Package ‘ClustIRR’

April 5, 2024

Type Package

Title Clustering of immune receptor repertoires

Version 1.0.0

Description ClustIRR is a quantitative method for clustering of immune receptor repertoires (IRRs). The algorithm identifies groups of T or B cell receptors (TCRs or BCRs) with similar specificity by comparing their sequences. ClustIRR uses graphs to visualize the specificity structures of IRRs.

License GPL-3 + file LICENSE

LazyData false

Depends R (>= 4.3.0)

Imports stringdist, future, future.apply, methods, stats, utils, igraph, visNetwork

Suggests BiocStyle, knitr, testthat, ggplot2, patchwork, ggrepel

Encoding UTF-8

NeedsCompilation no

biocViews Clustering, ImmunoOncology, SingleCell, Software, Classification

RoxygenNote 7.2.3

VignetteBuilder knitr

URL https://github.com/snaketron/ClustIRR

BugReports https://github.com/snaketron/ClustIRR/issues

git_url https://git.bioconductor.org/packages/ClustIRR

git_branch RELEASE_3_18

git_last_commit 739bbff

git_last_commit_date 2023-10-24

Repository Bioconductor 3.18

Date/Publication 2024-04-05

Author Simo Kitanovski [aut, cre] (<https://orcid.org/0000-0003-2909-5376>), Kai Wollek [aut] (<https://orcid.org/0009-0008-5941-9160>)

Maintainer Simo Kitanovski <simokitanovski@gmail.com>

R topics documented:

CDR3ab — *Mock data set of complementarity determining region 3 (CDR3) sequences from the α and β chains of 10,000 T cell receptors*

Description

Mock data set containing amino acid sequences of paired CDR3s from the α and β chains of 10,000 T cell receptors. All CDR3 sequences were drawn from a larger set of CDR3β sequences from human naive CD8+ T cells.

Usage

```
data(CDR3ab)
```

Format

`data.frame` with 10,000 rows and 2 columns `CDR3a` and `CDR3b`.

Value

data(CDR3ab) loads the object CDR3ab, which is a data.frame with two columns and 10,000 rows.

Source

GLIPH version 2

Examples

```
data("CDR3ab")
```

cluster_irr *Clustering of immune receptor repertoires*

Description

This algorithm finds groups of TCRs or BCRs with similar specificity. Two clustering strategies are employed:

1. Local clustering
2. Global clustering

Local clustering

1. CDR3 processing steps
   - each row of s and r is considered as a CDR3 sequence from an individual T- or B-cell (version = 2, default). If version=1 is specified, then we compute the set of non-redundant CDR3s from s and r and use them for clustering.
   - Trim CDR3 ends
2. Motif processing steps
   - motif frequencies in data set `s` (f_s) and `r` (f_r)
   - total number of motifs in data set `s` (n_s) and `r` (n_r)
   - ratio of observed vs. expected motif counts using the following formula: OvE=$(f_s/n_s)/(f_r/n_r)$
   - probability p_i of finding the observed or a larger OvE for motif i given that the null hypothesis is true is computed with the Fisher's exact test
   - classify motif i as `pass=TRUE` if the motif passes all filters specified in the user-provided control list, otherwise as `pass=FALSE`

Global clustering

The default ClustIRR algorithm for global clustering is simple. For each pair of equal-length CDR3 sequences i and j we compute the Hamming distance d_{ij}. If $d_{ij} \leq$ `global_max_dist` (user-defined input), then i and j are globally similar.

Alternatively, the user can provide a matrix of globally similar CDR3 sequence pairs, computed by a complementary approachs such as TCRdist.

Usage

```
cluster_irr(
    s,
    r,
    version = 2,
    ks = 4,
    cores = 1,
    control = list(global_max_dist = 1,
                   local_max_fdr = 0.05,
                   local_min_ove = 2,
                   local_min_o = 1,
                   trim_flank_aa = 0,
                   global_pairs = NULL,
                   low_mem = FALSE))
```

Arguments

`s` data.frame, complementarity determining region 3 (CDR3) amino acid sequences observed in an immune receptor repertoire (IRR). The data.frame can have either one column or two columns:

- One column: `s` contains CDR3s from a single chain: *CDR3b*, *CDR3a*, *CDR3g*, *CDR3d*, *CDR3h* or *CDR3l*
- Two columns: `s` contains CDR3s from both chains (paired), for instance:
  - *CDR3b* and *CDR3a* [for $\alpha\beta$ TCRs]
  - *CDR3g* and *CDR3d* [for $\gamma\delta$ TCRs]
  - *CDR3h* and *CDR3l* [for heavy/ligh chain BCRs]

`r` data.frame, reference (or control) repertoire of CDR3 sequences. Must have the same structure (number of columns and column names) as `s`

`version` integer, version of the algorithm: `version` = 1 or 2 (default)

`ks` integer or integer vector, motif lengths. `ks` = 4 (default)

`cores` integer, number of CPU cores, `cores` = 1 (default).

