Software Manual

Institute of Bioinformatics, Johannes Kepler University Linz

hapFabia: Identification of rare and short haplotype clusters in large sequencing data

— Manual for the R package —

Sepp Hochreiter

Institute of Bioinformatics, Johannes Kepler University Linz
Altenberger Str. 69, 4040 Linz, Austria
hochreit@bioinf.jku.at

Version 0.90.0, September 1, 2012

Contents

1 Introduction **3**

2 Getting Started **6**
2.1 Typical Analysis Pipeline 6
2.2 Examples 8

3 hapFabia Method **11**

4 Tools to Analyze `fabia` Results **17**

1 Introduction

This package `hapFabia` provides software for the method HapFABIA which identifies rare and short haplotype clusters in large sequencing data with a focus on rare variants.

Individuals that inherited a particular DNA segment from the same founder constitute a haplotype cluster by sharing minor alleles of variants that tag this segment. Knowledge of haplotype clusters are relevant for genetic and genomic studies and for population genetics where they shed light on the evolutionary history of humans.

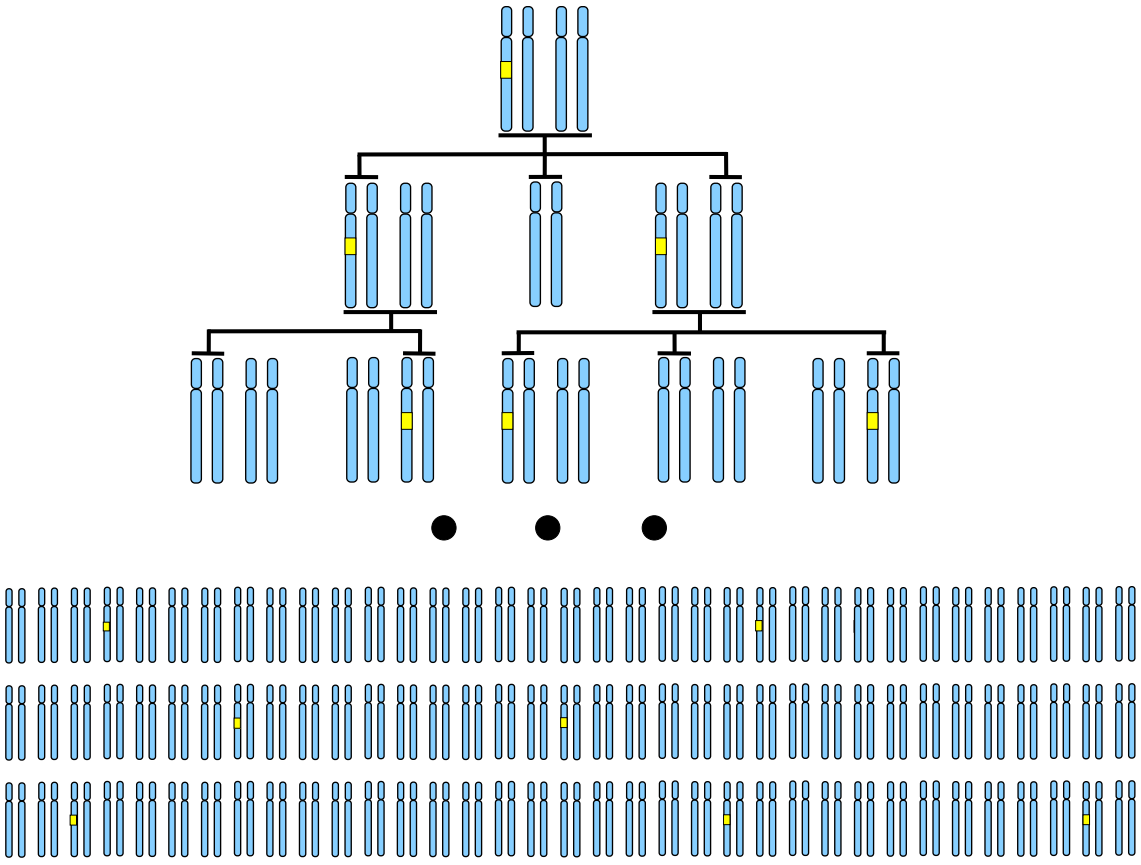


Figure 1: The DNA segment marked in yellow decended from a founder to different individuals which constitute a haplotype cluster by possessing the DNA segment.

Rare haplotype clusters are identified by biclustering that is clustering individuals only on a subset of single nucleotide variants (SNVs), the tagSNVs (these SNVs tag the haplotype cluster). Biclustering combines linkage disequilibrium (LD) information across individuals via correlations of SNVs and identity by descent (IBD) information along the chromosome via contiguous identical alleles. The LD information enters HapFABIA through FABIA implemented in the package `fabia` which detects correlations of multiple SNVs by a factor analysis model. The IBD information enters HapFABIA through identification of haplotype clusters as local agglomerations of correlated SNVs.

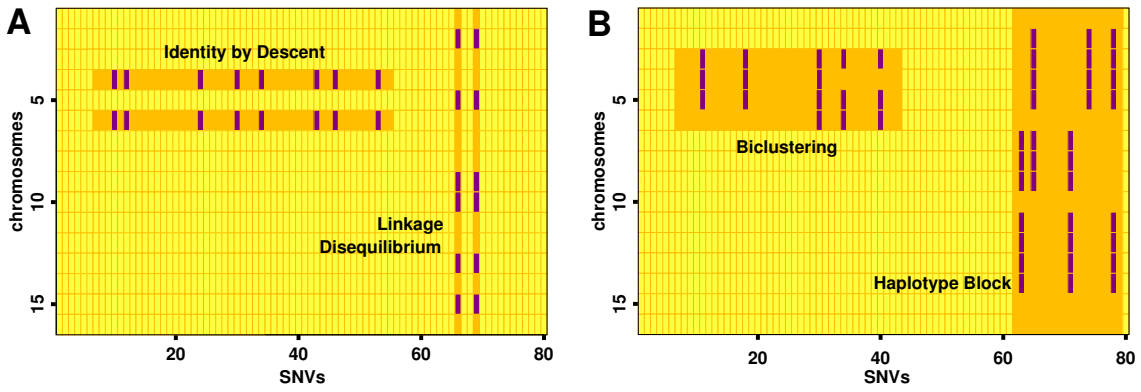


Figure 2: Different approaches to detect haplotype clusters. The y-axis enumerates individuals and the x-axis SNVs. Yellow and violet bars indicate major and minor alleles, respectively. Haplotype cluster segements are marked in gold. Panel **A**: Linkage disequilibrium (LD) is computed for two SNVs across all samples. Identity by descent (IBD) is detected for two individuals for a long haplotype region. Panel **B**: A haplotype block is depicted as a region of increased LD comprising different haplotypes. Biclustering clusters individuals together that have the same founder DNA segment indicated by identical minor alleles of tagSNVs.

Fig. 2 shows in panel A linkage disequilibrium (LD) and identity by descent (IBD) and in panel B a haplotype block and a bicluster. Individuals that belong to a haplotype cluster are similar to each other because they share the same minor alleles in an inherited founder DNA segment. This similarity define a bicluster according to the FABIA model Hochreiter *et al.* (2010). Consequently, a haplotype cluster constitutes a bicluster to which SNVs belong if they tag the haplotype cluster and to which individuals belong if they possess the founder DNA segment (see the bicluster in panel B of Fig. 2).

The multiplicative biclusters of FABIA are able to represent homozygous regions that is two occurrences of a haplotype in one diploid individual via their factors. Overlapping haplotype clusters in one diploid individual are represented through the additivity of biclusters in the FABIA model. Examples of haplotype clusters found by hapFabia are given in Fig. 3 and Fig. 4.

