Package ‘igblastr’

January 24, 2026

Title User-friendly R Wrapper to IgBLAST

Description The igblastr package provides functions to conveniently install and use a local IgBLAST installation from within R.
IgBLAST is described at <https://pubmed.ncbi.nlm.nih.gov/23671333/>.
IgBLAST web interface: <https://www.ncbi.nlm.nih.gov/igblast/>.

biocViews Immunology, Immunogenetics, ImmunoOncology, CellBiology

URL https://bioconductor.org/packages/igblastr

BugReports https://github.com/HyrienLab/igblastr/issues

Version 1.1.11

License Artistic-2.0

Encoding UTF-8

Depends R (>= 4.2.0), tibble, Biostrings

Imports methods, utils, stats, tools, R.utils, curl, httr, xml2, rvest, xtable, jsonlite, S4Vectors, IRanges, GenomeInfoDb (>= 1.47.2)

Suggests GenomicAlignments, parallel, testthat, knitr, rmarkdown, BiocStyle, ggplot2, dplyr, scales, ggseqlogo

VignetteBuilder knitr

Collate utils.R long_to_wide_airr.R translate_codons.R allele2gene.R compute_intdata.R compute_auxdata.R makeogrannote.R file-utils.R loci-utils.R db-utils.R LATIN_NAMES.R IMGT-utils.R IMGT-c_region-utils.R AIRR-utils.R precompiled-igblast-utils.R cache-utils.R get_igblast_root.R edit_imgt_file.R igblast_info.R update_live_igdata.R intdata-utils.R auxdata-utils.R install_igblast.R make_blastdbs.R create_region_db.R create_germline_db.R create_c_region_db.R builtin_db-utils.R list_germline_dbs.R list_c_region_dbs.R install_IMGT_germline_db.R install_AIRR_germline_db.R augment_germline_db.R igblastn-args-utils.R read_igblastn_fmt7_output.R read_igblastn_AIRR_output.R igblastn.R igbrowser.R summarizeMismatches.R OAS-utils.R zzz.R

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

git_branch devel

git_last_commit dcf8759

git_last_commit_date 2026-01-23

Repository Bioconductor 3.23

Date/Publication 2026-01-23

Author Hervé Pagès [aut, cre] (ORCID: <https://orcid.org/0009-0002-8272-4522>),
Ollivier Hyrien [aut, fnd] (ORCID:
<https://orcid.org/0000-0003-1909-2542>),
Kellie MacPhee [ctb] (ORCID: <https://orcid.org/0009-0008-0993-4009>),
Michael Duff [ctb] (ORCID: <https://orcid.org/0009-0008-4279-0756>),
Jason Taylor [ctb]

Maintainer Hervé Pagès <hpages.on.github@gmail.com>

Contents

allele2gene 3
augment_germline_db 4
auxdata-utils 7
compute_auxdata 11
compute_intdata 13
get_igblast_root 14
igblastn 15
igblastr_usage_report 21
igblast_info 23
IGBLAST_ROOT 25
igbrowser 26
install_igblast 27
install_IMGT_germline_db 28
intdata-utils 31
list_c_region_dbs 35
list_germline_dbs 37
makeogrannote 41
OAS-utils 42
read_igblastn_AIRR_output 45
read_igblastn_fmt7_output 46
summarizeMismatches 49
translate_codons 49
update_live_igdata 51

Index **54**

allele2gene *Go from germline gene allele names to germline gene names*

Description

A simple convenience function to remove the allele suffix from a vector of germline gene allele names.

Usage

```
allele2gene(allele_names)
```

Arguments

allele_names	A character vector of germline gene allele names.

Details

Germline gene allele names use specific nomenclature, often including a gene name followed by an asterisk and allele number, like in IGHD3-16*03.

The allele2gene() function simply removes the asterisk and anything that follows it to keep the gene name only. Note that the function is *vectorized*, that is, its input can be a *vector* of germline gene allele names, in which case the corresponding germline gene names are returned in a vector of the same length as the input vector. The names on the input vector are propagated, and so are the NA's in it.

Value

A character vector *parallel* to the input vector.

See Also

- load_germline_db to load the nucleotide sequences of the gene regions stored in a cached germline db.
- load_c_region_db to load the nucleotide sequences of the gene regions stored in a cached C-region db.
- intdata_utils to access and manipulate IgBLAST *internal data*.

Examples

```
allele2gene(c("IGHV1-2*04", "IGHV1-2*06", "IGHV5-51*01", "IGHD3-16*03"))

J_alleles <- load_germline_db("_AIRR.human.IGH+IGK+IGL.202410",
                              region_types="J")
names(J_alleles)
allele2gene(names(J_alleles))
```

```
C_alleles <- load_c_region_db("_IMGT.human.IGH+IGK+IGL.202412")
names(C_alleles)
allele2gene(names(C_alleles))
```

augment_germline_db *Add novel gene alleles to a germline db*

Description

Three functions to add novel V, D, or J gene alleles to a germline db.

Note that these functions can also be used to combine germline databases from two different organisms. See "COMBINE GERMLINE DATABASES FROM TWO ORGANISMS" in the Examples section below for how to do this.

Usage

```
augment_germline_db_V(db_name, novel_alleles, destdir=".", overwrite=FALSE)
augment_germline_db_D(db_name, novel_alleles, destdir=".", overwrite=FALSE)
augment_germline_db_J(db_name, novel_alleles, destdir=".", overwrite=FALSE)
```

Arguments

`db_name` A single string that is the name of the cached germline db that contains the set of gene alleles to augment. Use `list_germline_dbs()` to list the cached germline dbs.

The exact function used (i.e. `augment_germline_db_V()`, `augment_germline_db_D()`, or `augment_germline_db_J()`) determines the set of alleles to augment (i.e. alleles from the V, D, or J region).

`novel_alleles` A single string that is the path to a FASTA file (possibly gz-compressed) where the novel alleles are stored.

Alternatively, the novel alleles can be supplied as a *named* DNAStringSet object.

`destdir` A single string that is the path to the "destination directory", that is, the directory where the augmented V-, D-, or J-region db is to be created. This directory will be created if it doesn't exist already. Note that, by default, the augmented region db will be created in the current directory.

`overwrite` If the "destination directory" already contains a V-, D-, or J-region db, should it be overwritten?

Value

These functions don't return anything (invisible `NULL`).

See Also

- The `igblastn` function to run the `igblastn` *standalone executable* included in IgBLAST from R. This is the main function in the **igblastr** package.
- `list_germline_dbs` to list the cached germline dbs.
- IgBLAST is described at `https://pubmed.ncbi.nlm.nih.gov/23671333/`.

Examples

```
if (!has_igblast()) install_igblast()

query <- system.file(package="igblastr", "extdata",
                     "BCR", "heavy_sequences.fasta")

use_c_region_db("_IMGT.human.IGH+IGK+IGL.202412")

## ---------------------------------------------------------------------
## USE HUMAN GERMLINE DATABASE FROM AIRR
## ---------------------------------------------------------------------

use_germline_db("_AIRR.human.IGH+IGK+IGL.202410")

AIRR_df <- igblastn(query)

## ---------------------------------------------------------------------
## ADD NOVEL V ALLELES
## ---------------------------------------------------------------------

## 'fake_human_V_alleles.fasta' contains made-up novel V alleles:
## - 2 novel alleles for gene IGHV1-8: IGHV1-8*fake1, IGHV1-8*fake2
## - 1 novel allele for gene IGHV4-61: IGHV4-61*fake
my_novel_V_alleles <- system.file(package="igblastr", "extdata",
                                  "novel_germline_alleles",
                                  "fake_human_V_alleles.fasta")

## Take a quick look at these novel V alleles:
readDNAStringSet(my_novel_V_alleles)

## Create a new V germline database that combines the V alleles
## from _AIRR.human.IGH+IGK+IGL.202410 with our novel V alleles:
myVdb_path <- file.path(tempdir(), "myVdb")
augment_germline_db_V("_AIRR.human.IGH+IGK+IGL.202410",
                      my_novel_V_alleles,
                      destdir=myVdb_path)

## To use this new augmented V germline database with igblastn(),
## supply its path via the 'germline_db_V' argument:
AIRR_df2 <- igblastn(query, germline_db_V=myVdb_path)

## ---------------------------------------------------------------------
## A QUICK COMPARISON BETWEEN 'AIRR_df' AND 'AIRR_df2'
## ---------------------------------------------------------------------

## Index of rows where "v_call" has changed between 'AIRR_df'
## and 'AIRR_df2':
idx <- which(AIRR_df$v_call != AIRR_df2$v_call)
idx  # 2 rows

AIRR_df[idx, c("v_call", "v_cigar", "v_identity")]
```

```
AIRR_df2[idx, c("v_call", "v_cigar", "v_identity")]

## Besides these 2 rows, all the other rows are the same:
stopifnot(all.equal(AIRR_df[-idx, ], AIRR_df2[-idx, ]))

## ---------------------------------------------------------------------
## COMBINE GERMLINE DATABASES FROM TWO ORGANISMS
## ---------------------------------------------------------------------

## The augment_germline_db_[VDJ]() functions can be used to combine
## germline databases from two different organisms. This can be useful
## for example when working with BCR sequences from mice that have been
## engineered to have both mouse and some human immunoglobulin genes.
##
## To create a hybrid human/mouse V germline database, we can either:
##
## (1) Add all (or a subset of) mouse V alleles to all human V alleles.
##     This is done by extracting mouse V germline allele sequences from
##     a cached germline database and using them to augment a cached
##     germline database for human.
##
## (2) Add all (or a subset of) human V alleles to all mouse V alleles.
##     This is done by extracting human V germline allele sequences from
##     a cached germline database and using them to augment a cached
##     germline database for mouse.
##
## Note that:
## - We can choose to subset or not the V germline allele sequences
##   extracted from one V germline database before adding them to the
##   other V germline database.
## - The two approaches above are equivalent if we don't subset, that
##   is, if we combine **all** human V alleles with **all** mouse V
##   alleles.
## - However if our engineered mice only have a small known subset of
##   human immunoglobulin genes (e.g. IGHV1-2), then we might want to
##   create a hybrid human/mouse germline database that only adds the
##   human alleles for genes IGHV1-2 to the mouse V alleles. In this
##   case we need to use (2).

## Let's do (2):

db_name1 <- "_AIRR.mouse.PWD_PhJ.IGH+IGK+IGL.202501"
db_name2 <- "_AIRR.human.IGH+IGK+IGL.202410"

## Extract human V germline alleles:
human_V_alleles <- load_germline_db(db_name2, "V")

## Subset to keep only alleles for genes IGHV1-2:
idx <- grep("^IGHV[12]", names(human_V_alleles))
human_V12_alleles <- human_V_alleles[idx]

## Create a new V germline database that combines the mouse V
## alleles from 'db_name1' with the alleles in 'human_V12_alleles':
```

```
engmouseVdb_path <- file.path(tempdir(), "engmouseVdb")
augment_germline_db_V(db_name1, human_V12_alleles,
                      destdir=engmouseVdb_path)

## Then, assuming that 'query' contains BCR sequences from the
## engineered mice:
## Not run:
  use_germline_db(db_name1)
  use_c_region_db("_IMGT.mouse.IGH.202509")
  igblastn(query, germline_db_V=engmouseVdb_path, ...)

## End(Not run)

## Note that, by default, the mouse-only D and J databases that we
## selected above with 'use_germline_db(db_name1)' are being used.
## If we also want to create hybrid D and J databases, we need
## to repeat the above steps for each of them. Then we need to
## specify the paths to the 3 hybrid databases when we call igblastn():
## Not run:
  igblastn(query, germline_db_V=engmouseVdb_path,
                  germline_db_D=engmouseDdb_path,
                  germline_db_J=engmouseJdb_path,
                  ...)

## End(Not run)
```

auxdata-utils *Access, manipulate, and generate IgBLAST auxiliary data*

Description

IgBLAST *auxiliary data* is expected to annotate all the known germline J gene alleles for a given organism. It is provided by NCBI and is typically included in a standard IgBLAST installation.

The **igblastr** package provides a small set of utilities to access, manipulate, and generate IgBLAST *auxiliary data*.

Usage

```
## Access auxiliary data:
get_auxdata_path(organism, which=c("live", "original"))
load_auxdata(organism, which=c("live", "original"))

## Manipulate auxiliary data:
translate_J_alleles(J_alleles, auxdata)
J_allele_has_stop_codon(J_alleles, auxdata)
translate_fwr4(J_alleles, auxdata, max.codons=NA)
```

Arguments

`organism` A single string containing the name of an organism as returned by `list_igblast_organisms()`.

`which` By default, `get_auxdata_path()` and `load_auxdata()` access the "live IgBLAST data", that is, the IgBLAST data that the user has possibly updated with `update_live_igdata()`. Depending on whether updates were applied or not, the "live IgBLAST data" might differ from the original IgBLAST data.

Set `which` to `"original"` if you want to access the original IgBLAST data instead.

See ?`update_live_igdata` for more information about "live" and "original" IgBLAST data.

`J_alleles` A DNAStringSet object containing germline J gene allele sequences.

`auxdata` A data.frame as returned by `load_auxdata()` or `compute_auxdata()`.

`max.codons` The maximum number of FWR4 codons to translate. By default (i.e. when `max.codons` is `NA`) all the FWR4 codons are translated.

Details

IgBLAST *auxiliary data* is typically included in a standard IgBLAST installation. It's located in the `optional_file/` directory which is itself a subdirectory of IgBLAST *root directory*.

