Package 'tigger'

October 6, 2026

Type Package

Version 1.2.0

Date 2026-10-06

Title Infers Novel Immunoglobulin Alleles from Sequencing Data

Description Infers the V genotype of an individual from immunoglobulin (Ig) repertoire sequencing data (AIRR-Seq, Rep-Seq). Includes detection of any novel alleles. This information is then used to correct existing V allele calls from among the sample sequences.
Citations:
Gadala-Maria, et al (2015) <doi:10.1073/pnas.1417683112>,
Gadala-Maria, et al (2019) <doi:10.3389/fimmu.2019.00129>.

License AGPL-3

URL https://tigger.readthedocs.io

BugReports https://github.com/immcantation/tigger/issues

LazyData true

BuildVignettes true

VignetteBuilder knitr

Encoding UTF-8

Depends R (>= 4.0), ggplot2 (>= 3.4.0)

Imports alakazam (>= 1.5.0), dplyr (>= 1.0.0), doParallel, foreach, graphics, gridExtra, gtools, iterators, lazyeval, parallel, rlang, stats, stringi, tidyr (>= 1.1.0), utils

Suggests knitr, rmarkdown, testthat

Config/roxygen2/version 8.1.0

NeedsCompilation no

Author Daniel Gadala-Maria [aut],
Susanna Marquez [aut, cre],
Moriah Cohen [aut],
Ayelet Peres [aut],
Jason Vander Heiden [aut],

Gur Yaari [aut],
Steven Kleinstein [aut, cph]

Maintainer Susanna Marquez <susanna.marquez@yale.edu>

Repository CRAN

Date/Publication 2026-10-06 20:20:02 UTC

Contents

AIRRDb . . . 3
cleanSeqs . . . 3
findNovelAlleles . . . 4
findUnmutatedCalls . . . 8
generateEvidence . . . 9
genotypeFasta . . . 11
GermlineIGHV . . . 12
getMutatedPositions . . . 13
getMutCount . . . 14
getPopularMutationCount . . . 15
inferGenotype . . . 16
inferGenotypeBayesian . . . 17
insertPolymorphisms . . . 20
plotGenotype . . . 20
plotGenotypeConfidence . . . 21
plotNovel . . . 23
readIgFasta . . . 24
reassignAlleles . . . 25
SampleDb . . . 27
SampleGenotype . . . 28
SampleGermlineIGHV . . . 28
SampleNovel . . . 29
selectNovel . . . 29
sortAlleles . . . 30
subsampleDb . . . 31
tigger . . . 32
updateAlleleNames . . . 33
writeFasta . . . 34

Index **36**

AIRRDb — *Example human immune repertoire data*

Description

A `data.frame` of example V(D)J immunoglobulin sequences derived from a single individual (PGP1), sequenced on the Roche 454 platform, and assigned by IMGT/HighV-QUEST to IGHV1 family alleles.

Format

A `data.frame` where rows correspond to unique V(D)J sequences and columns include:

- `"sequence_alignment"`: IMGT-gapped V(D)J nucleotide sequence.
- `"v_call"`: IMGT/HighV-QUEST V segment allele calls.
- `"d_call"`: IMGT/HighV-QUEST D segment allele calls.
- `"j_call"`: IMGT/HighV-QUEST J segment allele calls.
- `"junction_length"`: Junction region length.

References

1. Gadala-Maria, et al. (2015) Automated analysis of high-throughput B cell sequencing data reveals a high frequency of novel immunoglobulin V gene segment alleles. PNAS. 112(8):E862-70.

See Also

See SampleDb for Change-O formatted version of `AIRRDb`.

cleanSeqs — *Clean up nucleotide sequences*

Description

`cleanSeqs` capitalizes nucleotides and replaces all characters besides `c("A", "C", "G", "T", "-", ".")` with `"N"`.

Usage

```
cleanSeqs(seqs)
```

Arguments

`seqs`	vector of nucleotide sequences.

Value

A modified vector of nucleotide sequences.

See Also

sortAlleles and updateAlleleNames can help format a list of allele names.

Examples

```
# Clean messy nucleotide sequences
seqs <- c("AGAT.taa-GAG...ATA", "GATACAGTXXZZAGNNPPACA")
cleanSeqs(seqs)
```

`findNovelAlleles` *Find novel alleles from repertoire sequencing data*

Description

`findNovelAlleles` analyzes mutation patterns in sequences thought to align to each germline allele in order to determine which positions might be polymorphic.

Usage

```
findNovelAlleles(
  data,
  germline_db,
  v_call = "v_call",
  j_call = "j_call",
  seq = "sequence_alignment",
  junction = "junction",
  junction_length = "junction_length",
  germline_min = 200,
  min_seqs = 50,
  auto_mutrange = TRUE,
  mut_range = 1:10,
  pos_range = 1:312,
  pos_range_max = NULL,
  y_intercept = 0.125,
  alpha = 0.05,
  j_max = 0.15,
  min_frac = 0.75,
  nproc = 1
)
```

Arguments

data	data.frame containing repertoire data. See details.
germline_db	vector of named nucleotide germline sequences matching the V calls in data. These should be the gapped reference germlines used to make the V calls.
v_call	name of the column in data with V allele calls. Default is v_call.
j_call	name of the column in data with J allele calls. Default is j_call.
seq	name of the column in data with the aligned, IMGT-numbered, V(D)J nucleotide sequence. Default is sequence_alignment.
junction	Junction region nucleotide sequence, which includes the CDR3 and the two flanking conserved codons. Default is junction.
junction_length	Number of junction nucleotides in the junction sequence. Default is junction_length.
germline_min	the minimum number of sequences that must have a particular germline allele call for the allele to be analyzed.
min_seqs	minimum number of total sequences (within the desired mutational range and nucleotide range) required for the samples to be considered.
auto_mutrange	if TRUE, the algorithm will attempt to determine the appropriate mutation range automatically using the mutation count of the most common sequence assigned to each allele analyzed.
mut_range	range of mutations that samples may carry and be considered by the algorithm.
pos_range	range of IMGT-numbered positions that should be considered by the algorithm.
pos_range_max	Name of the column in data with the ending positions of the V alignment in the germline (usually v_germline_end). The end of the alignment will be used to limit the range of positions to be considered to count mutations. With NULL all positions in the IMGT V region will be considered. In this case, in sequences where the V was trimmed on the 3', mutated nucleotides could include nucleotides from the CDR3.
y_intercept	y-intercept threshold above which positions should be considered potentially polymorphic.
alpha	alpha value used for determining whether the fit y-intercept is greater than the y_intercept threshold.
j_max	maximum fraction of sequences perfectly aligning to a potential novel allele that are allowed to utilize to a particular combination of junction length and J gene. The closer to 1, the less strict the filter for junction length and J gene diversity will be.
min_frac	minimum fraction of sequences that must have usable nucleotides in a given position for that position to considered.
nproc	number of processors to use.

