

Package ‘immunogenetr’

August 20, 2026

Type Package

Title A Comprehensive Toolkit for Clinical HLA Informatics

Version 1.5.0

Maintainer Nicholas Brown <nicholas.brown@penndmedicine.upenn.edu>

Description

A comprehensive toolkit for clinical Human Leukocyte Antigen (HLA) informatics, built on 'tidyverse' <<https://tidyverse.tidyverse.org/>> principles and making use of Genotype List String (GL String, Mack et al. (2023) <[doi:10.1111/tan.15126](https://doi.org/10.1111/tan.15126)>) for storing and computing HLA genotype data. Specific functionalities include: coercion of HLA data in tabular format to and from GL String; calculation of matching and mismatching in all directions, with multiple output formats; automatic formatting of HLA data for searching within a GL String; truncation of molecular HLA data to a specific number of fields; and reading HLA genotypes in HML files and extracting the GL String. This library is intended for research use. Any application making use of this package in a clinical setting will need to be independently validated according to local regulations.

License GPL (>= 3)

URL <https://immunogenetr.org>,
https://github.com/k96nb01/immunogenetr_package,
<https://doi.org/10.1016/j.humimm.2025.111619>

BugReports https://github.com/k96nb01/immunogenetr_package/issues

Depends R (>= 4.1.0)

Imports cli (>= 3.0.0), dplyr (>= 1.1.1), purrr (>= 1.0.2), rlang (>= 1.1.2), stringi (>= 1.7.0), stringr (>= 1.5.0), tibble (>= 3.1.3), tidyr (>= 1.3.0), tidyselect (>= 1.2.0), xml2 (>= 1.3.6)

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

Language en-US

LazyData true

Config/roxygen2/version 8.1.0

NeedsCompilation no

Author Nicholas Brown [cre, aut] (ORCID:

<<https://orcid.org/0000-0002-0046-2315>>),

Busra Coskun [aut] (ORCID: <<https://orcid.org/0009-0008-6828-3453>>)

Repository CRAN

Date/Publication 2026-08-19 22:10:08 UTC

Contents

ambiguity_table_to_GLstring	3
GLstring_drop_non_expressed	4
GLstring_expand_longer	5
GLstring_genes	6
GLstring_genes_expanded	7
GLstring_gene_copies_combine	7
GLstring_genotype_ambiguity	8
GLstring_regex	9
GLstring_to_ambiguity_table	11
Haplotype_frequencies	11
HLA_columns_to_GLstring	12
HLA_column_repair	14
HLA_dictionary	14
HLA_match_number	15
HLA_match_summary_HCT	16
HLA_mismatched_alleles	17
HLA_mismatch_alleles	19
HLA_mismatch_base	20
HLA_mismatch_logical	21
HLA_mismatch_number	22
HLA_prefix_add	24
HLA_prefix_remove	25
HLA_truncate	26
HLA_typing_1	27
HLA_typing_LIS	28
HLA_typing_synthetic_LIS	28
HLA_validate	29
mismatch_table_2010	30
mismatch_table_2016	31
read_HML	32

Index	33
--------------	-----------

```
ambiguity_table_to_GLstring
      ambiguity_table_to_GLstring
```

Description

A function that converts a data table of HLA allele ambiguities (e.g. as created by ‘GLstring_expand_longer’ or ‘GLstring_to_ambiguity_table’) into a GL String format. The function processes the table by combining allele ambiguities, haplotypes, gene copies, and loci into a structured GL String.

Usage

```
ambiguity_table_to_GLstring(data, remove_duplicates = FALSE)
```

Arguments

<code>data</code>	A data frame containing columns that represent possible gene locations, loci, genotype ambiguities, genotypes, and haplotypes.
<code>remove_duplicates</code>	A logical value indicating if the function will check for duplicate entries at each step and remove them before assembling the final GL String. Useful if the ambiguity table has been altered, for example by truncating allele designations. Default is FALSE.

Value

A GL String representing the combined gene locations, loci, genotype ambiguities, genotypes, and haplotypes. A zero-row table returns ‘character(0)’, so filtered pipelines (e.g. ‘GLstring_expand_longer’ followed by a ‘dplyr::filter’ that removes every row) compose without special-casing.

Examples

```
# Example data frame input
data <- tibble::tribble(
  ~value, ~entry, ~possible_gene_location,
  ~locus, ~genotype_ambiguity, ~genotype, ~haplotype, ~allele,
  "HLA-A*01:01:01:01", 1, 1,
  1, 1, 1, 1, 1,
  "HLA-A*01:02", 1, 1,
  1, 1, 1, 1, 2,
  "HLA-A*01:03", 1, 1,
  1, 1, 1, 1, 3,
  "HLA-A*01:95", 1, 1,
  1, 1, 1, 1, 4,
  "HLA-A*24:02:01:01", 1, 1,
  1, 1, 2, 1, 1,
  "HLA-A*01:01:01:01", 1, 1,
  1, 2, 1, 1, 1,
```

```

"HLA-A*01:03", 1, 1,
1, 2, 1, 1, 2,
"HLA-A*24:03:01:01", 1, 1,
1, 2, 2, 1, 1,
"HLA-B*07:01:01", 1, 1,
2, 1, 1, 1, 1,
"B*15:01:01", 1, 1,
2, 1, 2, 1, 1,
"B*15:02:01", 1, 1,
2, 1, 2, 1, 2,
"B*07:03", 1, 1,
2, 2, 1, 1, 1,
"B*15:99:01", 1, 1,
2, 2, 2, 1, 1,
"HLA-DRB1*03:01:02", 1, 1,
3, 1, 1, 1, 1,
"HLA-DRB5*01:01:01", 1, 1,
3, 1, 1, 2, 1,
"HLA-KIR2DL5A*0010101", 1, 1,
3, 1, 2, 1, 1,
"HLA-KIR2DL5A*0010201", 1, 1,
3, 1, 3, 1, 1,
"HLA-KIR2DL5B*0010201", 1, 2,
1, 1, 1, 1, 1,
"HLA-KIR2DL5B*0010301", 1, 2,
1, 1, 2, 1, 1
)

ambiguity_table_to_GLstring(data)

```

GLstring_drop_non_expressed

GLstring_drop_non_expressed

Description

This function removes alleles carrying a WHO expression suffix from a GL String. By default the suffixes N (null), S (secreted) and C (cytoplasmic) are removed, on the reasoning that these alleles do not produce a protein at the cell surface; L (low), Q (questionable) and A (aberrant) alleles are kept, as some surface expression is possible. Which suffixes to treat as non-expressed is a clinical judgement, so the 'suffixes' argument can be set to any combination of the six.