The data consists of one tabulated file per organism. Each file indicates the germline J gene coding frame start position, the J gene type, and the CDR3 end position for all known germline J allele sequences. See `https://ncbi.github.io/igblast/cook/How-to-set-up.html` for additional details.

Value

`get_auxdata_path()` returns a single string containing the path to the *auxiliary data* included in the IgBLAST installation used by **igblastr**, for the specified organism. Not necessarily suitable to use with `igblastn()` (see WARNING below).

`load_auxdata()` returns the *auxiliary data* in a data.frame with 1 row per germline J allele sequence and the following columns:

1. `allele_name`: allele name;
2. `coding_frame_start`: first coding frame start position (0-based);
3. `chain_type`: chain type;
4. `cdr3_end`: CDR3 end position (0-based);
5. `extra_bps`: extra base pairs beyond J coding end.

`translate_J_alleles()` returns a named character vector with 1 amino acid sequence per supplied allele. The vector contains an `NA` for any allele that is not annotated in `auxdata` or for which `auxdata$coding_frame_start` has an `NA`. The names on it are the names of the supplied alleles.

`J_allele_has_stop_codon()` returns a named logical vector with 1 value per supplied allele. The vector contains an `NA` for any allele that is not annotated in `auxdata` or for which `auxdata$coding_frame_start` has an `NA`. The names on it are the names of the supplied alleles.

`translate_fwr4()` returns a named character vector with 1 amino acid sequence per supplied allele. The vector contains an `NA` for any allele that is not annotated in `auxdata` or for which `auxdata$cdr3_end` has an `NA`.

WARNING

According to https://ncbi.github.io/igblast/cook/How-to-set-up.html the *auxiliary data* included in IgBLAST is specific to a particular NCBI or IMGT germline db. Unfortunately this means that this data is NOT guaranteed to be compatible with the germline db that you will use with igblastn(). See documentation of the auxiliary_data argument in ?igblastn for more information about this.

See Also

- compute_auxdata to annotate a set of germline J gene allele sequences.
- intdata_utils to access and manipulate IgBLAST *internal data*.
- update_live_igdata for more information about "live" and "original" IgBLAST data.
- https://ncbi.github.io/igblast/cook/How-to-set-up.html for important information about IgBLAST *auxiliary data*.
- DNAStringSet objects in the **Biostrings** package.
- The translate_codons function on which translate_J_alleles() and translate_fwr4() are based.
- The igblastn function to run the igblastn *standalone executable* included in IgBLAST from R. This is the main function in the **igblastr** package.
- IgBLAST is described at https://pubmed.ncbi.nlm.nih.gov/23671333/.

Examples

```
if (!has_igblast()) install_igblast()

igblast_info()

## ---------------------------------------------------------------------
## 1. Access and load IgBLAST auxiliary data for a given organism
## ---------------------------------------------------------------------

list_igblast_organisms()

get_auxdata_path("human")
human_auxdata <- load_auxdata("human")
head(human_auxdata)

## ---------------------------------------------------------------------
## 2. A close look at IgBLAST auxiliary data for rabbit
## ---------------------------------------------------------------------

get_auxdata_path("rabbit")
rabbit_auxdata <- load_auxdata("rabbit")

## It turns out that IgBLAST auxiliary data for rabbit matches exactly
## the set of rabbit germline J alleles available at IMGT:
db_name <- install_IMGT_germline_db("202531-1", "Oryctolagus cuniculus",
                                    force=TRUE)
```

```
J_alleles <- load_germline_db(db_name, region_types="J")
J_alleles  # DNAStringSet object
stopifnot(setequal(names(J_alleles), rabbit_auxdata$allele_name))

## Note that this might change with future IMGT releases.

## Let's put the allele sequences in 'J_alleles' in the same order as
## in 'rabbit_auxdata':
J_alleles <- J_alleles[rabbit_auxdata$allele_name]
stopifnot(identical(names(J_alleles), rabbit_auxdata$allele_name))

## The 'coding_frame_start' column in 'rabbit_auxdata' contains integer
## values that are >= 0 and <= 2. They indicate how many nucleotides
## precede the first codon on each allele sequence. In other words,
## this is the number of nucleotides that we need to trim on the 5'
## end of the germline J allele sequence before we start translating
## it. translate_J_alleles() uses this information to translate the
## DNA sequences in 'J_alleles':
J_aa <- translate_J_alleles(J_alleles, rabbit_auxdata)
J_aa

## No sequence in 'J_aa' should contain the letter "*" which is used
## by translate_J_alleles() to represent a stop codon. However, one
## J allele in IMGT-202531-1.Oryctolagus_cuniculus.IGH+IGK+IGL seems
## to disobey:
has_stop_codon <- grepl("*", J_aa, fixed=TRUE)
rabbit_auxdata[has_stop_codon, ]  # coding_frame_start = 0 for IGKJ1-2*04
J_alleles[has_stop_codon]         # first codon (TGA) is a stop codon
J_aa[has_stop_codon]              # indeed!

## ---------------------------------------------------------------------
## 3. About the "WGXG" and "FGXG" motifs
## ---------------------------------------------------------------------

## The FWR4 region is expected to start with the following amino acid
## motifs (X represents any amino acid):
##   - "WGXG" on the heavy chain
##   - "FGXG" on the light chain

## Let's use translate_fwr4() to extract and translate the first 4
## codons of the FWR4 region:
fwr4_head <- translate_fwr4(J_alleles, rabbit_auxdata, max.codons=4)

## We expect to see the "WGXG" and "FGXG" motifs here, and most of the
## time we do:
has_motif <- grepl("[FW]G.G", fwr4_head)
table(has_motif)

## However, there are a few exceptions:
fwr4_head[!has_motif]

## ---------------------------------------------------------------------
## 4. Compute auxiliary data for a set of J allele sequences
## ----------------------------------------------------------------------

## See '?compute_auxdata' for the details of what compute_auxdata() does.
computed_auxdata <- compute_auxdata(J_alleles)
head(computed_auxdata)

## Alleles for which the CDR3 end could not found:
cdr3_end_not_found <- is.na(computed_auxdata$cdr3_end)
stopifnot(identical(cdr3_end_not_found, !has_motif))
J_alleles[cdr3_end_not_found]
fwr4_head[cdr3_end_not_found]  # déjà vu

## 'computed_auxdata' is in agreement with 'rabbit_auxdata', except for
## the 8 alleles for which the CDR3 end could not be found:
keep_idx <- which(!cdr3_end_not_found)
stopifnot(identical(computed_auxdata[keep_idx, ],
                    rabbit_auxdata[keep_idx, ]))
```

`compute_auxdata` *Compute IgBLAST auxiliary data*

Description

A utility function to annotate a set of germline J gene allele sequences in a way similar to how they are annotated in IgBLAST *auxiliary data*.

Note that the annotation produced by the function can be used by IgBLAST as a substitute to the *auxiliary data* shipped with the IgBLAST software (and typically included in a standard IgBLAST installation).

See ?`load_auxdata` for more information about IgBLAST *auxiliary data*.

Usage

```
compute_auxdata(J_alleles)
```

Arguments

`J_alleles`	A DNAStringSet object containing germline J gene allele sequences.

Details

The FWR4 region is expected to start with the following amino acid motifs (X represents any amino acid):

- "WGXG" on the heavy chain;
- "FGXG" on the light chain.

compute_auxdata() searches for the "WGXG" and "FGXG" motifs in the supplied allele sequences to determine the start of their FWR4 region. From there it can easily infer the cdr3_end, coding_frame_start, and extra_bps columns.

Note that the function will emit a warning if the start of the FWR4 region (and therefore the CDR3 end) could not be found for some alleles, or if a stop codon was found in some alleles.

Value

Returns the computed *auxiliary data* in a data.frame with 1 row per supplied germline J allele sequence and the same columns as the data.frame returned by load_auxdata().

See Also

- load_auxdata to access and manipulate IgBLAST *auxiliary data*.
- compute_intdata to annotate a set of germline V gene allele sequences.
- https://ncbi.github.io/igblast/cook/How-to-set-up.html for important information about IgBLAST *auxiliary data*.
- DNAStringSet objects in the **Biostrings** package.
- The igblastn function to run the igblastn *standalone executable* included in IgBLAST from R. This is the main function in the **igblastr** package.
- IgBLAST is described at https://pubmed.ncbi.nlm.nih.gov/23671333/.

Examples

```
## ---------------------------------------------------------------------
## BASIC EXAMPLE
## ---------------------------------------------------------------------

## Let's load a set of human J allele sequences:
db_name <- "_AIRR.human.IGH+IGK+IGL.202410"
J_alleles <- load_germline_db(db_name, region_types="J")
J_alleles  # DNAStringSet object

computed_auxdata <- compute_auxdata(J_alleles)
head(computed_auxdata)

## ---------------------------------------------------------------------
## SANITY CHECK
## ---------------------------------------------------------------------

## Note that 'computed_auxdata' is in agreement with the auxiliary
## data included in IgBLAST for human, except for alleles IGHJ6*02
## and IGHJ6*03 (the 'extra_bps' column contains incorrect values
## for these two alleles, 0's instead of 1's):
human_auxdata0 <- load_auxdata("human", which="original")
bad_alleles <- c("IGHJ6*02", "IGHJ6*03")
subset(human_auxdata0, allele_name %in% bad_alleles)

## We manually correct this:
```

```
fixme <- human_auxdata0[ , "allele_name"] %in% bad_alleles
human_auxdata0[fixme, "extra_bps"] <- 1L  # replace 0L with 1L

## Now the data in 'computed_auxdata' matches exactly the corresponding
## data in 'human_auxdata0':
m <- match(computed_auxdata[ , "allele_name"],
           human_auxdata0[ , "allele_name"])
human_auxdata <- human_auxdata0[m, ]
rownames(human_auxdata) <- NULL
stopifnot(identical(computed_auxdata, human_auxdata))
```

compute_intdata *Compute IgBLAST internal data*

Description

A utility function to annotate a set of germline V gene allele sequences in a way similar to how they are annotated in IgBLAST *internal data*.

Note that the annotation produced by the function can be used by IgBLAST as a substitute to the *internal data* shipped with the IgBLAST software (and typically included in a standard IgBLAST installation).

See ?`load_intdata` for more information about IgBLAST *internal data*.

Usage

```
compute_intdata(V_alleles)
```

Arguments

`V_alleles`	A DNAStringSet object containing germline V gene allele *gapped* sequences.

Details

COMING SOON...

Value

Returns the computed *internal data* in a data.frame with 1 row per supplied germline V allele sequence and the same columns as the data.frame returned by `load_intdata()`.

See Also

- `load_intdata` to access and manipulate IgBLAST *internal data*.
- `compute_auxdata` to annotate a set of germline J gene allele sequences.
- DNAStringSet objects in the **Biostrings** package.
- The `igblastn` function to run the `igblastn` *standalone executable* included in IgBLAST from R. This is the main function in the **igblastr** package.
- IgBLAST is described at `https://pubmed.ncbi.nlm.nih.gov/23671333/`.

Examples

```
## COMING SOON...
```

`get_igblast_root` *Control IgBLAST installation to use*

Description

Get (or set) the IgBLAST installation used (or to be used) by the **igblastr** package.

Usage

```
get_igblast_root()
set_igblast_root(version_or_path)
```

Arguments

`version_or_path`
A single string that is either a version number (e.g. `"1.22.0"`) or the path to an IgBLAST installation.

Details

`set_igblast_root` can be used to set or change the path to the IgBLAST installation to use. This can be an *internal* or *external* installation.

In the former case, `version_or_path` should be the version of an existing *internal* installation. The setting will be persistent.

In the latter case, it should be the full path (absolute or relative) to the *root directory* of a valid *external* installation. Note that the setting won't be persistent i.e. it won't be remembered across R sessions. See `?IGBLAST_ROOT` for how to set the *external* IgBLAST installation to use in **igblastr** in a persistent manner.

Value

`get_igblast_root()` returns a single string containing the path to the *root directory* of the IgBLAST installation used by **igblastr**.

`set_igblast_root()` returns a single string containing the path to the *root directory* of the newly selected IgBLAST installation. The string is returned invisibly.

See Also

- The `igblastn` function to run the `igblastn` *standalone executable* included in IgBLAST from R. This is the main function in the **igblastr** package.
- `install_igblast` to perform an *internal* IgBLAST installation.
- `igblast_info` to collect basic information about the IgBLAST installation used by the **igblastr** package.

- IGBLAST_ROOT to set the *external* IgBLAST installation to be used by the **igblastr** package in a persistent manner.
- IgBLAST is described at `https://pubmed.ncbi.nlm.nih.gov/23671333/`.

Examples

```
if (!has_igblast()) install_igblast()

get_igblast_root()
```

`igblastn` *BLAST for BCR/Ig and TCR sequences*

Description

The `igblastn()` function is a wrapper to the `igblastn` *standalone executable* included in IgBLAST. This is the main function in the **igblastr** package.

Usage

```
igblastn(query, outfmt="AIRR",
         germline_db_V="auto", germline_db_V_seqidlist=NULL,
         germline_db_D="auto", germline_db_D_seqidlist=NULL,
         germline_db_J="auto", germline_db_J_seqidlist=NULL,
         organism="auto", c_region_db="auto",
         custom_internal_data="auto", auxiliary_data="auto",
         domain_system=c("imgt", "kabat"), ig_seqtype="auto",
         ...,
         out=NULL, parse.out=TRUE,
         show.in.browser=FALSE, show.command.only=FALSE)

igblastn_help(long.help=FALSE, show.in.browser=FALSE)
```

Arguments

`query` A character vector containing the paths to the input files (FASTA), or a *named* DNAStringSet object.

