RAbHIT: R Antibody Haplotype Infrence Tool

Ayelet Peres & Moriah Gidoni & Gur Yaari

Last modified 2020-01-29

Contents

Introduction

Analysis of antibody repertoires by high throughput sequencing is of major importance in understanding adaptive immune responses. Our knowledge of variations in the genomic loci encoding antibody genes is incomplete, mostly due to technical difficulties in aligning short reads to these highly repetitive loci. The partial knowledge results in conflicting V-D-J gene assignments between different algorithms, and biased genotype and haplotype inference. Previous studies have shown that haplotypes can be inferred by taking advantage of IGHJ6 heterozygosity, observed in approximately one third of the population [1], [2].

Here we provide a robust novel method for determining V-D-J haplotypes by adapting a Bayesian framework, **RAbHIT**. Our method extends haplotype inference to IGHD and IGHV based analysis, thereby enabling inference of complex genetic events like deletions and copy number variations in the entire population. It calculates a Bayes factor, a number that indicates the certainty level of the inference, for each haplotyped gene.

More details can be found here:

Gidoni, Moriah, et al. "Mosaic deletion patterns of the human antibody heavy chain gene locus shown by Bayesian haplotyping." Nature Communications 10.1 (2019): 628.

Input

RAbHIT requires two main inputs:

1. Pre-processed antibody repertoire sequencing data with heterozygosity in at least one gene
2. Database of germline gene sequences

Antibody repertoire sequencing data is in a data frame format. Each row represents a unique observation and columns represent data about that observation. The names of the required columns are provided below along with a short description.

Column Name	Description
`SUBJECT`	Subject name
`V_CALL`	(Comma separated) name(s) of the nearest V allele(s) (IMGT format)
`D_CALL`	(Comma separated) name(s) of the nearest D allele(s)
`J_CALL`	(Comma separated) name(s) of the nearest J allele(s)

An example dataset is provided with RAbHIT. It contains unique naive b-cell sequences, from multiple individuals. One individual in the example data set appears twice, once with full V coverage and once partial V coverage (I5 and I5_FR2 respectively).

The database of germline sequences should be provided in a FASTA format with sequences gaped according to the IMGT numbering scheme[3]. IGHV, IGHD, and IGHJ alleles in the IMGT database (release 2018-12-4) are provided with this package (HVGERM, HDGERM, and HJGERM). We removed the following IGHV and IGHD alleles, as they are duplicated from other allele assignments (All duplicates are mentioned in IMGT).

V/D allele	V/D allele duplicated (removed)
IGHV1-69*01	IGHV1-69D*01
IGHV2-70*04	IGHV2-70D*04
IGHV3-23*01	IGHV3-23D*01
IGHV3-30*04	IGHV3-30-3*03
IGHV3-29*01	IGHV3-30-42*01
IGHV3-30*18	IGHV3-30-5*01
IGHV3-30*02	IGHV3-30-5*02
IGHD4-11*01	IGHD4-4*01
IGHD5-18*01	IGHD5-5*01

Pre-processing of the data

To obtain the most reliable result we suggest following the data pre-processing steps below.

1. Align each sequence to infer the germline VDJ alleles.
2. Remove duplicated sequences (pRESTO[4]).
3. Infer novel alleles (TIgGER[5]).
4. Re-align the sequences with the extended V reference including the new alleles.
5. Infer the individual genotype according to the following:
   - Cell type:
     - If the dataset is from PBMCs then, to avoid memory cell influence on the results, it is important to infer clones prior to genotyping. After inferring clones, choose a representative sequence with the fewest mutations for each clone.
     - If the dataset is from naive B-cells then no additional reduction is needed.
   - V coverage:
     - If the dataset sequences are with partial V coverage, first determine reliable gene and alleles, then change the non-reliable alleles' annotation and genotype.
     - If the dataset is from naive B-cells then no additional reduction is needed.
6. Re-align the sequences with the individual personal genotype as a reference.

7. It is recommended to filter the sequences to only V genes with ≤ 3 mutations and no mutations in D gene.
8. For best haplotype inference results, ideally the dataset final unique VDJ count should be more than 2000 sequences.

Running RAbHIT

To load RAbHIT we'll run the following:

```
library(rabhit)
```

The functions provided by this package can be used to perform any combination of the following:

1. Infer haplotype by anchor gene
2. Infer D/J single chromosome deletions by the Vpooled approach
3. Infer double chromosome deletions by relative gene usage
4. Graphical output of the inferred haplotype
5. Graphical output of the inferred deletions

Infer haplotype by anchor gene

Haplotype with J6 as anchor

An individual's haplotype can be inferred using the function `createFullHaplotype`. The function infers the haplotype based on the provided anchor gene. Using this function, a contingency table is created for each gene, *from which a strand is inferred for each allele*. The user can set the anchor gene for haplotyping as well as the column for which a haplotype should be inferred.

