Package 'AlphaMissenseR'

December 4, 2025

Title Accessing AlphaMissense Data Resources in R

Version 1.7.1

Description The AlphaMissense publication <https://www.science.org/doi/epdf/10.1126/science.adg7492> outlines how a variant of AlphaFold / DeepMind was used to predict missense variant pathogenicity. Supporting data on Zenodo <https://zenodo.org/record/10813168> include, for instance, 71M variants across hg19 and hg38 genome builds. The 'AlphaMissenseR' package allows ready access to the data, downloading individual files to DuckDB databases for exploration and integration into *R* and *Bioconductor* workflows.

License Artistic-2.0

URL https://mtmorgan.github.io/AlphaMissenseR/

BugReports https://github.com/mtmorgan/AlphaMissenseR/issues

Depends R (>= 4.3.0), dplyr

Imports rjsoncons (>= 1.0.1), DBI, duckdb (>= 1.3.1), rlang, curl, BiocFileCache, spdl, memoise, BiocBaseUtils, utils, stats, tools, methods, whisker, ggplot2

Suggests BiocManager, BiocGenerics, S4Vectors, Seqinfo, GenomeInfoDb, GenomicRanges, AnnotationHub, ExperimentHub, ensembldb, httr, tidyr, r3dmol, bio3d, shiny, shiny.gosling, ggdist, colorspace, knitr, rmarkdown, testthat (>= 3.0.0)

biocViews SNP, Annotation, FunctionalGenomics, StructuralPrediction, Transcriptomics, VariantAnnotation, GenePrediction, ImmunoOncology

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

VignetteBuilder knitr

Config/testthat/edition 3

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

git_branch devel

git_last_commit b6b791d

git_last_commit_date 2025-11-26

Repository Bioconductor 3.23

Date/Publication 2025-12-04

Author Martin Morgan [aut, cre] (ORCID:
<https://orcid.org/0000-0002-5874-8148>),
Tram Nguyen [aut] (ORCID: <https://orcid.org/0000-0003-4809-6227>),
Tyrone Lee [ctb],
Nitesh Turaga [ctb],
Chan Zuckerberg Initiative DAF CZF2019-002443 [fnd],
NIH NCI ITCR U24CA180996 [fnd],
NIH NCI IOTN U24CA232979 [fnd],
NIH NCI ARTNet U24CA274159 [fnd]

Maintainer Martin Morgan <mtmorgan.xyz@gmail.com>

Contents

af_predictions 2
ALPHAMISSENSE_RECORD 5
am_aa_pos 7
clinvar_data 8
db_connect 10
gosling_plot 12
to_GPos 14

Index **15**

`af_predictions` *AlphaFold Protein Structure Retrieval and Display*

Description

`af_predictions()` retrieves information about AlphaFold predictions associated with UniProt accession identifiers.

`af_prediction_view()` summarizes effects of possible amino acid changes in a single UniProt protein. The changes are displayed on the AlphaFold-predicted structure.

`af_colorfunc_by_position()` generates a Javascript function to be used in `rd3mol::m_set_style()` to color residues by position, e.g., when visualizing median predicted pathogenicity.

Usage

```
af_predictions(uniprot_ids)

af_prediction_view(tbl, bfc = BiocFileCache())

af_colorfunc_by_position(
    tbl,
    pos,
    value,
    pos_max = NULL,
    palette = colorspace::diverging_hcl(11),
    palette_min = NULL,
    palette_max = NULL
)
```

