Package 'rtemis.a3'

November 3, 2026

Title Amino Acid Annotation (A3) Format

Version 0.6.1

Date 2026-11-02

Description Implements the Amino Acid Annotation (A3) format using 'S7' classes.
The A3 format is a structured schema for annotating amino acid sequences with site, region, post-translational modification (PTM), processing event, and variant information.
Provides functions to create, read, and write A3 objects, and to import annotations from 'UniProt', 'AlphaFold', 'PDBe', 'ClinVar', and 'Ensembl'.

URL https://a3.rtemis.org

BugReports https://github.com/rtemis-org/a3/issues

License GPL (>= 3)

Encoding UTF-8

Depends R (>= 4.1.0)

Imports cli, data.table, jsonlite, rtemis.core, S7, stats, utils

Suggests biomaRt, httr, seqinr, testthat (>= 3.0.0)

Config/testthat/edition 3

Config/roxygen2/version 8.1.1

NeedsCompilation no

Author E.D. Gennatas [aut, cre, cph] (ORCID:
<https://orcid.org/0000-0001-9280-3609>)

Maintainer E.D. Gennatas <gennatas@gmail.com>

Repository CRAN

Date/Publication 2026-11-03 00:10:10 UTC

Contents

rtemis.a3-package 2
aa_sub 3
annotation_position 3

annotation_range 4
annotation_variant 5
clinvar_variants 5
concat 7
create_A3 7
gene2sequence 9
get_alphafold 10
pdb_annotations 10
read_A3json 12
uniprot_sequence 12
uniprot_to_A3 13
validate_A3json 14
write_A3json 15

Index **16**

`rtemis.a3-package` *rtemis.a3: Amino Acid Annotation format*

Description

Amino Acid Annotation format utilities

Details

rtemis.a3: Amino Acid Annotation format

Author(s)

Maintainer: E.D. Gennatas <gennatas@gmail.com> (ORCID) [copyright holder]

Authors:

- E.D. Gennatas <gennatas@gmail.com> (ORCID) [copyright holder]

See Also

Useful links:

- https://a3.rtemis.org
- Report bugs at https://github.com/rtemis-org/a3/issues

aa_sub *Perform amino acid substitutions*

Description

Perform amino acid substitutions

Usage

```
aa_sub(x, substitutions, verbosity = 1L)
```

Arguments

x	Character vector: Amino acid sequence. e.g. "ARND" or c("A", "R", "N", "D").
substitutions	Character vector: Substitutions to perform in the format "OriginalPositionNew", e.g. c("C291A", "C322A").
verbosity	Integer: Verbosity level.

Value

Character vector with substitutions performed.

Author(s)

EDG

Examples

```
aa_sub(c("A", "R", "N", "D"), c("R2K", "N3S"))
```

annotation_position *Create position-based annotations for A3*

Description

Creates an annotation spec (named list) with an A3Position index and optional type, for use with create_A3().

Usage

```
annotation_position(x, type = NULL)
```

Arguments

x	Integer vector: Positions of the annotation (1-based indexing).
type	Optional character scalar: Annotation type. NULL is treated as the empty string in the canonical A3 representation.

Value

Named list with index (A3Position) and type (character)

Author(s)

EDG

Examples

```
# Create a site annotation for positions 5 and 17, of type "active site"
annotation_position(c(5L, 17L), type = "active site")
```

---

annotation_range *Create range-based annotations for A3*

Description

Creates an annotation spec (named list) with an A3Range index and optional type, for use with create_A3().

Usage

```
annotation_range(x, type = NULL)
```

Arguments

x	Integer matrix with 2 columns corresponding to start and end positions of the annotation (1-based indexing).
type	Optional character scalar: Annotation type. NULL is treated as the empty string in the canonical A3 representation.

Value

Named list with index (A3Range) and type (character)

Author(s)

EDG

Examples

```
# Create a region annotation for ranges 3-10 and 15-22, of type "repeat"
annotation_range(rbind(c(3L, 10L), c(15L, 22L)), type = "repeat")
```

annotation_variant *Create variant annotations for A3*

Description

Creates an `A3Variant` object for variant annotations with position and info.

