

Package: piglet (via r-universe)

October 8, 2026

Title Program for Inferring Immunoglobulin Allele Similarity Clusters and Genotypes

Version 1.5.2

Author Ayelet Peres [aut, cre], William Lees [aut], Gur Yaari [aut, cph]

Maintainer Ayelet Peres <ayelet.peres@yale.edu>

Description Improves genotype inference and downstream Adaptive Immune Receptor Repertoire Sequence data analysis. Inference of allele similarity clusters, an alternative naming scheme and genotype inference for immunoglobulin heavy chain repertoires. The main tools are allele similarity clusters, and allele based genotype. The first tool is designed to reduce the ambiguity within the immunoglobulin heavy chain V alleles. The ambiguity is caused by duplicated or similar alleles which are shared among different genes. The second tool is an allele based genotype, that determined the presence of an allele based on a threshold derived from a naive population. See Peres et al. (2023) <[doi:10.1093/nar/gkad603](https://doi.org/10.1093/nar/gkad603)>.

License CC BY-SA 4.0

URL <https://github.com/ayeletperes/piglet>

BugReports <https://github.com/ayeletperes/piglet/issues>

Encoding UTF-8

Roxygen list(markdown = TRUE)

Depends R (>= 3.5.0)

LinkingTo Rcpp

SystemRequirements GNU make

Imports Biostrings (>= 2.62.0), DECIPHER (>= 2.22.0), alakazam (>= 1.2.0), dendextend (>= 1.9.0), data.table (>= 1.12.2), tigger (>= 1.0.0), methods (>= 3.4.4), rlang (>= 0.4.0), RColorBrewer (>= 1.1.2), ggplot2 (>= 3.3.6), circlize (>= 0.4.15), gridExtra (>= 2.3), R6 (>= 2.5.1), jsonlite (>= 1.8.3), Rcpp (>= 0.11.0), magrittr, igraph (>= 1.3.0), stringdist (>= 0.9.0), cluster (>= 2.1.0), ape (>= 5.0)

Suggests knitr, rmarkdown, tidyr, htmltools, bookdown, dplyr, ggtree
(>= 3.0.0), testthat (>= 3.0.0)

RoxygenNote 7.3.3

Collate 'Data.R' 'GermlineCluster-class.R' 'RcppExports.R' 'piglet.R'
'allele_cluster.R' 'utils.R' 'allele_genotype.R'
'asc_iuis_labels.R' 'asc_regroup.R' 'community_detection.R'
'deprecated.R' 'piglet-package.R' 'qtl_pairing.R'
'qtl_pairing_leads.R' 'qtl_scan.R' 'qtl_usage.R' 'utils-pipe.R'
'visualization.R'

LazyData true

BuildVignettes true

VignetteBuilder knitr

Config/testthat/edition 3

Config/pak/sysreqs

libglpk-dev make libbz2-dev libicu-dev liblzma-dev libxml2-dev libx11-dev xz-utils zlib1g-dev

Repository <https://ayeletperes.r-universe.dev>

Date/Publication 2026-09-27 22:22:20 UTC

RemoteUrl <https://github.com/ayeletperes/piglet>

RemoteRef HEAD

RemoteSha 39d6600e67d99159f4185d830607c96be163d191

Contents

allele_cluster_table	4
allele_diff	4
allele_diff_indices	5
allele_diff_indices_parallel	6
allele_diff_indices_parallel2	7
allele_diff_paired	8
allele_diff_strings	9
allele_threshold_table	10
alleleClusterNames	10
allelePalette	11
annotateRepertoireIUIS	12
artificialFRW1Germline	13
ascIUISVocabulary	14
ascUsagePhenotype	16
assignAlleleClusters	17
detect_communities_leiden	18
distance_to_graph	19
extractASCTable	20
generateReferenceSet	20
genotypeToTigger	22
germlineASC	23

hv_functionality	24
HVGGERM	25
igClust	25
igDistance	27
ighvClust	28
ighvDistance	29
inferAlleleClusters	29
inferGenotypeAllele	32
inferGenotypeAllele_asc	35
insert_gaps	37
insert_gaps2_vec	38
new_germline_cluster	38
optimize_resolution	40
pairingCellTests	41
pairingLeadCharacter	42
pairingPhenotype	45
pairingScan	46
pairingTable	48
piglet	49
plot.GermlineCluster	50
plotAlleleCluster	51
plotClusterComparison	51
plotCommunityNetwork	52
plotGenotypeAllele	53
plotSilhouetteOptimization	54
plotTruncatedTree	55
print.GermlineCluster	57
qtlClump	57
qtlLDGroups	58
qtlScanMultivariate	59
qtlScanUnivariate	60
qtlSmallestGenotypeClass	61
recentAlleleClusters	62
regroupASCByLabel	63
runGeneUsageQTL	65
runPairingQTL	67
summary.GermlineCluster	69
zenodoArchive	70

allele_cluster_table *Allele similarity cluster table*

Description

A data.table of the allele similarity cluster table based on the HVGERM and hv_functionality germlie reference set. This is not the latest version of the allele similarity cluster table. For the latest version please refer either to the zenodo doi or you can use the recentAlleleClusters

Usage

```
allele_cluster_table
```

Format

An object of class data.table (inherits from data.frame) with 286 rows and 5 columns.

References

Peres, et al (2022) [doi:10.1101/2022.12.26.521922](https://doi.org/10.1101/2022.12.26.521922)

allele_diff *Alleles nucleotide position difference*

Description

Compare the sequences of two alleles (reference and sample alleles) and returns the differential nucleotide positions of the sample allele.

Usage

```
allele_diff(
  reference_allele,
  sample_allele,
  position_threshold = 0,
  snps = TRUE
)
```

Arguments

reference_allele	The nucleotide sequence of the reference allele, character object.
sample_allele	The nucleotide sequence of the sample allele, character object.
position_threshold	A position from which to check for differential positions. If zero checks all position. Default to zero.
snps	If to return the SNP with the position (e.g., A2G where A is for the reference and G is for the sample.). If false returns just the positions. Default to True

Details

The function utilizes c++ script to optimize the run time for large comparisons.

Value

A character vector of the differential nucleotide positions of the sample allele.

Examples

```
{
  reference_allele = "AAGG"
  sample_allele = "ATGA"

  # setting position_threshold = 0 will return all differences
  diff <- allele_diff(reference_allele, sample_allele)
  # "A2T", "G4A"
  print(diff)

  # setting position_threshold = 3 will return the differences from position three onward
  diff <- allele_diff(reference_allele, sample_allele, position_threshold = 3)
  # "G4A"
  print(diff)

  # setting snps = FALSE will return the differences as indices
  diff <- allele_diff(reference_allele, sample_allele, snps = FALSE)
  # 2, 4
  print(diff)
}
```

allele_diff_indices	<i>Calculate differences between characters in columns of germs and return their indices as an int vector.</i>
---------------------	--

Description

Calculate differences between characters in columns of germs and return their indices as an int vector.

Usage

```
allele_diff_indices(germs, X = 0L, non_mismatch_chars_nullable = NULL)
```

Arguments

germs	A vector of strings representing germ sequences.
X	The threshold index from which to return differences as indices.
non_mismatch_chars_nullable	A set of characters that are ignored when comparing sequences (default: 'N', '.', '-').

Value

A vector of integers containing indices of differing columns.

Examples

```
germs = c("ATCG", "ATCC")
X = 3
result = allele_diff_indices(germs, X)
# 1, 2, 3
```

allele_diff_indices_parallel

Deprecated: use [allele_diff_paired](#)

Description

Renamed, and **the counts change**. The old implementation ignored gaps and ambiguous bases only on the germline side, so an aligned position where the germline carried a base and the input carried a gap was counted as a mismatch. It also looped to the length of the germline while indexing the input, reading past the end of the input whenever the input was shorter – undefined behaviour, and on real IGHV data it returned counts derived partly from adjacent memory.

Usage

```
allele_diff_indices_parallel(
  germs,
  inputs,
  X = 0L,
  parallel = FALSE,
  return_count = FALSE
)
```

Arguments

germs	A vector of strings representing germline sequences.
inputs	A vector of strings representing input sequences, paired with germs element by element.
X	The position from which mismatches are counted, zero-based (default 0).
parallel	Ignored. Accepted so existing calls keep working.
return_count	Return the number of mismatches per pair rather than their positions (default FALSE).

Details

Calls now route to [allele_diff_paired](#), which ignores such positions on either side and pads the shorter sequence. Against the old behaviour, 194 of 400 real germline pairs differ, by a mean of 8 mismatches. Anything that depended on the old numbers needs re-checking rather than re-running.

Value

As [allele_diff_paired](#).

See Also

[allele_diff_paired](#)

allele_diff_indices_parallel2

Deprecated: use [allele_diff_paired](#)

Description

Renamed. The old name said `parallel`, which was an argument rather than a behaviour and never had any effect (the package is not built with OpenMP), and `indices`, which is wrong whenever `return_count = TRUE`.

Usage

```
allele_diff_indices_parallel2(  
  germs,  
  inputs,  
  X = 0L,  
  parallel = FALSE,  
  return_count = FALSE,  
  non_mismatch_chars_nullable = NULL  
)
```

Arguments

<code>germs</code>	A vector of strings representing germline sequences.
<code>inputs</code>	A vector of strings representing input sequences, paired with <code>germs</code> element by element.
<code>X</code>	The position from which mismatches are counted, zero-based (default 0).
<code>parallel</code>	Ignored. Accepted so existing calls keep working.
<code>return_count</code>	Return the number of mismatches per pair rather than their positions (default FALSE).
<code>non_mismatch_chars_nullable</code>	Characters ignored on either side when comparing (default N, ., -).

Details

Results are unchanged: this is the same implementation under a clearer name.

Value

As [allele_diff_paired](#).

See Also[allele_diff_paired](#)

allele_diff_paired	<i>Count or locate SNPs between paired germline and input sequences</i>
--------------------	---

Description

Compares each germline sequence in germs against the input sequence at the same position in inputs, and returns either the positions of the mismatches or how many there are. The two vectors are paired element by element, which is what distinguishes this from [allele_diff_indices](#), where every sequence is compared against the first.

Usage

```
allele_diff_paired(
  germs,
  inputs,
  X = 0L,
  return_count = FALSE,
  non_mismatch_chars_nullable = NULL
)
```

Arguments

germs	A vector of strings representing germline sequences.
inputs	A vector of strings representing input sequences, paired with germs element by element.
X	The position from which mismatches are counted, zero-based (default 0).
return_count	Return the number of mismatches per pair rather than their positions (default FALSE).
non_mismatch_chars_nullable	Characters ignored on either side when comparing (default N, ., -).

Details

A position is ignored when *either* sequence carries a non-mismatch character, by default a gap or an ambiguous base (N, ., -). Sequences of unequal length are padded to the longer with N, so the padded positions never count as mismatches.

Value

A list of integer vectors of one-based positions when return_count = FALSE, or an integer vector of counts when TRUE.

Examples

```

germs <- c("ACGTACGT", "ACGTACGT")
inputs <- c("ACGTTCGT", "ACGTTCGA")

allele_diff_paired(germs, inputs, return_count = TRUE) # 1, 2
allele_diff_paired(germs, inputs)                     # list(5), c(5, 8)

```

allele_diff_strings	<i>Calculate differences between characters in columns of germs and return them as a string vector.</i>
---------------------	---

Description

Calculate differences between characters in columns of germs and return them as a string vector.

Usage

```
allele_diff_strings(germs, X = 0L, non_mismatch_chars_nullable = NULL)
```

Arguments

germs	A vector of strings representing germ sequences.
X	The threshold index from which to return differences as strings.
non_mismatch_chars_nullable	A set of characters that are ignored when comparing sequences (default: 'N', '.', '-').

Value

A vector of strings containing differences between characters in columns.

Examples

```

germs = c("ATCG", "ATCC")
X = 3
result = allele_diff_strings(germs, X)
# "A2T", "T3C", "C2G"

```

allele_threshold_table

Allele thresholds table

Description

A data.table of the allele thresholds table. The V alleles are based on the HVGERM and hv_functionality germline reference set. The D, and the J are based on the AIRR-C reference set (<https://zenodo.org/records/10489725>). The table contains these columns: allele - the IUIS allele name, asc_allele - the allele name based on allele similarity clusters (only for V), threshold = the genotype threshold for the alleles.

Usage

```
allele_threshold_table
```

Format

An object of class data.table (inherits from data.frame) with 262 rows and 4 columns.

References

Peres, et al (2022) [doi:10.1101/2022.12.26.521922](https://doi.org/10.1101/2022.12.26.521922)

alleleClusterNames

Allele similarity cluster naming scheme

Description

For a given cluster the function collapse similar sequences and renames the sequences based on the ASC name scheme

Usage

```
alleleClusterNames(
  cluster,
  allele.cluster.table,
  germ.dist,
  chain,
  segment,
  family_prefix = TRUE,
  retain_subgroup = FALSE
)
```

Arguments

cluster	A vector with the cluster identifier - the family and allele cluster number.
allele.cluster.table	A data.frame with the list of all germline sequences and their clusters.
germ.dist	A matrix with the germline distance between the germline set sequences.
chain	A character with the chain identifier: IGH/IGL/IGK/TRB/TRA... (Currently only IGH is supported)
segment	A character with the segment identifier: IGHV/IGHD/IGHJ.... (Currently only IGHV is supported)
family_prefix	Logical. If TRUE (default), prepend "F" to the family number in ASC names (e.g. IGHVF1-G101). If FALSE, omit the "F" (e.g. IGHV1-G101).
retain_subgroup	Logical. If TRUE, retain the original IMGT subgroup in the ASC name instead of renumbering by the family clustering (e.g. IGKV1-12*01 stays in subgroup IGKV1-G.). When TRUE, family_prefix is ignored. Default FALSE.