If a character vector, then `query` must be of length >= 1 and each vector element must be the path to a FASTA file (possibly gz-compressed). In the context of IgBLAST, the DNA sequences in the FASTA files are referred to as *the query sequences*, and the sequence names found in the description lines of the FASTA records are referred to as *the query sequence ids*.

Note that the query sequences are typically (but not always) stored in a single file, in which case `query` will be a single string. If more than one FASTA file is specified via `query`, then `igblastn()` will concatenate all the files together and pass the resulting file to the `igblastn` *standalone executable*.

If `query` is a DNAStringSet object, then it must have names on it. These will be considered the query sequence ids.

`outfmt` One of `"AIRR"`, `3`, `4`, `7`, or `19`. `"AIRR"` is the default and is an alias for `19`. `outfmt` can also be a string describing a customized format 7 e.g. `"7 qseqid sseqid pident nident length score"`.

See ?`list_outfmt7_specifiers` for more information about customizing format 7.

`germline_db_V` `"auto"` (the default), or the path to a V-region db.

Note that, by default (i.e. when `germline_db_V` is omitted or set to `"auto"`), `igblastn()` uses the V-region db that belongs to the cached germline db that is currently selected.

See ?`use_germline_db` for how to select the cached germline db to use with `igblastn()`.

`germline_db_D` Same as `germline_db_V` but for the D-region db.

`germline_db_J` Same as `germline_db_V` but for the J-region db.

`germline_db_V_seqidlist`, `germline_db_D_seqidlist`, `germline_db_J_seqidlist`

Restrict search of germline database to list of gene alleles. A list of gene alleles can be specified either as a character vector of gene allele identifiers (e.g. `IGHV3-23*01`, `IGHV3-23*04`, etc...) or as the path to a file containing the identifiers (one identifier per line). In the latter case, a `file` object must be passed to the `germline_db_V_seqidlist`, `germline_db_D_seqidlist`, or `germline_db_J_seqidlist` argument. The `file` object will typically be constructed with something like `file("path/to/some/file")`.

`organism` `"auto"` (the default), or the organism associated with the query sequences. Supported organisms include human, mouse, rat, rabbit and rhesus_monkey. Use `list_igblast_organisms()` to obtain this list programmatically.

Note that, by default (i.e. when `organism` is omitted or set to `"auto"`), `igblastn()` infers the organism from the name of the cached germline db that is currently selected.

See ?`use_germline_db` for how to select the cached germline db to use with `igblastn()`.

`c_region_db` `"auto"` (the default), `NULL`, or the path to a C-region db.

Note that, by default (i.e. when `c_region_db` is omitted or set to `"auto"`), `igblastn()` uses the cached C-region db that is currently selected.

See ?`use_c_region_db` for how to select the cached C-region db to use with `igblastn()`.

`custom_internal_data`

`"auto"` (the default), or the path to a file containing custom FWR/CDR annotation, or `NULL`.

Note that, by default (i.e. when `custom_internal_data` is omitted or set to `"auto"`), `igblastn()` will use:

- the *internal data* included in the cached germline db that is currently selected, if it has any;
- the organism-specific IgBLAST *internal data* from NCBI if the cached germline db that is currently selected does not include its own *internal data*.

Use `list_germline_dbs(with.intdata.only=TRUE)` to list the cached germline dbs that include their own *internal data*.

See ?`use_germline_db` for how to select the cached germline db to use with `igblastn()`.

Finally, when `custom_internal_data` is set to `NULL`, `igblastn()` always uses the organism-specific IgBLAST *internal data* from NCBI.

`auxiliary_data` `"auto"` (the default), or the path to a file containing the coding frame start positions for the sequences in the J-region db, or `NULL`.

Note that, by default (i.e. when `auxiliary_data` is omitted or set to `"auto"`), `igblastn()` uses one of the auxiliary data files included in the IgBLAST installation used by **igblastr**. More precisely, `igblastn()` uses `get_auxdata_path()` internally to obtain the path to the organism-specific auxiliary data file.

IMPORTANT NOTES:

- Supplying *auxiliary data* that is not compatible with the V gene sequences of the selected germline db can cause `igblastn()` to return improper frame status or CDR3 information (other returned information will still be correct). See ?`get_auxdata_path` for more information.
- When `auxiliary_data` is set to `NULL`, then no *auxiliary data* is used. In this case, `igblastn()` can emit a significant number of the following warning:

  ```
  Warning: Auxilary data file could not be found
  ```

  and various columns of the returned AIRR-formatted tibble (e.g. columns `vj_in_frame`, `productive`, `cdr3`, `fwr4`, and others) will be filled with `NA`s.

`domain_system` Set to `"imgt"` or `"kabat"`.

`ig_seqtype` Set to `"Ig"` or `"TCR"` depending on whether the query sequences are BCR/Ig or TCR sequences.

Note that, by default (i.e. when `ig_seqtype` is omitted or set to `"auto"`), the value of `ig_seqtype` is inferred from the germline loci that appear in the name of the cached germline db that is currently selected (this name can be obtained with `use_germline_db`). If these are BCR/Ig germline loci (i.e. IGH, IGK, IGL), then the inferred value will be `"Ig"`. If they are TCR germline loci (TRA, TRB, TRG, TRD), then it will be `"TCR"`.

See ?`use_germline_db` for how to select the cached germline db to use with `igblastn()`.

`...` Extra arguments to be passed to the `igblastn` *standalone executable*. The list of valid arguments can be displayed with `igblastn_help()`.

Note that the argument/value pairs must be passed to the `igblastn()` function in the usual R fashion. For example, what would be passed as `-num_alignments 1 -num_threads 8` when invoking the `igblastn` *standalone executable* in a terminal should be passed as `num_alignments_V=1, num_threads=8` when calling the `igblastn()` function:

```
igblastn(query, num_alignments_V=1, num_threads=8)
```

For options that don't require a value (e.g. `-extend_align5end`, `-extend_align3end`, `-ungapped`, etc...), pass the empty string (or a white string) to the argument. For example:

```
igblastn(query, extend_align5end="", extend_align3end="")
```

`out` — `NULL` (the default), or the path to the file where the `igblastn` *standalone executable* should write its output.

Note that, by default (i.e. when `out` is omitted or set to `NULL`), `igblastn()` instructs the `igblastn` *standalone executable* to write its output to a temporary file.

`parse.out` — Whether `igblastn()` should parse the plain-text output produced by the `igblastn` *standalone executable* or not, before returning it to the user. `TRUE` by default.

If set to `FALSE`, then `igblastn()` returns the output as-is in a character vector, with one line per element in the vector. Note that `igblastn()` sets the `"igblastn_raw_output"` class attribute on this character vector, which allows compact display of the vector (this is achieved via a dedicated `print()` method defined in the **igblastr** package). The class attribute can be dropped with `unclass()`.

`show.in.browser`

For `igblastn()`: Whether the output of the `igblastn` *standalone executable* should also be displayed in a browser or not (in addition to being returned by the `igblastn()` function call). `FALSE` by default.

For `igblastn_help()`: Whether the help printed by the `igblastn` *standalone executable* (when invoked with the -h or -help argument) should be displayed in a browser or not. `FALSE` by default.

`show.command.only`

`TRUE` or `FALSE`. If set to `TRUE`, `igblastn()` won't invoke the `igblastn` *standalone executable* and instead will display the full command that shows how it would have invoked it. Note that the command is also returned in an invisible character vector. `FALSE` by default.

`long.help` — `TRUE` or `FALSE`. If set to `FALSE` (the default), the `igblastn` *standalone executable* is invoked with the -h argument. Otherwise, it's invoked with the -help argument.

Value

`igblastn()` captures the output produced by the `igblastn` *standalone executable* and returns it as:

- A tibble with 1 row per query sequence if `outfmt` is `"AIRR"` or `19` and `parse.out` is `TRUE`.
- A nested list with two top-level components (`records` and `footer`) if `outfmt` is `7` (or a customized format 7) and `parse.out` is `TRUE`. See `?read_igblastn_fmt7_output` for more information.
- A character vector with class attribute `"igblastn_raw_output"` on it in all other cases, that is, if `parse.out` is `FALSE` or `outfmt` is `3` or `4`. See the `parse.out` argument above for more information.

Note

By default, the NCBI BLAST+ and IgBLAST programs will "call home" to report usage when they run on a computer with internet access. See `https://www.ncbi.nlm.nih.gov/books/NBK569851/` for the details. This can induce a significant slowdown in some situations e.g. when the `igblastn` *standalone executable* is called in a loop on a small set of query sequences at each iteration.

For this reason, the "call home" feature is disabled in **igblastr** by default, unless environment variable BLAST_USAGE_REPORT is set to true. See ?igblastr_usage_report for more information.

See Also

- IgBLAST is described at https://pubmed.ncbi.nlm.nih.gov/23671333/.
- install_igblast to perform an *internal* IgBLAST installation.
- igblast_info to collect basic information about the IgBLAST installation used by the **igblastr** package.
- install_IMGT_germline_db to install a germline db from IMGT.
- use_germline_db to select the cached germline db to use with igblastn().
- use_c_region_db to select the cached C-region db to use with igblastn().
- igbrowser to display the annotated sequences returned by igblastn() in a browser.
- list_outfmt7_specifiers for how to customize output format 7.
- list_igblast_organisms to list the organisms supported by IgBLAST.
- augment_germline_db to add novel gene alleles to a germline db.
- igblastr_usage_report to turn "Usage Reporting" on or off.
- DNAStringSet objects implemented in the **Biostrings** package.
- tibble objects implemented in the **tibble** package.

Examples

```
if (!has_igblast()) install_igblast()

igblast_info()

## ---------------------------------------------------------------------
## Access query sequences and select germline and C-region dbs to use
## ---------------------------------------------------------------------

## Files 'heavy_sequences.fasta' and 'light_sequences.fasta' included
## in igblastr contain 250 paired heavy- and light- chain sequences (125
## sequences in each file) downloaded from OAS (the Observed Antibody
## Space database):
filenames <- paste0(c("heavy", "light"), "_sequences.fasta")
query <- system.file(package="igblastr", "extdata", "BCR", filenames)

## Install Human germline db from IMGT:
db_name <- install_IMGT_germline_db("202518-3", "Homo_sapiens", force=TRUE)

## Select germline db to use with igblastn():
use_germline_db(db_name)

## Select C-region db to use with igblastn():
use_c_region_db("_IMGT.human.IGH+IGK+IGL.202412")
```

```
## ---------------------------------------------------------------------
## Call igblastn()
## ---------------------------------------------------------------------

## We don't specify the 'outfmt' argument so output will be in AIRR
## format:
AIRR_df <- igblastn(query)
AIRR_df

## The result is a tibble with one row per query sequence:
class(AIRR_df)
dim(AIRR_df)

## You can call igbrowser() on 'AIRR_df' to visualize the annotated
## sequences in a browser. See '?igbrowser'.

## Note that this tibble can easily be converted to an ordinary data.frame
## with 'as.data.frame()', or to a DataFrame with 'as(., "DataFrame")':
as(AIRR_df, "DataFrame")

## To call igblastn() on a subset of the FASTA file, load the file as a
## DNAStringSet object with Biostrings::readDNAStringSet(), then subset
## the object, and finally pass the result of the subsetting operation
## to igblastn():
query_21_30 <- readDNAStringSet(query)[21:30]
query_21_30  # a DNAStringSet object with 10 sequences
igblastn(query_21_30)

## ---------------------------------------------------------------------
## Parallel computing
## ---------------------------------------------------------------------

## The igblastn standalone executable included in IgBLAST supports
## multithreading via command line argument -num_threads. To use it
## in igblastn(), set argument 'num_threads' to the desired value:
AIRR_df2 <- igblastn(query, num_threads=4)
stopifnot(identical(AIRR_df, AIRR_df2))

## Unfortunately, in our experience, using 'num_threads' on Linux
## doesn't achieve any significant speedup.

## Alternatively, one can parallelize execution at the R level using
## standard tools from the parallel or BiocParallel package. In this
## case it's the responsibility of the user to split the input in
## smaller batches and combine the results obtained for each batch:
if (.Platform$OS.type != "windows") {
  library(parallel)
  ## Split 'query' in 10 batches of 25 sequences each:
  batches <- split(readDNAStringSet(query), rep(1:10, each=25))
  AIRR_df3 <- mclapply(seq_along(batches),
                       function(i) igblastn(batches[[i]]),
                       mc.cores=4)
  ## Combine the results:
```

```
  AIRR_df3 <- do.call(rbind, AIRR_df3)
  stopifnot(identical(AIRR_df, AIRR_df3))
}

## Note that the actual speedup will depend on many factors like
## hardware, number and size of the batches, number of cores used,
## etc...

## ---------------------------------------------------------------------
## TCR analysis
## ---------------------------------------------------------------------

## NCBI IgBLAST can also be used for TCR sequence analysis, and so does
## igblastn().

## File 'SRR11341217.fasta.gz' included in igblastr contains 10,875 human
## beta chain TCR transcripts running from 5' of reverse transcription
## reaction to beginning of constant region:
filename <- "SRR11341217.fasta.gz"
query <- system.file(package="igblastr", "extdata", "TCR", filename)

## For this example, we're only keeping the first 100 sequences:
query <- head(readDNAStringSet(query), n=100)

## Install Human TCR germline db from IMGT:
db_name <- install_IMGT_germline_db("202518-3", "Homo_sapiens",
                                    tcr.db=TRUE, force=TRUE)

## Select germline db to use with igblastn():
use_germline_db(db_name)

## Select C-region db to use with igblastn():
use_c_region_db("_IMGT.human.TRA+TRB+TRG+TRD.202509")

## Call igblastn(). Note that the 'ig_seqtype' argument will be
## automatically set to "TCR" (see documentation of the 'ig_seqtype'
## argument above in this man page for more information):
AIRR_df <- igblastn(query)

## ---------------------------------------------------------------------
## More examples
## ---------------------------------------------------------------------

## See '?read_igblastn_fmt7_output' for more examples.
```

igblastr_usage_report *Turn "Usage Reporting" on or off*

Description

By default, the NCBI BLAST+ and IgBLAST programs will "call home" to report usage when they run on a computer connected to the internet. See https://www.ncbi.nlm.nih.gov/books/NBK569851/ for the details. This can induce a significant slowdown in some situations e.g. when the igblastn *standalone executable* is called in a loop on a small set of query sequences at each iteration.

