Identifying clones from high-throughput B cell repertoire sequencing data

Nima Nouri, Jason Vander Heiden, Edel Aron, and Susanna Marquez

2026-01-08

Contents

Introduction

A key step to higher-level quantitative analysis of Adaptive Immune Receptor Repertoire sequencing (AIRR-Seq) data is the identification of B cell clones (sequences derived from cells descended from a common ancestor) using computationally-driven approaches. Accurate identification of clonal relationships is critical as these clonal groups form the fundamental basis for a wide range of repertoire analyses, including diversity analysis, lineage reconstruction, and effector functionality.

`scoper` provides a computational framework for identification of B cell clones from AIRR-Seq data among Ig (immunoglobulin, B cell receptor, BCR) sequences which share the same V gene, J gene and junction length. It includes methods to infer clonal relationships using either a predefined clustering threshold or an adaptive threshold.

Example data

A small example AIRR Rearrangement dataset is included in the `scoper` package. The dataset consists of a subset of Ig sequencing data from an influenza vaccination study (Laserson and Vigneault et al., PNAS, 2014). It includes sequences from multiple time-points before and after the subject received an influenza vaccination. Identification of clones requires the following fields (columns) to be present in the table:

- `junction`
- `v_call`
- `j_call`
- `sequence_alignment`
- `germline_alignment_d_mask`

Cells with multiple heavy chains in single cell data

When calling clones from single cell data, if any cell in the data contains multiple heavy chains, SCOPer will throw an error message such as *"xxx cell(s) with multiple heavy chains found. One heavy chain per cell is expected"* and then stop running. In the example data we provided in this section, we have cleaned up and removed cells with multiple heavy chains. But if your data contains cells with multiple heavy chains, you need to handle it before calling clones.

A simple solution is just removing cells with multiple heavy chains from the single cell data:

```
heavy_count <- table(dplyr::filter(ExampleDb, locus=="IGH")$cell_id)
multi_heavy_cells <- names(heavy_count)[heavy_count > 1]
ExampleDb <- dplyr::filter(ExampleDb, !cell_id %in% multi_heavy_cells)
```

Identifying clones by sequence identity

The simplest method for inferring clonal relationships is to define clones among identical junction region sequences (the junction region is where the V, D, and J gene segments join). This can be done using the `identicalClones` function at either the nucleotide level (`method="nt"`) or the amino acid level (`method="aa"`):

```
# Imports
library(scoper)
library(dplyr)

# Clonal assignment using identical nucleotide sequences
results <- identicalClones(ExampleDb, method="nt", summarize_clones = TRUE)
```