Value

A data.frame with the clusters renamed alleles based on the ASC scheme.

allelePalette	<i>Allele color palette</i>
---------------	-----------------------------

Description

allelePalette returns a named vector of colors for a set of alleles, reusing the VDJbase allele color scheme (see the vjbasevis package) for the curated allele numbers and generating additional, visually distinct colors when there are more alleles than the curated palette covers. This lets genotype plots scale to an arbitrary number of alleles while staying consistent with VDJbase for the common ones.

Usage

```
allelePalette(alleles)
```

Arguments

alleles	a vector of allele identifiers, typically allele numbers such as "01", "02". A novel-allele suffix after "_" shares the color of its base allele.
---------	---

Details

The curated colors are taken from vjbasevis::allelePalette (alleles 01-19). Alleles beyond that first draw from a pool of additional distinct colors and then, if still short, from evenly spaced hues generated in HCL space.

Value

a named character vector mapping each unique allele to a hex color.

Examples

```
allelePalette(c("01", "02", "06"))
# also covers many alleles without running out of colors
length(allelePalette(sprintf("%02d", 1:40)))
```

```
annotateRepertoireIUIS
```

Add IUIS gene-group columns to a genotype-corrected repertoire

Description

Translates the ASC gene calls on a repertoire into the IUIS labels built by [ascIUISVocabulary](#), adding v_gene_iuis, j_gene_iuis and, where the chain has one, d_gene_iuis.

Usage

```
annotateRepertoireIUIS(
  data,
  vocabulary,
  chain,
  v_gene = "v_gene",
  j_gene = "j_gene",
  d_gene = "d_gene",
  d_call = "d_call_new"
)
```

Arguments

data	A data.frame of rearrangements carrying the ASC gene calls.
vocabulary	An ascIUISVocabulary from ascIUISVocabulary .
chain	The locus of data, one of "IGH", "IGK", "IGL".
v_gene, j_gene, d_gene	Column names holding the ASC gene calls. Defaults "v_gene", "j_gene", "d_gene".
d_call	Column name holding the ASC <i>allele</i> call for D, used for IGH only. Default "d_call_new".

Details

For IGH the D label is per allele, so it is mapped from the allele-level call rather than from the collapsed gene, and a call spanning more than one ASC group names no single gene and stays NA. V and J map straight from their ASC gene.

Rows in and rows out are reported: a call that maps to nothing is left NA rather than dropped, and the count is messaged so a labelling gap cannot pass unnoticed.

Value

data as a `data.table` with the `*_gene_iuis` columns added.

See Also

[ascIUISVocabulary](#)

Examples

```
husa <- data.frame(
  allele      = c("IGHV1-2*01", "IGHV1-2*02", "IGHD1-C05H*01"),
  chain       = "IGH",
  gene_type   = c("IGHV", "IGHV", "IGHD"),
  seq         = c("ACGTACGT", "ACGTACGA", "GGTTGGTT"),
  iuis_allele = c("IGHV1-2*01", "IGHV1-2*02", "IGHD1-20*01"))
voc <- ascIUISVocabulary(husa)

rep_dt <- data.frame(
  v_gene      = c("IGHV1-2", "IGHV1-2"),
  d_gene      = c("IGHD1-C05H", "IGHD1-C05H"),
  j_gene      = c("IGHJ4", "IGHJ4"),
  d_call_new  = c("IGHD1-C05H*01", "IGHD1-C05H*01"))

annotateRepertoireIUIS(rep_dt, voc, chain = "IGH")
```

artificialFRW1Germline

FWR1 artificial dataset generator

Description

A function to artificially create an IGHV reference set with framework1 (FWR1) primers (see Details).

Usage

```
artificialFRW1Germline(
  germline_set,
  mask_primer = TRUE,
  trimm_primer = FALSE,
```

```

    quite = FALSE
  )

```

Arguments

germline_set	A germline set distance matrix created by ighvDistance.
mask_primer	Logical (TRUE by default). If to mask with Ns the region of the primer from the germline sequence
trimm_primer	Logical (FALSE by default). If to trim the region of the primer from the germline sequence. If TRUE then, mask_primer is ignored.
quite	Logical (FALSE by default). Do you want to suppress informative messages

Details

The FRW1 primers used in this function were taken from the BIOMED-2 protocol. For more information on the protocol and primer design go to: van Dongen, J., Langerak, A., Brüggemann, M. et al. Design and standardization of PCR primers and protocols for detection of clonal immunoglobulin and T-cell receptor gene recombinations in suspect lymphoproliferations: Report of the BIOMED-2 Concerted Action BMH4-CT98-3936. Leukemia 17, 2257–2317 (2003). <https://doi.org/10.1038/sj.leu.2403202> Van Dongen, J. J. M., et al. "Design and standardization of PCR primers and protocols for detection of clonal immunoglobulin and T-cell receptor gene recombinations in suspect lymphoproliferations: report of the BIOMED-2 Concerted Action BMH4-CT98-3936." Leukemia 17.12 (2003): 2257-2317.

Value

A list with the input germline set allele and the trimmed/masked sequences.

ascIUISVocabulary	<i>Build the ASC subgroup to IUIS gene-group vocabulary</i>
-------------------	---

Description

Derives the display labels that translate an Allele Similarity Cluster assignment into IUIS gene names, and returns the allele set behind each label.

Usage

```
ascIUISVocabulary(husa)
```

Arguments

husa	A data.frame of the ASC reference (the husa.tsv produced by the ASC pipeline), with columns allele (the ASC allele name), chain, gene_type, seq and iuis_allele. iuis_allele may list several IUIS alleles separated by , or /.
------	---

Details

Two ASC subgroups that share any IUIS gene must carry the same label, so subgroups and IUIS parts are treated as a bipartite graph and every connected component gets one merged label. Where a component holds more than one subgroup the label alone does not identify the subgroup, so `asc_and_iuis_group` suffixes them A, B, C for display.

D is handled differently from V and J, and that asymmetry is the reason ASC labels look inconsistent downstream. D is relabelled at IUIS gene resolution by [regroupASCByLabel](#): an ASC group holding alleles of more than one IUIS gene is broken apart and groups naming the same gene are joined, so the D label is a property of the allele rather than of the ASC gene. V and J keep the ASC grouping. This is why the resulting labels carry the locus for D (IGHD5-12) but not for V and J (V1-18, J4).

The label is a label. It is a display string, and it moves when the reference set behind it moves. What identifies a group is the set of member allele sequences, returned in `identity`; join on that, or compare it to detect that an upstream reference changed underneath a downstream table.

Value

A list of class `ascIUISVocabulary`:

`labels` one row per ASC subgroup: `chain`, `subgroup`, `iuis_group`, `asc_and_iuis_group`, `iuis_group_size`.

`asc_to_iuis` named character vector, ASC subgroup to IUIS label, for V and J.

`d_allele_to_iuis` named character vector, ASC *allele* to IUIS label, for D.

`identity` one row per chain and `iuis_label`: `asc_genes`, `n_alleles`, `member_alleles` and `member_seq_md5`. This is what the label means; the label itself is not.

See Also

[annotateRepertoireIUIS](#), [regroupASCByLabel](#)

Examples

```
husa <- data.frame(
  allele      = c("IGHV1-2*01", "IGHV1-2*02", "IGHD1-C05H*01"),
  chain       = "IGH",
  gene_type   = c("IGHV", "IGHV", "IGHD"),
  seq         = c("ACGTACGT", "ACGTACGA", "GGTTGGTT"),
  iuis_allele = c("IGHV1-2*01", "IGHV1-2*02", "IGHD1-20*01"))

voc <- ascIUISVocabulary(husa)
voc$labels

# What each label stands for. The label is a display string that moves when the
# reference set moves; these member alleles are what identify the group.
voc$identity
```

ascUsagePhenotype	<i>Per-subject usage of each gene group</i>
-------------------	---

Description

A group’s share of its own segment’s repertoire in each subject, with a pseudocount so the logit is finite. This is the phenotype the usage scan explains.

Usage

```
ascUsagePhenotype(  
  data,  
  segments,  
  subject = "subject",  
  min_subject_fraction = 0.1,  
  pseudocount = 0.5  
)
```

Arguments

data	A data.frame of rearrangements, one row per sequence.
segments	Named character vector of the columns holding each segment’s gene call, e.g. c(V = "v_gene_iuis", D = "d_gene_iuis", J = "j_gene_iuis"). Names become the segment column.
subject	Column naming the subject. Default "subject".
min_subject_fraction	Minimum fraction of subjects a group must appear in to be retained. Default 0.1.
pseudocount	Added to the count, and pseudocount * n_asc to the total. Default 0.5.

Details

Zero counts are kept deliberately. A variant that deletes a gene drives its usage to zero, and that is the canonical signal in this family, so dropping zeros would drop the effect. Every subject gets a row for every retained group, whether or not they used it.

Groups seen in fewer than min_subject_fraction of subjects are dropped before the shares are taken, and the count kept is reported. n_asc is how many groups the share was taken over and is part of the pseudocount, so usage cannot be recomputed from count and total without it.

Value

A data.table with subject, asc, count, total, segment, n_asc, usage and logit_usage.

See Also

[runGeneUsageQTL](#)

Examples

```

set.seed(1)
rep_dt <- data.frame(
  subject = rep(sprintf("s%02d", 1:70), each = 20),
  v_gene = sample(c("V1-2", "V3-23", "V4-34"), 1400, replace = TRUE),
  j_gene = sample(c("J4", "J6"), 1400, replace = TRUE))

usage <- asclUsagePhenotype(rep_dt, segments = c(V = "v_gene", J = "j_gene"))
head(usage)

# Zero counts are kept on purpose: a variant that deletes a gene drives its
# usage to zero, and that is the signal rather than missing data.
sum(usage$count == 0)

```

assignAlleleClusters *Assign allele similarity clusters*

Description

assignAlleleClusters uses the allele clusters annotation to change the preliminary allele assignments to the new annotations before inferring a genotype.

Usage

```

assignAlleleClusters(
  data,
  alleleClusterTable,
  v_call = "v_call",
  from_col = "iuis_allele",
  to_col = "new_allele"
)

```

Arguments

data	data.frame in AIRR format, containing V allele calls from a single subject and the sample IMGT-gapped V(D)J sequences under seq.
alleleClusterTable	A data.frame of the allele clusters new annotations relative to the original reference set. See details.
v_call	name of the V allele call column. Default is v_call
from_col	name of the column in alleleClusterTable to use as the source for the dictionary. Default is iuis_allele
to_col	name of the column in alleleClusterTable to use as the target for the dictionary. Default is new_allele

Value

A modified input data.frame with the new assigned

Examples

```
# preferably obtain the latest ASC cluster table
# asc_archive <- recentAlleleClusters(doi="10.5281/zenodo.7429773", get_file = TRUE)

# allele_cluster_table <- extractASCTable(archive_file = asc_archive)

# example allele similarity cluster table
data(allele_cluster_table)

# loading TIgGER AIRR-seq b cell data
data <- tigger::AIRRDb

asc_data <- assignAlleleClusters(data, allele_cluster_table)
```

detect_communities_leiden

Leiden community detection

Description

Performs community detection on a weighted graph using the Leiden algorithm with CPM (Constant Potts Model) objective function.

Usage

```
detect_communities_leiden(g, resolution = 1)
```

Arguments

<code>g</code>	An igraph graph object with weighted edges
<code>resolution</code>	Resolution parameter for Leiden algorithm. Higher values produce more communities. Default is 1.0.

Details

The Leiden algorithm is a community detection method that optimizes a quality function (here CPM). It guarantees connected communities and is generally faster than Louvain while producing better quality partitions.

Value

An igraph communities object

See Also

[distance_to_graph](#), [optimize_resolution](#)

Examples

```
data(HVGGERM)
d <- igDistance(HVGGERM[1:10], method = "hamming")
g <- distance_to_graph(d)
comm <- detect_communities_leiden(g, resolution = 0.5)
```

distance_to_graph	<i>Convert distance matrix to weighted graph</i>
-------------------	--

Description

Converts a distance matrix to a weighted igraph object using a log transform that spreads small distances and produces weights in [0,1].

Usage

```
distance_to_graph(distance_matrix)
```

Arguments

distance_matrix
A distance matrix or dist object

Details

The transformation uses a log-based similarity measure:

1. Normalize distances by max distance
2. Apply -log transform to convert to similarity
3. Normalize similarities to [0,1] range
4. Create weighted undirected graph

Value

An igraph object with weighted edges

See Also

[detect_communities_leiden](#), [igClust](#)

Examples

```
data(HVGGERM)
d <- igDistance(HVGGERM[1:10], method = "hamming")
g <- distance_to_graph(d)
```

extractASCTable	<i>Extracts the allele cluster table from the archive file.</i>
-----------------	---

Description

Extracts the allele cluster table from the archive file.

Usage

```
extractASCTable(archive_file = NULL)
```

Arguments

archive_file A path to the asc archive file. Default is null. (see details)

Details

For downloading the latest archive file with the updated allele cluster table, use the function recentAlleleClusters.

Value

Returns the allele cluster table.

The table columns: new_allele - the ASC given allele name func_group - the ASC cluster number iuis_allele - the original IUIS allele name thresh - the allele threshold for ASC-based genotype inference amplicon_length - is the original length of the reference set.

