**Software Manual**
Institute of Bioinformatics, Johannes Kepler University Linz

# hapFabia: Identification of very short segments of identity by descent (IBD) characterized by rare variants 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 1.16.1, March 12, 2017**

# Contents

# 1 Introduction

This package `hapFabia` provides software for the method HapFABIA which identifies short identity by descent (IBD) segments that are tagged by rare variants in large sequencing data.

Two haplotypes are identical by descent (IBD) if they share a segment that both inherited from a common ancestor. Current IBD methods reliably detect long IBD segments because many minor alleles in the segment are concordant between the two haplotypes. However, many cohort studies contain unrelated individuals which share only short IBD segments. Short IBD segments contain too few minor alleles to distinguish IBD from random allele sharing by recurrent mutations. New sequencing techniques improve the situation by providing rare variants which convey more information on IBD than common variants, because random minor allele sharing of rare variants is less likely than for common variants.

Short IBD segments are of interest because (i) they resolve the genetic structure on a fine scale and (ii) they can be assumed to be old. In order to detect short IBD segments, both the information supplied by rare variants and information from more than two individuals should be utilized. The probability of a segment being IBD is typically computed via the probabilities of randomly sharing single alleles within the segment. The probability of randomly sharing a single allele depends (1) on the allele frequency, where lower frequency means lower probability of random sharing, and (2) on the number of individuals that share the allele, where more individuals means lower probability of random sharing. Therefore a segment that contains rare variants and is shared by more individuals has higher significance of being IBD. These two characteristics are the basis for detecting short IBD segments by HapFABIA.

We propose biclustering **?** to detect very short IBD segments that are shared among multiple individuals. Biclustering simultaneously clusters rows and columns of a matrix. In particular it clusters row elements that are similar to each other on a subset of column elements. A genotype matrix has individuals (unphased) or chromosomes (phased) as row elements and SNVs as column elements. Entries in the genotype matrix usually count how often the minor allele of a particular SNV is present in a particular individual. Alternatively, minor allele likelihoods or dosages may be used. Individuals that share an IBD segment are similar to each other at minor alleles of SNVs (tagSNVs) which tag the IBD segment (see Fig. **??**). Therefore an IBD segment that is shared among individuals corresponds to a bicluster because these individuals are similar to one another at this segment. Identifying a bicluster means identifying tagSNVs (column bicluster elements) that tag an IBD segment and, simultaneously, identifying individuals (row bicluster elements) that possess the IBD segment.

In contrast to standard IBD detection methods, biclustering considers multiple individuals. In contrast to standard clustering, biclustering allows for SNVs or individuals that do not belong to any cluster or to more than one bicluster. Multiple cluster membership suits IBD detection because diploid individuals can have two IBD segments at one locus and an SNV may tag more than one IBD segment.

FABIA is able to represent homozygous regions (the same IBD segment in both chromosomes) by means of its factors. At a locus, overlapping IBD segments in one diploid individual (a different IBD segment in each of the two chromosomes) are represented through additivity of biclusters in the FABIA model. Examples of short IBD segments found by hapFabia in chromosome 1 data from the 1000 Genomes Project are given in Fig. **??** and Fig. **??**.

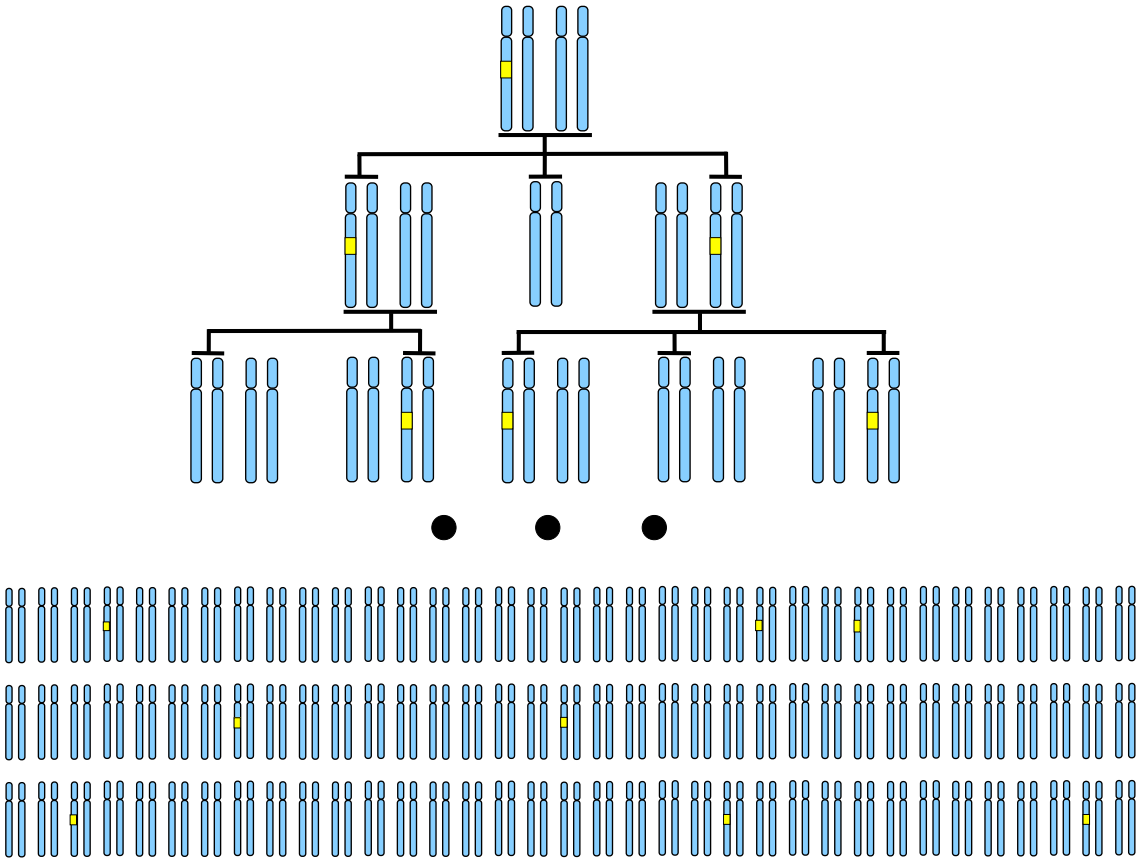


Figure 1: The IBD segment marked in yellow descended from a founder to different individuals.

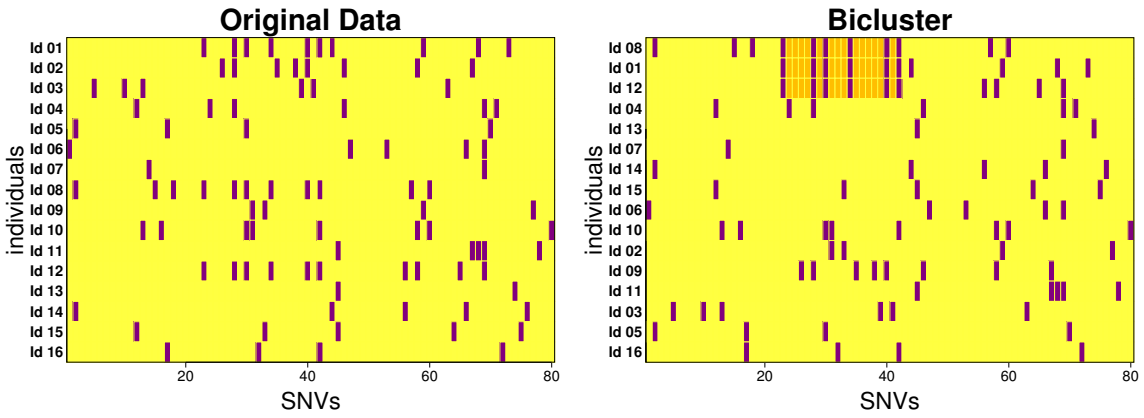



Figure 2: Biclustering of a genotyping matrix. Left: original genotyping data matrix with individuals as row elements and SNVs as column elements. Minor alleles are indicated by violet bars and major alleles by yellow bars for each individual-SNV pair. Right: after sorting the rows, the detected bicluster can be seen in the top three individuals. They contain the same IBD segment which is marked in gold. Biclustering simultaneously clusters rows and columns of a matrix so that row elements (here individuals) are similar to each other on a subset of column elements (here the tagSNVs).

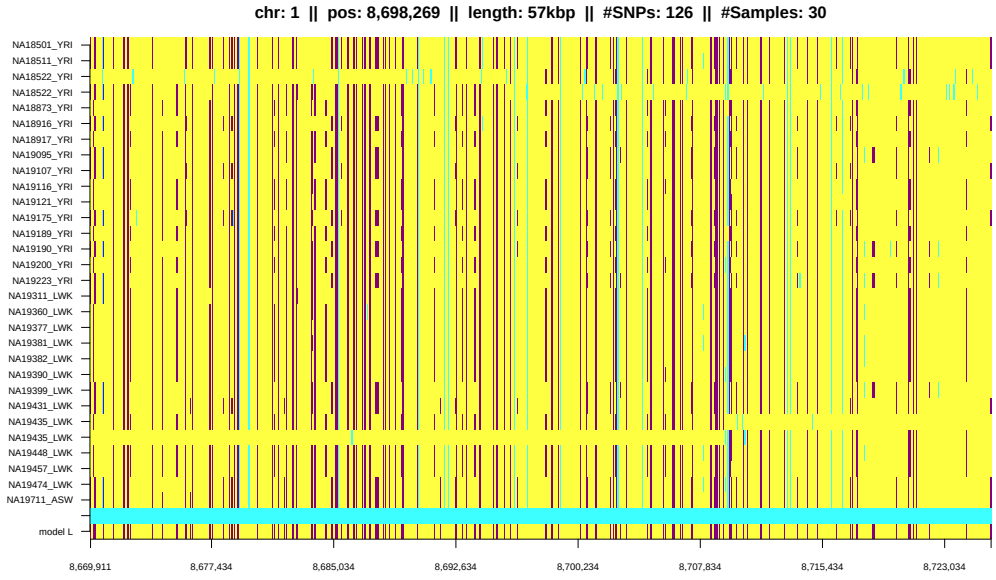



Figure 3: Example of an IBD segment in chromosome 1 found in the 1000 Genomes Project data. The $y$-axis gives chromosomes and the $x$-axis consecutive SNVs. Yellow indicates major alleles, violet minor alleles of tagSNVs, and blue minor alleles of other SNVs. "model L" indicates tagSNVs identified by hapFabia in violet. A probable phasing error can be seen in line 3 and 4 at individual NA18522. Another phasing error can be seen in the last but four and the last but five line at individual NA19435.