`control` list, a named list of auxiliary parameters to control algorithm's behavior. See the details below:

- `global_max_dist` - number, Hamming distance (HD) threshold to consider two CDR3s as globally clustered. CDR3s are globally clustered if HD(a, b) $\leq$ `global_max_dist`. `global_max_dist` = 1 (default)
- `local_max_fdr` - numeric, maximum False Discovery Rate (FDR) for the detection of enriched motifs. `local_max_fdr` = 0.05 (default)
- `local_min_ove` - numeric, minimum fold change between observed and expected relative abundances for the detection of enriched motifs. `local_min_ove` = 2 (default)
- `local_min_o` - numeric, minimum absolute frequency of a motif in the `s` in order for the motif to be used in the enrichment analysis. `local_min_o` = 1 (default)
- `trim_flank_aa` - integer, how many amino acids should be trimmed from the flanks of all CDR3 sequences (only used for local clustering. `trim_flank_aa` = 0 (default))
- `low_mem` - logical, allows low memory mode for global clustering. This will lead to increase in the CPU time but lead to a lower memory footprint. `low_mem` = `FALSE` (default)
- `global_pairs` - matrix, pre-computed global pairs. If `global_pairs` is provided by the user, then global clustering is not performed. Instead the CDR3 pairs from global_pairs are used as global clustering pairs. `global_pairs` is a character matrix with 3 columns. The first two columns contain pairs of CDR3 sequences. These are considered globally clustered. The third column contains information about the TCR chain of each pair of CDR3s: *TRA* or *TRB*. `global_pair` = `NULL` (default)

Value

The output is an S4 object of class `clust_irr`. This object contains two sublists:

`clust` list, contains clustering results for each TCR/BCR chain. The results are stored in separate sub-list named appropriately (e.g. CDR3a, CDR3b, CDR3g, etc.). In the following we who the typical structure of these lists:

- `local` - list, local clustering results
  - `m` - data.frame, motif enrichment results with columns:
    - `motif` - motif sequence
    - `f_s` - observed motif counts in `s`
    - `f_r` - observed motif counts in `r`
    - `n_s` - number of all observed motifs in `s`
    - `n_r` - number of all observed motifs in `r`
    - `k` - motif length
    - `ove` - mean observed/expected relative motif frequency
    - `ove_ci_l95` - 95% confidence intervals of ove (lower boundary)
    - `ove_ci_h95` - 95% confidence intervals of ove (upper boundary)
    - `p_value` - p-value from Fisher's exact test
    - `fdr` - false discovery rate, i.e. adjusted p-value by Benjamini & Hochberg correction
    - `pass` - logical value indicating whether a motifs are enriched (`pass=TRUE`) given the user-defined thresholds in control
  - `lp` - data.frame, enriched motifs are linked to their original CDR3 sequences and shown as rows in the data.frame with the following columns:
    - `cdr3` - CDR3 amino acid sequence
    - `cdr3_core` - core portion of the CDR3 sequence, obtained by trimming `trim_flank_aa` amino acids (user- defined parameter). If `trim_flank_aa` = 0, then `cdr3` = `cdr3_core`
    - `motif` - enriched motif from `cdr3_core`
- `global` - matrix, global clustering results. Pairs of globally similar CDR3s are shown in each row of the matrix (analogous to `lp`)

`inputs` list, contains all user provided inputs (see **Arguments**)