Arguments

uniprot_ids	character() UniProt accession identifiers (uniprot_id in AlphaMissense tables).
tbl	A tibble containing information on the UniProt protein and AlphaMissense predicted amino acid effects. For av_prediction_view() the tibble must have columns uniprot_id, protein_variant, am_pathogenicity, and am_class, as in tibbles returned by am_data("hg38") or am_data("aa_substitutions"), for instance. The uniprot_id must contain a single unique value. For af_colorfunc_by_position() the tibble must have columns pos and value, as described below.
bfc	An object created with BiocFileCache::BiocFileCache(), representing the location used to cache PDB files retrieved by av_prediction_view(). The default is the BiocFileCache installation-wide location.
pos	the symbol or name of the column in tbl containing amino acid residue positions in the protein.
value	the symbol or name of the column in tbl containing values to be used for coloring amino acid residues in the protein.
pos_max	integer(1) the maximum residue position in the protein to be visualized. Default: the maximum value in pos.
palette	character() vector of colors to be used in visualization. The default (colorspace::diverging_hcl(11)) produces colors ranging from blue (low) to red (high).
palette_min	numeric(1) the value bounding the minimum palette color. The default is the minimum of value; a common value when plotting pathogenicity might be 0.
palette_max	numeric(1) the value bounding the maximum palette color. The default is the maximum of value; a common value when plotting pathogenicity might be 1.

Details

af_predictions() queries the prediction endpoint of the AlphaFold API described at https://alphafold.ebi.ac.uk/api-docs.

`af_prediction_view()` uses `tbl` to calculate median pathogenicity at each amino acid position, using `am_aa_pathogenicity()`. Predicted protein structure is retrieved from the unique `uniprot_id` using `af_predictions()` and the `pdbUrl` returned by that function. Protein structure is visualized using the r3dmol https://cran.R-project.org/package=r3dmol package. Amino acids are colored using `aa_pathogenicity_median` and `af_colorfunc_by_position()` with default palette defined on the interval 0, 1.

`af_colorfunc_by_position()` uses a template mechanism to inject a vector of position-specific colors into a Javascript function used by `r3dmol::m_set_style()` / `r3dmol::m_style_cartoon()` to color residues by position. Positions for which no color is specified are colored `'gray'`. The template can be seen with `AlphaMissenseR:::js_template("colorfunc")`.

Value

`af_predictions()` returns a tibble. Each row represents the AlphaFold prediction associated with the corresponding uniprot accession. Columns include:

- `entryId`: AlphaFold identifier.
- `gene`: gene symbol corresponding to UniProt protein.
- `uniprotAccession`, `uniprotId`, `uniprotDescription`: UniProt characterization. AlphaMissense's `uniprot_id` is AlphaFold's `uniprotAccession`.
- `taxId`, `organismScientificName`: Organism information.
- `uniprotStart`, `uniprotEnd`, `uniprotSequence`: protein sequence information.
- `modelCreatedDate`, `latestVersion`, `allVersions`, `isReviewed`, `isReferenceProteome`: AlphaFold provenance information.
- `cifUrl`, `bcifUrl`, `pdbUrl`: URLs to AlphaFold 3-dimensional molecular representations.
- `paeImageUrl`, `paeDocUrl`: 'Predicted Aligned Error' heat map and underlying data. These can be used to assess the confidence in relative orientation of residues in different domains, as described in part in the AlphaFold FAQ https://alphafold.ebi.ac.uk/faq

`af_prediction_view()` displays an interactive view of the protein in an RStudio panel or browser tab.

`af_colorfunc_by_position()` returns a character(1) vector representation of the Javascript function, with color vector injected.

Examples

```
## af_predictions

uniprot_ids <-
    am_data("aa_substitutions") |>
    dplyr::filter(uniprot_id %like% "P3555%") |>
    dplyr::distinct(uniprot_id) |>
    pull(uniprot_id)
af_predictions(uniprot_ids)


## af_prediction_view()
```

```
P35557 <-
    am_data("hg38") |>
    dplyr::filter(uniprot_id == "P35557")
af_prediction_view(P35557)

## no AlphaFold prediction for this protein
P35555 <-
    am_data("aa_substitutions") |>
    dplyr::filter(uniprot_id == "P35555")
tryCatch({
    af_prediction_view(P35555)
}, error = identity)


## af_colorfunc_by_position()

df <- tibble(
    pos = 1 + 1:10, # no color information for position 1
    value = 10:1 / 10
)
colorfunc <- af_colorfunc_by_position(
    df,
    "pos", "value",
    pos_max = 12    # no color information for position 12
)
cat(colorfunc)

## template used for Javascript function
cat(
    AlphaMissenseR:::js_template("colorfunc", colors = "..."),
    "\n"
)
```

