Package ‘immunotation’

May 8, 2025

Type Package

Title Tools for working with diverse immune genes

Version 1.17.0

Date 2021-08-09

Description
MHC (major histocompatibility complex) molecules are cell surface complexes that present antigens to T cells. The repertoire of antigens presented in a given genetic background largely depends on the sequence of the encoded MHC molecules, and thus, in humans, on the highly variable HLA (human leukocyte antigen) genes of the hyperpolymorphic HLA locus. More than 28,000 different HLA alleles have been reported, with significant differences in allele frequencies between human populations worldwide. Reproducible and consistent annotation of HLA alleles in large-scale bioinformatics workflows remains challenging, because the available reference databases and software tools often use different HLA naming schemes. The package immunotation provides tools for consistent annotation of HLA genes in typical immunoinformatics workflows such as for example the prediction of MHC-presented peptides in different human donors. Converter functions that provide mappings between different HLA naming schemes are based on the MHC restriction ontology (MRO). The package also provides automated access to HLA alleles frequencies in worldwide human reference populations stored in the Allele Frequency Net Database.

License GPL-3

Encoding UTF-8

LazyData false

Collate 'external_resources_input.R' 'AFND_interface.R'
'MRO_interface_helper.R' 'MRO_interface.R' 'MACUI_interface.R'
'nomenclature_queries.R' 'AFND_queries.R' 'visualization.R'

Depends R (>= 4.1)

Imports stringr, ontologyIndex, curl, ggplot2, readr, rvest, tidyr, xml2, maps, rlang

Suggests BiocGenerics, rmarkdown, BiocStyle, knitr, testthat, DT

VignetteBuilder knitr

RoxygenNote 7.1.1

biocViews Software, ImmunoOncology, BiomedicalInformatics, Genetics, Annotation

BugReports https://github.com/imkeller/immunotation/issues

git_url https://git.bioconductor.org/packages/immunotation

git_branch devel

git_last_commit 22f5876

git_last_commit_date 2025-04-15

Repository Bioconductor 3.22

Date/Publication 2025-05-08

Author Katharina Imkeller [cre, aut]

Maintainer Katharina Imkeller <`k.imkeller@dkfz.de`>

Contents

`assemble_protein_complex` *Assemble protein complex*

Description

Assemble a table or MHC protein complexes for a given organism.

Usage

```
assemble_protein_complex(organism)
```

Arguments

`organism`	Organism for which the lookup should be built (e.g. "human", "mouse", ...). The list of valid organisms can be found using the function `get_valid_organisms`

Value

a data frame with the MHC complexes annotated in MRO (only completely annotated complexes are returned)

Examples

```
assemble_protein_complex(organism = "mouse")
```

`build_allele_group` *Building a list of alleles to cover*

Description

`build_allele_group` e.g. A*01:01 -> A*01:01:01, A*01:01:02, A*01:01:03

Usage

```
build_allele_group(allele_selection)
```

Arguments

`allele_selection`	HLA allele for whicht the allele group should be built.

Value

list of alleles

Examples

```
build_allele_group("A*01:01")
```

`check_geographics` *check_geographics, stops if input not adequate*

Description

check_geographics, stops if input not adequate

Usage

```
check_geographics(country, region, ethnic)
```

Arguments

`country`	country used for allele frequency selection
`region`	geographical region used for allele frequency selection
`ethnic`	ethical origin used for allele frequency selection

Value

TRUE

`check_hla_locus` *check_hla_locus, stops if input not adequate*

Description

check_hla_locus, stops if input not adequate

Usage

```
check_hla_locus(hla_locus)
```

Arguments

`hla_locus`	locus of hla frequencies to query

Value

TRUE

`check_hla_selection` *check_hla_selection, stops if input not adequate*

Description

check_hla_selection, stops if input not adequate

Usage

```
check_hla_selection(hla_selection, query_type)
```

Arguments

`hla_selection`	HLA alleles used for selection
`query_type`	["allele"|"haplotype"] type of AFND query