Examples

```
## Not run:
asc_archive <- recentAlleleClusters(doi="10.5281/zenodo.7429773", get_file = TRUE)

allele_cluster_table <- extractASCTable(archive_file = asc_archive)

## End(Not run)
```

generateReferenceSet	<i>Generate allele similarity reference set</i>
----------------------	---

Description

Generates the allele clusters reference set based on the clustering from [ighvClust](#). The function collapse similar alleles and assign them into their respective allele clusters and family clusters. See details for naming scheme

Usage

```
generateReferenceSet(
  germline_distance,
  germline_set,
  alleleClusterTable,
  trim_3prime_side = NULL,
  family_prefix = TRUE,
  retain_subgroup = FALSE
)
```

Arguments

germline_distance	A germline set distance matrix created by ighvDistance .
germline_set	A character list of the IMGT aligned IGHV allele sequences. See details for curating options.
alleleClusterTable	A data.frame of the alleles and their clusters created by ighvClust .
trim_3prime_side	If a 3' position trim is supplied, duplicated sequences will be checked for differential positions past the trim position. Default NULL; NULL will not activate the check. see @details
family_prefix	Logical. If TRUE (default), prepend "F" to the family number in ASC names (e.g. IGHVF1-G101). If FALSE, omit the "F" (e.g. IGHV1-G101).
retain_subgroup	Logical. If TRUE, retain the original IMGT subgroup in the ASC name instead of renumbering by the family clustering (e.g. IGKV1-12*01 stays in subgroup IGKV1-G..). When TRUE, family_prefix is ignored. Default FALSE.

Details

Each allele is named by this scheme: IGHVF1-G1*01 - IGH = chain, V = region, F1 = family cluster numbering (the "F" prefix can be omitted by setting family_prefix = FALSE), G1 - allele cluster numbering, and 01 = allele numbering (given by clustering order, no connection to the expression)

In case there are alleles that are differentiated in a nucleotide position past the trimming position used for the clustering, then the alleles are separated and are annotated with the differentiating position as so: Say A101 and A102 are similar up to position 318, and thus collapsed in the clusters to G101. Upon checking the sequences past the trim position (318), a differentiating nucleotide was seen in position 319, A101 has a G, and A102 has a T. Then the alleles will be separated, and the new annotation will be as so: A101 = G101, and A102 = G1*01_G319T. Where the first nucleotide indicate the base, the following number the position, and the last nucleotide the one the base changed into.

Value

A list with the re-named germline set, and a table of the allele clusters and thresholds.

genotypeToTigger

Convert a *PIgLET* genotype to a *TiGER/VDJbase* genotype table

Description

genotypeToTigger reshapes the output of [inferGenotypeAllele](#) or [inferGenotypeAllele_asc](#) into the per-gene table layout used by TiGER and VDJbase. The set of genotyped alleles is determined from the allele z-score and a z-score threshold.

Usage

```
genotypeToTigger(
  genotype,
  level = c("gene", "asc"),
  z_threshold = 0,
  file = NULL
)
```

Arguments

genotype	a genotype data.frame produced by inferGenotypeAllele (one row per allele, with a z_score column) or by inferGenotypeAllele_asc (one row per gene, with a genotype_confidence column). The input type is detected automatically.
level	the row key for the output table. "gene" (default) keys by V gene; "asc" keys by the allele similarity cluster. "asc" requires the genotype to carry ASC information (i.e. inferGenotypeAllele_asc output, or inferGenotypeAllele run with asc_annotation or translate_to_asc).
z_threshold	the z-score threshold for calling an allele present in the genotype. Alleles with z_score >= z_threshold are listed in genotyped_alleles. Default is 0.
file	optional path. When supplied, the table is also written as a tab-separated file with data.table::fwrite.

Value

A data.table with one row per gene (or ASC cluster), and columns:

- gene - the V gene or ASC cluster.
- alleles - the candidate allele numbers, comma-separated and ordered by descending count.
- counts - the read counts, comma-separated, matching alleles.
- total - the total read count for the gene.
- depth - the per-locus repertoire depth used as the denominator of the z-score.
- threshold - the per-allele presence thresholds, comma-separated, matching alleles.
- z_score - the per-allele z-scores, comma-separated, matching alleles.
- genotyped_alleles - the allele numbers with z_score >= z_threshold, comma-separated.

See Also

[inferGenotypeAllele](#) and [inferGenotypeAllele_asc](#) for producing the input genotype.

Examples

```
# loading TIGGER AIRR-seq b cell data
data <- tigger::AIRRDb

data(allele_threshold_table)
data(HVGERM)

genotype <- inferGenotypeAllele(
  data = data,
  allele_threshold_table = allele_threshold_table,
  germline_db = HVGERM, find_unmutated = TRUE)

# convert to the TIGGER/VDJbase table layout
geno_table <- genotypeToTigger(genotype)
head(geno_table)
```

germlineASC	<i>Converts IGHV germline set to ASC germline set.</i>
-------------	--

Description

Converts IGHV germline set to ASC germline set.

Usage

```
germlineASC(allele_cluster_table, germline)
```

Arguments

- allele_cluster_table
The allele cluster table.
- germline
An IGHV germline set with matching names to the "iuis_allele" column in the allele_cluster_table.

Value

Returns the IGHV germline set with the ASC allele names.

Examples

```
# preferably obtain the latest ASC cluster table
# asc_archive <- recentAlleleClusters(doi="10.5281/zenodo.7429773", get_file = TRUE)

# allele_cluster_table <- extractASCTable(archive_file = asc_archive)

data(HVGerm)

# example allele similarity cluster table
data(allele_cluster_table)

asc_germline <- germlineASC(allele_cluster_table, germline = HVGerm)
```

hv_functionality

Human IGHV germlines functionality description

Description

A data.table of all 498 human IGHV germline gene segment alleles in IMGT Gene-db release July 2022, with an additional 25 undocumented alleles from VDJbase. The first column is the allele name, the second column is the functionality annotation, the third column is the nt sequence and the last column is the aa sequence.

Usage

```
hv_functionality
```

Format

An object of class data.table (inherits from data.frame) with 521 rows and 4 columns.

References

Xochelli *et al.* (2014) Immunoglobulin heavy variable (IGHV) genes and alleles: new entities, new names and implications for research and prognostication in chronic lymphocytic leukaemia. *Immunogenetics*. 67(1):61-6.

HVGERM	<i>Human IGHV germlines</i>
--------	-----------------------------

Description

A character vector of all 498 human IGHV germline gene segment alleles in IMGT Gene-db release July 2022, with an additional 25 undocumented alleles from VDJbase.

Usage

HVGERM

Format

Values correspond to IMGT-gaped nucleotide sequences (with nucleotides capitalized and gaps represented by '.').

References

Xochelli *et al.* (2014) Immunoglobulin heavy variable (IGHV) genes and alleles: new entities, new names and implications for research and prognostication in chronic lymphocytic leukaemia. *Immunogenetics*. 67(1):61-6.

igClust	<i>Allele similarity clustering</i>
---------	-------------------------------------

Description

Cluster the distance matrix to create allele clusters. Supports both hierarchical clustering (default) and Leiden community detection.

Usage

```
igClust(  
  germline_distance,  
  method = c("hierarchical", "leiden"),  
  distance_method = "decipher",  
  family_threshold = 75,  
  allele_cluster_threshold = 95,  
  cluster_method = "complete",  
  resolution = NULL,  
  target_clusters = NULL,  
  optimize_silhouette = TRUE,  
  ncores = 1,  
  quiet = FALSE  
)
```

Arguments

<code>germline_distance</code>	A germline set distance matrix created by igDistance .
<code>method</code>	Clustering method. One of "hierarchical" (default) or "leiden".
<code>distance_method</code>	The distance method used to compute <code>germline_distance</code> . One of "decipher" (default), "hamming", or "lv". For "decipher", thresholds are similarity percentages in $[0, 100]$; for "hamming" and "lv", thresholds are raw integer mismatch counts.
<code>family_threshold</code>	The threshold for family-level grouping. For <code>distance_method = "decipher"</code> : similarity percentage (default 75). For "hamming" / "lv": maximum number of mismatches (must be $\geq$ <code>allele_cluster_threshold</code>).
<code>allele_cluster_threshold</code>	The threshold for allele-cluster-level grouping (hierarchical only). For <code>distance_method = "decipher"</code> : similarity percentage (default 95). For "hamming" / "lv": maximum number of mismatches (must be $\leq$ <code>family_threshold</code>).
<code>cluster_method</code>	The linkage method for hierarchical clustering (used for family assignment in both methods). Default is "complete".
<code>resolution</code>	Resolution parameter for Leiden clustering. If NULL, will be optimized.
<code>target_clusters</code>	Target number of clusters for Leiden optimization. Default is NULL.
<code>optimize_silhouette</code>	Logical. Optimize resolution using silhouette score (Leiden only). Default is TRUE.
<code>ncores</code>	Number of cores for parallel processing (Leiden only). Default is 1.
<code>quiet</code>	Logical. Suppress messages. Default is FALSE.

Value

A named list that includes:

- `alleleClusterTable`: data.frame of allele clusters
- `threshold`: list of threshold parameters
- `hclustAlleleCluster`: hierarchical clustering object (both methods)
- `communityObject`: community detection result (Leiden method)
- `graphObject`: igraph object (Leiden method)
- `silhouetteScore`: silhouette score (Leiden method)
- `resolutionParameter`: resolution used (Leiden method)

See Also

[igDistance](#), [inferAlleleClusters](#)

igDistance	<i>Germline set alleles distance</i>
------------	--------------------------------------

Description

Calculates the distance between pairs of alleles based on their aligned germline sequences. Supports multiple distance methods for different segment types.

Usage

```
igDistance(
  germline_set,
  AA = FALSE,
  method = c("decipher", "hamming", "lv"),
  trim_3prime = NULL,
  return_type = c("matrix", "dist"),
  quiet = TRUE
)
```

Arguments

germline_set	A character vector of aligned allele sequences. See details for curating options.
AA	Logical (FALSE by default). If TRUE, calculate the distance based on amino acid sequences.
method	Distance calculation method. One of: "decipher": Uses DECIPHER::DistanceMatrix (requires aligned sequences, best for V segments) "hamming": Hamming distance (requires equal length, sequences padded if needed) "lv": Levenshtein distance (handles variable length, good for D/J segments)
trim_3prime	Optional position to trim sequences from 3' end before distance calculation
return_type	One of "matrix" (default) or "dist" to return a dist object
quiet	Logical (TRUE by default). Suppress informative messages

Details

The aligned IMGT IGHV allele germline set can be downloaded from the IMGT site <https://www.imgt.org/> under the section genedb.

For V segments, the "decipher" method is recommended as it handles alignment gaps properly. For D and J segments which may have variable lengths, the "lv" (Levenshtein) method is appropriate.

Value

A matrix or dist object of the computed distances between allele pairs.

See Also

[ighvDistance](#) for backward compatibility wrapper

Examples

```
data(HVGERM)
# Using DECIPHER method (default, for V segments)
d1 <- igDistance(HVGERM[1:10], method = "decipher")

# Using Hamming distance
d2 <- igDistance(HVGERM[1:10], method = "hamming")

# Using Levenshtein distance (good for D/J segments)
d3 <- igDistance(HVGERM[1:10], method = "lv")
```

ighvClust	<i>Allele similarity clustering (deprecated)</i>
-----------	--

Description

This function is deprecated. Use [igClust](#) instead.

Usage

```
ighvClust(
  germline_distance,
  family_threshold = 75,
  allele_cluster_threshold = 95,
  cluster_method = "complete"
)
```

Arguments

germline_distance	A germline set distance matrix created by igDistance .
family_threshold	The threshold for family-level grouping. For distance_method = "decipher": similarity percentage (default 75). For "hamming" / "lv": maximum number of mismatches (must be >= allele_cluster_threshold).
allele_cluster_threshold	The threshold for allele-cluster-level grouping (hierarchical only). For distance_method = "decipher": similarity percentage (default 95). For "hamming" / "lv": maximum number of mismatches (must be <= family_threshold).
cluster_method	The linkage method for hierarchical clustering (used for family assignment in both methods). Default is "complete".

Value

A named list with clustering results.

See Also

[igClust](#) for the current implementation

ighvDistance	<i>Germline set alleles distance (deprecated)</i>
--------------	---

Description

This function is deprecated. Use [igDistance](#) instead.

Usage

```
ighvDistance(germline_set, AA = FALSE)
```

Arguments

germline_set	A character list of aligned IGHV allele sequences.
AA	Logical (FALSE by default). If to calculate the distance based on amino acid sequences.

Value

A matrix of computed distances between allele pairs.

See Also

[igDistance](#) for the current implementation

inferAlleleClusters	<i>Allele similarity cluster</i>
---------------------	----------------------------------

Description

A wrapper function to infer the allele clusters. Supports both hierarchical clustering (default) and Leiden community detection.

Usage

```
inferAlleleClusters(
  germline_set,
  locus = NULL,
  clustering_method = c("hierarchical", "leiden"),
  distance_method = c("decipher", "hamming", "lv"),
  trim_3prime_side = 318,
  mask_5prime_side = 0,
  family_threshold = 75,
  allele_cluster_threshold = 95,
  cluster_method = "complete",
  resolution = NULL,
  target_clusters = NULL,
  optimize_silhouette = TRUE,
  ncores = 1,
  aa_set = FALSE,
  quiet = FALSE,
  family_prefix = TRUE,
  retain_subgroup = FALSE
)
```

Arguments

germline_set	A character vector of Ig sequence alleles (must be gapped by IMGT scheme for optimal results).
locus	The locus type. One of "IGHV", "IGKV", "IGLV", "IGHD", "IGHJ", "IGKJ", "IGLJ". Default is NULL (auto-detected from sequence names).
clustering_method	Clustering method. One of "hierarchical" (default) or "leiden".
distance_method	Distance calculation method. One of "decipher" (default), "hamming", or "lv".
trim_3prime_side	Position to trim sequences from 3' end. Default is 318; NULL uses full length.
mask_5prime_side	Length to mask from 5' side. Default is 0.
family_threshold	Similarity threshold for family level. Default is 75.
allele_cluster_threshold	Similarity threshold for allele cluster level (hierarchical only). Default is 95.
cluster_method	The linkage method for hierarchical clustering (used for family assignment in both methods). Default is "complete".
resolution	Resolution parameter for Leiden clustering. Default is NULL (auto-optimized).
target_clusters	Target number of clusters for Leiden optimization. Default is NULL.
optimize_silhouette	Optimize resolution using silhouette score (Leiden only). Default is TRUE.

ncores	Number of cores for parallel processing (Leiden only). Default is 1.
aa_set	Logical. Is the sequence set amino acids? Default is FALSE.
quiet	Logical. Suppress messages. Default is FALSE.
family_prefix	Logical. If TRUE (default), prepend "F" to the family number in ASC names (e.g. IGHVF1-G101). If FALSE, omit the "F" (e.g. IGHV1-G101).
retain_subgroup	Logical. If TRUE, retain the original IMGT subgroup in the ASC name instead of renumbering the family by the clustering (e.g. IGKV1-12*01 stays in subgroup IGKV1-G.). When TRUE, family_prefix is ignored. Default FALSE.