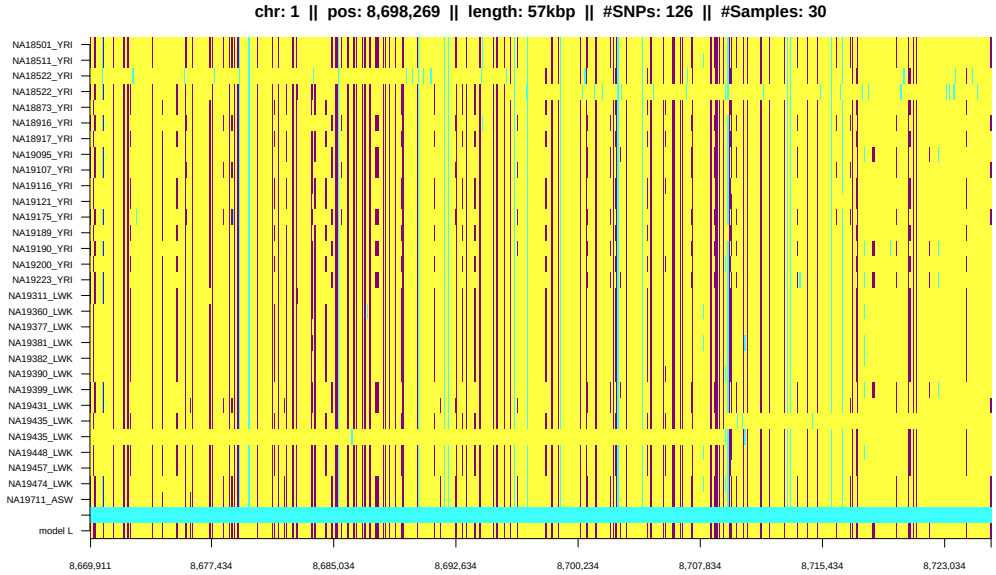


Figure 3: Example of a haplotype cluster. The y-axis gives the chromosomes and the x-axis consecutive SNVs/Indels/SVs. Yellow indicates major alleles, violet minor alleles of tagSNVs, and blue minor alleles of other SNVs. "model L" indicates tagSNVs identified by hapFabia in violet.

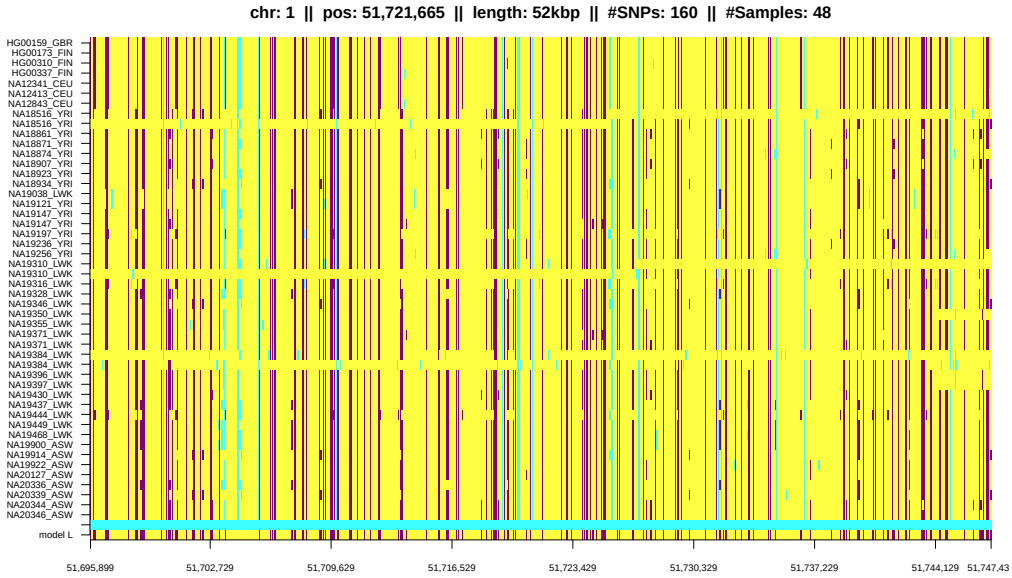


Figure 4: Another example of a haplotype cluster. See Fig. 3 for a description.

2 Getting Started

2.1 Typical Analysis Pipeline

First, we briefly describe a typical analysis pipeline. Assume we have the genotype data of chromosome 1 in the file `filename.vcf.gz` in compressed `vcf` format. To prepare the data for `hapFabia` we have to perform preprocessing steps. First `filename.vcf.gz` must be 1. uncompressed, then 2. converted to the sparse matrix format, and then 3. split into segments. The following command line commands perform these steps:

1. `gunzip filename.vcf.gz`
2. `./vcftoFABIA filename ./`
3. `./split_sparse_matrix filename _mat.txt 10000 5000 1`

The command line tools for step 2. and 3. are provided by the package `hapFabia` in `inst/commandline/arch/`. However step 2. and 3. can be performed in R as well (see below). The commandline parameters for `vcftoFABIA` are 1) filename without `.vcf` and 2) path to the file. The commandline parameters for `split_sparse_matrix` are 1) filename without `.vcf` 2) extension (default`_mat.txt`) 3) segment length 4) shift size 5) indicator whether annotation is present (is generated by `vcftoFABIA` as default). The data is split into segments of 10,000 SNVs where the distance between adjacent segments is 5,000 thus they overlap by 5,000 SNVs.

After providing the file `filename.vcf` the following steps constitute a typical analysis pipeline in R :

```
R> #####define segments, overlap, filename #######
R> shiftSize <- 5000
R> segmentSize <- 10000
R> fileName="filename" # without type
R>
R> #####load libraries#######
R> library(hapFabia)
R> library(fabia)
R>
R> #####convert from .vcf to _mat.txt: step 2. above#######
R> vcftoFABIA(fileName=fileName)
R>
R> #####split/ generate segments: step 3. above#######
R> split_sparse_matrix(fileName=fileName,segmentSize=segmentSize,
+  shiftSize=shiftSize,annotation=TRUE)
R>
R> #####compute how many segments we have#######
R> ina <- as.numeric(readLines(paste(fileName,"_mat.txt",sep=""),n=2))
R> snps <- ina[2]
R> over <- segmentSize%/%shiftSize
R> N1 <- snps%/%shiftSize
```

```
R> endRunA <- (N1-over+2)
R>
R> #####analyze each segment#######
R> #####may be done by parallel runs#######
R> iterateSegments(startRun=1,endRun=endRunA,shift=shiftSize,
+  segmentSize=segmentSize,fileName=fileName,samples=0,
+  upperBP=0.05,p=10,iter=40,alpha=0.03,cyc=50,IBDlength=50,
+  Lt = 0.1,Zt = 0.2,thresCount=1e-5,minSNPsFactor=3/4,
+  pMAF=0.035,haplotypes=TRUE)
R>
R> #####identify duplicates#######
R> identifyDuplicates(fileName=fileName,startRun=1,endRun=endRunA,
+  shift=shiftSize,segmentSize=segmentSize)
R>
R> #####analyze results; parallel#######
R> anaRes <- analyzeIBDs(fileName=fileName,startRun=1,endRun=endRunA,
+  shift=shiftSize,segmentSize=segmentSize)
R> print("Number haplotype clusters:")
R> print(anaRes$noIBD)
R> print("Haplotye cluster length:")
R> print(anaRes$avibdLengthS)
R> print("Number of individuals belonging to a haplotye cluster:")
R> print(anaRes$avnoSampS)
R> print("Number of tagSNVs of a haplotye cluster:")
R> print(anaRes$avnoSnpsS)
R> print("MAF of tagSNVs of a haplotype cluster:")
R> print(anaRes$avnoFreqS)
R> print("MAF within the group of tagSNVs of a haplotye cluster:")
R> print(anaRes$avnoGroupFreqS)
R> print("Number of changes between major and minor allele frequency:")
R> print(anaRes$avnosnpChangeS)
R> print("Number of tagSNVs per individual:")
R> print(anaRes$avnosnpsPerSampleS)
R> print("Number of individuals having a tagSNV:")
R> print(anaRes$avnosamplePerSnpS)
R>
R> #####load result for segment 50#######
R> posAll <- 50 # (50-1)*5000 = 245000: segment 245000 to 255000
R> start <- (posAll-1)*shiftSize
R> end <- start + segmentSize
R> pRange <- paste("_",format(start,scientific=FALSE),"_",
+  format(end,scientific=FALSE),sep="")
R> load(file=paste(fileName,pRange,"_resAnno",".Rda",sep=""))
R> IBD <- resHapFabia$mergedIBD # $
R>
R> #####plot result for segment 50#######
```

```
R> plotIBDs(IBD,filename=paste(fileName,pRange,"_mat",sep=""))
R>    ##attention: filename without type ".txt"
R>
R> #####plot specific haplotype clusters (the first) in segment 50#######
R> plotOneIBD(IBD,filename=paste(fileName,pRange,"_mat",sep=""),
+  ibdC=1)
R>    ##attention: filename without type ".txt"
```

First the packages `hapFabia` and `fabia` are loaded. Then `filename.vcf` is converted by `vcftoFABIA` to sparse matrix format `filename_mat.txt` where the annotation file `filename_annot.txt` and the sample labels `filename_samples.txt` are generated. The function `split_sparse_matrix` splits chromosome 1 into segments of length 10,000 SNVs with a distance of 5,000 SNVs between the segments. This results in 640 segments because we had more than 3,200,000 and less than 3,210,000 SNVs in the genotype data. Using the function `iterateSegments` haplotype clusters are identified in these segments and the results stored in a EXCEL like `.csv` format and as R data object. The function `identifyDuplicates` marks and memorizes duplicates of haplotype custers which occur because the segments overlap. Next the function `analyzeIBDs` analyzes the results where duplicates as marked in previous step are not considered. Results are listed by `anaRes`.