Value

TRUE

`check_population` *check_population, stops if input not adequate*

Description

check_population, stops if input not adequate

Usage

```
check_population(hla_population)
```

Arguments

`hla_population`	population id

Value

TRUE

`check_sample_size` *check_sample_size, stops if input not adequate*

Description

check_sample_size, stops if input not adequate

Usage

```
check_sample_size(hla_sample_size_pattern, hla_sample_size)
```

Arguments

`hla_sample_size_pattern`	one of "bigger_than", "equal", "less_than", "less_equal_than", "bigger_equal_than","different"
`hla_sample_size`	integer number used for population size

Value

TRUE

`check_standard` *check_standard, stops if input not adequate*

Description

check_standard, stops if input not adequate

Usage

```
check_standard(standard)
```

Arguments

`standard`	one of "a" - all,"s" - silver,"g" - gold

Value

TRUE

`create_encode_handle` *create_encode_handle*

Description

create_encode_handle

Usage

```
create_encode_handle(allele_names)
```

Arguments

`allele_names`	list of HLA alleles

Value

curl handle

`decode_MAC` *Decode MAC*

Description

Decode a multiple allele code (MAC) into a list of HLA alleles. #' The National Marrow Donor Program (NMDP) uses [MAC](https://bioinformatics.bethematchclinical.org/ hla-resources/allele-codes/allele-code-lists/) to facilitate the reporting and comparison of HLA alleles. MAC represent groups of HLA alleles and are useful when the HLA typing is ambiguous and does not allow to narrow down one single allele from a list of alleles.

Usage

```
decode_MAC(MAC)
```

Arguments

`MAC`	multiple allele code (e.g. "A*01:ATJNV")

Value

list of HLA alleles

Examples

```
MAC <- "A*01:ATJNV"
decode_MAC(MAC)
```

`encode_MAC` *Encode MAC*

Description

Encode a list of HLA alleles into multiple allele code (MAC). The National Marrow Donor Program (NMDP) uses [MAC](https://bioinformatics.bethematchclinical.org/ hla-resources/allele-codes/allele-code-lists/) to facilitate the reporting and comparison of HLA alleles. MAC represent groups of HLA alleles and are useful when the HLA typing is ambiguous and does not allow to narrow down one single allele from a list of alleles.

Usage

```
encode_MAC(allele_list)
```

Arguments

`allele_list`	list of HLA alleles (e.g. c("A*01:01:01", "A*02:01:01", "A*03:01"))

Value

encoded MAC

Examples

```
allele_list <- c("A*01:01:01", "A*02:01:01", "A*03:01")
encode_MAC(allele_list)
```

`extract_population_id` *extract_population_id*

Description

extract the population ids from the html result

Usage

```
extract_population_id(data)
```

Arguments

`data`	html from AFND website

Value

population ids

`extract_population_info`
extract_population_info

Description

extract_population_info

Usage

```
extract_population_info(data)
```

Arguments

`data`	html input page

Value

population information

`extract_population_name`
extract_population_name

Description

extract_population_name

Usage

```
extract_population_name(data)
```

Arguments

`data`	html input page

Value

population name

`extract_sample_info` *extract_sample_info*

Description

extract_sample_info

Usage

```
extract_sample_info(data)
```

Arguments

`data`	html input page

Value

sample information

`fetch_encoded_MAC` *fetch_encoded_MAC*

Description

fetch_encoded_MAC

Usage

```
fetch_encoded_MAC(handle)
```

Arguments

`handle`	curl handle

Value

curl handle fetch

`getURL` *getURL*

Description

getURL

Usage

```
getURL(
  URL,
  N.TRIES = 2L,
  read_method = c("delim", "lines", "html"),
  skip = 0,
  delim = "\t",
  col_names = TRUE
)
```

Arguments

`URL`	Indicated the url that will be read
`N.TRIES`	Integer, how often should the function try to read the URL?
`read_method`	Method to be used for reading of URL content ("delim" -> `readr::read_delim`, "lines" -> `readr::read_lines`, "html" -> `xml2::read_html`)
`skip`	integer indicating how many lines to skip when reading URL
`delim`	pattern used for delim (passed to `delim` of read functions)
`col_names`	list of colnames to use

Value

returns a the content of the URL. The format of the return object depends on the read_method that was used.