Details

The TIgGER allele-finding algorithm, briefly, works as follows: Mutations are determined through comparison to the provided germline. Mutation frequency at each *position* is determined as

a function of *sequence-wide* mutation counts. Polymorphic positions exhibit a high mutation frequency despite sequence-wide mutation count. False positive of potential novel alleles resulting from clonally-related sequences are guarded against by ensuring that sequences perfectly matching the potential novel allele utilize a wide range of combinations of J gene and junction length.

Value

A `data.frame` with a row for each known allele analyzed. Besides metadata on the the parameters used in the search, each row will have either a note as to where the polymorphism-finding algorithm exited or a nucleotide sequence for the predicted novel allele, along with columns providing additional evidence.

The output contains the following columns:

- `germline_call`: The input (uncorrected) V call.
- `note`: Comments regarding the inference.
- `polymorphism_call`: The novel allele call.
- `nt_substitutions`: Mutations identified in the novel allele, relative to the reference germline (`germline_call`)
- `novel_imgt`: The novel allele sequence.
- `novel_imgt_count`: The number of times the sequence `novel_imgt` is found in the input data. Considers the subsequence of `novel_imgt` in the `pos_range`.
- `novel_imgt_unique_j`: Number of distinct J calls associated to `novel_imgt` in the input data. Considers the subsequence of `novel_imgt` in the `pos_range`.
- `novel_imgt_unique_cdr3`: Number of distinct CDR3 sequences associated with `novel_imgt` in the input data. Considers the subsequence of `novel_imgt` in the `pos_range`.
- `perfect_match_count`: Final number of sequences retained to call the new allele. These are unique sequences that have V segments that perfectly match the predicted germline in the `pos_range`.
- `perfect_match_freq`: `perfect_match_count / germline_call_count`
- `germline_call_count`: The number of sequences with the `germline_call` in the input data that were initially considered for the analysis.
- `germline_call_freq`: The fraction of sequences with the `germline_call` in the input data initially considered for the analysis.
- `germline_imgt`: Germline sequence for `germline_call`.
- `germline_imgt_count`: The number of times the `germline_imgt` sequence is found in the input data.
- `mut_min`: Minimum mutation considered by the algorithm.
- `mut_max`: Maximum mutation considered by the algorithm.
- `mut_pass_count`: Number of sequences in the mutation range.
- `pos_min`: First position of the sequence considered by the algorithm (IMGT numbering).
- `pos_max`: Last position of the sequence considered by the algorithm (IMGT numbering).
- `y_intercept`: The y-intercept above which positions were considered potentially polymorphic.

- y_intercept_pass: Number of positions that pass the y_intercept threshold.
- snp_pass: Number of sequences that pass the y_intercept threshold and are within the desired nucleotide range (min_seqs).
- unmutated_count: Number of unmutated sequences.
- unmutated_freq: Number of unmutated sequences over germline_imgt_count.
- unmutated_snp_j_gene_length_count: Number of distinct combinations of SNP, J gene, and junction length.
- snp_min_seqs_j_max_pass: Number of SNPs that pass both the min_seqs and j_max thresholds.
- alpha: Significance threshold to be used when constructing the confidence interval for the y-intercept.
- min_seqs: Input min_seqs. The minimum number of total sequences (within the desired mutational range and nucleotide range) required for the samples to be considered.
- j_max: Input j_max. The maximum fraction of sequences perfectly aligning to a potential novel allele that are allowed to utilize to a particular combination of junction length and J gene.
- min_frac: Input min_frac. The minimum fraction of sequences that must have usable nucleotides in a given position for that position to be considered.

The following comments can appear in the note column:

- *Novel allele found*: A novel allele was detected.
- *Same as:*: The same novel allele sequence has been identified multiple times. If this happens, the function will also throw the message 'Duplicated polymorphism(s) found'.
- *Plurality sequence too rare*: No sequence is frequent enough to pass the J test (j_max).
- *A J-junction combination is too prevalent*: Not enough J diversity (j_max).
- *No positions pass y-intercept test*: No positions above y_intercept.
- *Insufficient sequences in desired mutational range*: mut_range and pos_range.
- *Not enough sequences*: Not enough sequences in the desired mutational range and nucleotide range (min_seqs).
- *No unmutated versions of novel allele found*: All observed variants of the allele are mutated.

See Also

selectNovel to filter the results to show only novel alleles. plotNovel to visualize the data supporting any novel alleles hypothesized to be present in the data and inferGenotype and inferGenotypeBayesian to determine if the novel alleles are frequent enough to be included in the subject's genotype.

Examples

```
# Note: In this example, with SampleGermlineIGHV,
# which contains reference germlines retrieved on August 2014,
# TIgGER finds the allele IGHV1-8*02_G234T. This allele
# was added to IMGT as IGHV1-8*03 on March 28, 2018.
```

```
# Find novel alleles and return relevant data
novel <- findNovelAlleles(AIRRDb, SampleGermlineIGHV)
selectNovel(novel)
```

`findUnmutatedCalls` *Determine which calls represent an unmutated allele*

Description

`findUnmutatedCalls` determines which allele calls would represent a perfect match with the germline sequence, given a vector of allele calls and mutation counts. In the case of multiple alleles being assigned to a sequence, only the subset that would represent a perfect match is returned.

Usage

```
findUnmutatedCalls(allele_calls, sample_seqs, germline_db)
```

Arguments

`allele_calls`	vector of strings representing Ig allele calls, where multiple calls are separated by a comma.
`sample_seqs`	V(D)J-rearranged sample sequences matching the order of the given `allele_calls`.
`germline_db`	vector of named nucleotide germline sequences

Value

A vector of strings containing the members of `allele_calls` that represent unmutated sequences.

Examples

```
# Find which of the sample alleles are unmutated
calls <- findUnmutatedCalls(AIRRDb$v_call, AIRRDb$sequence_alignment,
                            germline_db=SampleGermlineIGHV)
```

generateEvidence *Generate evidence*

Description

generateEvidence builds a table of evidence metrics for the final novel V allele detection and genotyping inferences.