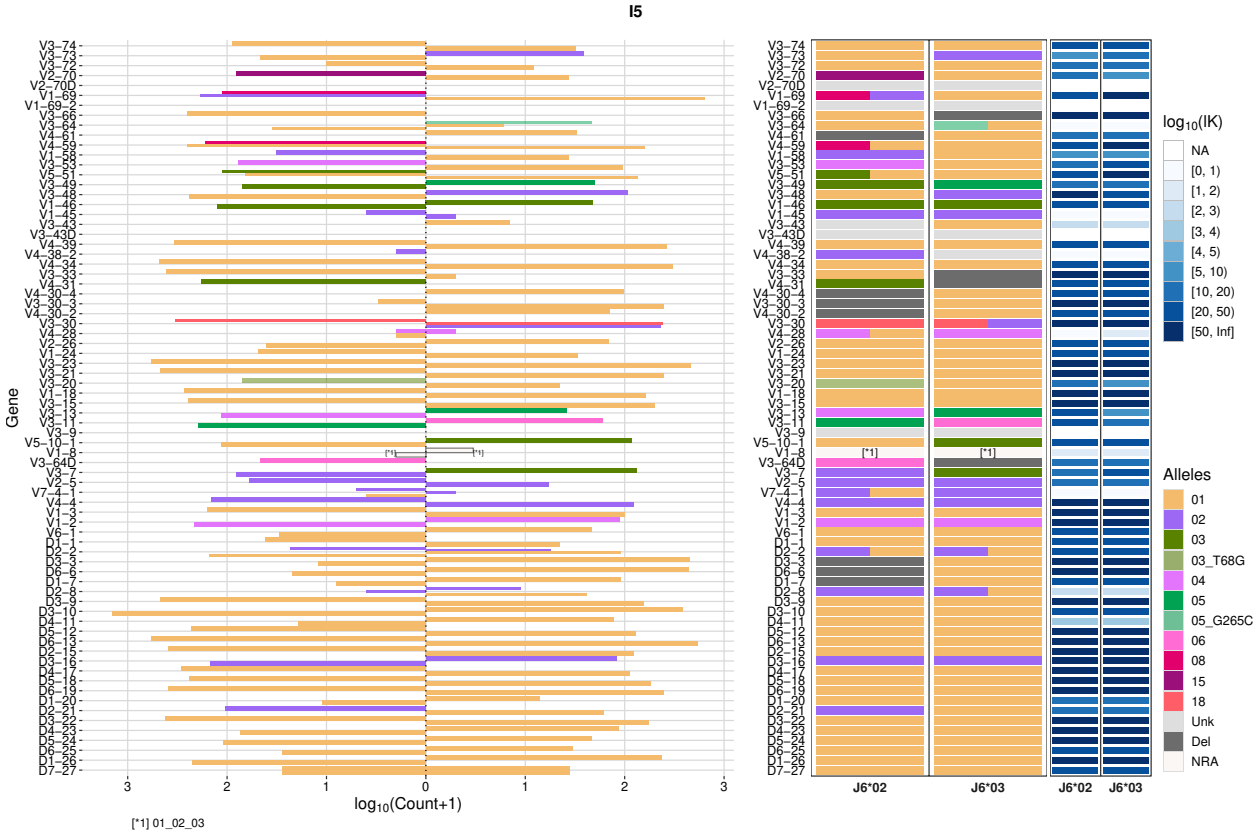
```
# Load example sequence data and example germline database
data(samples_db, HVGERM, HDGERM)
# Selecting a single individual
clip_db <- samples_db[samples_db$SUBJECT=='I5', ]
```

```
# Inferring haplotype using J6 as anchor
haplo_db_J6 <- createFullHaplotype(clip_db, toHap_col=c("V_CALL","D_CALL"),
                                   hapBy_col="J_CALL", hapBy="IGHJ6",
                                   toHap_GERM=c(HVGERM, HDGERM))

head(haplo_db_J6,3)
```

```
##   SUBJECT     GENE IGHJ6_02 IGHJ6_03 ALLELES PRIORS_ROW PRIORS_COL COUNTS1
## 1      I5 IGHV4-59    01,08       01   01,08  0.58,0.42  0.67,0.33 248,156
## 2      I5 IGHV4-34       01       01      01  0.58,0.42          1 483,306
## 3      I5 IGHV1-18       01       01      01  0.58,0.42          1 268,161
##                  K1 COUNTS2               K2 COUNTS3   K3 COUNTS4   K4
## 1 196.832971198849   163,3 33.5489056273634     <NA> <NA>     <NA> <NA>
## 2 38.6912184865818    <NA>             <NA>     <NA> <NA>     <NA> <NA>
## 3 200.298250354403    <NA>             <NA>     <NA> <NA>     <NA> <NA>
```

We can plot the haplotype map using the `plotHaplotype` function. Each row represents a different gene and the genes are ordered by the chromosome location. Each color represents the different alleles, light gray represents an unknown, dark gray a deletion and off-white a non-reliable allele annotation. The blues represent the confidence level of the inference (lK). The most left panels show the count for each allele on each chromosome. The middle panels show the allele inference for each gene on each chromosome, The most right panels show the confidence level of the inference. non reliable alleles are annotated with `[*]` and a number and are found below the graph.

```
# Plotting the haplotype map
plotHaplotype(haplo_db_J6)
```



Haplotype with D2-21 as anchor

One of the advantages of RAbHIT, is the ability to infer haplotype by other anchor genes than J6. RAbHIT offers utilizing any gene as anchor, however, we recommend that the fraction of the least frequent allele of the anchor gene will be above 0.3. So far only J6, D2-8, and D2-21 has been proven to infer the heavy chain loci correctly. Here, we chose a single individual from our example dataset with heterozygousity in J6 and D2-21, inferred haplotypes according to both anchors, compared them using the `hapDendo` function, and calculated the Jaccard distance between the inferred haplotypes.

```
# Inferring haplotype using D2-21 as anchor
haplo_db_D2_21 <- createFullHaplotype(clip_db, toHap_col="V_CALL",
                        hapBy_col="D_CALL", hapBy="IGHD2-21", toHap_GERM=HVGERM)
```