For this reason, the "call home" feature is disabled in **igblastr** by default, unless environment variable BLAST_USAGE_REPORT is set to true.

More precisely, the "call home" feature is controlled by global option igblastr_usage_report in **igblastr**. On package startup, this option is set to TRUE if environment variable BLAST_USAGE_REPORT is set to true. Otherwise (i.e. if BLAST_USAGE_REPORT is not set, or is set to false or gibberish) it is set to FALSE.

Details

The user can change the value of global option igblastr_usage_report any time with:

```
options(igblastr_usage_report=TRUE)
```

or with:

```
options(igblastr_usage_report=FALSE)
```

To get the value of this option, use:

```
getOption("igblastr_usage_report")
```

Note that changing the value of a global option interactively with options(...) won't be remembered across R sessions. For a persistent change, you can either:

- Put the options(...) command in your .Rprofile file. See ?Rprofile for more information. Note that this is the standard way of setting global options persistently.
- In the particular case of global option igblastr_usage_report an alternative is to define environment variable BLAST_USAGE_REPORT outside R. The exact way to do this is OS-dependent e.g. on Linux and Mac you can define it in your user's .profile by adding the following line to it:

  ```
  export BLAST_USAGE_REPORT=true
  ```

See Also

- The igblastn function to run the igblastn *standalone executable* included in IgBLAST from R. This is the main function in the **igblastr** package.
- IgBLAST is described at https://pubmed.ncbi.nlm.nih.gov/23671333/.

Examples

```
## Check current status of usage reporting:
getOption("igblastr_usage_report")

## Turn on usage reporting:
options(igblastr_usage_report=TRUE)

## Turn off usage reporting:
options(igblastr_usage_report=FALSE)
```

`igblast_info` *Check IgBLAST used by igblastr*

Description

Collect basic information about the IgBLAST installation used by the **igblastr** package, or about any IgBLAST installation on the user machine.

Usage

```
igblast_info(igblast_root=get_igblast_root())

igblast_build(igblast_root=get_igblast_root())
igblastn_version(igblast_root=get_igblast_root(), raw.version=FALSE)
makeblastdb_version(igblast_root=get_igblast_root(), raw.version=FALSE)
list_igblast_organisms(igblast_root=get_igblast_root())

has_igblast()
```

Arguments

`igblast_root` A single string that is the path to an IgBLAST installation. By default `igblast_root` is set to `get_igblast_root()`, which is the path to the IgBLAST installation used by the **igblastr** package. See ?`get_igblast_root` for more information.

Note that the supplied string must contain the path to the *root directory* of an IgBLAST installation, that is, to a directory with a `bin` subdirectory in it that has the `igblastn`, `igblastp`, and `makeblastdb` *standalone executables* (on Windows these executables are files named `igblastn.exe`, `igblastp.exe`, and `makeblastdb.exe`, respectively).

`raw.version` By default (i.e. when `raw.version` is omitted or set to `FALSE`), `igblastn_version()` and `makeblastdb_version()` return the version string of the `igblastn` and `makeblastdb` *standalone executables* included in IgBLAST. This string is extracted from the output produced by system commands:

```
igblastn -version
```

and

```
makeblastdb -version
```

When `raw.version` is set to `TRUE`, `igblastn_version()` and `makeblastdb_version()` return the *full ouput* produced by the above commands.

Value

`igblast_info()` returns a named list containing basic information about the IgBLAST installation.

`igblast_build()` returns a single string containing IgBLAST build information.

By default, `igblastn_version()` returns a single string containing the version of the `igblastn` *standalone executable* included in IgBLAST.

By default, `makeblastdb_version()` returns a single string containing the version of the `makeblastdb` *standalone executable* included in IgBLAST.

`list_igblast_organisms()` returns a character vector that lists the organisms for which IgBLAST provides *internal data*. Note that this is obtained by simply listing the content of the `internal_data` directory located in the IgBLAST installation.

`has_igblast()` returns `TRUE` or `FALSE`.

See Also

- The `igblastn` function to run the `igblastn` *standalone executable* included in IgBLAST from R. This is the main function in the **igblastr** package.
- `install_igblast` to perform an *internal* IgBLAST installation.
- `get_igblast_root` to get (or set) the IgBLAST installation used (or to be used) by the **igblastr** package.
- IGBLAST_ROOT to set the *external* IgBLAST installation to be used by the **igblastr** package in a persistent manner.
- intdata_utils to access and manipulate IgBLAST *internal data*.
- IgBLAST is described at `https://pubmed.ncbi.nlm.nih.gov/23671333/`.

Examples

```
if (!has_igblast()) install_igblast()

igblast_info()

list_igblast_organisms()
```

IGBLAST_ROOT *Use an external IgBLAST installation*

Description

Select the *external* IgBLAST installation to use in **igblastr** in a persistent manner.

Details

The **igblastr** package can use 2 types of IgBLAST installation:

1. Internal (a.k.a. igblastr-managed): refers to an installation obtained with `install_igblast()`.
2. External: refers to an installation that is not managed by the **igblastr** package. This is usually an installation that was manually performed by you or a system administrator on your machine. It can be a system-wide installation or a per-user installation.

To use an *external* installation of IgBLAST in **igblastr**, set environment variable IGBLAST_ROOT to the path of the installation. Note that this must be the path to the *root directory* of the IgBLAST installation, that is, to a directory with a `bin` subdirectory in it that has the `igblastn`, `igblastp`, and `makeblastdb` *standalone executables* (on Windows these executables are files named `igblastn.exe`, `igblastp.exe`, and `makeblastdb.exe`, respectively).

This can be done within your current R session with `Sys.setenv(IGBLAST_ROOT="path/to/igblast_root")` for testing. However, this won't be remembered across R sessions.

To set IGBLAST_ROOT in a persistent manner, define it outside R. The exact way to do this is OS-dependent e.g. on Linux and Mac you can define it in your user's `.profile` by adding the following line to it:

```
export IGBLAST_ROOT="path/to/igblast_root"
```

See Also

- The `igblastn` function to run the `igblastn` *standalone executable* included in IgBLAST from R. This is the main function in the **igblastr** package.
- `install_igblast` to perform an *internal* IgBLAST installation.
- `igblast_info` to collect basic information about the IgBLAST installation used by the **igblastr** package.
- IgBLAST is described at `https://pubmed.ncbi.nlm.nih.gov/23671333/`.

igbrowser *Display annotated BCR sequences in a browser*

Description

Use igbrowser() to display the annotated BCR sequences returned by igblastn() in a browser. For each sequence, the V, D, and J segments are shown as well as the FWR1-4 and CDR1-3 regions. Additionally, the C segments are shown if the C-region information is available.

Usage

```
igbrowser(AIRR_df, show.full.sequence=FALSE, dna.coloring=TRUE,
          Vcolor="#FFDDD2", Dcolor="#CFC", Jcolor="#CEF", Ccolor="#EEC",
          FWRcolor="#C9D", CDRcolor="#EE4")
```

Arguments

AIRR_df
: The AIRR-formatted data.frame or tibble returned by igblastn(). Note that calling igbrowser() on a data.frame with thousands of rows is quite resource-intensive (it can even crash your browser!), so in this case we recommend subsetting the data.frame before passing it to igbrowser() to keep the number of rows under 2000.

show.full.sequence
: By default, the part of the BCR sequences upstream of the V region is not shown. Set show.full.sequence to TRUE to show it.

dna.coloring
: Whether the nucleotides in the BCR sequences (sequence column in AIRR_df) should be colored or not.

Vcolor, Dcolor, Jcolor, Ccolor
: The background colors of the V, D, J, and C segments of the BCR sequences. Note that the C segments are shown only if AIRR_df contains C-region information.

FWRcolor, CDRcolor
: The background colors of the Framework Regions (FWR1-4) and Complementarity-Determining Regions (CDR1-3), respectively.

Value

0 or the error code returned by the internal call to browseURL(), invisibly.

See Also

- The igblastn function to run the igblastn *standalone executable* included in IgBLAST from R. This is the main function in the **igblastr** package.
- IgBLAST is described at https://pubmed.ncbi.nlm.nih.gov/23671333/.
- tibble objects implemented in the **tibble** package.

Examples

```
if (!has_igblast()) install_igblast()

query <- system.file(package="igblastr", "extdata",
                     "BCR", "heavy_sequences.fasta")
use_germline_db("_AIRR.human.IGH+IGK+IGL.202410")

## ---------------------------------------------------------------------
## With C regions
## ---------------------------------------------------------------------

use_c_region_db("_IMGT.human.IGH+IGK+IGL.202412")
AIRR_df <- igblastn(query)
igbrowser(AIRR_df)

## By default, the part of the sequences upstream of the V region is
## not shown. Use 'show.full.sequence=TRUE' to show the full sequences:
igbrowser(AIRR_df, show.full.sequence=TRUE)

## ---------------------------------------------------------------------
## No C regions
## ---------------------------------------------------------------------

use_c_region_db("")
AIRR_df2 <- igblastn(query)
igbrowser(AIRR_df2)
```

`install_igblast` *Install IgBLAST*

Description

Download and install a pre-compiled IgBLAST from NCBI FTP site for use with **igblastr**.

Usage

```
install_igblast(release="LATEST", force=FALSE, ...)
```

Arguments

`release`	A single string specifying the IgBLAST release version to install. For example `"LATEST"` (recommended), or one of the IgBLAST release versions listed at `https://ftp.ncbi.nih.gov/blast/executables/igblast/release/` (e.g. `"1.21.0"`). Note that old versions have not been tested and are not guaranteed to be compatible with the **igblastr** package.
`force`	Set to `TRUE` to reinstall if the specified IgBLAST release version is already installed.
`...`	Extra arguments to be passed to the internal call to `download.file()`. See `?download.file` in the **utils** package for more information.

Value

The path to the *root directory* of the IgBLAST installation, as an invisible string.

See Also

- The `igblastn` function to run the `igblastn` *standalone executable* included in IgBLAST from R. This is the main function in the **igblastr** package.
- IGBLAST_ROOT to use an *external* IgBLAST installation.
- `igblast_info` to collect basic information about the IgBLAST installation used by the **igblastr** package.
- IgBLAST is described at `https://pubmed.ncbi.nlm.nih.gov/23671333/`.

Examples

```
if (!has_igblast()) install_igblast()

igblast_info()
```

`install_IMGT_germline_db` *Install a germline db from IMGT*

Description

The `install_IMGT_germline_db()` function downloads V/D/J germline gene allele sequences from the IMGT website for a given organism, and stores them in a local germline database. This local database gets installed in **igblastr**'s persistent cache. It can then be used later with `igblastn()`.

CONDITIONS OF USE AND LICENSE: The IMGT data is provided to the academic users and NPO's (Not for Profit Organization(s)) under the CC BY-NC-ND 4.0 license. See `https://creativecommons.org/licenses/by-nc-nd/4.0/`. Any other use of IMGT material, from the private sector, needs a financial arrangement with CNRS.

Usage

```
install_IMGT_germline_db(release, organism="Homo sapiens", tcr.db=FALSE,
                         loci="auto", force=FALSE, ...)

## Related utilities:
list_IMGT_releases(recache=FALSE)
list_IMGT_organisms(release)
IMGT_is_up()

## Can only be used to "subset" a built-in C-region databse (see
## advanced example in Examples section below):
install_IMGT_c_region_db(organism, loci, force=FALSE)
```

Arguments

`release` A single string specifying the IMGT/V-QUEST release to get the germline gene allele sequences from (or to list the organisms from for `list_IMGT_organisms()`). Use `list_IMGT_releases()` to list all releases.

`organism` A single string specifying the latin name of the organism for which to get the germline gene allele sequences.

Note that `install_IMGT_c_region_db()` also accepts the common name of the organism (e.g. "human").

`tcr.db` Should the database to install be populated with allele sequences from the BCR (B-cell Receptor) or TCR (T-cell Receptor) germline loci?

The BCR germline loci are: IGH, IGK, IGL.

The TCR germline loci are: TRA, TRB, TRG, TRD.

By default, the V/D/J allele sequences from the BCR germline loci are downloaded. Set `tcr.db` to `TRUE` to download the V/D/J allele sequences from the TCR germline loci instead.