`get_external_file` *get_external_file*

Description

get_external_file

Usage

```
get_external_file(file, skip = 0, delim = "\t", col_names = TRUE)
```

Arguments

`file`	Indicated the file that will be read
`skip`	integer indicating how many lines to skip when reading URL
`delim`	pattern used for delim (passed to `delim` of read functions)
`col_names`	list of colnames to use

Value

returns a the content of the file. The format of the return object depends on the read_method that was used.

`get_G_group` *G groups*

Description

Get the G groups for a list of HLA alleles. [G groups](http://hla.alleles.org/alleles/g_groups.html) are groups of HLA alleles that have identical nucleotide sequences across the exons encoding the peptide binding domains.

Usage

```
get_G_group(allele_list)
```

Arguments

`allele_list`	List of alleles.

Value

Named list of G-groups the input alleles belong to.

Examples

```
allele_list <- c("DQB1*02:02:01", "DQB1*06:09:01")
get_G_group(allele_list)
```

`get_mhcpan_input` *Get format for NetMHCpan tools*

Description

NetMHCpan tools for MHC-peptide binding prediction require HLA complex names in a specific format. `get_mhcpan_input` formats a list of HLA alleles into a list of NetMHC-formated complexes.

Usage

```
get_mhcpan_input(allele_list, mhc_class)
```

Arguments

`allele_list`	list of HLA alles (e.g. c("A*01:01:01","B*27:01"))
`mhc_class`	["MHC-I"|"MHC-II"] indicated which NetMHC you want to use.

Value

protein chain list as formatted for MHCpan input

Examples

```
allele_list <- c("A*01:01:01","B*27:01")
get_mhcpan_input(allele_list, mhc_class = "MHC-I")
```

`get_nb_pages` *get_nb_pages*

Description

get_nb_pages

Usage

```
get_nb_pages(page_tbl)
```

Arguments

`page_tbl`	html input page

Value

integer number of pages in the html input

`get_P_group` *P groups*

Description

Get the P groups for a list of HLA alleles. [P groups](http://hla.alleles.org/alleles/p_groups.html) are groups of HLA alleles that have identical protein sequences in the peptide binding domains.

Usage

```
get_P_group(allele_list)
```

Arguments

`allele_list`	list of HLA alleles

Value

Named list of P-groups the input alleles belong to.

Examples

```
allele_list <- c("DQB1*02:02:01", "DQB1*06:09:01")
get_P_group(allele_list)
```

`get_serotypes` *Serotypes*

Description

Get the serotypes of the MHC complexes encoded by a list of MHC alleles.

Usage

```
get_serotypes(allele_list, organism = "human", mhc_type)
```

Arguments

`allele_list`	List of allele
`organism`	Organism to be used for MRO lookup. If the organism does not match the given allele, a empty object is returned.
`mhc_type`	["MHC-I" or "MHC-II"] MHC class to use for MRO lookup.

Value

Named list of serotypes, which only contains complexes contained in the MRO. If no serotype is annoted for a given complex, the list element is NA.

