.053321431 0.16293845  1.305343e-59   TRUE             5
## 5  0.002630466 0.002630466 0.31365236  7.799894e-02  FALSE             0
## 6  0.064678085 0.065566439 0.34043322  4.110016e-47   TRUE             5
## 7  0.000000000 0.000000000 0.01000000  1.000000e+00  FALSE             0
## 8  0.032558617 0.032114835 0.15420086  4.482678e-32   TRUE              5
## 9  0.088544940 0.088090605 0.09224188 2.712757e-124   TRUE              5
## 10 0.072720886 0.071474851 0.15923940  1.631883e-82   TRUE              5
```

If you are interested in the values of f and a accross the submaps, you can find them in `F1@estimation_summary`.

The function also computed HBD probabilities that can be found in `F1@HBD_recap`, which are used to determine the position of HBD segments, which are recapitulated in `F1@HBDsegments`. It is required that at least `n.consecutive.marker` markers are HBD before calling a HBD segment. Default value is`n.consecutive.marker=5`.

Moreover, FLOD and HFLOD linkage statistics are computed. The FLOD of each individual is in `F1@FLOD_recap`; they are used to compute the whole sample statistics HFLOD in `F1@HFLOD`.

By default, the package only computes HBD, FLOD statistics and HFLOD statistics for affected individuals (phenotype = 2). In case there are no affected individuals, all individuals are taken into account (a warning is issued). The `list.id` argument allows to specify a list of individuals.

3.2 Hotspots by Segments

For the "Hotspots" method, the results can also be summarized globally for each segment using option `recap.by.segments=TRUE`. In that case, `n.consecutive.marker` should be set to 1.

```
F2 <- Fantasio(bedmatrix=x.be, segments="Hotspots", recap.by.segments=TRUE,
  n.consecutive.marker=1, n=5, verbose=FALSE)
```

```
## Warning in setSummary(h, probs = run.proba, recap.by.segments =
## recap.by.segments, : No cases (pheno = 2). HBD, FLOD and HFLOD are computed
## on all individuals
```

3.3 Distance

```
F3 <- Fantasio(bedmatrix=x.be, segments="Distance", n=5)
```

3.4 How to use the segment.option argument

In order to use the `segment.option` argument you need to pass a list of arguments, each variable name in the list must be an argument name in the function. The function that will be called is either `createsSegmentsListByDistance` if `segments` argument is equal to "Distance" or `segmentsListByHotspots` if `segments` argument is equal to "Hotspots" and the arguments list will be passed to it. So refer to these functions for possible arguments.