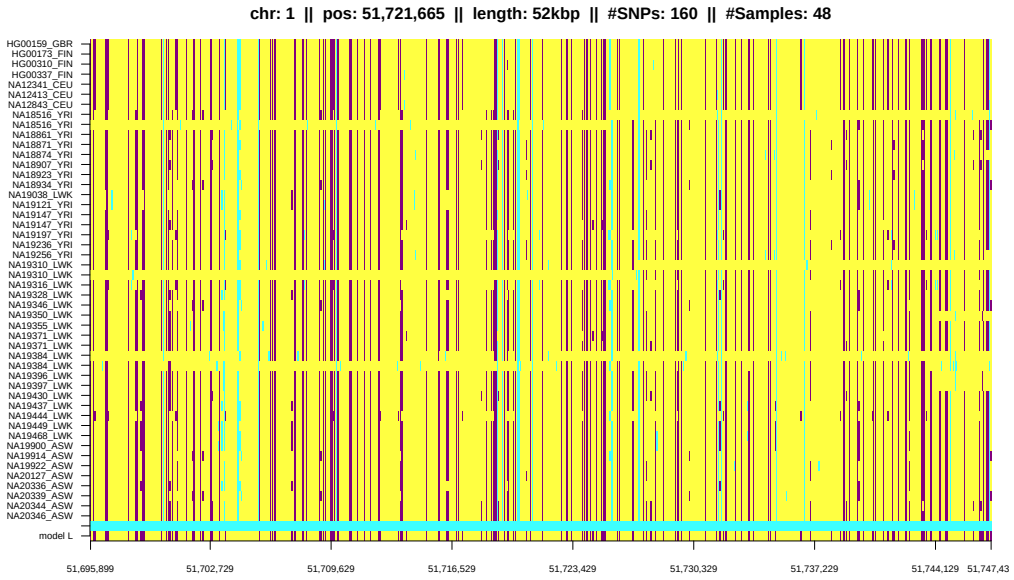



Figure 4: Another example of an IBD segment from chromosome 1 of the 1000 Genomes Project. See Fig. **??** for a description. Again probable phasing errors at individuals NA18516, NA19310, and NA19384.

# 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, 3. copy genotype matrix to the matrix that is processed, and then 4. split into intervals. The following command line commands perform these steps:

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

In `inst/commandline/arch/` command line tools for steps 2. to 4. are provided by the package `hapFabia`. However step 2. to 4. can be performed in R as well (see below). The commandline parameters for `vcftoFABIA` are

1. filename without `.vcf`
2. path to the file, e.g. `./`
3. optional: `-s snps`, where snps gives the number of SNVs in the input data file
4. optional: `-o outputFileName`, which gives the prefix of the output files.

The commandline parameters for `split_sparse_matrix` are

1. filename without `.vcf`
2. extension (default`_mat.txt`)
3. interval length
4. shift size
5. indicator whether annotation is present (is generated by `vcftoFABIA` as default).

The data is split into intervals of 10,000 SNVs where the distance between adjacent intervals 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 intervals, overlap, filename #######
R> shiftSize <- 5000
R> intervalSize <- 10000
R> fileName="filename" # without type
R>
R> ##### load library #######
R> library(hapFabia)
R>
R> ##### convert from .vcf to _mat.txt: step 2. above #######
R> vcftoFABIA(fileName=fileName)
R>
##### copy genotype matrix to matrix: step 3. above #######
R> file.copy(paste(fileName,"_matG.txt",sep=""),
+  paste(fileName,"_mat.txt",sep=""))
R>
R> ##### split/ generate intervals: step 4. above #######
R> split_sparse_matrix(fileName=fileName,intervalSize=intervalSize,
+  shiftSize=shiftSize,annotation=TRUE)
R>
R> ##### compute how many intervals we have #######
R> ina <- as.numeric(readLines(paste(fileName,"_mat.txt",sep=""),n=2))
R> noSNVs <- ina[2]
R> over <- intervalSize%/%shiftSize
R> N1 <- noSNVs%/%shiftSize
R> endRunA <- (N1-over+2)
R>
R> ##### analyze each interval #######
R> ##### may be done by parallel runs #######
R> iterateIntervals(startRun=1,endRun=endRunA,shift=shiftSize,
+  intervalSize=intervalSize,fileName=fileName,individuals=0,
+  upperBP=0.05,p=10,iter=40,alpha=0.03,cyc=50,IBDsegmentLength=50,
+  Lt = 0.1,Zt = 0.2,thresCount=1e-5,mintagSNVsFactor=3/4,
+  pMAF=0.035,haplotypes=FALSE,cut=0.8,procMinIndivids=0.1,thresPrune=1e-3,
+  simv="minD",minTagSNVs=6,minIndivid=2,avSNVsDist=100,SNVclusterLength=100)
R>
R> ##### identify duplicates #######
R> identifyDuplicates(fileName=fileName,startRun=1,endRun=endRunA,
+  shift=shiftSize,intervalSize=intervalSize)
R>
R> ##### analyze results; parallel #######
R> anaRes <- analyzeIBDsegments(fileName=fileName,startRun=1,endRun=endRunA,
+  shift=shiftSize,intervalSize=intervalSize)
R> print("Number IBD segments:")
R> print(anaRes$noIBDsegments)
R> print("Statistics on IBD segment lengths in SNVs (all SNVs in the
IBD segment):")
```

```
R> print(anaRes$avIBDsegmentLengthSNVS)
R> print("Statistics on IBD segment lengths in bp:")
R> print(anaRes$avIBDsegmentLengthS)
R> print("Statistics on number of individuals that share an IBD segment:")
R> print(anaRes$avnoIndividS)
R> print("Statistics on number of IBD segment tagSNVs:")
R> print(anaRes$avnoTagSNVsS)
R> print("Statistics on MAF of IBD segment tagSNVs:")
R> print(anaRes$avnoFreqS)
R> print("Statistics on MAF within the group of IBD segment tagSNVs:")
R> print(anaRes$avnoGroupFreqS)
R> print("Statistics on number of changes between major and minor allele frequency:")
R> print(anaRes$avnotagSNVChangeS)
R> print("Statistics on tagSNVs per individual that shares an IBD segment:")
R> print(anaRes$avnotagSNVsPerIndividualS)
R> print("Statistics on number of individuals that have the minor allele of tagSNVs:")
R> print(anaRes$avnoindividualPerTagSNVS)
R>
R> ##### load result for interval 50 #######
R> posAll <- 50 # (50-1)*5000 = 245000: segment 245000 to 255000
R> start <- (posAll-1)*shiftSize
R> end <- start + intervalSize
R> pRange <- paste("_",format(start,scientific=FALSE),"_",
+  format(end,scientific=FALSE),sep="")
R> load(file=paste(fileName,pRange,"_resAnno",".Rda",sep=""))
R> IBDsegmentList <- resHapFabia$mergedIBDsegmentList # $
R>
R> summary(IBDsegmentList)
R> ##### plot IBD segments in interval 50 #######
R> plot(IBDsegmentList,filename=paste(fileName,pRange,"_mat",sep=""))
R>    ##attention: filename without type ".txt"
R>
R> ##### plot the first IBD segment in interval 50 #######
R>
R> IBDsegment <- IBDsegmentList[[1]]
R> plot(IBDsegment,filename=paste(fileName,pRange,"_mat",sep=""))
R>    ##attention: filename without type ".txt"
```

First the packages `hapFabia` and `fabia` are loaded. Then `vcftoFABIA` converts `filename.vcf` to sparse matrix format giving:

- `filename_matH.txt` (haplotype data),
- `filename_matG.txt` (genotype data),
- `filename_matD.txt` (dosage data),

together with the SNV annotation file and individual's label file:

- `filename_annot.txt` and
- `filename_individuals.txt`.

The function `split_sparse_matrix` splits the data into intervals. The function `iterateIntervals` identifies IBD segments in these intervals and stores the results in an EXCEL like `.csv` format and as an R data object. The function `identifyDuplicates` marks and memorizes duplicates of IBD segments which occur because the intervals overlap. Next the function `analyzeIBDsegments` 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 IBD segments of a segment, for which we chose interval 50 which corresponds to chromosome 1 range from 245,000 to 255,000 ($(50-1)*5000 = 245000$). Then we plot a specific IBD segment, in this case the first (`IBDsegmentList[[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. **??** and Fig. **??**.

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

```
R> makePipelineFile("filename",shiftSize=5000,
+  intervalSize=10000,haplotypes=FALSE)
R>
R> source("pipeline.R")
```

NOTE: sourcing may take a while for large datasets.

The next example shows how to use the pipeline.

## 2.2 Examples

Next we show an example how to use `hapFabia`. This example shows how to run the whole pipeline if the genotype data is given as `.vcf` file. The data is first converted to a sparse matrix format by `vcftoFABIA` and then divided into overlapping intervals by `split_sparse_matrix`. Then the IBD segments are extracted by `iterateIntervals` and duplicates due to overlapping intervals marked by `identifyDuplicates`. Subsequently the IBD segments are analyzed by `analyzeIBDsegments`, where only simple statistics are computed.

Work in a temporary directory.