To combine the two data frames we need to associate each of the D2-21 alleles with J6 alleles. We can do that from the haplotype inference with the J6 as anchor

```
haplo_db_J6[haplo_db_J6$GENE == "IGHD2-21", ]
```

```
##    SUBJECT     GENE IGHJ6_02 IGHJ6_03 ALLELES PRIORS_ROW PRIORS_COL COUNTS1
## 69      I5 IGHD2-21       02       01   01,02  0.58,0.42  0.43,0.57    2,61
##                   K1 COUNTS2               K2 COUNTS3   K3 COUNTS4   K4
## 69 18.8792593606633   104,4 17.9911570372869    <NA> <NA>    <NA> <NA>
```

D2-21*02 goes with J6*02 and that D2-21*01 goes with J6*03. Now to bind the data frames together, will change the sample name to match the anchor gene used in each haplotype, and change the anchor gene columns to a generic name.

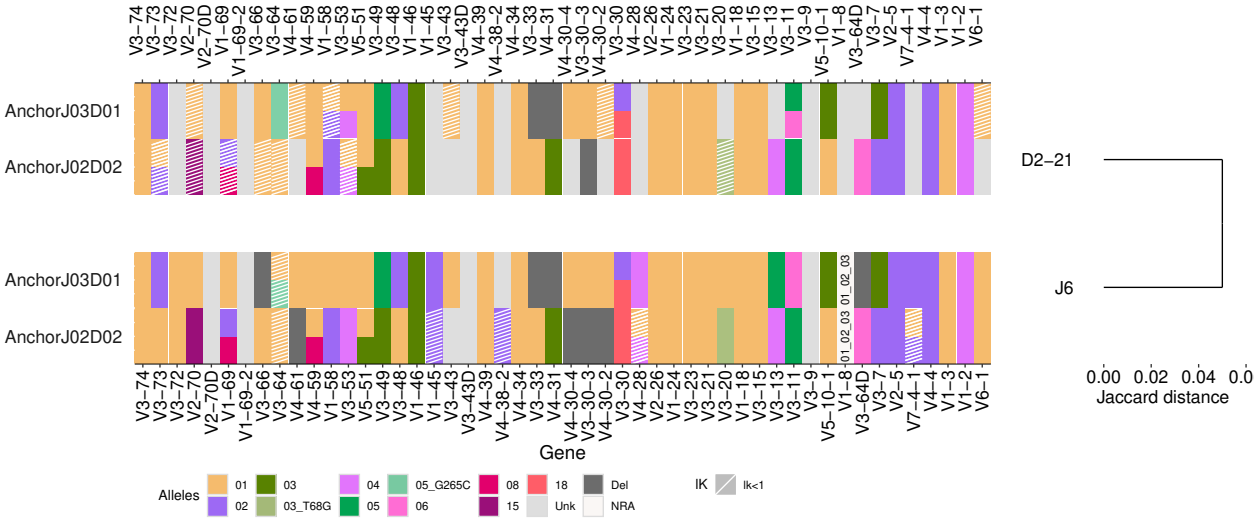
```
# rename the subject
haplo_db_J6$SUBJECT <- 'J6'
haplo_db_D2_21$SUBJECT <- 'D2-21'

# change the anchor gene columns
# For D2-21*01 and J6*03 we will change the column to Anchor_J03_D01
names(haplo_db_J6)[which(names(haplo_db_J6)=='IGHJ6_03')] <- "AnchorJ03D01"
names(haplo_db_D2_21)[which(names(haplo_db_D2_21)=='IGHD2-21_01')] <- "AnchorJ03D01"

# For D2-21*02 and J6*02 we will change the column to Anchor_J02_D02
names(haplo_db_J6)[which(names(haplo_db_J6)=='IGHJ6_02')] <- "AnchorJ02D02"
names(haplo_db_D2_21)[which(names(haplo_db_D2_21)=='IGHD2-21_02')] <- "AnchorJ02D02"

# Subseting the haplo_db_J6 dataset to include only the V genes
haplo_db_J6 <- haplo_db_J6[grep('IGHV',haplo_db_J6$GENE),]

# Combining the datasets rowwise
haplo_comb <- rbind(haplo_db_J6,haplo_db_D2_21)
```

Now, we can compare the haplotypes using the `hapDendo` function.

```
# Plot the haplotype inferred dendogram
hapDendo(haplo_comb)
```

```
##  [1] 1.00 1.00 1.00 1.00  NaN 1.00 0.75 1.00  NaN  NaN 1.00 1.00 1.00  NaN 0.75
```

