Package: HLAtools (via r-universe)

November 15, 2024

Type Package

Title Toolkit for HLA Immunogenomics

Version 1.3.0

Maintainer Steven Mack <Steven.Mack@ucsf.edu>

Description A toolkit for the analysis and management of data for genes in the so-called ``Human Leukocyte Antigen'' (HLA) region. Functions extract reference data from the Anthony Nolan HLA Informatics Group/ImmunoGeneTics HLA 'GitHub' repository (ANHIG/IMGTHLA) <https://github.com/ANHIG/IMGTHLA>, validate Genotype List (GL) Strings, convert between UNIFORMAT and GL String Code (GLSC) formats, translate HLA alleles and GLSCs across ImmunoPolymorphism Database (IPD) IMGT/HLA Database release versions, identify differences between pairs of alleles at a locus, generate customized, multi-position sequence alignments, trim and convert allele-names across nomenclature epochs, and extend existing data-analysis methods.

Encoding UTF-8

RoxygenNote 7.3.1

Imports DescTools, dplyr, stringr, tibble, utils, xfun, fmsb

Depends R (>= 3.5.0)

LazyData true

LazyDataCompression xz

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

License GPL (>= 3)

URL <https://github.com/sjmack/HLAtools>

NeedsCompilation no

Author Livia Tran [aut], Ryan Nickens [aut], Leamon Crooms IV [aut], Derek Pappas [aut], Vinh Luu [ctb], Josh Bredeweg [ctb], Steven Mack [aut, cre] (<https://orcid.org/0000-0001-9820-9547>)

Repository CRAN

Date/Publication 2024-11-14 05:20:03 UTC

Config/pak/sysreqs make libicu-dev libssl-dev libx11-dev zlib1g-dev

Contents

`addCodonLine` *Add an 'AA codon' Line to Alignments When Missing.*

Description

Modifies cDNA alignment objects that are missing "AA codon" lines to include these lines in the correct location with the correct codon position information.

Usage

```
addCodonLine(cDNAalign, firstPos, afterLine = 8, codons = 25)
```

Arguments

cDNAalign	A matrix of cDNA alignment lines, generated from an alignment file that is missing "AA codon" lines.
firstPos	A numeric value identifying the position number of the transcript's N-terminal codon, based on a complete cDNA alignment in another release.
afterLine	A numeric value identifying the number of the line below the first "cDNA" line in the alignment. The default value is 8.
codons	A numeric value identifying the number of codons in each line of the nucleotide alignment. The default value is 25.

Value

A complete cDNA alignment data frame that includes "AA codon" rows.

Note

For internal HLAtools use.

---

alignmentFull *Build Sets of Protein, Codon, Coding Nucleotide and Genomic Nucleotide Alignments for Specified Loci*

Description

Applies buildAlignments() to build a set of alignments for loci supported in the ANHIG/IMGTHLA GitHub repository.

Usage

```
alignmentFull(loci = "all", alignType = "all", version = HLAgazeteer$version)
```

Arguments

loci	A characer vector of the locus names for which alignments should be built. The default value ("all") generates alignments for all loci.
alignType	A character vector of alignment types. The allowed values are "all", "prot", "codon", "nuc", and "gen", which specify either all available alignments for a given locus or the respective protein, codon, nucleotide and genomic alignments, as determined by the HLAgazeteer.
version	A character string describing the version of the ANHIG/IMGTHLA Github repository from which alignments are obtained. The default value, 'HLAgazeteer$version', generates alignments for the IPD-IMGT/HLA Database release under which the HLAgazeteer was built.

Value

A list object containing data frames of protein (prot), codon (codon), coding nucleotide (nuc), or genomic nucleotide (gen) alignments, for specified genes in the specified IPD-IMGT/HLA Database release, along with the pertinent reference database release.

Note

AlignmentFull() must be run to build the 'HLAalignments' object (HLAalignments <- alignmentFull()) before other functions that use alignments can be used, and therefore requires internet access.

Depending on local download speeds, building all available alignments for all loci can take several minutes.

Prior to IPD-IMGT/HLA Database release version 3.24.0, the HLA-DP and HLA-DQ sequence alignment files in the IPD-IMGT/HLA GitHub Repository did not all include a numerical suffix in the gene name (e.g., the protein sequence alignment file for the DQA1 gene was named 'DQA_prot.txt') because alignment files for the DPA2, DPB2, DQA2 and DQB2 genes had not been made available. Building DPA1, DPB1, DQA1, and DQB1 sequence alignments from releases prior to 3.24.0 require using a gene name that does not include the numerical suffix.

---

alignmentSearch *Search Alignments for Specific Positions in a Specific Allele*

Description

Searches ANHIG/IMGT-HLA alignments and returns protein, codons or nucleotide sequences for a submitted allele name and position(s).

Usage

```
alignmentSearch(alignType, allelename, positions, prefix = TRUE, sep = "~")
```

Arguments

alignType	The type of alignment being searched. Allowed values are "codon","gen", nuc" and "prot". Only one 'alignType' value is allowed.
allelename	A full or 2-field HLA allele name, excluding the "HLA-" prefix.
positions	A vector of sequence positions (e.g., c(-17,1,22,130)); in nucleotide and genomic alignments, indel positions are named using decimals. So the first indel between positions 26 and 27 is named 26.1, and the second indel between 26 and 27 is 26.2, etc.
prefix	A logical that indicates if the position number(s) should be included in the result.
sep	The value that will be used to separate the sequences for each position. No value can be specified (sep=""). If prefix=FALSE and sep="", a sequence block is returned.

Value

A character string containing the corresponding peptide, codon or nucleotide sequence for each position in 'positions' for 'allelename'. A sequence is not returned if 'allelename' is not not found in the pertinent alignment. A position will be empty if 'allelename' does not have a value at the specified position.

Note

This function requires that the HLAalignments object has been populated with alignments via the alignmentFull() function.

Indel positions must be text-formatted (e.g. "607.12"). C.f., alignmentSearch("nuc","DRB1*15:07:01",11:22) *vs alignmentSearch("nuc","DRB1*11:250N",c(605,"607.1","607.12",608)).

---

alleleListHistory *Allele Names Across All Release Versions*

Description

A large list object of two elements. The first element contains version information. The second element contains all releases of allele name lists in the IMGT/HLA database. This object is built by the UpdateAlleleListHistory() function.

Usage

```
data(alleleListHistory)
```

Format

A large list of 2 elements that identify the table of allele names for each IPD-IMGT/HLA reference Database release and the source release

- (Version: the Date and IPD-IMGT HLA Release version under which the alleleListHistory object was made)
- (AlleleListHistory: a dataframe version of the alleleListHistory.txt file)

Source

https://raw.githubusercontent.com/ANHIG/IMGTHLA/Latest/Allelelist_history.txt

`alleleTrim` *Trim All Versions of HLA Allele Names*

Description

Trims an HLA allele name to a specified number of fields or number of digits, depending on the nomenclature epoch. Expression variant suffixes in full-length allele names can be appended to truncated allele names.

Usage

```
alleleTrim(allele, resolution, version = 3, append = FALSE)
```

Arguments

`allele`	A character string of an HLA allele name formatted as locus*allele_name, optionally including the "HLA-" prefix.
`resolution`	A number identifying the number of fields to trim the allele name to.
`version`	A number identifying the HLA nomenclature epoch under which the allele was named. Epoch 1 allele names are found in IPD-IMGT/HLA Database releases 1.0.0 to 1.16.0. Epoch 2 allele names are found in IPD-IMGT/HLA Database releases 2.0.0 to 2.28.0. Epoch 3 allele names are found in IPD_IMGT/HLA Database releases 3.0.0 and onward.
`append`	A logical. When append = TRUE, the expression variant suffix of a full-length allele name is appended to a truncated allele name. The default value is FALSE.

Value

A character string of the trimmed allele name, shortened according to the input parameters.

Note

Expression variant suffixes will not be removed from full-length allele names that have fewer than four fields or eight digits.

Examples

```
alleleTrim(allele = "A*03:01:01", resolution = 2)
alleleTrim(allele = "A*030101", resolution = 2,version = 2)
alleleTrim(allele = "A*0303N",resolution = 1,version = 1, append = TRUE)
alleleTrim("HLA-A*24020102L",resolution = 3,version = 2, append = TRUE)
```

atlasFull — *Generate a Complete set of Protein, Coding nucleotide and Genomic Nucleotide Atlases*

Description

Applies atlasMaker() to build a set of 'atlases' identifying the gene-feature (exon, intron, and UTR) boundaries in the protein, coding nucleotide, and genomic nucleotide alignments of all HLA loci.

Usage

```
atlasFull(version = "Latest")
```

Arguments

version	A character string identifying the version of the ANHIG/IMGTHLA Github repository used to generate atlases. The default value, "Latest", generates an atlas for the most recent IPD-IMGT/HLA Database release.

Value

A list object containing a list of data frames of protein (prot), coding nucleotide (nuc), and genomic nucleotide (gen) atlases for all genes in the IPD-IMGT/HLA Database release, along with a character vector identifying the pertinent IPD-IMGT/HLA Database release.

Note

Data informing the atlases were downloaded from the ANHIG/IMGTHLA Github repository.

For internal HLAtools use.

atlasMaker — *Identify the Gene-Feature Boundaries in HLA Region Genes*

Description

The 'AA' atlas identifies the amino acid (AA) positions encoded by codons that span each exon (E) boundary. The 'cDNA' atlas describes the 3' nucleotide position in the cDNA sequence that follows each exon boundary, as well as the codon that spans each exon boundary. The 'gDNA' atlas describes first 3' nucleotide position following each UTR (U), exon or intron (I) boundary. Each feature is followed by its rank (e.g U3 is the 3' UTR). Non-standard gene-features (H, J, N, and S) are detailed in the documentation for the ffN() function.