`loci` By default, the database to install will be populated with the allele sequences from all the BCR or TCR loci. However, if you want to restrict the database to specific loci, you can use the `loci` argument to specify these loci. The subset of loci can be specified as a character vector with one element per locus (e.g. `"IGH"` or `c("TRA", "TRB")`), or as a `+`-separated list in a single string (e.g. `"TRA+TRB"`).

`force` Set to `TRUE` to reinstall if the requested database is already installed.

`...` Extra arguments to be passed to the internal call to `download.file()`. See ?`download.file` in the **utils** package for more information.

`recache` `list_IMGT_releases()` uses a caching mechanism so that the list of IMGT/V-QUEST releases gets downloaded only once from the IMGT website during an R session (note that this caching is done in memory so it does not persist across sessions). Set `recache` to `TRUE` to force a new download (and recaching) of the list of IMGT/V-QUEST releases.

Details

The following naming scheme is used to form the name of the installed database:

```
IMGT-<release>.<organism>.<loci>
```

where:

1. <release> is the IMGT/V-QUEST release e.g. 202518-3 or 202449-1. Use `list_IMGT_releases()` to get the list of releases currently available at IMGT/V-QUEST.
2. <organism> is the latin name (a.k.a. binomial name) of the organism with all spaces replaced with underscores (_). For example Homo_sapiens or Macaca_mulatta. Use `list_IMGT_organisms("<release>")` to get the list of organisms included in a given IMGT/V-QUEST release. Note that, starting with release 202405-2, IMGT/V-QUEST provides BCR and TCR germline gene allele sequences for mouse strain C57BL6J (Mus_musculus_C57BL6J).

3. <loci> is a string obtained by concatenating the germline loci together separated with the + sign. For example IGH+IGK+IGL or TRA+TRB+TRG+TRD. The list of loci depends on whether the germline gene allele sequences for BCR or TCR were requested. See `tcr.db` argument above for more information. Note that for some IMGT/V-QUEST releases/organisms, only a subset of the loci are available. For example, in release 202343-3, the only TCR germline loci available for Mus_musculus_C57BL6J are TRA and TRB. This will be automatically reflected in the name of the installed germline db.

Value

`install_IMGT_germline_db()` returns the name to the newly installed germline db as an invisible string.

`list_IMGT_releases()` returns the list of IMGT/V-QUEST releases in a character vector. The releases are sorted from newest to oldest (latest release is first).

`list_IMGT_organisms()` returns the list of organisms included in the specified IMGT/V-QUEST release in a character vector.

`IMGT_is_up()` returns `TRUE` or `FALSE`, indicating whether the IMGT website at https://www.imgt.org is up and running or down.

`install_IMGT_c_region_db()` returns the name to the newly installed C-region db as an invisible string.

Note

`install_IMGT_germline_db()` generates the local database by performing the instructions provided at https://ncbi.github.io/igblast/cook/How-to-set-up.html.

See Also

- The `igblastn` function to run the `igblastn` *standalone executable* included in IgBLAST from R. This is the main function in the **igblastr** package.
- `use_germline_db` to select the cached germline db to use with `igblastn()`.
- The IMGT website: https://www.imgt.org/.
- The IMGT/V-QUEST download site: https://www.imgt.org/download/V-QUEST/.
- IgBLAST is described at https://pubmed.ncbi.nlm.nih.gov/23671333/.

Examples

```
if (!has_igblast()) install_igblast()

if (IMGT_is_up()) {
  ## ---------------------------------------------------------------------
  ## BASIC EXAMPLES
  ## ---------------------------------------------------------------------

  ## As of March 26, 2025, the latest IMGT/V-QUEST release is 202518-3:
  list_IMGT_releases()

  list_IMGT_organisms("202518-3")
  ## Download Mouse BCR germline gene allele sequences from IMGT/V-QUEST
  ## 202518-3, and store them in a cached germline database:
  install_IMGT_germline_db(”202518-3”, organism=”Mus musculus”, force=TRUE)

  ## List the cached germline databases:
  list_germline_dbs()

  ## Select newly installed germline db to use with igblastn():
  use_germline_db(”IMGT-202518-3.Mus_musculus.IGH+IGK+IGL”)

  ## Download Mouse TCR germline gene allele sequences from IMGT/V-QUEST
  ## 202518-3, and store them in a cached germline database:
  install_IMGT_germline_db(”202518-3”, organism=”Mus musculus”,
                           tcr.db=TRUE, force=TRUE)
  list_germline_dbs()

  ## ---------------------------------------------------------------------
  ## ADVANCED EXAMPLES
  ## ---------------------------------------------------------------------

  ## Install an IMGT database for a subset of TCR loci:
  install_IMGT_germline_db(”202518-3”, organism=”Homo sapiens”,
                           loci=”TRA+TRB”, force=TRUE)
  list_germline_dbs()

  ## Install the corresponding C-region database:
  install_IMGT_c_region_db(”human”, ”TRA+TRB”, force=TRUE)
  list_c_region_dbs()

  ## Note that install_IMGT_c_region_db() can only be used to "subset"
  ## a built-in C-region databse.
}
```

intdata-utils *Access and manipulate IgBLAST internal data*

Description

IgBLAST *internal data* is expected to annotate all the known germline V gene alleles for a given organism. It is provided by NCBI and is typically included in a standard IgBLAST installation.

The **igblastr** package provides a small set of utilities to access and manipulate IgBLAST *internal data*.

Usage

```
## Access internal data:
get_intdata_path(organism, for.aa=FALSE, domain_system=c("imgt", "kabat"),
                 which=c(”live”, ”original”))
```

```
load_intdata(organism, for.aa=FALSE, domain_system=c("imgt", "kabat"),
             which=c("live", "original"))

## Manipulate internal data:
V_genes_with_varying_fwrcdr_boundaries(intdata, V_segment=NULL)
translate_V_alleles(V_alleles, intdata, V_segment=NULL)
V_allele_has_stop_codon(V_alleles, intdata)
```

Arguments

`organism` A single string containing the name of an organism as returned by `list_igblast_organisms()`.

Alternatively, this can be the name of a cached germline db. Use `list_germline_dbs()` to list the cached germline dbs. Note that:

- This works only for germline dbs that include their own *internal data*.
- At the moment, only the built-in AIRR germline dbs for human and rhesus monkey include their own *internal data* (this data is provided by AIRR-community/OGRDB).

`for.aa` By default, the data.frame returned by `load_intdata()` contains FWR/CDR boundaries reported with respect to the nucleotide sequences of the germline V alleles. Setting `for.aa` to `TRUE` will return a data.frame where they are reported with respect to the amino acid sequences of the germline V alleles.

`domain_system` Domain system to be used for segment annotation. Must be `"imgt"` (the default) or `"kabat"`.

`which` By default, `get_intdata_path()` and `load_intdata()` access the "live IgBLAST data", that is, the IgBLAST data that the user has possibly updated with `update_live_igdata()`. Depending on whether updates were applied or not, the "live IgBLAST data" might differ from the original IgBLAST data.

Set `which` to `"original"` if you want to access the original IgBLAST data instead.

See ?`update_live_igdata` for more information about "live" and "original" IgBLAST data.

`intdata` A data.frame as returned by `load_intdata()` for `V_genes_with_varying_fwrcdr_boundaries()`, or by `load_intdata(..., for.aa=FALSE)` for `translate_V_alleles()` and `V_allele_has_stop_codon()`.

`V_segment` The name of a V gene segment. This can be set to `"fwr1"`, `"cdr1"`, `"fwr2"`, `"cdr2"`, or `"fwr3"`.

By default (i.e. when `V_segment` is omitted or set to `NULL`), `V_genes_with_varying_fwrcdr_boundarie` will identify V genes for which any segment has varying boundaries across alleles. Otherwise, it will identify V genes for which the specified segment has varying boundaries.

By default `translate_V_alleles()` will translate the entire coding frame in each allele. Otherwise, it will translate the specified segment only.

`V_alleles` A DNAStringSet object containing germline V gene allele sequences.

Details

IgBLAST *internal data* is typically included in a standard IgBLAST installation. It's located in the `internal_data/` directory which is itself a subdirectory of IgBLAST *root directory*.

Value

`get_intdata_path()` returns a single string containing the path to the *internal data* included in the IgBLAST installation used by **igblastr**, for the specified organism.

`load_intdata()` returns the *internal data* in a data.frame with 1 row per germline V allele sequence and the following columns:

- `allele_name`: allele name;
- `fwr1_start`, `fwr1_end`: FWR1 start/end positions (1-based);
- `cdr1_start`, `cdr1_end`: CDR1 start/end positions (1-based);
- `fwr2_start`, `fwr2_end`: FWR2 start/end positions (1-based);
- `cdr2_start`, `cdr2_end`: CDR2 start/end positions (1-based);
- `fwr3_start`, `fwr3_end`: FWR3 start/end positions (1-based);
- `chain_type`: chain type;
- `coding_frame_start`: first coding frame start position (0-based).

`V_genes_with_varying_fwrcdr_boundaries()` returns a character vector containing the names of the germline V genes for which the FWR/CDR boundaries are not the same across all alleles.

`translate_V_alleles()` returns a named character vector with 1 amino acid sequence per supplied allele. The vector contains an `NA` for any allele that is not annotated in `intdata` or for which the required information is `NA`. The names on it are the names of the supplied alleles.

`V_allele_has_stop_codon()` returns a named logical vector with 1 value per supplied allele. The vector contains an `NA` for any allele that is not annotated in `intdata` or for which `intdata$coding_frame_start` has an `NA`. The names on it are the names of the supplied alleles.

See Also

- `compute_intdata` to annotate a set of germline V gene allele sequences.
- auxdata_utils to access, manipulate, and generate IgBLAST *auxiliary data*.
- `update_live_igdata` for more information about "live" and "original" IgBLAST data.
- `list_igblast_organisms` to list the organisms supported by IgBLAST.
- `list_germline_dbs` to list the cached germline dbs.
- DNAStringSet objects in the **Biostrings** package.
- The `translate_codons` function on which `translate_V_alleles()` is based.
- `allele2gene` to go from germline gene allele names to germline gene names.
- The `igblastn` function to run the `igblastn` *standalone executable* included in IgBLAST from R. This is the main function in the **igblastr** package.
- IgBLAST is described at `https://pubmed.ncbi.nlm.nih.gov/23671333/`.

Examples

```
if (!has_igblast()) install_igblast()

igblast_info()

## ---------------------------------------------------------------------
## list_igblast_organisms() and get_intdata_path()
## ---------------------------------------------------------------------

list_igblast_organisms()

get_intdata_path("rabbit")
rabbit_intdata <- load_intdata("rabbit")
head(rabbit_intdata)

rabbit_intdata2 <- load_intdata("rabbit", for.aa=TRUE)
head(rabbit_intdata2)

## The values in the "end" cols in 'rabbit_intdata' are exactly 3 times
## those in the "end" cols in 'rabbit_intdata2':
end_colnames <- grep("_end$", colnames(rabbit_intdata), value=TRUE)
stopifnot(identical(rabbit_intdata [ , end_colnames],
                    rabbit_intdata2[ , end_colnames] * 3L))

## Get the internal data included in the _AIRR.human.IGH+IGK+IGL.202410
## germline db (this data is provided by AIRR-community/OGRDB):
db_name <- "_AIRR.human.IGH+IGK+IGL.202410"
human_intdata <- load_intdata(db_name)
head(human_intdata)

## ---------------------------------------------------------------------
## V_genes_with_varying_fwrcdr_boundaries()
## ---------------------------------------------------------------------

## Note that the alleles of a given germline V gene don't necessarily
## share the same FWR/CDR boundaries. You can use utility function
## V_genes_with_varying_fwrcdr_boundaries() to identify them:
human_intdata0 <- load_intdata("human")
var_genes <- V_genes_with_varying_fwrcdr_boundaries(human_intdata0)
var_genes
subset(human_intdata0, allele2gene(allele_name) == "IGHV4-31")

## Human germline V genes for which the CDR1 boundaries are not the same
## across all alleles:
var_genes <- V_genes_with_varying_fwrcdr_boundaries(human_intdata0,
                                                     V_segment="cdr1")
var_genes
subset(human_intdata0, allele2gene(allele_name) %in% var_genes)

## ---------------------------------------------------------------------
## translate_V_alleles() and V_allele_has_stop_codon()
## ---------------------------------------------------------------------
```

```
V_alleles <- load_germline_db(db_name, region_types="V")
V_alleles  # DNAStringSet object

V_aa <- translate_V_alleles(V_alleles, human_intdata)
head(V_aa)

fwr2 <- translate_V_alleles(V_alleles, human_intdata, V_segment="fwr2")
head(fwr2)

## Surprisingly, 13 V alleles in _AIRR.human.IGH+IGK+IGL.202410 contain
## the stop codon:
has_stop_codon <- grepl("*", V_aa, fixed=TRUE)
table(has_stop_codon)
V_aa[has_stop_codon]
V_alleles[has_stop_codon]
```

list_c_region_dbs *List cached C-region dbs and select one to use with igblastn()*

Description

A small set of utilities for basic manipulation of *cached C-region dbs*:

- list_c_region_dbs(): List all the *cached C-region dbs*, that is, all the C-region databases currently installed in **igblastr**'s persistent cache.
- use_c_region_db(): Select the cached C-region db to use with igblastn(). This choice will be remembered for the duration of the current R session but can be changed anytime.
- load_c_region_db(): Load the nucleotide sequences of the gene regions stored in a cached C-region db.
- rm_c_region_db(): Remove a C-region db from **igblastr**'s persistent cache.

