Package 'tigger'

September 21, 2017

Type Package

Version 0.2.11

Date 2017-09-21

Title R Tools for Inferring New Immunoglobulin Alleles from Rep-Seq Data

Description Infers the V genotype of an individual from immunoglobulin (Ig) repertoire-sequencing (Rep-Seq) data, including detection of any novel alleles. This information is then used to correct existing V allele calls from among the sample sequences.

License CC BY-SA 4.0

URL http://tigger.readthedocs.io

BugReports https://bitbucket.org/kleinstein/tigger/issues

LazyData true

BuildVignettes true

VignetteBuilder knitr

Depends R (>= 3.2.5), ggplot2 (>= 2.0.0)

Imports alakazam (>= 0.2.6), tidyr, dplyr (>= 0.5.0), doParallel, foreach, graphics, grid, iterators, lazyeval, parallel, stats

Suggests knitr, testthat

RoxygenNote 6.0.1

NeedsCompilation no

Author Daniel Gadala-Maria [aut], Jason Vander Heiden [ctb, cre], Steven Kleinstein [aut, cph]

Maintainer Jason Vander Heiden <jason.vanderheiden@yale.edu>

Repository CRAN

Date/Publication 2017-09-21 18:36:16 UTC

R topics documented:

`cleanSeqs` *Clean up nucleotide sequences*

Description

`cleanSeqs` capitalizes nucleotides, replaces "." with "-", and then replaces all characters besides ACGT- with "N".

Usage

```
cleanSeqs(seqs)
```

Arguments

`seqs`	a vector of nucleotide sequences

Value

A vector of nucleotide sequences

See Also

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

Examples

```
# Create messy nucleotide sequences
seqs = c("AGAT.taa-GAG...ATA",
         "GATACAGTXXXXXAGNNNPPPACA")
# Clean them up
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(clip_db, germline_db, v_call = "V_CALL",
  germline_min = 200, min_seqs = 50, auto_mutrange = TRUE,
  mut_range = 1:10, pos_range = 1:312, y_intercept = 0.125,
  alpha = 0.05, j_max = 0.15, min_frac = 0.75, nproc = 1)
```

Arguments

clip_db	a data.frame in Change-O format. See details.
germline_db	a vector of named nucleotide germline sequences matching the V calls in clip_db
v_call	name of the column in clip_db with V allele calls. Default is V_CALL.
germline_min	the minimum number of sequences that must have a particular germline allele call for the allele to be analyzed
min_seqs	the 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	the range of mutations that samples may carry and be considered by the algorithm
pos_range	the range of IMGT-numbered positions that should be considered by the algorithm
y_intercept	the y-intercept above which positions should be considered potentially polymorphic

alpha	the alpha cutoff to be used when constructing the confidence interval for the y-intercept
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	the minimum fraction of sequences that must have usable nucleotides in a given position for that position to considered
nproc	the number of processors to use

Details

A data.frame in Change-O format contains the following columns:

- "SEQUENCE_IMGT" containing the IMGT-gapped nucleotide sequence
- "V_CALL" containing the IMGT/V-QUEST V allele call(s)
- "J_CALL" containing the IMGT/V-QUEST J allele call(s)
- "JUNCTION_LENGTH" containing the junction length

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.

See Also

plotNovel to visualize the data supporting any novel alleles hypothesized to be present in the data and inferGenotype to determine if the novel alleles are frequent enought to be included in the subject's genotype

Examples

```
# Load example data and germlines
data(sample_db)
data(germline_ighv)

# Find novel alleles and return relevant data
## Not run: novel_df = findNovelAlleles(sample_db, germline_ighv)
```

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	a vector of strings respresenting 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	a vector of named nucleotide germline sequences

Value

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

Examples

```
# Load data
data(germline_ighv)
data(sample_db)

# Find which of the sample alleles are unmutated
calls <- findUnmutatedCalls(sample_db$V_CALL, sample_db$SEQUENCE_IMGT,
         germline_db=germline_ighv)
```

genotype *Example of an Inferred Genotype*

Description

Example VDJ-rearranged immunoglobulin Rep-Seq sequences derived from a single individual (PGP1), sequenced on the Roche 454 platform, and thought by IMGT/V-QUEST to utilize IGHV1 family alleles, as processed by findNovelAlleles and inferGenotype

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

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.