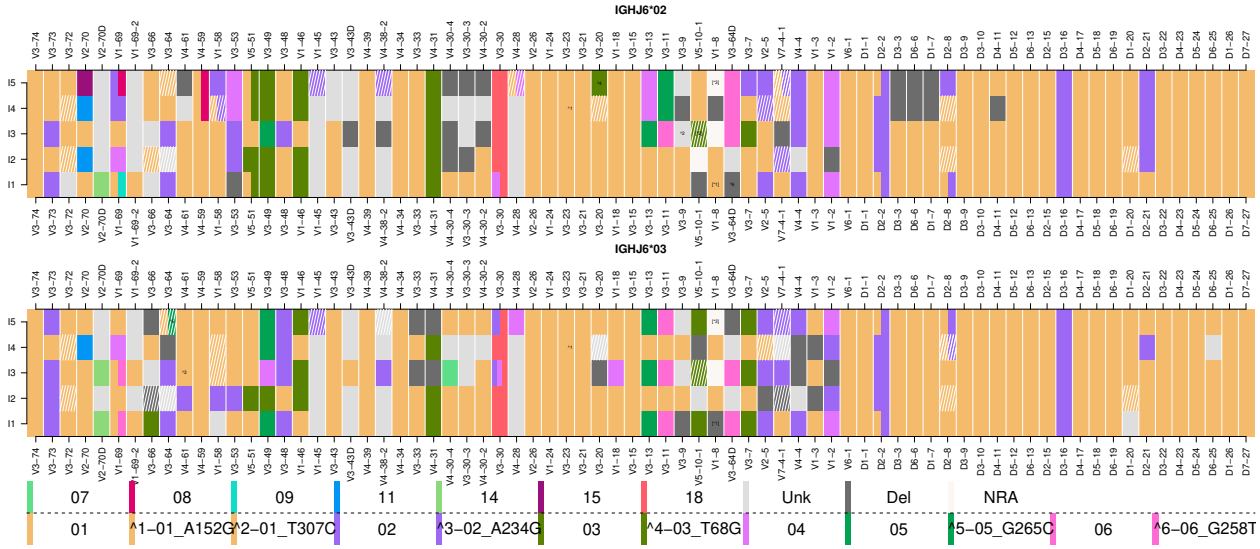
```
## [16] 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00  NaN 1.00 1.00 0.50 0.75 1.00
## [31] 0.50 1.00  NaN 0.75 1.00  NaN  NaN 1.00 1.00 1.00 1.00  NaN 1.00 1.00 1.00
## [46] 1.00 1.00 1.00 1.00  NaN
```



Compare multiple haplotypes

The `createFullHaplotype` function infers haplotypes for multiple individuals at once, using the `SUBJECT` column to distinguish between the different individuals. We can then compare the individual haplotypes using the `hapHeatmap` function. This function generates a heatmap of the alleles inferred for each chromosome. The novel alleles are annotated with ^ and a number, while the non reliable alleles are annotated with `[*]` and a number and are found below the graph. This new annotation allows to distinguish better between different alleles.

```
# Removing the individual I5_FR1 with the paritial V coverage sequence.
clip_dbs <- samples_db[samples_db$SUBJECT!='I5_FR2', ]

# Inferred haplotype summary table
haplo_db <- createFullHaplotype(clip_dbs, toHap_col=c("V_CALL","D_CALL"),
                        hapBy_col="J_CALL", hapBy="IGHJ6", toHap_GERM=c(HVGERM, HDGERM))
# Plot the haplotype inferred heatmap
p.list <- hapHeatmap(haplo_db)

# The function return a list with the plot and the optimal width and height
# we can use both parametrs to render the plot to the desired size.
width <- p.list$width
height <- p.list$height
```



[*2] 01_03_04, [*1] 01_02, [*3] 01_02_03

Inferring double chromosome deletion by relative gene usage

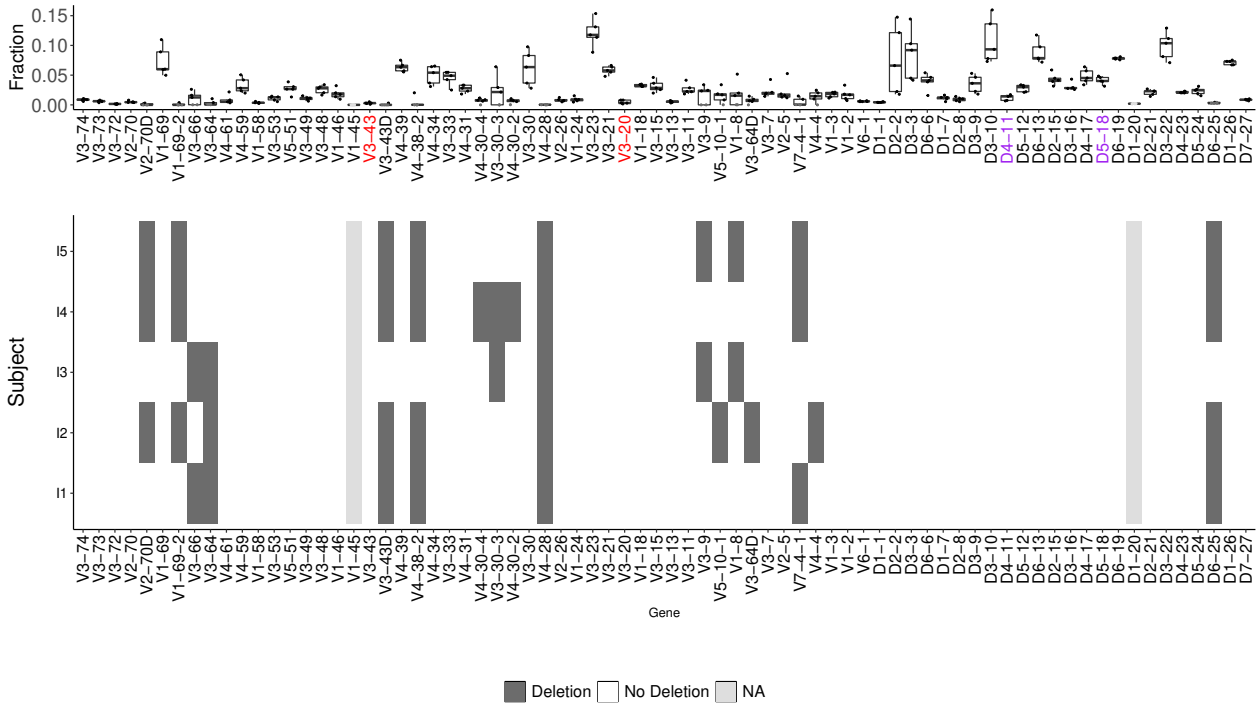
Gene usage tends to vary between individuals, and in some cases the relative gene usage of certain individuals are much lower than the rest of the population. To asses whether the low frequency arises from a deleted gene, a binomial test described in Gidoni *et al.* (2018)[6] was implemented. There it was checked whether the relative gene usage of a specific gene or set of genes in an individual is lower than a chosen cutoff. For example for the IGHV genes, the chosen cutoff was 0.001. The `deletionsByBinom` function implements the binomial test and returns the detected gene deletion for a certain individual.

```
# Infering double choromosome deletions
del_binom_db <- deletionsByBinom(clip_dbs)
head(del_binom_db)
```

```
## # A tibble: 6 x 6
##   SUBJECT GENE        FRAC  CUTOFF  PVAL DELETION
##   <chr>   <fct>      <dbl>   <dbl> <dbl> <fct>
## 1 I3      IGHV3-74 0.0105  0.00471     1 No Deletion
## 2 I1      IGHV3-74 0.00582 0.00471     1 No Deletion
## 3 I2      IGHV3-74 0.00793 0.00471     1 No Deletion
## 4 I4      IGHV3-74 0.00955 0.00471     1 No Deletion
## 5 I5      IGHV3-74 0.00879 0.00471     1 No Deletion
## 6 I3      IGHV3-73 0.00554 0.00198     1 No Deletion
```