```
> old_dir <- getwd()
> setwd(tempdir())
```

First the package is loaded.

```
> library(hapFabia)
```

Load data and write to `vcf` file. In a real application the data is already given, therefore this chunk of code would not be necessary.

```
> data(chr1ASW1000G)
> write(chr1ASW1000G,file="chr1ASW1000G.vcf")
```

Create the analysis pipeline, where intervals contain only 1,000 SNVs (that is a length of about 100 kbps), while default is 10,000 SNVs (that is about a length of 1 Mbp).

```
> makePipelineFile(fileName="chr1ASW1000G",shiftSize=500,
+                  intervalSize=1000,haplotypes=TRUE)
```

Now the pipeline can be executed by sourcing it.

```
> source("pipeline.R")
```

Next we list the files that were generated, where `_N1_N2_` indicates that the file contains information concerning the interval that starts at N1 and ends at N2, `_ALL.Rda` stores just the number of individuals and the number of SNVs, `_individuals.txt` contains annotation for the individuals (in particular their names), `_matG.txt` denotes phased genotype data in sparse matrix format, `_matD.txt` contains the genotype data as dosage in sparse matrix format, `_matG.txt` contains unphased genotype data in sparse matrix format, `_annot.txt` supplies information on the SNVs, `_resAnno.Rda` is the result from hapFabia, the result is also available as `csv` file with extension `_csv.txt`.

Following files have been generated:

```
> list.files(pattern="chr1")
```

```
 [1] "chr1ASW1000G.vcf"
 [2] "chr1ASW1000G_0_1000.csv"
 [3] "chr1ASW1000G_0_1000_annot.txt"
 [4] "chr1ASW1000G_0_1000_mat.txt"
 [5] "chr1ASW1000G_0_1000_resAnno.Rda"
 [6] "chr1ASW1000G_1000_2000.csv"
 [7] "chr1ASW1000G_1000_2000_annot.txt"
 [8] "chr1ASW1000G_1000_2000_mat.txt"
 [9] "chr1ASW1000G_1000_2000_resAnno.Rda"
[10] "chr1ASW1000G_1500_2500.csv"
[11] "chr1ASW1000G_1500_2500_annot.txt"
[12] "chr1ASW1000G_1500_2500_mat.txt"
[13] "chr1ASW1000G_1500_2500_resAnno.Rda"
[14] "chr1ASW1000G_2000_3000.csv"
[15] "chr1ASW1000G_2000_3000_annot.txt"
[16] "chr1ASW1000G_2000_3000_mat.txt"
[17] "chr1ASW1000G_2000_3000_resAnno.Rda"
[18] "chr1ASW1000G_2500_3022.csv"
[19] "chr1ASW1000G_2500_3022_annot.txt"
[20] "chr1ASW1000G_2500_3022_mat.txt"
```

```
[21] "chr1ASW1000G_2500_3022_resAnno.Rda"
[22] "chr1ASW1000G_500_1500.csv"
[23] "chr1ASW1000G_500_1500_annot.txt"
[24] "chr1ASW1000G_500_1500_mat.txt"
[25] "chr1ASW1000G_500_1500_resAnno.Rda"
[26] "chr1ASW1000G_All.Rda"
[27] "chr1ASW1000G_annot.txt"
[28] "chr1ASW1000G_individuals.txt"
[29] "chr1ASW1000G_mat.txt"
[30] "chr1ASW1000G_matD.txt"
[31] "chr1ASW1000G_matG.txt"
[32] "chr1ASW1000G_matH.txt"
```

In the following we show the results of calling the function `analyzeIBDsegments` in above pipeline:

```
> print("Number IBD segments:")

[1] "Number IBD segments:"

> print(anaRes$noIBDsegments)

[1] 42

> print("Statistics on IBD segment length in SNVs (all SNVs in the IBD segment):")

[1] "Statistics on IBD segment length in SNVs (all SNVs in the IBD segment):"

> print(anaRes$avIBDsegmentLengthSNVS)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   9181   29370   37720   36980   44980   73990

> print("Statistics on IBD segment length in bp:")

[1] "Statistics on IBD segment length in bp:"

> print(anaRes$avIBDsegmentLengthS)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   9181   29370   37720   36980   44980   73990

> print("Statistics on number of individuals belonging to IBD segments:")
```

```
[1] "Statistics on number of individuals belonging to IBD segments:"

> print(anaRes$avnoIndividS)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  2.000   2.000   3.000   2.833   3.000   8.000

> print("Statistics on number of tagSNVs of IBD segments:")

[1] "Statistics on number of tagSNVs of IBD segments:"

> print(anaRes$avnoTagSNVsS)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   7.00    8.50   12.50   15.33   19.75   54.00

> print("Statistics on MAF of tagSNVs of IBD segments:")

[1] "Statistics on MAF of tagSNVs of IBD segments:"

> print(anaRes$avnoFreqS)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.01639 0.01639 0.02459 0.03673 0.04098 0.97540

> print("Statistics on MAF within the group of tagSNVs of IBD segments:")

[1] "Statistics on MAF within the group of tagSNVs of IBD segments:"

> print(anaRes$avnoGroupFreqS)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.01639 0.01639 0.02459 0.02966 0.04098 0.04918

> print("Statistics on number of changes between major and minor allele frequency:")

[1] "Statistics on number of changes between major and minor allele frequency:"

> print(anaRes$avnotagSNVChangeS)

    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
0.000000 0.000000 0.000000 0.007764 0.000000 1.000000
```

```
> print("Statistics on number of tagSNVs per individual of an IBD segment:")

[1] "Statistics on number of tagSNVs per individual of an IBD segment:"

> print(anaRes$avnotagSNVsPerIndividualS)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   7.00    8.50   12.00   14.42   17.00   41.00

> print("Statistics on number of individuals that have the minor allele of tagSNVs:")

[1] "Statistics on number of individuals that have the minor allele of tagSNVs:"

> print(anaRes$avnoindividualPerTagSNVS)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  2.000   2.000   2.000   2.759   3.000   6.000
```

Next we load interval 5 and there the first and second IBD segment

```
> posAll <- 5
> start <- (posAll-1)*shiftSize
> end <- start + intervalSize
> pRange <- paste("_",format(start,scientific=FALSE),"_",
+                 format(end,scientific=FALSE),sep="")
> load(file=paste(fileName,pRange,"_resAnno",".Rda",sep=""))
> IBDsegmentList <- resHapFabia$mergedIBDsegmentList
> summary(IBDsegmentList)

An object of class IBDsegmentList
Number of IBD segments:  8
Statistics:
$avIBDsegmentPosS
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 928900  949600  966400  969800  988800 1019000

$avIBDsegmentLengthSNVS
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   9181   11860   22180   26600   40520   51350

$avIBDsegmentLengthS
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   9181   11860   22180   26600   40520   51350

$avnoIndividS
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  2.000   2.000   2.000   2.375   3.000   3.000

$avnoTagSNVsS
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   7.00   10.00   12.50   16.75   26.00   30.00

$avnoFreqS
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.01639 0.01639 0.01639 0.02386 0.02459 0.04918

$avnoGroupFreqS
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.01639 0.01639 0.01639 0.02386 0.02459 0.04918

$avnotagSNVChangeS
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      0       0       0       0       0       0

$avnotagSNVsPerIndividualS
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   7.00   10.00   11.00   15.68   26.00   29.00

$avnoindividualPerTagSNVS
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  2.000   2.000   2.000   2.246   2.000   3.000

> IBDsegment1 <- IBDsegmentList[[1]]
> summary(IBDsegment1)

An object of class IBDsegment
IBD segment ID:  1
From bicluster:  2
Chromosome:  1
Position:  928,942
Length SNVs:  51353
Length:  51 kbp
Number of individuals/chromosomes:  3
Number of tagSNVs:  14

> IBDsegment2 <- IBDsegmentList[[2]]
> summary(IBDsegment2)

An object of class IBDsegment
IBD segment ID:  2
From bicluster:  4
```

```
Chromosome:  1
Position:  953,918
Length SNVs:  46863
Length:  47 kbp
Number of individuals/chromosomes:  3
Number of tagSNVs:  30
```

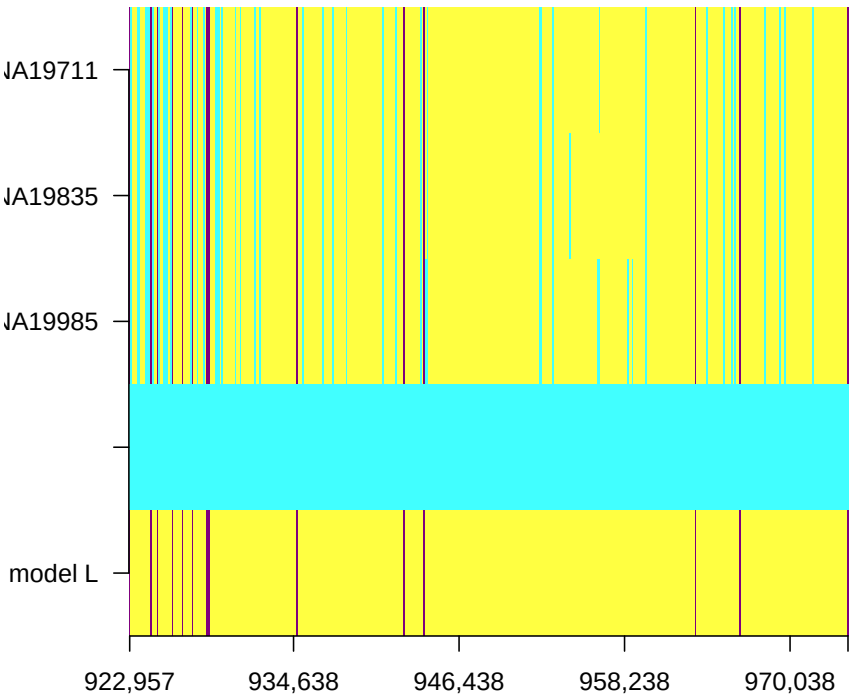
Finally go back to old directory.