Details

The distance between pairs of allele sequences is calculated, then the alleles are clustered. For hierarchical clustering, two similarity thresholds define family and allele clusters. For Leiden clustering, community detection identifies clusters at a specified resolution.

The allele cluster names follow this scheme: IGHVF1-G1*01 - IGH = chain, V = region, F1 = family cluster numbering, G1 = allele cluster numbering, 01 = allele numbering (by clustering order)

For V segments, the "decipher" distance method is recommended. For D and J segments with variable lengths, "lv" (Levenshtein) is more appropriate.

Value

An object of class [GermlineCluster](#) containing:

- germlineSet: Modified germline set (3' trimming and 5' masking)
- alleleClusterSet: Renamed germline set with ASC names
- alleleClusterTable: data.frame of allele similarity clusters
- threshold: List of threshold parameters
- hclustAlleleCluster: hclust object (both methods)
- clusteringMethod: Method used ("hierarchical" or "leiden")
- communityObject: Community object (Leiden method)
- graphObject: igraph object (Leiden method)
- silhouetteScore: Silhouette score (Leiden method)
- resolutionParameter: Resolution used (Leiden method)
- locus: Locus identifier

See Also

[igDistance](#), [igClust](#), [plot.GermlineCluster](#)

Examples

```
# load the initial germline set

data(HVGERM)

germline <- HVGERM[!grepl("^[.]", HVGERM)]

# Hierarchical clustering (default)
asc <- inferAlleleClusters(germline)

# Leiden community detection
asc_leiden <- inferAlleleClusters(germline[1:50],
                                clustering_method = "leiden",
                                target_clusters = 10)

## plotting the clusters
plot(asc)
```

inferGenotypeAllele *Allele based genotype inference*

Description

inferGenotypeAllele infer an individual's genotype based on the allele-base method. The method utilize the allele specific threshold to determine the presence of an allele in the genotype. More specifically, based on the allele frequency, repertoire depth, and the specific allele threshold, a confidence level (Z score) is calculated for the presence of the allele in the genotype. The user can select the confidence level for the genotype inference.

Usage

```
inferGenotypeAllele(
  data,
  allele_threshold_table = NULL,
  call = "v_call",
  asc_annotation = FALSE,
  single_assignment = FALSE,
  translate_to_asc = FALSE,
  germline_db = NA,
  novel = NA,
  find_unmutated = FALSE,
  seq = "sequence_alignment",
  default_allele_threshold = 1e-04,
  depth_adjusted_threshold = FALSE,
  z_score_threshold = 0,
  quiet = TRUE
)
```

Arguments

<code>data</code>	data.frame in AIRR format, containing allele calls from a single subject and the sample IMGT-gapped V(D)J sequences under seq.
<code>allele_threshold_table</code>	A data.frame of the alleles and their thresholds.
<code>call</code>	name of the V,D, or J allele call column, i.e v_call, d_call, j_call. Default is v_call
<code>asc_annotation</code>	Logical (FALSE by default). Are the allele calls annotated with the allele similarity clusters.
<code>single_assignment</code>	if TRUE, the method only considers sequence with single assignment for the genotype inference.
<code>translate_to_asc</code>	For V allele calls, collapse identical allele for the genotype inference. Default is FALSE.
<code>germline_db</code>	named vector of sequences containing the germline sequences named in V allele calls and the alleleClusterTable. Only required if find_unmutated is TRUE.
<code>novel</code>	an optional data.frame of novel alleles, as returned by <code>tigger::findNovelAlleles</code> (columns <code>germline_call</code> , <code>polymorphism_call</code> , <code>novel_imgt</code>). When supplied, the novel germline sequences are added to <code>germline_db</code> and the novel alleles become genotype candidates. A novel allele inherits the threshold of its base allele (or the default when the base is absent). Default NA (no novel alleles).
<code>find_unmutated</code>	if TRUE, use <code>germline_db</code> to find which samples are unmutated. Not needed if V allele calls only represent unmutated samples. Only meaningful for V calls: D and J segments are heavily trimmed, so unmutated-call detection is biased and is skipped (with a warning) when the call column is a D or J segment.
<code>seq</code>	name of the column in data with the aligned, IMGT-numbered, V(D)J nucleotide sequence. Default is <code>sequence_alignment</code> .
<code>default_allele_threshold</code>	The default allele threshold for the genotype inference, in case the allele threshold is not in the <code>allele_threshold_table</code> . Default is 1e-04.
<code>depth_adjusted_threshold</code>	Logical (FALSE by default). If TRUE, each allele's presence threshold is raised to a depth-aware floor $\max(T_{ai}, \text{default_allele_threshold}, 1/N)$, where T_{ai} is the allele threshold and N is the per-locus repertoire depth. This prevents shallow repertoires from clearing an unrealistically low threshold.
<code>z_score_threshold</code>	Numeric (0 by default). Alleles with <code>z_score</code> $\geq$ <code>z_score_threshold</code> are flagged as present in the returned <code>in_genotype</code> column. This is a flag only; no rows are dropped.
<code>quiet</code>	Logical (TRUE by default). Do you want to suppress informative messages

Details

In naive repertoires, allele calls where more than one assignment is assigned is rare. Hence, in case the data represents the naive repertoire of a subject it is recommended to use the `find_unmutated=TRUE`

option, to remove mutated sequences. For non-naive population, the allele calls in cases of multiple assignment are treated as belonging to all groups.

Value

A `data.frame` with the inferred V genotype. The table contains the following columns:

- `allele`: The alleles in the `allele_threshold_table`.
- `counts`: The number of reads for each alleles.
- `depth`: The total number of reads in the genotype (Sum of counts).
- `threshold`: The population driven allele thresholds for genotype presence.
- `z_score`: The confidence level for the presence of the allele in the genotype.
- `observed`: Logical, whether the allele was seen in the data (in single or multiple assignment). Alleles that were never seen carry a smoothing pseudo-count and are marked FALSE.
- `in_genotype`: Logical, whether the allele passes `z_score_threshold` (`z_score >= z_score_threshold`).
- `asc_allele`: If `translate_to_asc` is true, the asc allele value from `allele_threshold_table`.

See Also

[inferAlleleClusters](#) will infer the allele clusters based on a supplied V reference set and set the default allele threshold of 1e-04. See [recentAlleleClusters](#) to obtain the latest version of the IGHV allele clusters and the naive population based allele threshold.

Examples

```
# loading TIGGER AIRR-seq b cell data
data <- tigger::AIRRdb

# allele threshold table
data(allele_threshold_table)

data(HVGERM)

# inferring the genotype
genotype <- inferGenotypeAllele(
  data = data,
  allele_threshold_table = allele_threshold_table,
  germline_db = HVGERM, find_unmutated=TRUE)

# filter alleles with z_score >= 0

head(genotype[genotype$z_score >= 0,])
```

inferGenotypeAllele_asc

Allele similarity cluster based genotype inference Testing function

Description

inferGenotypeAllele_asc infer an individual's genotype based on the allele-base method. The method utilize the allele specific threshold to determine the presence of an allele in the genotype. More specifically, the absolute frequency of each allele is calculated and checked against the threshold.

Usage

```
inferGenotypeAllele_asc(
  data,
  alleleClusterTable,
  v_call = "v_call",
  single_assignment = FALSE,
  germline_db = NA,
  novel = NA,
  find_unmutated = FALSE,
  seq = "sequence_alignment",
  confidence_level = NULL,
  default_allele_threshold = 1e-04
)
```

Arguments

data	data.frame in AIRR format, containing V allele calls from a single subject and the sample IMGT-gapped V(D)J sequences under seq.
alleleClusterTable	A data.frame of the allele similarity clusters thresholds.
v_call	name of the V allele call column. Default is v_call
single_assignment	if TRUE, the method only considers sequence with single assignment for the genotype inference.
germline_db	named vector of sequences containing the germline sequences named in V allele calls and the alleleClusterTable. Only required if find_unmutated is TRUE.
novel	an optional data.frame of novel alleles, as returned by <code>tigger::findNovelAlleles</code> (columns <code>germline_call</code> , <code>polymorphism_call</code> , <code>novel_imgt</code>). When supplied, the novel germline sequences are added to <code>germline_db</code> and the novel alleles become genotype candidates. A novel allele inherits the threshold of its base allele (or the default when the base is absent). Default NA (no novel alleles).
find_unmutated	if TRUE, use <code>germline_db</code> to find which samples are unmutated. Not needed if V allele calls only represent unmutated samples.

seq	name of the column in data with the aligned, IMGT-numbered, V(D)J nucleotide sequence. Default is sequence_alignment.
confidence_level	The confidence level on which to filter the inferred genotype alleles. Default is NULL, meaning filtering only based on allele threshold.
default_allele_threshold	The default allele threshold for the genotype inference, in case the allele threshold is not in the alleleClusterTable. Default is 1e-04.

Details

In naive repertoires, allele calls where more than one assignment is assigned is rare. Hence, in case the data represents the naive repertoire of a subject it is recommended to use the `find_unmutated=TRUE` option, to remove mutated sequences. For non-naive population, the allele calls in cases of multiple assignment are treated as belonging to all groups.

Value

A a data.frame with the inferred V genotype. The table contains the following columns:

gene	alleles	iuis_alleles	counts	absolute_fraction	absolute_threshold
allele cluster	the present alleles in the repertoire	the IUIS nomenclature of the alleles	the number of reads for each alleles	the absolute fraction of the alleles	the population distribution thresholds for germline

See Also

[inferAlleleClusters](#) will infer the allele clusters based on a supplied V reference set and set the default allele threshold of 1e-04. See [recentAlleleClusters](#) to obtain the latest version of the IGHV allele clusters and the naive population based allele threshold.

Examples

```
# loading TIGGER AIRR-seq b cell data
data <- tigger::AIRRDb

# preferably obtain the latest ASC cluster table
# asc_archive <- recentAlleleClusters(doi="10.5281/zenodo.7429773", get_file = TRUE)

# allele_cluster_table <- extractASCTable(archive_file = asc_archive)

# example allele similarity cluster table
data(allele_cluster_table)

data(HVGGERM)

# reforming the germline set
asc_germline <- germlineASC(allele_cluster_table, germline = HVGGERM)
```

```
# assigning the ASC alleles
asc_data <- assignAlleleClusters(data, allele_cluster_table)

# inferring the genotype
asc_genotype <- inferGenotypeAllele_asc(
  data = asc_data,
  alleleClusterTable = allele_cluster_table,
  germline_db = asc_germline, find_unmutated=TRUE)
```

insert_gaps	<i>Insert gaps into an ungapped sequence based on a gapped reference sequence.</i>
-------------	--

Description

This function inserts gaps (e.g., . or -) into an ungapped sequence (ungapped) to match the positions of gaps in a reference sequence (gapped). It ensures that the aligned sequence has the same gap structure as the reference. Vectorised over both arguments, which are paired element by element.

Usage

```
insert_gaps(gapped, ungapped)
```

Arguments

gapped	A vector of strings representing the reference sequences with gaps.
ungapped	A vector of strings representing the sequences without gaps.

Value

A vector of strings with gaps inserted to match the gapped reference.

Examples

```
gapped <- c("caggtc..aact", "caggtc---aact")
ungapped <- c("caggtcaact", "caggtcaact")

insert_gaps(gapped, ungapped)  # "caggtc..aact", "caggtc---aact"
```

insert_gaps2_vec	<i>Deprecated: use insert_gaps</i>
------------------	--

Description

Renamed. The 2 marked a second attempt rather than anything about behaviour, and _vec restated that it is vectorised. Results are unchanged.

Usage

```
insert_gaps2_vec(gapped, ungapped, parallel = FALSE)
```

Arguments

gapped	A vector of strings representing the reference sequences with gaps.
ungapped	A vector of strings representing the sequences without gaps.
parallel	Ignored. Accepted so existing calls keep working.

Value

As [insert_gaps](#).

See Also

[insert_gaps](#)

new_germline_cluster	<i>Create a GermlineCluster object</i>
----------------------	--

Description

GermlineCluster is an S3 class that stores the output of [inferAlleleClusters](#). It contains the allele cluster table, clustering objects, and threshold parameters used for inference.