For visualizing the deletion detected by `deletionsByBinom`, the `plotDeletionsByBinom` can be used. It is recommended to use this function for multiple individuals.

```
# Don't plot IGHJ
del_binom_db <- del_binom_db[grep('IGHJ', del_binom_db$GENE, invert = T),]
# Inferred deletion summary table
plotDeletionsByBinom(del_binom_db)
```



The detection of a double chromosome deletion from the function can be then used for haplotype inference. This prior knowledge of deletion can raise the certainty level in the inference of the genes for where a deletion was detected. The `createFullHaplotype` receives a vector of the deleted genes detected. V gene labels marked in red represent low expressed genes for which deletions are inferred with lowly certainty. D gene labels marked in purple represent indistinguishable genes due to high sequence similarity, therefore the alignment call is less reliable.

Interactive Haplotype inference with double chromosome deletions

Individuals with partial V coverage tend to have multiple gene or allele assignments. This multiplicity requires special attention, as it can hinder the haplotype inference results by introducing biases. To infer the haplotypes correctly from those datasets we recommend following the initial pre-processing steps and detecting non reliable V genes using the `nonReliableVGenes` function.

The individual in the example dataset with the partial V coverage is I5_FR2. We shortened the sequences of individual I5 using pRESTO[4] `MaskPrimers` function with BIOMED-2 framework 2 (FR2) primers.

```
# Selecting a single individual with partial V coverage
clip_db <- samples_db[samples_db$SUBJECT=='I5_FR2', ]

# Detecting non reliable genes
nonReliable_Vgenes <- nonReliableVGenes(clip_db)
```

As mentioned above the double chromosome deletions can be used within the haplotype inference. For the partial V coverage dataset it is important to input the `deletionsByBinom` with the detect non reliable V gene list.

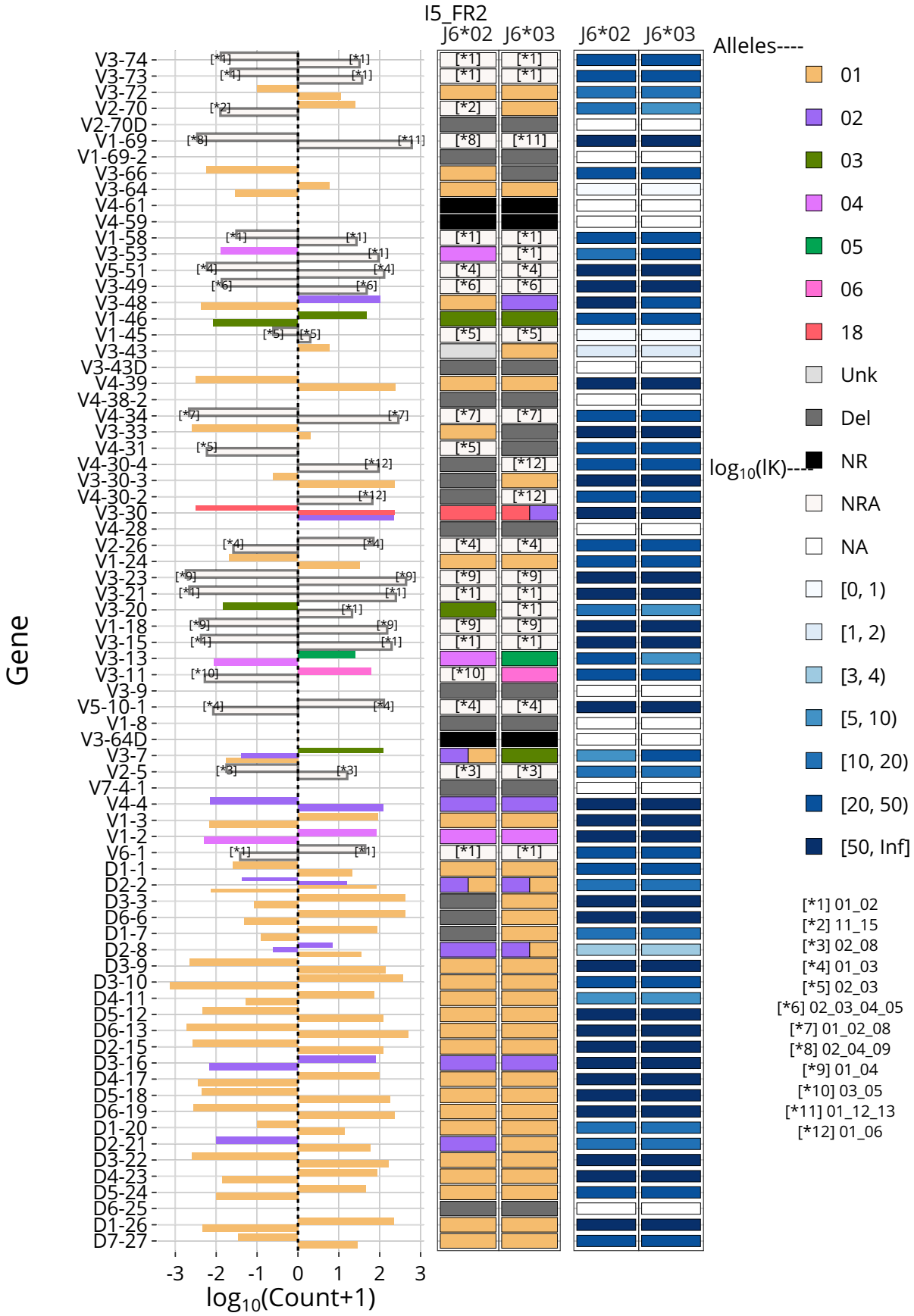
```
# Inferred deletion summary table
del_binom_db <- deletionsByBinom(clip_db, chain = "IGH",
                                 nonReliable_Vgenes = nonReliable_Vgenes)
```