Usage

```
generateEvidence(
  data,
  novel,
  genotype,
  genotype_db,
  germline_db,
  j_call = "j_call",
  junction = "junction",
  fields = NULL
)
```

Arguments

data	a data.frame containing sequence data that has been passed through reassignAlleles to correct the allele assignments.
novel	the data.frame returned by findNovelAlleles.
genotype	the data.frame of alleles generated with inferGenotype denoting the genotype of the subject.
genotype_db	a vector of named nucleotide germline sequences in the genotype. Returned by genotypeFasta.
germline_db	the original uncorrected germline database used to by findNovelAlleles to identify novel alleles.
j_call	name of the column in data with J allele calls. Default is j_call.
junction	Junction region nucleotide sequence, which includes the CDR3 and the two flanking conserved codons. Default is junction.
fields	character vector of column names used to split the data to identify novel alleles, if any. If NULL then the data is not divided by grouping variables.

Value

Returns the genotype input data.frame with the following additional columns providing supporting evidence for each inferred allele:

- field_id: Data subset identifier, defined with the input parameter fields.
- A variable number of columns, specified with the input parameter fields.

- polymorphism_call: The novel allele call.
- novel_imgt: The novel allele sequence.
- closest_reference: The closest reference gene and allele in the germline_db database.
- closest_reference_imgt: Sequence of the closest reference gene and allele in the germline_db database.
- germline_call: The input (uncorrected) V call.
- germline_imgt: Germline sequence for germline_call.
- nt_diff: Number of nucleotides that differ between the new allele and the closest reference (closest_reference) in the germline_db database.
- nt_substitutions: A comma separated list of specific nucleotide differences (e.g. 112G>A) in the novel allele.
- aa_diff: Number of amino acids that differ between the new allele and the closest reference (closest_reference) in the germline_db database.
- aa_substitutions: A comma separated list with specific amino acid differences (e.g. 96A>N) in the novel allele.
- sequences: Number of sequences unambiguously assigned to this allele.
- unmutated_sequences: Number of records with the unmutated novel allele sequence.
- unmutated_frequency: Proportion of records with the unmutated novel allele sequence (unmutated_sequences / sequences).
- allelic_percentage: Percentage at which the (unmutated) allele is observed in the sequence dataset compared to other (unmutated) alleles.
- unique_js: Number of unique J sequences found associated with the novel allele. The sequences are those who have been unambiguously assigned to the novel allele (polymorphism_call).
- unique_cdr3s: Number of unique CDR3s associated with the inferred allele. The sequences are those who have been unambiguously assigned to the novel allele (polymorphism_call).
- mut_min: Minimum mutation considered by the algorithm.
- mut_max: Maximum mutation considered by the algorithm.
- pos_min: First position of the sequence considered by the algorithm (IMGT numbering).
- pos_max: Last position of the sequence considered by the algorithm (IMGT numbering).
- y_intercept: The y-intercept above which positions were considered potentially polymorphic.
- alpha: Significance threshold to be used when constructing the confidence interval for the y-intercept.
- min_seqs: Input min_seqs. The minimum number of total sequences (within the desired mutational range and nucleotide range) required for the samples to be considered.
- j_max: Input j_max. The maximum fraction of sequences perfectly aligning to a potential novel allele that are allowed to utilize to a particular combination of junction length and J gene.
- min_frac: Input min_frac. The minimum fraction of sequences that must have usable nucleotides in a given position for that position to be considered.
- note: Comments regarding the novel allele inference.

See Also

See findNovelAlleles, inferGenotype and genotypeFasta for generating the required input.

Examples

```
# Generate input data
novel <- findNovelAlleles(AIRRDb, SampleGermlineIGHV,
    v_call="v_call", j_call="j_call", junction="junction",
    junction_length="junction_length", seq="sequence_alignment")
genotype <- inferGenotype(AIRRDb, find_unmutated=TRUE,
                          germline_db=SampleGermlineIGHV,
                          novel=novel,
                          v_call="v_call", seq="sequence_alignment")
genotype_db <- genotypeFasta(genotype, SampleGermlineIGHV, novel)
data_db <- reassignAlleles(AIRRDb, genotype_db,
v_call="v_call", seq="sequence_alignment")

# Assemble evidence table
evidence <- generateEvidence(data_db, novel, genotype,
                             genotype_db, SampleGermlineIGHV,
                             j_call = "j_call",
                             junction = "junction")
```

---

genotypeFasta *Return the nucleotide sequences of a genotype*

Description

genotypeFasta converts a genotype table into a vector of nucleotide sequences.

Usage

```
genotypeFasta(
  genotype,
  germline_db,
  novel = NA,
  include_unseen = FALSE,
  strip_d = TRUE
)
```

Arguments

genotype	data.frame of alleles denoting a genotype, as returned by inferGenotype.
germline_db	vector of named nucleotide germline sequences matching the alleles detailed in genotype.

novel	an optional data.frame containing putative novel alleles of the type returned by findNovelAlleles.
include_unseen	if TRUE, include germline database alleles for genes that are not present in genotype. For genes present in genotype, include only the genotyped alleles.
strip_d	if TRUE (default) remove the "D" from the end of gene annotations (denoting a duplicate gene in the locus) when matching genotype alleles to germline_db. If FALSE, alleles are matched exactly.

Value

A named vector of strings containing the germline nucleotide sequences of the alleles in the provided genotype.

See Also

inferGenotype

Examples

```
# Find the sequences that correspond to the genotype
genotype_db <- genotypeFasta(SampleGenotype, SampleGermlineIGHV, SampleNovel)
```

GermlineIGHV *Human IGHV germlines*

Description

A character vector of all human IGHV germline gene segment alleles in IMGT/GENE-DB (2019-06-01, 372 alleles). See IMGT data updates: https://www.imgt.org/IMGTgenedbdoc/dataupdates.html.

Format

Values correspond to IMGT-gaped nucleotide sequences (with nucleotides capitalized and gaps represented by ".") while names correspond to stripped-down IMGT allele names (e.g. "IGHV1-18*01").

References

1. 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.

`getMutatedPositions` *Find the location of mutations in a sequence*

Description

`getMutatedPositions` takes two vectors of aligned sequences and compares pairs of sequences. It returns a list of the nucleotide positions of any differences.

Usage

```
getMutatedPositions(
  samples,
  germlines,
  ignored_regex = "[\\.N-]",
  match_instead = FALSE
)
```

Arguments

`samples`	vector of strings representing aligned sequences
`germlines`	vector of strings representing aligned sequences to which `samples` will be compared. If only one string is submitted, it will be used for all `samples`.
`ignored_regex`	regular expression indicating what characters should be ignored (such as gaps and N nucleotides).
`match_instead`	if `TRUE`, the function returns the positions that are the same instead of those that are different.

Value

A list of the nucleotide positions of any differences between the input vectors.