The next example shows how to view all haplotype clusters of a segment, for which we chose segment 50 which corresponds to chromosome 1 interval from 245,000 to 255,000 ($(50-1)*5000 = 245000$). Then we plot a specific haplotype cluster, in this case the first (`ibdC=1`), which can also be used to store a `.pdf` or a `.fig` for editing with Xfig. Examples of this plot function are given in Fig. 3 and Fig. 4.

An R source file `pipeline.R` of above pipeline can be created and executed as follows:

```
R> makePipelineFile("filename",shiftSize=5000,segmentSize=10000)
R>
R> source("pipeline.R")
```

NOTE: sourcing may take a while for large datasets.

2.2 Examples

Next we show an example how to use `hapFabia`.

First the packages are loaded and then the data `simu`. The data `simu` is written into three files: `dataSim1fabia_samples.txt` (sample names), `dataSim1fabia_annot.txt` (SNV annotation information), and `dataSim1fabia_mat.txt` (the data in sparse matrix format). These are files which are in the standard pipeline produced by `vcftoFABIA` and by `split_sparse_matrix`.

```
> library(hapFabia)
> library(fabia)
> data(simu)
> namesL <- simu[[1]]
> haploN <- simu[[2]]
```

```
> snps <- simu[[3]]
> annot <- simu[[4]]
> alleleImp <- simu[[5]]
> write.table(namesL,file="dataSim1fabia_samples.txt",
+     quote = FALSE,row.names = FALSE,col.names = FALSE)
> write(as.integer(haploN),file="dataSim1fabia_annot.txt",
+     ncolumns=100)
> write(as.integer(snps),file="dataSim1fabia_annot.txt",
+     append=TRUE,ncolumns=100)
> write.table(annot,file="dataSim1fabia_annot.txt", sep = " ",
+     quote = FALSE,row.names = FALSE,col.names = FALSE,append=TRUE)
> write(as.integer(haploN),file="dataSim1fabia_mat.txt",ncolumns=100)
> write(as.integer(snps),file="dataSim1fabia_mat.txt",
+     append=TRUE,ncolumns=100)
> for (i in 1:haploN) {
+
+   a1 <- which(alleleImp[i,]>0.01)
+
+   al <- length(a1)
+   b1 <- alleleImp[i,a1]
+
+   a1 <- a1 - 1
+   dim(a1) <- c(1,al)
+   b1 <- format(as.double(b1),nsmall=1)
+   dim(b1) <- c(1,al)
+
+   write.table(al,file="dataSim1fabia_mat.txt", sep = " ",
+      quote = FALSE,row.names = FALSE,col.names = FALSE,append=TRUE)
+   write.table(a1,file="dataSim1fabia_mat.txt", sep = " ",
+      quote = FALSE,row.names = FALSE,col.names = FALSE,append=TRUE)
+   write.table(b1,file="dataSim1fabia_mat.txt", sep = " ",
+      quote = FALSE,row.names = FALSE,col.names = FALSE,append=TRUE)
+
+ }
```

Now the haplotype clusters can be extracted from the data:

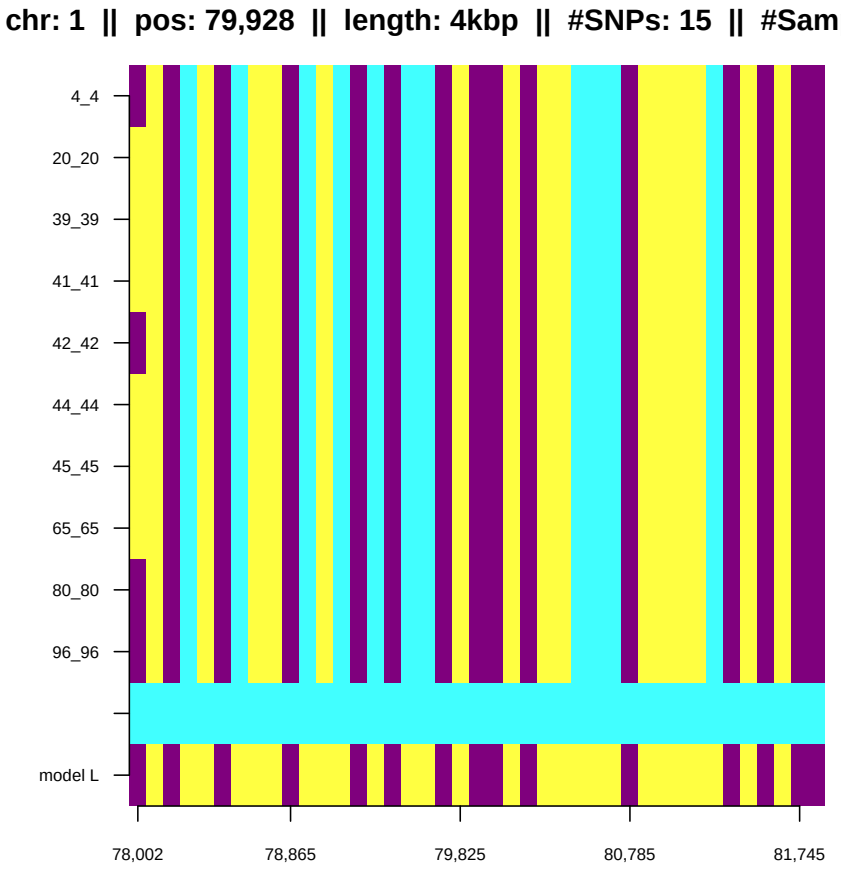
```
> hapRes <- hapFabia(fileName="dataSim1fabia",prefixPath="",
+      upperBP=0.15,p=10,iter=1,IBDlength=10,pMAF=0.1)
```

To view the results the first haplotype cluster is plotted:

```
> mergedIBD <- hapRes$mergedIBD # $

> plotOneIBD(mergedIBD,filename="dataSim1fabia_mat",ibdC=1)

Using 10 samples!
```



Finally we remove the temporary data files:

```
> unlink("dataSim1fabia_mat.txt")
> unlink("dataSim1fabia_annot.txt")
> unlink("dataSim1fabia_samples.txt")
> unlink("dataSim1fabia_resIBD.Rda")
> unlink("dataSim1fabia_mat_res_L.txt")
> unlink("dataSim1fabia_mat_res_lapla.txt")
> unlink("dataSim1fabia_mat_res_Psi.txt")
> unlink("dataSim1fabia_mat_res_Z.txt")
> unlink("dataSim1fabia_mat_res_Z_full.txt")
```

Here another example with random data:

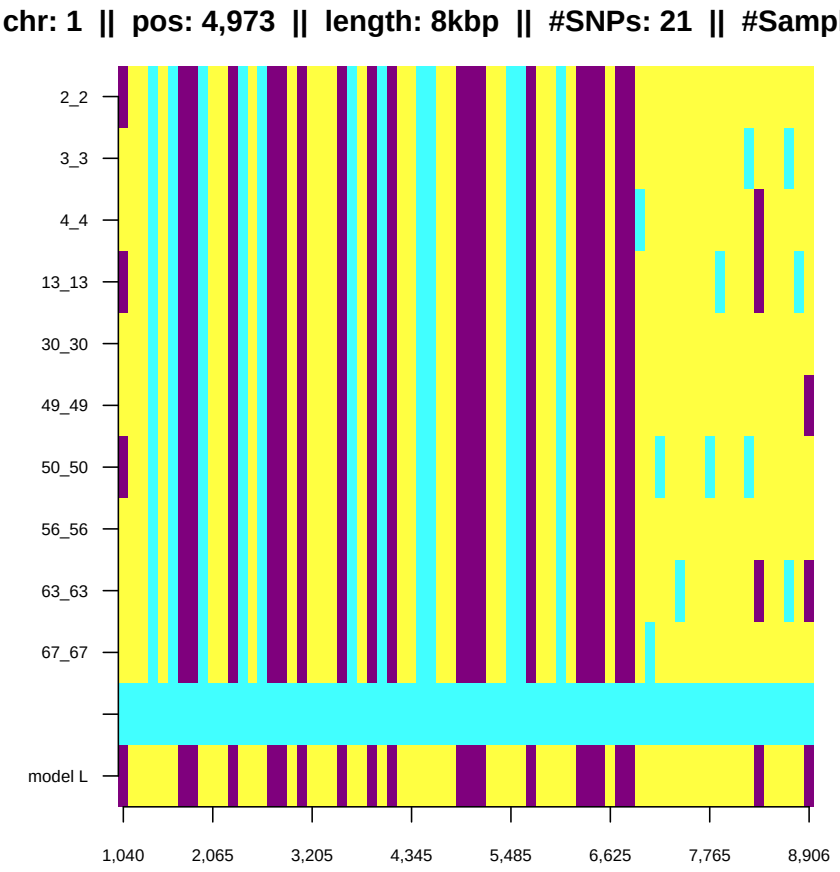
```
> simulateForFabia(minruns=2,maxruns=2)

> hapRes <- hapFabia(fileName="dataSim2fabia",prefixPath="",
+       upperBP=0.15,p=10,iter=1,IBDlength=10,pMAF=0.1)

> mergedIBD <- hapRes$mergedIBD # $
```

```
> plotOneIBD(mergedIBD,filename="dataSim2fabia_mat",ibdC=1)

Using 10 samples!
```




```
> unlink("dataSim2fabia_mat.txt")
> unlink("dataSim2fabia_annot.txt")
> unlink("dataSim2fabia_samples.txt")
> unlink("dataSim2fabia_resIBD.Rda")
> unlink("dataSim2fabia_mat_res_L.txt")
> unlink("dataSim2fabia_mat_res_lapla.txt")
> unlink("dataSim2fabia_mat_res_Psi.txt")
> unlink("dataSim2fabia_mat_res_Z.txt")
> unlink("dataSim2fabia_mat_res_Z_full.txt")
```

3 hapFabia Method

We introduce our novel method HapFABIA for extracting rare and short haplotype clusters from large sequencing data. HapFABIA first applies FABIA biclustering to phased or unphased genotype data. The first step, biclustering, extracts similarities between individuals based on a subset

of SNVs but does not consider that haplotype clusters consist of contiguous nucleotides. In the second step, HapFABIA therefore extracts haplotype clusters from FABIA models by considering local tagSNV accumulations and pruning spurious correlations of SNVs with a haplotype cluster. These two HapFABIA steps are described in the next two subsections.

FABIA for genotype data

We propose identifying similarities between individuals by biclustering. A bicluster corresponds to a haplotype cluster where individuals are similar to each other by virtue of being identical at minor alleles of tagSNVs. Panel B in Fig. 2 shows such a bicluster.

For biclustering we use the "Factor Analysis for Bicluster Acquisition" (FABIA) biclustering model Hochreiter *et al.* (2010). In contrast to other biclustering methods such as BIMAX Prelic *et al.* (2006) and QUBIC Li *et al.* (2009), FABIA can represent homozygous regions and overlapping haplotype clusters because of its multiplicative bicluster model and a data model that is additive in its biclusters (see below). FABIA can be applied to discrete genotype data but also to real values that give the minor allele likelihood or the minor allele dosage. We use FABIA not only because it is well suited for genotyping data, but also because it outperformed other biclustering methods in extensive comparisons on different data sets Hochreiter *et al.* (2010).

FABIA models genotype data by haplotype clusters

FABIA describes genotype data $\boldsymbol{X}$ by an outer product $\boldsymbol{z}\,\boldsymbol{\lambda}^T$ of two vectors $\boldsymbol{\lambda}$ and $\boldsymbol{z}$. The vector $\boldsymbol{\lambda}$ corresponds to a haplotype cluster that contains ones for haplotype cluster tagSNVs and zeros otherwise. Vector $\boldsymbol{z}$ indicates the number of segments of an individual that belong to the haplotype cluster (multiploidy). FABIA can represent a homozygous region, that is, two occurrences of a haplotype in one diploid individual, by $z_i = 2$. Fig. **??** visualizes this representation of a genotype matrix by an outer product for one haplotype cluster.

A diploid individual may participate at two haplotype clusters at a particular locus. In this case genotyping sums the occurrences of minor alleles, which is reflected by the additive FABIA model. If we assume genotyping errors, the FABIA model for genotype data $\boldsymbol{X}$ is

$$\boldsymbol{X} \;=\; \sum_{i=1}^{p} \boldsymbol{z}_i\, \boldsymbol{\lambda}_i^T \;+\; \boldsymbol{\Upsilon} \;=\; \boldsymbol{Z}\,\boldsymbol{\Lambda} \;+\; \boldsymbol{\Upsilon}\,, \tag{1}$$

where $\boldsymbol{X} \in \mathbb{R}^{l \times n}$ is the genotyping data; $\boldsymbol{Z} \in \mathbb{R}^{l \times p}$ is the matrix that gives for each individual the number of segments that belong to a haplotype cluster; $\boldsymbol{\Lambda} \in \mathbb{R}^{p \times n}$ is the haplotype cluster tagSNV matrix; $\boldsymbol{\Upsilon} \in \mathbb{R}^{l \times n}$ is additive noise; n is the number of SNVs; l is the number of individuals (or chromosomes for phased genotypes); p is the number of haplotype clusters; $\boldsymbol{\lambda}_i \in \mathbb{R}^n$ is the ith haplotype cluster tagSNV vector (the ith row of $\boldsymbol{\Lambda}$); and $\boldsymbol{z}_i \in \mathbb{R}^l$ gives for each individual the number of segments that belong to the i-th haplotype cluster (the ith column of $\boldsymbol{Z}$). The additive noise not only covers genotyping errors but also genotypes which cannot be explained by haplotype clusters. Such unexplained genotypes may arise from recently acquired SNVs or haplotype cluster segments that were fragmented by recombination events.

The FABIA model in Eq. (1) allows a generative interpretation by a factor analysis model with p factors:

$$\boldsymbol{x} \;=\; \sum_{i=1}^{p} \boldsymbol{\lambda}_i \, z_i \;+\; \boldsymbol{\epsilon} \;=\; \boldsymbol{\Lambda}^T \, \boldsymbol{z} \;+\; \boldsymbol{\epsilon} \,, \tag{2}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ is an observed genotype (a row of $\boldsymbol{X}$ written as column vector), the vector of factors $\boldsymbol{z} \in \mathbb{R}^p$ gives the number of segments belonging to each haplotype cluster for genotype $\boldsymbol{x}$, z_i is the number of segments that belong to the i-th haplotype cluster (a component of both $\boldsymbol{z}_i$ and $\boldsymbol{z}$), and $\boldsymbol{\epsilon} \in \mathbb{R}^n$ is the additive noise (a row of $\boldsymbol{\Upsilon}$ written as column vector).