Next, the haplotype can be inferred inputting the deletion summary table to the `deleted_genes` flag and the non reliable V genes list to the `nonRelaible_Vgenes` flag in the `createFullHaplotype` function.

```
# Using the deleted_genes and nonRelaible_Vgenes flags
# to infer haplotype for a partial V coverage sequence dataset
haplo_db <- createFullHaplotype(clip_db, toHap_col=c("V_CALL","D_CALL"),
                        hapBy_col="J_CALL", hapBy="IGHJ6",
                        toHap_GERM=c(HVGERM, HDGERM),
                        deleted_genes = del_binom_db,
                        nonReliable_Vgenes = nonReliable_Vgenes)
```

Another way to visualize the inferred haplotype is using the interactive HTML5 plot. To plot the HTML5 visualization, the `plotHaplotype` function should be used with the `html_output` flag marked `TRUE`. The function `saveWidget` from the `htmlwidgets` package can be used to save the interactive plot into an html file.

```
# Generate interactive haplotype plot
p <- plotHaplotype(haplo_db, html_output = TRUE)

# Saving the plot to html output
htmlwidgets::saveWidget(p, "haplotype.html", selfcontained = T)

# Plotting the interactive haplotype inference
p
```

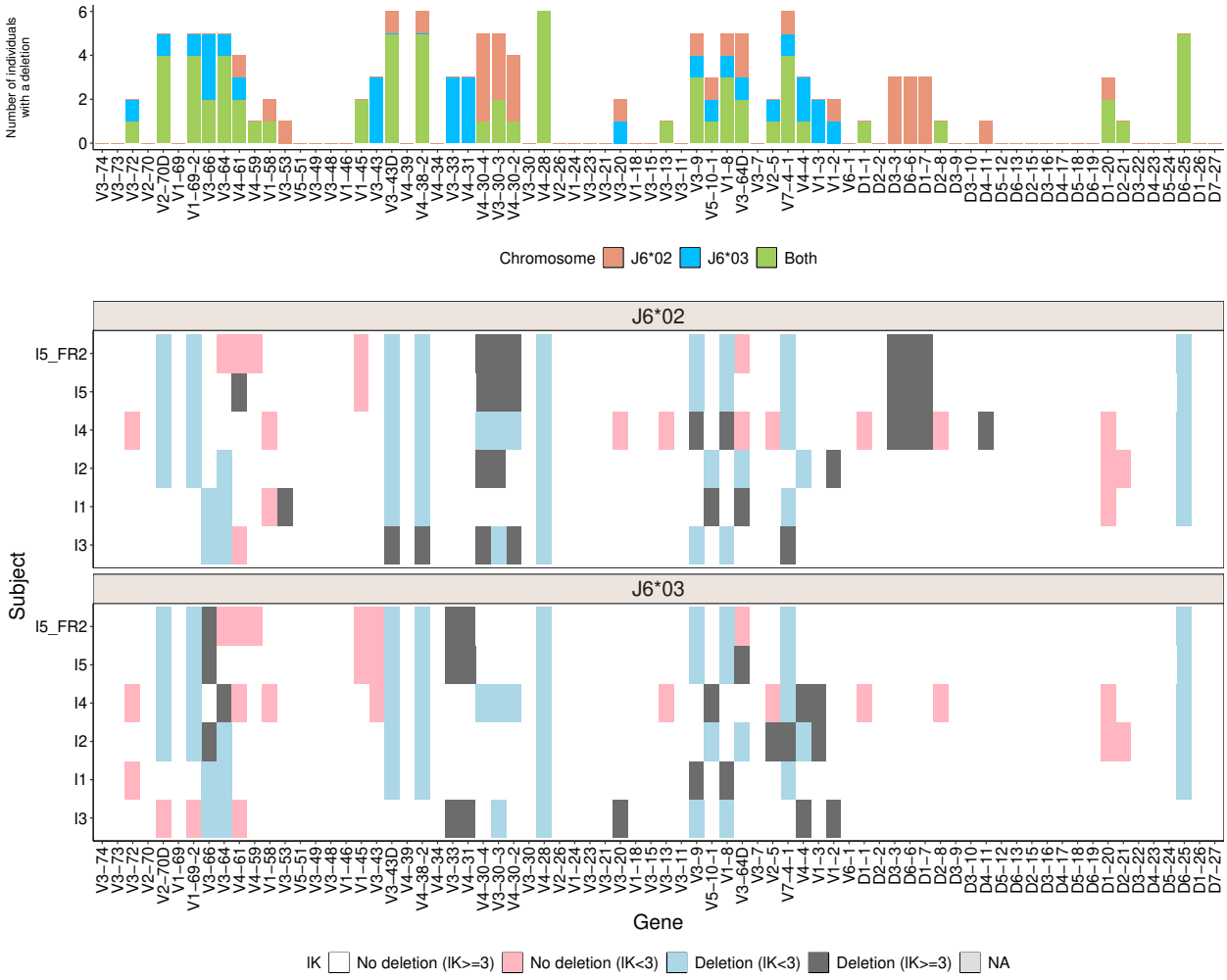

I5_FR2
J6*02
J6*03
J6*02
J6*03
Gene
log10(Count+1)
-3
-2
-1
0
1
2
3
Alleles----
01
02
03
04
05
06
18
Unk
Del
NR
log10(lK)----
NRA
NA
[0, 1)
[1, 2)
[3, 4)
[5, 10)
[10, 20)
[20, 50)
[50, Inf]
[*1] 01_02
[*2] 11_15
[*3] 02_08
[*4] 01_03
[*5] 02_03
[*6] 02_03_04_05
[*7] 01_02_08
[*8] 02_04_09
[*9] 01_04
[*10] 03_05
[*11] 01_12_13
[*12] 01_06

Haplotype inference deletion heatmap

`createFullHaplotype` can infer single chromosome deletion events by the threshold of the certainty level of the inference. The function utilizes the Bayes factor (K) obtained from the posterior probability of the inference. A gene deletion event is defined when the log 10 of the K value ($lK = log_{10}(K)$) for an "unknown" gene is larger than the threshold of 3 (The default value of the `kThreshDel` flag).