Examples

```
# load package input data
data("CDR3ab")
s <- data.frame(CDR3b = CDR3ab[1:1000, "CDR3b"])
r <- data.frame(CDR3b = CDR3ab[1:5000, "CDR3b"])

# artificially enrich motif 'RQWW' inside sample dataset
base::substr(x = s$CDR3b[1:20], start = 6, stop = 9) <- "RQWW"

# add an artificial clonal expansion of two sequences to the sample dataset
s <- base::rbind(s, base::data.frame(CDR3b = rep(x = c("CATSRAAKPDGLRALETQYF",
                                    "CATSRAAKPDRQWWLSTQYF"),
                                    times = 15)))

# run analysis
out <- cluster_irr(s = s,
                   r = r,
                   version = 2,
                   ks = 4,
                   cores = 1,
                   control = list(
                        global_max_dist = 1,
                        local_max_fdr = 0.05,
                        local_min_ove = 2,
                        local_min_o = 1,
                        trim_flank_aa = 3,
                        global_pairs = NULL,
                        low_mem = FALSE))

# output class
base::class(out)

# output structure
utils::str(out)

# inspect motif enrichment results
knitr::kable(utils::head(slot(out, "clust")$CDR3b$local$m))

# inspect which CDR3bs are globally similar
knitr::kable(utils::head(slot(out, "clust")$CDR3b$global))

# plot graph
plot_graph(out)
```

clust_irr-class *clust_irr class*

Description

Objects of the class `clust_irr` are generated by the function `cluster_irr`. These objects are used to store the clustering results in a structured way, such that they may be used as inputs of other ClustIRR functions (e.g. get_graph, plot_graph, etc.). Below we provide a detailed description of the slots of `clust_irr`. `clust_irr` objects contain two sublists:

- clust:list, contains clustering results for each TCR/BCR chain. The results are stored in separate sub-list named appropriately (e.g. CDR3a, CDR3b, CDR3g, etc.). In the following we who the typical structure of these lists:
  - `local` - list, local clustering results
    - `m` - data.frame, motif enrichment results with columns:
      - `motif` - motif sequence

      - `f_s` - observed motif counts in `s`
      - `f_r` - observed motif counts in `r`
      - `n_s` - number of all observed motifs in `s`
      - `n_r` - number of all observed motifs in `r`
      - `k` - motif length
      - `ove` - mean observed/expected relative motif frequency
      - `ove_ci_l95` - 95% confidence intervals of ove (lower boundary)
      - `ove_ci_h95` - 95% confidence intervals of ove (upper boundary)
      - `p_value` - p-value from Fisher's exact test
      - `fdr` - false discovery rate, i.e. adjusted p-value by Benjamini & Hochberg correction
      - `pass` - logical value indicating whether a motifs are enriched (`pass=TRUE`) given the user-defined thresholds in control
    - `lp` - data.frame, enriched motifs are linked to their original CDR3 sequences and shown as rows in the data.frame with the following columns:
      - `cdr3` - CDR3 amino acid sequence
      - `cdr3_core` - core portion of the CDR3 sequence, obtained by trimming `trim_flank_aa` amino acids (user- defined parameter). If `trim_flank_aa` = 0, then `cdr3` = `cdr3_core`
      - `motif` - enriched motif from `cdr3_core`
  - `global` - matrix, global clustering results. Pairs of globally similar CDR3s are shown in each row of the matrix (analogous to `lp`)
- inputs:list, contains all user provided inputs

Arguments

`clust` list, contains clustering results for each TCR/BCR chain. The results are stored in separate sub-list named appropriately (e.g. CDR3a, CDR3b, CDR3g, etc.)

`inputs` list, contains all user provided inputs

Value

The output is an S4 object of class `clust_irr`

Accessors

To access the slots of `clust_irr` object we have two accessor functions. In the description below, `x` is a `clust_irr` object.

get_clustirr_clust `get_clustirr_clust(x)`: Extract the clustering results (slot clust)

get_clustirr_inputs `get_clustirr_inputs(x)`: Extract the processed inputs (slot inputs)

Examples

```
# inputs
data("CDR3ab")
s <- data.frame(CDR3b = CDR3ab[1:1000, "CDR3b"])
r <- data.frame(CDR3b = CDR3ab[1:5000, "CDR3b"])
```

```
# controls: auxiliary inputs
control <- list(global_max_dist = 1,
                local_max_fdr = 0.05,
                local_min_ove = 2,
                local_min_o = 1,
                trim_flank_aa = 3,
                global_pairs = NULL,
                low_mem = FALSE)

# clust_irr S4 object generated by function cluster_irr
clust_irr_output <- cluster_irr(s = s, r = r, version = 2,
                                ks = 4, cores = 1, control = control)

# clust_irr S4 object generated 'manually' from the individual results
new_clust_irr <- new("clust_irr",
                     clust = slot(object = clust_irr_output, name = "clust"),
                     inputs = slot(object = clust_irr_output, name = "inputs"))

# we should get identical outputs
identical(x = new_clust_irr, y = clust_irr_output)
```

get_graph *Get graph structure from* `clust_irr` *object*

Description

The main output of this function is an igraph object.