As illustrated in Fig. **??**, both the vector $\boldsymbol{z}_i$ and the vector $\boldsymbol{\lambda}_i$ of haplotype cluster tagSNVs should be sparse. Sparse $\boldsymbol{z}_i$ means that only few individuals belong to the haplotype cluster, that is, the haplotype cluster is rare. Sparse $\boldsymbol{\lambda}_i$ means that only few SNVs are tagSNVs, which implies short haplotype clusters. Sparse $\boldsymbol{z}_i$ can be achieved if all components z_i are sparse, that is, if the vector of factors $\boldsymbol{z}$ is sparse. Note, that $\boldsymbol{z}_i \in \mathbb{R}^l$ in Eq. (1) gives for each of the l individuals the number of segments that belong to the i-th haplotype cluster, while $\boldsymbol{z} \in \mathbb{R}^p$ gives the number of segments belonging to each of the p haplotype clusters for one genotype $\boldsymbol{x}$. In contrast to standard factor analysis, FABIA's model selection is tailored to sparse factors and sparse loadings, which are essential for haplotype cluster detection. Sparseness in the FABIA model is obtained by a component-wise independent Laplace distribution both for the prior on the parameters $\boldsymbol{\lambda}_i$ and the distribution of the factors $\boldsymbol{z}$ **?**:

$$p(\boldsymbol{z}) \;=\; \left(\tfrac{1}{\sqrt{2}}\right)^p \prod_{i=1}^{p} e^{-\sqrt{2}\,|z_i|} \tag{3}$$

$$p(\boldsymbol{\lambda}_i) \;=\; \left(\tfrac{1}{\sqrt{2}}\right)^n \prod_{k=1}^{n} e^{-\sqrt{2}\,|\lambda_{ki}|} \tag{4}$$

The Laplace distribution of the factors leads to the analytically intractable likelihood:

$$p(\boldsymbol{x} \mid \boldsymbol{\Lambda}, \boldsymbol{\Psi}) \;=\; \int p(\boldsymbol{x} \mid \boldsymbol{z}, \boldsymbol{\Lambda}, \boldsymbol{\Psi}) \, p(\boldsymbol{z}) \, d\boldsymbol{z} \,. \tag{5}$$

Therefore, FABIA model selection is performed by means of variational expectation maximization, which is a variational optimization in the expectation maximization (EM) framework to maximize the posterior of the parameters Girolami (2001); **?**); Hochreiter *et al.* (2010); Clevert *et al.* (2011); Klambauer *et al.* (2012). The idea of the variational approach is to express the prior $p(\boldsymbol{z})$ by the maximum

$$p(\boldsymbol{z}) \;=\; \max_{\boldsymbol{\xi}} p(\boldsymbol{z} \mid \boldsymbol{\xi}) \tag{6}$$

over a model family $p(\boldsymbol{z} \mid \boldsymbol{\xi})$ that is parametrized by the variational parameter $\boldsymbol{\xi}$ or by scale mixtures

$$p(\boldsymbol{z}) \;=\; \int p(\boldsymbol{z} \mid \boldsymbol{\xi}) \, d\mu(\boldsymbol{\xi}) \,. \tag{7}$$

A Laplace distribution can be expressed exactly by the maximum of a Gaussian family or by Gaussian scale mixtures Girolami (2001); **?**. Therefore for each $\boldsymbol{x}$, the maximum $\hat{\boldsymbol{\xi}}$ of the variational parameter $\boldsymbol{\xi}$ allows to represent the Laplacian prior by a Gaussian:

$$\hat{\boldsymbol{\xi}} \;=\; \arg\max_{\boldsymbol{\xi}} p(\boldsymbol{\xi} \mid \boldsymbol{x}) \;. \tag{8}$$

The maximum $\hat{\boldsymbol{\xi}}$ can be computed analytically (see Eq. (11) below) because for each Gaussian the likelihood Eq. (5) can be computed analytically.

If we denote the jth genotype by $\boldsymbol{x}_j \in \mathbb{R}^n$ with corresponding factors $\boldsymbol{z}_j \in \mathbb{R}^p$, then we obtain the following variational E-step Hochreiter *et al.* (2010):

$$\mathrm{E}\big(\boldsymbol{z}_j \mid \boldsymbol{x}_j\big) \;=\; \big(\boldsymbol{\Lambda}^T\, \boldsymbol{\Psi}^{-1}\, \boldsymbol{\Lambda} \;+\; \boldsymbol{\Xi}_j^{-1}\big)^{-1}\, \boldsymbol{\Lambda}^T\, \boldsymbol{\Psi}^{-1}\, \boldsymbol{x}_j \;, \tag{9}$$

$$\begin{aligned} \mathrm{E}\big(\boldsymbol{z}_j\, \boldsymbol{z}_j^T \mid \boldsymbol{x}_j\big) \;&=\; \big(\boldsymbol{\Lambda}^T\, \boldsymbol{\Psi}^{-1}\, \boldsymbol{\Lambda} \;+\; \boldsymbol{\Xi}_j^{-1}\big)^{-1} \;+ \\ &\qquad \mathrm{E}\big(\boldsymbol{z}_j \mid \boldsymbol{x}_j\big)\, \mathrm{E}(\boldsymbol{z}_j \mid \boldsymbol{x}_j)^T \;, \end{aligned} \tag{10}$$

where $\boldsymbol{\Xi}_j$ means $\mathrm{diag}\,(\boldsymbol{\xi}_j)$. The update for the variational parameter $\boldsymbol{\xi}_j$ is

$$\boldsymbol{\xi}_j \;=\; \mathrm{diag}\,\Big(\sqrt{\mathrm{E}(\boldsymbol{z}_j\, \boldsymbol{z}_j^T \mid \boldsymbol{x}_j)}\Big) \;. \tag{11}$$

The variational M-step is Hochreiter *et al.* (2010)

$$\boldsymbol{\Lambda}^{\mathrm{new}} \;=\; \frac{\frac{1}{l}\, \sum_{j=1}^{l} \boldsymbol{x}_j\, \mathrm{E}(\boldsymbol{z}_j \mid \boldsymbol{x}_j)^T \;-\; \frac{\alpha}{l}\, \boldsymbol{\Psi}\, \mathrm{sign}(\boldsymbol{\Lambda})}{\frac{1}{l} \sum_{j=1}^{l} \mathrm{E}(\boldsymbol{z}_j\, \boldsymbol{z}_j^T \mid \boldsymbol{x}_j)} \tag{12}$$

$$\mathrm{diag}\,(\boldsymbol{\Psi}^{\mathrm{new}}) \;=\; \boldsymbol{\Psi}^{\mathrm{EM}} \;+\; \mathrm{diag}\Big(\frac{\alpha}{l}\, \boldsymbol{\Psi}\, \mathrm{sign}(\boldsymbol{\Lambda})(\boldsymbol{\Lambda}^{\mathrm{new}})^T\Big) \;, \tag{13}$$

$$\boldsymbol{\Psi}^{\mathrm{EM}} \;=\; \mathrm{diag}\Big(\frac{1}{l} \sum_{j=1}^{l} \boldsymbol{x}_j\, \boldsymbol{x}_j^T \;-\; \boldsymbol{\Lambda}^{\mathrm{new}} \frac{1}{l} \sum_{j=1}^{l} \mathrm{E}\,(\boldsymbol{z}_j \mid \boldsymbol{x}_j)\; \boldsymbol{x}_j^T\Big). \tag{14}$$

The parameter α controls the degree of sparseness (an expectation of how rare the haplotype clusters are) and can be introduced as a parameter of the Laplacian prior of the factors Hochreiter *et al.* (2010).

Note, that the number of bicluster must not be determined a priori if p is chosen large enough. The sparseness constraint will remove spurious biclusters by setting $\boldsymbol{\lambda}$ to a zero vector. In this way FABIA automatically determines the number of biclusters.

Adaptation of FABIA for haplotype cluster detection

Since an entry in the genotype matrix $\boldsymbol{X}$ reports how often the minor allele is present, FABIA must explain occurrences of minor alleles through haplotype clusters. Because both the counts of minor alleles and the the occurrences of segments in haplotype clusters are non-negative, we modified FABIA to enforce non-negative loadings $\boldsymbol{\lambda}_i$ by projecting negative components of $\boldsymbol{\lambda}_i$ to zero. If both $\boldsymbol{x}_j$ and $\boldsymbol{\Lambda}$ are non-negative, the posterior mean $\mathrm{E}\big(\boldsymbol{z}_j \mid \boldsymbol{x}_j\big)$ of $\boldsymbol{z}_j$ is non-negative, too. Therefore also $\boldsymbol{\Lambda}^{\mathrm{new}}$ is non-negative, because the prior term in the update rule is not allowed to change the sign of the loadings. Therefore it is sufficient to initialize $\boldsymbol{\Lambda}$ by positive values to

enforce non-negative factors and loadings. $\boldsymbol{Z}$ is estimated by the prior mean $\mathrm{E}(\boldsymbol{z}_j \mid \boldsymbol{x}_j)$ and used to identify individuals belonging to a haplotype cluster.

Further we developed a sparse matrix algebra which represents only non-zero values and indices for FABIA biclustering in order to efficiently analyze large genotyping data. In Eq. (9) to Eq. (14) the values of the genotype vectors $\boldsymbol{x}_j$ and the values of the loading vector $\boldsymbol{\lambda}_i$ and the loading matrix $\boldsymbol{\Lambda}$ are sparse. Therefore we introduced a sparse matrix algebra for multiplying both two sparse vectors and a sparse vector by a dense vector.