Examples

```
# Create strings to act as a sample sequences and a reference sequence
seqs <- c("----GATA", "GAGAGAGA", "TANA")
ref <- "GATAGATA"

# Find the differences between the two
getMutatedPositions(seqs, ref)
```

getMutCount *Determine the mutation counts from allele calls*

Description

getMutCount takes a set of nucleotide sequences and their allele calls and determines the distance between that sequence and any germline alleles contained within the call

Usage

```
getMutCount(samples, allele_calls, germline_db)
```

Arguments

samples	vector of IMGT-gapped sample V sequences
allele_calls	vector of strings representing Ig allele calls for the sequences in samples, where multiple calls are separated by a comma
germline_db	vector of named nucleotide germline sequences matching the calls detailed in allele_calls

Value

A list equal in length to samples, containing the Hamming distance to each germline allele contained within each call within each element of samples

Examples

```
# Insert a mutation into a germline sequence
s2 <- s3 <- SampleGermlineIGHV[1]
stringi::stri_sub(s2, 103, 103) <- "G"
stringi::stri_sub(s3, 107, 107) <- "C"

sample_seqs <- c(SampleGermlineIGHV[2], s2, s3)

# Pretend that one sample sequence has received an ambiguous allele call
sample_alleles <- c(paste(names(SampleGermlineIGHV[1:2]), collapse=","),
                    names(SampleGermlineIGHV[2]),
                    names(SampleGermlineIGHV[1]))

# Compare each sequence to its assigned germline(s) to determine the distance
getMutCount(sample_seqs, sample_alleles, SampleGermlineIGHV)
```

getPopularMutationCount *Find mutation counts for frequency sequences*

Description

getPopularMutationCount determines which sequences occur frequently for each V gene and returns the mutation count of those sequences.

Usage

```
getPopularMutationCount(
  data,
  germline_db,
  v_call = "v_call",
  seq = "sequence_alignment",
  gene_min = 0.001,
  seq_min = 50,
  seq_p_of_max = 1/8,
  full_return = FALSE
)
```

Arguments

data	data.frame in the Change-O format. See findNovelAlleles for a list of required columns.
germline_db	named list of IMGT-gapped germline sequences.
v_call	name of the column in data with V allele calls. Default is v_call.
seq	name of the column in data with the aligned, IMGT-numbered, V(D)J nucleotide sequence. Default is sequence_alignment.
gene_min	portion of all unique sequences a gene must constitute to avoid exclusion.
seq_min	number of copies of the V that must be present for to avoid exclusion.
seq_p_of_max	ror each gene, the fraction of the most common V sequence count that a sequence must meet to avoid exclusion.
full_return	if TRUE, will return all data columns and will include sequences with mutation count < 1.

Value

A data frame of genes that have a frequent sequence mutation count above 1.

See Also

getMutatedPositions can be used to find which positions of a set of sequences are mutated.

Examples

```
getPopularMutationCount(AIRRDb, SampleGermlineIGHV)
```

`inferGenotype` *Infer a subject-specific genotype using a frequency method*

Description

`inferGenotype` infers an subject's genotype using a frequency method. The genotype is inferred by finding the minimum number set of alleles that can explain the majority of each gene's calls. The most common allele of each gene is included in the genotype first, and the next most common allele is added until the desired fraction of alleles can be explained. In this way, mistaken allele calls (resulting from sequences which by chance have been mutated to look like another allele) can be removed.

Usage

```
inferGenotype(
  data,
  germline_db = NA,
  novel = NA,
  v_call = "v_call",
  seq = "sequence_alignment",
  fraction_to_explain = 0.875,
  gene_cutoff = 1e-04,
  find_unmutated = TRUE
)
```

Arguments

`data`	`data.frame` containing V allele calls from a single subject.
`germline_db`	named vector of sequences containing the germline sequences named in `allele_calls`. Only required if `find_unmutated` is `TRUE`.
`novel`	optional `data.frame` of the type novel returned by findNovelAlleles containing germline sequences that will be utilized if `find_unmutated` is `TRUE`. See Details.
`v_call`	column in `data` with V allele calls. Default is `"v_call"`.
`seq`	name of the column in `data` with the aligned, IMGT-numbered, V(D)J nucleotide sequence. Default is `sequence_alignment`.
`fraction_to_explain`	the portion of each gene that must be explained by the alleles that will be included in the genotype.
`gene_cutoff`	either a number of sequences or a fraction of the length of `allele_calls` denoting the minimum number of times a gene must be observed in `allele_calls` to be included in the genotype.
`find_unmutated`	if `TRUE`, use `germline_db` to find which samples are unmutated. Not needed if `allele_calls` only represent unmutated samples.

Details

Allele calls representing cases where multiple alleles have been assigned to a single sample sequence are rare among unmutated sequences but may result if nucleotides for certain positions are not available. Calls containing multiple alleles are treated as belonging to all groups. If `novel` is provided, all sequences that are assigned to the same starting allele as any novel germline allele will have the novel germline allele appended to their assignment prior to searching for unmutated sequences.

Value

A `data.frame` of alleles denoting the genotype of the subject containing the following columns:

- `gene`: The gene name without allele.
- `alleles`: Comma separated list of alleles for the given `gene`.
- `counts`: Comma separated list of observed sequences for each corresponding allele in the `alleles` list.
- `total`: The total count of observed sequences for the given `gene`.
- `note`: Any comments on the inference.

Note

This method works best with data derived from blood, where a large portion of sequences are expected to be unmutated. Ideally, there should be hundreds of allele calls per gene in the input.

See Also

plotGenotype for a colorful visualization and genotypeFasta to convert the genotype to nucleotide sequences. See inferGenotypeBayesian to infer a subject-specific genotype using a Bayesian approach.

Examples

```
# Infer IGHV genotype, using only unmutated sequences, including novel alleles
inferGenotype(AIRRDb, germline_db=SampleGermlineIGHV, novel=SampleNovel,
              find_unmutated=TRUE)
```

`inferGenotypeBayesian` *Infer a subject-specific genotype using a Bayesian approach*

Description

`inferGenotypeBayesian` infers an subject's genotype by applying a Bayesian framework with a Dirichlet prior for the multinomial distribution. Up to four distinct alleles are allowed in an individual's genotype. Four likelihood distributions were generated by empirically fitting three high coverage genotypes from three individuals (Laserson and Vigneault et al, 2014). A posterior probability is calculated for the four most common alleles. The certainty of the highest probability model was calculated using a Bayes factor (the most likely model divided by second-most likely model). The larger the Bayes factor (K), the greater the certainty in the model.