The vertices in the graph represent clones. Undirected edges are drawn between a pair of vertices if the corresponding clones that are locally and/or globally similar.

Usage

```
get_graph(clust_irr)
```

Arguments

`clust_irr`	S4 object generated by the function `cluster_irr`

Value

The main output of this function is an igraph object.

Examples

```
# load package input data
data("CDR3ab")
s <- base::data.frame(CDR3b = CDR3ab[1:100, "CDR3b"])
r <- base::data.frame(CDR3b = CDR3ab[1:5000, "CDR3b"])
```

```
# artificially enrich motif 'RWGW' inside sample dataset
base::substr(x = s$CDR3b[1:20], start = 6, stop = 9) <- "RWGW"

# add an artificial clonal expansion of two sequences to the sample dataset
s <- rbind(s, base::data.frame(CDR3b = rep(x = c("CATSRADKPDGLDALETQYF",
                                                 "CATSRAAKPDGLAALSTQYF"),
                                           times = 5)))

# run ClustIRR analysis
out <- cluster_irr(s = s,
                   r = r,
                   version = 2,
                   ks = 4,
                   cores = 1,
                   control = list(trim_flank_aa = 3))

# get graph
g <- get_graph(out)

names(g)
```

get_joint_graph *Joins two graphs obtained from two* `clust_irr` *objects*

Description

As input we take two `clust_irr` objects generated by the function `cluster_irr`.

Using each `clust_irr` object we generate a graph (with the function `get_graph`) in which the different vertices represent clones, and undirected edges are drawn between a pair of vertices if the corresponding clones are locally and/or globally similar (see definitions of local/global clustering in the documentation of `cluster_irr`.

The function `get_joint_graph` performs the following operation on the the two graphs:

First it performs an union of the vertices. Second, it performs global clustering between the two graphs, i.e. it compares the CDR3 sequences of the clones between the two graphs. If two clones have similar CDR3 sequences, then the corresponding vertices are connected by an edge.

The results is another `igraph` object.

Usage

```
get_joint_graph(clust_irr_1, clust_irr_2)
```

Arguments

`clust_irr_1` S4 object generated by the function `cluster_irr`

`clust_irr_2` S4 object generated by the function `cluster_irr`

Value

The main output of this function is an igraph object.

Examples

```
# load package input data
data("CDR3ab")
s <- base::data.frame(CDR3b = CDR3ab[1:100, "CDR3b"])
r <- base::data.frame(CDR3b = CDR3ab[1:5000, "CDR3b"])

# artificially enrich motif 'RWGW' inside sample dataset
base::substr(x = s$CDR3b[1:20], start = 6, stop = 9) <- "RWGW"

# add an artificial clonal expansion of two sequences to the sample dataset
s <- rbind(s, base::data.frame(CDR3b = rep(x = c("CATSRADKPDGLDALETQYF",
                                                  "CATSRAAKPDGLAALSTQYF"),
                                           times = 5)))

# run ClustIRR analysis
c1 <- cluster_irr(s = s,
                  r = r,
                  version = 2,
                  ks = 4,
                  cores = 1,
                  control = list(trim_flank_aa = 3))

# run ClustIRR analysis
c2 <- cluster_irr(s = s,
                  r = r,
                  version = 2,
                  ks = 4,
                  cores = 1,
                  control = list(trim_flank_aa = 3))

# get graph
g <- get_joint_graph(c1, c2)

names(g)
```

---

plot_graph *Plot ClustIRR graph*

Description

This this function visualizes a graph. The input is clust_irr object created by the function cluster_irr.

Usage

```
plot_graph(clust_irr, as_visnet=FALSE)
```

Arguments

`clust_irr`	S4 object of type `clust_irr`, result of `clust_irr` function
`as_visnet`	logical, if as_visnet=TRUE we plot an interactive graph with visNetwork. If as_visnet=FALSE, we plot a static graph with igraph.

Value

The output is an igraph plot.