Usage

```
new_germline_cluster(  
  germlineSet,  
  alleleClusterSet,  
  alleleClusterTable,  
  threshold,  
  hclustAlleleCluster = NULL,  
  clusteringMethod = "hierarchical",  
  communityObject = NULL,  
  graphObject = NULL,
```

```

    distanceMatrix = NULL,
    silhouetteScore = NA_real_,
    resolutionParameter = NA_real_,
    locus = "IGHV",
    .familiesCut = NULL
  )

```

Arguments

germlineSet The original germline set provided.

alleleClusterSet
 The renamed germline set with allele clusters.

alleleClusterTable
 The allele cluster table.

threshold The threshold used for family and allele clusters.

hclustAlleleCluster
 A hierarchical clustering object for the germline set, or NULL.

clusteringMethod
 The clustering method used, either "hierarchical" or "leiden".

communityObject
 A community detection object for Leiden clustering, or NULL.

graphObject An **igraph** graph object for Leiden clustering, or NULL.

distanceMatrix The distance matrix used for clustering, or NULL.

silhouetteScore
 The silhouette score for community detection.

resolutionParameter
 The resolution parameter used for Leiden clustering.

locus The locus identifier, for example "IGHV", "IGHD", "IGHJ".

.familiesCut The family-level cut data stored with the clustering result, or NULL.

Value

An object of class "GermlineCluster".

See Also

[inferAlleleClusters](#)

[GermlineCluster](#)

optimize_resolution *Optimize resolution parameter using silhouette score*

Description

Performs a grid search over resolution parameters and selects the one that maximizes the silhouette score.

Usage

```
optimize_resolution(  
  g,  
  distance_matrix,  
  target_clusters = 80,  
  resolution_range_low = 0.1,  
  resolution_range_high = 0.5,  
  max_steps = 20,  
  ncores = 1  
)
```

Arguments

<code>g</code>	An igraph graph object with weighted edges
<code>distance_matrix</code>	The distance matrix (as dist object) used for silhouette calculation
<code>target_clusters</code>	Target number of clusters for initial tuning. Default is 80.
<code>resolution_range_low</code>	Fractional range below tuned resolution. Default is 0.1.
<code>resolution_range_high</code>	Fractional range above tuned resolution. Default is 0.5.
<code>max_steps</code>	Maximum steps for initial tuning. Default is 20.
<code>ncores</code>	Number of cores for parallel processing. Default is 1.

Value

A list containing:

- `results`: data.frame with Resolution, ClusterCount, Silhouette
- `partitions`: list of membership vectors for each resolution
- `best_resolution`: optimal resolution parameter
- `best_partition`: membership vector at optimal resolution
- `best_clusters`: number of clusters at optimal resolution

See Also

[detect_communities_leiden](#), [igClust](#)

pairingCellTests	<i>Per-cell follow-up under a significant pairing hit</i>
------------------	---

Description

Regresses each single anchor-partner cell on the variant, so a significant omnibus can be read as which partners moved and in which direction. Returns the fitted slope alongside the raw phenotype means in the lowest and highest observed genotype class, and how many subjects sat either side, which is what says whether a difference in means is worth anything.

Usage

```
pairingCellTests(
  pairs,
  dosage,
  omnibus,
  conditional,
  alpha = 0.05,
  min_subjects = 60L,
  min_complete_fraction = 0.9
)
```

Arguments

<code>pairs</code>	A <code>data.table</code> from pairingTable .
<code>dosage</code>	Numeric matrix of genotype dosages, variants in rows, subjects in columns.
<code>omnibus</code>	A pairingScan result carrying a logical significant column. It may hold both anchorings stacked together, and normally should: a variant that cleared the omnibus on either side is worth reading cells for on both, and the per-row <code>omnibus_p_value</code> and <code>omnibus_significant</code> are matched on variant and anchor_gene, so the other side's rows simply do not match this side's anchors.
<code>conditional</code>	A string naming what this scan estimates, e.g. "P(J D)". Required.
<code>alpha</code>	Nominal significance for a cell inside a cleared row. Default 0.05.
<code>min_subjects</code>	Minimum complete-case subjects for an anchor to be scanned, and for a missingness group inside it. Default 60.
<code>min_complete_fraction</code>	Minimum fraction of subjects in which a partner gene must be observed for it to stay in the response. Default 0.9.

Details

A cell is only meaningful inside a row the omnibus already called significant, so the scan is restricted to those variants and `omnibus_significant` travels with every row. `marked` applies nominal alpha inside a cleared row; `marked_strict` divides alpha by the row width, the family being that row's partner genes. Both are returned so a downstream table and a figure make the same call rather than each re-deriving it.

Value

A data.table with conditional, variant, anchor_gene, partner_gene, n, beta, p_value, n_low, n_high, mean_low, mean_high, delta_mean, omnibus_p_value, omnibus_significant, min_genotype_group, marked and marked_strict.

See Also

[pairingScan](#)

Examples

```
set.seed(1)
subjects <- sprintf("s%02d", 1:70)
rep_dt <- data.frame(
  subject = rep(subjects, each = 20),
  d_gene = sample(c("D1", "D2", "D3"), 1400, replace = TRUE),
  j_gene = sample(c("J1", "J2"), 1400, replace = TRUE))
pairs <- pairingTable(rep_dt, anchor = "j_gene", partner = "d_gene")
dosage <- matrix(rep(c(0, 1, 2), length.out = 70), nrow = 1,
  dimnames = list("v1", subjects))

omnibus <- pairingScan(pairs, dosage, conditional = "P(J|D)", min_subjects = 60)
omnibus$significant <- omnibus$p_value < 0.05

pairingCellTests(pairs, dosage, omnibus, conditional = "P(J|D)", min_subjects = 60)
```

pairingLeadCharacter *Characterise pairing leads: what kind of signal is this?*

Description

A significant omnibus says a variant is associated with an anchor gene's partner distribution. It does not say the partners were reallocated: the same p-value is produced by a rigid shift of the whole profile, by one unusual subject, and by a variant that simply changes how much the anchor or a partner is used. This runs the follow-ups that separate those, per lead, and returns a verdict.

Usage

```
pairingLeadCharacter(
  leads,
  pairs,
  dosage,
  positions = NULL,
  ancestry = NULL,
  replication_ancestry = "EUR",
  min_replication_n = 40L,
  min_genotype_group = 5L,
  alpha = 0.05,
```

```

    min_sep_d = 0.8,
    max_overlap = 0.7,
    min_mahalanobis = 1,
    min_subjects = 60L,
    min_complete_fraction = 0.9
  )

```

Arguments

leads	Significant leads for one anchoring, from <code>runPairingQTL</code> , carrying variant, anchor_gene, p_value and threshold.
pairs	The <code>pairingTable</code> the scan used.
dosage	Numeric matrix of genotype dosages, variants in rows, subjects in columns.
positions	Optional data.frame of partner_gene and partner_position. Without it rho_position is NA and the per-partner table is left in column order.
ancestry	Optional data.frame of subject and ancestry. Without it ancestry_p and p_replication are NA.
replication_ancestry	Ancestry group to re-test within. Default "EUR".
min_replication_n	Minimum subjects in that group before it is attempted. Default 40.
min_genotype_group	Smallest genotype class at or above which a lead is called well powered. Default 5.
alpha	Nominal significance for the per-partner and margin tests. Default 0.05.
min_sep_d, max_overlap	A partner counts as separated at or above this Cohen's d and at or below this overlap. Defaults 0.8 and 0.70.
min_mahalanobis	Joint separation a lead needs before it is called separated at all. Default 1.
min_subjects	Minimum complete-case subjects for an anchor to be scanned, and for a missingness group inside it. Default 60.
min_complete_fraction	Minimum fraction of subjects in which a partner gene must be observed for it to stay in the response. Default 0.9.

Value

A list of two data.tables: leads, the input with the summary columns and verdict merged on, and per_partner, one row per lead and partner with slopes and effect sizes.

What is tested

Rigid shift against reallocation The mean across partners regressed on dosage is the shift every partner shares (p_common). Removing that mean and testing what is left asks whether partners moved by *different* amounts (p_heterogeneous). The second is the reallocation claim; the

first is not. One deviation column is dropped before the test because deviations from a row mean sum to zero and the covariance would be singular.

Separation mahalanobis, the distance between the highest and lowest genotype class centroids in units of the pooled within-genotype covariance, after residualising on the per-partner slopes. With `overlap_joint` beside it.

One subject `p_worst_loo` refits the omnibus dropping each subject in turn and keeps the worst p-value. A signal that rests on one person shows up here and nowhere else, and a small genotype class is exactly when it happens.

Both margins The enrichment divides both marginals out, so a lead is not pairing-specific until neither margin moved. Logit marginal usage is regressed on dosage for the anchor (`anchor_usage_p`) and for every partner (`partner_usage_p_min`, Bonferroni across partners).

Ancestry `ancestry_p`, whether the dosage itself is stratified, and `p_replication`, whether the omnibus holds inside one ancestry group. A cis variant is expected to track ancestry, so neither is a rejection on its own.

Position `rho_position`, Spearman of per-partner slope against partner position. A monotone trend along the locus is a different claim from a scattered one. Needs `positions`.

The verdict

`verdict` is assigned in order: `single_subject_driven`, `overlapping_small_effect`, `pairing_plus_both_margins`, `pairing_plus_anchor_usage`, `pairing_plus_partner_usage`, and otherwise `pairing_only`. Its cutoffs are arguments rather than constants, and their defaults are the ones the heavy-chain D/J analysis used. They are that analysis's policy, not facts about the method: a different cohort or a different pair of segments should set them deliberately rather than inherit them.

See Also

[runPairingQTL](#), [pairingCellTests](#)

Examples

```
set.seed(1)
subjects <- sprintf("s%02d", 1:70)
rep_dt <- data.frame(
  subject = rep(subjects, each = 20),
  d_gene = sample(c("D1", "D2", "D3"), 1400, replace = TRUE),
  j_gene = sample(c("J1", "J2"), 1400, replace = TRUE))
pairs <- pairingTable(rep_dt, anchor = "j_gene", partner = "d_gene")
dosage <- matrix(rep(c(0, 1, 2), length.out = 70), nrow = 1,
  dimnames = list("v1", subjects))

leads <- pairingScan(pairs, dosage, conditional = "P(J|D)", min_subjects = 60)
leads$threshold <- 0.05

out <- pairingLeadCharacter(leads, pairs, dosage, min_subjects = 60)
out$leads$verdict
out$per_partner
```

pairingPhenotype*The response matrix for one anchor gene*

Description

Subjects by partner gene, of enrichment: the multivariate response [pairingScan](#) tests each variant against. Exported because the follow-up analyses that characterise a hit need the same matrix the scan saw, and rebuilding it slightly differently is how a follow-up stops describing the thing it followed up.

Usage

```
pairingPhenotype(  
  pairs,  
  anchor,  
  subjects,  
  min_complete_fraction = 0.9,  
  min_subjects = 60L  
)
```

Arguments

<code>pairs</code>	A data.table from pairingTable .
<code>anchor</code>	The anchor gene to build the response for.
<code>subjects</code>	Subjects to keep, normally <code>colnames(dosage)</code> .
<code>min_complete_fraction</code>	Minimum fraction of subjects in which a partner gene must be observed. Default 0.9.
<code>min_subjects</code>	Minimum complete-case subjects. Default 60.

Details

Partner genes observed in too few subjects are dropped *before* complete cases are taken, so one rare partner cannot cost the anchor its whole subject set. Returns NULL when fewer than two partners or fewer than `min_subjects` subjects survive, which is the signal to skip this anchor rather than an error.

Value

A numeric matrix, subjects in rows (named) and partner genes in columns, or NULL.

See Also

[pairingScan](#)

Examples

```
set.seed(1)
subjects <- sprintf("s%02d", 1:70)
rep_dt <- data.frame(
  subject = rep(subjects, each = 20),
  d_gene = sample(c("D1", "D2", "D3"), 1400, replace = TRUE),
  j_gene = sample(c("J1", "J2"), 1400, replace = TRUE))
pairs <- pairingTable(rep_dt, anchor = "j_gene", partner = "d_gene")

# Subjects by partner gene, of enrichment: what one anchor is scanned on.
pheno <- pairingPhenotype(pairs, "J1", subjects, min_subjects = 60)
dim(pheno)
```

pairingScan

Scan variants against a gene's partner distribution

Description

One anchor gene at a time, tests every variant against the whole vector of partner enrichments by MANOVA (Pillai's trace, via [qtlScanMultivariate](#)). A variant that merely raises or lowers how much the anchor is used shifts every partner in the same direction and produces no signal; one that repartitions the anchor's partners does.

Usage

```
pairingScan(
  pairs,
  dosage,
  conditional,
  anchors = NULL,
  min_subjects = 60L,
  min_complete_fraction = 0.9,
  min_genotype_group = 5L
)
```

Arguments

<code>pairs</code>	A data.table from pairingTable .
<code>dosage</code>	Numeric matrix of genotype dosages, variants in rows, subjects in columns.
<code>conditional</code>	A string naming what this scan estimates, e.g. "P(J D)". Required.
<code>anchors</code>	Anchor genes to scan. Default all of them.
<code>min_subjects</code>	Minimum complete-case subjects for an anchor to be scanned, and for a missingness group inside it. Default 60.
<code>min_complete_fraction</code>	Minimum fraction of subjects in which a partner gene must be observed for it to stay in the response. Default 0.9.

min_genotype_group

Smallest genotype class at or above which a fit is called well powered. Default 5.

Details

conditional has no default and must be stated. The two anchorings are separate scans over the same variants, so a result that does not say which one it is cannot be pooled, counted or plotted without double-reporting. The direction is easy to invert by mistake and has been inverted once: at fixed partner,

$$\text{enrichment} = \log_2 P(\text{partner} \mid \text{anchor}) - \log_2 P(\text{partner})$$

and the second term is constant across the anchors, so the anchored response vector is $\log_2 P(\text{partner} \mid \text{anchor})$ shifted by a constant. Anchoring on J therefore gives the P(J|D) scan, and anchoring on D gives P(D|J).

min_genotype_group is counted on each anchor's own subject set, since complete cases differ between anchors and a variant can be well powered for one anchor and not another. It is returned on every row, not only on the leads: a p-value from this scan cannot be read without it.