`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_df = NA)
```

Arguments

`genotype`	a table of alleles denoting a genotype, as returned by inferGenotype
`germline_db`	a vector of named nucleotide germline sequences matching the alleles detailed in `genotype`
`novel_df`	an optional `data.frame` containing putative novel alleeles of the type returned by findNovelAlleles

Value

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

See Also

inferGenotype

Examples

```
# Load example data
data(germline_ighv)
data(novel_df)
data(genotype)

# Find the sequences that correspond to the genotype
genotype_seqs = genotypeFasta(genotype, germline_ighv, novel_df)
```

germline_ighv *Human IGHV germlines*

Description

A `character` vector of all 344 human IGHV germline gene segment alleles in IMGT Gene-db release 201408-4.

Format

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

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.

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`	a vector of strings respresenting aligned sequences
`germlines`	a vector of strings respresenting aligned sequences to which `samples` will be compared. If only one string is submitted, it will be used for all `samples`.
`ignored_regex`	a 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 seqeunce and any germline alleles contained within the call

Usage

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

Arguments

`samples`	a vector of IMGT-gapped sample V sequences
`allele_calls`	a vector of strings respresenting Ig allele calls for the sequences in `samples`, where multiple calls are separated by a comma
`germline_db`	a 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

```
# Load germline database
data(germline_ighv)

# Use createGermlines to insert a mutation into a germline sequence
#sample_seqs = c(germline_ighv[2],
#                createGermlines(germline_ighv[1], 103, "G"),
#                createGermlines(germline_ighv[1], 107, "C"))
```

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

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

`getPopularMutationCount`

Find Frequent Sequences' Mutation Counts

Description

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

Usage

```
getPopularMutationCount(sample_db, germline_db, gene_min = 0.001,
  seq_min = 50, seq_p_of_max = 1/8, full_return = FALSE)
```

Arguments

`sample_db`	A Change-O db data frame. See findNovelAlleles for a list of required columns.
`germline_db`	A named list of IMGT-gapped germline sequences.
`gene_min`	The portion of all unique sequences a gene must constitute to avoid exclusion.
`seq_min`	The number of copies of the V that must be present for to avoid exclusion.
`seq_p_of_max`	For each gene, fraction of the most common V sequence's count that a sequence must meet to avoid exclusion.
`full_return`	If true, will return all `sample_db` 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

```
data(sample_db, germline_ighv)
getPopularMutationCount(sample_db, germline_ighv)
```

`inferGenotype` *Infer a subject-specific genotype*

Description

`inferGenotype` infers an subject's genotype 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(clip_db, v_call = "V_CALL", fraction_to_explain = 0.875,
  gene_cutoff = 1e-04, find_unmutated = TRUE, germline_db = NA,
  novel_df = NA)
```

Arguments

`clip_db`	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
`v_call`	column in `clip_db` with V allele calls. Default is `"V_CALL"` be provided in a column `"SEQUENCE_IMGT"`
`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.
`germline_db`	named vector of sequences containing the germline sequences named in `allele_calls`. Only required if `find_unmutated` is `TRUE`.
`novel_df`	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.

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_df` 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 assignent prior to searching for unmutated sequences.

Value

A table of alleles denoting the genotype of the subject

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.

Examples

```
# Infer the IGHV genotype, using only unmutated sequences, including any
# novel alleles
data(sample_db)
data(germline_ighv)
data(novel_df)
inferGenotype(sample_db, find_unmutated = TRUE, germline_db = germline_ighv,
              novel_df = novel_df)
```

---

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`	the starting nucletide sequence
`positions`	a vector of positions which to be changed
`nucleotides`	a vector of nucletides to which to change the positions

Value

a sequence with the desired nucleotides in provided locations

Examples

```
insertPolymorphisms("hugged", c(1,6,2), c("t","r","i"))
```

novel_df *Example of Analyzed Rep-Seq data*

Description

Example VDJ-rearranged immunoglobulin Rep-Seq sequences derived from a single individual (PGP1), sequenced on the Roche 454 platform, and thought by IMGT/V-QUEST to utilize IGHV1 family alleles, as processed by findNovelAlleles.

Format

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

References

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.