Usage

```
atlasMaker(loci, source, version = "Latest")
```

Arguments

loci	A character string identifying a specific HLA gene (ex. "DRB1"). More than one gene can be specified (e.g., c("DRB1","DQB1")).
source	A character string identifying the type of alignment used to build the atlas. The allowed options are "AA", for protein alignments, "cDNA", for nucleotide alignments, and "gDNA", for genomic alignments. More than one alignment type can be specified (e.g., c("AA","gDNA")).
version	A character string identifying the release version of the IPD-IMGT/HLA Github repository for which alignments should be generated.

Details

Builds an 'atlas' of the gene-feature (exon, intron and UTR) boundaries for IPD-IMGT/HLA loci, using IPD-IMGT/HLA Database alignments.

Value

A list object of 'atlas' dataframes and a 'release version' character string for each locus and alignment type.

Note

For internal HLAtools use.

Nucleotide atlases for pseudogenes will include a 'codon' row populated with NA values.

Some HLA-DQB1*05 and *06 alleles include a 5th exon that is not present in the DQB1*05:01:01:01 reference allele. In this case, the Exon 4 to Exon 5 boundary is defined as 'Ins' (insertion). For all other alleles there is no E.4-5 boundary.

Boundaries for non-standard hybrid (H), join (J), novel (N) and segment (S) features may be included in gene fragment and pseudogene atlases.

BDstrat *Stratify BIGDAWG Datasets by Specific Alleles*

Description

Divides a BIGDAWG-formatted dataset into two subsets (strata) that do and do not include specified alleles.

Usage

```
BDstrat(dataset, alleles, warnBelow = 21)
```

Arguments

dataset	A BIGDAWG-formatted data frame or a path to a BIGDAWG-formatted, tab-delimited text file.
alleles	A vector of allele names in the locus-asterisk-allele_name format (e.g., "A*01:01:01:01").
warnBelow	An integer that defines a low number of subjects in a stratum, generating a warning message. The default value is 21.

Value

A list-object of two BIGDAWG-formatted data frames titled dataset$<alleles>-positive and dataset$<alleles>-negative. The positive list element includes all subjects with the specified alleles, and the negative list element includes all subjects without those specified alleles.

References

BIGDAWG Data Format

Examples

```
HLA_data.multi.strat <- BDstrat(sHLAdata,c("DRB1*16:02:01:01","DRB1*04:07:01:01","A*25:01:01:01"))
HLA_data.single.strat <- BDstrat(sHLAdata,"DRB1*16:02:01:01")
```

BDtoPyPop *Convert BIGDAWG datasets to PyPop datasets Converts a BIGDAWG-formatted data frame into a pair of PyPop-formatted case and control data frames.*

Description

Convert BIGDAWG datasets to PyPop datasets

Converts a BIGDAWG-formatted data frame into a pair of PyPop-formatted case and control data frames.

Usage

```
BDtoPyPop(dataset, filename, save.file = TRUE, save.path = tempdir())
```

Arguments

dataset	A data frame containing a BIGDAWG-formatted case-control dataset
filename	A character string identifying the desired path and name for the generated files or the elements of the returned list object.
save.file	A logical that determines if a pair of files should be written (TRUE) or if a list object should be returned. The default value is TRUE.
save.path	A character string identifying the path in which to write the pair of files when save.file is TRUE. The default value is tempdir().

Value

When save.file = TRUE, a pair of files named "'filename'.positive.pop" and "'filename'.negative.pop" are generated in the working directory. When save.file = FALSE, a list of two-elements, named "'filename'.positive" and "'filename'.negative", is returned.

Note

Files generated by BDtoPyPop are intended for analysis using Pypop version 1.#.#. PyPop is not an R package, and must be installed separately.

References

Lancaster et al. Front Immunol. 2024 Apr 2;15:1378512. https://pubmed.ncbi.nlm.nih.gov/38629078/

Pappas et al. Hum Immunol. 2016 Mar 77(3):283-287. https://pubmed.ncbi.nlm.nih.gov/26708359/

Examples

```
HLAdata.PP <- BDtoPyPop(sHLAdata,"BDHLA",FALSE)
```

---

buildAlignments *Build Amino Acid, cDNA and gDNA Alignments*

Description

Returns a list of data frames of amino acid, codon, nucleotide and genomic alignments, with version information.

Usage

```
buildAlignments(loci, source, version = "Latest")
```

Arguments

loci	A character vector of HLA gene names (e.g., "DRB1", c("A","C")).
source	A character vector of alignment types. The allowed values are "AA", "cDNA", and "gDNA". If 'source' is "cDNA", both codon and cDNA nucleotide alignments are generated. If source is 'AA' or 'gDNA', a single peptide or genomic nucleotide alignment is generated. Up to four alignments will be returned for a locus, as determined by its ability to be transcribed or translated.
version	A character string describing the desired release version (branch) of the ANHIG/IMGTHLA Github repository (e.g. '3.53.0'). The default value ('Latest') returns alignments for the most recent release.

Value

A list object with a data frame of all allele names (and trimmed, two-field allele names) and their corresponding sequences (Amino Acid, codon, cDNA, or gDNA) for a specific locus, as well as version details for the returned information. These alignments identify locations of feature boundaries in relation to amino acid, codon, cDNA, and gDNA sequences. When a three- or four-field allele name includes an expression variant suffix, that suffix is appended to the trimmed name.

Note

For internal HLAtools use.

`buildGazeteer` *Define Categories of Genes Supported by the IPD-IMGT/HLA Database*

Description

Consumes information in the ANHIG/IMGTHLA GitHub repository and at hla.alleles.org/genes to define specific categories of genes supported by the IPD-IMGT/HLA Database, which are represented as nineteen elements of the HLAgazeteer object.

Elements:

- All genes with alignments ($align)
- Genes that that do and do not have amino acid ($prot/$noprot), nucleotide ($nuc/$nonuc), and genomic ($gen/$nogen) alignments
- HLA genes ($hla), pseudogenes ($pseudo), gene fragments ($frag)
- Genes that are expressed ($expressed) or not expressed ($notexpressed)
- Genes in the Class I region ($classireg), Class I HLA genes ($classIhla), Genes in the Class II region ($classiireg), and Class II HLA genes ($classiihla)
- Classical HLA genes ($classical) and non-classical exprssed HLA genes ($nonclassical)
- All genes presented in map order ($map)

The twentieth element ($version) identifies the IPD-IMGT/HLA Database version used to build the HLAgazeteer.

Usage

```
buildGazeteer(version = getLatestVersion())
```

Arguments

`version`	A string identifying of the desired IPD-IMGT/HLA Database release version to which the gazeteer should be updated. The default value is most recent IPD-IMGT/HLA Database release version.

Value

A list object of vectors organizing the genes in the IPD-IMGT/HLA Database into specific categories.

Note

The *$prot* and *$nuc* vectors include a 'DRB' "gene". While 'DRB' is not a gene name, the DRB_prot.txt file includes combined alignments for the DRB1, DRB3, DRB4, and DRB5 genes, and the DRB_nuc.txt file includes combined alignments for the DRB1, DRB2, DRB3, DRB4, DRB5, DRB6, DRB7, DRB8, and DRB9 genes. 'DRB' is included in these vectors for the purpose of validation when these combined alignments are desired.

For interal HLAtools use.

These elements are constructed using data compiled from hla.alleles.org/genes/, github.com/ANHIG/IMGTHLA/tree/Latest/a
DOI:10.2741/a317, DOI:10.1111/tan.15180, and Human Genome Assembly GRCh38.p14 reference assembly NC_000006.12.

Additional information about these genes can be found in HLAtools::IMGTHLAGeneTypes.

The $map element does not distinguish the order of the DRB3, DRB4 and DRB5 genes (DRB3/4/5) or the DR6 and DR7 (DR6/7) genes, as the individual elements of these gene sets are found on different DRB haplotypes.

This function requires internet access to function.

Source

Andersson G. Evolution of the human HLA-DR region. Front. Biosci. 1998 Jul 27:3:d739-45.

Alexandrov et al. HLA-OLI: A new MHC class I pseudogene and HLA-Y are located on a 60 kb indel in the human MHC between HLA-W and HLA-J. HLA 2023 Nov; 102(5):599-606.

---

buildIMGTHLAGeneTypes *Describe IPD-IMGT/HLA Database Genes, Identifying Pseudogenes and Gene Fragments*

Description

buildIMGTHLAGeneTypes() scrapes 'hla.alleles.org/genes/index.html' and generates a data frame identifying each gene. It requires internet access to function. As such, this function always returns data for the current IPD-IMGT/HLA Database release.

Usage

```
buildIMGTHLAGeneTypes()
```

Value

A list object of two elements – 'version' and 'GeneTypes'. The 'version' element identifies the date that the source table was generated. The 'GeneTypes' element is a data frame of three columns, identifying each gene supported by the IPD-IMGT/HLA Database, along with its molecular characteristics and its status as either a pseudogene or a gene fragment.

Note

For internal HLAtools use.

`checkAlignType` *Ensure that AlignType Values are Valid*

Description

Evaluates 'alignType' values to ensure that only "prot", "nuc", "codon" and "gen" are passed to downstream functions. If any other values are entered, a message describing excluded values is generated. If no valid 'alignType' values are are present, an error is generated, and any calling function is ended.

Usage

```
checkAlignType(alignType)
```

Arguments

`alignType`	A vector of character values specifying sequence alignment types to be used for a function.

Value

A character vector that includes only allowed 'alignType' values.