```
> new_dir <- getwd()
> setwd(old_dir)
```

Plot the first IBD segment in interval 5. In the plot the $y$-axis gives the individuals or the chromosomes and the $x$-axis consecutive SNVs. The default color coding uses yellow for major alleles, violet for minor alleles of tagSNVs, and blue for minor alleles of other SNVs. "model L" indicates tagSNVs identified by hapFabia in violet.

```
> plot(IBDsegment1,filename=paste(new_dir,"/",fileName,pRange,"_mat",sep=""))
```

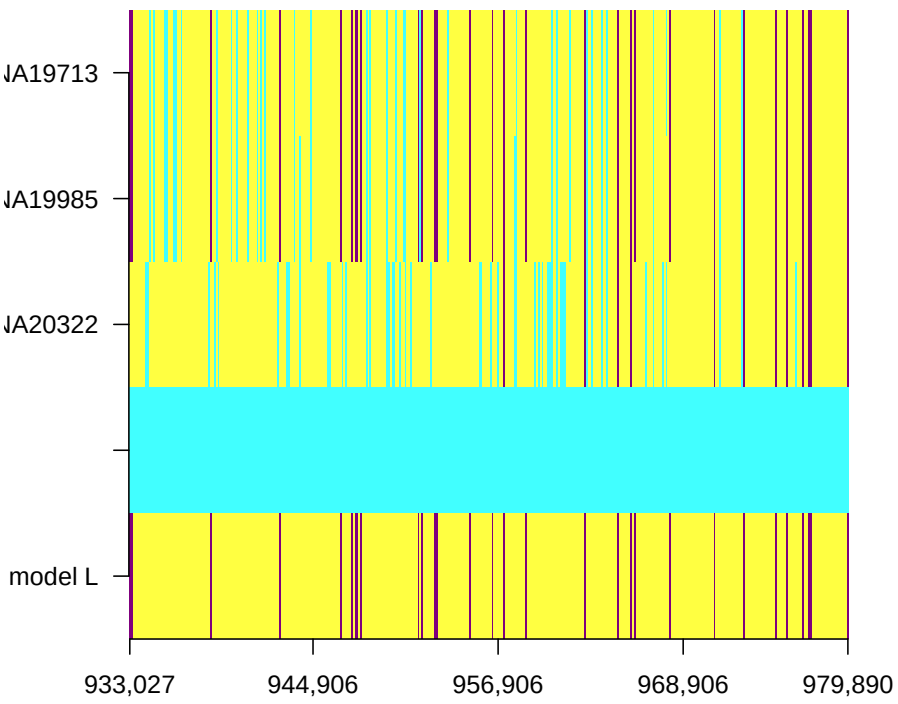
```
Using 3 samples!
```




Plot the second IBD segment in interval 5.

```
> plot(IBDsegment2,filename=paste(new_dir,"/",fileName,pRange,"_mat",sep=""))
```

```
Using 3 samples!
```




Here an example with simulated data.

Work in temporary directory.

```
> old_dir <- getwd()
> setwd(tempdir())
```

The data `simu` is loaded and written into three files:

- `dataSim1fabia_individuals.txt` (sample names),
- `dataSim1fabia_annot.txt` (SNV annotation information), and
- `dataSim1fabia_mat.txt` (the data in sparse matrix format).

These files are generated by the standard pipeline by `vcftoFABIA` and by `split_sparse_matrix`.

```
> data(simu)
> namesL <- simu[["namesL"]]
> haploN <- simu[["haploN"]]
> snvs <- simu[["snvs"]]
> annot <- simu[["annot"]]
> alleleIimp <- simu[["alleleIimp"]]
> write.table(namesL,file="dataSim1fabia_individuals.txt",
+     quote = FALSE,row.names = FALSE,col.names = FALSE)
> write(as.integer(haploN),file="dataSim1fabia_annot.txt",
+     ncolumns=100)
> write(as.integer(snvs),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(snvs),file="dataSim1fabia_mat.txt",
+     append=TRUE,ncolumns=100)
> for (i in 1:haploN) {
+
+   a1 <- which(alleleIimp[i,]>0.01)
+
+   al <- length(a1)
+   b1 <- alleleIimp[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 IBD segments can be extracted from the data:

```
> hapRes <- hapFabia(fileName="dataSim1fabia",prefixPath="",
+    sparseMatrixPostfix="_mat",
+    annotPostfix="_annot.txt",individualsPostfix="_individuals.txt",
+    labelsA=NULL,pRange="",individuals=0,lowerBP=0,upperBP=0.15,
+    p=10,iter=1,quant=0.01,eps=1e-5,alpha=0.03,cyc=50,non_negative=1,
+    write_file=0,norm=0,lap=100.0,IBDsegmentLength=10,Lt = 0.1,
```

```
+    Zt = 0.2,thresCount=1e-5,mintagSNVsFactor=3/4,pMAF=0.1,
+    haplotypes=FALSE,cut=0.8,procMinIndivids=0.1,thresPrune=1e-3,
+    simv="minD",minTagSNVs=6,minIndivid=2,avSNVsDist=100,SNVclusterLength=100)

> summary(hapRes$mergedIBDsegmentList)

An object of class IBDsegmentList
Number of IBD segments:  1
Statistics:
$avIBDsegmentPosS
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  79430   79430   79430   79430   79430   79430

$avIBDsegmentLengthSNVS
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   3853    3853    3853    3853    3853    3853

$avIBDsegmentLengthS
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   3853    3853    3853    3853    3853    3853

$avnoIndividS
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
     10      10      10      10      10      10

$avnoTagSNVsS
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
     15      15      15      15      15      15

$avnoFreqS
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.04500 0.07750 0.08500 0.08267 0.09250 0.11000

$avnoGroupFreqS
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.0950  0.1175  0.1300  0.1267  0.1400  0.1450

$avnotagSNVChangeS
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      0       0       0       0       0       0

$avnotagSNVsPerIndividualS
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   14.0    14.0    14.0    14.4    15.0    15.0

$avnoindividualPerTagSNVS
```

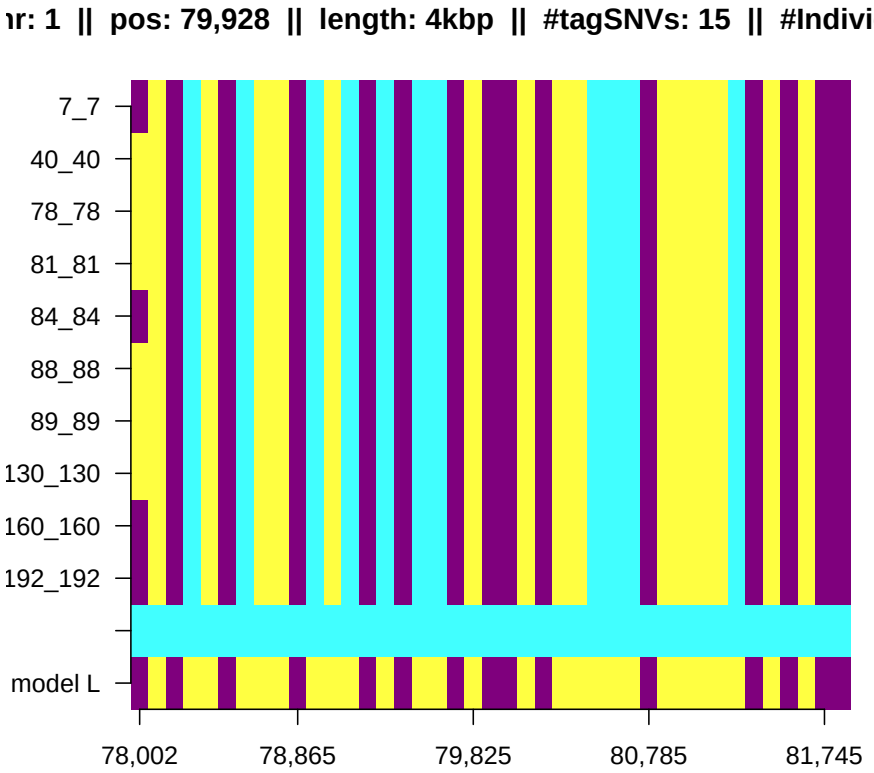
```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    4.0    10.0    10.0     9.6    10.0    10.0
```

To view the results the first IBD segment is plotted:

```
> mergedIBDsegmentList <- hapRes$mergedIBDsegmentList # $
> IBDsegment <- mergedIBDsegmentList[[1]]

> new_dir <- getwd()
> setwd(old_dir)
```

Again, in the plot the $y$-axis gives the individuals or the chromosomes and the $x$-axis consecutive SNVs. The default color coding uses yellow for major alleles, violet for minor alleles of tagSNVs, and blue for minor alleles of other SNVs. "model L" indicates tagSNVs identified by hapFabia in violet.

```
> plot(IBDsegment,filename=paste(new_dir,"/dataSim1fabia_mat",sep=""))

Using 10 samples!
```