Value

A data.table with conditional, variant, anchor_gene, n, n_response, pillai, f_stat, p_value, min_genotype_group and well_powered. No significance threshold is applied: set it from the number of independent variants ([qt1LDGroups](#)), not the number tested.

See Also

[pairingTable](#), [pairingCellTests](#)

Examples

```
set.seed(1)
subjects <- sprintf("s%02d", 1:70)
rep_dt <- data.frame(
  subject = rep(subjects, each = 20),
  d_gene = sample(c("D1", "D2", "D3"), 1400, replace = TRUE),
  j_gene = sample(c("J1", "J2"), 1400, replace = TRUE))
pairs <- pairingTable(rep_dt, anchor = "j_gene", partner = "d_gene")

dosage <- matrix(rep(c(0, 1, 2), length.out = 70), nrow = 1,
  dimnames = list("v1", subjects))

# `conditional` has no default: anchoring on J tests P(J|D), anchoring on D
# tests P(D|J), and the two are separate scans over the same variants.
pairingScan(pairs, dosage, conditional = "P(J|D)", min_subjects = 60)
```

pairingTable	<i>Observed against expected gene pairing, per subject</i>
--------------	--

Description

Builds the anchor-by-partner contingency of one subject's repertoire and compares it to what independence of the two marginals would give. This is the phenotype the conditional pairing scan explains.

Usage

```
pairingTable(data, anchor, partner, subject = "subject", pseudocount = 0.5)
```

Arguments

data	A data.frame of rearrangements, one row per sequence.
anchor	Column holding the gene to condition on.
partner	Column holding the partner gene whose distribution is the response.
subject	Column naming the subject. Default "subject".
pseudocount	Added to both count and expectation before the log ratio, so a zero cell gives a finite enrichment. Default 0.5.

Details

Both marginals are divided out. expected is the product of the two margins over the subject's depth, so enrichment measures pairing preference rather than how much either gene is used. That distinction is the whole analysis: in the heavy chain the individual-level signal in the raw conditional p_partner_given_anchor was found to be almost entirely marginal usage, and it went to roughly zero once both marginals were removed. The raw conditionals are returned for inspection, but enrichment is what should be scanned.

Zero cells are created only where both genes occur somewhere in that subject, so a gene the subject does not carry stays a structural absence rather than a measured zero. This matters: a deletion is not the same observation as a pairing the subject never makes.

The enrichment cell is symmetric – swapping which margin conditions leaves the number unchanged – so to scan the other side, call this again with anchor and partner swapped. The omnibus test is *not* symmetric, because grouping the cells by one segment gives each anchor a response vector of a different width, which is a different multivariate test on a different covariance structure. Both sides are therefore separate scans and their results must not be pooled.

Value

A data.table with one row per subject, anchor gene and partner gene: subject, anchor_gene, partner_gene, count, depth, anchor_total, partner_total, expected, enrichment, p_anchor, p_partner, p_partner_given_anchor, p_anchor_given_partner. The anchor and partner column names are kept on the result as attributes.

Which combinations are established

IGH D-anchored and J-anchored pairing is the analysis this was developed for and the one that has been checked. IGK/IGL V-J, and IGH V-D and V-J, run through the same code and are **exploratory**: nobody has shown that pairing preference in those segments survives removing the marginals, and the heavy-chain result above is a reason to expect trouble rather than reassurance. Treat their output as a hypothesis.

See Also

[pairingScan](#), [pairingCellTests](#)

Examples

```
set.seed(1)
rep_dt <- data.frame(
  subject = rep(sprintf("s%02d", 1:70), each = 20),
  d_gene = sample(c("D1", "D2", "D3"), 1400, replace = TRUE),
  j_gene = sample(c("J1", "J2"), 1400, replace = TRUE))

pairs <- pairingTable(rep_dt, anchor = "j_gene", partner = "d_gene")
head(pairs)

# The enrichment is symmetric, so swapping the roles gives the mirrored table.
# The omnibus that follows is not, which is why the two are separate scans.
mirrored <- pairingTable(rep_dt, anchor = "d_gene", partner = "j_gene")
```

piglet

The Program for Ig clusters (PIgLET) package

Description

PIgLET is a suite of computational tools that improves genotype inference and downstream AIRR-seq data analysis. The package has two main tools. The first is Allele Clusters, this tool is designed to reduce the ambiguity within the IGHV alleles. The ambiguity is caused by duplicated or similar alleles which are shared among different genes. The second tool is an allele based genotype, that determined the presence of an allele based on a threshold derived from a naive population.

Allele Similarity Cluster

This section provides the functions that support the main tool of creating the allele similarity cluster from an IGHV germline set.

- [inferAlleleClusters](#): The main function of the section to create the allele clusters based on a germline set.
- [ighvDistance](#): Calculate the distance between IGHV aligned germline sequences.
- [ighvClust](#): Hierarchical clustering of the distance matrix from ighvDistance.
- [generateReferenceSet](#): Generate the allele clusters reference set.

- [plotAlleleCluster](#): Plots the Hierarchical clustering.
- [artificialFRW1Germline](#): Artificially create an IGHV reference set with framework1 (FWR1) primers.

Allele based genotype

This section provides the functions to infer the IGHV genotype using the allele based method and the allele clusters thresholds

- [inferGenotypeAllele](#): Infer the IGHV genotype using the allele based method.
- [assignAlleleClusters](#): Renames the v allele calls based on the new allele clusters.
- [germlineASC](#): Converts IGHV germline set to ASC germline set.
- [recentAlleleClusters](#): Download the most recent version of the allele clusters table archive from zenodo.
- [extractASCTable](#): Extracts the allele cluster table from the zenodo archive file.
- [zenodoArchive](#): An R6 object to query the zenodo api.

References

1. ##

plot.GermlineCluster *Plot method for GermlineCluster*

Description

Plot method for GermlineCluster

Usage

```
## S3 method for class 'GermlineCluster'
plot(x, y = NULL, cex = 1, seed = 9999, ...)
```

Arguments

x	GermlineCluster object
y	Not used
cex	Controls the size of the allele label. Default is 1.
seed	Set a seed number for drawing the dendrogram. Default 9999.
...	Additional arguments passed to plotting functions

Value

A plot of the allele clusters dendrogram

plotAlleleCluster	<i>Plotting the dendrogram of the clusters</i>
-------------------	--

Description

Plotting the dendrogram of the clusters

Usage

```
plotAlleleCluster(x, y = NULL, cex = 1, seed = 9999)
```

Arguments

x	The GermlineCluster object. See inferAlleleClusters
y	NULL. not in use.
cex	Controls the size of the allele label. Default is 1.
seed	Set a seed number for drawing the dendrogram. Default 9999.

Value

A plot of the allele clusters dendrogram

plotClusterComparison	<i>Compare hierarchical and Leiden clustering</i>
-----------------------	---

Description

Creates a comparison visualization showing cluster assignments from both methods.

Usage

```
plotClusterComparison(hierarchical_result, leiden_result, ...)
```

Arguments

hierarchical_result	GermlineCluster object from hierarchical clustering
leiden_result	GermlineCluster object from Leiden clustering
...	Additional arguments

Value

A ggplot object showing cluster agreement

See Also

[inferAlleleClusters](#)

plotCommunityNetwork *Plot community network*

Description

Creates a network visualization of allele clusters from community detection.

Usage

```
plotCommunityNetwork(
  x,
  layout = c("fr", "kk", "circle"),
  node_color = "cluster",
  node_size = "degree",
  edge_alpha = 0.3,
  show_labels = TRUE,
  label_size = 3,
  ...
)
```

Arguments

x	A GermlineCluster object with Leiden clustering
layout	Network layout: "fr" (Fruchterman-Reingold, default), "kk" (Kamada-Kawai), or "circle"
node_color	Variable for node color: "cluster" (default), "family", or a color value
node_size	Variable for node size: "degree" (default), "fixed", or a numeric value
edge_alpha	Alpha transparency for edges. Default is 0.3.
show_labels	Logical. Show node labels. Default is TRUE.
label_size	Size of node labels. Default is 3.
...	Additional arguments

Details

This function creates a network visualization showing:

- Nodes representing alleles, colored by cluster
- Edges weighted by sequence similarity
- Layout optimized by specified algorithm

Value

A ggplot object

See Also

[inferAlleleClusters](#), [detect_communities_leiden](#)

Examples

```
data(HVGEM)
asc <- inferAlleleClusters(HVGEM[1:30],
                           clustering_method = "leiden",
                           target_clusters = 5)
plotCommunityNetwork(asc)
```

plotGenotypeAllele	<i>Plot a PIGLET genotype with a per-allele confidence panel</i>
--------------------	--

Description

plotGenotypeAllele draws a genotype bar plot in the style of `tigger::plotGenotype`, with an aligned confidence panel. Unlike TIGGER, which carries a single confidence value per gene, PIGLET assigns a z-score to each allele; the confidence panel therefore mirrors the allele bars and colors each allele segment by its own z-score.

Usage

```
plotGenotypeAllele(
  genotype,
  level = c("gene", "asc"),
  z_threshold = 0,
  gene_sort = c("name", "position"),
  text_size = 12,
  silent = FALSE,
  ...
)
```

Arguments

genotype	a genotype table from genotypeToTigger , or a raw genotype from inferGenotypeAllele / inferGenotypeAllele_asc (converted automatically via genotypeToTigger).
level	row key when converting a raw genotype: "gene" (default) or "asc". Ignored when genotype is already a converted table.
z_threshold	z-score threshold used to select the genotyped alleles when converting a raw genotype. Default 0.
gene_sort	gene ordering, passed to <code>alakazam::sortGenes</code> : "name" (default) or "position".
text_size	point size of the plotted text.
silent	if TRUE return the grob without drawing it.
...	additional arguments passed to <code>ggplot2::theme</code> .

Details

Only the genotyped alleles (those with `z_score >= z_threshold`) are shown. Each gene/ASC row is split into equal segments, one per genotyped allele; the left panel colors the segments by allele identity and the right panel colors the same segments by their z-score on a continuous blue scale.

Value

A gridExtra grob combining the allele panel and the per-allele z-score confidence panel (returned invisibly).

See Also

[genotypeToTigger](#), [inferGenotypeAllele](#), [inferGenotypeAllele_asc](#)

Examples

```
## Not run:
data <- tigger::AIRRDb
data(allele_threshold_table)
data(HVGGERM)
genotype <- inferGenotypeAllele(data, allele_threshold_table = allele_threshold_table,
                                germline_db = HVGGERM, find_unmutated = TRUE)
plotGenotypeAllele(genotype)

## End(Not run)
```

plotSilhouetteOptimization

Plot silhouette optimization results

Description

Creates a plot showing silhouette score and cluster count across resolution values.

Usage

```
plotSilhouetteOptimization(optimization_result, highlight_best = TRUE, ...)
```

Arguments

```
optimization_result      Result from optimize\_resolution
highlight_best           Logical. Highlight optimal resolution. Default is TRUE.
...                      Additional arguments
```

Value

A ggplot object

See Also

[optimize_resolution](#), [igClust](#)

Examples

```
data(HVGERM)
d <- igDistance(HVGERM[1:30], method = "hamming")
g <- distance_to_graph(d)
opt <- optimize_resolution(g, d, target_clusters = 5)
plotSilhouetteOptimization(opt)
```

plotTruncatedTree	<i>Plot truncated tree visualization</i>
-------------------	--

Description

Creates a circular or dendrogram tree visualization collapsed to ASC subgroup level, with optional heatmap annotations showing family assignments.

Usage

```
plotTruncatedTree(
  x,
  layout = c("circular", "dendrogram"),
  collapse_to = c("asc_subgroup", "iuis_subgroup", "family"),
  label_style = c("asc", "iuis", "both"),
  show_threshold_line = TRUE,
  threshold = 0.25,
  tip_size_by = "n_alleles",
  tip_color_by = "present",
  show_heatmap = TRUE,
  label_size = 7,
  ...
)
```

Arguments

x	A GermlineCluster object from inferAlleleClusters
layout	Tree layout: "circular" (default) or "dendrogram"
collapse_to	Level to collapse tree: "asc_subgroup" (default, based on ASC names), "iuis_subgroup" (based on original IUIS gene names), or "family"

label_style	Label style for tips: "asc" (default, show ASC names like IGHVF1-G1), "iuis" (show IUIS names with superscript markers if ASC splits IUIS group), or "both" (show both names)
show_threshold_line	Logical. Show threshold line on tree. Default is TRUE.
threshold	Threshold height for threshold line (0-1 scale). Default is 0.25.
tip_size_by	Variable for tip point size: "n_alleles" (default), "fixed", or NULL
tip_color_by	Variable for tip point color: "present" (default), "fraction_novel", or NULL
show_heatmap	Logical. Show heatmap annotation for IUIS vs ASC families. Default is TRUE.
label_size	Size of tip labels. Default is 7.
...	Additional arguments passed to ggtree

Details

This function creates a publication-quality tree visualization that:

- Renames tree tips from original allele names to ASC names (new_allele)
- Collapses alleles to ASC subgroup level (single representative per ASC group)
- Shows tip point size by number of alleles in cluster
- Adds optional heatmap track showing IUIS vs ASC family assignments
- Draws threshold line at specified height

When using label_style = "iuis", if multiple ASC groups split a single IUIS subgroup, the labels are marked with superscript letters (e.g., IGHV1-2^A, IGHV1-2^B) to distinguish them.

Requires the ggtree package to be installed.