Removal operates on whole alleles at any level of the GL String hierarchy: an allele ambiguity list narrows, a gene copy with no expressed alleles collapses, and a locus with no expressed alleles disappears along with its delimiter. If nothing in a GL String survives, 'NA' is returned for that entry.

Usage

```
GLstring_drop_non_expressed(GL_string, suffixes = c("N", "S", "C"))
```

Arguments

GL_string	A character vector of GL Strings.
suffixes	A character vector of WHO expression suffixes to remove. Any combination of "N", "Q", "L", "S", "C" and "A". Defaults to 'c("N", "S", "C")'.

Value

A character vector of GL Strings with the selected alleles removed, the same length as 'GL_string'. Entries with no remaining alleles are 'NA'.

Examples

```
# A null allele is removed from an allele ambiguity list, and a gene copy
# with no expressed alleles collapses:
GLstring_drop_non_expressed(
  "HLA-A*01:01N+HLA-A*02:01^HLA-B*07:02/HLA-B*07:02N+HLA-B*08:01"
)

# A locus with no expressed alleles disappears along with its delimiter:
GLstring_drop_non_expressed("HLA-A*01:01N+HLA-A*02:01N^HLA-B*07:02+HLA-B*08:01")

# L, Q and A alleles are kept by default; narrow `suffixes` to remove only
# null alleles:
GLstring_drop_non_expressed("HLA-A*24:02Q+HLA-A*01:01")
GLstring_drop_non_expressed("HLA-A*01:01N+HLA-A*30:14L", suffixes = "N")
```

GLstring_expand_longer
<i>GLstring_expand_longer</i>

Description

A function that expands a GL String into a longer, more detailed format (also known as an ambiguity table) by separating the string into its components resulting from its hierarchical set of operators, including gene locations, loci, genotypes, haplotypes, and alleles. The function processes each level of the GL String and assigns identifiers for each hierarchical component. The resulting table can be assembled back into a GL String using the function 'ambiguity_table_to_GLstring'.

Usage

```
GLstring_expand_longer(GL_string)
```

Arguments

GL_string	A GL String that encodes HLA alleles and their potential ambiguities
-----------	--

Value

A tibble that contains the expanded GL String with separate columns for possible gene locations, loci, genotype ambiguities, genotypes, haplotypes, and alleles, each with associated identifiers

Examples

```
file <- HLA_typing_1[, -1]
GL_string <- HLA_columns_to_GLstring(file, HLA_typing_columns = everything())
result <- GLstring_expand_longer(GL_string[1])
print(result)
```

GLstring_genes	<i>GLstring_genes</i>
----------------	-----------------------

Description

This function processes a specified column in a data frame that contains GL Strings. It separates the GL Strings, identifies the HLA loci, and transforms the data into a wider format with loci as column names.

Usage

```
GLstring_genes(data, gl_string)
```

Arguments

data	A data frame
gl_string	The name of the column in the data frame that contains GL Strings

Value

A data frame with GL Strings separated, loci identified, and data transformed to a wider format with loci as columns.

Examples

```
file <- HLA_typing_1[, -1]
GL_string <- data.frame("GL_string" = HLA_columns_to_GLstring(
  file,
  HLA_typing_columns = everything()
))
GL_string <- GL_string[1, , drop = FALSE] # When considering first patient
result <- GLstring_genes(GL_string, "GL_string")
print(result)
```

GLstring_genes_expanded
GLstring_genes_expanded

Description

This function processes a specified column in a data frame that contains GL Strings. It separates the GL Strings, identifies the HLA loci, and transforms the data into a wider format with loci as column names. It also creates multiple rows to separate each locus in the allele.

Usage

```
GLstring_genes_expanded(data, gl_string)
```

Arguments

<code>data</code>	A data frame containing GL Strings for HLA data.
<code>gl_string</code>	The name of the column in the data frame that contains GL Strings.

Value

A data frame with expanded columns, where each row has a single allele for a specific locus.

Examples

```
file <- HLA_typing_1[, -1]
GL_string <- data.frame("GL_string" = HLA_columns_to_GLstring(
  file,
  HLA_typing_columns = everything()
))
GL_string <- GL_string[1, , drop = FALSE] # When considering first patient
result <- GLstring_genes_expanded(GL_string, "GL_string")
print(result)
```

GLstring_gene_copies_combine
GLstring_gene_copies_combine

Description

A function for combining two columns of typing from the same locus into a single column in the appropriate GL String format.

Usage

```
GLstring_gene_copies_combine(.data, columns, sample_column = "sample")
```

Arguments

<code>.data</code>	A data frame
<code>columns</code>	The names of the columns in the data frame that contain typing information to be combined
<code>sample_column</code>	The name of the column that identifies samples in the data frame. Default is "sample".

Value

A data frame with the specified columns combined into a single column for each locus, in GL String format.

Examples

```
library(dplyr)
HLA_typing_1 %>%
  mutate(across(A1:B2, ~ HLA_prefix_add(.))) %>%
  GLstring_gene_copies_combine(c(A1:B2), sample_column = patient)
```

GLstring_genotype_ambiguity
GLstring_genotype_ambiguity

Description

This function processes GL Strings in the specified columns of a data frame to retain only the first genotype ambiguity, optionally retaining the remaining ambiguities in a separate column with "_ambiguity" appended. The function ensures that genes have been separated from the GL strings prior to execution; otherwise, an error will be thrown if a "^" is detected in the GL Strings.

Usage

```
GLstring_genotype_ambiguity(data, columns, keep_ambiguities = FALSE)
```

Arguments

<code>data</code>	A data frame
<code>columns</code>	The names of the columns in the data frame that contain GL Strings
<code>keep_ambiguities</code>	A logical value indicating whether to retain the remaining ambiguities in separate columns with "_genotype_ambiguity" appended to the original column names. Default is FALSE.

Value

A data frame with the first genotype ambiguity retained in the original columns. If `keep_ambiguities` is TRUE, the remaining ambiguities are placed in separate columns.