Usage

```
list_c_region_dbs(builtin.only=FALSE, names.only=FALSE, long.listing=FALSE)

use_c_region_db(db_name=NULL, verbose=FALSE)

load_c_region_db(db_name)

rm_c_region_db(db_name)
```

Arguments

builtin.only By default list_c_region_dbs() returns the list of all cached C-region dbs, including built-in C-region dbs. Set builtin.only to TRUE to return only the list of built-in C-region dbs. Note that built-in dbs are prefixed with an underscore (_).

`names.only` By default `list_c_region_dbs()` returns the list of cached C-region dbs in a data.frame with one db per row. Set `names.only` to `TRUE` to return only the db names in a character vector.

`long.listing` `TRUE` or `FALSE`. If set to `TRUE`, then `list_c_region_dbs()` returns a named list with one list element per C-region db. Each list element is a named integer vector that indicates the number of C-region sequences per locus.

Ignored if `names.only` is set to `TRUE`.

`db_name` For `use_c_region_db()`:

`NULL` or a single string specifying the name of the cached C-region db to use. Use `list_c_region_dbs()` to list all the cached C-region dbs.

If set to `NULL` (the default), then `use_c_region_db()` returns the name of the cached C-region db that is currently in use, if any. Otherwise it returns the empty string (`""`).

Note that the current selection can be cancelled with `use_c_region_db("")`.

For `load_c_region_db()`:

A single string specifying the name of the cached C-region db from which to load the gene regions. Use `list_c_region_dbs()` to list all the cached C-region dbs.

For `rm_c_region_db()`:

A single string specifying the name of the C-region db to remove from the cache. This cannot be a built-in db.

`verbose` If set to `TRUE`, then `use_c_region_db()` will display some information about its internal operations.

Details

The **igblastr** package provides utility functions to perform basic manipulation of the cached germline databases and cached C-region databases to use with `igblastn()`.

Terminology:

- A *cached germline db* contains the nucleotide sequences of the V, D, and J gene regions for a given organism.
- A *cached C-region db* contains the nucleotide sequences of the C regions (i.e. constant gene regions) for a given organism.

This man page documents the basic utilities to operate on the cached C-region dbs: `list_c_region_dbs()`, `use_c_region_db()`, and `load_c_region_db()`, and `rm_c_region_db()`.

The basic utilities to operate on the cached germline dbs are documented in the man page for `list_germline_dbs`.

Value

`list_c_region_dbs()` returns the list of all cached C-region dbs in a data.frame with one db per row (if `names.only` is `FALSE`, which is the default), or in a character vector (if `names.only` is `TRUE`). Column C in the data.frame indicates the number of C-region sequences in each db.

Built-in dbs are prefixed with an underscore (_). Note that the built-in C-region dbs from IMGT were downloaded from https://www.imgt.org/vquest/refseqh.html#constant-sets and included in the **igblastr** package on the date indicated by the suffix of the db name.

When called with no argument, `use_c_region_db()` returns a single string containing the name of the cached C-region db currently used by `igblastn()` if any, or the empty string (`""`) if `igblastn()` is not using any C-region db.

When called with the `db_name` argument, `use_c_region_db(db_name)` returns `db_name` invisibly.

`load_c_region_db()` returns the nucleotide sequences from the specified C-region db in a named DNAStringSet object.

`rm_c_region_db()` does not return anything (invisible `NULL`).

See Also

- The `igblastn` function to run the `igblastn` *standalone executable* included in IgBLAST from R. This is the main function in the **igblastr** package.
- `use_germline_db` to select the cached germline db to use with `igblastn()`.
- DNAStringSet objects in the **Biostrings** package.
- IgBLAST is described at https://pubmed.ncbi.nlm.nih.gov/23671333/.

Examples

```
if (!has_igblast()) install_igblast()

## 7 built-in C-region dbs (prefixed with an underscore):
list_c_region_dbs()
list_c_region_dbs(names.only=TRUE)    # db names only
list_c_region_dbs(long.listing=TRUE)  # long listing

## Select C-region db to use with igblastn():
use_c_region_db("_IMGT.human.IGH+IGK+IGL.202412")
use_c_region_db()    # get current selection
use_c_region_db("")  # cancel current selection
use_c_region_db()

## Load C-region sequences:
load_c_region_db("_IMGT.human.IGH+IGK+IGL.202412")
load_c_region_db("_IMGT.mouse.IGH.202509")
```

`list_germline_dbs` *List cached germline dbs and select one to use with igblastn()*

Description

A small set of utilities for basic manipulation of *cached germline dbs*:

- `list_germline_dbs()`: List all the *cached germline dbs*, that is, all the germline databases currently installed in **igblastr**'s persistent cache.

- use_germline_db(): Select the cached germline db to use with igblastn(). This choice will be remembered for the duration of the current R session but can be changed anytime.
- load_germline_db(): Load the nucleotide sequences of the gene regions stored in a cached germline db.
- rm_germline_db(): Remove a germline db from **igblastr**'s persistent cache.

Usage

```
list_germline_dbs(builtin.only=FALSE, with.intdata.only=FALSE,
                  names.only=FALSE, long.listing=FALSE)

use_germline_db(db_name=NULL, verbose=FALSE)

load_germline_db(db_name, region_types=NULL)

rm_germline_db(db_name)
```

Arguments

builtin.only
By default list_germline_dbs() returns the list of all cached germline dbs, including built-in germline dbs. Set builtin.only to TRUE to return only the list of built-in germline dbs. Note that built-in dbs are prefixed with an underscore (_).

with.intdata.only
Set with.intdata.only to TRUE to return only the list of germline dbs that include their own *internal data*. Note that whether a germline db includes its own *internal data* or not is indicated by the intdata column. See ?intdata_utils for more information about IgBLAST *internal data*.

names.only
By default list_germline_dbs() returns the list of cached germline dbs in a data.frame with one db per row. Set names.only to TRUE to return only the db names in a character vector.

long.listing
TRUE or FALSE. If set to TRUE, then list_germline_dbs() returns a named list with one list element per germline db. Each list element is an integer matrix that indicates the number of germline gene allele sequences per locus and region type.

Ignored if names.only is set to TRUE.

db_name
For use_germline_db():

NULL or a single string specifying the name of the cached germline db to use. Use list_germline_dbs() to list all the cached germline dbs.

If set to NULL (the default), then use_germline_db() returns the name of the cached germline db that is currently in use, if any. Otherwise it raises an error.

For load_germline_db():

A single string specifying the name of the cached germline db from which to load the V, D, and/or J regions. Use list_germline_dbs() to list all the cached germline dbs.

For rm_germline_db():

A single string specifying the name of the germline db to remove from the cache. This cannot be a built-in db.

`verbose` If set to `TRUE`, then `use_germline_db()` will display some information about its internal operations.

`region_types` The types of regions (V, D, and/or J) to load from the database. Specified as a single string (e.g. `"DJ"`) or as a character vector of single-letter elements (e.g. `c("D", "J")`). By default (i.e. when `region_types` is `NULL`), all the regions are returned.

Details

The **igblastr** package provides utility functions to perform basic manipulation of the cached germline databases and cached C-region databases to use with `igblastn()`.

Terminology:

- A *cached germline db* contains the nucleotide sequences of the V, D, and J gene regions for a given organism.
- A *cached C-region db* contains the nucleotide sequences of the C regions (i.e. constant gene regions) for a given organism.

This man page documents the basic utilities to operate on the cached germline dbs: `list_germline_dbs()`, `use_germline_db()`, `load_germline_db()`, and `rm_germline_db()`.

The basic utilities to operate on the cached C-region dbs are documented in the man page for `list_c_region_dbs`.

Value

`list_germline_dbs()` returns the list of all cached germline dbs in a data.frame with one db per row (if `names.only` is `FALSE`, which is the default), or in a character vector (if `names.only` is `TRUE`). Columns V, D, J in the data.frame indicate the number of germline gene allele sequences for each region in each db.

Built-in dbs are prefixed with an underscore (`_`). Note that the built-in germline dbs starting with `_AIRR` are made of the AIRR-community/OGRDB germline sets available at `https://ogrdb.airr-community.org/germline_sets/Homo%20sapiens` for human and at `https://ogrdb.airr-community.org/germline_sets/Mus%20musculus`. for mouse. Each AIRR db is populated with the latest germline datasets that were available at AIRR-community/OGRDB at the time indicated by the date (in YYYYMM format) embedded in the db name. The AIRR dbs with the `.src` suffix contain the *Source Sets*. The AIRR dbs without the `.src` suffix contain the *Reference Sets*. See `https://github.com/HyrienLab/igblastr/tree/devel/inst/extdata/germline_sets/AIRR/human/202410/README.md` for more information. The AIRR-community/OGRDB maintainers recommend to use the *Reference Sets* for AIRR-seq analysis. See `https://ogrdb.airr-community.org/germline_set/75`

When called with no argument, `use_germline_db()` returns a single string containing the name of the cached germline db currently used by `igblastn()` if any, or it raises an error if no germline db has been selected yet.

When called with the `db_name` argument, `use_germline_db(db_name)` returns `db_name` invisibly.

load_germline_db() returns the nucleotide sequences from the specified germline db in a named DNAStringSet object.

rm_germline_db() does not return anything (invisible NULL).

See Also

- The igblastn function to run the igblastn *standalone executable* included in IgBLAST from R. This is the main function in the **igblastr** package.
- install_IMGT_germline_db to install a germline db from IMGT.
- use_c_region_db to select the cached C-region db to use with igblastn().
- intdata_utils to access and manipulate IgBLAST *internal data*.
- DNAStringSet objects in the **Biostrings** package.
- IgBLAST is described at https://pubmed.ncbi.nlm.nih.gov/23671333/.

Examples

```
if (!has_igblast()) install_igblast()

## Get list of built-in germline dbs only.
list_germline_dbs(builtin.only=TRUE)
list_germline_dbs(builtin.only=TRUE, names.only=TRUE)  # db names only

## Get list of germline dbs that include their own internal data:
list_germline_dbs(with.intdata.only=TRUE)
list_germline_dbs(with.intdata.only=TRUE, names.only=TRUE)

## Long listing:
list_germline_dbs(long.listing=TRUE)
list_germline_dbs(with.intdata.only=TRUE, long.listing=TRUE)

if (IMGT_is_up()) {
  ## Install Mouse germline db from IMGT:
  install_IMGT_germline_db("202518-3", "Mus_musculus", force=TRUE)

  list_germline_dbs()  # all germline dbs

  ## Select germline db to use with igblastn():
  db_name <- "IMGT-202518-3.Mus_musculus.IGH+IGK+IGL"
  use_germline_db(db_name)  # select germline db to use

  use_germline_db()  # get current selection

  ## Load germline gene allele sequences:
  load_germline_db(db_name)
  load_germline_db(db_name, region_types="D")
  load_germline_db(db_name, region_types="DJ")
}
```

makeogrannote *Extract V gene FWR/CDR annotation from OGRDB JSON file*

Description

The `makeogrannote` function is a reimplementation in R of the Python script `makeogrannote.py` included in IgBLAST.

Usage

```
makeogrannote(germline_file)

## Related utilities:
check_V_ndm_data(V_ndm_data, allow.dup.entries=FALSE)
read_V_ndm_data(filepath)
write_V_ndm_data(V_ndm_data, file="", check.data=FALSE)
```

Arguments

`germline_file`	COMING SOON...
`V_ndm_data`	COMING SOON...
`allow.dup.entries`	COMING SOON...
`filepath`	COMING SOON...
`file`	COMING SOON...
`check.data`	COMING SOON...

Details

COMING SOON...

Value

COMING SOON...

See Also

- `compute_intdata` to annotate a set of germline V gene allele sequences.
- intdata_utils to access and manipulate IgBLAST *internal data*.
- The `igblastn` function to run the `igblastn` *standalone executable* included in IgBLAST from R. This is the main function in the **igblastr** package.
- IgBLAST is described at `https://pubmed.ncbi.nlm.nih.gov/23671333/`.

Examples

```
## COMING SOON...
```

OAS-utils *Download and manipulate OAS data*

Description

Some utility functions to query the Observed Antibody Space database, a.k.a. OAS, and to download and manipulate data from OAS.

OAS's homepage: https://opig.stats.ox.ac.uk/webapps/oas/

Note that OAS has two databases: the "Unpaired Sequences" database and the "Paired Sequences" database. Some of the utilities documented in this man page only work on data coming from the latter.

Usage

```
## Read metadata/data from a single OAS unit file:
read_OAS_csv_metadata(file)
read_OAS_csv(file, skip=1, ...)
extract_sequences_from_paired_OAS_df(df, add.prefix=FALSE)

## Basic query of OAS website:
list_paired_OAS_studies(as.df=FALSE, recache=FALSE)
list_paired_OAS_units(study, as.df=FALSE, recache=FALSE)
download_paired_OAS_units(study, units=NULL, destdir=".", ...)

## Read metadata/data from a batch of downloaded OAS unit files:
extract_metadata_from_OAS_units(dir=".", pattern="\\.csv\\.gz$")
extract_sequences_from_paired_OAS_units(dir=".", pattern="\\.csv\\.gz$")
```

Arguments

`file` A single string that is the path to an *OAS unit file*.