Examples

```
checkAlignType(c("nuc","prot","gDNA"))
```

`checkgDNAstart` *Identify gDNA Alignments in Which the First Feature Boundary is not Identified as Position +1.*

Description

Checks the position of the first feature boundary (the 5' UTR - Exon 1 boundary in expressed genes) in each gDNA alignment, and identifies those alignments in which that position is not 1.

Usage

```
checkgDNAstart(verbose = FALSE)
```

Arguments

verbose	A logical indicating if loci with first feature boundary positions that are not 1 should be reported in the console (verbose = TRUE).

Details

This function reviews the gDNA atlases in the HLAatlas object and returns a data frame of the first feature boundary position for each locus. For expressed genes and some pseudogenes, this is the position of the start of Exon 1.

Value

A one-row data frame with one column for each locus with a gDNA alignment.

Note

This function requires that the HLAalignments object has been populated with alignments via the alignmentFull() function.

For internal HLAtools use.

---

checkSource *Ensure that Source Values are Valid*

Description

Evaluates 'source' values to ensure that only "AA", "cDNA", and "gDNA" are passed to downstream functions. If any other values are entered, a message describing excluded values is generated.

Usage

```
checkSource(source)
```

Arguments

source	A vector of character values specifying sequence alignment file sources to be used for a function.

Value

A character vector that includes only allowed 'source' values.

Examples

```
checkSource(c("AA","cDNA","codon"))
```

checkVersion *Check IPD-IMGT/HLA Release Version Allele Names*

Description

Determines if allele name data for a given IPD-IMGT/HLA Release Version is present in the local HLAtools package.

Usage

```
checkVersion(version)
```

Arguments

version	A character string identifying the release version (branch) of the ANHIG/IMGTHLA Github repository (e.g. '3.53.0'). The value 'Latest' refers to the most recent release.

Value

A logical. TRUE indicates that data for 'version' is available to the local HLAtools package. FALSE means that data for 'version' is not available.

Note

For internal HLAtools use.

Examples

```
checkVersion(version = "3.25.0")
checkVersion(version = "Latest")
```

compareSequences *Identify Sequence Differences Between Two Alleles at a Locus*

Description

Compares the sequences of two alleles at a locus, and identifies the differences between them at specific positions

Usage

```
compareSequences(alignType, alleles)
```

Arguments

alignType	A character string identifying the type of alignment being searched. Allowed values are "codon","gen", nuc" and "prot". Only one 'alignType' value is permitted.
alleles	A character vector containing two full-length names for alleles at the same locus.

Value

A two-row data frame identifying the positions and sequences at which the two alleles differ. E.g., compareSequences(alignType = "gen", alleles = c("DPA1*01:03:38:01","DPA1*01:03:38:02"). Positions for which the sequence of either allele is unknown are ignored.

---

convertAny — *Convert Values Across an Entire Data Frame or Vector Converts all instances of a value in a data frame or vector to a specified value.*

Description

Convert Values Across an Entire Data Frame or Vector

Converts all instances of a value in a data frame or vector to a specified value.

Usage

```
convertAny(dataset, change.from = NA, change.to = "****")
```

Arguments

dataset	A data frame or vector of values.
change.from	The value present in the data frame or vector that should be changed. The default value is NA.
change.to	The value to which 'change.from' should be changed to. The default value is "****".

Value

A data frame or vector in which 'change.from' has been converted to 'change.to'.

Examples

```
dataset <- convertAny(sHLAdata)
```

countSpaces *Count the Spaces in a Character String*

Description

Counts the number of spaces in a character string.

Usage

```
countSpaces(x)
```

Arguments

x	A character string.

Value

A numeric value representing the number of spaces (' ') in x.

Author(s)

Josh Bredeweg, Reddit user jbraids1421

Source

https://www.reddit.com/r/rstats/comments/2th8ic/function_to_count_the_number_of_white_spaces/

Examples

```
countSpaces("abc def")
```

customAlign *Generate a Customized Peptide, Codon or Nucleotide Sequence Alignment.*

Description

Generates a peptide, codon, coding or genomic nucleotide alignment table for user-specified HLA alleles at user-specified positions.

Usage

```
customAlign(alignType, alleles, positions)
```

Arguments

alignType	The type of alignment being searched. Allowed values are "prot", codon", "nuc" and "gen". Only one 'alignType' value is allowed.
alleles	A vector of un-prefixed HLA allele names.
positions	Either a vector of variant positions, against which all loci will be aligned, or a list of vectors of nucleotide positions, exactly one vector for each allele, against which each corresponding allele will be aligned.

Value

A data frame of allele names and the corresponding nucleotide sequences for each desired nucleotide position. an error message is returned if input locus is not available in the ANHIG/IMGTHLA Github Repository.

Note

This function requires that the HLAalignments object has been populated with alignments via the alignmentFull() function. C.f., customAlign("codon",c("DRB1*01:01","DQB1*02:01","DPB1*01:01"),c(1,2,3,7,8,9,13,14,15)) *and customAlign("codon",c("DPB1*01:01:01:01","DQA1*01:01:01:01","DQB1*05:01:01:01"),list(19:35,1:4,6:9)).*

---

expandVersion *Add 'Dot' Delimiters to a Numeric Release Version*

Description

Adds the '.' delimiters to a 'squashed' IPD-IMGT/HLA Database Release Version name

Usage

```
expandVersion(ver)
```

Arguments

ver	A four character/numeral IPD-IMGT/HLA Database Release Version name (e.g., '3540' or 3450).

Value

A character vector of the expanded IPD-IMGT/HLA Database Release Version name.

Note

For internal HLAtools use.

This function assumes single-character-length first and third fields of IPD-IMGT/HLA Database Release Version names.

Examples

```
expandVersion(3450)
```

`ffN` *Identify and Annotate Gene Features in Pseudogenes and Gene Fragments.*

Description

HLA pseudogenes and gene fragments many not share all of the gene features of their expressed homologs. The ffN() function generates a list object identifying the features present in a given gene fragment or pseudogene with a genomic alignment file. This includes annotations of the differences between the pseudogenes and gene fragments, and their expressed homologs. Pseudogenes and gene fragments are identified in the IMGTHLAGeneTypes data object.

Standard Features

- E - Exon, a peptide-encoding sequence
- I - Intron, an intervening sequence found between Exons
- U - UTR, an untranslated region of sequence preceding the first Exon or following the last Exon

Additional Features used in these annotations, based on boundaries indicated in the sequence alignment, are:

- H - Hybrid, a sequence that includes at least one known feature sequence and one nucleotide sequence that does not correspond to a known feature
- J - Join, a sequence that includes two or more features that are separated by a feature boundary in the reference
- N - Novel, a novel sequence that does correspond to a known feature sequence
- S - Segment, a subset of a longer feature sequence

Reference Genes

- HLA-C genomic sequence is used as the reference for class I pseudogenes and gene fragments.
- HLA-DPA1 and -DPB1 genomic sequences are used as the references for class II pseudogenes -DPA2 and -DPB2, respectively.

Usage

```
ffN(version)
```

Arguments

`version`	A character string identifying the pertinent IPD-IMGT/HLA Database release version (e.g., "3.55.0") under which the returned object is generated. This parameter does not impact the generation of the feature names or annotations, and is only included to provide IPD-IMGT/HLA Database release version context.

Value

A list object where each element is the name of a pseudogene or gene fragment. Each of these elements is a list of 'features' and 'annotation'. "Features" identifies the gene features for that gene, ordered from 5' to 3'. Because these genes are not-expressed, the three standard gene features (U, Untranslated Region (UTR); E, Exon; and I, Intron) may not always be present. In these cases, H, J, N and S features (defined below) are returned. Each feature identifier is followed by an identifying number (e.g. U.5 is the 5' UTR, and E.3 is Exon 3). "Annotation" provides the composition of the non-standard H, J, N and S features for each gene. #'

Note

Features and their annotations have been identified manually. Feature annotations will not change unless a new pseudogene or gene fragment is added in a future release, in which case new annotations will be generated.

The H, J, N and S features are described for class I pseudogenes and gene fragment sequences, all of which are described relative to the HLA-C reference sequence. Feature length differences for DPA2 and DPB2, relative to the DPA1 and DPB1 references, are noted in annotations of standard feature abbreviations (E, I and U).

No annotations are included for the DRB2, DRB6, DRB7, and DRB9 genes, as genomic alignments for these genes are not available.

For internal HLAtools use.

Examples

```
fragmentFeatureNames <- ffN("3.35.0")
```

---

formatHead *Format PyPop Data Frame Headers Format the header of a PyPop-formatted data frame.*

Description

Format PyPop Data Frame Headers

Format the header of a PyPop-formatted data frame.

Usage

```
formatHead(colHead)
```

Arguments

colHead	A vector of column names. PyPop format requires that paired locus/gene names should end in '_1' and '_2', respectively.

Value

A vector in which the locus names are suffixed with '_1' and '_2'.

Note

This function assumes that the first two elements the 'colHead' vector are not locus/gene names.

References

Lancaster et al. Front Immunol. 2024 Apr 2;15:1378512. https://pubmed.ncbi.nlm.nih.gov/38629078/

Pappas et al. Hum Immunol. 2016 Mar 77(3):283-287. https://pubmed.ncbi.nlm.nih.gov/26708359/

Examples

```
formatHead(colHead = colnames(sHLAdata))
```

---

fragmentFeatureNames *Gene Features of HLA Pseudogenes and Gene Fragments*

Description

A list object of 15 elements. Each of the 15 elements corresponds to one of the 15 HLA pseudogenes and gene fragments, and contains two items. The first item identifies the gene features for that locus. The second item contains an annotation detailing information regarding non-standard gene structures. This object is built by the ffN() function.