Examples

```
HLA_type <- data.frame(
  sample = c("sample1", "sample2"),
  HLA_A = c("A*01:01+A*68:01|A*01:02+A*68:55|A*01:99+A*68:66", "A*02:01+A*03:01|A*02:02+A*03:03"),
  HLA_B = c("B*07:02+B*58:01|B*07:03+B*58:09", "B*08:01+B*15:01|B*08:02+B*15:17"),
  stringsAsFactors = FALSE
)

GLstring_genotype_ambiguity(HLA_type, columns = c("HLA_A", "HLA_B"), keep_ambiguities = TRUE)
```

GLstring_regex

GLstring_regex

Description

This function will format an HLA allele (e.g. "HLA-A*02:01") to a regex pattern for searching within a GL String. The allele must carry its locus designation, but the "HLA-" prefix is optional in both the allele and the GL String being searched: "A*02:01" and "HLA-A*02:01" produce equivalent patterns, and either matches the allele whether the GL String writes it with or without the prefix (as wild-caught GL Strings often omit it). A bare field value with no locus ("02:01") is rejected, since the locus cannot be inferred; the 'HLA_prefix_add' function is useful for repairing such values.

The pattern matches whole alleles only, on these rules:

- * An allele supplied with fewer fields will match a higher-resolution allele in the GL String ("HLA-A*02:01" matches "HLA-A*02:01:01:01"), but partial fields never match ("HLA-A*02:14" will not match "HLA-A*02:149").
- * An allele without an expression suffix matches regardless of expression status: "HLA-A*24:09" matches "HLA-A*24:09N". It likewise matches G and P group names, which include the allele in their name ("HLA-A*01:01:01" matches "HLA-A*01:01:01G").
- * An allele supplied with a WHO expression suffix (N, Q, L, S, C or A) or a G/P group letter matches only alleles carrying that same letter, at the same or higher resolution: "HLA-A*01:01N" matches "HLA-A*01:01:03N" but not "HLA-A*01:01:03" or "HLA-A*01:01:03Q". Note that a G or P group query matches only the group name as written in the GL String; it does not expand to the group's member alleles ("HLA-A*01:01:01G" will not match "HLA-A*01:01:01" or "HLA-A*01:32"), since group membership is defined by the IPD-IMGT/HLA database rather than by the allele name. To find any member of a group, search the base allele name instead.

The matched text is always the full allele name as it appears in the GL String, so 'stringr::str_extract' returns the allele that was found (not the search string), and 'stringr::str_replace' replaces the whole allele.

Usage

```
GLstring_regex(data)
```

Arguments

data A string containing an HLA allele.

Value

A string with the HLA allele formatted as a regex pattern.

Examples

```
# To understand how the function works we can see how it alters the allele "HLA-A*02:01":
```

```
GLstring_regex("HLA-A*02:01")
```

```
# The result is the same allele with extra formatting to escape special characters found
# in a GL String, as well as the ability to accurately search for an allele in a GL String.
# For example, we would not want the allele "HLA-A*02:14" to match to "HLA-A*02:149:01",
# which would happen if we simply escaped the special characters:
```

```
library(stringr)
str_view("HLA-A*02:149:01", str_escape("HLA-A*02:14"), match = NA)
```

```
# Using `GLstring_regex` prevents this:
```

```
str_view("HLA-A*02:149:01", GLstring_regex("HLA-A*02:14"), match = NA)
```

```
# Using a longer GL String with multiple alleles and loci:
```

```
GL_string <- "HLA-A*02:01:01+HLA-A*68:01^HLA-B*07:01+HLA-B*15:01"
```

```
# We can match any allele accurately:
```

```
str_view(GL_string, GLstring_regex("HLA-A*68:01"), match = NA)
```

```
# Note that alleles supplied with fewer fields than in the GL String will also match,
# and that the match is the full allele as written in the GL String:
```

```
str_view(GL_string, GLstring_regex("HLA-A*02:01"), match = NA)
```

```
# An allele without an expression suffix matches its expression variants:
```

```
str_view("HLA-A*24:09N+HLA-A*02:01", GLstring_regex("HLA-A*24:09"), match = NA)
```

```
# An allele with an expression suffix matches only alleles carrying that suffix,
# at the same or higher resolution:
```

```
str_view("HLA-A*01:01:03N+HLA-A*02:01", GLstring_regex("HLA-A*01:01N"), match = NA)
str_view("HLA-A*01:01:03Q+HLA-A*02:01", GLstring_regex("HLA-A*01:01N"), match = NA)
```

```
# The "HLA-" prefix is optional in both the allele and the GL String, so a
# canonical allele finds its match in a wild-caught GL String without prefixes,
# and vice versa:
```

```
str_view("A*02:01:01+A*68:01", GLstring_regex("HLA-A*02:01"), match = NA)
str_view("HLA-A*02:01:01+HLA-A*68:01", GLstring_regex("A*02:01"), match = NA)
```

```
GLstring_to_ambiguity_table
      GLstring_to_ambiguity_table
```

Description

A function that expands a GL String into a longer, more detailed format (also known as an ambiguity table) by separating the string into its components resulting from its hierarchical set of operators, including gene locations, loci, genotypes, haplotypes, and alleles. The function processes each level of the GL String and assigns identifiers for each hierarchical component. The resulting table can be assembled back into a GL String using the function ‘ambiguity_table_to_GLstring’. This function is an alias of ‘GLstring_expand_longer’.

Usage

```
GLstring_to_ambiguity_table(GL_string)
```

Arguments

GL_string A GL String that encodes HLA alleles and their potential ambiguities

Value

A tibble that contains the expanded GL String with separate columns for possible gene locations, loci, genotype ambiguities, genotypes, haplotypes, and alleles, each with associated identifiers

Examples

```
file <- HLA_typing_1[, -1]
GL_string <- HLA_columns_to_GLstring(file, HLA_typing_columns = everything())
result <- GLstring_to_ambiguity_table(GL_string[1])
print(result)
```

```
Haplotype_frequencies    Ten      HLA-A~C~B~DRB3/4/5~DRB1~DQA1~DQB1~DPA1~DPB1
                        haplotypes and their frequencies in the global population.
```

Description

Ten HLA-A~C~B~DRB3/4/5~DRB1~DQA1~DQB1~DPA1~DPB1 haplotypes and their frequencies in the global population.

Usage

```
data(Haplotype_frequencies)
```

Format

A tibble:

HLA-A:HLA-DPB1 HLA allele names

Global_Fq haplotype frequencies

Global_n haplotype observations

Global_rank haplotype rank ...

Source

<<https://doi.org/10.1016/j.humimm.2021.04.007>>

HLA_columns_to_GLstring

HLA_columns_to_GLstring

Description

A function to take HLA typing data spread across different columns, as is often found in wild-caught data, and transform it to a GL String. If column names have anything besides the locus name and a number (e.g. "mA1Cd" instead of just "A1"), the function will have trouble determining the locus from the column name. The 'prefix_to_remove' and 'suffix_to_remove' arguments can be used to clean up the column names. See the example for how these arguments are used.