Value

A ggplot/ggtree object

See Also

[inferAlleleClusters](#), [plot.GermlineCluster](#)

Examples

```
data(HVGERM)
asc <- inferAlleleClusters(HVGERM[1:50])

# Basic truncated tree with ASC labels
if (requireNamespace("ggtree", quietly = TRUE)) {
  plotTruncatedTree(asc, show_heatmap = FALSE)

  # With IUIS labels (marked if ASC splits IUIS group)
  plotTruncatedTree(asc, label_style = "iuis", show_heatmap = FALSE)
}
```

print.GermlineCluster *Print method for GermlineCluster*

Description

Print method for GermlineCluster

Usage

```
## S3 method for class 'GermlineCluster'
print(x, ...)
```

Arguments

x	A GermlineCluster object
...	Additional arguments (ignored)

Value

Invisibly returns x

qt1Clump *Independent lead variants by greedy clumping*

Description

Takes the strongest signal, absorbs everything correlated with it above r^2 , and repeats. What comes back is one row per independent signal rather than one row per significant variant, which is the number to quote.

Usage

```
qt1Clump(assoc, dosage, r2 = 0.8)
```

Arguments

assoc	A data.table of significant associations carrying at least variant and p_value. Reordered in place.
dosage	Numeric matrix of genotype dosages, variants in rows, subjects in columns, covering every variant in assoc. Missing calls are mean-imputed within the variant for the correlation only.
r2	Squared-correlation threshold above which a variant is absorbed into an already-taken lead.

Details

Ties are broken on the variant name, not on the p-value alone. Variants in perfect linkage disequilibrium give the same fit to the last bit that the arithmetic carries, and their p-values then differ by around 1e-13 purely from the order of operations in the matrix product. Sorting on the p-value alone lets that noise decide which variant is named as the lead, and the lead variant name is what reaches a figure and a database label. So the sort is on the p-value and then the name, which is still an arbitrary choice between variants no data distinguishes, but the same arbitrary choice every run. A lead should be read as one representative of its linkage group.

Value

The subset of assoc that are leads, in increasing p-value order.

Examples

```
dosage <- rbind(lead = c(0, 1, 2, 0, 1, 2, 1, 0),
                 linked = c(0, 1, 2, 0, 1, 2, 1, 0),
                 other = c(0, 0, 1, 2, 2, 1, 0, 1))
colnames(dosage) <- sprintf("s%d", 1:8)

assoc <- data.table::data.table(
  variant = c("lead", "linked", "other"),
  p_value = c(1e-8, 2e-8, 1e-4))

# `linked` is absorbed into `lead`; two independent signals remain.
qtlClump(assoc, dosage, r2 = 0.8)
```

qtlLDGroups	<i>Collapse variants carrying identical genotype information</i>
-------------	--

Description

Two variants carry the same information when their standardised dosage vectors match up to sign, so the mirrored case is folded in. This is exact linkage disequilibrium rather than a correlation threshold; use [qtlClump](#) for the latter.

Usage

```
qtlLDGroups(dosage, contig)
```

Arguments

- dosage Numeric matrix of genotype dosages, variants in rows, subjects in columns. Missing calls are mean-imputed within the variant for the comparison only.
- contig Character vector, one contig per row of dosage.

Details

The number of distinct groups is the number of independent tests, which is what a Bonferroni threshold should be set from – the number of variants tested overstates it, often by half. The `contig` prefix stops identical patterns at different loci from merging, so positions from different loci are never pooled.

Value

A character vector named by variant, one group key per row of dosage. The key is opaque: use it to group and count, not to read. Variants with no variance each get their own group rather than collapsing together.

Examples

```
dosage <- rbind(a      = c(0, 1, 2, 0, 1, 2),
               copy    = c(0, 1, 2, 0, 1, 2), # identical
               mirror  = c(2, 1, 0, 2, 1, 0), # identical up to sign
               other    = c(0, 0, 1, 2, 2, 1))
colnames(dosage) <- sprintf("%d", 1:6)

groups <- qtlLDGroups(dosage, rep("chr2", 4))
length(unique(groups)) # three independent tests, not four
```

qtlScanMultivariate *Multivariate QTL scan by Pillai's trace*

Description

Scores every variant against a multivariate response, one MANOVA per variant. For a one degree of freedom predictor Pillai's trace has a closed form,

$$V = (u'S^{-1}u)/(g'g), \quad u = Y'_c g_c, \quad S = Y'_c Y_c$$

and the F approximation is exact, so a whole missingness group is scored by one matrix product instead of a `stats::manova()` call per variant. The package tests check this against `stats::manova()`.

Usage

```
qtlScanMultivariate(pheno, dosage, min_n = 60L)
```

Arguments

<code>pheno</code>	Numeric matrix of the multivariate response, subjects in rows, response variables in columns. Complete cases only.
<code>dosage</code>	Numeric matrix of genotype dosages, variants in rows (row names are variant identifiers) and subjects in columns, in the same subject order as <code>pheno</code> . NA marks no call.
<code>min_n</code>	Minimum number of complete-case subjects a missingness group needs before it is scanned. Default 60.

Details

Response columns with zero variance are dropped within each missingness group, and a group is skipped when fewer than two response columns survive or when n is too small for the covariance to be estimable ($n < p + 3$).

Value

A data.table with variant, n, n_response (response columns retained), pillai, f_stat and p_value.

See Also

[pairingScan](#), which uses this for the conditional pairing scan

Examples

```
set.seed(1)
subjects <- sprintf("s%02d", 1:80)
dosage <- rbind(v_hit = rep(c(0, 1, 2), length.out = 80),
               v_null = rep(c(0, 0, 1, 2), length.out = 80))
colnames(dosage) <- subjects
pheno <- matrix(rnorm(80 * 4), 80, 4,
               dimnames = list(subjects, paste0("partner", 1:4)))
pheno[, 1] <- pheno[, 1] + 0.8 * dosage["v_hit", ]

qtlScanMultivariate(pheno, dosage, min_n = 60)
```

qtlScanUnivariate	<i>Univariate QTL scan of every phenotype on every variant</i>
-------------------	--

Description

Ordinary least squares of each phenotype column on each variant, for the whole matrix at once. The slope of a simple regression is a ratio of cross products, so the entire scan is two matrix products per missingness group rather than one `lm()` call per variant-phenotype pair.

Usage

```
qtlScanUnivariate(pheno, dosage, min_n = 60L)
```

Arguments

pheno	Numeric matrix of phenotypes, subjects in rows (row names are subject identifiers) and phenotypes in columns (column names name the phenotype). No missing values; drop or impute before calling.
dosage	Numeric matrix of genotype dosages, variants in rows (row names are variant identifiers) and subjects in columns, in the same subject order as pheno. NA marks no call.

`min_n` Minimum number of complete-case subjects a missingness group needs before it is scanned. Default 60.

Details

Variants are grouped by which subjects have no call, so every fit inside a group uses the same complete-case subject set and `n` is exact for each row rather than an upper bound. Groups smaller than `min_n` are skipped entirely.

A p-value from this scan must not be read without the smallest genotype class behind the fit: the extreme tail is anti-conservative, and a non-significant result means either no effect or no power. Use [qtlSmallestGenotypeClass](#) on the same dosage matrix, as [runGeneUsageQTL](#) does. Filtering associations on the p-value alone preferentially removes the underpowered rows, which leaves what survives looking better powered than it is.

Value

A `data.table` with one row per variant-phenotype pair that produced a finite p-value: `variant`, `phenotype`, `n` (subjects in the fit), `beta`, `se`, `t_stat`, `p_value`. Effect direction is relative to the coding of dosage, which is not necessarily the minor allele, so `beta` must not be described in terms of a named allele without resolving polarity first.

See Also

[qtlScanMultivariate](#) for a multivariate response, [qtlSmallestGenotypeClass](#), [runGeneUsageQTL](#)

Examples

```
set.seed(1)
subjects <- sprintf("s%02d", 1:80)
dosage <- rbind(v_hit = rep(c(0, 1, 2), length.out = 80),
               v_null = rep(c(0, 0, 1, 2), length.out = 80))
colnames(dosage) <- subjects
pheno <- cbind(gene_a = 0.7 * dosage["v_hit", ] + rnorm(80),
               gene_b = rnorm(80))
rownames(pheno) <- subjects

qtlScanUnivariate(pheno, dosage, min_n = 60)

# Read a p-value with the smallest genotype class beside it, never alone.
qtlSmallestGenotypeClass(dosage)
```

`qtlSmallestGenotypeClass`

Smallest observed genotype class per variant

Description

How many subjects sit in the least populated genotype class at each variant. This is what says whether a fit is worth reading: an association driven by two homozygotes is not the same claim as one driven by forty, and the scan's extreme tail is anti-conservative precisely where this number is small.

Usage

```
qtlSmallestGenotypeClass(dosage)
```

Arguments

dosage	Numeric matrix of genotype dosages, variants in rows, subjects in columns. Dosages are rounded to 0, 1 or 2.
--------	--

Details

Classes with no subject are not counted, so a variant seen only as 0 and 1 reports the smaller of those two rather than zero. Computed for the whole matrix at once; index the result by variant name rather than calling this per variant.

Value

A named numeric vector, one entry per row of dosage, NA where no class had a subject.

Examples

```
dosage <- rbind(all_three = c(0, 0, 0, 1, 1, 2),
               two_only   = c(0, 0, 0, 0, 1, 1))
colnames(dosage) <- sprintf("s%d", 1:6)

# The empty class is not counted as zero: two_only reports 2, not 0.
qtlSmallestGenotypeClass(dosage)
```

recentAlleleClusters *Retrieving allele similarity clusters Zenodo archive*

Description

A wrapper function for zenodoArchive, download the most recent allele similarity clusters and thresholds from the zenodo archive. The clusters and thresholds are based on https://yaarilab.github.io/IGHV_reference_book/ At the moment only available for human IGHV reference set.

Usage

```
recentAlleleClusters(
  doi = "10.5281/zenodo.7401189",
  path,
  get_file = FALSE,
  quiet = FALSE
)
```

Arguments

doi	The doi for the archive to download. Default is the IGHV set.
path	The output folder for saving the archive files. Default is to a temporary directory.
get_file	Logical (FALSE by default). Do you want to return the path for the file downloaded.
quiet	Logical (FALSE by default). Do you want to suppress informative messages

Value

If get_file is TRUE, the function returns the path to the archive file

Examples

```
## Not run:
recentAlleleClusters(doi="10.5281/zenodo.7401189")

## End(Not run)
```

regroupASCByLabel	<i>Regroup Allele Similarity Clusters by a per-allele label</i>
-------------------	---

Description

Post-processes an ASC assignment (e.g. the alleleClusterTable produced by [inferAlleleClusters](#)) by re-grouping alleles according to a per-allele label – typically the IMGT gene held in imgt_allele. It corrects the two ways an ASC grouping and a gene-level view disagree: an ASC group that merges several distinct genes, and a gene whose alleles are split across ASC groups. The clustering itself is not re-run; only the output is relabelled, so this drops onto the end of the pipeline without changing anything upstream.

Usage

```
regroupASCByLabel(
  x,
  group_col = "allele_cluster",
  label_col = "imgt_allele",
```

```

    duplicated_col = "duplicated",
    action = c("both", "split", "merge")
)

```

Arguments

x	A data.frame / AlleleClusterTable carrying at least the ASC group column and a label column.
group_col	Name of the ASC group column. Default "allele_cluster". An ASC-allele column such as "asc_allele" works too: an allele suffix (*NN) is stripped, so IGHD1-C05H*01 collapses to the ASC gene IGHD1-C05H.
label_col	Name of the label column (e.g. "imgt_allele" or "iuis_allele"). A single value names one gene; a value listing several genes separated by , or / (a shared / ambiguous allele) names them all and will force those genes together.
duplicated_col	Optional column naming <i>other</i> genes that share the same allele, separated by , or /. Default "duplicated"; ignored if the column is absent or is a logical flag (which carries no gene names). Not needed when the shared genes are already listed in label_col.
action	One of: "both" (default) re-group so each label becomes one bin; a shared allele (a label present under two genes) forces those genes together. "split" only break ASC groups that mix labels; a label split across two ASC groups stays split. "merge" only merge ASC groups that share a label or a shared allele; groups that mix labels are left intact.

Details

The label is not produced by the clustering; supply it on the table (the IMGT gene in `imgt_allele`, and any other genes that share the allele in `duplicated`). A gene is taken as the part of a label before *, so allele suffixes and mutation tags (IGHD1-20*01_a7g) are ignored.

Genes are grouped within a locus. A gene name is unique only inside its locus, so a label stripped of its prefix ("J1" rather than "IGHJ1") would otherwise put IGHJ1, IGKJ1 and IGLJ1 in one bin. The locus is read from the label where it still carries one and from `group_col` otherwise; if neither supplies it, the grouping falls back to plain gene names and warns. `action = "split"` is unaffected, since its output is prefixed by the ASC group.

Value

x with an added character column regrouped. A group made of several genes (a merge, e.g. a shared allele) is tagged with all of them joined by /, e.g. "IGHD2-2/IGHD2-8", so no gene name is dropped.

See Also

[inferAlleleClusters](#)

Examples

```
tb <- data.frame(
  allele      = paste0("a", 1:5),
  allele_cluster = c("G1", "G1", "G2", "G3", "G3"),
  imgt_allele  = c("IGHD1-1*01", "IGHD1-2*01", "IGHD1-1*02",
                  "IGHD2-2*01", "IGHD2-8*01"),
  duplicated   = c("", "", "", "IGHD2-8*01", ""), # a4 shared: 2-2 and 2-8
  stringsAsFactors = FALSE)
regroupASCBByLabel(tb, action = "both")$regrouped
```

runGeneUsageQTL

Gene-usage QTL scan for one locus

Description

Runs the whole usage analysis in memory: builds the phenotype, scans every variant against every gene group, sets the significance threshold from the number of independent variants, and clumps the significant hits into leads.