`skip` The number of lines of the data file to skip before beginning to read data. The first line in an OAS unit file contains metadata in JSON format, so must always be skipped.

`...` For `read_OAS_csv()`: Extra arguments to be passed to the internal call to `read.table()`. See ?`read.table` in the **utils** package for more information.

For `download_paired_OAS_units()`: Extra arguments to be passed to the internal call to `download.file()`. See ?`download.file` in the **utils** package for more information.

`df` The data.frame or tibble returned by `read_OAS_csv()`.

`add.prefix` `TRUE` or `FALSE`. Should the names on the returned DNAStringSet object be the original sequence ids as-is (this is the default), or should the `heavy_chain_` and `light_chain_` prefixes be added to them?

`extract_sequences_from_paired_OAS_df()` returns a DNAStringSet object with the sequence ids as names. The sequence ids are obtained from the `sequence_id_heavy` and sequence_id_light columns of the supplied data.frame or tibble. By default, they are propagated as-is to the DNAStringSet object, which makes it difficult to recognize which chain (heavy or light) the antibody sequences are coming from. Setting add.prefix to TRUE will add the heavy_chain_ or light_chain_ prefix to the names on the DNAStringSet object, hence making it easy to identify which chain a given antibody sequence is coming from.

as.df
: TRUE or FALSE. By default, i.e. when as.df is FALSE, list_paired_OAS_studies() and list_paired_OAS_units() return the list of studies or units in a character vector. Alternatively you can set as.df to TRUE to get the list in a 3-column data.frame that contains a directory index as displayed at https://opig.stats.ox.ac.uk/webapps/ngsdb/paired/ or at https://opig.stats.ox.ac.uk/webapps/ngsdb/paired/Jaffe_2022/csv/.

recache
: TRUE or FALSE. list_paired_OAS_studies() and list_paired_OAS_units() both cache the information retrieved from OAS website for the duration of the R session (note that this caching is done in memory so it does not persist across sessions). Set recache to TRUE to force a new retrieval (and recaching) of the results.

study
: A single string containing the name of a study as returned by list_paired_OAS_studies().

units
: NULL, or a character vector that must be a subset of list_paired_OAS_units(study) in which case the download will be restricted to these units only.

destdir
: A single string that is the path to the directory where the OAS unit files are to be downloaded.

dir
: A single string that is the path to a directory containing OAS unit files. This will typically be the same as destdir above if the unit files were downloaded with download_paired_OAS_units().

pattern
: Regular expression passed to the internal call to list.files() to obtain the list of OAS unit files located in dir. No reason to change this unless you know what you are doing.

Details

OAS delivers data in the form of *OAS unit files*. These files are typically obtained by running the bulk_download.sh script that OAS generates based on one's search criteria. They are compressed CSV (comma-separated values) files with the .csv.gz extension.

OAS unit files can vary a lot in size: from only a few KB to 25 MB or more.

The first line in an OAS unit file contains metadata in JSON format (which means that these files cannot strictly be considered CSV files).

The CSV data is MiAIRR-compliant (see The "MiAIRR format" paper in the References section below).

Value

read_OAS_csv_metadata() extracts the metadata from the specified OAS unit file and returns it in a named list.

read_OAS_csv() extracts the data from the specified OAS unit file and returns it in a tibble. The tibble has 1 row per antibody sequence if the data is unpaired (i.e. comes from the "Unpaired

Sequences" database), or 1 row per sequence pair if the data is paired (i.e. comes from the "Paired Sequences" database).

`extract_sequences_from_paired_OAS_df()` returns the sequence pairs in a named DNAStringSet object where the names are the sequence ids. See `add.prefix` above for how the sequence ids are obtained.

`list_paired_OAS_studies()` returns the list of studies that populate the "Paired Sequences" database in a character vector. This list can be seen here: `https://opig.stats.ox.ac.uk/webapps/ngsdb/paired/`.

`list_paired_OAS_units()` returns the list of all the OAS unit files that belong to a given study from the "Paired Sequences" database.

`download_paired_OAS_units()` does not return anything (invisible `NULL`).

`extract_metadata_from_OAS_units()` returns the metadata of all the OAS unit files found in the specified directory in a data.frame with 1 row per file.

`extract_sequences_from_paired_OAS_units()` extracts the sequence pairs from all the OAS unit files found in the specified directory and returns them in a named DNAStringSet object where the names are the sequence ids. The sequence ids are obtained by prefixing the original sequence ids found in the files with the name of the unit followed by `_heavy_chain_` or `_light_chain_`.

References

- The OAS paper:

  Tobias H. Olsen, Fergus Boyles, Charlotte M. Deane. Observed Antibody Space: A diverse database of cleaned, annotated, and translated unpaired and paired antibody sequences. Protein Science (2021). `https://doi.org/10.1002/pro.4205`
- The "MiAIRR format" paper:

  Rubelt, F., Busse, C., Bukhari, S. et al. Adaptive Immune Receptor Repertoire Community recommendations for sharing immune-repertoire sequencing data. Nat Immunol 18, 1274–1278 (2017). `https://doi.org/10.1038/ni.3873`

See Also

- OAS's homepage at: `https://opig.stats.ox.ac.uk/webapps/oas/`
- The `igblastn` function to run the `igblastn` *standalone executable* included in IgBLAST from R. This is the main function in the **igblastr** package.
- tibble objects implemented in the **tibble** package.
- DNAStringSet objects implemented in the **Biostrings** package.

Examples

```
list_paired_OAS_studies()

list_paired_OAS_units("Eccles_2020")

## Import all the pairs of antibody sequences from the Eccles_2020 study:

download_dir <- tempdir()
```

```
download_paired_OAS_units("Eccles_2020", destdir=download_dir)

metadata <- extract_metadata_from_OAS_units(download_dir)
metadata  # data.frame with 1 row per unit file

sequences <- extract_sequences_from_paired_OAS_units(download_dir)
sequences  # DNAStringSet object

## Odd indices correspond to heavy chain sequences and even indices
## to light chain sequences:

head(names(sequences))

sequences[1:2]  # 1st pair
sequences[3:4]  # 2nd pair
sequences[5:6]  # 3rd pair
# etc...
```

`read_igblastn_AIRR_output` *igblastn output format 19 (AIRR format)*

Description

Read `igblastn` output format 19 (AIRR format). This is the output produced by `igblastn` when `outfmt` is set to `"AIRR"` or 19.

This format is sometimes called "Rearrangement summary report" or simply "AIRR rearrangement tabular" format. See for example IgBLAST web interface at `https://www.ncbi.nlm.nih.gov/igblast/`.

Usage

```
read_igblastn_AIRR_output(out)
```

Arguments

`out`	The path to a file containing the output produced by `igblastn` when `outfmt` is set to `"AIRR"` or 19.

Value

A data.frame with 1 row per query sequence and many columns.

See Also

- The `igblastn` function to run the `igblastn` *standalone executable* included in IgBLAST from R. This is the main function in the **igblastr** package.
- `read_igblastn_fmt7_output` to read and parse `igblastn` output format 7.
- IgBLAST web interface at `https://www.ncbi.nlm.nih.gov/igblast/`.
- IgBLAST is described at `https://pubmed.ncbi.nlm.nih.gov/23671333/`.

Examples

```
if (!has_igblast()) install_igblast()

## ---------------------------------------------------------------------
## Access query sequences and select germline and C-region dbs to use
## ---------------------------------------------------------------------

## Files 'heavy_sequences.fasta' and 'light_sequences.fasta' included
## in igblastr contain 250 paired heavy- and light- chain sequences (125
## sequences in each file) downloaded from OAS (the Observed Antibody
## Space database):
filenames <- paste0(c("heavy", "light"), "_sequences.fasta")
query <- system.file(package="igblastr", "extdata", "BCR", filenames)

## Keep only the first 10 sequences from each file:
query <- c(head(readDNAStringSet(query[[1L]]), n=10),
           head(readDNAStringSet(query[[2L]]), n=10))

## Select the germline and C-region dbs to use with igblastn():
use_germline_db("_AIRR.human.IGH+IGK+IGL.202410")
use_c_region_db("_IMGT.human.IGH+IGK+IGL.202412")

## ---------------------------------------------------------------------
## Call igblastn()
## ---------------------------------------------------------------------

out <- tempfile()
igblastn(query, out=out)
AIRR_df <- read_igblastn_AIRR_output(out)

class(AIRR_df)
dim(AIRR_df)  # 1 row per query sequence

tibble(AIRR_df)
```

read_igblastn_fmt7_output

igblastn output format 7

Description

Read and parse `igblastn` output format 7. This is the output produced by `igblastn` when `outfmt` is set to 7.

This format is sometimes called "Tabular with comment lines" or simply "Tabular" format. See for example IgBLAST web interface at https://www.ncbi.nlm.nih.gov/igblast/.

Usage

```
read_igblastn_fmt7_output(out)

## Related utilities:
parse_outfmt7(out_lines)
list_outfmt7_specifiers()
```

Arguments

`out` The path to a file containing the output produced by `igblastn` when `outfmt` is set to 7.

`out_lines` The character vector returned by `igblatsn(query, outfmt=7, parse.out=FALSE, ...)`.

Value

`read_igblastn_fmt7_output(out)` returns `parse_outfmt7(readLines(out))`.

`parse_outfmt7(out_lines)` returns the parsed form of `out_lines` in a list.

`list_outfmt7_specifiers()` returns the list of format specifiers supported by `igblastn` formatting option 7.

See Also

- The `igblastn` function to run the `igblastn` *standalone executable* included in IgBLAST from R. This is the main function in the **igblastr** package.
- `read_igblastn_AIRR_output` to read `igblastn` output format 19 (AIRR format).
- IgBLAST web interface at `https://www.ncbi.nlm.nih.gov/igblast/`.
- IgBLAST is described at `https://pubmed.ncbi.nlm.nih.gov/23671333/`.

Examples

```
if (!has_igblast()) install_igblast()

## ---------------------------------------------------------------------
## Access query sequences and select germline and C-region dbs to use
## ---------------------------------------------------------------------

## Files 'heavy_sequences.fasta' and 'light_sequences.fasta' included
## in igblastr contain 250 paired heavy- and light- chain sequences (125
## sequences in each file) downloaded from OAS (the Observed Antibody
## Space database):
filenames <- paste0(c("heavy", "light"), "_sequences.fasta")
query <- system.file(package="igblastr", "extdata", "BCR", filenames)

## Keep only the first 10 sequences from each file:
query <- c(head(readDNAStringSet(query[[1L]]), n=10),
           head(readDNAStringSet(query[[2L]]), n=10))
```

```
## Select the germline and C-region dbs to use with igblastn():
use_germline_db("_AIRR.human.IGH+IGK+IGL.202410")
use_c_region_db("_IMGT.human.IGH+IGK+IGL.202412")

## ---------------------------------------------------------------------
## FIRST igblastn RUN: GET OUTPUT IN FORMAT 7
## ---------------------------------------------------------------------

parsed_out7 <- igblastn(query, outfmt=7)

## Note that the above is equivalent to:
out <- tempfile()
igblastn(query, outfmt=7, out=out)
parsed_out7b <- read_igblastn_fmt7_output(out)
stopifnot(identical(parsed_out7b, parsed_out7))

## and to:
out_lines <- igblastn(query, outfmt=7, parse.out=FALSE)
out_lines  # raw output
parsed_out7c <- parse_outfmt7(out_lines)
stopifnot(identical(parsed_out7c, parsed_out7))

## Now taking a closer look at the output...

## Output contains one record per query sequence:
length(parsed_out7$records)  # 20

## Each record can have 5 or 6 sections:
##   1. query_details
##   2. VDJ_rearrangement_summary
##   3. VDJ_junction_details
##   4. subregion_sequence_details (can be missing)
##   5. alignment_summary
##   6. hit_table

## Taking a close look at the first record:
rec1 <- parsed_out7$records[[1]]
rec1

qseqid(rec1)    # query sequence id associated with this record

rec1$hit_table  # data.frame with the standard columns

## ---------------------------------------------------------------------
## SECOND igblastn RUN: GET OUTPUT IN CUSTOMIZED FORMAT 7
## ---------------------------------------------------------------------

## For this second run we request a customized format 7 by supplying
## space delimited format specifiers. Use list_outfmt7_specifiers() to
## get the list of format specifiers supported by igblastn formatting
## option 7:
list_outfmt7_specifiers()
outfmt <- "7 qseqid sseqid pident nident length score"
```

```
parsed_out7 <- igblastn(query, outfmt=outfmt)

## Taking a close look at the first record:
rec1 <- parsed_out7$records[[1]]
rec1$hit_table  # data.frame with the requested columns (+ the
                # automatic "chaintype" column)
```

summarizeMismatches — *Summarize mismatches and indels between query and germline sequences*

Description

TODO

translate_codons — *Extract and translate codons from a set of DNA sequences*

Description

The `translate_codons()` function extracts and translates codons from a set of DNA sequences.

Usage

```
translate_codons(dna, offset=0, with.init.codon=FALSE)

## Used internally by translate_codons():
extract_codons(dna, offset=0)
```

Arguments

`dna` A DNAStringSet (or DNAString) object containing the codons to translate.

`offset` The number of nucleotides that precede the first codon to translate. This must be supplied as a numeric vector with one value per sequence in `dna`, or as a single value. If the latter, then the same offset is used for all sequences.

`with.init.codon`

Is the first codon to translate in each DNA sequence the initiation codon? By default, `with.init.codon` is set to `FALSE`, in which case `translate_codons()` assumes that the first codon to translate in each DNA sequence is *not* the initiation codon.