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, ...)
```

Arguments

`genotype`	a table of alleles denoting a genotype, as returned by inferGenotype
`facet_by`	a column name in `genotype` to facet the plot by. If `NULL`, then do not facet the plot.
`gene_sort`	a 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`	the 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.
`...`	additional arguments to pass to ggplot2::theme.

Value

A ggplot object defining the plot.

See Also

inferGenotype

Examples

```
# Load example data
data(novel_df)
data(genotype)

# Plot genotype
plotGenotype(genotype)

# Facet by subject
genotypea = genotypeb = genotype
genotypea$SUBJECT = "A"
genotypeb$SUBJECT = "B"
geno_sub = rbind(genotypea, genotypeb)
plotGenotype(geno_sub, facet_by="SUBJECT", gene_sort="pos")
```

---

plotNovel *Visualize evidence of novel V alleles*

Description

`plotNovel` is be used to visualize the evidence of any novel V alleles found using findNovelAlleles.

Usage

```
plotNovel(clip_db, novel_df_row, ncol = 1, v_call = "V_CALL")
```

Arguments

`clip_db`	a `data.frame` in Change-O format. See findNovelAlleles for details.
`novel_df_row`	a single row from a data frame as output by findNovelAlleles that contains a polymorphism-containing germline allele
`ncol`	number of columns to use when laying out the plots
`v_call`	name of the column in `clip_db` with V allele calls. Default is "V_CALL"

Examples

```
# Load example data and germlines
data(sample_db)
data(germline_ighv)

# Find novel alleles and return relevant data
## Not run: novel_df = findNovelAlleles(sample_db, germline_ighv)
data(novel_df)
# Plot the evidence for the first (and only) novel allele in the example data
novel = selectNovel(novel_df)
plotNovel(sample_db, novel[1,])
```

`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 immunoglobuling 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

a named vector of strings respresenting Ig alleles

See Also

writeFasta to do the inverse.

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(clip_db, genotype_db, v_call = "V_CALL", method = "hamming",
  path = NA, keep_gene = TRUE)
```

Arguments

clip_db	a data.frame containing V allele calls from a single subject and the sample IMGT-gapped V(D)J sequences under "SEQUENCE_IMGT"
genotype_db	a 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 clip_db with V allele calls. Default is "V_CALL"
method	the method to be used when realigning sequences to the genotype_db sequences. Currently only "hammming" (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	logical indicating if gene assignments should be maintained when possible. Increases speed by minimizing required number of alignments. Currently only "TRUE" is implemented.

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 single-column data.frame corresponding to clip.db and containing the best allele call from among the sequences listed in genotype_db

Examples

```
# Load example data
data(germline_ighv)
data(sample_db)
data(genotype)
data(novel_df)

# Extract the database sequences that correspond to the genotype
```

```
genotype_seqs = genotypeFasta(genotype, germline_ighv, novel_df)

# Use the personlized genotype to determine corrected allele assignments
V_CALL_GENOTYPED = reassignAlleles(sample_db, genotype_seqs)
sample_db = cbind(sample_db, V_CALL_GENOTYPED)
```

`sample_db` *Example human Rep-Seq data*

Description

Example VDJ-rearranged immunoglobulin Rep-Seq sequences derived from a single individual (PGP1), sequenced on the Roche 454 platform, and thought by IMGT/V-QUEST to utilize IGHV1 family alleles.

Format

A `data.frame` where rows correspond to unique VDJ sequences and columns include:

- IMGT-gapped nucleotide sequence (`"SEQUENCE_IMGT"`)
- IMGT/V-QUEST allele calls (`"V_CALL"`, `"D_CALL"`, and `"J_CALL"`)
- Junction length (`"JUNCTION_LENGTH"`)

References

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.

`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_df, keep_alleles = FALSE)
```

Arguments

`novel_df`	A `data.frame` of the type returned by findNovelAlleles
`keep_alleles`	A `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

```
data(novel_df)
novel = selectNovel(novel_df)
```

`sortAlleles` *Sort allele names*

Description

`sortAlleles` returns a sorted vector of strings respresenting 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`	a vector of strings respresenting Ig allele names
`method`	a 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 respresenting 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")
```

---

tigger *tigger*

Description

Here we provide a **T**ool for **I**mmuno**g**lobulin **G**enotype **E**lucidation via **R**ep-Seq (TIgGER). TIgGER inferrs 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 (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 percent, notably among sequences carrying a large number of somatic mutations. The purpose of TIgGER is to address these issues.

Core tigger functions

- findNovelAlleles: Detect novel alleles
- plotNovel: Plot evidence of novel alleles
- inferGenotype: Infer an Ig genotype
- plotGenotype: A colorful genotype visualization
- genotypeFasta: Convert a genotype to sequences
- reassignAlleles: Correct allele calls

Mutation-related functions

- 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 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

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`	a vector of strings respresenting IGHV allele names

Details

The updated allele names are based on IMGT release 201408-4.

Value

vector of strings respresenting updated IGHV allele names

Note

IGMT has removed IGHV2-5*10 and IGHV2-5*07 as it has determined they are actually alleles *02 and *04, respectively.

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

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	a 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 respresenting Ig alleles

See Also

readIgFasta to do the inverse.

Index