Usage

```
data(fragmentFeatureNames)
```

Format

A list of 15 elements that identify the gene features for each locus wih a genomic alignment and annotate non-standard features

Source

https://hla.alleles.org/genes/index.html

https://github.com/ANHIG/IMGTHLA/tree/Latest/alignments

getAlignmentNames *Retrieve Alignment Filenames for HLA Genes*

Description

Retrieves the filenames of the protein, nucleotide and genomic alignments available a specific branch of the IMGTHLA GitHub Repository

Usage

```
getAlignmentNames(URL)
```

Arguments

URL	A chararacter string containing a Uniform Resource Locator (URL) identifying of the desired IPD-IMGT/HLA Database release version from which the alignment filenames should be retrieved.

Value

A character vector of all of the filenames.

Note

For internal HLAtools use.

getField *Trim Colon-Delimited HLA Allele Names by Field*

Description

Trims a properly formatted colon-delimited HLA allele name to a desired number of fields.

If an allele name with an expression-variant suffix is truncated, the suffix can be appended to the end of the truncated allele name. If a resolution value greater then the number of fields in the submitted field is specified, the original allele is returned.

Usage

```
getField(allele, res, append = FALSE)
```

Arguments

allele	A character string representing an HLA allele.
res	A numeric describing the resolution desired.
append	A logical. When append = TRUE, the expression variant suffix of a full-length allele name is appended to a truncated allele name. The default value is FALSE.

Value

A version of the 'allele' character string that has been trimmed to 'res' resolution.

Note

For internal HLAtools use.

Examples

```
getField("HLA-A*01:01:01:01", 3)
getField("DRB1*11:01:01:12N", 2,TRUE)
```

`getLatestVersion` *Identify the Latest IPD-IMGT/HLA Database Release*

Description

Identifies the most recent version of the IPD-IMGT/HLA Database available in the ANHIG/IMGTHLA Github repository.

Usage

```
getLatestVersion()
```

Value

A dot-delimited character string identifying the latest release version (branch) of the ANHIG/IMGTHLA Github repository.

Note

For internal HLAtools use.

Examples

```
getLatestVersion()
```

GIANT *GLupdate-Integrated Allele Name Translation*

Description

Calls GLupdate() to translate the HLA allele names in a vector or data frame between two IPD-IMGT/HLA Database release versions.

Usage

```
GIANT(data, transFrom, transTo)
```

Arguments

data	A data frame or vector of HLA allele name character string data. Allele names in vectors must include locus prefixes. Data frames must include column headers identifying a single locus for the allele names in that column.
transFrom	A dot-formatted character string identifying the IPD-IMGT/HLA Database version (e.g. 3.58.0) under which the HLA data were generated.
transTo	A dot-formatted character string identifying the IPD-IMGT/HLA Database version (e.g. 2.25.0) to which which the HLA data are to be translated.

Value

A data frame or vector of HLA allele name data in the 'transTo' release.

Note

GIANT() will ignore the first two columns of BIGDAWG-formatted data frames, which include non-HLA data. All columns in non-BIGDAWG-formatted data frames are expected to contain HLA allele names.

Examples

```
GIANT(c("A*01:01:01:01","DOA*01:01:01:01"),"3.56.0","2.20.0")
updsHLAdata <- GIANT(sHLAdata,"3.56.0","2.25.0")
```

GLSC.ex *Example Data Frame of Genotype List String Code Data*

Description

A two-column data frame (modified from pould::hla.hap.demo) including GL String data in Gl String Code format. Column one identifies each subject's status, while column two identifies each subject's HLA genotype in GL String Code format. This data is provided for use in examples and demonstrations.

Usage

```
data(GLSC.ex)
```

Format

A dataframe with 419 rows and 2 columns.

Source

pould::hla.hap.demo

References

Mack et al. HLA 2023 Oct;102(4):501-507 https://doi.org/10.1111/tan.15145

GLStoUNI *Translate GL String to UNIFORMAT*

Description

A wrapper function for GLtoUN, which translates strings from GL String format to UNIFORMAT format.

Usage

```
GLStoUNI(GLstring, prefix = "HLA-", pre = FALSE)
```

Arguments

GLstring	A character string of HLA allele names and operators in the GL String format signifying their relation with one another.
prefix	A character string of the pertinent gene-system prefix (default is "HLA-").
pre	A logical that indicates whether returned allele names should contain 'prefix' (TRUE), or if 'prefix' should be excluded from returned names (FALSE).

Value

A version of 'GLstring' converted to UNIFORMAT format, or FALSE if 'GLstring' is invalid.

Note

This function does not function with the "?" operator, as the "?" operator has no cognate in UNIFORMAT.

References

Nunes Tissue Antigens 2007;69(s1):203-205 https://doi.org/10.1111/j.1399-0039.2006.00808.x

Mack et al. HLA 2023;102(2):206-212 https://doi.org/10.1111/tan.15126

Examples

```
GLStoUNI("HLA-A*02:01/HLA-A*02:02+HLA-A*03:01/HLA-A*03:02")
GLStoUNI("A+B/C~D^G|E^W+X/Y^Z+J")
```

GLstring.ex *Example Data Frame of Genotype List String Data.*

Description

A two-column example data frame (from pould::hla.hap.demo) including GL String Data. Column one identifies each subject's status, while column two identifies each subject's HLA genotype in GL String format. This data is provided for use in examples and demonstrations.

Usage

```
data(GLstring.ex)
```

Format

A dataframe with 419 rows and 2 columns.

Source

pould::hla.hap.demo

References

Mack et al. HLA 2023;102(2):206-212 https://doi.org/10.1111/tan.15126

GLtoUN *Translate GL Strings to UNIFORMAT Strings*

Description

A function that translates strings from GL String format to UNIFORMAT format.

Usage

```
GLtoUN(GLstring, prefix = "HLA-", pre = FALSE)
```

Arguments

GLstring	A character string of HLA allele names and operators in the GL String format signifying their relation with one another.
prefix	A character string of the desired gene-system prefix (default is "HLA-").
pre	A logical that indicates whether returned allele names should contain 'prefix' (TRUE), or if 'prefix' should be excluded from returned names (FALSE).

Value

A version of the input string converted to UNIFORMAT format.

Note

For internal use only.

This function does not function with the "?" operator, as the "?" operator has no cognate in UNIFORMAT.

References

Nunes Tissue Antigens 2007;69(s1):203-205 https://doi.org/10.1111/j.1399-0039.2006.00808.x

Mack et al. HLA 2023;102(2):206-212 https://doi.org/10.1111/tan.15126

Examples

```
GLtoUN("HLA-A*02:01/HLA-A*02:02+HLA-A*03:01/HLA-A*03:02")
GLtoUN("A+B/C~D^G|E^W+X/Y^Z+J")
```

GLupdate *Update a GL String Code.*

Description

Updates the allele names in a Genotype List (GL) String Code to the desired reference database version using the IPD-IMGT/HLA Database's Allele List History table.

Usage

```
GLupdate(GLString, Version, expand = FALSE)
```

Arguments

GLString	A character string of HLA allele names and operators in GL String Code format signifying their relation with one another and the pertinent HLA Allele List version.
Version	A character string of the desired version to which the alleles to be updated, going back to version 1.05.0.
expand	A logical that determines whether to return only the direct HLA ID allele match or all possible HLA allele matches.

Value

An updated version the GL String Code input (in the form of a character string) updated to the input desired version.

Note

For internal use only.

Source

https://github.com/ANHIG/IMGTHLA/blob/Latest/Allelelist_history.txt

References

Mack et al. HLA 2023;102(2):206-212 https://doi.org/10.1111/tan.15126

Mack et al. HLA 05 July 2023 https://doi.org/10.1111/tan.15145

Examples

```
GLupdate("hla#3.36.0#HLA-B*15:35", "3.52.0")
GLupdate("hla#3.45.0#HLA-A*02:08", "3.52.0")
GLupdate("hla#3.45.0#HLA-A*02:08", "3.52.0", expand = TRUE)

GLupdate("hla#3.1.0#HLA-A*02:01+HLA-A*01:01:01:01", Version = "3.53.0")
GLupdate("hla#3.44.0#HLA-DPA1*02:01:01:05", "3.45.0")
```

```
GLupdate("hla#3.45.0#HLA-DPA1*02:01:01:05", "3.46.0")
GLupdate("hla#3.37.0#HLA-A*01:02", "3.52.0", expand = TRUE)
```

GLV *Retrieve version from input GL String.*

Description

Extracts the version data from an input GL String Code, or provides appropriate options if version input is not present in the IPD-IMGT/HLA Database.

Usage

```
GLV(GLString)
```

Arguments

GLString	A character string of HLA allele names and operators in GL String Code format, signifying their relation with one another and the pertinent IPD-IMGT/HLA Database release version.

Value

Returns A character string of the version of the release as it appears in the pertinent alleleListHistory$AlleleListHistory-column-header.

Note

For internal use only.

References

Mack et al. HLA 2023;102(2):206-212 https://doi.org/10.1111/tan.15126

Mack et al. HLA 05 July 2023 https://doi.org/10.1111/tan.15145

Examples

```
GLV("hla#3.25.0#HLA-B15:35")
```

GLV2 *Format GL String Code version number.*

Description

Returns a compressed IPD-IMGT/HLA Database release version or a vector of dot-delimited release version options if the specified version is not present in the alleleListHistory object.

Usage

```
GLV2(Version)
```

Arguments

Version	A character string of the desired IPD-IMGT/HLA Database version, going back to version 1.05.0.

Value

Returns either a single, compressed character string-formatted release version, or a character vector of potential release versions.