Usage

```
annotation_variant(x, info = list())
```

Arguments

`x`	Integer scalar: Position of the variant (1-based indexing).
`info`	Named list: Additional information about the variant (e.g. reference and alternate amino acids, variant type, etc.)

Value

A3Variant object

Author(s)

EDG

Examples

```
# Create a variant annotation for position 10 with info about the variant
annotation_variant(10L, info = list(ref = "A", alt = "T", type = "missense"))
```

clinvar_variants *Fetch ClinVar variants for a gene and parse them into A3Variant objects*

Description

Queries the NCBI ClinVar database via E-utilities for all variants associated with a gene, and returns those with protein-level amino acid changes as a named list of `A3Variant` objects.

Usage

```
clinvar_variants(
  gene,
  significance = c("pathogenic", "likely_pathogenic"),
  max_variants = 500L,
  esearch_url = "https://eutils.ncbi.nlm.nih.gov/entrez/eutils/esearch.fcgi",
  esummary_url = "https://eutils.ncbi.nlm.nih.gov/entrez/eutils/esummary.fcgi",
  batch_size = 500L,
  verbosity = 1L
)
```

Arguments

gene	Character scalar: HGNC gene symbol, e.g. "MAPT".
significance	Character vector: Clinical significance filter. Any combination of "pathogenic", "likely_pathogenic", "uncertain_significance", "likely_benign", "benign". Pass NULL to fetch all records regardless of significance.
max_variants	Integer scalar: Maximum number of ClinVar records to fetch after filtering. Defaults to 500L.
esearch_url	Character scalar: NCBI E-utilities esearch endpoint.
esummary_url	Character scalar: NCBI E-utilities esummary endpoint.
batch_size	Integer scalar: Number of UIDs per esummary request. NCBI recommends no more than 500 per request.
verbosity	Integer scalar: Verbosity level.

Details

Variants without a parseable protein change (intronic, regulatory, etc.) are silently skipped.

Value

Named list of A3Variant objects. Names use {from}{position}{to} notation (e.g. "R406W"), with a numeric suffix to disambiguate duplicates.

Author(s)

EDG

Examples

```
# Requires internet connection and fetches data from ClinVar.
## Not run:
# Pathogenic and likely pathogenic variants (default)
mapt_variants <- clinvar_variants("MAPT")
# All variants regardless of significance
mapt_all <- clinvar_variants("MAPT", significance = NULL)

## End(Not run)
```

concat *Concatenate character vector to single string for sequence input*

Description

Concatenate character vector to single string for sequence input

Usage

```
concat(x)
```

Arguments

x	Character vector to concatenate

Value

Single character string

Author(s)

EDG

Examples

```
# Concatenate a character vector of single-letter amino acids into a sequence string
concat(c("M", "A", "E", "P", "R"))
```

create_A3 *Create an A3 object from sequence, annotations, and metadata*

Description

Create an A3 object from sequence, annotations, and metadata

Usage

```
create_A3(
  sequence,
  site = list(),
  region = list(),
  ptm = list(),
  processing = list(),
  variant = list(),
  uniprot_id = NULL,
  description = NULL,
```

```
  reference = NULL,
  organism = NULL
)
```

Arguments

sequence	Character scalar: Amino acid sequence string.
site	Named list of site annotations
region	Named list of region annotations
ptm	Named list of PTM annotations
processing	Named list of processing annotations
variant	Named list of variant annotations
uniprot_id	Optional character scalar: UniProt accession.
description	Optional character scalar: Protein description.
reference	Optional character scalar: Citation or URL.
organism	Optional character scalar: Species name.