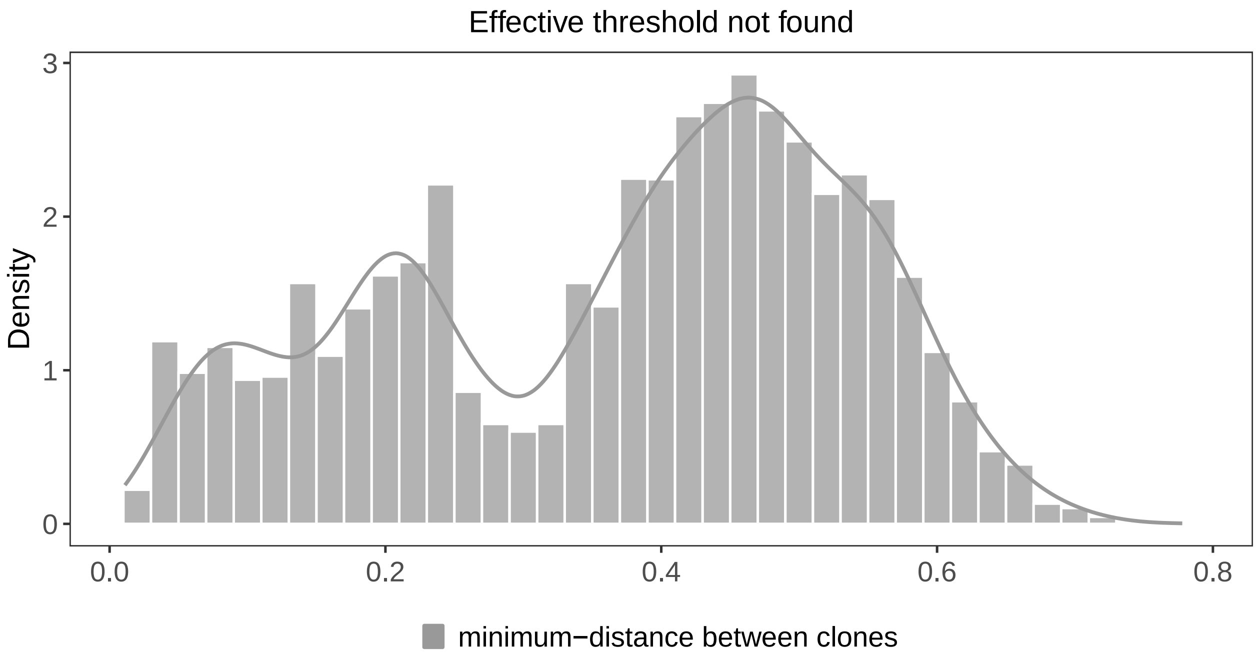
A modified input data.frame with clonal identifiers in the `clone_id` column is contained in the `db` slot of the returned returned `ScoperClones` object, which can also be accessed by calling `as.data.frame` on the result object.

```
# Get results data.frame
results_db <- as.data.frame(results)
glimpse(results_db)
```

```
## Rows: 2,000
## Columns: 17
## $ sequence_id               <chr> "GN5SHBT02B1YTK", "GN5SHBT04A8IJV", "GN5SHBT~
## $ sequence_alignment        <chr> "GAGGTGCAGCTGGTGGAGTCTGGGGGA...GGCTTGGTCCAGC~
## $ germline_alignment_d_mask <chr> "GAGGTGCAGCTGGTGGAGTCTGGGGGA...GGCTTGGTCCAGC~
## $ v_call                    <chr> "Homsap IGHV3-72*01 F", "Homsap IGHV3-72*01 ~
## $ v_call_genotyped          <chr> "IGHV3-72*01", "IGHV3-72*01", "IGHV3-73*01",~
## $ d_call                    <chr> "Homsap IGHD3-22*01 F", "Homsap IGHD6-13*01 ~
## $ j_call                    <chr> "Homsap IGHJ4*02 F", "Homsap IGHJ5*01 F", "H~
## $ junction                  <chr> "TGTGCTAGGGCCCAGATCGATTACTATGATAGTAGTGGTTATT~
## $ junction_length           <dbl> 72, 60, 51, 51, 51, 57, 57, 57, 39, 39, 39, ~
## $ np1_length                <dbl> 11, 10, 13, 10, 8, 6, 9, 9, 5, 9, 7, 8, 8, 7~
## $ np2_length                <dbl> 7, 20, 14, 1, 14, 8, 14, 6, 0, 2, 4, 7, 7, 4~
## $ sample_id                 <chr> "-1h", "-1h", "-1h", "-1h", "-1h", "-1h", "-~
## $ c_call                    <chr> "IGHM", "IGHA", "IGHM", "IGHD", "IGHM", "IGH~
## $ duplicate_count          <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,~
## $ locus                    <chr> "IGH", "IGH", "IGH", "IGH", "IGH", "IGH", "I~
## $ germline_alignment       <chr> "GAGGTGCAGCTGGTGGAGTCTGGGGGA...GGCTTGGTCCAGC~
## $ clone_id                 <chr> "1", "2", "3", "4", "5", "6", "7", "7", "8",~
```