Note

For internal use only.

References

Mack et al. HLA 2023;102(2):206-212 https://doi.org/10.1111/tan.15126

Mack et al. HLA 05 July 2023 https://doi.org/10.1111/tan.15145

Examples

```
GLV2("3.34.0")
GLV2("3.0.0")
```

`GLvalidate` *Validates a GL String Code.*

Description

A lightweight validator that inspects a GL String Code for correct structure. If the namespace field contains a value other than value of 'namespace', the namespace is changed to 'hla'. The version and GL String fields are not evaluated.

Usage

```
GLvalidate(GLString, namespace = c("hla", "kir"))
```

Arguments

`GLString`	A character string describing a string formatted to GL String Code specifications.
`namespace`	A character vector of allowed namespace strings. The default value is 'c("hla","kir")'.

Value

A character string describing wither a well-formatted GL String Code or the value FALSE.

Note

For internal use only.

References

Mack et al. HLA 2023;102(2):206-212 https://doi.org/10.1111/tan.15126

Mack et al. HLA 05 July 2023 https://doi.org/10.1111/tan.15145

Examples

```
GLvalidate("ha#3.25.0#hla-B15:35")
```

GLVhelper *Locate matches for an incomplete IPD-IMGT/HLA Database version.*

Description

Uses the provided version information to locate and return possible matches for an incompletely defined IPD-IMGT/HLA Database release version.

Usage

```
GLVhelper(Version)
```

Arguments

Version	A character string of the desired version to which the alleles should be updated, going back to version 1.05.0.

Value

A list of character strings of possible matches to a given incomplete input.

Note

For internal use only.

References

Mack et al. HLA 2023;102(2):206-212 https://doi.org/10.1111/tan.15126

Mack et al. HLA 05 July 2023 https://doi.org/10.1111/tan.15145

Examples

```
GLVhelper("2.25")
GLVhelper("3.9.0")
```

HLAatlas — *Boundary Positions of Exons, Introns and UTRs in Amino Acid, cDNA and gDNA Alignments*

Description

A list object of sub-lists of R dataframes (atlases) for each locus with a protein (prot), cDNA (nuc), and gDNA (gen) alignment. Each atlas identifies the position of the exon (E), intron (I) or UTR (U) gene-feature boundaries in an alignment. Boundaries for non-standard hybrid (H), join (J), novel (N) and segment (S) features may be included in gene fragment and pseudogene atlases. This object is built by the atlasFull() function.

Usage

```
data(HLAatlas)
```

Format

A list of 4 elements that include the gene features boundaries in each ANHIG/IMGTHLA sequence alignment.

- (prot: peptide-alignment atlases)
- (nuc: cDNA-alignment atlases)
- (gen: gDNA-alignment atlases)
- (version: The IPD-IMGT/HLA Database release version under which these data were generated)

Source

https://github.com/ANHIG/IMGTHLA/tree/Latest/alignments

HLAgazeteer — *Functional and Organizational Categories of Genes Supported by the IPD-IMGT/HLA Database*

Description

A list object of nineteen vectors that identify genes in the HLA region that share specific commonalities. This object is built by the buildGazeteer() function.

Usage

```
data(HLAgazeteer)
```

Format

A large list of 19 vectors that define specific categories of genes supported by the IPD-IMGT/HLA Database

.

- (align: all genes with alignments in the IPD/IMGT-HLA GitHub Repository)
- (gen: genes with genomic alignments in the IPD/IMGT-HLA GitHub Repository)
- (nuc: genes with nucleotide alignments in the IPD/IMGT-HLA GitHub Repository)
- (prot: genes with protein alignments in the IPD/IMGT-HLA GitHub Repository)
- (nogen: genes with no genomic alignments)
- (nonuc: genes with no nucleotide alignments)
- (noprot: genes with no protein alignments)
- (pseudo: pseudogenes)
- (frag: gene fragments)
- (hla: HLA genes)
- (expressed: genes that are expressed)
- (notexpressed: genes that are not expressed)
- (classireg: genes found in the HLA class I region)
- (classihla: class I HLA genes)
- (classiireg: genes found in the HLA class II region)
- (classiihla: class II HLA genes)
- (classical: classical HLA genes)
- (nonclassical: non-classical HLA genes)
- (map: all genes organized by 5' to 3' map order on the genomic reference + strand)
- (version: IPD-IMGT/HLA Database version used to build the HLAgazeteer)

Source

https://hla.alleles.org/genes

https://github.com/ANHIG/IMGTHLA/tree/Latest/alignments

Andersson Front. Biosci. 1998, 3(4), 739–745. https://doi.org/10.2741/a317

Alexandrov et al. HLA 2023, Vol.102(5), p.599-606. https://doi.org/10.1111/tan.15180

https://www.ncbi.nlm.nih.gov/nucleotide/NC_000009.12

IMGTHLAGeneTypes — *Molecular characteristics of the Genes Curated by the IPD-IMGT/HLA Database*

Description

A data frame of three columns identifying each gene supported by the IPD-IMGT/HLA Database, its molecular characteristics, and its status as a gene fragment or pseudogene. This object is built by the BuildIMGTHLAGeneTypes() function.

Usage

```
data(IMGTHLAGeneTypes)
```

Format

A list object of two elements:

- (GeneTypes: a data frame of three columns)
- (version: a character string identifying the date that the source file was written)

Source

https://hla.alleles.org/genes/index.html

motifMatch — *Identify Alleles that Share a Sequence Motif*

Description

Searches alignments for alleles that share a specific sequence motif.

Usage

```
motifMatch(motif, alignType, full = TRUE)
```

Arguments

motif	A character string identifying a sequence variant motif in the following format: Locus*#$~#$~#$, where ## identifies a variant position, and $ identifies the sequence variant. Both nucleotide and peptide motifs can be provided, and any number of variants can be specified.
alignType	A character string identifying the type of alignment being searched. Allowed values are "codon","gen", nuc" and "prot". Only one 'alignType' value is allowed.
full	A logical value that specifies if full (TRUE) or truncated (FALSE) allele names should be returned.

Value

A character vector of allele names that contain the motif, or NA when no alleles contain the motif, NULL when no alignment is available for specified locus, and FALSE when the locus or motif is invalid.

Note

This function requires an HLAalignments object that has been populated with alignments via alignmentFull().

---

multiAlign *Generate an Alignment for Specific Alleles at Different Positions*

Description

Generates a peptide, codon, coding nucleotide or genomic alignment for HLA alleles allowing each allele to be aligned to a different set of positions.

Usage

```
multiAlign(alignType, alleles, positions)
```

Arguments

alignType	The type of alignment being searched. Allowed values are "prot", codon", "nuc" and "gen". Only one 'alignType' value is allowed.
alleles	A vector of un-prefixed HLA locus names.
positions	A list of vectors of nucleotide positions, exactly one vector for each allele, against which each corresponding allele will be aligned.

Value

A data frame of 'allele' and the corresponding nucleotide sequence for specified positions designated for an allele. a nucleotide sequence is not returned for a specific allele if input allele is not available in the ANHIG/IMGTHLA Github Repository. position will be left empty if specific allele does not have a nucleotide at the input position.

Note

For internal HLAtools use.

`multiAlleleTrim` *Trim Multiple HLA Allele Names*

Description

Trim a vector of allele names to a specified number of fields or digits.

Usage

```
multiAlleleTrim(alleles, resolution, version = 3, append = FALSE)
```

Arguments

`alleles`	A vector of HLA allele names from a single nomenclature epoch.
`resolution`	A number identifying the number of fields to which the alleles should be trimmed.
`version`	A number identifying the HLA nomenclature epoch under which the allele was named. Epoch 1 allele names are found in IPD-IMGT/HLA Database releases 1.0.0 to 1.16.0. Epoch 2 allele names are found in IPD-IMGT/HLA Database releases 2.0.0 to 2.28.0. Epoch 3 allele names are found in IPD_IMGT/HLA Database releases 3.0.0 and onward.
`append`	A logical. When append = TRUE, the expression variant suffix of a full-length allele name is appended to a truncated allele name. The default value is FALSE.

Value

A vector of the trimmed allele name, shortened according to the input parameters.

Note

Expression variant suffixes will not be removed from full-length allele names that have fewer than four fields or eight digits.

Examples

```
alleles <- c("A*02:01:01:02L","DRB1*08:07", "DQB1*04:02:01:16Q")
multiAlleleTrim(alleles,2)
multiAlleleTrim(alleles,2,append = TRUE)

alleles <- c("A*01010102L","DRB1*1613N","HLA-Cw*0322Q")
multiAlleleTrim(alleles,1,2,TRUE)
multiAlleleTrim(alleles,2,2)
```

multiGLStoUNI *Translate Multiple GL Strings to UNIFORMAT*

Description

Translate a data frame or a vector of GL Strings to UNIFORMAT strings.

Usage

```
multiGLStoUNI(GLstringArray, prefix = "HLA-", pre = FALSE)
```

Arguments

GLstringArray	A data frame or a vector containing GL String formatted data. If 'GLstringArray' is a data frame with more than one column, the first column should contain only identifiers. If 'GLstringArray' is a vector, it should contain only GL Strings.
prefix	A string of the desired locus prefix (default is "HLA-").
pre	A logical. If 'pre' is TRUE, all allele names will be prefixed with 'prefix'. If 'pre' is FALSE, no allele names will be prefixed.

Value

A version of 'GLStringArray' in which the GL String data has been converted to UNIFORMAT format. If a 'GLstringArray' was a data frame, a data frame is returned. If 'GLstringArray' was a vector, a vector is returned.

Note

GL Strings that include the "?" operator will not be translated, as the "?" operator has no cognate in UNIFORMAT.