Usage

```
inferGenotypeBayesian(
  data,
  germline_db = NA,
  novel = NA,
  v_call = "v_call",
  seq = "sequence_alignment",
  find_unmutated = TRUE,
  priors = c(0.6, 0.4, 0.4, 0.35, 0.25, 0.25, 0.25, 0.25, 0.25),
  genotyped_alleles = FALSE
)
```

Arguments

data	a data.frame containing V allele calls from a single subject. If find_unmutated is TRUE, then the sample IMGT-gapped V(D)J sequence should be provided in column sequence_alignment
germline_db	named vector of sequences containing the germline sequences named in allele_calls. Only required if find_unmutated is TRUE.
novel	an optional data.frame of the type novel returned by findNovelAlleles containing germline sequences that will be utilized if find_unmutated is TRUE. See Details.
v_call	column in data with V allele calls. Default is "v_call".
seq	name of the column in data with the aligned, IMGT-numbered, V(D)J nucleotide sequence. Default is "sequence_alignment".
find_unmutated	if TRUE, use germline_db to find which samples are unmutated. Not needed if allele_calls only represent unmutated samples.
priors	a numeric vector of priors for the multinomial distribution. The priors vector must be nine values that defined the priors for the heterozygous (two allele), trizygous (three allele), and quadrozygous (four allele) distributions. The first two values of priors define the prior for the heterozygous case, the next three values are for the trizygous case, and the final four values are for the quadrozygous case. Each set of priors should sum to one. Note, each distribution prior is actually defined internally by set of four numbers, with the unspecified final values assigned to 0; e.g., the heterozygous case is c(priors[1], priors[2], 0, 0). The prior for the homozygous distribution is fixed at c(1, 0, 0, 0).
genotyped_alleles	if TRUE, add a genotyped_alleles column containing the most likely alleles based on the highest Bayesian zygosity likelihood. Default is FALSE.

Details

Allele calls representing cases where multiple alleles have been assigned to a single sample sequence are rare among unmutated sequences but may result if nucleotides for certain positions are not available. Calls containing multiple alleles are treated as belonging to all groups. If novel is provided, all sequences that are assigned to the same starting allele as any novel germline allele

will have the novel germline allele appended to their assignment prior to searching for unmutated sequences.

Value

A `data.frame` of alleles denoting the genotype of the subject with the log10 of the likelihood of each model and the log10 of the Bayes factor. The output contains the following columns:

- `gene`: The gene name without allele.
- `alleles`: Comma separated list of alleles for the given `gene`.
- `counts`: Comma separated list of observed sequences for each corresponding allele in the `alleles` list.
- `total`: The total count of observed sequences for the given `gene`.
- `note`: Any comments on the inference.
- `kh`: log10 likelihood that the `gene` is homozygous.
- `kd`: log10 likelihood that the `gene` is heterozygous.
- `kt`: log10 likelihood that the `gene` is trizygous
- `kq`: log10 likelihood that the `gene` is quadrozygous.
- `k_diff`: log10 ratio of the highest to second-highest zygosity likelihoods.
- `genotyped_alleles`: If requested, comma separated list of alleles in the most likely genotype for the given `gene`.

Note

This method works best with data derived from blood, where a large portion of sequences are expected to be unmutated. Ideally, there should be hundreds of allele calls per gene in the input.

References

1. Laserson U and Vigneault F, et al. High-resolution antibody dynamics of vaccine-induced immune responses. PNAS. 2014 111(13):4928-33.

See Also

plotGenotype for a colorful visualization and genotypeFasta to convert the genotype to nucleotide sequences. See inferGenotype to infer a subject-specific genotype using a frequency method

Examples

```
# Infer IGHV genotype, using only unmutated sequences, including novel alleles
inferGenotypeBayesian(AIRRDb, germline_db=SampleGermlineIGHV, novel=SampleNovel,
                      find_unmutated=TRUE, v_call="v_call", seq="sequence_alignment")
```

insertPolymorphisms *Insert polymorphisms into a nucleotide sequence*

Description

insertPolymorphisms replaces nucleotides in the desired locations of a provided sequence.

Usage

```
insertPolymorphisms(sequence, positions, nucleotides)
```

Arguments

sequence	starting nucleotide sequence.
positions	numeric vector of positions which to be changed.
nucleotides	character vector of nucleotides to which to change the positions.

Value

A sequence with the desired nucleotides in the provided locations.

Examples

```
insertPolymorphisms("HUGGED", c(1, 6, 2), c("T", "R", "I"))
```

plotGenotype *Show a colorful representation of a genotype*

Description

plotGenotype plots a genotype table.

Usage

```
plotGenotype(
  genotype,
  facet_by = NULL,
  gene_sort = c("name", "position"),
  text_size = 12,
  silent = FALSE,
  allele_col = "alleles",
  ...
)
```

Arguments

genotype	data.frame of alleles denoting a genotype, as returned by inferGenotype.
facet_by	column name in genotype to facet the plot by. if NULL, then do not facet the plot.
gene_sort	string defining the method to use when sorting alleles. if "name" then sort in lexicographic order. If "position" then sort by position in the locus, as determined by the final two numbers in the gene name.
text_size	point size of the plotted text.
silent	if TRUE do not draw the plot and just return the ggplot object; if FALSE draw the plot.
allele_col	name of the column in genotype holding the comma-separated alleles to plot. Defaults to "alleles"; set to "genotyped_alleles" to plot the most likely alleles from inferGenotypeBayesian.
...	additional arguments to pass to ggplot2::theme.

Value

A ggplot object defining the plot.

See Also

inferGenotype, plotGenotypeConfidence

Examples

```
# Plot genotype
plotGenotype(SampleGenotype)

# Facet by subject
genotype_a <- genotype_b <- SampleGenotype
genotype_a$SUBJECT <- "A"
genotype_b$SUBJECT <- "B"
geno_sub <- rbind(genotype_a, genotype_b)
plotGenotype(geno_sub, facet_by="SUBJECT", gene_sort="pos")
```

---

plotGenotypeConfidence

Show a genotype with a confidence evidence panel

Description

plotGenotypeConfidence draws a genotype with plotGenotype and adds a color panel beside it showing a per-gene confidence value, such as the k_diff produced by inferGenotypeBayesian.

Usage

```
plotGenotypeConfidence(
  genotype,
  confidence_col,
  allele_col = "alleles",
  gene_sort = c("name", "position"),
  text_size = 12,
  confidence_breaks = c(0, 1, 2, 3, 4, 5, 10, 20, 50, Inf),
  silent = FALSE,
  ...
)
```

Arguments

genotype	data.frame of alleles denoting a genotype, as returned by inferGenotypeBayesian.
confidence_col	name of the column in genotype holding the per-gene confidence value, e.g. "k_diff". The value is binned and shown on a blue color scale, with the column name as the legend title; white marks genes with no value.
allele_col	name of the column in genotype holding the alleles to plot, passed to plotGenotype. Set to "genotyped_alleles" to plot the most likely alleles.
gene_sort	string defining the method to use when sorting alleles, passed to plotGenotype.
text_size	point size of the plotted text.
confidence_breaks	numeric vector of breaks used to bin the confidence value into color groups. The default c(0, 1, 2, 3, 4, 5, 10, 20, 50, Inf) matches the binning used by RAbHIT for the haplotype lK panel.
silent	if TRUE do not draw the plot and just return the grob; if FALSE draw the plot.
...	additional arguments to pass to ggplot2::theme of the genotype panel.