```
l <- list(minMarkers=50) #default is 0
F1.l <- Fantasio(bedmatrix=x.be, segments="Hotspots", segment.options=l)
```

In the case of "Hotspots", by default, we do not require to have a minimum number of markers in each segment (`minMarkers = 0`). With the above command line, we impose to have at least 50 markers in a segment. Otherwise it is merged with the previous segment.

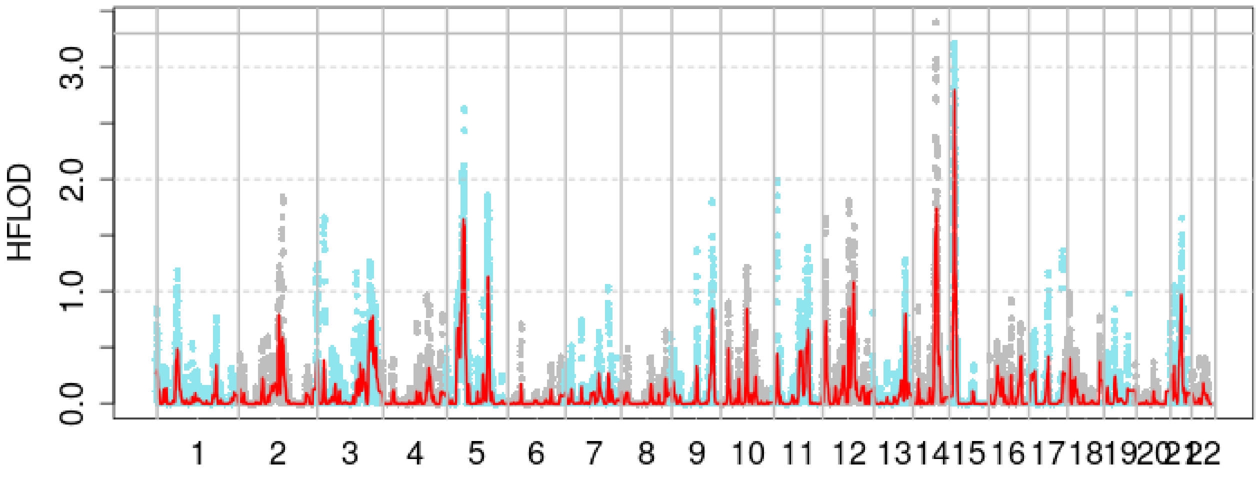
4 Plotting results

We illustrate the plotting function on the `F1` object obtained with `segments="Hotspots"` and `F2` obtained with `segments="Hotspots"` and `recap.by.segments=TRUE`.

4.1 HFLOD plots