See the `no.init.codon` argument in `?translate` in the **Biostrings** package for more information.

Value

`translate_codons()` returns an AAStringSet object with one amino acid sequence per input sequence.

`extract_codons()` returns a DNAStringSet object with one sequence per input sequence. The output sequences are obtained by trimming the original sequences as follow:

- On their 5' end, sequences are trimmed by the amount of nucleotides specified in `offset`.
- On their 3' end, sequences are trimmed by the smallest amount of nucleotides that makes the length of the trimmed sequence a multiple of 3. Note that this will always be 0, 1, or 2 nucleotides.

See Also

- DNAStringSet and AAStringSet objects in the **Biostrings** package.
- The `translate` function in the **Biostrings** package on which `translate_codons()` is based.
- `list_germline_dbs` to list all the *cached germline dbs*, that is, all the germline databases currently installed in **igblastr**'s persistent cache.

Examples

```
## ---------------------------------------------------------------------
## translate_codons()
## ---------------------------------------------------------------------

## Load germline V gene allele sequences for human:
list_germline_dbs()
db_name <- "_AIRR.human.IGH+IGK+IGL.202410"
V_alleles <- load_germline_db(db_name, region_types="V")
V_alleles  # DNAStringSet object

## Translate them:
V_aa <- translate_codons(V_alleles)
V_aa  # AAStringSet object

## Some human germline V gene allele sequences have a stop codon:
has_stop_codon <- grepl("*", as.character(V_aa), fixed=TRUE)
V_aa[has_stop_codon]

## Handling of initiation codons (see '?translate' in the Biostrings
## package for how initiation codons are handled):
dna2 <- DNAStringSet(c("TTGTCCTTTATA", "GAATCATTTATC", "CTGTCGTTTATT"))
translate_codons(dna2)
translate_codons(dna2, with.init.codon=TRUE)

## ---------------------------------------------------------------------
## extract_codons()
## ---------------------------------------------------------------------
dna <- DNAStringSet(c("CCCAAAGGGTTT",
                      "CCAAAGGGTTT",
                      "CAAAGGGTTT",
                          "AAAGGGTTT"))
extract_codons(dna)
extract_codons(dna, offset=1)
extract_codons(dna, offset=2)
extract_codons(dna, offset=3)
extract_codons(dna, offset=4)

extract_codons(dna, offset=3:0)
extract_codons(dna, offset=4:1)
extract_codons(dna, offset=5:2)
extract_codons(dna, offset=6:3)
```

update_live_igdata *Update and manage IgBLAST auxiliary and internal data*

Description

A small set of low-level utility functions to update and manage IgBLAST *auxiliary data* and *internal data*.

Usage

```
update_live_igdata(check.only=FALSE)

igdata_info()

time_since_live_igdata_last_checked(units="days")

reset_live_igdata(subdirs=c("all", "internal_data", "optional_file"))
```

Arguments

check.only	By default, update_live_igdata() checks for new IgBLAST auxiliary or internal data files available at NCBI, and it installs them if any are found. Set check.only to TRUE to only do the check without installing anything.
units	See ?base::difftime for valid units.
subdirs	By default, reset_live_igdata() resets both internal_data/ and optional_file/ directories to their original states. Set subdirs to "internal_data" or "optional_file" to reset only a particular directory.

Details

Auxiliary data and internal data: A standard IgBLAST installation – like the one used by the **igblastr** package – typically includes *auxiliary data* and *internal data* that are normally found in directories internal_data/ and optional_file/, respectively. Both directories should be subdirectories of the *root directory* of the IgBLAST installation, that is, of the directory returned by get_igblast_root().

We sometimes refer to this data simply as the *IgBLAST data*.

NCBI updates: NCBI occasionally updates some of the files in the internal_data/ and optional_file/ directories between IgBLAST releases, and it is recommended to use the new files. They make the new files available at https://ftp.ncbi.nih.gov/blast/executables/igblast/release/patch/.

To download and install these new files, simply call update_live_igdata(). This will check for new IgBLAST auxiliary or internal data files available at NCBI, and install them if any are found.

You can restore the original files at any moment with reset_live_igdata().

Value

update_live_igdata() and reset_live_igdata() don't return anything (invisible NULL).

igdata_info() returns a named list containing information about the state of the "live" and "original" IgBLAST data.

time_since_live_igdata_last_checked() returns the time passed since the last run of update_live_igdata() in the specified units (days by default).

See Also

- intdata_utils to access and manipulate IgBLAST *internal data*.
- auxdata_utils to access, manipulate, and generate IgBLAST *auxiliary data*.
- The igblastn function to run the igblastn *standalone executable* included in IgBLAST from R. This is the main function in the **igblastr** package.
- install_igblast to perform an *internal* IgBLAST installation.
- get_igblast_root to get (or set) the IgBLAST installation used (or to be used) by the **igblastr** package.
- IgBLAST is described at https://pubmed.ncbi.nlm.nih.gov/23671333/.

Examples

```
if (!has_igblast()) install_igblast()

igblast_info()

## ---------------------------------------------------------------------
## Check for NCBI updates
## ---------------------------------------------------------------------

igdata_info()
update_live_igdata(check.only=TRUE)
igdata_info()

## ---------------------------------------------------------------------
## "live" vs "original" IgBLAST data
## ---------------------------------------------------------------------

## By default, the "live IgBLAST data" gets accessed or returned:
get_auxdata_path("human")
live_human_auxdata <- load_auxdata("human")
```

```
## Access the original IgBLAST data:
get_auxdata_path("human", which="original")
orig_human_auxdata <- load_auxdata("human", which="original")

## "live" and "original" IgBLAST data can differ if the former was
## updated with update_live_igdata(). Otherwise, they'll be the same:
identical(live_human_auxdata, orig_human_auxdata)
```

Index

∗ **manip**
    igblastn, 15
    read_igblastn_AIRR_output, 45
    read_igblastn_fmt7_output, 46
∗ **misc**
    IGBLAST_ROOT, 25
∗ **utilities**
    allele2gene, 3
    augment_germline_db, 4
    auxdata-utils, 7
    compute_auxdata, 11
    compute_intdata, 13
    get_igblast_root, 14
    igblast_info, 23
    igblastr_usage_report, 21
    igbrowser, 26
    install_igblast, 27
    install_IMGT_germline_db, 28
    intdata-utils, 31
    list_c_region_dbs, 35
    list_germline_dbs, 37
    makeogrannote, 41
    OAS-utils, 42
    read_igblastn_AIRR_output, 45
    read_igblastn_fmt7_output, 46
    summarizeMismatches, 49
    translate_codons, 49
    update_live_igdata, 51

AAStringSet, *50*
allele2gene, 3, *33*
annotate_heavy_V_alleles
    (intdata-utils), 31
annotate_light_V_alleles
    (intdata-utils), 31
augment_germline_db, 4, *19*
augment_germline_db_D
    (augment_germline_db), 4
augment_germline_db_J
    (augment_germline_db), 4
augment_germline_db_V
    (augment_germline_db), 4
auxdata (auxdata-utils), 7
auxdata-utils, 7
auxdata_utils, *33*, *52*
auxdata_utils (auxdata-utils), 7

bcr_browser (igbrowser), 26
BLAST_USAGE_REPORT
    (igblastr_usage_report), 21
browseURL, *26*

check_V_ndm_data (makeogrannote), 41
compute_auxdata, *8*, *9*, 11, *13*
compute_intdata, *12*, 13, *33*, *41*

difftime, *51*
DNAString, *49*
DNAStringSet, *4*, *8*, *9*, *11–13*, *15*, *19*, *32*, *33*,
    37, *40*, *42*, *44*, *49*, *50*
download.file, *27*, *29*, *42*
download_paired_OAS_units (OAS-utils),
    42

extract_codons (translate_codons), 49
extract_metadata_from_OAS_units
    (OAS-utils), 42
extract_sequences_from_paired_OAS_df
    (OAS-utils), 42
extract_sequences_from_paired_OAS_units
    (OAS-utils), 42

fmt7-utils (read_igblastn_fmt7_output),
    46
fmt7_utils (read_igblastn_fmt7_output),
    46

get_auxdata_path, *17*
get_auxdata_path (auxdata-utils), 7
get_igblast_auxiliary_data
    (auxdata-utils), 7

get_igblast_root, 14, *23*, *24*, *51*, *52*
get_intdata_path (intdata-utils), 31

has_igblast (igblast_info), 23

igblast_build (igblast_info), 23
igblast_info, *14*, *19*, 23, *25*, *28*
IGBLAST_ROOT, *15*, *24*, 25, *28*
igblastn, *4*, *8*, *9*, *12–14*, 15, *22*, *24–26*, *28*, *30*, *33*, *35–41*, *44*, *45*, *47*, *52*
igblastn_help (igblastn), 15
igblastn_version (igblast_info), 23
igblastr_usage_report, *19*, 21
igbrowser, *19*, 26
igdata_info (update_live_igdata), 51
IMGT_is_up (install_IMGT_germline_db), 28
install_igblast, *14*, *19*, *24*, *25*, 27, *52*
install_IMGT_c_region_db (install_IMGT_germline_db), 28
install_IMGT_germline_db, *19*, 28, *40*
intdata (intdata-utils), 31
intdata-utils, 31
intdata_utils, *3*, *9*, *24*, *38*, *40*, *41*, *52*
intdata_utils (intdata-utils), 31

J_allele_has_stop_codon (auxdata-utils), 7

list_c_region_dbs, 35, *39*
list_germline_dbs, *4*, *17*, *32*, *33*, *36*, 37, *50*
list_igblast_organisms, *8*, *16*, *19*, *32*, *33*
list_igblast_organisms (igblast_info), 23
list_IMGT_organisms (install_IMGT_germline_db), 28
list_IMGT_releases (install_IMGT_germline_db), 28
list_outfmt7_specifiers, *16*, *19*
list_outfmt7_specifiers (read_igblastn_fmt7_output), 46
list_paired_OAS_studies (OAS-utils), 42
list_paired_OAS_units (OAS-utils), 42
load_auxdata, *11*, *12*
load_auxdata (auxdata-utils), 7
load_c_region_db, *3*
load_c_region_db (list_c_region_dbs), 35
load_germline_db, *3*
load_germline_db (list_germline_dbs), 37
load_igblast_auxiliary_data (auxdata-utils), 7
load_intdata, *13*
load_intdata (intdata-utils), 31

makeblastdb_version (igblast_info), 23
makeogrannote, 41

OAS-utils, 42
OAS_utils (OAS-utils), 42
outfmt7-utils (read_igblastn_fmt7_output), 46
outfmt7_utils (read_igblastn_fmt7_output), 46

parse_outfmt7 (read_igblastn_fmt7_output), 46
print.alignment_summary (read_igblastn_fmt7_output), 46
print.auxdata_md5sum_df (update_live_igdata), 51
print.c_region_dbs_df (list_c_region_dbs), 35
print.fmt7footer (read_igblastn_fmt7_output), 46
print.fmt7record (read_igblastn_fmt7_output), 46
print.germline_dbs_df (list_germline_dbs), 37
print.hit_table (read_igblastn_fmt7_output), 46
print.igblast_info (igblast_info), 23
print.igblastn_raw_output (igblastn), 15
print.igdata_info (update_live_igdata), 51
print.outfmt7_specifiers (read_igblastn_fmt7_output), 46
print.query_details (read_igblastn_fmt7_output), 46
print.subregion_sequence_details (read_igblastn_fmt7_output), 46
print.VDJ_junction_details (read_igblastn_fmt7_output), 46
print.VDJ_rearrangement_summary (read_igblastn_fmt7_output), 46

qseqid (read_igblastn_fmt7_output), 46

read.table, *42*

read_igblastn_AIRR_output, 45, *47*
read_igblastn_fmt7_output, *18*, *45*, 46
read_OAS_csv (OAS-utils), 42
read_OAS_csv_metadata (OAS-utils), 42
read_V_ndm_data (makeogrannote), 41
reset_live_igdata (update_live_igdata), 51
rm_c_region_db (list_c_region_dbs), 35
rm_germline_db (list_germline_dbs), 37
Rprofile, *22*

set_igblast_root (get_igblast_root), 14
summarizeMismatches, 49
summary.query_details (read_igblastn_fmt7_output), 46

tabulate_deletions (summarizeMismatches), 49
tabulate_insertions (summarizeMismatches), 49
tabulate_mismatches (summarizeMismatches), 49
tibble, *17–19*, *26*, *42–44*
time_since_live_igdata_last_checked (update_live_igdata), 51
translate, *49, 50*
translate_codons, *9*, *33*, 49
translate_fwr4 (auxdata-utils), 7
translate_J_alleles (auxdata-utils), 7
translate_V_alleles (intdata-utils), 31

update_live_igdata, *8*, *9*, *32*, *33*, 51
Usage_report (igblastr_usage_report), 21
usage_report (igblastr_usage_report), 21
Usage_reporting (igblastr_usage_report), 21
usage_reporting (igblastr_usage_report), 21
use_c_region_db, *16*, *19*, *40*
use_c_region_db (list_c_region_dbs), 35
use_germline_db, *16, 17*, *19*, *30*, *37*
use_germline_db (list_germline_dbs), 37

V_allele_has_stop_codon (intdata-utils), 31
V_genes_with_varying_fwrcdr_boundaries (intdata-utils), 31

write_V_ndm_data (makeogrannote), 41