Value

A gridExtra grob combining the genotype plot with the confidence panel.

See Also

plotGenotype, inferGenotypeBayesian

Examples

```
# The Bayesian genotype carries a per-gene confidence (k_diff) and, optionally,
# a genotyped_alleles column
geno_bayesian <- inferGenotypeBayesian(AIRRDb, germline_db=SampleGermlineIGHV,
                                       novel=SampleNovel, genotyped_alleles=TRUE)
plotGenotypeConfidence(geno_bayesian, confidence_col="k_diff",
                       allele_col="genotyped_alleles")
```

plotNovel *Visualize evidence of novel V alleles*

Description

plotNovel is be used to visualize the evidence of any novel V alleles found using findNovelAlleles. It can also be used to visualize the results for alleles that did

Usage

```
plotNovel(
  data,
  novel_row,
  v_call = "v_call",
  j_call = "j_call",
  seq = "sequence_alignment",
  junction = "junction",
  junction_length = "junction_length",
  pos_range_max = NULL,
  ncol = 1,
  multiplot = TRUE
)
```

Arguments

data	data.frame containing repertoire data. See findNovelAlleles for details.
novel_row	single row from a data frame as output by findNovelAlleles that contains a polymorphism-containing germline allele.
v_call	name of the column in data with V allele calls. Default is v_call.
j_call	name of the column in data with J allele calls. Default is j_call.
seq	name of the column in data with the aligned, IMGT-numbered, V(D)J nucleotide sequence. Default is sequence_alignment.
junction	Junction region nucleotide sequence, which includes the CDR3 and the two flanking conserved codons. Default is junction.
junction_length	number of junction nucleotides in the junction sequence. Default is junction_length.
pos_range_max	Name of the column in data with the ending positions of the V alignment in the germline (usually v_germline_end).
ncol	number of columns to use when laying out the plots.
multiplot	whether to return one single plot (TRUE) or a list with the three individual plots (FALSE).

Details

The first panel in the plot shows, for all sequences which align to a particular germline allele, the mutation frequency at each position along the aligned sequence as a function of the sequence-wide mutation count. Each line is a position. Positions that contain polymorphisms (rather than somatic hypermutations) will exhibit a high apparent mutation frequency for a range of sequence-wide mutation counts. The positions are color coded as follows:

- red: the position(s) pass(ess) the novel allele test
- yellow: the position(s) pass(ess) the y-intercept test but not other tests
- blue: the position(s) didn't pass the y-intercept test and was(were) not further considered

The second panel shows the nucleotide usage at each of the polymorphic positions as a function of sequence-wide mutation count. If no polymorphisms were identified, the panel will show the mutation count.

To avoid cases where a clonal expansion might lead to a false positive, TIgGER examines the combinations of J gene and junction length among sequences which perfectly match the proposed germline allele. Clonally related sequences usually share the same V gene, J gene and junction length. Requiring the novel allele to be found in different combinations of J gene and junction lengths is a proxy for requiring it to be found in different clonal lineages.

Examples

```
# Plot the evidence for the first (and only) novel allele in the example data
novel <- selectNovel(SampleNovel)
plotNovel(AIRRDb, novel[1, ], v_call="v_call", j_call="j_call",
       seq="sequence_alignment", junction="junction", junction_length="junction_length",
          multiplot=TRUE)
```

---

readIgFasta *Read immunoglobulin sequences*

Description

`readIgFasta` reads a fasta-formatted file of immunoglobulin (Ig) sequences and returns a named vector of those sequences.

Usage

```
readIgFasta(fasta_file, strip_down_name = TRUE, force_caps = TRUE)
```

Arguments

`fasta_file`	fasta-formatted file of immunoglobulin sequences.
`strip_down_name`	if `TRUE`, will extract only the allele name from the strings fasta file's sequence names.
`force_caps`	if `TRUE`, will force nucleotides to uppercase.

Value

Named vector of strings representing Ig alleles.

See Also

writeFasta to do the inverse.

Examples

```
## Not run:
# germlines <- readIgFasta("ighv.fasta")

## End(Not run)
```

`reassignAlleles` *Correct allele calls based on a personalized genotype*

Description

`reassignAlleles` uses a subject-specific genotype to correct correct preliminary allele assignments of a set of sequences derived from a single subject.

Usage

```
reassignAlleles(
  data,
  genotype_db,
  v_call = "v_call",
  seq = "sequence_alignment",
  method = "hamming",
  path = NA,
  keep_gene = c("gene", "family", "repertoire"),
  trim_seq = FALSE,
  overwrite = FALSE,
  ignored_regex = "[\\.N-]",
  treat_multigene_as_uncalled = FALSE,
  top_k = NULL,
  top_by = c("alphabetical", "mutation_count"),
  strip_d = TRUE,
  reassign_uncalled = TRUE
)
```

Arguments

data	data.frame containing V allele calls from a single subject and the sample IMGT-gapped V(D)J sequences under seq.
genotype_db	vector of named nucleotide germline sequences matching the calls detailed in allele_calls and personalized to the subject
v_call	name of the column in data with V allele calls. Default is v_call.
seq	name of the column in data with the aligned, IMGT-numbered, V(D)J nucleotide sequence. Default is SEQUENCE_IMGT
method	method to use when realigning sequences to the genotype_db sequences. Currently, only "hamming" (for Hamming distance) is implemented.
path	directory containing the tool used in the realignment method, if needed. Hamming distance does not require a path to a tool.
keep_gene	string indicating if the gene ("gene"), family ("family") or complete repertoire ("repertoire") assignments should be performed. Use of "gene" increases speed by minimizing required number of alignments, as gene level assignments will be maintained when possible.
trim_seq	if TRUE, trim sample and germline sequences to the segment boundaries before calculating Hamming distance. Boundaries are determined from the segment prefix of v_call, such as v_*, d_*, or j_* columns.
overwrite	if TRUE, replace v_call with reassigned calls instead of writing a *_call_genotyped column.
ignored_regex	regular expression indicating characters to ignore when comparing sequences. May also be TRUE to ignore nothing (every position counts), as used for D and J segments.
treat_multigene_as_uncalled	if TRUE, sequences whose call spans more than one gene are treated as uncalled and realigned against the whole genotype rather than kept at their first gene. Only applies when keep_gene is "gene" or "repertoire".
top_k	maximum number of equally-best alleles to report per sequence. NULL (default) keeps all ties.
top_by	how to break ties when more than top_k alleles are equally close. "alphabetical" keeps the first top_k by name; "mutation_count" keeps all ties.
strip_d	if TRUE (default) remove the "D" from the end of gene annotations (denoting a duplicate gene in the locus) when grouping the genotype_db alleles by gene. If FALSE, the "D" is kept, so the genotype grouping and the sequence calls are matched consistently (use together with genotypeFasta(strip_d=FALSE)).
reassign_uncalled	if TRUE (default), sequences whose call is empty or NA are also realigned against the whole genotype. If FALSE, they are left unassigned.