To further speed up the computation, we developed an iterative version of FABIA. Each FABIA iteration detects p biclusters. These p biclusters are removed from the genotype matrix $\boldsymbol{X}$ before starting the next iteration.

The vectors $\boldsymbol{\lambda}_i$ and $\boldsymbol{z}_i$ acquire a new interpretation when detecting rare haplotype clusters. Components of $\boldsymbol{\lambda}_i$ correspond to tagSNVs and indicate to what degree they belong to the i-th haplotype cluster. These components in particular indicate how many bicluster individuals possess the minor allele of the corresponding tagSNV. Components of $\boldsymbol{z}_i$ correspond to individuals and indicate both the number of segments that belong to the haplotype cluster and to what degree the individual's genotype matches the haplotype cluster.

Haplotype cluster extraction from FABIA models

FABIA biclustering extracts correlated SNVs but does not consider that a haplotype cluster consists of contiguous nucleotides which lead to local accumulations of a haplotype cluster's tagSNVs. This information can be used to separate unwanted spurious correlations between SNVs from desired correlations between tagSNVs. Further, haplotype clusters may overlap at ancient, preexisting SNVs and therefore may be joined in a FABIA model. Note that not all ancient SNVs are common and can be filtered out in a preprocessing step. FABIA may also join haplotype clusters if they contain the same individuals.

FABIA biclustering does not regard the order of SNVs or individuals, thus random shuffling of SNVs does not change its result. Therefore randomly correlated SNVs that are found by FABIA would be uniformly distributed along the chromosome. However SNVs that are correlated because they are tagSNVs of a haplotype cluster agglomerate locally as they originate from an ancient segment. Deviations from the null hypothesis of uniformly distributed SNVs can be detected by a binomial test for the number of expected SNVs within an interval if the SNV frequency is given. A low p-value hints at local agglomerations of bicluster SNVs stemming from a haplotype cluster.

We propose a four-step procedure to extract haplotype clusters from FABIA models:

1. identifying local agglomerations of correlated SNVs based on a binomial test,
2. disentangling haplotype clusters and re-assigning individuals or chromosomes to haplotype clusters,
3. pruning haplotype clusters from SNVs with spurious correlations based on an exponential test,
4. merging similar haplotype clusters, and joining the parts of large haplotype clusters that were divided by the bins from the first step.

Step 1: FABIA model selection is independent of the order of both the individuals/chromosomes and the SNVs. Therefore, spurious correlated SNVs are unlikely to agglomerate at a DNA locus, whereas local SNV agglomerations hint at a haplotype cluster. To detect agglomerations, we compute histogram counts of the largest values of the FABIA model parameters $\boldsymbol{\lambda}_i$. The threshold "Lt" (fixed to 0.1) gives the percentage of largest $\boldsymbol{\lambda}_i$ values that are used for the histogram. The HapFABIA parameter "IBDlength" gives the largest haplotype cluster (in kbp or centiMorgans) to be considered. The histogram bin size in number of SNVs is computed from "IBDlength" using the average genomic distance between adjacent SNVs. To account for haplotype clusters that exceed bin borders, the histogram is computed a second time with bins shifted by half the bin size.

The histogram bins with more counts than expected are assumed to contain haplotype clusters. We rank bin counts that exceed the expected value by a binomial test in order to select bins. We need to compute how many SNVs are expected in a bin, that is, the probability of observing k or more bin counts. Without errors in the factor analysis model, the minor allele of each tagSNV is present in each of the t individuals belonging to the haplotype cluster. Let p be the probability of a random minor allele match between t individuals. The probability of observing k or more matches for n SNVs in a bin is given by one minus the binomial distribution $F(k; n, p)$:

$$1 \; - \; F(k-1; n, p) \; = \; \Pr(K \geq k) \; = \; \sum_{i=k}^{n} \binom{n}{i} \, p^i \, (1-p)^{n-i} \; . \tag{15}$$

If q is the minor allele frequency (MAF) for one SNV, the probability p of observing the minor allele of this SNV in all t individuals is $p = q^t$. We assume that all SNVs have the same MAF q — in the experiments we used the average MAF. For b bins, the probability of observing k or more counts in at least one bin is

$$b \binom{l}{t} \sum_{i=k}^{n} \binom{n}{i} \, q^{it} \, \left(1 - q^t\right)^{n-i} \; , \tag{16}$$

where l is the number of individuals and $\binom{l}{t}$ is the number of possibilities to chose t individuals from the l individuals. If the probability Eq. (16) is below the threshold "thresCount", the according bin is selected for haplotype cluster extraction. If k_{min} is the minimum k for which Eq. (16) is below the threshold "thresCount", then all bins with counts $k \geq k_{\text{min}}$ are selected. In our experiments, we allow haplotype clusters of only two individuals (IBD runs), and therefore set $t = 2$.

If a bin is selected, SNVs and individuals must be assigned to it. Note that bicluster memberships of FABIA biclusters cannot be used directly, because they include all bins. First, those SNVs that contributed to its count are assigned to the selected bin. Then, individuals or chromosomes are assigned to the selected bin if they possess a minor allele at one or more SNVs that have been assigned to the bin. Individuals are only chosen from the top z-values of the FABIA model to ensure that individuals are indeed similar to each other. The parameter "Zt" (fixed to 0.2) gives the percentage of top z-scores that are considered.

Step 2: In this step, haplotype clusters in a selected bin are disentangled, considering only SNVs and individuals that have been assigned to the bin. A haplotype cluster is initialized by an IBD run between two core individuals that are identical at m or more minor alleles. The number m is computed as $m = \text{minSNPsFactor} \times k_{\text{min}}$. All individuals that are identical in least m

minor alleles to one of the two core haplotype cluster individuals are classified as belonging to the haplotype cluster.

The tagSNVs of a haplotype cluster are SNVs that have their minor allele in at least 2 individuals of the haplotype cluster. Alternatively, tagSNVs must have their minor allele in at least a specific percentage of haplotype cluster individuals (but in at least 2).

Step 2 is repeated after removing the current haplotype cluster by deleting the cluster's tagSNVs until no more core individuals are found.

Step3 : This step prunes haplotype cluster borders of SNVs that have spurious correlations to the haplotype cluster. Spurious correlations may still be present in a bin leading to an overestimation of the cluster length. Such SNVs can be identified by deviations of their MAFs from those of other tagSNVs. However, this criterion is not reliable for rare SNVs. Therefore, we identify SNVs with spurious correlations to the haplotype cluster on the basis of unusually large distances to other tagSNVs. The deviation from an expected distance is quantified by means of an exponential distribution with the median distance between tagSNVs as parameter. SNVs with distances leading to p-values below 1e-3 are removed. The two furthest upstream and the two furthest downstream tagSNVs are tested for their distances to other tagSNVs. If the second-furthest up- or downstream tagSNV is removed, then the furthest up- or downstream tagSNV is removed, too.

Step 4: Haplotype clusters which are very similar to each other are merged. Further, large haplotype cluster that were divided by the bins into smaller parts are reconstructed in this step. Thus, Haplotype clusters larger than given by the bin sizes can be detected. In order to compute similarities, we assess how many tagSNVs or individuals of the smaller haplotype cluster are explained by the larger haplotype cluster. This criterion is expressed by the "overlap coefficient"

$$O(A,B) \;=\; \frac{|A \cap B|}{\min\{|A|,|B|\}} \,. \tag{17}$$

Using the overlap coefficient for both tagSNVs and individuals, we define a distance-like measure between haplotype clusters H_1 and H_2 by

$$D(H_1,H_2) \;=\; 1 \;-\; O(S_{H_1},S_{H_2})\, O(I_{H_1},I_{H_2}) \,, \tag{18}$$

where S_{H_i} and I_{H_i} are the tagSNVs and individuals belonging to haplotype cluster H_i, respectively. Using measure D, haplotype clusters within overlapping (shifted) or neighboring bins are clustered by hierarchical clustering using complete linkage. Haplotype clusters are merged if their segments are clustered together below a cutting height of 0.8.

4 Tools to Analyze fabia Results

To analyze the fabia results we provide some functions. This might be convenient if parameters are optimized for a specific data set.