Usage

```
HLA_columns_to_GLstring(
  data,
  HLA_typing_columns,
  prefix_to_remove = "",
  suffix_to_remove = "",
  nomenclature = NULL,
  take_first_allele = TRUE
)
```

Arguments

data	A data frame with each row including an HLA typing result, with individual columns containing a single allele. One allele per cell is the function's contract: a cell holding a GL String is reduced to its first allele (see 'take_first_allele').
HLA_typing_columns	A list of columns containing the HLA alleles. Tidysselect is supported.
prefix_to_remove	An optional string of characters to remove from the locus names. The goal is to get the column names to the locus and a number. For example, columns named "mDRB11Cd" and "mDRB12Cd" should use the 'prefix_to_remove' value of "m".

suffix_to_remove	An optional string of characters to remove from the locus names. Using the example above, the 'suffix_to_remove' value will be "Cd".
nomenclature	An optional declaration of the output nomenclature, applied per locus. By default ('NULL') each cell is auto-detected: a value is molecular if it contains a colon, starts with a zero, contains a non-leading asterisk, or sits in a DQA1/DPA1/DPB1 column (a bare leading asterisk such as "*17" is treated as serologic). Supply "mol" or "ser" to force every selected locus to that nomenclature, or a named vector to set it per locus, e.g. 'c("HLA-Cw" = "ser", "HLA-DRB1" = "mol")'. Keys may be given in either the molecular or serologic spelling of a locus ("HLA-C" and "HLA-Cw" are equivalent). Relabeling is structural only: "ser" strips any asterisk and uses the serologic locus name; "mol" uses the molecular locus name. No cross-nomenclature allele translation is performed.
take_first_allele	A logical value, passed through to 'HLA_validate' when each cell is cleaned. If TRUE (the default), a cell containing GL String delimiters ("^", " ", "+", "~", "/" or "?") is silently reduced to its first allele. If FALSE, such cells raise an error instead, for callers that want malformed multi-allele cells caught rather than truncated.

Value

A list of GL Strings in the order of the original data frame.

Examples

```
# The HLA_typing_LIS dataset contains a table as might be found in a clinical laboratory
# information system:
print(HLA_typing_LIS)

# The `HLA_columns_to_GLstring` function can be used to coerce typing spread across
# multiple columns into a GL String:
library(dplyr)
HLA_typing_LIS %>%
  mutate(
    GL_string = HLA_columns_to_GLstring(
      ., # Note that if this function is used inside a `mutate` call "." will have to be
      # used as the first argument to extract data from the working data frame.
      HLA_typing_columns = mA1Cd.recipient:mDPB12cd.recipient,
      prefix_to_remove = "m",
      suffix_to_remove = "Cd.recipient"
    ),
    .after = patient
  ) %>%
  select(patient, GL_string)
```

HLA_column_repair	<i>HLA_column_repair</i>
-------------------	--------------------------

Description

This function will change column names that have the official HLA nomenclature (e.g. "HLA-A*" or "HLA-A") to a format more easily selected in tidyverse functions (e.g. "HLA_A"). The dash and asterisk are special characters in R, and makes selecting columns by name difficult. This function will also allow for easily changing back to WHO-compliant nomenclature (e.g. "HLA-A*").

Usage

```
HLA_column_repair(data, format = "tidyverse", asterisk = FALSE)
```

Arguments

data	A data frame
format	Either "tidyverse" or "WHO".
asterisk	Logical value to return column with an asterisk.

Value

A data frame object with column names renamed in the specified format.

Examples

```
HLA_type <- data.frame(
  "HLA-A*" = c("01:01", "02:01"),
  "HLA-B*" = c("07:02", "08:01"),
  "HLA-C*" = c("03:04", "04:01"),
  stringsAsFactors = FALSE
)

HLA_column_repair(HLA_type, format = "tidyverse")
```

HLA_dictionary	<i>Data on HLA alleles in the 2008 HLA dictionary</i>
----------------	---

Description

Data on HLA alleles in the 2008 HLA dictionary

Usage

```
data(HLA_dictionary)
```

Format

A tibble:

IPD.Accession IPD-IMGT/HLA Accession number

HLA.Allele HLA allele name

WHO.Assigned.Type WHO-assigned serologic equivalent ...

Source

<<https://doi.org/10.1111/j.1399-0039.2008.01183.x>>

HLA_match_number	<i>HLA_match_number</i>
------------------	-------------------------

Description

Calculates the number of HLA matches as two minus the number of mismatches from 'HLA_mismatch_number'. Homozygous mismatches are counted twice. Supports match calculations for host-vs-graft (HvG), graft-vs-host (GvH), or bidirectional. Bidirectional matching is the default, but can be overridden using the "direction" argument.

Usage

```
HLA_match_number(  
  GL_string_recip,  
  GL_string_donor,  
  loci,  
  direction = "bidirectional"  
)
```

Arguments

- | | |
|-----------------|---|
| GL_string_recip | A GL String representing the recipient's HLA genotype. |
| GL_string_donor | A GL String representing the donor's HLA genotype. |
| loci | A character vector specifying the loci to be considered for mismatch calculation. HLA-DRB3/4/5 (and their serologic equivalents DR51/52/53) are considered once locus for this function, and should be called in this argument as "HLA-DRB3/4/5" or "HLA-DR51/52/53", respectively. |
| direction | A character string indicating the direction of match. Options are "HvG" (host vs. graft), "GvH" (graft vs. host), "bidirectional" (the minimum value of "HvG" and "GvH"). |

Value

An integer value or a character string: - If 'loci' includes only one locus, the function returns an integer mismatch count for that locus. - If 'loci' includes multiple loci, the function returns a character string in the format "Locus1=Count1, Locus2=Count2, ...".

Examples

```
file <- HLA_typing_1[, -1]
GL_string <- HLA_columns_to_GLstring(file, HLA_typing_columns = everything())
GL_string_recip <- GL_string[1]
GL_string_donor <- GL_string[2]

loci <- c("HLA-A", "HLA-B")

# Calculate mismatch numbers (Host vs. Graft)
HLA_match_number(GL_string_recip, GL_string_donor, loci, direction = "HvG")

# Calculate mismatch numbers (Graft vs. Host)
HLA_match_number(GL_string_recip, GL_string_donor, loci, direction = "GvH")

# Calculate mismatch numbers (Bidirectional)
HLA_match_number(GL_string_recip, GL_string_donor,
  loci,
  direction = "bidirectional"
)
```

HLA_match_summary_HCT *HLA_match_summary_HCT*

Description

Calculates the match summary for either the HLA-A, B, C and DRB1 loci (out-of-8 matching) or the HLA-A, B, C, DRB1 and DQB1 loci (out-of-10 matching), as is commonly used for hematopoietic cell transplantation (HCT). Homozygous mismatches are counted twice. Bidirectional matching is the default, but can be overridden with the "direction" argument.