Usage

```
runGeneUsageQTL(
  data,
  dosage,
  variants,
  segments,
  positions = NULL,
  locus = NA_character_,
  subject = "subject",
  min_subject_fraction = 0.1,
  pseudocount = 0.5,
  min_subjects = 60L,
  alpha = 0.05,
  r2_clump = 0.8,
  min_genotype_group = 5L,
  cis_window = 50000
)
```

Arguments

data	A data.frame of rearrangements, one row per sequence.
dosage	Numeric matrix of genotype dosages, variants in rows, subjects in columns. Subjects are matched by name against the phenotype; filter on MAF and missingness before calling.

variants	A data.frame describing the rows of dosage: variant, contig, pos, maf. Not derived from the variant names, which are a convention of one genotype matrix rather than a fact.
segments	Named character vector of the columns holding each segment's gene call, e.g. c(V = "v_gene_iuis", D = "d_gene_iuis", J = "j_gene_iuis"). Names become the segment column.
positions	Optional gene coordinates for the locus, with columns segment, short and mid, used for asc_position, asc_span, n_member, distance_to_asc, is_cis and nearest_gene. Those columns are omitted when it is NULL.
locus	Locus label written onto the output. Optional.
subject	Column naming the subject. Default "subject".
min_subject_fraction	Minimum fraction of subjects a group must appear in to be retained. Default 0.1.
pseudocount	Added to the count, and pseudocount * n_asc to the total. Default 0.5.
min_subjects	Minimum complete-case subjects for a missingness group to be scanned. Default 60.
alpha	Family-wise significance before the independent-variant correction. Default 0.05.
r2_clump	Squared correlation above which a variant is absorbed into a lead. Default 0.8.
min_genotype_group	Smallest genotype class at or above which a lead is called well powered. Default 5.
cis_window	Distance within which a lead is called cis. Default 50000.

Details

The threshold is α over the number of *independent* variants after exact-LD collapse ([qt1LDGroups](#)), not over the number tested. Those differ by about half in this data, and using the number tested is the more conservative but wrong denominator.

Power columns are attached to the leads, where `igqt1.R` put them, and the counts behind every step are messaged: a filter that drops variants without saying so is a bug even when it returns cleanly.

Value

A list of data.tables: phenotype, associations, leads, per_asc and thresholds.

Ceiling

The scan is unweighted OLS on usage fractions, so its standard errors assume constant variance while binomial sampling variance across subjects spans roughly twentyfold. Any R-squared read off it is diluted and is an underestimate. Do not report an effect size from this as exact.

See Also

[qt1ScanUnivariate](#), [runPairingQTL](#)

Examples

```

set.seed(1)
n <- 70L
subjects <- sprintf("s%02d", seq_len(n))
dosage <- rbind(v_hit = rep(c(0, 1, 2), length.out = n),
               v_null = rep(c(0, 0, 1, 2), length.out = n))
colnames(dosage) <- subjects

genes <- c("V1-2", "V3-23", "V4-34")
rep_dt <- do.call(rbind, lapply(seq_len(n), function(i) {
  w <- c(1 + dosage["v_hit", i], 1, 1) # dosage raises V1-2 usage
  data.frame(subject = subjects[i],
             v_gene = sample(genes, 40, replace = TRUE, prob = w / sum(w)))
}))

variants <- data.frame(variant = rownames(dosage), contig = "igh",
                      pos = c(1000L, 50000L), maf = rowMeans(dosage) / 2)

res <- runGeneUsageQTL(rep_dt, dosage, variants,
                     segments = c(V = "v_gene"), min_subjects = 60)
res$associations
res$thresholds

```

runPairingQTL

Conditional gene-pairing QTL scan for one anchoring

Description

Runs one side of the pairing analysis: builds the enrichment table, scans every variant against each anchor's partner distribution, sets the threshold from the number of independent variants, and clumps the hits into leads.

Usage

```

runPairingQTL(
  data,
  dosage,
  variants,
  anchor,
  partner,
  conditional,
  subject = "subject",
  locus = NA_character_,
  pseudocount = 0.5,
  min_subjects = 60L,
  min_complete_fraction = 0.9,
  alpha = 0.05,
  r2_clump = 0.8,

```

```

    min_genotype_group = 5L
)
```

Arguments

<code>data</code>	A <code>data.frame</code> of rearrangements, one row per sequence.
<code>dosage</code>	Numeric matrix of genotype dosages, variants in rows, subjects in columns.
<code>variants</code>	A <code>data.frame</code> describing the rows of dosage: <code>variant</code> , <code>contig</code> , <code>pos</code> , <code>maf</code> . Not derived from the variant names, which are a convention of one genotype matrix rather than a fact.
<code>anchor</code> , <code>partner</code>	Columns holding the gene to condition on and the partner gene.
<code>conditional</code>	A string naming what this scan estimates, e.g. "P(J D)". Required.
<code>subject</code>	Column naming the subject. Default "subject".
<code>locus</code>	Locus label written onto the output. Optional.
<code>pseudocount</code>	Added to the count, and <code>pseudocount * n_asc</code> to the total. Default 0.5.
<code>min_subjects</code>	Minimum complete-case subjects for an anchor to be scanned, and for a missingness group inside it. Default 60.
<code>min_complete_fraction</code>	Minimum fraction of subjects in which a partner gene must be observed for it to stay in the response. Default 0.9.
<code>alpha</code>	Family-wise significance before the independent-variant correction. Default 0.05.
<code>r2_clump</code>	Squared correlation above which a variant is absorbed into a lead. Default 0.8.
<code>min_genotype_group</code>	Smallest genotype class at or above which a fit is called well powered. Default 5.

Details

One anchoring per call, on purpose. Anchoring on each of the two segments gives two different multivariate tests over the same variants, and a function that ran both and returned them stacked would invite exactly the pooling that double-reports. Call it twice, with the `conditional` each side estimates.

Value

A list of `data.tables`: `pairs`, `associations`, `leads` and `thresholds`.

See Also

[pairingTable](#), [pairingScan](#), [pairingCellTests](#)

Examples

```

set.seed(1)
subjects <- sprintf("s%02d", 1:70)
rep_dt <- data.frame(
  subject = rep(subjects, each = 20),
  d_gene = sample(c("D1", "D2", "D3"), 1400, replace = TRUE),
  j_gene = sample(c("J1", "J2"), 1400, replace = TRUE))

dosage <- matrix(rep(c(0, 1, 2), length.out = 70), nrow = 1,
  dimnames = list("v1", subjects))
variants <- data.frame(variant = "v1", contig = "igh", pos = 1000L, maf = 0.33)

# One anchoring per call. Running both and stacking them would double-report,
# so the caller states which conditional each side estimates.
jd <- runPairingQTL(rep_dt, dosage, variants,
  anchor = "j_gene", partner = "d_gene",
  conditional = "P(J|D)", min_subjects = 60)
jd$associations

```

summary.GermlineCluster

Summary method for GermlineCluster

Description

Summary method for GermlineCluster

Usage

```

## S3 method for class 'GermlineCluster'
summary(object, ...)

```

Arguments

object	A GermlineCluster object
...	Additional arguments (ignored)

Value

A list with summary statistics

zenodoArchive

zenodoArchive

Description

zenodoArchive

zenodoArchive

Format

R6Class object.

Value

Object of R6Class for modelling an zenodoArchive for ASC cluster files

Public fields

doi zenodoArchive doi, NULL is not supplied

all_versions zenodoArchive if to return all versions, true when not specified

sort zenodoArchive how to sort the records, mostrecent when not specified

page zenodoArchive which page to pull in query, 1 when not specified

size zenodoArchive how many records per page, 20 when not specified

zenodoVersions zenodoArchive doi available version, a storing variable.

zenodoQuery zenodoArchive doi version query, a storing variable.

download_file zenodoArchive doi downloads files, a storing variable.

download_url zenodoArchive doi downloads urls, a storing variable.

Methods

Public methods:

- [zenodoArchive\\$new\(\)](#)
- [zenodoArchive\\$clean_doi\(\)](#)
- [zenodoArchive\\$zenodo_query\(\)](#)
- [zenodoArchive\\$get_versions\(\)](#)
- [zenodoArchive\\$get_version_files\(\)](#)
- [zenodoArchive\\$download_zenodo_files\(\)](#)
- [zenodoArchive\\$clone\(\)](#)

Method `new()`: initializes the zenodoArchive

Usage:

```
zenodoArchive$new(
  doi,
  page = 1,
  size = 20,
  all_versions = "true",
  sort = "mostrecent"
)
```

Arguments:

doi A zenodo doi. To retrieve all records supply a concept doi (a generic doi common to all versions).

page Which page to query. Default is 1

size How many records per page. Default is 20

all_versions If to return all concept doi versions. If true returns all, if false returns the latest. Default is true

sort Which sorting to apply on the records. Default is mostrecent. Possible sortings "best-match", "mostrecent", "-mostrecent" (ascending), "version", "-version" (ascending).

Method `clean_doi()`: cleans the doi record for query

Usage:

```
zenodoArchive$clean_doi(doi = self$doi)
```

Arguments:

doi The zenodo archive doi

Returns: the clean doi

Method `zenodo_query()`: Query the zenodo archive according to the initial parameters.

Usage:

```
zenodoArchive$zenodo_query(...)
```

Arguments:

... Expects the self created by initialize

Returns: a list with the query values.

Method `get_versions()`: Extract all concept doi available versions.

Usage:

```
zenodoArchive$get_versions(...)
```

Arguments:

... Expects the self created by initialize

Returns: a data.frame of the available versions.

Method `get_version_files()`: get the chosen doi archive version available files

Usage:

```
zenodoArchive$get_version_files(version = "latest")
```

Arguments:

version which archive version files to get. Default to latest. To see all available version use `get_versions`

Returns: a list of the available files in the archive version.

Method `download_zenodo_files()`: get the chosen doi archive version available files

Usage:

```
zenodoArchive$download_zenodo_files(
  file = NULL,
  path = tempdir(),
  version = "latest",
  all_files = F,
  get_file_path = F,
  quiet = F
)
```

Arguments:

`file` If supplied, downloads the specific file from the archive.

`path` The output folder for saving the archive files. Default is to a temporary directory.

`version` which archive version files to get. Default to latest. To see all available version use `get_versions`

`all_files` Logical (FALSE by default). Do you want to download all files in the archive.

`get_file_path` Logical (FALSE by default). Do you want to return the path for the file downloaded.

`quiet` Logical (FALSE by default). Do you want to suppress informative messages

Returns: If `get_file_path` is TRUE, the function returns the path to the archive file

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
zenodoArchive$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
## Not run:
zenodo_archive <- zenodoArchive$new(
  doi = "10.5281/zenodo.7401189"
)

# view available version ins the archive
archive_versions <- zenodo_archive$get_versions()

# Getting the available files in the latest zenodo archive version
files <- zenodo_archive$get_version_files()

# downloading the first file from the latest archive version
zenodo_archive$download_zenodo_files()

## End(Not run)
```

Index

- * **GermlineCluster**
 - new_germline_cluster, 38
- * **asc**
 - allele_cluster_table, 4
 - allele_threshold_table, 10
- * **data**
 - hv_functionality, 24
 - HVGERM, 25
- * **table**
 - allele_cluster_table, 4
 - allele_threshold_table, 10

allele_cluster_table, 4

allele_diff, 4

allele_diff_indices, 5, 8

allele_diff_indices_parallel, 6

allele_diff_indices_parallel2, 7

allele_diff_paired, 6, 7, 8, 8

allele_diff_strings, 9

allele_threshold_table, 10

alleleClusterNames, 10

allelePalette, 11

annotateRepertoireIUIS, 12, 15

artificialFRW1Germline, 13, 50

ascIUISVocabulary, 12, 13, 14

ascUsagePhenotype, 16

assignAlleleClusters, 17, 50

detect_communities_leiden, 18, 19, 40, 53

distance_to_graph, 18, 19

extractASCTable, 20, 50

generateReferenceSet, 20, 49

genotypeToTigger, 22, 53, 54

germlineASC, 23, 50

GermlineCluster, 31, 39

igClust, 19, 25, 28, 29, 31, 40, 55

igDistance, 26, 27, 28, 29, 31

ighvClust, 20, 21, 28, 49

ighvDistance, 21, 28, 29, 49

inferAlleleClusters, 26, 29, 34, 36, 38, 39, 49, 51, 53, 55, 56, 63, 64

inferGenotypeAllele, 22, 23, 32, 50, 53, 54

inferGenotypeAllele_asc, 22, 23, 35, 53, 54

insert_gaps, 37, 38

insert_gaps2_vec, 38

new_germline_cluster, 38

optimize_resolution, 18, 40, 54, 55

pairingCellTests, 41, 44, 47, 49, 68

pairingLeadCharacter, 42

pairingPhenotype, 45

pairingScan, 41, 42, 45, 46, 49, 60, 68

pairingTable, 41, 43, 45–47, 48, 68

piglet, 49

plot.GermlineCluster, 31, 50, 56

plotAlleleCluster, 50, 51

plotClusterComparison, 51

plotCommunityNetwork, 52

plotGenotypeAllele, 53

plotSilhouetteOptimization, 54

plotTruncatedTree, 55

print.GermlineCluster, 57

qtlClump, 57, 58

qtlLDGroups, 47, 58, 66

qtlScanMultivariate, 46, 59, 61

qtlScanUnivariate, 60, 66

qtlSmallestGenotypeClass, 61, 61

recentAlleleClusters, 34, 36, 50, 62

regroupASCByLabel, 15, 63

runGeneUsageQTL, 16, 61, 65

runPairingQTL, 43, 44, 66, 67

summary.GermlineCluster, [69](#)

zenodoArchive, [50](#), [70](#)