A summary of the results can be plotted by calling `plot` on the returned `ScoperClones` object. This will show the minimum inter (between) clonal distances. A detailed summary of clonal relationships as a `data.frame` is also available through a call to `summary` on the results object.

```
# Plot a histogram of inter clonal distances
plot(results, binwidth=0.02)
```



```
# Get summary data.frame
glimpse(summary(results))
```

```
## Rows: 156
## Columns: 7
## $ vjl_group       <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16,~
## $ sequence_count  <int> 20, 1, 1, 1, 12, 7, 3, 1, 12, 5, 37, 4, 5, 10, 6, 9, 5~
## $ v_call          <chr> "Homsap IGHV1-18*01 F", "Homsap IGHV1-46*01 F,Homsap I~
## $ j_call          <chr> "Homsap IGHJ4*03 F,Homsap IGHJ4*02 F,Homsap IGHJ4*01 F~
## $ junction_length <int> 81, 81, 60, 102, 69, 66, 84, 96, 66, 75, 75, 84, 93, 6~
## $ clone_count     <int> 19, 1, 1, 1, 10, 7, 3, 1, 11, 4, 36, 4, 3, 8, 5, 9, 5,~
## $ clone_id        <chr> "518,519,520,521,522,523,524,525,526,527,528,529,530,5~
```

Identifying clones by hierarchical clustering

Most current studies uses a more sophisticated definition for clonal relationships. These studies leverage the high diversity of the junction region as a fingerprint to identify each B cell clone. Since it is unlikely that two separate recombination events would lead to identical junctions, sequences with junction regions that are similar enough are determined to share a common B cell ancestor (i.e., be clonally related) rather than to have arisen independently. Hierarchical clustering is a widely used distance-based method for identify clonally related sequences. An implementation of the hierarchical clustering approach is provided via the `hierachicalClones` function.

It is important to determine an appropriate threshold for trimming the hierarchical clustering into B cell clones before using this method. The ideal threshold for separating clonal groups is the value that separates the two modes of the nearest-neighbor distance distribution. The nearest-neighbor distance distribution can be generated by using the `distToNearest` function in the shazam R package. The threshold itself can be found using the `findThreshold` function in the same package. The resulting distribution should be bimodal, with the first mode representing sequences with clonal relatives in the dataset and the second mode representing singletons. For further details regarding inferring an appropriate threshold for the hierarchical clustering method, see the Distance to Nearest Neighbor vignette in the shazam package. Technical details can be found in:

```
Gupta NT, et al. (2017). Hierarchical clustering can identify B cell clones with
    high confidence in Ig repertoire sequencing data.
    The Journal of Immunology 198(6):2489-2499.
```

Identifying clonal groups using `hierachicalClones` is largely the same as the approach using the `identicalClones` function, with the additional requirement of a distance threshold:

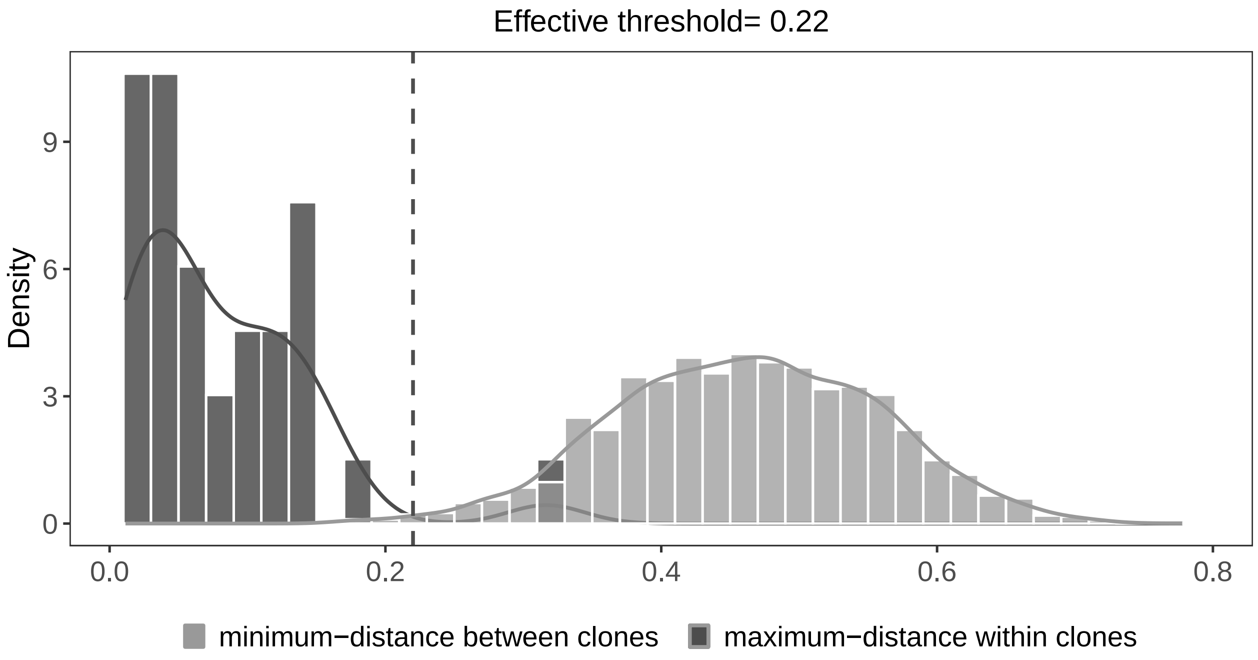
```
# Clonal assignment using hierarchical clustering
results <- hierarchicalClones(ExampleDb, threshold=0.15, summarize_clones = TRUE)
```

The results data.frame, summary plots, and summary table are accessed in the same manner as above. This will show the minimum inter (between) and maximum intra (within) clonal distances along with an effective threshold, which is an explanatory value calculated to represent the cut-off separating the inter and intra clonal distances. The effective threshold may differ from clustering threshold provided as input to `hierarchicalClones`.

```
# Get results data.frame
results_db <- as.data.frame(results)
glimpse(results_db)
```

```
## Rows: 2,000
## Columns: 17
## $ sequence_id               <chr> "GN5SHBT02B1YTK", "GN5SHBT04A8IJV", "GN5SHBT~
## $ sequence_alignment        <chr> "GAGGTGCAGCTGGTGGAGTCTGGGGGA...GGCTTGGTCCAGC~
## $ germline_alignment_d_mask <chr> "GAGGTGCAGCTGGTGGAGTCTGGGGGA...GGCTTGGTCCAGC~
## $ v_call                    <chr> "Homsap IGHV3-72*01 F", "Homsap IGHV3-72*01 ~
## $ v_call_genotyped          <chr> "IGHV3-72*01", "IGHV3-72*01", "IGHV3-73*01",~
## $ d_call                    <chr> "Homsap IGHD3-22*01 F", "Homsap IGHD6-13*01 ~
## $ j_call                    <chr> "Homsap IGHJ4*02 F", "Homsap IGHJ5*01 F", "H~
## $ junction                  <chr> "TGTGCTAGGGCCCAGATCGATTACTATGATAGTAGTGGTTATT~
## $ junction_length           <dbl> 72, 60, 51, 51, 51, 