Usage

```
HLA_match_summary_HCT(
  GL_string_recip,
  GL_string_donor,
  direction = "bidirectional",
  match_grade,
  scope = "locus"
)
```

Arguments

GL_string_recip	A GL String representing the recipient's HLA genotype, and minimally containing the HLA-A, B, C and DRB1 loci (for Xof8 matching) or the HLA-A, B, C, DRB1 and DQB1 loci (for Xof10 matching).
GL_string_donor	A GL String representing the donor's HLA genotype, and minimally containing the HLA-A, B, C and DRB1 loci (for Xof8 matching) or the HLA-A, B, C, DRB1 and DQB1 loci (for Xof10 matching).
direction	"GvH", "HvG" or "bidirectional". Default is "bidirectional".
match_grade	"Xof8" for HLA-A, B, C and DRB1 matching or "Xof10" for HLA-A, B, C, DRB1 and DQB1 matching.
scope	"locus" or "genotype". Default is "locus". When "locus" (the default), bidirectional matching takes the minimum match count at each locus independently before summing. When "genotype", the GvH and HvG match summaries are calculated separately across all loci, and the maximum of the two totals is returned. Only affects results when direction is "bidirectional"; ignored otherwise.

Value

An integer value of the match grade summary.

Examples

```
# Example recipient and donor GL Strings
file <- HLA_typing_1[, -1]
GL_string <- HLA_columns_to_GLstring(file, HLA_typing_columns = everything())

GL_string_recip <- GL_string[1]
GL_string_donor <- GL_string[2]

# Calculate mismatch numbers
HLA_match_summary_HCT(GL_string_recip, GL_string_donor,
  direction = "bidirectional", match_grade = "Xof8"
)

# Genotype-level bidirectional matching (max of GvH and HvG totals)
HLA_match_summary_HCT(GL_string_recip, GL_string_donor,
  direction = "bidirectional", match_grade = "Xof8", scope = "genotype"
)
```

HLA_mismatched_alleles

HLA_mismatched_alleles

Description

A function to return a string of mismatches between recipient and donor HLA genotypes represented as GL Strings. The function finds mismatches based on the direction of comparison specified in the inputs and also handles homozygosity.

'HLA_mismatch_alleles' and 'HLA_mismatched_alleles' are synonyms.

Usage

```
HLA_mismatched_alleles(
  GL_string_recip,
  GL_string_donor,
  loci,
  direction,
  homozygous_count = 2
)
```

Arguments

GL_string_recip	A GL Strings representing the recipient's HLA genotypes.
GL_string_donor	A GL Strings representing the donor's HLA genotypes.
loci	A character vector specifying the loci to be considered for mismatch calculation. HLA-DRB3/4/5 (and their serologic equivalents DR51/52/53) are considered once locus for this function, and should be called in this argument as "HLA-DRB3/4/5" or "HLA-DR51/52/53", respectively.
direction	A character string indicating the direction of mismatch. Options are "HvG" (host vs. graft), "GvH" (graft vs. host), "bidirectional" (the max value of "HvG" and "GvH"), or "SOT" (host vs. graft, as is used for mismatching in solid organ transplantation).
homozygous_count	An integer specifying how to handle homozygosity. Defaults to 2, where homozygous alleles are treated as duplicated for mismatch calculations. Can be specified as 1, in which case homozygous alleles are treated as single occurrences without duplication (in other words, homozygous mismatches are only "counted" once).

Value

A character vector, where each element is a string summarizing the mismatches for the specified loci. The strings are formatted as comma-separated locus mismatch entries if multiple loci were supplied, or as simple GL Strings if a single locus was supplied.

Examples

```
file <- HLA_typing_1[, -1]
GL_string <- HLA_columns_to_GLstring(file, HLA_typing_columns = everything())
```

```

GL_string_recip <- GL_string[1]
GL_string_donor <- GL_string[2]

loci <- c("HLA-A", "HLA-DRB3/4/5", "HLA-DPB1")
mismatches <- HLA_mismatched_alleles(GL_string_recip, GL_string_donor, loci, direction = "HvG")
print(mismatches)

```

HLA_mismatch_alleles *HLA_mismatch_alleles*

Description

A function to return a string of mismatches between recipient and donor HLA genotypes represented as GL Strings. The function finds mismatches based on the direction of comparison specified in the inputs and also handles homozygosity.

‘HLA_mismatch_alleles’ and ‘HLA_mismatched_alleles’ are synonyms.

Usage

```

HLA_mismatch_alleles(
  GL_string_recip,
  GL_string_donor,
  loci,
  direction,
  homozygous_count = 2
)

```

Arguments

GL_string_recip	A GL Strings representing the recipient's HLA genotypes.
GL_string_donor	A GL Strings representing the donor's HLA genotypes.
loci	A character vector specifying the loci to be considered for mismatch calculation. HLA-DRB3/4/5 (and their serologic equivalents DR51/52/53) are considered once locus for this function, and should be called in this argument as "HLA-DRB3/4/5" or "HLA-DR51/52/53", respectively.
direction	A character string indicating the direction of mismatch. Options are "HvG" (host vs. graft), "GvH" (graft vs. host), "bidirectional" (the max value of "HvG" and "GvH"), or "SOT" (host vs. graft, as is used for mismatching in solid organ transplantation).
homozygous_count	An integer specifying how to handle homozygosity. Defaults to 2, where homozygous alleles are treated as duplicated for mismatch calculations. Can be specified as 1, in which case homozygous alleles are treated as single occurrences without duplication (in other words, homozygous mismatches are only "counted" once).

Value

A character vector, where each element is a string summarizing the mismatches for the specified loci. The strings are formatted as comma-separated locus mismatch entries if multiple loci were supplied, or as simple GL Strings if a single locus was supplied.