This plot shows the values of the HFLOD linkage statistics along the genome (contained in `F1@HFLOD`).

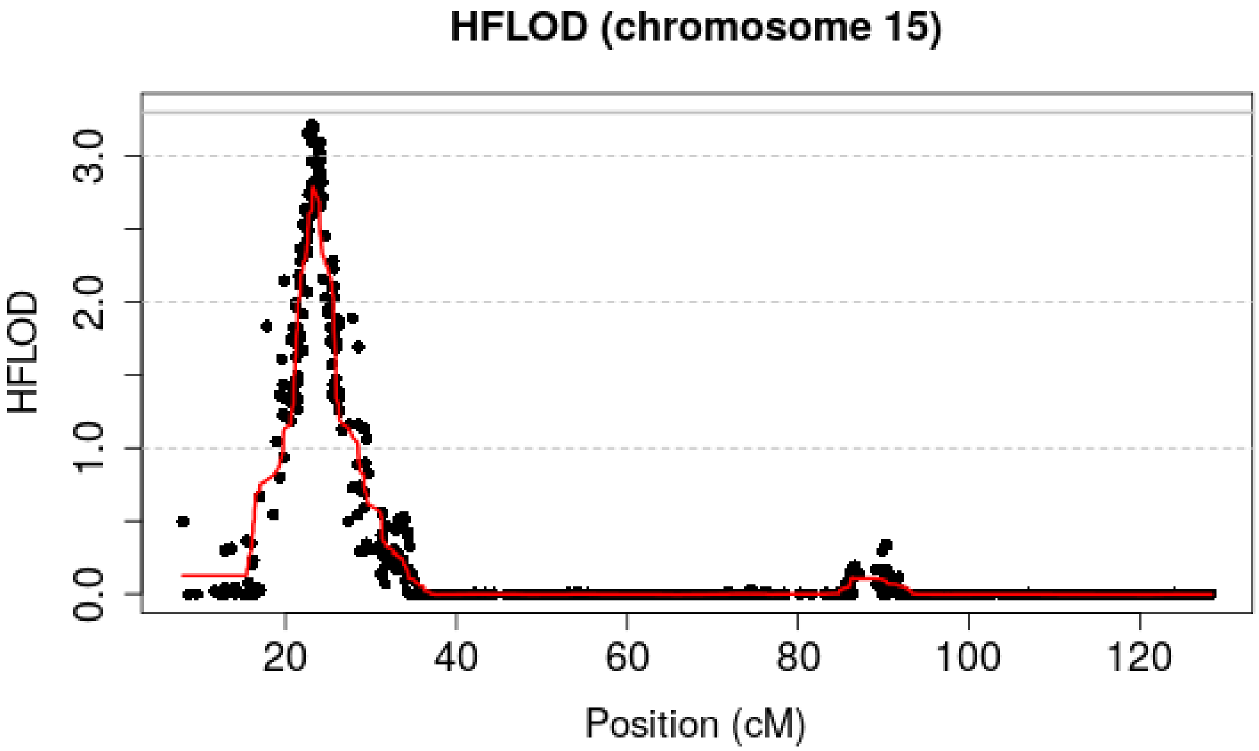
```
HFLODManhattanPlot(F1)
```



The points are the HFLOD values. In the case of `F1`, they are computed at every marker included in at least one submaps. The red line is the value of their moving average (see below).

To plot a single chromosome, use the following function:

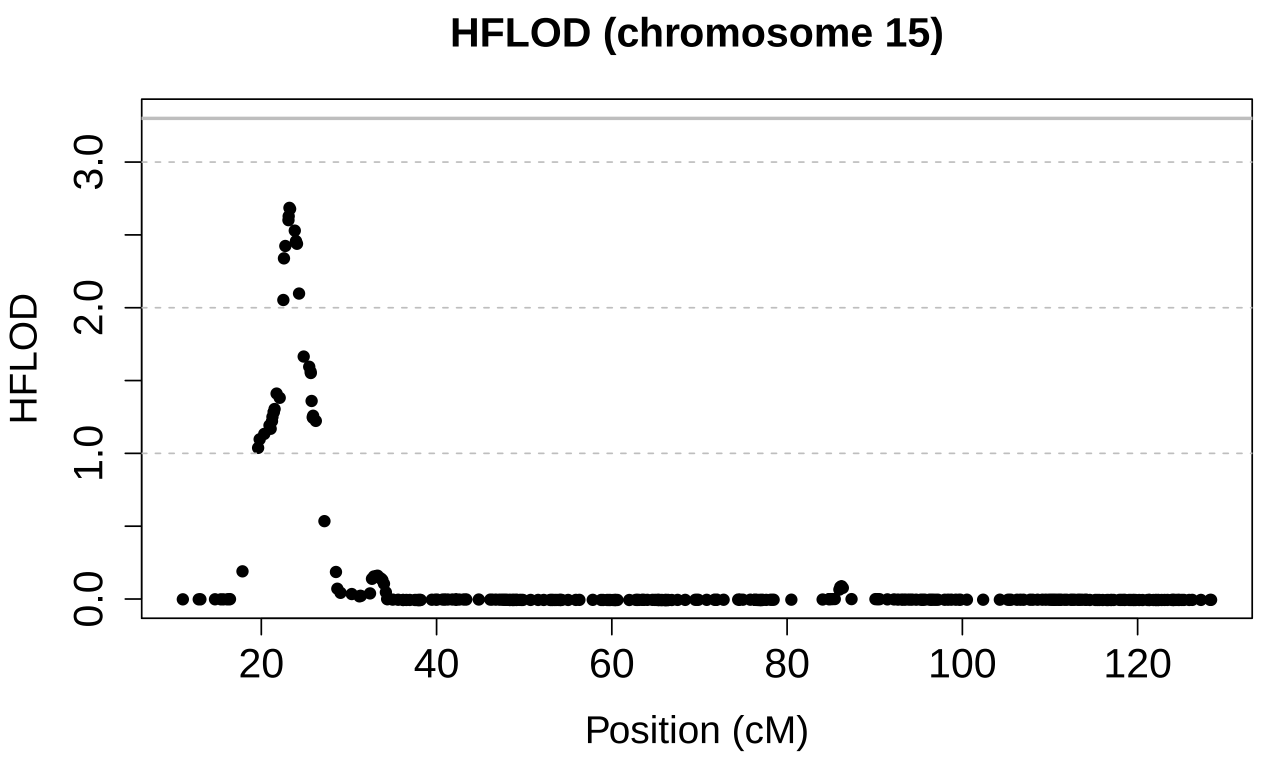
```
HFLODplotChr(F1, chr=15)
```



- The red line plotted is the moving average of the HFLOD, calculated on moving windows of 50 markers. This allows checking the consistency of HFLOD calculations (i.e. checking the fact that a high HFLOD score is not due to one submap only).
