Scoper

Nima Nouri and Jason Vander Heiden

2019-08-05

Contents

Description 1
Model 1
Clonal partitioning 2

Description

`scoper` provides a computational framework for B cell clones identification from adaptive immune receptor repertoire sequencing (AIRR-Seq) datasets. Three models are included (identical, hierarchical, and spectral) that perform clustering among sequences of BCRs/IGs (B cell receptors/immunoglobulins) which share the same V gene, J gene and junction length.

Model

identical: Defines clones among identical junctions. The two available methods are: (1) `nt` (nucleotide based clustering), and (2) `aa` (amino acid based clustering).

hierarchical: Groups sequences using a fixed distance supervised threshold at which to cut the hierarchy. The three available agglomeration methods are: (1) `single`, (2) `average`, and (3) `complete`. It is important to determine an appropriate threshold for trimming the hierarchical clustering into B cell clones before using this model. The ideal threshold for separating clonal groups is the value that separates the two modes of the distance-to-nearest distribution and can be found using the `findThreshold` function in the `SHazaM` R package. The distribution can be generated by using the `distToNearest` function in the same package which calculates the distance between each sequence in the data and its nearest-neighbor. The result should be bimodal, with the first mode representing sequences with clonal relatives in the dataset and the second mode representing singletons. The `hierarchical` model may not be used if the bi-modality in distance-to-nearest distribution is not observed. 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.
```

spectral: While hierarchical clustering-based models group 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 distance-to-nearest distribution is unimodal (so `findThreshold` wasn't able to find the threshold at which to cut the hierarchy, see above). The two available methods are: (1) `novj`: clonal relationships are inferred using an adaptive threshold that indicates the level of similarity among junction sequences in a local neighborhood, and (2) `vj`: clonal relationships are

inferred not only based on the junction region homology, but also taking into account the mutation profiles in the V and J segments. 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.
```

Clonal partitioning

```
# Load scoper
library("scoper")
```

There are several parameter choices when grouping Ig sequences into B cell clones:

```
# Load scoper
library("scoper")
defineClonesScoper(db,
                   model = c("identical", "hierarchical", "spectral"),
                   method = c("nt", "aa", "single", "average", "complete",
                              "novj", "vj"),
                   germline_col = "GERMLINE_IMGT", sequence_col = "SEQUENCE_IMGT",
                   junction_col = "JUNCTION",
                   v_call_col = "V_CALL", j_call_col = "J_CALL",
                   clone_col = c("CLONE", "clone_id"),
                   targeting_model = NULL, len_limit = NULL, first = FALSE,
                   cdr3 = FALSE, mod3 = FALSE, max_n = NULL, threshold = NULL,
                   base_sim = 0.95, iter_max = 1000, nstart = 1000, nproc = 1,
                   verbose = FALSE, log_verbose = FALSE, out_dir = ".",
                   summerize_clones = FALSE)
```

The following discussion is applicable for all three models.

1. The data set needs to be passed to the argument `db`, which at the end will be returned as a modified `db` `data.frame` with clone identifiers in the column specified by argument `clone_col`.
2. The names of the columns containing nucleotide sequences (in the junction region), V-segment allele calls and J-segment allele calls needs to be assigned to the arguments `junction_col`, `v_call_col` and `j_call_col` respectively.
3. If a genotype has been inferred using the methods in the `tigger` package, and a `V_CALL_GENOTYPED` field has been added to the database, then this column may be used instead of the default `V_CALL` column by specifying the `v_call_col` argument. This will allow the more accurate V call from `tigger` to be used for grouping of the sequences.
4. For more leniency toward ambiguous V(D)J segment calls the parameter `first` can be set to `FALSE`.
5. To remove 3 nucleotides from both ends of the junction region (i.e., converting an IMGT junction to a Complementarity-Determining Region 3 region) the logical argument `cdr3` needs to be set as `TRUE` (the default is `FALSE`). This also leads to the removal of junctions with length less than 7 nucleotides from the original `db` dataset.

6. To remove a junction(s) with a number of nucleotides not modulus of 3, the logical argument `mod3` should be set as `TRUE` (the default is `FALSE`).
7. A summary of each step cloning process would be reported if `verbose` set to `TRUE` (the default is `FALSE`).
8. If the argument `log_verbose` be set as `TRUE`, the `verbose` output is written to a file in the current input directory (by default).
9. If the `out_dir` is specified, then its path will be used to save `log_verbose`.
10. If `summerize_clones` set to be `FALSE` (default), the `defineClonesScoper` function will return a modified `data.frame` with clone identifiers in the `clone_col` column. Otherwise, if `summerize_clones` set to be `TRUE`, the `defineClonesScoper` function will perform a series of analyses to assess the clonal landscape and return a list containing summary statistics and visualization of the clonal clustering results:
    - **db**: a modified `data.frame` with clone identifiers in the `clone_col` column.
    - **vjl_group_summ**: a `data.frame` of clones summary, e.g. size, V-gene, J-gene, junction lentgh, and so on.
    - **inter_intra**: a `data.frame` containing minimum inter (between) and maximum intra (within) clonal distances.
    - **eff_threshold**: effective cut-off separating the inter (between) and intra (within) clonal distances.
    - **plot_inter_intra**: a `ggplot` histogram of inter (between) versus intra (within) clonal distances. The effective threshold is shown with a horizental dashed-line.