For a group of individuals, the single chromosome deletions coupled with the double chromosome deletions can be visualized with the `deletionHeatmap` function. The function creates a heatmap of the deletions inferred and colors them by the certainty level (lK).

```
# Detecting non reliable genes
nonReliable_Vgenes <- nonReliableVGenes(samples_db)

# Infering double chromosome deletion
del_binom_db <- deletionsByBinom(samples_db, nonReliable_Vgenes = nonReliable_Vgenes)

# Inferred haplotype summary table for multiple subjects
haplo_db <- createFullHaplotype(samples_db, toHap_col=c("V_CALL","D_CALL"),
                                hapBy_col="J_CALL", hapBy="IGHJ6",
                                toHap_GERM=c(HVGERM, HDGERM),
                                deleted_genes = del_binom_db,
                                nonReliable_Vgenes = nonReliable_Vgenes)
# plot deletion heatmap
deletionHeatmap(haplo_db)
```



Inferring D/J single chromosome deletion by V pooled approach

Since V gene heterozygosity is extremely common, using V genes as anchors for haplotype inference could increase the number of people for which D and J haplotype can be inferred. However, since there are far more V genes than J genes, the relative frequencies of them are lower. Therefore, to obtain a reliable haplotype inference using V genes as anchors, the sequence depth of the dataset has to be much greater than when using J6.

The RAbHIT package offers a solution to overcome the low number of sequences that connect a given V-D allele pair. RAbHIT applies an aggregation approach, in which connects information from several heterozygous V genes can be combined to infer D gene deletions. The `deletionsByVpooled` function uses the V pooled approach to detect single chromosomal deletions for D and J.

```
# Inferred deletion summary table
del_db <- deletionsByVpooled(samples_db, nonReliable_Vgenes = nonReliable_Vgenes)
```

```
## [1] "The following genes used for pooled deletion detection for sample I3"
## [1] "IGHV1-18"  "IGHV1-46"  "IGHV3-64D" "IGHV3-49"  "IGHV2-5"
## [1] "The following genes used for pooled deletion detection for sample I1"
## [1] "IGHV3-30" "IGHV3-48" "IGHV3-7"  "IGHV3-11" "IGHV3-49" "IGHV1-46"
```