Value

A3 object

Author(s)

EDG

Examples

```
# Minimal: sequence only
a <- create_A3("MAEPRQEFEVMEDHAGTYGLGDRK")

# With site, region, and PTM annotations
a <- create_A3(
  "MAEPRQEFEVMEDHAGTYGLGDRK",
  site = list(
    Active_site = annotation_position(c(5L, 17L), type = "active site")
  ),
  region = list(
    Domain = annotation_range(rbind(c(3L, 10L), c(15L, 22L)),
      type = "repeat"
    )
  ),
  ptm = list(
    Phosphorylation = annotation_position(c(2L, 18L), type = "phosphoserine")
  ),
  uniprot_id = "P10636",
  organism = "Homo sapiens"
)
```

gene2sequence *Get the coding sequence of a gene*

Description

Get the coding sequence of a gene

Usage

```
gene2sequence(
  gene,
  organism = "hsapiens",
  biomart = "ensembl",
  host = "https://www.ensembl.org",
  verbosity = 1L
)
```

Arguments

gene	Character vector: One or more HGNC gene symbols.
organism	Character scalar: Organism short name (Ensembl convention, e.g. "hsapiens").
biomart	Character scalar: BioMart name.
host	Character scalar: Host address.
verbosity	Integer: Verbosity level.

Value

data.frame with columns "gene", "ensembl_transcript_id" and "sequence".

Author(s)

EDG

Examples

```
# Requires internet connection and fetches data from Ensembl using biomaRt.
## Not run:
  mapt_seq <- gene2sequence("MAPT")

## End(Not run)
```

get_alphafold *Get AlphaFold info for a given UniProt ID*

Description

Get AlphaFold info for a given UniProt ID

Usage

```
get_alphafold(uniprotid)
```

Arguments

uniprotid	Character: UniProt ID.

Value

data frame with AlphaFold info.

Author(s)

EDG

Examples

```
# Requires internet connection and fetches data from AlphaFold.
## Not run:
get_alphafold("P10636")

## End(Not run)
```

pdb_annotations *Fetch PDB structural annotations for a UniProt accession*

Description

Queries the PDBe APIs for secondary structure and ligand binding sites from experimental PDB structures, converts residue numbers to UniProt canonical coordinates via SIFTS, and returns named lists of `A3Region` and `A3Site` objects ready for use with `create_A3()`.

Usage

```
pdb_annotations(
  accession,
  pdb_id = NULL,
  pdbe_graph_url = "https://www.ebi.ac.uk/pdbe/graph-api",
  pdbe_api_url = "https://www.ebi.ac.uk/pdbe/api",
  verbosity = 1L
)
```

Arguments

accession	Character scalar: UniProt accession, e.g. "P10636".
pdb_id	Optional character scalar: Four-character PDB ID (e.g. "2mz7"). If NULL, the top-ranked structure from PDBe is used.
pdbe_graph_url	Character scalar: PDBe Graph API base URL.
pdbe_api_url	Character scalar: PDBe REST API base URL.
verbosity	Integer scalar: Verbosity level.

Details

Structure ranking and SIFTS residue mappings come from the PDBe Graph API best_structures endpoint. Secondary structure and binding sites are fetched from the PDBe REST API.

When pdb_id is NULL the top-ranked structure (by PDBe coverage score) is used automatically.

Value

Named list with two elements:

region Named list of A3Region objects for secondary structure (types: "helix", "betaStrand", "turn").

site Named list of A3Site objects for ligand binding sites (type: "bindingSite").

Author(s)

EDG

Examples

```
# Requires internet connection and fetches data from PDBe.
## Not run:
  ann  <- pdb_annotations("P10636")
  mapt <- create_A3(
    sequence = uniprot_sequence("P10636"),
    region   = ann[["region"]],
    site     = ann[["site"]]
  )

## End(Not run)
```

read_A3json *Read* A3 *object from JSON file*

Description

Validation is **collect-all**: the document is checked in four stages and every issue in the earliest failing stage is reported at once, each with a stable code and an RFC 6901 JSON Pointer. Use validate_A3json() for a non-throwing entry point that returns the issues instead.

Usage

```
read_A3json(filepath, verbosity = 1L)
```

Arguments

filepath	Character: Path to JSON file.
verbosity	Integer: if greater than 0, print messages.

Value

A3 object.