Details

In order to save time, initial gene assignments are preserved and the allele calls are chosen from among those provided in genotype_db, based on a simple alignment to the sample sequence.

Value

A modified input data.frame containing the best allele call from among the sequences listed in genotype_db in the *_call_genotyped column, or in v_call when overwrite=TRUE.

Examples

```
# Extract the database sequences that correspond to the genotype
genotype_db <- genotypeFasta(SampleGenotype, SampleGermlineIGHV, novel=SampleNovel)

# Use the personalized genotype to determine corrected allele assignments
output_db <- reassignAlleles(AIRRDb, genotype_db, v_call="v_call",
                             seq="sequence_alignment")
```

SampleDb *Example human immune repertoire data*

Description

A data.frame of example V(D)J immunoglobulin sequences derived from a single individual (PGP1), sequenced on the Roche 454 platform, and assigned by IMGT/HighV-QUEST to IGHV1 family alleles.

Format

A data.frame where rows correspond to unique V(D)J sequences and columns include:

- "SEQUENCE_IMGT": IMGT-gapped V(D)J nucleotide sequence.
- "V_CALL": IMGT/HighV-QUEST V segment allele calls.
- "D_CALL": IMGT/HighV-QUEST D segment allele calls.
- "J_CALL": IMGT/HighV-QUEST J segment allele calls.
- "JUNCTION_LENGTH": Junction region length.

References

1. Gadala-Maria, et al. (2015) Automated analysis of high-throughput B cell sequencing data reveals a high frequency of novel immunoglobulin V gene segment alleles. PNAS. 112(8):E862-70.

See Also

See AIRRDb for an AIRR formatted version of SampleDb.

SampleGenotype *Example genotype inference results*

Description

A `data.frame` of genotype inference results from inferGenotype after novel allele detection via findNovelAlleles. Source data was a collection of V(D)J immunoglobulin sequences derived from a single individual (PGP1), sequenced on the Roche 454 platform, and assigned by IMGT/HighV-QUEST to IGHV1 family alleles.

Format

A `data.frame` where rows correspond to genes carried by an individual and columns lists the alleles of those genes and their counts.

References

1. Gadala-Maria, et al. (2015) Automated analysis of high-throughput B cell sequencing data reveals a high frequency of novel immunoglobulin V gene segment alleles. PNAS. 112(8):E862-70.

See Also

See inferGenotype for detailed column descriptions.

SampleGermlineIGHV *Example Human IGHV germlines*

Description

A `character` vector of all 344 human IGHV germline gene segment alleles in IMGT/GENE-DB release 2014-08-4.

Format

Values correspond to IMGT-gaped nucleotide sequences (with nucleotides capitalized and gaps represented by ".") while names correspond to stripped-down IMGT allele names (e.g. "IGHV1-18*01").

References

1. 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.

SampleNovel *Example novel allele detection results*

Description

A `data.frame` of novel allele detection results from findNovelAlleles. Source data was a collection of V(D)J immunoglobulin sequences derived from a single individual (PGP1), sequenced on the Roche 454 platform, and assigned by IMGT/HighV-QUEST to IGHV1 family alleles.

Format

A `data.frame` where rows correspond to alleles checked for polymorphisms and columns give results as well as parameters used to run the test.

References

1. Gadala-Maria, et al. (2015) Automated analysis of high-throughput B cell sequencing data reveals a high frequency of novel immunoglobulin V gene segment alleles. PNAS. 112(8):E862-70.

See Also

See findNovelAlleles for detailed column descriptions.

selectNovel *Select rows containing novel alleles*

Description

`selectNovel` takes the result from findNovelAlleles and selects only the rows containing unique, novel alleles.

Usage

```
selectNovel(novel, keep_alleles = FALSE)
```

Arguments

`novel`	`data.frame` of the type returned by findNovelAlleles.
`keep_alleles`	`logical` indicating if different alleles leading to the same novel sequence should be kept. See Details.

Details

If, for instance, subject has in his genome IGHV1-2*02 and a novel allele equally close to IGHV1-2*02 and IGHV1-2*05, the novel allele may be detected by analyzing sequences that best align to either of these alleles. If keep_alleles is TRUE, both polymorphic allele calls will be retained. In the case that multiple mutation ranges are checked for the same allele, only one mutation range will be kept in the output.

Value

A data.frame containing only unique, novel alleles (if any) that were in the input.

Examples

```
novel <- selectNovel(SampleNovel)
```

---

sortAlleles *Sort allele names*

Description

sortAlleles returns a sorted vector of strings representing Ig allele names. Names are first sorted by gene family, then by gene, then by allele. Duplicated genes have their alleles are sorted as if they were part of their non-duplicated counterparts (e.g. IGHV1-69D*01 comes after IGHV1-69*01 but before IGHV1-69*02), and non-localized genes (e.g. IGHV1-NL1*01) come last within their gene family.

Usage

```
sortAlleles(allele_calls, method = c("name", "position"))
```

Arguments

allele_calls	vector of strings representing Ig allele names.
method	string defining the method to use when sorting alleles. If "name" then sort in lexicographic order. If "position" then sort by position in the locus, as determined by the final two numbers in the gene name.

Value

A sorted vector of strings representing Ig allele names.

See Also

Like sortAlleles, updateAlleleNames can help format a list of allele names.

Examples

```
# Create a list of allele names
alleles <- c("IGHV1-69D*01","IGHV1-69*01","IGHV1-2*01","IGHV1-69-2*01",
             "IGHV2-5*01","IGHV1-NL1*01", "IGHV1-2*01,IGHV1-2*05",
             "IGHV1-2", "IGHV1-2*02", "IGHV1-69*02")

# Sort the alleles by name
sortAlleles(alleles)

# Sort the alleles by position in the locus
sortAlleles(alleles, method="pos")
```

---

subsampleDb *Subsample repertoire*

Description

subsampleDb will sample the same number of sequences for each gene, family or allele (specified with mode) in data. Samples or subjects can be subsampled independent by setting group.