References

Nunes Tissue Antigens 2007;69(s1):203-205 https://doi.org/10.1111/j.1399-0039.2006.00808.x

Mack et al. HLA 2023;102(2):206-212 https://doi.org/10.1111/tan.15126

Examples

```
multiGLStoUNI(GLstring.ex[[2]][1:5],version) ## converting a vector of GL Strings to UNIFORMAT
multiGLStoUNI(GLstring.ex[1:5,1:2])  ## converting a data.frame of GL Strings to UNIFORMAT
```

multiLocusValidation *Apply validateLocus() to Multiple Loci*

Description

Applies validateLocus() to a vector of locus names, validates them against HLAgazeteer$gen, and returns a vector of validated locus names.

Usage

```
multiLocusValidation(loci, source = "gDNA", verbose = TRUE)
```

Arguments

loci	A character vector of locus names found in the current HLAgazeteer.
source	A character vector describing the type of alignment 'loci' should be validated against. Allowed options are 'AA' (protein), 'cDNA' (nucleotide) and 'gDNA' (genomic). Only a single value should be provided. The default value is 'gen'.
verbose	A logical value. If 'verbose' = TRUE, messages describing invalid 'loci' or 'source' values are generated. If 'verbose' = FALSE, no messages are generated.

Value

A character vector of locus names that are present in HLAgazeteer$gen.

Note

The results of this check should only be considered valid for the IPD-IMGT/HLA Database release version of the current HLAgazeteer.

Examples

```
multiLocusValidation(loci = c("DRB1","DPB1","DQB8"))
multiLocusValidation(loci = c("A","B","C","D","Q"))
```

multiQueryRelease *Search Multiple Elements of Allele Names Across Release Versions*

Description

Searches specific release versions in the AlleleListHistory object for a set of user-defined allele-name elements.

Usage

```
multiQueryRelease(rel, variants = "", all = TRUE)
```

Arguments

rel	An IPD-IMGT/HLA Database release version, represented as either a character (e.g., "3.58.0") or a numeric (e.g., 3580) value.
variants	A vector of character strings. The values in 'variants' can be any part of a locus or allele name (e.g., "DR", "02:01", "DRB1*08:07"). The default ("") specifies all alleles in 'rel'.
all	A logical. When 'all' = TRUE, a vector of all alleles in 'rel' that share all elements of 'variants' is returned. When 'all' = FALSE, the number of alleles that share all elements of 'variants' in 'rel' is returned.

Value

A character vector of all alleles that share all 'variants' in 'rel' or the number of alleles that share all 'variants' in 'rel'.

Examples

```
# Identify DRB1 allele names that include a 'Q' character in release 3.58.0.
multiQueryRelease("3.58.0",c("DRB1","Q"),TRUE)
# Identify the number of null (N) DRB1 alleles.
multiQueryRelease("3.58.0",c("DRB1","N"),FALSE)
```

---

multiSearch *Search Alignment Sequences at Multiple Positions for a Specified Allele*

Description

Generates a character string of multiple protein, codon or nucleotide or genomic sequences at the specified positions for the specified allele name.

Usage

```
multiSearch(
  alignType,
  locus,
  allele,
  positions,
  prefix = TRUE,
  sep = "~",
  trimmed = FALSE
)
```

Arguments

alignType	The type of alignment being searched. Allowed values are "prot", codon", "nuc" and "gen". Only one 'alignType' value is allowed.
locus	A specific locus.
allele	An allele name.
positions	Nucleotide positions to search.
prefix	A logical that indicates if the position number(s) should be included in the result.
sep	Defines the separator between position sequences.
trimmed	A logical identifying whether 'allele' is a two-field, 'trimmed' allele name (trimmed=TRUE), or a full-length name (trimmed=FALSE). The default value is FALSE.

Value

codons (possibly with position number) at 'positions' for 'allele'. If 'allele' does not have a codon at 'position', '' will be returned.

Note

For internal HLAtools use only.

---

multiUNItoGLS *Translate Multiple UNIFORMAT Strings to GL Strings*

Description

Translate a data frame or vector of UNIFORMAT strings to GL Strings.

Usage

```
multiUNItoGLS(UniformatArray, prefix = "HLA-", pre = TRUE)
```

Arguments

UniformatArray	A data frame or vector of UNIFORMAT formatted strings. If 'UniformatArray' is a data frame with more than one column, the first column should contain only identifiers. If 'UniformatgArray' is a vector, it should contain only UNIFORMAT strings.
prefix	A character string of the desired locus prefix (default is "HLA-").
pre	A logical. If 'pre' is TRUE, all allele names will be prefixed with 'prefix'. If 'pre' is FALSE, no allele names will be prefixed.

Value

A version of 'UniformatArray' in which the UNIFORMAT data have been converted to GL String format. If a 'UniformatArray' was a data frame, a data frame is returned. If 'UniformatArray' was a vector, a vector is returned.

Note

This function does not return the GL String "?" operator, as the "?" operator has no cognate in UNIFORMAT.

References

Nunes Tissue Antigens 2007;69(s1):203-205 https://doi.org/10.1111/j.1399-0039.2006.00808.x

Mack et al. HLA 2023;102(2):206-212 https://doi.org/10.1111/tan.15126

Examples

```
multiUNItoGLS(unname(as.vector(UNIFORMAT.example[2]))[[1]])
multiUNItoGLS(UNIFORMAT.example[1:2])
```

`multiUpdateGL` *Update a column of GL String Code data to a desired IPD-IMGT/HLA Database version.*

Description

Updates a column from a data frame in GL String Code format to a desired reference database version.

Usage

```
multiUpdateGL(GLstringArray, Version, expand = FALSE)
```

Arguments

`GLstringArray`	An array of HLA allele names and operators in GL String code format identifying their relation with one another and the pertinent IPD-IMGT/HLA Database release version.
`Version`	A character string identifying the desired version to which the alleles should be updated, going back to version 1.05.0.
`expand`	A logical to determine whether to include only the direct HLA ID match, or all possible allele matches.

Value

A version of the input array of GL String Codes (in a data frame) updated to the desired version. NA values are returned in place of alleles that are not present in Version.

References

Mack et al. HLA 2023;102(2):206-212 https://doi.org/10.1111/tan.15126

Mack et al. HLA 05 July 2023 https://doi.org/10.1111/tan.15145

Examples

```
# Example update of two GL Strings Codes containing truncated alleles from version 3.1.0 to 3.53.0.
GLSC.ex[[2]][1:2]
multiUpdateGL(GLSC.ex[[2]][1:2], Version = "3.53.0")
```

`numFields` *Identify the Number of Fields in a Colon-Delimited Allele Name*

Description

Returns the number of fields in a colon-delimited HLA allele name. A value of 1 is returned for digit-delimited HLA allele names.

Usage

```
numFields(allele)
```

Arguments

`allele`	A character string of a colon-delimited HLA allele name.

Value

A numeric value describing the number of fields in the allele name.

Examples

```
numFields("HLA-A*01:01")
numFields("DRB1*04:03:01")
numFields("13:02:01:01")
```

`parseAlignmentHead` *Guides For Parsing the Header Blocks of Alignment Files*

Description

Returns a vector describing the location of key information in the header blocks of alignment files.

Usage

```
parseAlignmentHead(version)
```

Arguments

version	A character string of a validated IPD-IMGT/HLA Database release version (e.g., '3.25.0' or '3250').

Value

Either FALSE if the version is not valid, or two-value numerical vector describing (1) the header line in which alignment version data appears and (2) the length of the character string in that line preceding version data.

Examples

```
parseAlignmentHead("3.25.0")
```

---

posSort *Numerical Sort of Alignment Positions that Contain Indels*

Description

Sorts sequence alignment positions that contain indels in numerical order; e.g., three indels following position X are are identified as X.1, X.2 and X.3.

Positions are validated against a specified locus for a specified alignment type. Positions that do not exist for that locus-alignment combination are not returned.

Usage

```
posSort(posVec, alignType, locus)
```

Arguments

posVec	A character vector of variant positions.
alignType	A character string describing the type of alignment the positions are found in. The values 'prot', 'codon', 'nuc' and 'gen' are valid options. Only one 'alignType' value is allowed.
locus	A character string describing a locus in the HLAalignments object for the specified alignType.

Value

A character string of the correctly sorted sequence.

Note

Indel positions must be text-formatted (e.g. "607.12"). C.f., posSort(c(2,4,3,1,5), "nuc","DRB1") vs posSort(c("607.23","607.10","607.3","607.4"),"nuc","DRB1").

pypopHeaders *Convert BIGDAWG File Headers to PyPop Format*

Description

Convert the header of a BIGDAWG-formatted data frame into a Python for Population Genomics (PyPop) formatted header.

Usage

```
pypopHeaders(colHead)
```

Arguments

colHead	A vector of column names. BIGDAWG format requires that the first two header values are the sample identifier character string and the subject status (0 or 1). All other header values should be paired locus/gene names.

Value

A PyPop-formatted column header vector.

Note

This function returns unique locus names for each column. For example, "A_1" and "A_2" headers will be returned for two "A" locus columns. PyPop configuration (.ini) files allow this format to be customized.

References

Lancaster et al. Front Immunol. 2024 Apr 2;15:1378512. https://pubmed.ncbi.nlm.nih.gov/38629078/

Pappas et al. Hum Immunol. 2016 Mar 77(3):283-287. https://pubmed.ncbi.nlm.nih.gov/26708359/

See Also

PyPop Configuration File

Examples

```
pyHead <- pypopHeaders(colnames(sHLAdata))
```

queryRelease *Search Allele Names Across Release Versions*

Description

Searches specific release versions in the AlleleListHistory object for user-defined allele variants.