Author(s)

EDG

Examples

```
a3 <- create_A3("MAEPRQEFEVMEDHAGTYGLGDRK", uniprot_id = "P10636")
tmp <- tempfile(fileext = ".json")
write_A3json(a3, tmp)
a3_read <- read_A3json(tmp, verbosity = 0L)
unlink(tmp)
```

uniprot_sequence *Fetch a protein sequence from UniProt*

Description

Lightweight FASTA-only fetch. Use this when you only need the amino acid sequence. For full annotations and metadata use uniprot_to_A3().

Usage

```
uniprot_sequence(
  accession,
  base_url = "https://rest.uniprot.org/uniprotkb",
  verbosity = 1L
)
```

Arguments

accession	Character scalar: UniProt accession, e.g. "P10636".
base_url	Character scalar: UniProt REST API base URL.
verbosity	Integer scalar: Verbosity level.

Value

Character scalar: amino acid sequence in single-letter code.

Author(s)

EDG

Examples

```
# Requires internet connection and fetches data from UniProt.
## Not run:
  seq <- uniprot_sequence("P10636")

## End(Not run)
```

---

uniprot_to_A3 *Fetch a UniProt entry and parse it into an A3 object*

Description

Fetches the UniProt JSON for the given accession via the UniProt REST API, maps all sequence annotations to the A3 schema, and returns a fully populated A3 object.

Usage

```
uniprot_to_A3(
  accession,
  base_url = "https://rest.uniprot.org/uniprotkb",
  verbosity = 1L
)
```

Arguments

accession	Character scalar: UniProt accession, e.g. "P10636".
base_url	Character scalar: UniProt REST API base URL.
verbosity	Integer scalar: Verbosity level.

Details

Feature types not mapped to A3 (mutagenesis, alternative sequences, sequence conflicts, etc.) are silently ignored.

Value

A3 object with sequence, annotations, and UniProt metadata.

Author(s)

EDG

Examples

```
# Requires internet connection and fetches data from UniProt.
## Not run:
mapt <- uniprot_to_A3("P10636")

## End(Not run)
```

---

validate_A3json *Validate an A3 JSON file without throwing*

Description

Returns every validation issue found in the earliest failing stage, as a data frame. An empty data frame means the document is valid. Use this when you want to present issues in a form or a report; use read_A3json() when you want the object and an error if it cannot be built.

Usage

```
validate_A3json(filepath)
```

Arguments

filepath	Character scalar: Path to a JSON file.

Details

Each issue carries a stable code and an RFC 6901 JSON Pointer path, both of which are identical across the R, Python, Julia, TypeScript, and Rust implementations of A3. The message is for humans and is not contractual.

Value

Data frame with columns `code`, `stage`, `path`, and `message`, with zero rows when the document is valid.

Author(s)

EDG

Examples

```
a3 <- create_A3("MAEPRQEFEVMEDHAGTYGLGDRK", uniprot_id = "P10636")
tmp <- tempfile(fileext = ".json")
write_A3json(a3, tmp)
validate_A3json(tmp) # zero rows: the document is valid
unlink(tmp)
```

---

write_A3json *Write* `A3` *object to JSON file*

Description

Write `A3` object to JSON file

Usage

```
write_A3json(x, filepath, overwrite = FALSE)
```

Arguments

`x`	`A3` object.
`filepath`	Character: Path to save JSON file.
`overwrite`	Logical: If TRUE, overwrite existing file.

Value

Invisible `x`. Writes JSON file as side effect.

Author(s)

EDG

Examples

```
a3 <- create_A3("MAEPRQEFEVMEDHAGTYGLGDRK", uniprot_id = "P10636")
tmp <- tempfile(fileext = ".json")
write_A3json(a3, tmp)
a3_read <- read_A3json(tmp)
unlink(tmp)
```

Index

aa_sub, 3
annotation_position, 3
annotation_range, 4
annotation_variant, 5

clinvar_variants, 5
concat, 7
create_A3, 7

gene2sequence, 9
get_alphafold, 10

pdb_annotations, 10

read_A3json, 12
read_A3json(), *14*
rtemis.a3 *(*rtemis.a3-package*)*, 2
rtemis.a3-package, 2

uniprot_sequence, 12
uniprot_to_A3, 13
uniprot_to_A3(), *12*

validate_A3json, 14
validate_A3json(), *12*

write_A3json, 15