Note: models specific arguments:

1. **hierarchical**: The argument `threshold` (a numeric scalar where the tree should be cut) must be provided.
2. **spectral**: The arguments `iter_max` and `nstart` are required to perform the k-means clustering step of the pipeline. They will pass the maximum allowed number of kmean clustering iterations and the number of random sets chosen for kmean clustering initialization respectively. The argument `base_sim` is required to be used as the similarity cut-off for sequences in equal distances from each other. It is not mandatory, but the argument `threshold` can also be used for the model `spectral` in order to enforce an upper-limit cut-off. The arguments `germline_col` and `sequence_col` must be provided if method `vj` is used. Therefore, mutation counts are determined by comparing the input sequences (in the column specified by `sequence_col`) to the effective germline sequence (calculated from sequences in the column specified by `germline_col`). Arguments `len_limit` can be used to focus only on the V segment. It is not mandatory, but the influence of SHM hot- and cold-spot biases in the clonal inference process will be noted if a SHM targeting model is provided through the argument `targeting_model` (see the function `createTargetingModel` from `SHazaM` R package for more technical details).

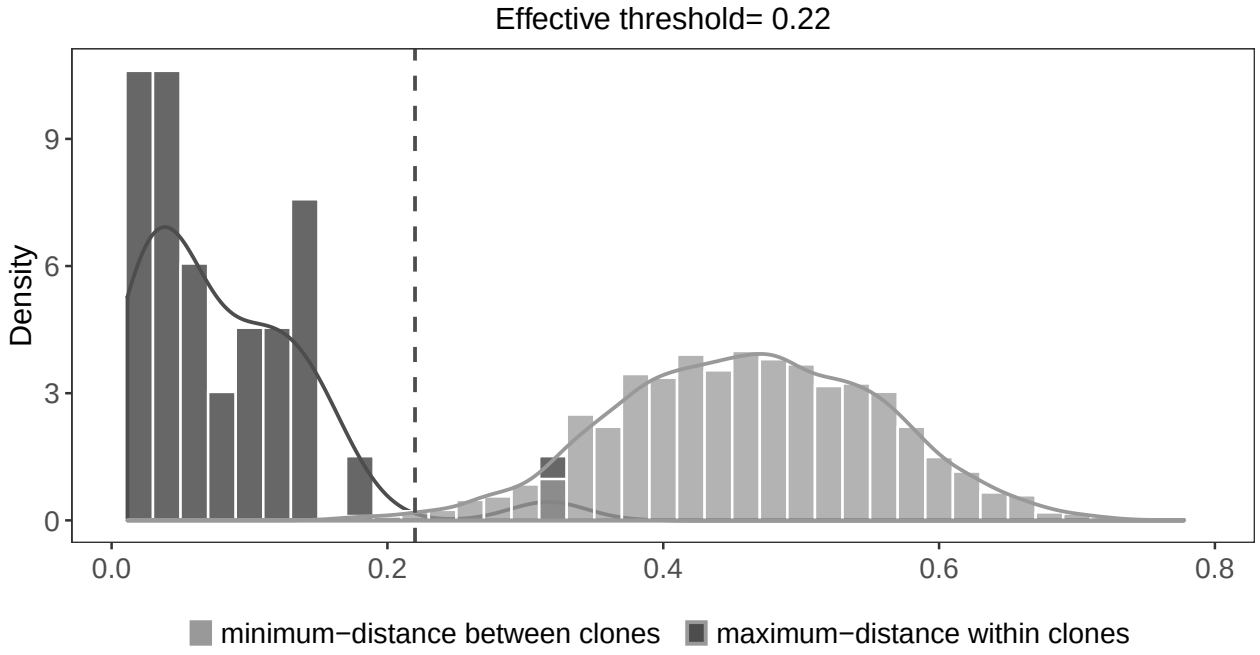
A small example Change-O database is included in the `scoper` package:

```
# Clonal assignment using hierarchical model
results <- defineClonesScoper(db = ExampleDb, clone_col = "CLONE",
                              model = "hierarchical", method = "single",
                              threshold = 0.15, summerize_clones = TRUE)
```

```
# cloned data (a data.frame)
cloned_db <- results$db
# print effective threshold (a numeric)
results$eff_threshold
```

```
## [1] 0.22
```

```
# get inter and intra conal distances (a data.frame)
df <- results$inter_intra
# histogram of inter versus intra clonal distances  (a ggplot).
results$plot_inter_intra
```




```
# Clonal assignment using spectral model
# IMGT_V object from shazam package to identify sequence limit length
library("shazam")
results <- defineClonesScoper(db = ExampleDb, clone_col = "clone_id",
                              model = "spectral", method = "vj",
                              len_limit = shazam::IMGT_V,
                              targeting_model = shazam::HH_S5F,
                              sequence_col = "SEQUENCE_IMGT",
                              germline_col = "GERMLINE_IMGT_D_MASK",
                              threshold = 0.15,
                              summerize_clones = TRUE)
# cloned data (a data.frame)
cloned_db <- results$db
# print effective threshold (a numeric)
results$eff_threshold
```

```
## [1] 0.28
```

```
# get inter and intra conal distances (a data.frame)
```

```
df <- results$inter_intra
# histogram of inter versus intra clonal distances  (a ggplot).
results$plot_inter_intra
```