- For method “Hotspots by segments”, the use of the moving average does not make much sense as

results are already averaged over the SNPs of a segment between hospots regions. We recommend using `MA=FALSE` as this:

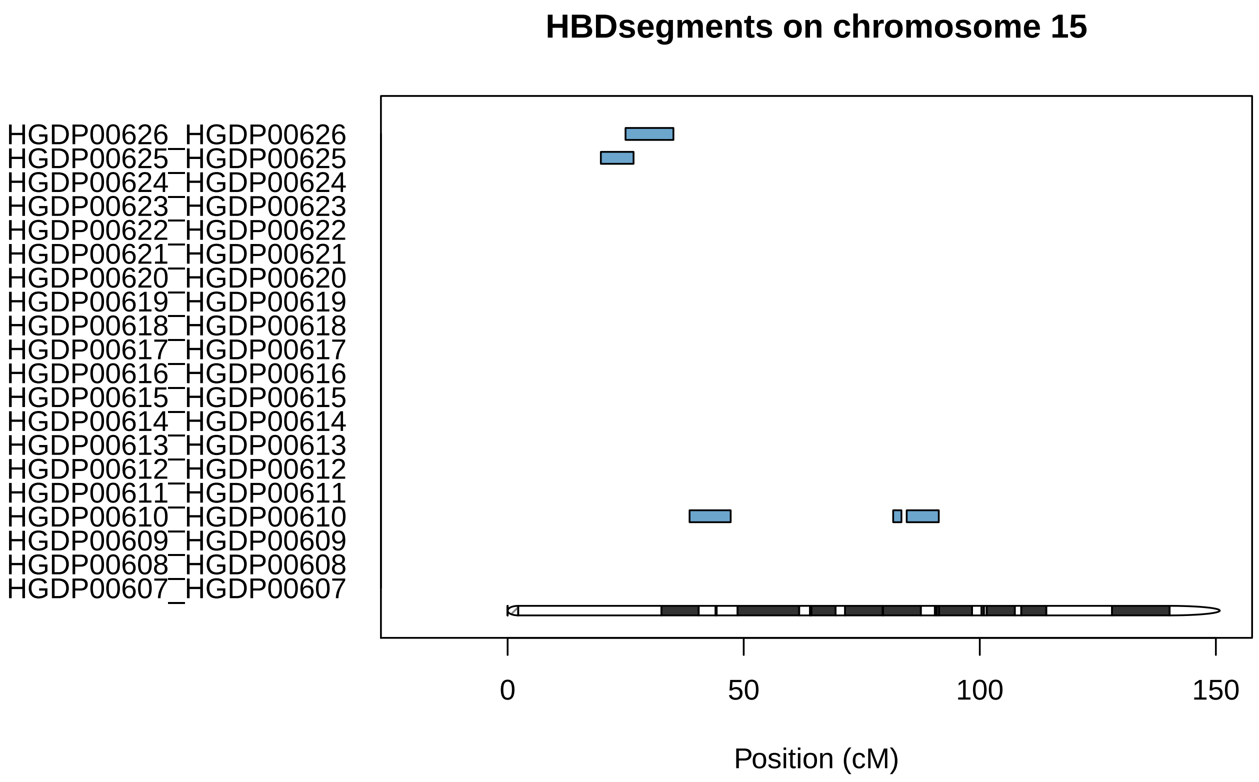
```r
HFLODplotChr(F2, chr=15, MA=FALSE)
```



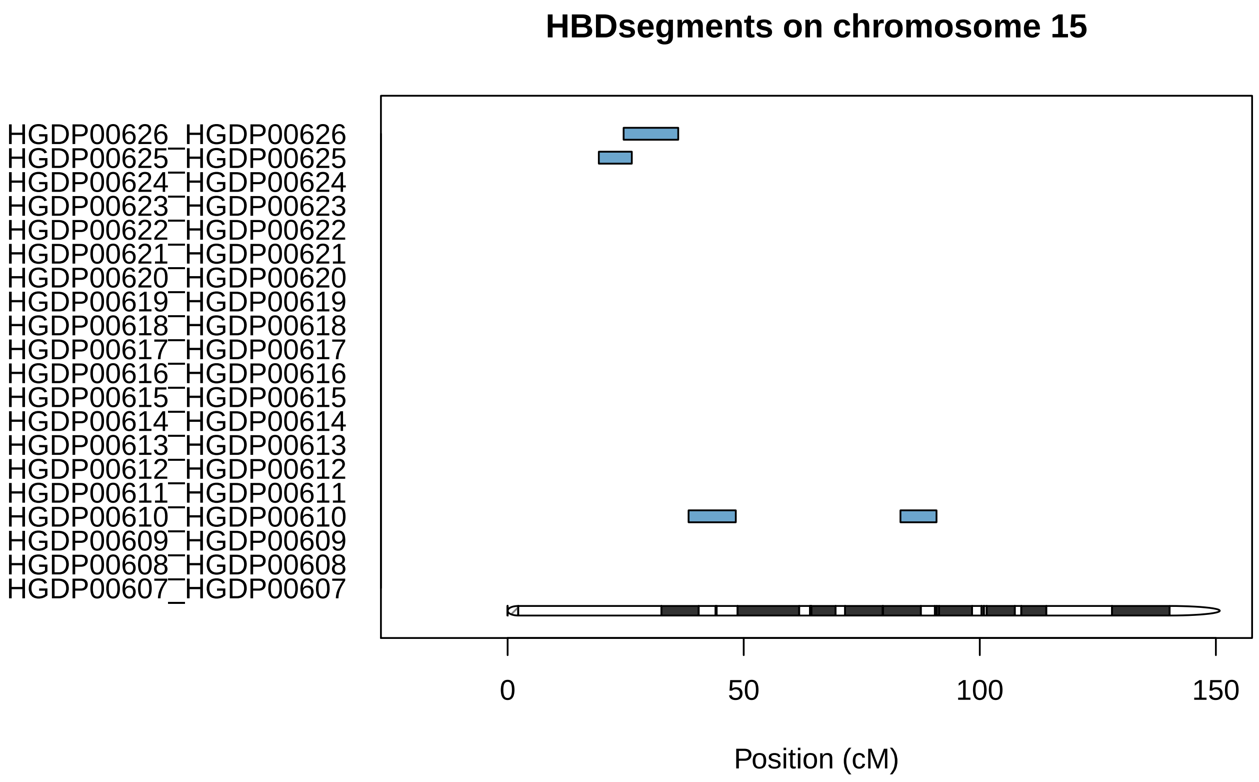
4.2 HBD plots

The HBD plots allow to visualize the positions of the HBD segments on the genome, contained in `F1@HBDsegments`. You can plot multiple individuals for one chromosome (by default, all individuals with f significantly positive are plotted), as illustrated below with `F1` and `F2`:

```r
HBDplotChr(F1, chr=15)
```

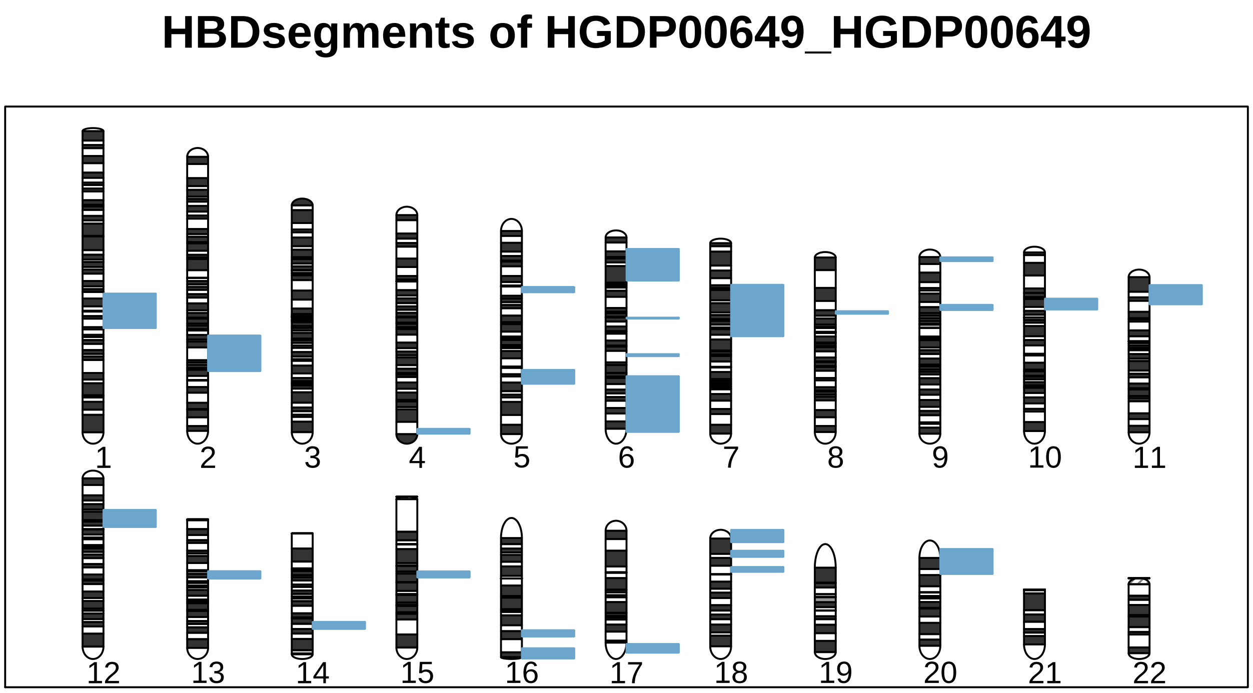


```
HBDplotChr(F2, chr=15)
```

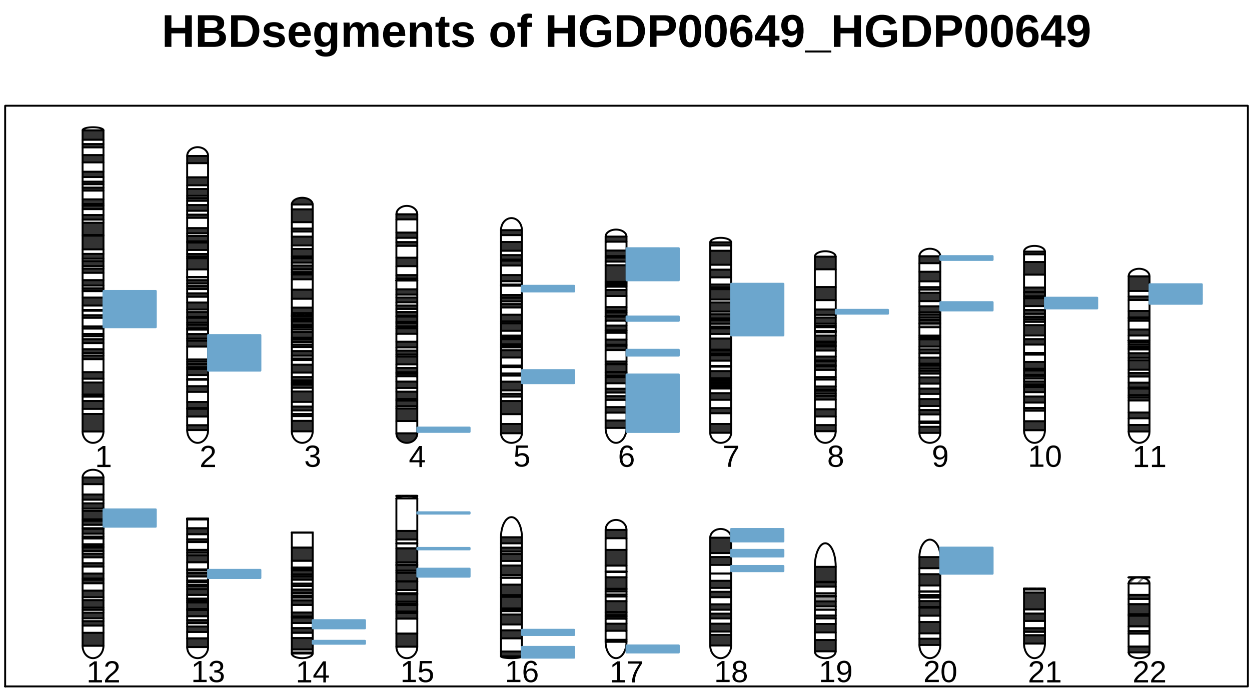


It is also possible to look at all HBD segments of a given individual (again we show the results for `F1` and `F2`):

```r
HBDplotId(F1, individual.id = "HGDP00649", family.id = "HGDP00649")
```



```r
HBDplotId(F2, individual.id = "HGDP00649", family.id = "HGDP00649")
```



5 Step by step usage of the package Fantasio

5.1 Hotspots

5.1.1 Creation of the segments list

We will now create segments, which will be use to create the submaps later, further explication below, for now use this command :

```
s1 <- segmentsListByHotspots(x.be)
```

```
## Using hotspots from  hotspot_hg19
## Gathering all hotspots for the genome : ......................
## Gathering all the genome's markers : ......................
## Finding which markers are between two hotspots : ......................
```

This function creates a list of chromosomes, in each, you have a list of several segments created thanks to the hotspots file given in argument (files are given with the package), in each segments you have SNPs index.

You can watch a summary of what was done with :

```
segmentsListSummary(s1)
```

```
##    chromosome number_of_segments number_of_markers
## 1           1                904             48532
## 2           2                895             52722
## 3           3                750             43507
## 4           4                702             39040
## 5           5                721             39933
## 6           6                719             42184
## 7           7                564             34754
## 8           8                589             36417
## 9           9                563             30276
## 10         10                669             33427
## 11         11                537             31255
## 12         12                601             31039
## 13         13                475             24595
## 14         14                401             20926
## 15         15                383             19098
## 16         16                429             19028
## 17         17                374             16052
## 18         18                442             19496
## 19         19                217             10397
## 20         20                384             16228
## 21         21                225              9301
## 22         22                207              9379
```

This function creates a dataframe with three colums :

- chromosome
- number_of_segments
- number_of_marker

5.1.2 Creation of the submaps and computation

We will now head toward the creation of submaps using the following commands :