Accumulations of fabia loadings can be given as histogram counts to see locations of accumulations:

```
> data(res)
> h1 <- histL(res,n=1,p=0.9,intervv=50,off=0)
> print(h1$counts)
```

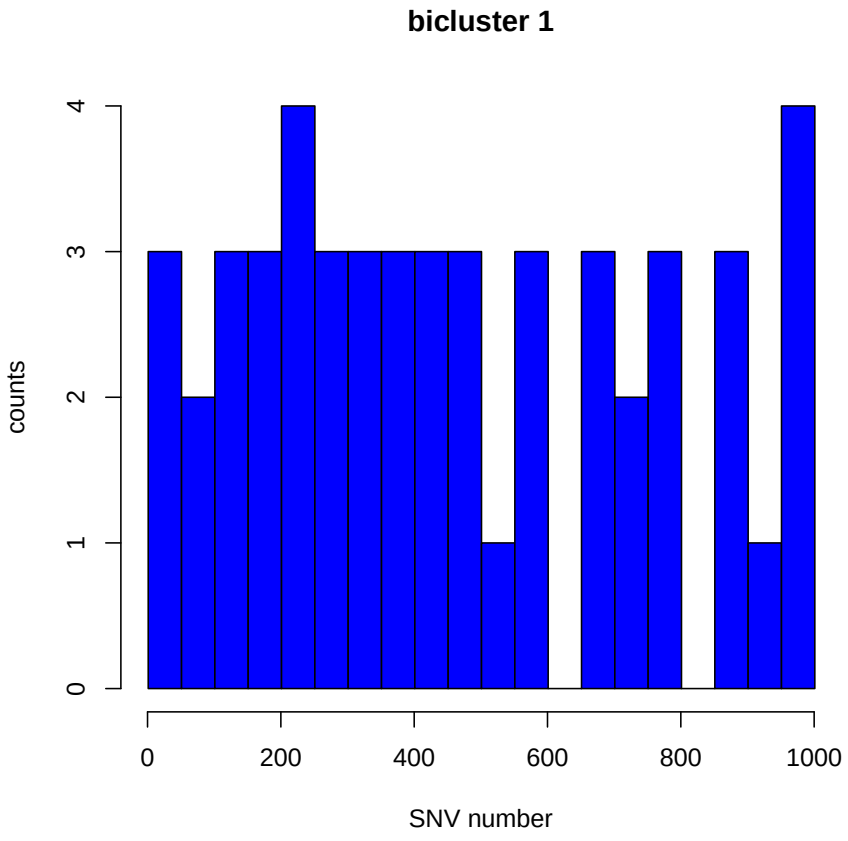
```
 [1]  9  5  7  7  5  4  4  4  5  4  3  6  1  5  6  4  1  4
[19]  5 11
```

```
> h1 <- histL(res,n=1,w=0.5,intervv=50,off=0)
> print(h1$counts)
```

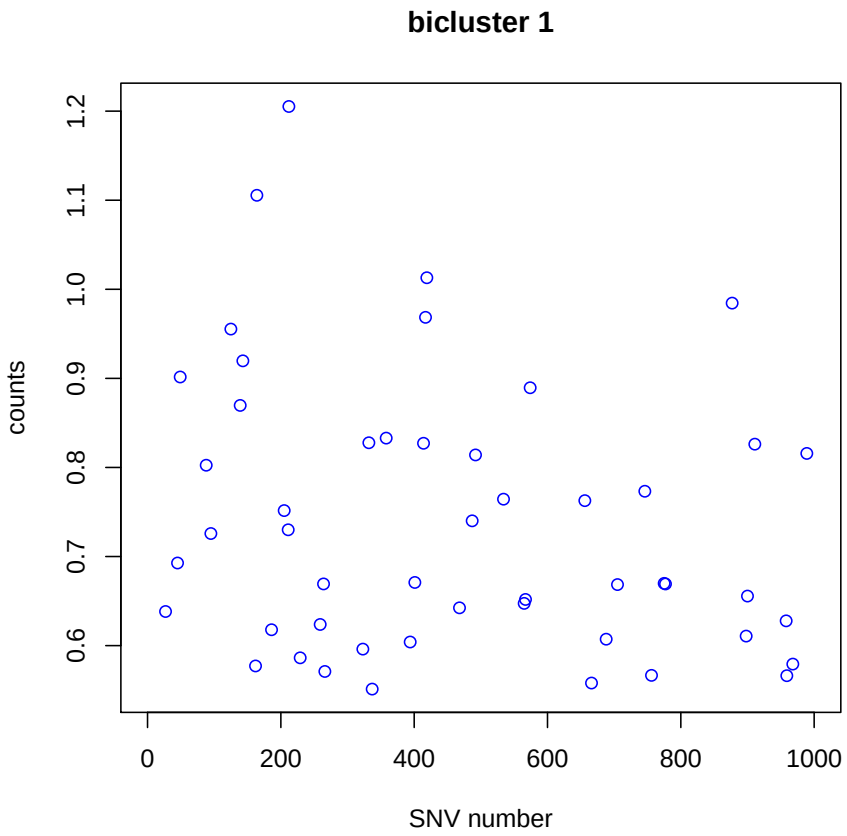
```
 [1] 4 2 6 5 4 3 3 4 3 3 1 4 0 4 3 3 1 3 3 7
```

fabia loadings can be plotted to identify locations of accumulations:

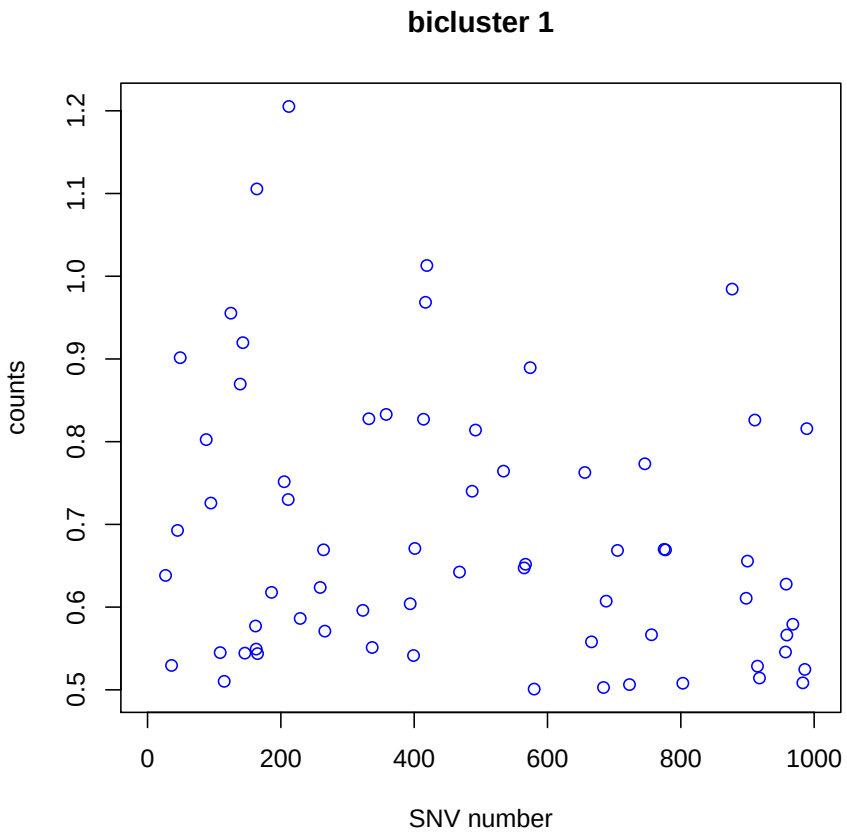
```
> data(res)
> plotL(res,1,p=0.95,type="histogram",intervv=50)
```