Here another example with random data:

```
> old_dir <- getwd()
> setwd(tempdir())

> simulateIBDsegmentsFabia(minruns=2,maxruns=2)

> hapRes <- hapFabia(fileName="dataSim2fabia",prefixPath="",
+    sparseMatrixPostfix="_mat",
+    annotPostfix="_annot.txt",individualsPostfix="_individuals.txt",
+    labelsA=NULL,pRange="",individuals=0,lowerBP=0,upperBP=0.15,
+    p=10,iter=1,quant=0.01,eps=1e-5,alpha=0.03,cyc=50,non_negative=1,
+    write_file=0,norm=0,lap=100.0,IBDsegmentLength=10,Lt = 0.1,
+    Zt = 0.2,thresCount=1e-5,mintagSNVsFactor=3/4,pMAF=0.1,
+    haplotypes=FALSE,cut=0.8,procMinIndivids=0.1,thresPrune=1e-3,
+    simv="minD",minTagSNVs=6,minIndivid=2,avSNVsDist=100,SNVclusterLength=100)

> summary(hapRes$mergedIBDsegmentList)

An object of class IBDsegmentList
Number of IBD segments:  1
Statistics:
$avIBDsegmentPosS
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  94310   94310   94310   94310   94310   94310

$avIBDsegmentLengthSNVS
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   8379    8379    8379    8379    8379    8379

$avIBDsegmentLengthS
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   8379    8379    8379    8379    8379    8379

$avnoIndividS
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
     10      10      10      10      10      10

$avnoTagSNVsS
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
     22      22      22      22      22      22

$avnoFreqS
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.05500 0.07250 0.08250 0.08364 0.09375 0.11500

$avnoGroupFreqS
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
```

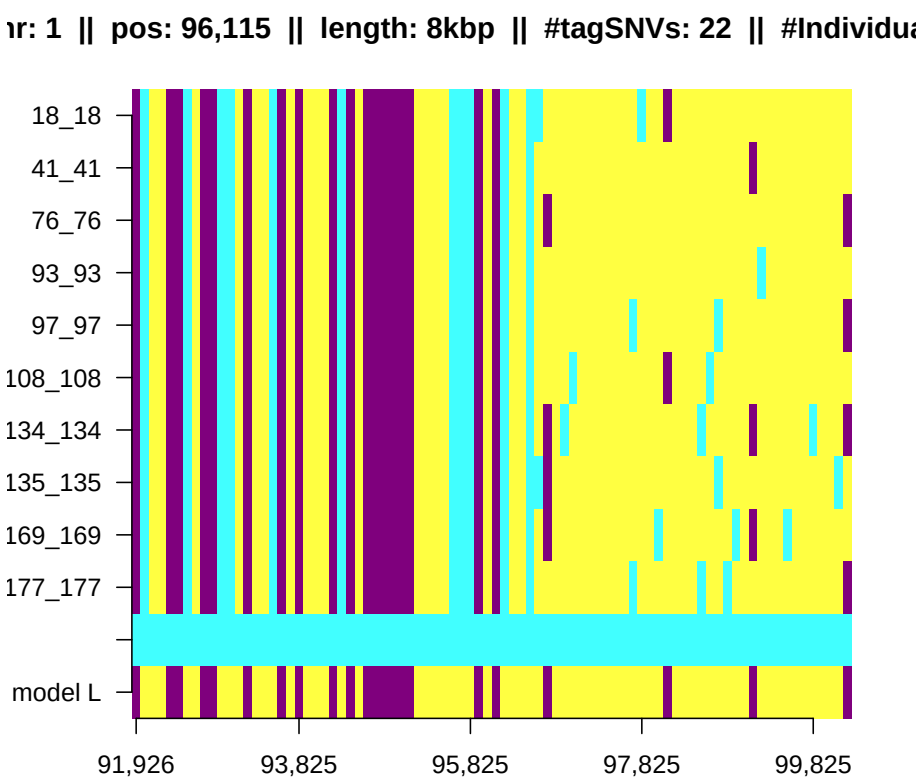
```
 0.1000  0.1100  0.1250  0.1216  0.1300  0.1450

$avnotagSNVChangeS
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      0       0       0       0       0       0

$avnotagSNVsPerIndividualS
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  18.00   19.00   19.00   19.30   19.75   21.00

$avnoindividualPerTagSNVS
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  2.000  10.000  10.000   8.773  10.000  10.000
```

```
> mergedIBDsegmentList <- hapRes$mergedIBDsegmentList # $
> IBDsegment <- mergedIBDsegmentList[[1]]
```

```
> new_dir <- getwd()
> setwd(old_dir)
```

```
> plot(IBDsegment,filename=paste(new_dir,"/dataSim2fabia_mat",sep=""))
```

```
Using 10 samples!
```




# 3 hapFabia Method

Our novel method HapFABIA extracts short IBD segments that are tagged by rare variants from large sequencing data. In the first stage, HapFABIA applies FABIA biclustering to phased or unphased genotype data. Biclustering extracts similarities between individuals based on a subset of SNVs, but does not consider that IBD segments consist of contiguous nucleotides. In the second stage, HapFABIA extracts IBD segments from FABIA models by considering local tagSNV accumulations. SNVs that tag an IBD segment are within this segment and, therefore, accumulate locally. It is important for the second stage that the SNVs which are extracted in the first step are independent of their DNA location. This justifies to use statistical methods for identifying local SNV accumulations in FABIA models which would not be expected randomly. Finally, HapFABIA prunes spurious correlated SNVs from the extracted IBD segments and joins segments. The two HapFABIA steps (biclustering and IBD segment extraction) are described in the next two subsections.

## 3.1 FABIA for genotype data

In the following, we describe the first step of HapFABIA. We propose identifying similarities between individuals by biclustering. Biclustering simultaneously clusters rows and columns of a

matrix. More specifically, it clusters row elements that are similar to each other on a subset of column elements. An IBD segment corresponds to a bicluster because individuals that possess the IBD segment are similar to each other at this segment. The similarity is given by identical minor alleles of tagSNVs. Fig. **??** depicts a bicluster identified in genotype data.

We employ the "Factor Analysis for Bicluster Acquisition" (FABIA) biclustering model **?**. In contrast to other biclustering methods such as BIMAX **?** and QUBIC **?**, FABIA can represent homozygous regions (two times the same IBD segment in one diploid individual) by its multiplicative bicluster model. Furthermore, FABIA can represent overlapping IBD segments (a different IBD segment on each chromosome) by its additive biclusters. FABIA can be applied to discrete phased or unphased genotype data but also to real values that correspond to minor allele likelihoods or the minor allele dosages. 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 **?**.

### 3.1.1 FABIA describes genotype data by IBD segments

FABIA describes an IBD segment in genotype data $\boldsymbol{X}$ by an outer product $\boldsymbol{z}\,\boldsymbol{\lambda}^T$ of two vectors $\boldsymbol{\lambda}$ and $\boldsymbol{z}$, where the vector $\boldsymbol{\lambda}$ indicates tagSNVs by non-zero values and the vector $\boldsymbol{z}$ indicates individuals that possess the IBD segment. FABIA can represent a homozygous region of an IBD segment by $z = 2$, that is, two occurrences of an IBD segment in one diploid individual. Fig. **??** visualizes this description of a genotype matrix by one IBD segment as an outer product.

A diploid individual may possess two IBD segments at a particular locus where genotyping sums up the occurrences of minor alleles. This fact is reflected by the additive FABIA model which sums up bicluster contributions. If we assume genotyping errors which are accounted for by $\boldsymbol{\Upsilon}$, 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 indicates which individuals possess an IBD segment; $\boldsymbol{\Lambda} \in \mathbb{R}^{p \times n}$ indicates IBD segment tagSNVs; $\boldsymbol{\Upsilon} \in \mathbb{R}^{l \times n}$ is an additive noise term; $n$ is the number of SNVs; $l$ is the number of individuals (or chromosomes for phased genotypes); $p$ is the number of IBD segments; $\boldsymbol{\lambda}_i \in \mathbb{R}^n$ is tagSNV indicator vector for the $i$-th IBD segment (the $i$-th row of $\boldsymbol{\Lambda}$); and $\boldsymbol{z}_i \in \mathbb{R}^l$ corresponds to the number of times each of the $l$ individuals contains the $i$-th IBD segment (the $i$-th column of $\boldsymbol{Z}$). The additive noise $\boldsymbol{\Upsilon}$ not only covers genotyping errors but also genotypes which cannot be explained by IBD segments. Such unexplained genotypes may arise from recently acquired SNVs, segments contained in only one individual, and IBD segments that are too short, or segments that are shared by too few individuals to be called.

According to Eq. (**??**), the $j$-th genotype $\boldsymbol{x}_j$, i.e., the $j$-th column of $\boldsymbol{X}$, is

$$\boldsymbol{x}_j \;=\; \sum_{i=1}^{p} \boldsymbol{\lambda}_i\, z_{ij} \;+\; \boldsymbol{\epsilon}_j \;=\; \boldsymbol{\Lambda}\,\tilde{\boldsymbol{z}}_j \;+\; \boldsymbol{\epsilon}_j\;, \tag{2}$$

where $\boldsymbol{\epsilon}_j$ is the $j$-th column of the noise matrix $\boldsymbol{\Upsilon}$ and $\tilde{\boldsymbol{z}}_j \;=\; (z_{1j}, \ldots, z_{pj})^T$ indicates which IBD segments are present in genotype $\boldsymbol{x}_j$ ($j$-th column of the matrix $\boldsymbol{Z}$ with length equal to the

number of IBD segments $p$). Recall that $\boldsymbol{z}_i = (z_{i1}, \ldots, z_{il})^T$ in Eq. (**??**) corresponds to the number of times each of the $l$ individuals contains the $i$-th IBD segment.