Usage

```
subsampleDb(
  data,
  gene = "v_call",
  mode = c("gene", "allele", "family"),
  min_n = 1,
  max_n = NULL,
  group = NULL
)
```

Arguments

data	data.frame containing repertoire data.
gene	name of the column in data with allele calls. Default is v_call.
mode	one of c("gene", "family", "allele") defining the degree of specificity regarding allele calls when subsetting sequences. Determines how data will be split into subsets from which the same number of sequences will be subsampled. See also group.
min_n	minimum number of observations to sample from each group. A group with less observations than the minimum is excluded.
max_n	maximum number of observations to sample for all mode groups. If NULL, it will be set automatically to the size of the smallest group. If max_n is larger than the available number of sequences for any mode group, it will be automatically adjusted and the effective max_n used will be the size of the smallest mode group.

group columns containing additional grouping variables, e.g. sample_id. These groups will be subsampled independently. If max_n is NULL, a max_n will be automatically set for each group.

Details

data will be split into gene, allele or family subsets (mode) from which the same number of sequences will be subsampled. If mode=gene, for each gene in the field gene from data, a maximum of max_n sequences will be subsampled. Input sequences that have multiple gene calls (ties), can be subsampled from any of their calls, but these duplicated samplings will be removed, and the final subsampled data will contain unique rows.

Value

Subsampled version of the input data.

See Also

selectNovel

Examples

```
set.seed(1)
subsampleDb(AIRRDb)
```

---

tigger *tigger*

Description

Here we provide a **T**ool for **I**mmuno**g**lobulin **G**enotype **E**lucidation via **R**ep-Seq (TIgGER). TIgGER infers the set of Ig alleles carried by an individual (including any novel alleles) and then uses this set of alleles to correct the initial assignments given to sample sequences by existing tools.

Details

Immunoglobulin repertoire sequencing (AIRR-Seq, Rep-Seq) data is currently the subject of much study. A key step in analyzing these data involves assigning the closest known V(D)J germline alleles to the (often somatically mutated) sample sequences using a tool such as IMGT/HighV-QUEST. However, if the sample utilizes alleles not in the germline database used for alignment, this step will fail. Additionally, this alignment has an associated error rate of ~5 mutations. The purpose of TIgGER is to address these issues.

Allele detection and genotyping

- findNovelAlleles: Detect novel alleles.
- plotNovel: Plot evidence of novel alleles.
- inferGenotype: Infer an Ig genotype using a frequency approach.
- inferGenotypeBayesian: Infer an Ig genotype using a Bayesian approach.
- plotGenotype: A colorful genotype visualization.
- genotypeFasta: Convert a genotype to sequences.
- reassignAlleles: Correct allele calls.
- generateEvidence: Generate evidence for the genotype and allele detection inference.

Mutation handling

- getMutatedPositions: Find mutation locations.
- getMutCount: Find distance from germline.
- findUnmutatedCalls: Subset unmutated sequences.
- getPopularMutationCount: Find most common sequence's mutation count.
- insertPolymorphisms: Insert SNPs into a sequence.

Input, output and formatting

- readIgFasta: Read a fasta file of Ig sequences.
- updateAlleleNames: Correct outdated allele names.
- sortAlleles: Sort allele names intelligently.
- cleanSeqs: Standardize sequence format.

References

1. Gadala-Maria, et al. (2015) Automated analysis of high-throughput B cell sequencing data reveals a high frequency of novel immunoglobulin V gene segment alleles. PNAS. 112(8):E862-70.

`updateAlleleNames` *Update IGHV allele names*

Description

`updateAlleleNames` takes a set of IGHV allele calls and replaces any outdated names (e.g. IGHV1-f) with the new IMGT names.

Usage

```
updateAlleleNames(allele_calls)
```

Arguments

allele_calls	vector of strings representing IGHV allele names.

Value

Vector of strings representing updated IGHV allele names.

Note

IMGT has removed IGHV2-5*10 and IGHV2-5*07 as it has determined they are actually alleles 02 and 04, respectively. The updated allele names are based on IMGT release 2014-08-4.

References

1. 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

See Also

Like updateAlleleNames, sortAlleles can help format a list of allele names.

Examples

```
# Create a vector that uses old gene/allele names.
alleles <- c("IGHV1-c*01", "IGHV1-f*02", "IGHV2-5*07")

# Update the alleles to the new names
updateAlleleNames(alleles)
```

---

writeFasta *Write to a fasta file*

Description

writeFasta writes a named vector of sequences to a file in fasta format.

Usage

```
writeFasta(named_sequences, file, width = 60, append = FALSE)
```

Arguments

named_sequences	vector of named string representing sequences
file	the name of the output file.
width	the number of characters to be printed per line. if not between 1 and 255, width with be infinite.
append	logical indicating if the output should be appended to file instead of overwriting it

Value

A named vector of strings representing Ig alleles.

See Also

readIgFasta to do the inverse.

Examples

```
## Not run:
# writeFasta(germlines, "ighv.fasta")

## End(Not run)
```

Index

∗ **data**
    AIRRDb, 3
    GermlineIGHV, 12
    SampleDb, 27
    SampleGenotype, 28
    SampleGermlineIGHV, 28
    SampleNovel, 29

AIRRDb, 3, *27*

cleanSeqs, 3, *33*

findNovelAlleles, 4, *9*, *11, 12*, *15, 16*, *18*, *23*, *28, 29*, *33*
findUnmutatedCalls, 8, *33*

generateEvidence, 9, *33*
genotypeFasta, *9*, *11*, 11, *17*, *19*, *33*
GermlineIGHV, 12
getMutatedPositions, 13, *15*, *33*
getMutCount, 14, *33*
getPopularMutationCount, 15, *33*

inferGenotype, *7*, *9*, *11, 12*, 16, *19*, *21*, *28*, *33*
inferGenotypeBayesian, *7*, *17*, 17, *21, 22*, *33*
insertPolymorphisms, 20, *33*

plotGenotype, *17*, *19*, 20, *21, 22*, *33*
plotGenotypeConfidence, *21*, 21
plotNovel, *7*, 23, *33*

readIgFasta, 24, *33*, *35*
reassignAlleles, *9*, 25, *33*

SampleDb, *3*, 27
SampleGenotype, 28
SampleGermlineIGHV, 28
SampleNovel, 29
selectNovel, *7*, 29, *32*
sortAlleles, *4*, 30, *33, 34*
subsampleDb, 31

tigger, 32

updateAlleleNames, *4*, *30*, *33*, 33

writeFasta, *25*, 34