Examples

```
file <- HLA_typing_1[, -1]
GL_string <- HLA_columns_to_GLstring(file, HLA_typing_columns = everything())

GL_string_recip <- GL_string[1]
GL_string_donor <- GL_string[2]

loci <- c("HLA-A", "HLA-DRB3/4/5", "HLA-DPB1")
mismatches <- HLA_mismatch_alleles(GL_string_recip, GL_string_donor, loci, direction = "HvG")
print(mismatches)
```

HLA_mismatch_base	<i>HLA_mismatch_base</i>
-------------------	--------------------------

Description

A function to return a string of mismatches between recipient and donor HLA genotypes represented as GL Strings. The function finds mismatches based on the direction of comparison specified in the inputs and also handles homozygosity. As the name implies, this function is the base for all other mismatch (and matching) functions. This function is not meant to be called directly; it is better to use one of the derivative functions.

Usage

```
HLA_mismatch_base(
  GL_string_recip,
  GL_string_donor,
  loci,
  direction,
  homozygous_count = 2
)
```

Arguments

GL_string_recip
A GL String representing the recipient's HLA genotype.

GL_string_donor
A GL String representing the donor's HLA genotype.

loci	A character vector specifying the loci to be considered for mismatch calculation. HLA-DRB3/4/5 (and their serologic equivalents DR51/52/53) are considered once locus for this function, and should be called in this argument as "HLA-DRB3/4/5" or "HLA-DR51/52/53", respectively.
direction	A character string indicating the direction of mismatch. Options are "HvG" (host vs. graft) or "GvH" (graft vs. host).
homozygous_count	An integer specifying how to handle homozygosity. Defaults to 2, where homozygous alleles are treated as duplicated for mismatch calculations. Can be specified to be 1, in which case homozygous alleles are treated as single occurrences without duplication.

Value

A character vector, where each element is a string summarizing the mismatches for the specified loci. The strings are formatted as comma-separated locus mismatch entries if multiple loci are supplied, or simple GL Strings if a single locus is supplied. A pair in which either GL String is NA returns NA for that pair rather than erroring, so cohorts with missing typing can be processed in one call. Note that in single-locus mode NA is also the no-mismatch result; the derivative functions distinguish the two cases from their inputs.

Examples

```
file <- HLA_typing_1[, -1]
GL_string <- HLA_columns_to_GLstring(file, HLA_typing_columns = everything())

GL_string_recip <- GL_string[1]
GL_string_donor <- GL_string[2]

loci <- c("HLA-A", "HLA-DRB3/4/5", "HLA-DPB1")
mismatches <- HLA_mismatch_base(GL_string_recip, GL_string_donor, loci, direction = "HvG")
print(mismatches)
```

HLA_mismatch_logical *HLA_mismatch_logical*

Description

Determines if there are any mismatches between recipient and donor HLA alleles for the specified loci. Returns 'TRUE' if mismatches are present, and 'FALSE' otherwise.

Usage

```
HLA_mismatch_logical(GL_string_recip, GL_string_donor, loci, direction)
```

Arguments

GL_string_recip	A GL Strings representing the recipient’s HLA genotypes.
GL_string_donor	A GL Strings representing the donor’s HLA genotypes.
loci	A character vector specifying the loci to be considered for mismatch calculation. HLA-DRB3/4/5 (and their serologic equivalents DR51/52/53) are considered once locus for this function, and should be called in this argument as "HLA-DRB3/4/5" or "HLA-DR51/52/53", respectively.
direction	A character string indicating the direction of mismatch. Options are "HvG" (host vs. graft), "GvH" (graft vs. host), "bidirectional" (if either "HvG" or "GvH" is TRUE), or "SOT" (host vs. graft, as is used for mismatching in solid organ transplantation).

Value

A logical value (‘TRUE’ or ‘FALSE’): - ‘TRUE’ if there are mismatches between recipient and donor HLA alleles. - ‘FALSE’ if there are no mismatches. - ‘NA’ if either GL String of a pair is NA (missing typing).

Examples

```
file <- HLA_typing_1[, -1]
GL_string <- HLA_columns_to_GLstring(file, HLA_typing_columns = everything())

GL_string_recip <- GL_string[1]
GL_string_donor <- GL_string[2]

loci <- c("HLA-A", "HLA-DRB3/4/5", "HLA-DPB1")
mismatches <- HLA_mismatch_logical(GL_string_recip, GL_string_donor, loci, direction = "HvG")
print(mismatches)
```

HLA_mismatch_number	<i>HLA_mismatch_number</i>
---------------------	----------------------------

Description

Calculates the number of mismatched HLA alleles between a recipient and a donor across specified loci. Supports mismatch calculations for host-vs-graft (HvG), graft-vs-host (GvH), or bidirectional.

Usage

```
HLA_mismatch_number(
  GL_string_recip,
  GL_string_donor,
  loci,
```

```

    direction,
    homozygous_count = 2
)

```

Arguments

GL_string_recip	A GL String representing the recipient's HLA genotype.
GL_string_donor	A GL String representing the donor's HLA genotype.
loci	A character vector specifying the loci to be considered for mismatch calculation. HLA-DRB3/4/5 (and their serologic equivalents DR51/52/53) are considered once locus for this function, and should be called in this argument as "HLA-DRB3/4/5" or "HLA-DR51/52/53", respectively.
direction	A character string indicating the direction of mismatch. Options are "HvG" (host vs. graft), "GvH" (graft vs. host), "bidirectional" (the max value of "HvG" and "GvH"), or "SOT" (host vs. graft, as is used for mismatching in solid organ transplantation).
homozygous_count	An integer specifying how to count homozygous mismatches. Defaults to 2, where homozygous mismatches are treated as two mismatches, regardless if one or two alleles are supplied in the GL String (in cases where one allele is supplied, it is duplicated by the function). If specified as 1, homozygous mismatches are only counted once, regardless of whether one or two alleles are supplied in the GL String (in cases where two alleles are supplied, the second identical allele is deleted).

Value

An integer value or a character string: - If 'loci' includes only one locus, the function returns an integer mismatch count for that locus. - If 'loci' includes multiple loci, the function returns a character string in the format "Locus1=Count1, Locus2=Count2, ...".

Examples

```

file <- HLA_typing_1[, -1]
GL_string <- HLA_columns_to_GLstring(file, HLA_typing_columns = everything())

GL_string_recip <- GL_string[1]
GL_string_donor <- GL_string[2]

loci <- c("HLA-A", "HLA-DRB3/4/5", "HLA-DPB1")

# Calculate mismatch numbers (Host vs. Graft)
HLA_mismatch_number(GL_string_recip, GL_string_donor, loci, direction = "HvG")

# Calculate mismatch numbers (Graft vs. Host)
HLA_mismatch_number(GL_string_recip, GL_string_donor, loci, direction = "GvH")

# Calculate mismatch numbers (Bidirectional)

```

```
HLA_mismatch_number(GL_string_recip, GL_string_donor,
  loci,
  direction = "bidirectional"
)
```

HLA_prefix_add	<i>HLA_prefix_add</i>
----------------	-----------------------

Description

This function adds a specified prefix to the beginning of each HLA type, and works on a single allele or all alleles in a GL String. Useful for adding HLA or gene prefixes.