If we drop the index $j$ which indicates the individual, the formulation in Eq. (**??**) facilitates a generative interpretation by a factor analysis model with $p$ factors:

$$\boldsymbol{x} \;=\; \sum_{i=1}^{p} \boldsymbol{\lambda}_i \, \tilde{z}_i \;+\; \boldsymbol{\epsilon} \;=\; \boldsymbol{\Lambda} \, \tilde{\boldsymbol{z}} \;+\; \boldsymbol{\epsilon} \,, \tag{3}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ is the observed genotype. The vector of factors $\tilde{\boldsymbol{z}} = (\tilde{z}_1, \ldots, \tilde{z}_p)^T$ indicates which IBD segments are present in genotype $\boldsymbol{x}$, where component $\tilde{z}_i$ indicates how often the individual with genotype $x$ possesses the $i$-th IBD segment. $\boldsymbol{\epsilon} \in \mathbb{R}^n$ is the additive noise.

As illustrated in Fig. **??**, both the vector $\boldsymbol{z}_i$ and the vector $\boldsymbol{\lambda}_i$ should be sparse to describe an IBD segment. Sparse $\boldsymbol{z}_i$ means that only few individuals possess the IBD segment, which implies rare tagSNVs. Sparse $\boldsymbol{\lambda}_i$ means that only few SNVs are tagSNVs, which implies short IBD segments. Sparse $\boldsymbol{z}_i$ can be achieved if components $z_{ij}$ are sparse, that is, if the vector of factors $\tilde{\boldsymbol{z}}_j$ in Eq. (**??**) is sparse or, equivalently, the vector $\tilde{\boldsymbol{z}}$ in Eq. (**??**) is sparse. In contrast to standard factor analysis, FABIA's model selection is tailored to sparse factors and sparse loadings, which are essential for IBD 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 for the distribution of the factors $\tilde{\boldsymbol{z}}$ **?**:

$$p(\tilde{\boldsymbol{z}}) \;=\; \left(\tfrac{1}{\sqrt{2}}\right)^p \prod_{i=1}^{p} e^{-\sqrt{2}\,|\tilde{z}_i|} \tag{4}$$

$$p(\boldsymbol{\lambda}_i) \;=\; \left(\tfrac{1}{\sqrt{2}}\right)^n \prod_{k=1}^{n} e^{-\sqrt{2}\,|\lambda_{ki}|} \,. \tag{5}$$

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

$$p(\boldsymbol{x} \mid \boldsymbol{\Lambda}, \boldsymbol{\Psi}) \;=\; \int p(\boldsymbol{x} \mid \tilde{\boldsymbol{z}}, \boldsymbol{\Lambda}, \boldsymbol{\Psi}) \, p(\tilde{\boldsymbol{z}}) \, d\tilde{\boldsymbol{z}} \,. \tag{6}$$

Therefore, the model selection of FABIA 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 **?????**. The idea of the variational approach is to express the prior $p(\tilde{\boldsymbol{z}})$ by the maximum

$$p(\tilde{\boldsymbol{z}}) \;=\; \max_{\boldsymbol{\xi}} p(\tilde{\boldsymbol{z}} \mid \boldsymbol{\xi}) \tag{7}$$

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

$$p(\tilde{\boldsymbol{z}}) \;=\; \int p(\tilde{\boldsymbol{z}} \mid \boldsymbol{\xi}) \, d\mu(\boldsymbol{\xi}) \,. \tag{8}$$

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

$$\hat{\boldsymbol{\xi}} \;=\; \arg\max_{\boldsymbol{\xi}} p(\boldsymbol{\xi} \mid \boldsymbol{x}) \,. \tag{9}$$

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

If we denote the $j$-th 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 **?**:

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

$$\mathrm{E}\big(\boldsymbol{z}_j\, \boldsymbol{z}_j^T \mid \boldsymbol{x}_j\big) \;=\; \big(\boldsymbol{\Lambda}^T\, \boldsymbol{\Psi}^{-1}\, \boldsymbol{\Lambda} \;+\; \boldsymbol{\Xi}_j^{-1}\big)^{-1} \;+\; \mathrm{E}\big(\boldsymbol{z}_j \mid \boldsymbol{x}_j\big)\, \mathrm{E}(\boldsymbol{z}_j \mid \boldsymbol{x}_j)^T \;, \tag{11}$$

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

$$\boldsymbol{\xi}_j \;=\; \mathrm{diag}\left(\sqrt{\mathrm{E}(\boldsymbol{z}_j\, \boldsymbol{z}_j^T \mid \boldsymbol{x}_j)}\right) \;. \tag{12}$$

The variational M-step is **?**

$$\boldsymbol{\Lambda}^{\mathrm{new}} \;=\; \frac{\frac{1}{l}\, \sum_{j=1}^{l} \boldsymbol{x}_j\, \mathrm{E}(\boldsymbol{z}_j \mid \boldsymbol{x}_j)^T \;-\; \frac{\alpha}{l}\, \boldsymbol{\Psi}\, \mathrm{sign}(\boldsymbol{\Lambda})}{\frac{1}{l} \sum_{j=1}^{l} \mathrm{E}(\boldsymbol{z}_j\, \boldsymbol{z}_j^T \mid \boldsymbol{x}_j)} \tag{13}$$

$$\mathrm{diag}\,(\boldsymbol{\Psi}^{\mathrm{new}}) \;=\; \boldsymbol{\Psi}^{\mathrm{EM}} \;+\; \mathrm{diag}\Big(\frac{\alpha}{l}\, \boldsymbol{\Psi}\, \mathrm{sign}(\boldsymbol{\Lambda})(\boldsymbol{\Lambda}^{\mathrm{new}})^T\Big) \;, \tag{14}$$

$$\boldsymbol{\Psi}^{\mathrm{EM}} \;=\; \mathrm{diag}\Big(\frac{1}{l} \sum_{j=1}^{l} \boldsymbol{x}_j\, \boldsymbol{x}_j^T \;-\; \boldsymbol{\Lambda}^{\mathrm{new}} \frac{1}{l} \sum_{j=1}^{l} \mathrm{E}\,(\boldsymbol{z}_j \mid \boldsymbol{x}_j)\; \boldsymbol{x}_j^T\Big). \tag{15}$$

The parameter $\alpha$ controls the degree of sparseness (an expectation of how rare the SNVs that tag the IBD segment are) and can be introduced as a parameter of the Laplacian prior of the factors **?**.

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

### 3.1.2 Adaptation of FABIA for IBD 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 by IBD segments. The genotype matrix $\boldsymbol{X}$ is non-negative as are both the indicator matrix of tagSNVs $\boldsymbol{\Lambda}$ and indicator matrix of IBD segments in individuals $\boldsymbol{Z}$. Regarding these constraints, we modified FABIA to enforce non-negative loadings $\boldsymbol{\Lambda}$. If, for individual $j$, both genotype data $\boldsymbol{x}_j$ 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. Consequently, the new loading matrix $\boldsymbol{\Lambda}^{\mathrm{new}}$ is non-negative according to Eq. (**??**). Note that the prior term $\frac{\alpha}{l}\, \boldsymbol{\Psi}\, \mathrm{sign}(\boldsymbol{\Lambda})$ in the update rule Eq. (**??**) is not allowed to change the signs of the loadings $\boldsymbol{\Lambda}$ in one update step. Therefore, it is sufficient to initialize $\boldsymbol{\Lambda}$ by positive values to enforce non-negative factors and loadings. $\boldsymbol{Z}$ is estimated by the posterior mean $\mathrm{E}\big(\boldsymbol{z}_j \mid \boldsymbol{x}_j\big)$ and is used to identify individuals that possess an IBD segment.

To allow IBD segment detection in large sequencing data, we developed a specialized sparse matrix algebra. This sparse matrix algebra only considers non-zero values with their indices for vector and matrix computations. In Eqs. (**??**)-(**??**) the values of the genotype vectors $\boldsymbol{x}_j$ and the values of the loading matrix $\boldsymbol{\Lambda}$ are sparse. Our sparse matrix algebra not only allows for multiplying a sparse vector by a dense vector (as usual in sparse matrix computations), but also for multiplying two sparse vectors.

To further speed up the computation, we extended FABIA to an iterative version, where, in each iteration, $p$ biclusters are detected. 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 short IBD segments. Components of $\boldsymbol{\lambda}_i$ indicate to which degree tagSNVs tag the $i$-th IBD segment. In particular, these components measure how many individuals possess the IBD segment (more precisely the corresponding tagSNV). Components of $\boldsymbol{z}_i$ indicate how often a specific IBD segment is present in one multiploid individual. Additionally components of $\boldsymbol{z}_i$ also indicate how well an individual segment matches an IBD segment as individual segments may contain genotyping errors or may be broken up during meiosis.

## 3.2 Extraction of IBD segments from FABIA models

FABIA biclustering does not consider that an IBD segment consists of contiguous nucleotides. This essential information is incorporated into HapFABIA in the second stage. SNVs that tag an IBD segment are locally accumulated. FABIA may have accidentally merged separated IBD segments that are shared by the same individuals or IBD segments that are located on different chromosomes. The second stage, therefore disentangles falsely merged IBD segments, prunes IBD segments from spurious SNVs, and finally joins parts of single long IBD segments.

As mentioned above, 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 tag an IBD segment agglomerate locally in this 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 minor allele frequency of SNVs is known. A low $p$-value hints at local agglomerations of bicluster SNVs stemming from an IBD segment.

We propose a four-step procedure to extract IBD segments from FABIA models:

1. Identify local agglomerations of correlated SNVs based on a binomial test;
2. Disentangle IBD segments and re-assigning individuals or chromosomes to IBD segments;
3. Prune IBD segments off SNVs with spuriously correlations based on an exponential test for a long genomic distance;
4. Join similar IBD segments stemming from long IBD segments that were divided by the bins at the first step.