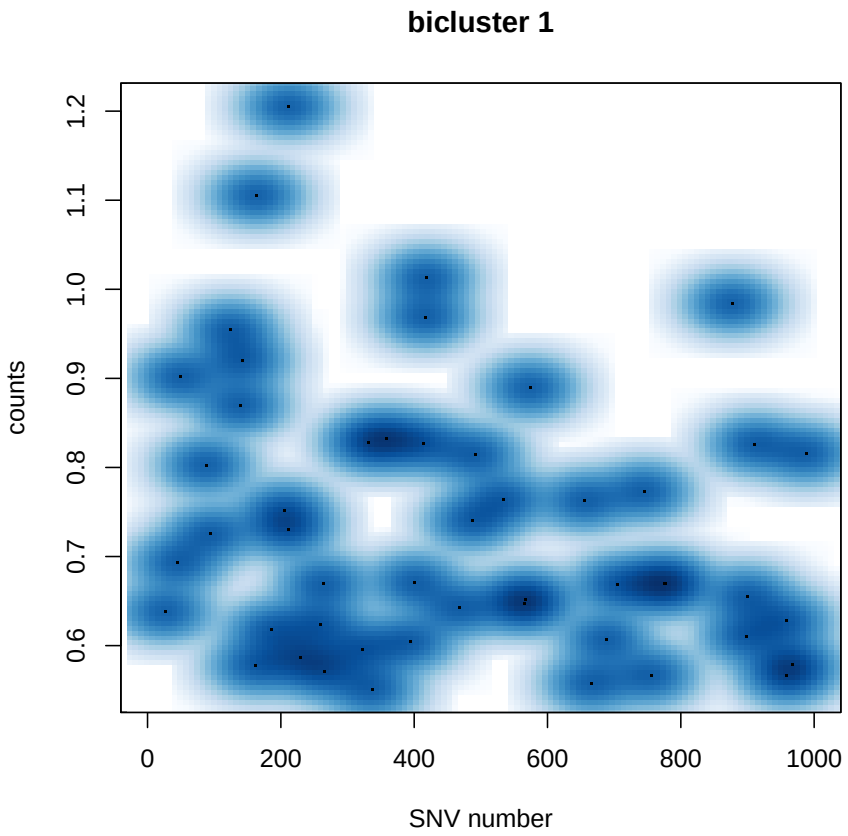

```
> data(res)
> plotL(res,1,p=0.95)
```



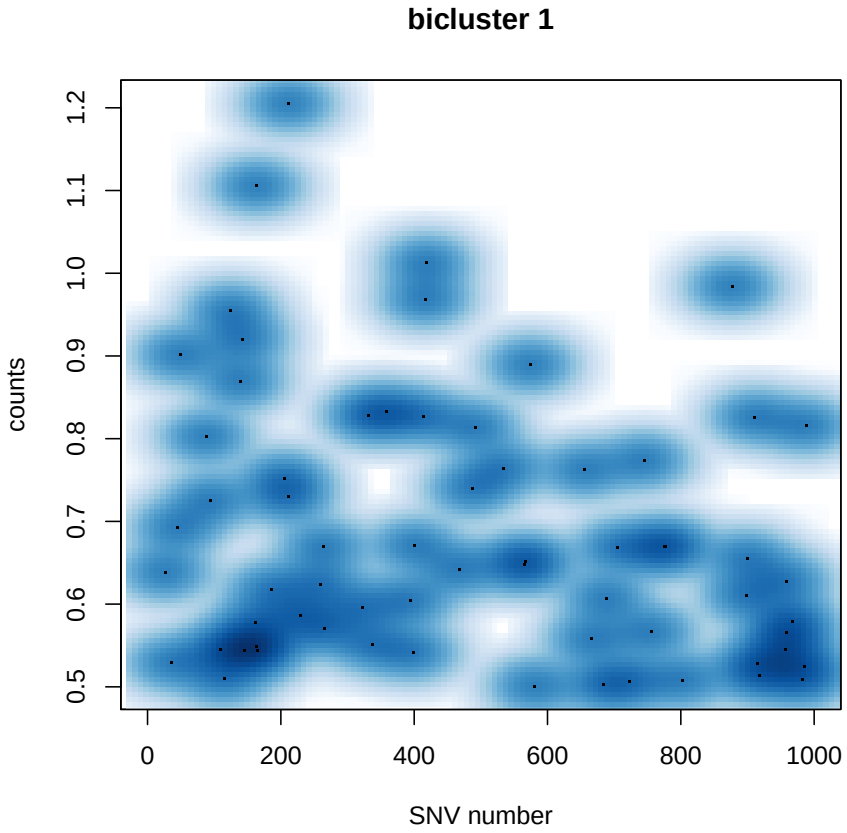

```
> data(res)
> plotL(res,1,w=0.5)
```




```
> data(res)
> plotL(res,1,p=0.95,type="smooth")
```




```
> data(res)
> plotL(res,1,w=0.5,type="smooth")
```



Finally the largest fabia loadings L and factors Z can be listed. The largest values must exceed a threshold either given by quantile p or a value w:

```
> data(res)
> topLZ(res,1,LZ="L",p=0.95)

 [1]  27  45  49  88  95 125 139 143 162 164 186 205 211 212
[15] 229 259 264 266 323 332 337 358 394 401 414 417 419 468
[29] 487 492 534 565 567 574 656 666 688 705 746 756 775 777
[43] 877 898 900 911 958 959 968 989

> topLZ(res,1,LZ="L",w=0.95)

[1] 125 164 212 417 419 877

> topLZ(res,1,LZ="Z",p=0.95)

 [1]   6  20  35  58  91  94 102 105 108 114

> topLZ(res,1,LZ="Z",w=0.4)
```

```
[1]   6 102
```

```
> topLZ(res,1,LZ="L",indices=FALSE,p=0.95)
```

```
 [1] 0.6383142 0.6927502 0.9015600 0.8025189 0.7258389
 [6] 0.9553665 0.8696658 0.9197025 0.5771737 1.1055338
[11] 0.6178381 0.7516020 0.7300434 1.2052466 0.5862968
[16] 0.6237540 0.6692794 0.5709860 0.5960892 0.8277471
[21] 0.5512068 0.8329555 0.6040622 0.6709233 0.8271526
[26] 0.9685284 1.0129973 0.6424376 0.7400939 0.8140626
[31] 0.7644123 0.6474202 0.6518362 0.8894892 0.7627514
[36] 0.5580086 0.6072268 0.6685370 0.7733036 0.5666094
[41] 0.6699066 0.6692214 0.9845007 0.6107024 0.6556427
[46] 0.8261591 0.6278021 0.5662607 0.5792704 0.8157592
```

```
> topLZ(res,1,LZ="L",indices=FALSE,w=0.95)
```

```
[1] 0.9553665 1.1055338 1.2052466 0.9685284 1.0129973
[6] 0.9845007
```

```
> topLZ(res,1,LZ="Z",indices=FALSE,p=0.95)
```

```
 [1] 0.4015947 0.3433146 0.3677638 0.3482399 0.3199918
 [6] 0.2772825 0.7713440 0.3260151 0.2891986 0.3086591
```

```
> topLZ(res,1,LZ="Z",indices=FALSE,w=0.4)
```

```
[1] 0.4015947 0.7713440
```

References

Clevert, D.-A., Mitterecker, A., Mayr, A., Klambauer, G., Tuefferd, M., DeBondt, A., Talloen, W., Göhlmann, H. W. H., and Hochreiter, S. (2011). cn.FARMS: a latent variable model to detect copy number variations in microarray data with a low false discovery rate. *Nucleic Acids Res*, **39**(12), e79.

Girolami, M. (2001). A variational method for learning sparse and overcomplete representations. *Neural Comput.*, **13**(11), 2517–2532.

Hochreiter, S., Bodenhofer, U., Heusel, M., Mayr, A., Mitterecker, A., Kasim, A., VanSanden, S., Lin, D., Talloen, W., Bijnens, L., Göhlmann, H. W. H., Shkedy, Z., and Clevert, D.-A. (2010). FABIA: factor analysis for bicluster acquisition. *Bioinformatics*, **26**(12), 1520–1527.

Klambauer, G., Schwarzbauer, K., Mayr, A., Clevert, D.-A., Mitterecker, A., Bodenhofer, U., and Hochreiter, S. (2012). cn.MOPS: mixture of poissons for discovering copy number variations in next generation sequencing data with a low false discovery rate. *Nucleic Acids Res*.

Li, G., Ma, Q., Tang, H., Paterson, A. H., and Xu, Y. (2009). QUBIC: a qualitative biclustering algorithm for analyses of gene expression data. *Nucleic Acids Res.*, **37**(15), e101.

Prelic, A., Bleuler, S., Zimmermann, P., Wille, A., Bühlmann, P., Gruissem, W., Hennig, L., Thiele, L., and Zitzler, E. (2006). A systematic comparison and evaluation of biclustering methods for gene expression data. *Bioinformatics*, **22**(9), 1122–1129.