Examples

```
allele_list <- c("A*01:01:01","B*27:01")
get_serotypes(allele_list, mhc_type = "MHC-I")
```

`get_valid_geographics` *get_valid_geographics*

Description

get a list of valid countries, regions and ethnic origins

Usage

```
get_valid_geographics()
```

Value

list of valid countries, regions and ethnic origin

`get_valid_organisms` *get_valid_organisms*

Description

get the list of organisms that are part of the MRO annotation

Usage

```
get_valid_organisms()
```

Value

list of organisms

Examples

```
get_valid_organisms()
```

`human_protein_complex_table` *human_protein_complex_table*

Description

human_protein_complex_table

Usage

```
human_protein_complex_table
```

Format

An object of class `data.frame` with 12385 rows and 8 columns.

Details

`human_protein_complex_table`: human_protein_complex_table.

Examples

```
# The human protein complex table is available in the following
# exported variable
human_protein_complex_table
```

`load_mro` *load_mro*

Description

load_mro

Usage

```
load_mro()
```

Value

MRO in ontology_index format

`mro.obo` *MRO*

Description

`mro.obo`: MHC Restriction Ontology obo file.

Usage

```
mro.obo
```

Format

An object of class `ontology_index` of length 18.

`parse_allele_freq_html` *parse_allele_freq_html*

Description

format the allele frequency table and select columns of interest

Usage

```
parse_allele_freq_html(allele_freq_table)
```

Arguments

`allele_freq_table`	allele frequency table parsed from AFND website

Value

allele_freq_table reformatted

`parse_haplotype_freq_html` *parse_haplotype_freq_html*

Description

format the haplotype frequency table and select columns of interest

Usage

```
parse_haplotype_freq_html(haplotype_freq_table)
```

Arguments

`haplotype_freq_table`	haplotype frequency table parsed from AFND website

Value

haplotype_freq_table reformatted

`plot_allele_frequency` *Plotting allele frequencies*

Description

`plot_allele_frequency` Generate a World map displaying the frequency of a given table of HLA alleles. Use the function query_allele_frequencies to generate a table with allele frequencies.

Usage

```
plot_allele_frequency(allele_frequency)
```

Arguments

`allele_frequency`	returned by query_allele_frequencies

Value

ggplot2 object displaying the allele frequencies on a world map.

Examples

```
# select frequency of given allele
sel_allele_freq <- query_allele_frequencies(hla_selection = "A*02:01",
hla_sample_size_pattern = "bigger_than",
hla_sample_size = 10000, standard="g")

plot_allele_frequency(sel_allele_freq)
```

`query_allele_frequencies`

Query allele frequencies

Description

Query allele frequencies

Usage

```
query_allele_frequencies(
  hla_locus = NA,
  hla_selection = NA,
  hla_population = NA,
  hla_country = NA,
  hla_region = NA,
  hla_ethnic = NA,
  hla_sample_size_pattern = NA,
  hla_sample_size = NA,
  standard = "a"
)
```

Arguments

`hla_locus`	HLA locus that will be used for filtering data. A, B, C, DPA1, DPB1, DQA1, DQB1, DRB1
`hla_selection`	Allele that will be used for filtering data. e.g. A*01:01
`hla_population`	Numeric identifier of the population that will be used for filtering. This identifier is defined by the Allele Frequency Net Database.
`hla_country`	Country of interest (e.g. Germany, France, ...).
`hla_region`	Geographic region of interest (e.g. Europe, North Africa, ...)
`hla_ethnic`	Ethnic origin of interest (e.g. Caucasoid, Siberian, ...)
`hla_sample_size_pattern`	Keyword used to define the filtering for a specific population size. e.g. "bigger_than", "equal", "less_than", "less_equal_than", "bigger_equal_than"
`hla_sample_size`	Integer number used to define the filtering for a specific population size, together with the hla_sample_size_pattern argument.
`standard`	Population standards, as defined in the package vignette. "g" - gold, "s" - silver, "a" - all

Value

data.frame object containing the result of the allele frequency query

Examples

```
# select frequencies of the A*02:01 allele,
# for gold standard population with more than 10,000 individuals
sel <- query_allele_frequencies(hla_selection = "A*02:01",
hla_sample_size_pattern = "bigger_than", hla_sample_size = 10000,
standard="g")
```