**Step 1:** FABIA model selection is independent of the order of the SNVs. Therefore, spuriously correlated SNVs are unlikely to agglomerate at a DNA locus, whereas tagSNVs do, as they are within an IBD segment. To detect agglomerations, we compute histogram counts of FABIA model SNVs within bins where bins with large counts are assumed to contain IBD segments. For computing the histogram of counts of FABIA model SNVs, we consider those SNVs for which the FABIA model parameter $\boldsymbol{\lambda}_i$ is largest (threshold "Lt" with 10% being the default value). Large $\boldsymbol{\lambda}$ values ensure IBD segments that are shared by multiple individuals. These segments are therefore more reliable. The HapFABIA parameter "IBDlength" determines the maximal length of IBD

segments. The histogram bin size in number of SNVs (all SNVs and not only model SNVs) is computed from "IBDlength" using the average genomic distance between adjacent SNVs. To account for IBD segments that exceed the borders of the bins, the histogram is computed a second time with bins shifted by half the bin size.

The histogram bins with more model SNVs than expected by chance are assumed to contain IBD segments. We select bins for which the model SNV count exceeds the expected value by a binomial test for random counts. We need to compute how many model SNVs are expected in a bin if they are random and not in an IBD segment. Thus, we have to compute the probability of observing $k$ or more bin counts by chance. Let $p$ be the probability of a random minor allele match between $t$ individuals. If $n$ SNVs are in a bin, the probability of observing $k$ model SNVs by chance is given by one minus the binomial distribution $F(k; n, p)$:

$$1 \; - \; F(k-1; n, p) \; = \; \Pr(K \geq k) \; = \; \sum_{i=k}^{n} \binom{n}{i} \, p^i \, (1-p)^{n-i} \, . \tag{16}$$

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 assumed 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 of model SNVs by chance in at least one bin is

$$b \binom{l}{t} \sum_{i=k}^{n} \binom{n}{i} \, q^{it} \, \left(1 - q^t\right)^{n-i} \, , \tag{17}$$

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 in Eq. (**??**) is below the threshold "thresCount", the according bin is selected for IBD segment extraction because more FABIA model SNVs are in this bin than expected by chance. If $k_{\text{min}}$ is the minimum $k$ for which Eq. (**??**) is below the threshold "thresCount", then all bins with model SNV counts $k \geq k_{\text{min}}$ are selected. In our experiments, we allow IBD segments of only two individuals (standard IBD), 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 and therefore different IBD segments. First, those model SNVs that contributed to the count of the selected bin are assigned to it. 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 assigned individuals are indeed similar to each other. The parameter "Zt" (default 0.2) gives the percentage of top $z$-scores that are considered.

**Step 2:** In this step, IBD segments in a selected bin are disentangled, where only SNVs and individuals are considered that have been assigned to the bin. An IBD segment is initialized by two core individuals that are identical at $m$ or more minor alleles. The number $m$ is computed as $m = \text{mintagSNVsFactor} \times k_{\text{min}}$. All individuals that are identical in at least $m$ minor alleles to one of the two IBD core individuals are classified as possessing the IBD segment. The tagSNVs of this IBD segment are model SNVs that have their minor allele in at least 2 individuals that possess the IBD segment.

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

**Step 3:** This step prunes IBD segment borders of SNVs that have spurious correlations to the IBD segment. Spurious correlations may still be present in a bin leading to an overestimation of the segment 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 an IBD segment 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:** IBD segments which are very similar to each other are merged. In this way, long IBD segments that were divided by the bins into smaller parts are reconstructed. Note that IBD segments longer than given by "IBDlength" can be detected. In order to compute similarities, we assess how many tagSNVs and individuals of the smaller IBD segment are explained by the larger IBD segment. This criterion is expressed by the "overlap coefficient"

$$O(A, B) \;=\; \frac{|A \cap B|}{\min\{|A|, |B|\}} \,. \tag{18}$$

Using the overlap coefficient for both tagSNVs and individuals, we define a distance-like measure between IBD segments $\mathrm{IBD}_1$ and $\mathrm{IBD}_2$ by

$$D(\mathrm{IBD}_1, \mathrm{IBD}_2) \;=\; 1 \;-\; O(S_{\mathrm{IBD}_1}, S_{\mathrm{IBD}_2})\, O(I_{\mathrm{IBD}_1}, I_{\mathrm{IBD}_2}) \,, \tag{19}$$

where $S_{\mathrm{IBD}_i}$ and $I_{\mathrm{IBD}_i}$ are the tagSNVs that tag IBD segment $\mathrm{IBD}_i$ and individuals possessing IBD segment $\mathrm{IBD}_i$, respectively. Using the measure $D$, IBD segments are clustered by hierarchical clustering using complete linkage. IBD segments are merged if their segments are clustered together below a cutting height of 0.8.

## 3.3 Further Advantages of HapFABIA

HapFABIA further has the following two advantages over existing IBD detection methods:

1. If an IBD segment is known, but its tagSNVs are unknown, existing IBD detections methods have still difficulties to identify IBD sharing among individuals. Most tagSNVs are separated by common, private, or random SNVs, but also by tagSNVs of other IBD segments. Moreover, the distances between tagSNVs of an IBD segment vary much, both in terms of genomic distance and the number of tagSNVs between them. These complex data characteristics impair the detecting abilities of existing IBD detection methods. HapFABIA's biclustering approach, however, does not consider the order of SNVs in first place, therefore, it is well suited for detecting non-consecutive tagSNVs with varying distances between them.

2. HapFABIA is well suited both for phased and unphased genotype data. It represents homozygous regions (i.e. two occurrences of an IBD segment in one diploid individual) by means of its factor. Overlapping IBD segments in a diploid individual (i.e. different IBD segments on each chromosome) can be represented by the additive FABIA model. The results

of HapFABIA for phased and unphased genotypes hardly differ for short IBD segments because homozygous regions are rarely observed. Another important aspect is that HapFABIA can also be applied to minor allele likelihoods or minor allele dosages. We applied HapFABIA to data from the Korean Personal Genome Project (KPGP, http://opengenome.net), which was merged with the 1000 Genomes Project data. In this analysis we only used unphased genotype data and re-discovered the IBD segments which were already found in the 1000 Genomes Project data. Additionally, we discovered more Asian specific IBD segments. Results can be found at http://www.bioinf.jku.at/research/short-IBD.