Vertices are clones and edges represent local or global similarities. Edge attributes 'color', 'linetype' and 'thickness' can be interpreted as follows:

- Edge colors
  - purple: local CDR3 similarity
  - green: global CDR3 similarity
  - black: local + global CDR3 similarity
- Edge linetypes
  - dashed: similarity between CDR3β, CDR3δ, CDR3H
  - dotted: similarity between CDR3α, CDR3γ, CDR3L
  - solid: similarity between CDR3s from both chains (e.g. CDR3α and CDR3β)
- Edge thickness: number of edges between two clones

The size of the vertices increases linearly as the logarithm of the degree of the clonal expansion (number of cells per clone) in the corresponding clones.

Examples

```
# load package input data
data("CDR3ab")
s <- base::data.frame(CDR3b = CDR3ab[1:1000, "CDR3b"])
r <- base::data.frame(CDR3b = CDR3ab[1:5000, "CDR3b"])

# artificially enrich motif 'RWGW' inside sample dataset
base::substr(x = s$CDR3b[1:20], start = 6, stop = 9) <- "RWGW"

# add an artificial clonal expansion of two sequences to the sample dataset
s <- rbind(s, base::data.frame(CDR3b = rep(x = c("CATSRADKPDGLDALETQYF",
                                 "CATSRAAKPDGLAALSTQYF"),
                                 times = 5)))

# run analysis
out <- cluster_irr(s = s,
                   r = r,
                   version = 2,
                   ks = 4,
                   cores = 1,
                   control = list(
                        global_max_dist = 1,
                        local_max_fdr = 0.05,
                        local_min_ove = 2,
                        local_min_o = 1,
                        trim_flank_aa = 3,
                        global_pairs = NULL,
                        low_mem = FALSE))

# plot graph with vertices as clones
p1 <- plot_graph(out, as_visnet=FALSE)
p1

# access nodes and edges of the graph as data.frame
n <- p1$x$nodes
str(n)
class(n)
head(n)

e <- p1$x$edges
str(e)
class(e)
head(e)
```

plot_joint_graph *Plot joint ClustIRR graph*

Description

This this function creates a joint graph from two clust_irr objects, and visualizes the graph.

Usage

```
plot_joint_graph(clust_irr_1, clust_irr_2, as_visnet = FALSE)
```

Arguments

clust_irr_1	S4 object of type clust_irr_1
clust_irr_2	S4 object of type clust_irr_2
as_visnet	logical, if as_visnet=TRUE we plot an interactive graph with visNetwork. If as_visnet=FALSE, we plot a static graph with igraph.

Value

The output is an igraph plot.

Vertices are clones and edges represent local or global similarities. Edge attributes 'color', 'linetype' and 'thickness' can be interpreted as follows:

- Edge colors
  - purple: local CDR3 similarity
  - green: global CDR3 similarity

- black: local + global CDR3 similarity

- Edge linetypes
  - dashed: similarity between CDR3β, CDR3δ, CDR3H
  - dotted: similarity between CDR3α, CDR3γ, CDR3L
  - solid: similarity between CDR3s from both chains (e.g. CDR3α and CDR3β)
- Edge thickness: number of edges between two clones

The size of the vertices increases linearly as the logarithm of the degree of the clonal expansion (number of cells per clone) in the corresponding clones.

Examples

```
# load package input data
data("CDR3ab")
s <- base::data.frame(CDR3b = CDR3ab[1:1000, "CDR3b"])
r <- base::data.frame(CDR3b = CDR3ab[1:5000, "CDR3b"])

# artificially enrich motif 'RWGW' inside sample dataset
base::substr(x = s$CDR3b[1:20], start = 6, stop = 9) <- "RWGW"

# add an artificial clonal expansion of two sequences to the sample dataset
s <- rbind(s, base::data.frame(CDR3b = rep(x = c("CATSRADKPDGLDALETQYF",
                                "CATSRAAKPDGLAALSTQYF"),
                                times = 5)))

# run analysis
out <- cluster_irr(s = s,
                   r = r,
                   version = 2,
                   ks = 4,
                   cores = 1,
                   control = list(
                        global_max_dist = 1,
                        local_max_fdr = 0.05,
                        local_min_ove = 2,
                        local_min_o = 1,
                        trim_flank_aa = 3,
                        global_pairs = NULL,
                        low_mem = FALSE))

# plot graph with vertices as clones
plot_joint_graph(out, out, as_visnet=FALSE)
```

Index