`query_haplotype_frequencies`

Query haplotype frequencies

Description

Query haplotype frequencies

Usage

```
query_haplotype_frequencies(
  hla_selection = NA,
  hla_population = NA,
  hla_country = NA,
  hla_region = NA,
  hla_ethnic = NA,
  hla_sample_size_pattern = NA,
  hla_sample_size = NA
)
```

Arguments

`hla_selection`	Alleles that will be used to build the haplotype query. One entry per locus. If no entry for a given locus, the function will search for haplotypes that do not include specifications for this locus. If any allele for a given locus should be considered, the list entry should be "A*" or other locus in same format.
`hla_population`	Numeric identifier of the population that will be used for filtering. Thie identifier is defined by the Allele Frequency Net Database.
`hla_country`	Country of interest (e.g. Germany, France, ...).
`hla_region`	Geographic region of interest (e.g. Europe, North Africa, ...)
`hla_ethnic`	Ethnic origin of interest (e.g. Caucasoid, Siberian, ...)
`hla_sample_size_pattern`	Keyword used to define the filtering for a specific population size. e.g. "bigger_than", "equal", "less_than", "less_equal_than", "bigger_equal_than"
`hla_sample_size`	Integer number used to define the filtering for a specific population size, together with the hla_sample_size_pattern argument.

Value

data.frame object containing the result of the allele frequency query

Examples

```
# works only for one haplotype at a time
query_haplotype_frequencies(hla_selection = c("A*02:01", "B*", "C*"),
hla_region = "Europe")
```

`query_population_detail` *Query population metainformation*

Description

Query population metainformation

Usage

```
query_population_detail(population_ids)
```

Arguments

`population_ids`	List of numeric identifiers of the population that will be used for filtering. The identifier is defined by the Allele Frequency Net Database.

Value

data.frame object containing the result of the population detail query

Examples

```
population_detail <- query_population_detail(0001986)
```

`read_complete_freq_table` *read_complete_freq_table*

Description

read_complete_freq_table

Usage

```
read_complete_freq_table(url, type)
```

Arguments

`url`	URL of the website containing frequency table
`type`	["allele"|"haplotype"]

Value

frequency table

`read_population_detail` *read_population_detail*

Description

read_population_detail

Usage

```
read_population_detail(url, population_id)
```

Arguments

`url`	url of the page containing population information
`population_id`	numeric population identifier

Value

population information

`retrieve_chain_lookup_table` *Retrieve MHC chain lookup table*

Description

Retrieve MHC chain lookup table

Usage

```
retrieve_chain_lookup_table(organism)
```

Arguments

`organism`	name of organism (e.g. "human")

Value

Table containing MHC chain information for the organism. It contains chain names, MHC restriction and protein sequence.

Examples

```
retrieve_chain_lookup_table("mouse")
```

`verify_parameters` *verify_parameters*

Description

verify_parameters

Usage

```
verify_parameters(
  hla_locus,
  hla_selection,
  hla_population,
  hla_country,
  hla_region,
  hla_ethnic,
  hla_sample_size_pattern,
  hla_sample_size,
  standard = "a",
  query_type
)
```

Arguments

`hla_locus`	locus of hla frequencies to query
`hla_selection`	HLA alleles used for selection
`hla_population`	population id
`hla_country`	country used for allele frequency selection
`hla_region`	geographical region used for allele frequency selection
`hla_ethnic`	ethical origin used for allele frequency selection
`hla_sample_size_pattern`	one of "bigger_than", "equal", "less_than", "less_equal_than", "bigger_equal_than","different"
`hla_sample_size`	integer number used for population size
`standard`	one of "a" - all,"s" - silver,"g" - gold
`query_type`	"allele" or "haplotype"

Value

boolean to indicate, whether tests were passed

Index