```
F1 <- makeAtlasByHotspots(x.be, 5, s1)
F1 <- festim(F1)
F1 <- setSummary(F1, list.id = "all")
```

For the sake of clarity we have only created 5 submaps, but generally we do 100.

This function will creates 5 submaps, all the parameters can be modified (use args(makeAtlasByHotspots) for more informations).

The variable submaps becomes an submapsList object, you can watch the different elements of it with :

```
str(F1) #careful it can become huge depending on your data sizes
```

5.1.3 Descrition of the object F1

This object contains all the results of the different computations executed during the process of creating n submaps. Here is a complete description of each structure in this object :

- segments_list : the object Segments created previously (s1)

```
str(F1@segments_list) #careful it can become huge depending on your data sizes
```

- submaps_list : the list of all the submaps created during the process. Each element of the list is a S4 object. Depending on the method you used the object can be either a `snsp.matrix` or a `HostspotsMatrix` (here we use the hotspots method). Each submaps contains 15 slots :
  - submap : the index of each marker picked (index from the bed.matrix object)
  - ncol : the total number of marker picked
  - nrow : the number of individuals
  - ped : a dataframe with all the individuals' genotypes
  - map : a dataframe with all the SNP information
  - epsilon : genotyping error rate
  - delta.dist : distance between each marker in cM/bp
  - log.emiss : log of all the emission probabilities of the hidden Markov model
  - a : a matrix with all the a's estimation
  - f : a matrix with all the f's estimation
  - likelihood0 : a matrix with all the likehood under the null hypothesis ($f = 0$)
  - likelihood1 : a matrix with all the likehood under the inbred hypothesis ($f = 1$)
  - p.lrt : p value of the likelihood ratio test
  - HBD.prob : a matrix with all the HBD probabilities computed for each individual
  - FLOD : a matrix with all the FLOD score computed

```
str(F1@submaps_list)
```

- likelihood_summary : a dataframe with all the likelihood0 and likelihood1 computed over the submaps.

```
str(F1@likelihood_summary)
```

- estimation_summary : a dataframe with all the a and f computed over the submaps

```
str(F1@estimation_summary)
```

- marker_summary : a dataframe, which gives the number of markers and the number of times it has been picked,
  - number_of_time_picked
  - number_of_marker

```
str(F1@marker_summary)
```

- submaps_summary : a dataframe which gives several informations about the a and f computed over the submaps. The dataframe contains 13 columns:
  - FID: family identifier
  - IID: individual identifier
  - STATUS: status (1 non-affected, 2 affected, 0 unknown)
  - SUBMAPS: number of submaps used
  - QUALITY: percentage of valid submaps (i.e. submaps with a < 1)
  - F_MIN: minimum f on valid submaps
  - F_MAX: maximum f on valid submaps
  - F_MEAN: mean f on valid submaps
  - F_MEDIAN: median f on valid submaps (recommended to estimate f)
  - A_MEDIAN: median a on valid submaps (recommended to estimate a)
  - pLRT_MEDIAN: median p-value of LRT tests on valid submaps
  - INBRED: a flag indicating if the individual is inbred (pLRT_MEDIAN < 0.05) or not
  - pLRT_<0.05: number of valid submaps with a LRT (likelihood ratio test) having a p-value below 0.05

```
head(F1@submap_summary)
```

```
##         FID       IID STATUS SUBMAPS QUALITY      F_MIN      F_MAX
## 1 HGDP00607 HGDP00607      1   5 / 5     100 0.01918105 0.02129699
## 2 HGDP00608 HGDP00608      1   5 / 5     100 0.03719914 0.04073571
## 3 HGDP00609 HGDP00609      1   5 / 5     100 0.03877422 0.04605564
## 4 HGDP00610 HGDP00610      1   5 / 5     100 0.04800044 0.05806033
## 5 HGDP00611 HGDP00611      1   0 / 5      NA         NA         NA
## 6 HGDP00612 HGDP00612      1   5 / 5     100 0.05513765 0.07051555
##       F_MEAN   F_MEDIAN   A_MEDIAN  pLRT_MEDIAN INBRED pLRT_inf_0.05
## 1 0.02030402 0.02035782 0.12747229 6.906977e-23   TRUE             5
## 2 0.03879327 0.03864001 0.06919637 1.660688e-58   TRUE             5
## 3 0.04332706 0.04309106 0.12445629 1.981924e-51   TRUE             5
## 4 0.05247901 0.05214597 0.14542570 2.853227e-65   TRUE             5
## 5         NA         NA         NA           NA     NA            NA
## 6 0.05985072 0.05592216 0.31530679 4.907440e-41   TRUE             5
```

- bySegments : a boolean indicating whether the creation of summary statistics for HBD and FLOD has to be made by segments or not. By default, it is FALSE. The description below of HBD_recap and FLOD_recap depends on this choice.

- HBD_recap : a dataframe with individuals in row and marker positions in column. It contains the mean of the HBD probability at a marker over the submaps for each individual. If the marker only appears in one submap (no mean needed), then it is just the HBD proba for that marker in that submap.