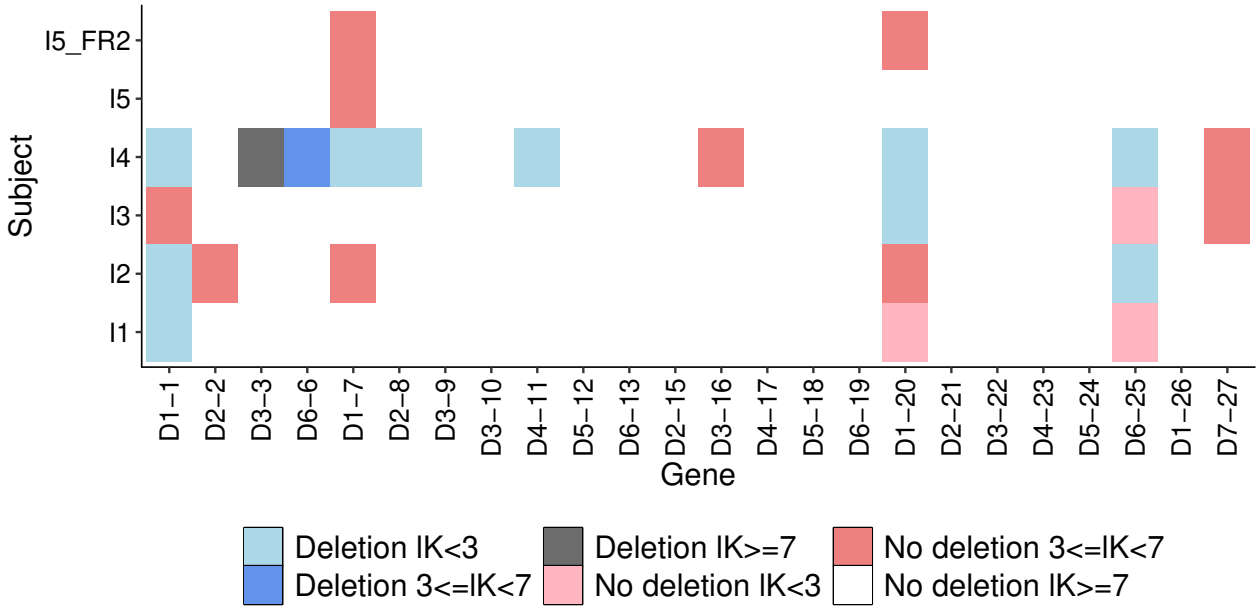
```
## [1] "The following genes used for pooled deletion detection for sample I2"
## [1] "IGHV3-9"  "IGHV1-69" "IGHV3-73"
## [1] "The following genes used for pooled deletion detection for sample I4"
## [1] "IGHV3-53" "IGHV3-48" "IGHV3-11" "IGHV1-46" "IGHV1-69" "IGHV2-5"  "IGHV3-49"
## [1] "The following genes used for pooled deletion detection for sample I5"
##  [1] "IGHV4-59"   "IGHV3-48"   "IGHV3-30"   "IGHV5-51"   "IGHV3-7"
##  [6] "IGHV3-64"   "IGHV2-70"   "IGHV1-58"   "IGHV3-53"   "IGHV5-10-1"
## [11] "IGHV3-49"   "IGHV3-73"
## [1] "The following genes used for pooled deletion detection for sample I5_FR2"
## [1] "IGHV2-70" "IGHV3-30" "IGHV3-53" "IGHV1-69" "IGHV3-48"
```

```
head(del_db)
```

```
##   SUBJECT     GENE DELETION     K COUNTS
## 1      I3  IGHD1-1        0  4.16    4,9
## 2      I3 IGHD1-20        1  0.07    1,6
## 3      I3 IGHD1-26        0 99.67 86,159
## 4      I3  IGHD1-7        0 14.11  14,33
## 5      I3 IGHD2-15        0 28.75  29,70
## 6      I3  IGHD2-2        0 42.69 86,351
```

For visualizing the deletions detected by `deletionsByVpooled`, the `plotDeletionsByVpooled` can be used. However, it is recommended to use this function for multiple individuals.

```
# Plot the deletion heatmap
plotDeletionsByVpooled(del_db)
```



Contact

For help, questions, or suggestions, please contact:

- Ayelet Peres
- Moriah Gidoni
- Gur Yaari
- Issue tracker

References

[1] M. J. Kidd *et al.*, "The Inference of Phased Haplotypes for the Immunoglobulin H Chain V Region Gene Loci by Analysis of VDJ Gene Rearrangements," *J Immunol*, vol. 188, no. 3, pp. 1333–1340, Feb. 2012 [Online]. Available: http://www.jimmunol.org/content/188/3/1333

[2] U. Kirik, L. Greiff, F. Levander, and M. Ohlin, "Parallel antibody germline gene and haplotype analyses support the validity of immunoglobulin germline gene inference and discovery," *Molecular Immunology*, vol. 87, pp. 12–22, 2017 [Online]. Available: https://www.ncbi.nlm.nih.gov/pubmed/28388445

[3] M.-P. Lefranc *et al.*, "IMGT unique numbering for immunoglobulin and t cell receptor variable domains and ig superfamily v-like domains," *Developmental & Comparative Immunology*, vol. 27, no. 1, pp. 55–77, 2003 [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/12477501

[4] J. A. Vander Heiden *et al.*, "pRESTO: a toolkit for processing high-throughput sequencing raw reads of lymphocyte receptor repertoires," *Bioinformatics*, vol. 30, no. 13, pp. 1930–1932, 2014 [Online]. Available: http://dx.doi.org/10.1093/bioinformatics/btu138

[5] D. Gadala-Maria *et al.*, "Identification of subject-specific immunoglobulin alleles from expressed repertoire sequencing data," *bioRxiv*, p. 405704, 2018 [Online]. Available: https://tigger.readthedocs.io/en/0.3.1/

[6] M. Gidoni *et al.*, "Mosaic deletion patterns of the human antibody heavy chain gene locus shown by bayesian haplotyping," *Nature Communications*, vol. 10, no. 1, p. 628, 2019.