Usage

```
queryRelease(rel, variant = "", all = FALSE)
```

Arguments

rel	An IPD-IMGT/HLA Database release version, represented as either a character (e.g., "3.56.0") or a numeric (e.g., 3560) value.
variant	A character string. The value of 'variant' can be any part of a locus or allele name (e.g., "DR", "02:01", "DRB1*08:07"). The default ("") specifies all alleles in 'rel'.
all	A logical. When 'all' = TRUE, a vector of all instances of 'variant' in 'rel' is returned. When 'all' = FALSE, the number of instances of 'var' in 'rel' is returned.

Value

A character vector of all matches to 'variant' in 'rel' or the number of all matches to 'variant' in 'rel'.

Examples

```
# Identify the number of DRB9 alleles in releases 3.30.0 and 3.31.0.
queryRelease("3.30.0","DRB9",FALSE)
queryRelease("3.31.0","DRB9",FALSE)

# Identify the total number of alleles in release 3.56.0.
queryRelease(3560)
```

redec *Reintroduce version decimals.*

Description

Correctly put decimals back into an AlleleListHistory column-formatted version name.

Usage

```
redec(Cname)
```

Arguments

Cname	A character string describing a version name in AlleleListHistory column format.

Value

A character string describing a version name with decimals in appropriate places according to the IPD-IMGT/HLA Database.

Note

For internal use only.

Examples

```
redec("X3090")
```

---

relRisk — *Calculate Relative Risk for Individual Alleles and Genotypes in BIGDAWG-formatted Non-Case-Control Datasets*

Description

This function returns a list object containing relative risk, confidence interval and p-value data for the individual alleles and individual genotypes at each locus in a BIGDAWG-formatted non-case-control genotype data frame or file.

Usage

```
relRisk(dataset, return = TRUE, save.path = tempdir())
```

Arguments

dataset	A character string describing the name of a non-case-control genotype dataset using the BIGDAWG format. Here, "non-case-control" means that while two subject categories are required, the categories should not be patients and controls; instead, the categories may be, e.g., for a dataset of patients, either of two disease states, where one disease state is coded as 0 and the other is coded as 1 in the second column of the dataset. Either a tab-delimited file or a data frame can be specified.
return	A logical identifying if the list object should be returned (return=TRUE), or if pairs of tab-delimited text files of results (one for alleles and one for genotypes) should be written to the working directory for each locus.
save.path	A character string identifying the path in which to write the pair of files when return is FALSE. The default value is tempdir().

Value

A list object of two lists ("alleles" and "genotypes"), each of which contains a list of nine-column data frames containing results for each unique allele or genotype (in rows) at each locus. Column headers in each dataframe are, *Locus*, *Variant*, *Status_1*, *Status_0*, *RelativeRisk*, *CI.low*, *CI.high*, *p.value*, and *Significant*.

References

BIGDAWG Data Format

Examples

```
rr <- relRisk(sHLAdata)
```

---

repoVersion *Convert an AlleleListHistory Release Version to the GitHub Repository Version*

Description

Removes periods from IMGT/HLA Database release versions, and converts release versions '3.00.0'-'3.09.0' to '300'-'390' to facilitate working with these GitHub repo branches.

Usage

```
repoVersion(version)
```

Arguments

`version`	A character string of a validated IPD-IMGT/HLA Database release version.

Value

A character string of the release version suitable for use in an IMGT/HLA GitHub repository URL.

Note

For internal HLAtools use.

Examples

```
repoVersion("3.05.0")
```

sHLAdata *Synthetic HLA Data for use with Package Examples*

Description

A BIGDAWG-formatted data frame of 18 columns and 47 rows containing synthetic HLA-A, -C, -B, -DRB1, -DQA1, -DQB1, -DPA1, and -DPB1 genotype data for 24 control subjects and 23 case subjects. Allele name data were recorded under IPD-IMGT/HLA Database release version 3.56.0. These are synthetic data generated for the imaginary UchiTelle population, and do not represent biologically-derived HLA genotypes.

Usage

```
data(sHLAdata)
```

Format

A data frame of 18 columns and 47 rows.

Source

These synthetic data were generated as part of the 13th International HLA Workshop to demonstrate PyPop functions. See http://pypop.org/docs/guide-chapter-usage.html#data-minimal-noheader for additional details.

squashVersion *Reduce a Release Version to Numerals*

Description

Removes the '.' delimiters from an IPD-IMGT/HLA Database Release Version name

Usage

```
squashVersion(ver, num = FALSE)
```

Arguments

ver	A character vector-formatted IPD-IMGT/HLA Database Release Version name (e.g., '3.54.0').
num	A Logial. A numeric value is returned when num = TRUE. A character vector is returned when num = FALSE. The default value is FALSE.

Value

A 4-digit value, as either a character string or a number.

Note

For internal HLAtools use.

Examples

```
squashVersion("3.45.0",TRUE)
```

---

typeToSource *Convert AlignType Values to Source Values*

Description

Converts between 'alignType' values, identifying four types of sequence alignments and 'source' values, identifying three kinds of alignment files.

Usage

```
typeToSource(alignVector, toSource = TRUE)
```

Arguments

alignVector	A vector of character values specifying kinds of four sequence alignment types ("prot","nuc","codon" and "gen"), or three source alignment files ("AA","cDNA", and "gDNA").
toSource	A logical. If 'toSource' is true, 'alignType' values are converted to 'source' values. If 'toSource' is false, 'source' values are concerted to 'alignType' values

Value

A character string of the converted 'align' or 'source' value.

Examples

```
typeToSource(c("nuc","prot","gen"),TRUE)
```

uniAlign *Generate an Alignment for Specific Alleles at Specific Positions*

Description

Generates a peptide, codon, coding or genomic nucleotide alignment at a single set of positions for HLA alleles at one or more loci.

Usage

```
uniAlign(alignType, alleles, positions)
```

Arguments

alignType	The type of alignment being searched. Allowed values are "prot", codon", "nuc" and "gen". Only one 'alignType' value is allowed.
alleles	A vector of un-prefixed HLA locus names.
positions	A vector of codon positions, against which all loci will be aligned.

Value

A data frame of allele names and the corresponding codon sequence for specified position. a codon sequence is not returned for a specific allele if input allele is not available in the ANHIG/IMGTHLA Github Repository. position will be left empty if specific allele does not have a codon at the input position.

Note

For internal HLAtools use.

UNIFORMAT.example *Example Data Frame of UNIFORMAT Data.*

Description

A two-column example data frame including UNIFORMAT data. Column one contains sample identifiers, while column two identifies each subject's HLA genotype in UNIFORMAT format. To be used for example runs and demonstrations.

Usage

```
data(UNIFORMAT.example)
```

Format

A dataframe with 20 rows and 2 columns.

Source

https://hla-net.eu/wp/wp-content/uploads/example-three-loci.unif_.txt

References

Nunes Tissue Antigens 2007;69(s1):203-205 https://doi.org/10.1111/j.1399-0039.2006.00808.x

---

uniSearch *Search Sequences at a Single Position for an Allele*

Description

Generates a character string of the peptide, codon or nucleotide sequence at the specified position for the specified HLA allele.

Usage

```
uniSearch(alignType, locus, allele, position, prefix = TRUE, trimmed = FALSE)
```

Arguments

`alignType`	The type of alignment being searched. Allowed values are "prot", "codon", "nuc" and "gen". Only one 'alignType' value is allowed.
`locus`	A specific HLA locus.
`allele`	The name of the allele being searched.
`position`	The specified position.
`prefix`	A logical that indicates if the position number(s) should be included in the result.
`trimmed`	A logical identifying whether 'allele' is a two-field, 'trimmed' allele name (trimmed=TRUE), or a full-length name (trimmed=FALSE). The default value is FALSE.

Value

The peptide residue, codon or nucleotide sequence at specified position for specified allele as available in the ANHIG/IMGTHLA Github Repository.

Note

For internal HLAtools use.

UNItoGLS *Translate UNIFORMAT to GL String*

Description

A wrapper function for UNtoGL() which translates strings from UNIFORMAT format to GL String format.

Usage

```
UNItoGLS(uniformat, prefix = "HLA-", pre = TRUE)
```

Arguments

uniformat	A character string of HLA allele names and operators in the UNIFORMAT format signifying their relation with one another.
prefix	A character string of the desired gene-system prefix (default is "HLA-").
pre	A logical that indicates whether returned allele names should contain 'prefix' (TRUE), or if 'prefix' should be excluded from returned names (FALSE).

Value

An version of 'uniformat' converted to GL String format, or FALSE if 'uniformat' is invalid.

Note

This function does not return the "?" operator, as the "?" operator has no cognate in UNIFORMAT.

References

Nunes Tissue Antigens 2007;69(s1):203-205 https://doi.org/10.1111/j.1399-0039.2006.00808.x

Mack et al. HLA 2023;102(2):206-212 https://doi.org/10.1111/tan.15126

Examples

```
UNItoGLS("A*02:01,A*03:01|A*02:01,A*03:02|A*02:02,A*03:01|A*02:02,A*03:02")
UNItoGLS("A,B|A,C~D G|E W,X|W,Y Z,J J,Z")
```

UNtoGL *Translate UNIFORMAT Strings to GL Strings*

Description

A function that translates strings from UNIFORMAT format to GL String format.

Usage

```
UNtoGL(uniformat, prefix = "HLA-", pre = TRUE)
```

Arguments

uniformat	A character string of HLA allele names and operators in the UNIFORMAT format signifying their relation with one another.
prefix	A character string of the desired prefix (default is "HLA-").
pre	A logical that indicates whether user would like all allele names to contain the prefix of their choice (TRUE), or if the prefix should not be appended to allele names (FALSE).