57, 57, 57, 39, 39, 39, ~
## $ np1_length                <dbl> 11, 10, 8, 13, 10, 6, 9, 9, 5, 9, 12, 7, 7, ~
## $ np2_length                <dbl> 7, 20, 14, 14, 1, 8, 14, 6, 0, 2, 7, 4, 4, 4~
## $ sample_id                 <chr> "-1h", "-1h", "-1h", "-1h", "-1h", "-1h", "-~
## $ c_call                    <chr> "IGHM", "IGHA", "IGHM", "IGHM", "IGHD", "IGH~
## $ duplicate_count           <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,~
## $ locus                     <chr> "IGH", "IGH", "IGH", "IGH", "IGH", "IGH", "I~
## $ germline_alignment        <chr> "GAGGTGCAGCTGGTGGAGTCTGGGGGA...GGCTTGGTCCAGC~
## $ clone_id                  <chr> "1", "2", "3", "4", "5", "6", "7", "7", "8",~
```

```
# Plot a histogram of inter and intra clonal distances
plot(results, binwidth=0.02)
```



```
# Get summary data.frame
glimpse(summary(results))
```

```
## Rows: 156
## Columns: 7
## $ vjl_group       <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16,~
## $ sequence_count  <int> 20, 1, 1, 1, 12, 7, 3, 1, 12, 5, 37, 4, 5, 10, 6, 9, 5~
## $ v_call          <chr> "Homsap IGHV1-18*01 F", "Homsap IGHV1-46*01 F,Homsap I~
## $ j_call          <chr> "Homsap IGHJ4*02 F,Homsap IGHJ4*03 F,Homsap IGHJ4*01 F~
## $ junction_length <int> 81, 81, 60, 102, 69, 66, 84, 96, 66, 75, 75, 84, 93, 6~
## $ clone_count     <int> 19, 1, 1, 1, 5, 7, 3, 1, 11, 4, 36, 4, 3, 6, 5, 9, 5, ~
## $ clone_id        <chr> "491,492,493,494,495,496,497,498,499,500,501,502,503,5~
```