# 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,w=NULL,intervv=50,off=0)
> print(h1$counts)
```

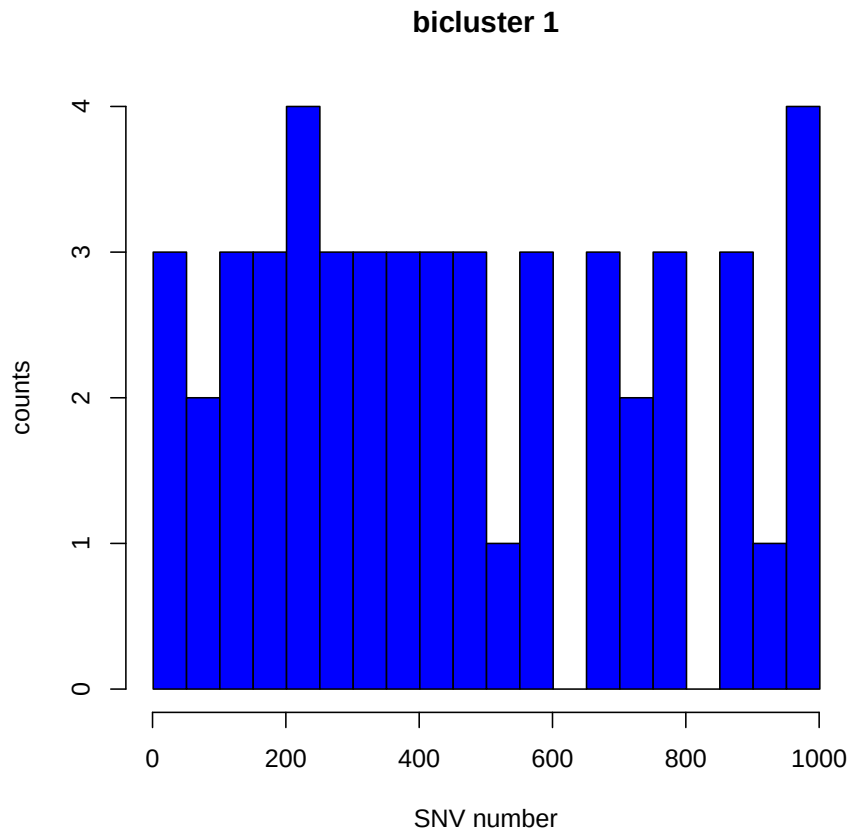
```
 [1]  9  5  7  7  5  4  4  4  5  4  3  6  1  5  6  4  1  4
[19]  5 11
```

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

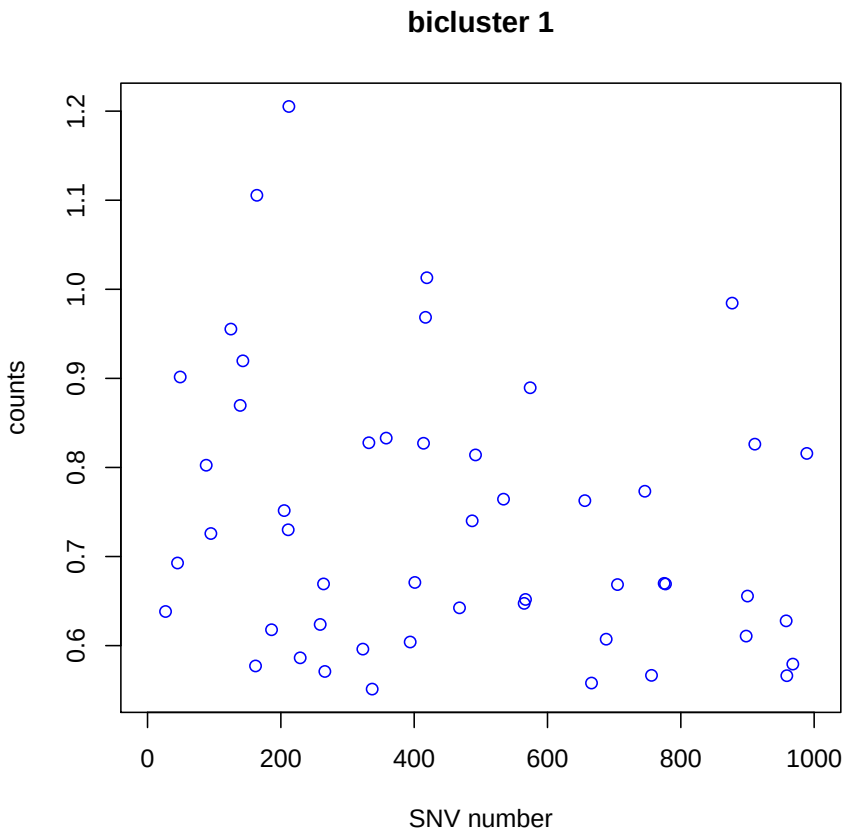
```
 [1] 4 2 6 5 4 3 3 4 3 3 1 4 0 4 3 3 1 3 3 7
```

fabia loadings can be plotted to identify locations of accumulations:

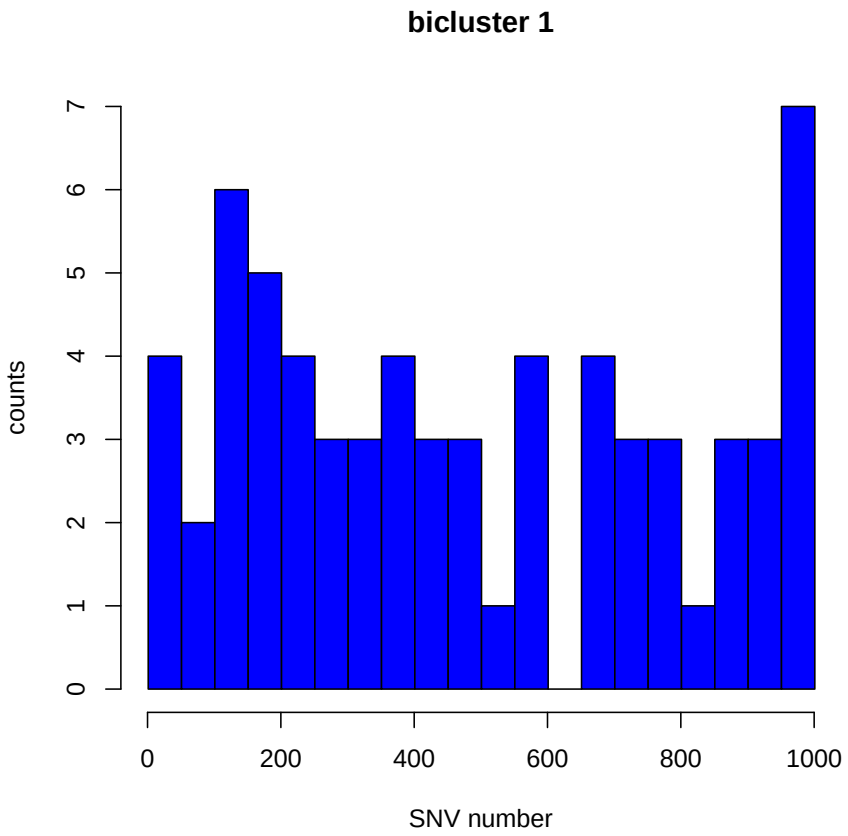
```
> data(res)
> plotL(res,n=1,p=0.95,w=NULL,type="histogram",intervv=50,off=0,t="p",cex=1)
```



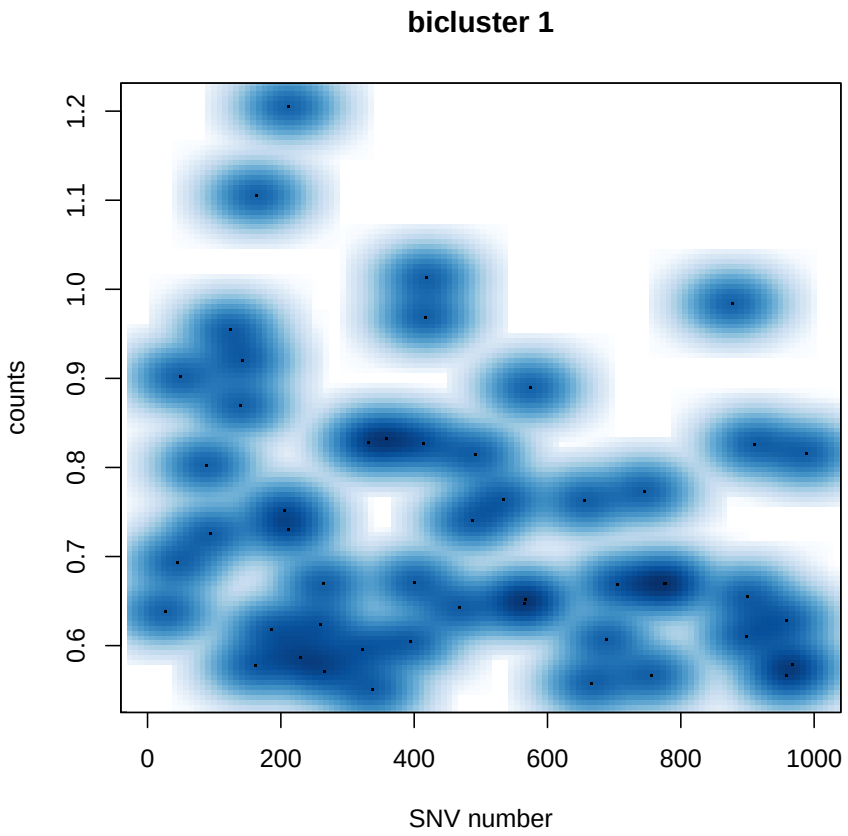

```
> data(res)
> plotL(res,n=1,p=0.95,w=NULL,type="points",intervv=50,off=0,t="p",cex=1)
```



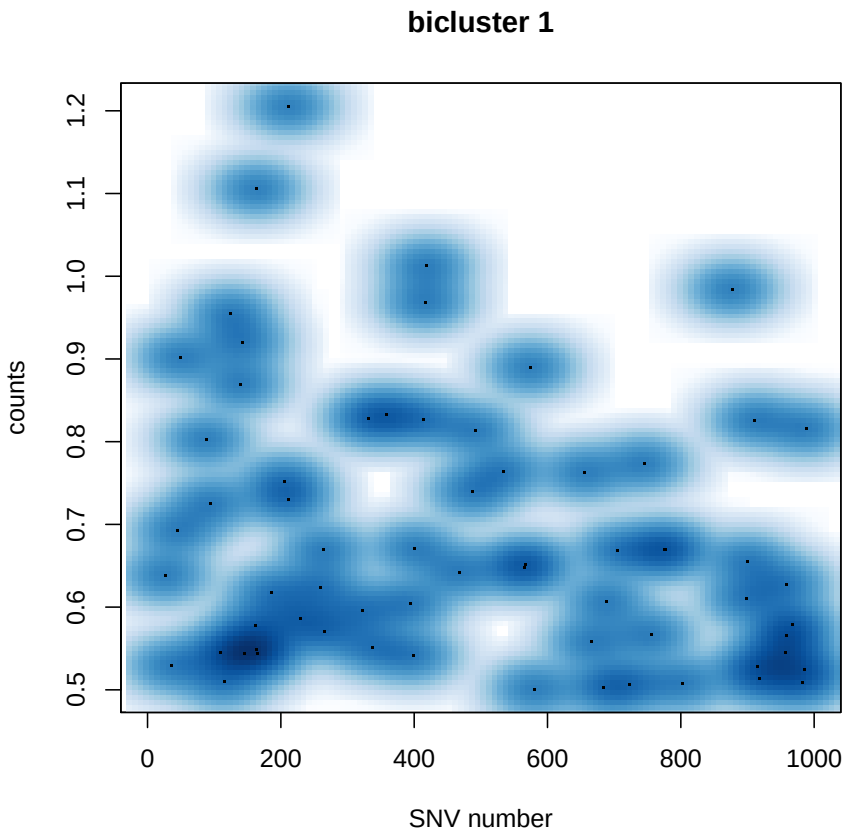

```
> data(res)
> plotL(res,n=1,p=NULL,w=0.5,type="histogram",intervv=50,off=0,t="p",cex=1)
```




```
> data(res)
> plotL(res,n=1,p=0.95,w=NULL,type="smooth",intervv=50,off=0,t="p",cex=1)
```




```
> data(res)
> plotL(res,n=1,p=NULL,w=0.5,type="smooth",intervv=50,off=0,t="p",cex=1)
```




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,n=1,LZ="L",indices=TRUE,p=0.95,w=NULL)

 [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,n=1,LZ="L",indices=TRUE,p=NULL,w=0.95)

[1] 125 164 212 417 419 877

> topLZ(res,n=1,LZ="Z",indices=TRUE,p=0.95,w=NULL)

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

> topLZ(res,n=1,LZ="Z",indices=TRUE,p=NULL,w=0.4)
```

```
[1]   6 102

> topLZ(res,n=1,LZ="L",indices=FALSE,p=0.95,w=NULL)

 [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,n=1,LZ="L",indices=FALSE,p=NULL,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,w=NULL)

 [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,p=NULL,w=0.4)

[1] 0.4015947 0.7713440
```