Value

An altered version of the input character string, converted to GL String format.

Note

For internal use only.

This function does not return the "?" operator, as the "?" operator has no cognate in UNIFORMAT.

References

Nunes Tissue Antigens 2007;69(s1):203-205 https://doi.org/10.1111/j.1399-0039.2006.00808.x

Mack et al. HLA 2023;102(2):206-212 https://doi.org/10.1111/tan.15126

Examples

```
UNtoGL("A*02:01,A*03:01|A*02:01,A*03:02|A*02:02,A*03:01|A*02:02,A*03:02")
UNtoGL("A,B|A,C|D,B|D,C|E,B|E,C", pre = FALSE)
```

`updateAll` *Update All Package Data Objects Derived from IPD-IMGT/HLA Database Resources*

Description

Applies updateAlleleListHistory(), atlasFull(), buildGazeteer(), extractGeneTypes(), and ffN() to update the alleleListHistory, HLAatlas, HLAgazeteer, IMGTHLAGeneTypes and fragmentFeatureNames data objects.

A new alleleListHistory data object should be generated with each IPD-IMGT/HLA Database release. The other data objects will likely only change when new genes are added to the IPD-IMGT/HLA Database.

Usage

```
updateAll(updateType = "all", version = getLatestVersion())
```

Arguments

`updateType`	A character vector of the names of data objects to be updated. By default, updateAll() builds all five data objects (updateType="all"). Alternatively specific data objects can be updated; e.g., updateType="alleleListHistory" or updateType=c("alleleListHistory","fragmentFeatureNames").
`version`	A numeric value or character string identifying the version of the ANHIG/IMGTHLA Github repository to build these objects from. By default, updateAll() calls the getLatestVersion() function to identify the most recent IPD-IMGT/HLA Database release.

Value

No value is returned. The desired versions of the specified data objects are built into the environment that called updateAll().

Note

Generating a new HLAatlas can take 5 minutes or more to complete.

`updateAlleleListHistory` *Build the AlleleList History R Object*

Description

Consumes the ANHIG/IMGTHLA GitHub repository's allelelisthistory.txt file to create the AlleleListHistory object.

Usage

```
updateAlleleListHistory()
```

Value

A list object containing two data frames. alleleListHistory$Version identifies the date and version of alleleListHistory$AlleleListHistory. alleleListHistory$AlleleListHistory relates each IPD-IMGT/HLA Database Accession ID (HLA_ID) to the corresponding HLA allele name in each release version since 1.05.0.

---

updateGL — *Update a GL String Code to a Specified IPD-IMGT/HLA Database Version.*

Description

A quality control wrapper for GLupdate, which updates a GL String Code to a desired reference database version.

Usage

```
updateGL(GLStringCode, Version, expand = FALSE)
```

Arguments

GLStringCode	A character string of HLA allele names and operators in GL String Code format, signifying their relation with one another and the associated IMGT/HLA Database release version.
Version	A character string identifying of the desired IPD-IMGT/HLA Database release version to which the alleles should be updated, going back to version 1.05.0
expand	A logical that indicates whether user would like to return all allele names that contain the input allele name (TRUE), or if only the direct HLA ID match should be returned (FALSE).

Value

A version of the input GL String code (in the form of a character string) updated to the desired version.

References

Mack et al. HLA 2023;102(2):206-212 https://doi.org/10.1111/tan.15126

Mack et al. HLA 05 July 2023 https://doi.org/10.1111/tan.15145

Examples

```
updateGL("hla#1.05.0#HLA-DPA1*0106", "3.52.0")
updateGL("hla#3.36.0#HLA-B*15:35", "3.52.0")
updateGL("hla#3.45.0#HLA-DPA1*02:01:01:04", "3.52.0")

updateGL("hla#3.45.0#HLA-A*02:08", "3.52.0", expand = TRUE)
updateGL("hla#1.05.0#HLA-DPA1*0106", "3.52.0", expand = TRUE)
updateGL("hla#1.05.0#HLA-DPA1*0106", "2.27.0", expand = TRUE)
```

---

validateAllele *Validate Allele-Name Format and Presence in HLAalignments*

Description

Returns TRUE if an allele name is found in HLAalignments in either the 'allele_name' column of full-length allele names or the 'trimmed_allele' column of two-field allele names in the pertinent genomic alignment. Returns FALSE if the allele name is not properly formed, or if the allele name is not found in HLAalignments.

Usage

```
validateAllele(allele)
```

Arguments

allele	A character string of the colon-delimited HLA allele name.

Value

A logical identifying if the allele name is present in the alignments (TRUE) or, if it is not in the alignments or is not valid not (FALSE).

Note

Messages will be returned to the console if the allele name is malformed, or the locus is invalid; e.g., validateAllele("C*01:01:01:01") or validateAllele("A*01:01:01:01:01").

The locus being evaluated must be included in HLAalignments.

validateGLstring *Validate a GL String*

Description

Evaluates a GL String to identify unsupported characters for a specified GL String version.

Usage

```
validateGLstring(GLstring, version)
```

Arguments

GLstring	A character string of allele names and operators in the GL String format.
version	A character string identifying the GL String version for evaluation. Options are "1.0" and "1.1".

Value

A logical. TRUE is returned when all characters in a 'version' GL String are permitted. FALSE is returned when forbidden characters are present, or when an incorrect 'version' is specified.

References

Mack et al. HLA 2023;102(2):206-212 https://doi.org/10.1111/tan.15126

Examples

```
validateGLstring("HLA-A*02:01/HLA-A*02:02+HLA-A*03:01/HLA-A*03:02", version = "1.0")
```

validateLocus *Determine if a locus name is in the HLAgazeteer*

Description

Checks a vector of HLA locus names against the HLAgazeteer to determine if the locus name is valid for a specific type of alignment.

Usage

```
validateLocus(loci, source)
```

Arguments

loci	A character vector of HLA gene names (ex. "DRB1", c("DRB1","DQB1")).
source	A character vector of alignment source types. "AA", "cDNA", and "gDNA" are allowed types.

Value

A logical value. TRUE indicates that all of the names and source types are valid. FALSE indicates that at least one locus name or alignment source type is invalid.

Note

The results of this check should only be considered valid for the version of the HLAgazeteer included in the HLAtools package.

Examples

```
validateLocus(loci = "DRB1", source = "AA")
validateLocus(loci = c("V"), source = c("cDNA","gDNA"))
validateLocus(loci = c("E","F","G"), source = "gDNA")
```

---

validateMotif *Determine if a Motif is Properly Formatted*

Description

Evaluates a motif to determine if the locus is valid and variants are valid.

Usage

```
validateMotif(motif, alignType)
```

Arguments

motif	A character string identifying a sequence variant motif in the following format: Locus*#$~#$~#$, where ## identifies a variant position, and $ identifies the sequence variant. Both nucleotide and peptide motifs can be provided, and any number of variants can be specified.
alignType	A character string identifying the type of alignment being searched. Allowed values are "codon","gen", nuc" and "prot". Only one 'alignType' value is allowed.

Value

If the motif is valid, TRUE is returned. If the locus or body of the motif is invalid, FALSE is returned along with a brief message.

Examples

```
validateMotif("A*-21M~2P","prot")
validateMotif("A*196G~301A~3046T","gen")
```

---

validateUniformat *Validate a UNIFORMAT String*

Description

Evaluates a UNIFORMAT string to identify unsupported characters.

Usage

```
validateUniformat(uniformat)
```

Arguments

uniformat	A character string of allele names and operators in the UNIFORMAT format.

Value

A logical. TRUE is returned when all characters in 'uniformat' are permitted. FALSE is returned when forbidden characters are present.

References

Nunes Tissue Antigens 2007;69(s1):203-205 https://doi.org/10.1111/j.1399-0039.2006.00808.x

Examples

```
validateUniformat("A*02:01,A*03:01|A*02:01,A*03:02|A*02:02,A*03:01|A*02:02,A*03:02")
```

---

validateVersion *Validate an IPD-IMGT/HLA Release Version*

Description

Determines if an IPD-IMGT/HLA Release Version is referenced in the AlleleListHistory file.

Usage

```
validateVersion(version)
```

Arguments

version	A character string identifying the release version (branch) of the ANHIG/IMGTHLA Github repository (e.g. '3.53.0'). The value 'Latest' refers to the most recent release.

Value

A logical. TRUE indicates that data for 'version' is available in the ANHIG/IMGTHLA GitHub repository. FALSE means that data for 'version' is not available.

Note

For internal HLAtools use.

Examples

```
validateVersion(version = "3.31.1")
validateVersion(version = "3.13.0")
```

---

verifyAllele *Determine if an Allele Name Ever Existed, and (if so) its Most Recent IPD-IMGT/HLA Database Release*

Description

Returns TRUE if an allele name is present in AlleleListHistory or FALSE it is absent, or c(TRUE,version), where 'version' is the most recent IPD-IMGT/HLA Database release in which that name appeared, when version = TRUE.

Usage

```
verifyAllele(allele, version = FALSE)
```

Arguments

allele	A character string of an HLA allele name. Colon-delimited and field-delimited names are both accepted.
version	A logical that indicates if the most recent nomenclature release version in which that name was valid should be returned.

Value

A logical identifying if the allele name is found in AlleleListHistory (TRUE) or not (FALSE), or c(TRUE,version) if version = TRUE.

Examples

```
verifyAllele("A*01:01:01:01")
verifyAllele("A*01:01:01:01",TRUE)
verifyAllele("A*010101",TRUE)
verifyAllele("A*0101",TRUE)
```

Index