Identifying clones by spectral clustering

While the hierarchical clustering method groups sequences using a fixed distance supervised threshold, the spectral clustering-based model uses an adaptive unsupervised threshold to tune the required level of similarity among sequences in different local neighborhoods. It can be used as an alternative if the nearest-neighbor distance distribution is unimodal, meaning that `findThreshold` wasn't able to find the threshold at which to cut the hierarchy.

There are two available spectral clustering methods provided by the `spectralClones` function:

1. `method="novj"`: Infers clonal relationships using an adaptive threshold that indicates the level of similarity among junction sequences in a local neighborhood

2. `method="vj"`: Infers clonal relationships not only based on the junction region homology, but also taking into account the mutation profiles in the V and J segments

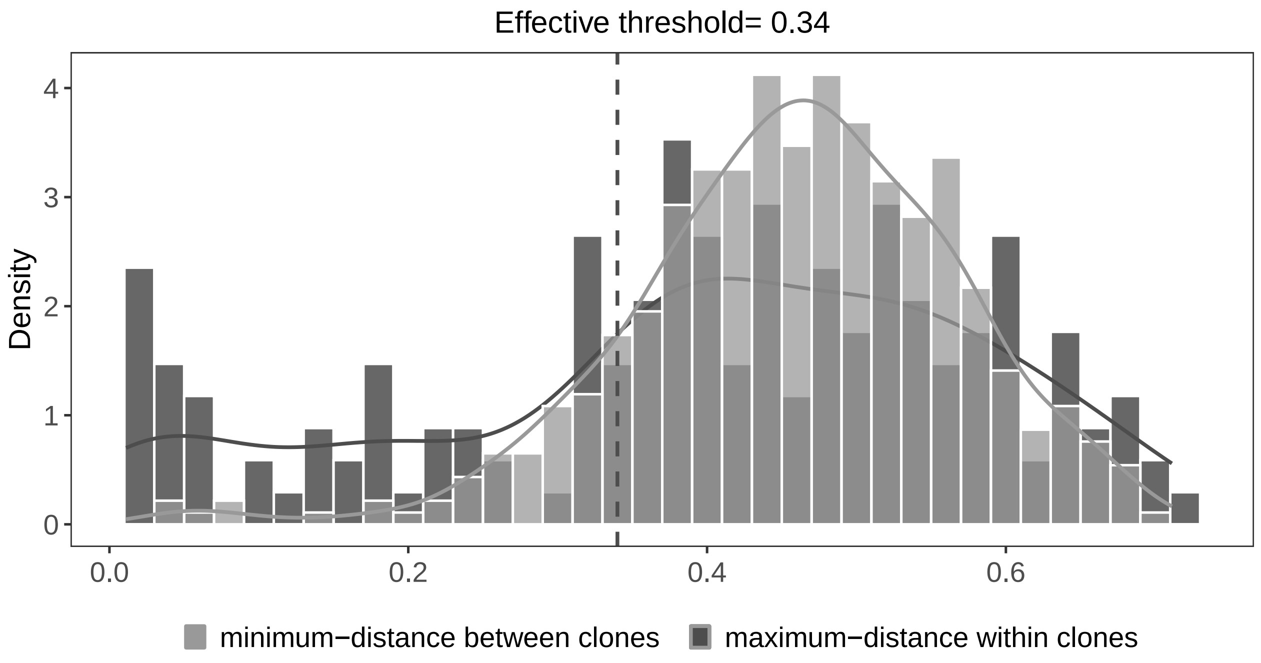
It is not mandatory, but a fixed threshold can also be provided to `spectralClones` (`threshold` argument) which will enforce an upper-limit cut-off. When specifying the `threshold` argument, any sequence with distances above the threshold value from all sequences will become a singleton. The threshold can be defined as discussed above using nearest-neighbor distance distribution methods provided in the shazam. If `method="vj"`, the `threshold` will be applied on the distances calculated combining junction region homology and V/J mutation profiles. Technical details can be found in:

```
Nouri N and Kleinstein SH (2018). A spectral clustering-based method for
    identifying clones from high-throughput B cell repertoire sequencing data.
    Bioinformatics, 34(13):i341-i349.

Nouri N and Kleinstein SH (2020). Somatic hypermutation analysis for improved
    identification of B cell clonal families from next-generation sequencing data,
    PLoS Comput Biol 16(6): e1007977. https://doi.org/10.1371/journal.pcbi.1007977
```

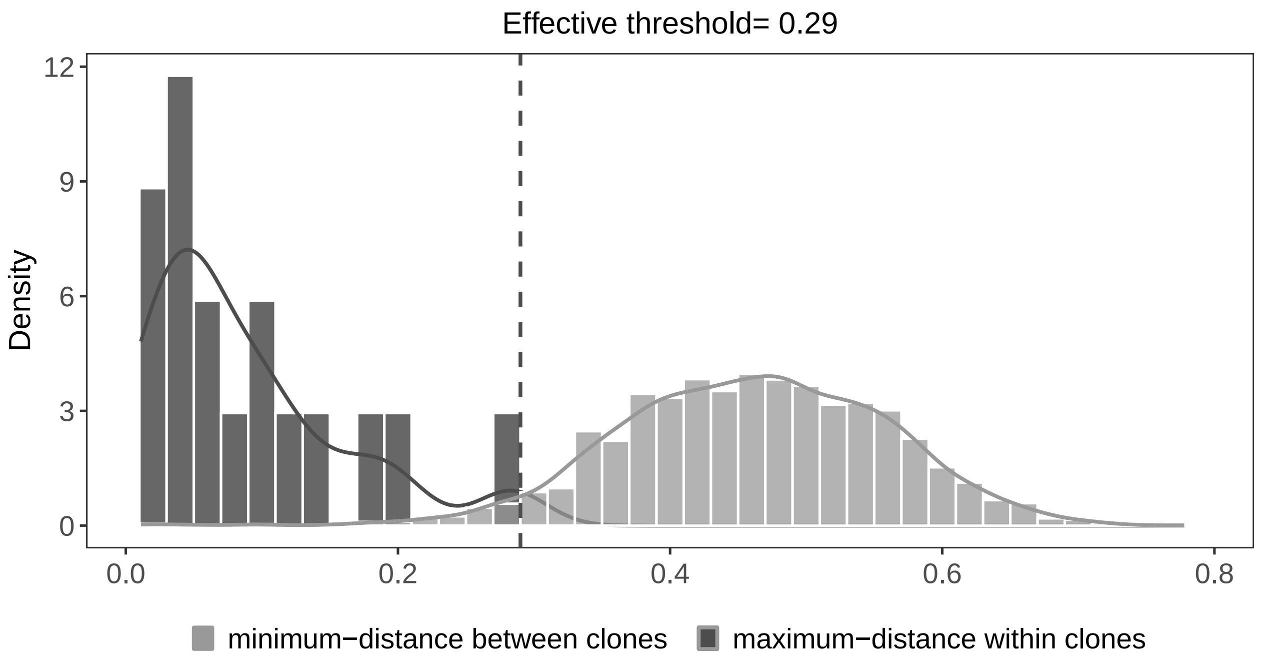
The following examples calls the `spectralClones` function with(out) the same (optional) threshold used previously to define clones using the hierarchical approach. The results data.frame, summary plots, and summary table are accessed in the same manner as shown above using the hierarchical approach.

```
# Clonal assignment using the spectral clustering method novj
results <- spectralClones(ExampleDb, method="novj", summarize_clones = TRUE)
# Plot a histogram of inter and intra clonal distances
plot(results, binwidth=0.02)
```



```
# Clonal assignment using the spectral clustering method novj with threshold
results <- spectralClones(ExampleDb, method="novj",
                          threshold=0.15, summarize_clones = TRUE)
# Plot a histogram of inter and intra clonal distances
```

```r
plot(results, binwidth=0.02)
```