Usage

```
HLA_prefix_add(data, prefix = "HLA-")
```

Arguments

- data A string with a single HLA allele, a GL String of HLA alleles, or a character vector containing either of the previous.
- prefix A character string to be added as a prefix to the alleles. Default is "HLA-".

Value

A vector with the specified prefix added to the values.

Examples

```
# The HLA_typing_LIS dataset contains a table as might be found in a clinical
# laboratory information system:
print(HLA_typing_LIS)

# The `HLA_prefix_add` function can be used to add the correct HLA prefixes to the table:
library(dplyr)
HLA_typing_LIS %>% mutate(
  across(
    mA1Cd.recipient:mA2Cd.recipient,
    ~ HLA_prefix_add(., "HLA-A*")
  )
)
```

HLA_prefix_remove	<i>HLA_prefix_remove</i>
-------------------	--------------------------

Description

This function removes HLA and optionally locus prefixes from a string of HLA typing: "HLA-A2" changes to "A2" or "2". By default, HLA and locus prefixes are removed. This function also works on each allele in a GL String.

Usage

```
HLA_prefix_remove(data, keep_locus = FALSE)
```

Arguments

data	A string with a single HLA allele, a GL String of HLA alleles, or a character vector containing either of the previous.
keep_locus	A logical value indicating whether to retain any locus values. The default value is FALSE.

Value

A vector modified to remove HLA and optionally locus prefixes.

Examples

```
# The HLA_typing_1 dataset contains a table with HLA typing spread across multiple columns:
print(HLA_typing_1)

# The `HLA_prefix_remove` function can be used to get each column to have only the
# colon-separated fields:
library(dplyr)
HLA_typing_1 %>% mutate(
  across(
    A1:DPB1_2,
    ~ HLA_prefix_remove(.)
  )
)
```

HLA_truncate

*HLA_truncate***Description**

This function truncates HLA typing values in molecular nomenclature (for example from 4 fields to 2 fields). The truncation is based on the number of fields specified and optionally retains any WHO-recognized suffixes (L, S, C, A, Q, or N) or G and P group designations (G or P). This function will work on individual alleles (e.g. "HLA-A*02:01:01:01") or on all alleles in a GL String (e.g. "HLA-A*02:01:01:01+HLA-A*68:01:01^HLA-DRB1*01:01:01+HLA-DRB1*03:01:01").

Note: depending on arguments used, this function can output HLA alleles that do not exist in the IPD-IMGT/HLA database. For example, truncating the allele "DRB4*01:03:01:02N" to 2 fields would result in "DRB4*01:03N," which does not exist in the IPD-IMGT/HLA database. Users should take care in setting the parameters for this function.

Usage

```
HLA_truncate(
  data,
  fields = 2,
  keep_suffix = TRUE,
  keep_G_P_group = FALSE,
  remove_duplicates = FALSE
)
```

Arguments

<code>data</code>	A string containing an HLA allele or a GL String.
<code>fields</code>	An integer specifying the number of fields to retain in the truncated values. Default is 2.
<code>keep_suffix</code>	A logical value indicating whether to retain any WHO-recognized suffixes. Default is TRUE.
<code>keep_G_P_group</code>	A logical value indicating whether to retain any G or P group designations. Default is FALSE.
<code>remove_duplicates</code>	A logical value indicating whether to remove duplicated values from a GL String after truncation. Default is FALSE.

Value

A string with the HLA typing truncated according to the specified number of fields and optional suffix retention.

Examples

```
# The Haplotype_frequencies dataset contains a table with HLA typing spread across multiple columns:
print(Haplotype_frequencies)

# The `HLA_truncate` function can be used to truncate the typing results to 2 fields:
library(dplyr)
Haplotype_frequencies %>% mutate(
  across(
    "HLA-A":"HLA-DPB1",
    ~ HLA_truncate(
      .,
      fields = 2,
      keep_suffix = TRUE,
      keep_G_P_group = FALSE
    )
  )
)
```

HLA_typing_1

Synthetic HLA typing data for 10 individuals for the HLA-A, B, C, DRB1, DRB3/4/5, DQB1, DQA1, DPB1 and DPA1 loci.

Description

Synthetic HLA typing data for 10 individuals for the HLA-A, B, C, DRB1, DRB3/4/5, DQB1, DQA1, DPB1 and DPA1 loci.

Usage

```
data(HLA_typing_1)
```

Format

A tibble:

patient Patients 1-10

A1:DPB1_2 HLA allele names ...

Source

Synthetic data from Nicholas K. Brown.

HLA_typing_LIS	<i>Synthetic HLA typing data for 10 individuals for the HLA-A, B, C, DRB1, DRB3, DRB4, DRB5, DQB1, DQA1, DPB1 and DPA1 loci. Data formatted to resemble a HistoTrac table.</i>
----------------	--

Description

Synthetic HLA typing data for 10 individuals for the HLA-A, B, C, DRB1, DRB3, DRB4, DRB5, DQB1, DQA1, DPB1 and DPA1 loci. Data formatted to resemble a HistoTrac table.

Usage

```
data(HLA_typing_LIS)
```

Format

A tibble:

patient Patients 1-10

mA1Cd.recipient:mDPB12cd.recipient HLA allele names ...

Source

Synthetic data from Nicholas K. Brown.

HLA_typing_synthetic_LIS	<i>Synthetic clinical-laboratory HLA typing table (HistoTrac "Patient" layout).</i>
--------------------------	---

Description

A larger, deliberately messy synthetic data set patterned on a real laboratory information system "Patient" table, with one individual per row. It is intended as realistic development and test data for [HLA_columns_to_GLstring](#) and related functions: the typing columns exercise the full range of wild-caught notation, including serologic values with and without a leading "w" (e.g. "w4", "4"), leading zeros ("06"), bare-asterisk molecular shorthand ("*17"), colon-delimited molecular alleles ("02:01"), blanks, and NAs. Both serologic (A1Cd:dq2cd) and molecular (mA1Cd:mDPB12cd) typings are present for the HLA-A, B, Bw, C, DRB1, DRB3/4/5, DQB1, DQA1, DPB1 and DPA1 loci.