```
F1@HBD_recap[1:10, 1:10] # an individual * marker matrix
```

- FLOD_recap : a dataframe like above. It contains the mean of the FLOD score at a marker over the submaps for each individual. Again if a marker only appears in one submap, no mean is needed.

```
F1@FLOD_recap[1:10, 1:10] # an individual * marker matrix
```

- HBDsegments : a list of dataframe, each datafram is for an individual and it contains a list of HBDsegments (start and end positions). By default, a region is HBD if it contains at least 5 markers (n.consecutive.marker=5) with a HBD probability larger than 0.5 (threshold=0.5).

```
str(F1@HBDsegments[[1]])
```

- HFLOD : a dataframe with the value of HFLOD scores for every markers that appeared at least once in a submap. It uses the formula (2) with FLOD_recap to obtain HFLOD.

```
str(F1@HFLOD)
```

- bedmatrix : the bedmatrix object

```
str(F1@bedmatrix)
```

- unit : the unit of the marker (cM or Bp).

5.2 Hotspots by segments

We implemented a second inner method for the "Hotspots" method. The only paramater that changes is `recap.by.segments`, it is put to TRUE.

In the default "Hotspots" method the HBD probabilities and FLOD scores are computed for each marker (see above).

With the "Hotspots by segment" method, HBD probabilities and FLOD scores correspond to the mean of HBD probabilities and FLOD score of all the markers of a segment that have been selected in one of the sumaps. There is hence only one value for a segment delimited by hotspots. So we also recommend to set n.consecutive.marker to 1. Since results over multiple markers have already been averaged.

5.2.1 Creation of the segments list

We use the same segment list that is used before (s1).

5.2.2 Creation of the submaps and computation

As said before the only argument that changes is "recap.by.segments", it is put to TRUE. And we recommend to adjust "n.consecutive.marker".

```
F2 <- makeAtlasByHotspots(x.be, 5, s1)
F2 <- festim(F2)
F2 <- setSummary(F2, recap.by.segments = TRUE, n.consecutive.marker = 1, list.id = "all")
```

5.3 Distance

5.3.1 Creation of the segments list

We will now create segments, which will be used to create the submaps later :

```
s3 <- segmentsListByDistance(x.be)
```

```
## Finding segments for the genome : .......................
## Finding which markers are between two segments: .......................
## Finding mini segments .......................
```

This function creates segments based on the markers available in the data and the gaps between them. The value of the gap is imposed by the minimun step required between markers. If 2 adjacent markers have an inter-distance larger than the step, then there is a gap between them. The first marker of the pair is the end of a segment and the second marker is the start of a new segment. The function creates an object which will contain three slots :

- gap : the value of the minimal fix step required between sampled markers of a submap
- unit : the unit of the markers ("cM" or "Bp") snpsSegments : for each chromosome, the

- list of segments; each segment being a list of SNP indexes

You can watch a summary of what was done with :

```
segmentsListSummary(s3)
```

This function creates a dataframe with three colums :

- chromosome
- number_of_segments
- number_of_marker

5.3.2 Creation of the submaps and computation

We will now head toward the creation of submaps using the following commands :

```
F3 <- makeAtlasByDistance(x.be, 5, s3)
F3 <- festim(F3)
F3 <- setSummary(F3, list.id = "all")
```

The variable submaps becomes an submapsList object, you can watch the different elements of it with :

```
str(F3) #careful it can become huge depending on your data sizes
```

6 Parallelism with the package

We implemented a paralellism method to make the creation of the submaps more efficient. We paralellized the creation of the submaps, that is to say, the selection of markers. Make sure to have an environment that can support the usage of multiple CPU.

Use the `n.cores` argument, i.e the number of CPU that will be used to make the differents submaps in the following functions :

- Fantasio makeAtlasByHotspots makeAtlasByDistance

```
F6 <- Fantasio(bedmatrix=x.be, segments="Hotspots", n=5, verbose=FALSE, n.cores=10)
```