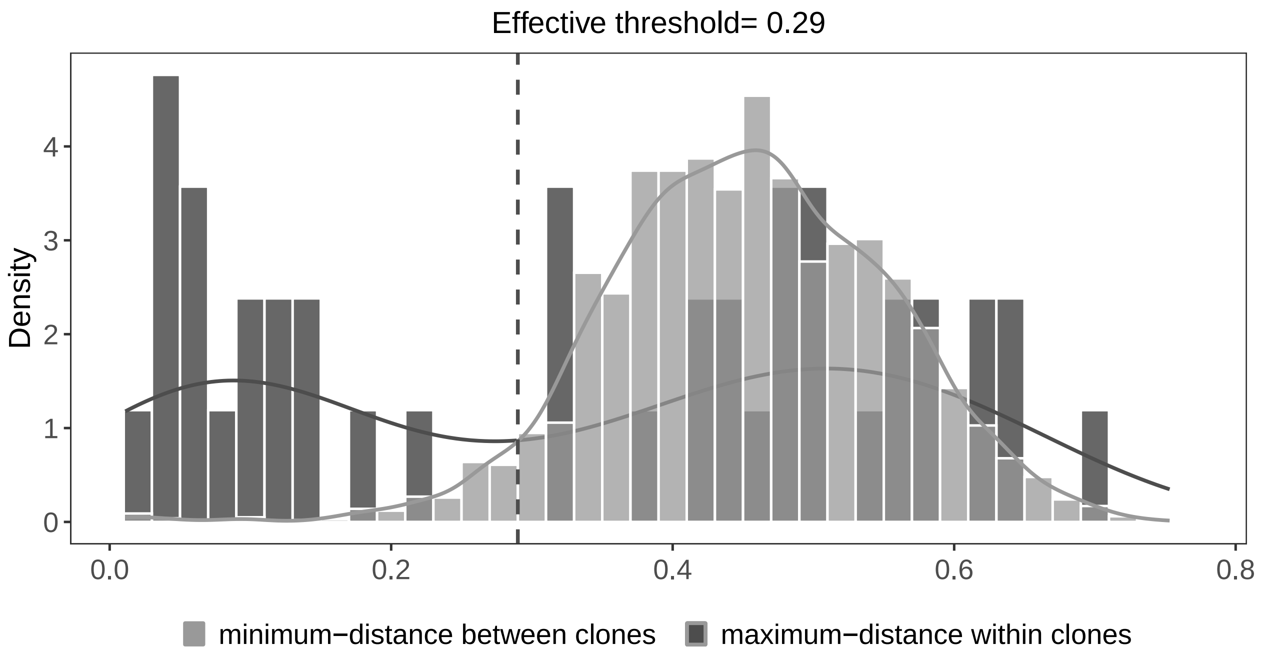


```r
# Clonal assignment using the spectral clustering method vj with threshold
results <- spectralClones(ExampleDb, method="vj",
                          threshold=0.15,
                          germline="germline_alignment_d_mask",
                          summarize_clones = TRUE)

# Get results data.frame
results_db <- as.data.frame(results)
glimpse(results_db)
```

```
## Rows: 2,000
## Columns: 17
## $ sequence_id               <chr> "GN5SHBT02B1YTK", "GN5SHBT04A8IJV", "GN5SHBT~
## $ sequence_alignment        <chr> "GAGGTGCAGCTGGTGGAGTCTGGGGGA...GGCTTGGTCCAGC~
## $ germline_alignment_d_mask <chr> "GAGGTGCAGCTGGTGGAGTCTGGGGGA...GGCTTGGTCCAGC~
## $ v_call                    <chr> "Homsap IGHV3-72*01 F", "Homsap IGHV3-72*01 ~
## $ v_call_genotyped          <chr> "IGHV3-72*01", "IGHV3-72*01", "IGHV3-73*01",~
## $ d_call                    <chr> "Homsap IGHD3-22*01 F", "Homsap IGHD6-13*01 ~
## $ j_call                    <chr> "Homsap IGHJ4*02 F", "Homsap IGHJ5*01 F", "H~
## $ junction                  <chr> "TGTGCTAGGGCCCAGATCGATTACTATGATAGTAGTGGTTATT~
## $ junction_length           <dbl> 72, 60, 51, 51, 51, 57, 57, 57, 39, 39, 39, ~
## $ np1_length                <dbl> 11, 10, 8, 13, 10, 6, 9, 9, 5, 9, 12, 7, 7, ~
## $ np2_length                <dbl> 7, 20, 14, 14, 1, 8, 14, 6, 0, 2, 7, 4, 4, 4~
## $ sample_id                 <chr> "-1h", "-1h", "-1h", "-1h", "-1h", "-1h", "-~
## $ c_call                    <chr> "IGHM", "IGHA", "IGHM", "IGHM", "IGHD", "IGH~
## $ duplicate_count           <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,~
## $ locus                     <chr> "IGH", "IGH", "IGH", "IGH", "IGH", "IGH", "I~
## $ germline_alignment        <chr> "GAGGTGCAGCTGGTGGAGTCTGGGGGA...GGCTTGGTCCAGC~
## $ clone_id                  <chr> "1", "2", "3", "4", "5", "6", "7", "7", "8",~
```

```r
# Plot a histogram of inter and intra clonal distances
plot(results, binwidth=0.02)
```



```r
# Get summary data.frame
glimpse(summary(results))
```

```
## Rows: 156
## Columns: 7
## $ vjl_group       <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16,~
## $ sequence_count  <int> 20, 1, 1, 1, 12, 7, 3, 1, 12, 5, 37, 4, 5, 10, 6, 9, 5~
## $ v_call          <chr> "Homsap IGHV1-18*01 F", "Homsap IGHV1-46*01 F,Homsap I~
## $ j_call          <chr> "Homsap IGHJ4*02 F,Homsap IGHJ4*03 F,Homsap IGHJ4*01 F~
## $ junction_length <int> 81, 81, 60, 102, 69, 66, 84, 96, 66, 75, 75, 84, 93, 6~
## $ clone_count     <int> 19, 1, 1, 1, 6, 7, 3, 1, 11, 4, 9, 4, 3, 6, 5, 9, 4, 3~
## $ clone_id        <chr> "438,439,440,441,442,443,444,445,446,447,448,449,450,4~
```