Usage

```
data(HLA_typing_synthetic_LIS)
```

Format

A tibble with 63 rows and 53 columns:

PatientId, UNOSId, HospitalID, PhysicianID Synthetic record identifiers.

firstnm, lastnm, DOB Synthetic demographics (donors carry their donor ID in firstnm).

categoryCd, RaceCd, GenderCd, ABOCd, PatientTypeCd, StatusCd Synthetic categorical fields.

A1Cd:dq2cd Serologic HLA typing (A, B, Bw, Cw, DR, DRw, DQ), as character.

mA1Cd:mDPB12cd Molecular HLA typing (A, B, C, DRB1, DRB3/4/5, DQA1, DQB1, DPA1, DPB1), as character.

UNOSCPRAAmt Synthetic calculated PRA value.

UnacceptAntigenTxt, ModerateRiskAntibodyTxt, UNOSUnacceptAntigenTxt Synthetic anti-body text fields.

Details

Note that the molecular (m...) columns deliberately contain a *mix* of true molecular values (e.g. "31:01") and serologic values (e.g. "w6"): the source laboratory historically recorded serologic typing in the molecular columns, so this captures that real-world mixed-nomenclature-within-a-column case.

All identifiers, names and dates are synthetic.

Source

Synthetic data from Nicholas K. Brown, patterned on a clinical laboratory information system.

HLA_validate

HLA_validate

Description

Returns only HLA alleles in valid nomenclature, either serologic or molecular. Simple numbers, such as "2" or "27" will be returned as-is. Suffixes that are not WHO-recognized suffixes (L, S, C, A, Q, N) or G or P group designations will be removed. For example "novel" at the end of the allele will be removed, while "n" at the end of the allele will be retained. Other values, such as "blank" or "-" will be converted to NA values. This function is helpful for cleaning up the typing of an entire table of HLA values.

Each value is expected to hold a single allele. If a value contains GL String delimiters ("^", "|", "+", "~", "/" or "?"), only the first allele is retained (the historical behavior, controlled by 'take_first_allele'); set 'take_first_allele = FALSE' to treat such values as malformed input and get an error instead. Use the GL String functions (e.g. 'GLstring_expand_longer') to work with multi-allele values.

Usage

```
HLA_validate(data, take_first_allele = TRUE)
```

Arguments

data A string containing an HLA allele.

take_first_allele A logical value. If TRUE (the default), a value containing GL String delimiters is silently reduced to its first allele. If FALSE, such values raise an error instead, for callers that want malformed multi-allele input caught rather than truncated.

Value

A string with a valid HLA allele or NA if no valid allele was present.

Examples

```
HLA_validate("HLA-A2")
HLA_validate("A*02:01:01:01N")
HLA_validate("A*02:01:01N")
HLA_validate("HLA-DRB1*02:03novel")
HLA_validate("HLA-DQB1*03:01v")
HLA_validate("HLA-DRB1*02:03P")
HLA_validate("HLA-DPB1*04:01:01G")
HLA_validate("2")
HLA_validate(2)
HLA_validate("B27")
HLA_validate("A*010101")
HLA_validate("-")
HLA_validate("blank")

# The HLA_typing_LIS dataset contains a table with HLA typing spread across multiple columns:
print(HLA_typing_LIS)

# Cleaning up the entire table. Note that blank values will be converted to "NA".
library(dplyr)
HLA_typing_LIS %>% mutate(
  across(
    mA1Cd.recipient:mDPB12cd.recipient,
    ~ HLA_validate(.)
  )
)
```

mismatch_table_2010	<i>Consensus mismatch numbers for every possible allele combination at a single locus, from the 2010 publication.</i>
---------------------	---

Description

Consensus mismatch numbers for every possible allele combination at a single locus, from the 2010 publication.

Usage

```
data(mismatch_table_2010)
```

Format

A tibble:

patient:donor allele combinations for patient and donor

#GvH:#Max consensus mismatch number for each direction ...

Source

<<https://www.nature.com/articles/bmt2010132>>

mismatch_table_2016	<i>Consensus mismatch numbers for every possible allele combination at a single locus, from the 2016 publication.</i>
---------------------	---

Description

Consensus mismatch numbers for every possible allele combination at a single locus, from the 2016 publication.

Usage

```
data(mismatch_table_2016)
```

Format

A tibble:

Patient:Donor allele combinations for patient and donor

#GvH:#Max consensus mismatch number for each direction ...

Source

<<https://www.nature.com/articles/bmt2010132>>

`read_HML`*read_HML*

Description

Reads the GL Strings of HML files and returns a tibble with the full genotype for each sample.

Usage

```
read_HML(HML_file)
```

Arguments

`HML_file` The path to an HML file.

Value

A tibble with the sample name and the GL String.

Examples

```
HML_1 <- system.file("extdata", "HML_1.hml", package = "immunogenetr")
HML_2 <- system.file("extdata", "hml_2.hml", package = "immunogenetr")

read_HML(HML_1)
read_HML(HML_2)
```

Index

* datasets

- Haplotype_frequencies, [11](#)
- HLA_dictionary, [14](#)
- HLA_typing_1, [27](#)
- HLA_typing_LIS, [28](#)
- HLA_typing_synthetic_LIS, [28](#)
- mismatch_table_2010, [30](#)
- mismatch_table_2016, [31](#)

ambiguity_table_to_GLstring, [3](#)

- GLstring_drop_non_expressed, [4](#)
- GLstring_expand_longer, [5](#)
- GLstring_gene_copies_combine, [7](#)
- GLstring_genes, [6](#)
- GLstring_genes_expanded, [7](#)
- GLstring_genotype_ambiguity, [8](#)
- GLstring_regex, [9](#)
- GLstring_to_ambiguity_table, [11](#)

- Haplotype_frequencies, [11](#)
- HLA_column_repair, [14](#)
- HLA_columns_to_GLstring, [12](#), [28](#)
- HLA_dictionary, [14](#)
- HLA_match_number, [15](#)
- HLA_match_summary_HCT, [16](#)
- HLA_mismatch_alleles, [19](#)
- HLA_mismatch_base, [20](#)
- HLA_mismatch_logical, [21](#)
- HLA_mismatch_number, [22](#)
- HLA_mismatched_alleles, [17](#)
- HLA_prefix_add, [24](#)
- HLA_prefix_remove, [25](#)
- HLA_truncate, [26](#)
- HLA_typing_1, [27](#)
- HLA_typing_LIS, [28](#)
- HLA_typing_synthetic_LIS, [28](#)
- HLA_validate, [29](#)

mismatch_table_2010, [30](#)

mismatch_table_2016, [31](#)

read_HML, [32](#)