`ALPHAMISSENSE_RECORD` *Retrieve AlphaMissense Resources as DuckDB Databases*

Description

`ALPHAMISSENSE_RECORD` is a constant identifier corresponding to the default version of the AlphaMissense resource to use.

`am_browse()` opens a web browser at the Zenodo record for the AlphaMissense data.

`am_available()` reports available datasets in the record.

`am_data()` retrieves a single key from the the AlpahMissense Zenodo site and parses the file into a DuckDB database.

Usage

```
ALPHAMISSENSE_RECORD

am_browse(record = ALPHAMISSENSE_RECORD)

am_available(record = ALPHAMISSENSE_RECORD, bfc = BiocFileCache())

am_data(
    key,
    record = ALPHAMISSENSE_RECORD,
    bfc = BiocFileCache(),
    as = c("tbl", "tsv")
)
```

Arguments

record	character(1) Zenodo record for the AlphaMissense data resources.
bfc	an object returned by BiocFileCache() representing the location where downloaded files and the parsed database will be stored. The default is the 'global' BiocFileCache.
key	a character(1) 'key' from the result of am_available(), or a single row of the tibble returned by am_available().
as	chracter(1) type of return value.

- "tbl": a dbplyr tbl representation of the database resource.
- "tsv": path to the tsv.gz file representing the resource and downloaded from Zenodo

Format

An object of class character of length 1.

Details

ALPHAMISSENSE_RECORD can be set **before the package is loaded** with the environment variable of the same name, e.g., Sys.setenv(ALPHAMISSENSE_RECORD = "10813168"). The default is the most recent version (version 3) as checked on 11 April, 2024.

am_data() uses BiocFileCache to download and store the file and the corresponding DuckDB database.

Value

am_available() returns a tibble with columns key, size, and link. The meaning of key must be determined with reference to the information at am_browse().

am_data() returns a dbplyr (database) tibble represented the downloaded and parsed file. Fields in the database are as described on the Zenodo resource page.

Examples

```
ALPHAMISSENSE_RECORD

if (interactive())
    am_browse()

am_available()

am_data("hg38")

## close the connection opened when adding the data
db_disconnect()
```

---

am_aa_pos *Common Amino Acid-level Transformations*

Description

am_aa_pos() separates protein_variant columns into amino acide 'pos', 'ref', and 'alt' columns.

am_aa_pathogenicity() summarizes pathogenicity scores at each protein amino acid position.

Usage

```
am_aa_pos(tbl)

am_aa_pathogenicity(tbl)
```

Arguments

tbl	a tibble, usually derived from am_data("aa_substitutions"), 'am_data("hg38"), etc. See details.

Details

tbl is collect()ed before computation, so all rows must fit into memory.

For am_aa_pos(), tbl must contain a column protein_variant with entries in the form "Q465H", as in the AlphaMissense data.

For am_aa_pathogenicity(), tbl must contain columns uniprot_id, protein_variant, am_pathogenicity and am_class. If am_pos and friends are not already calculated, then am_aa_pos() is called.

Value

am_aa_pos() returns the original table with additional columns

- aa_pos: the position of the protein variant, as an integer().
- aa_ref: the single-character reference amino acid in the protein variant.

- aa_alt: the single-character alternate amino acid in the protein variant.

am_aa_pathogenicity() returns a tibble with columns

- uniprot_id, aa_pos, aa_ref: the UniProt id, and the position and reference amino acid being summarized
- aa_pathogenicity_n, aa_pathogenicity_mean, aa_pathogenicity_median, aa_pathogenicity_min, aa_pathogenicity_max: the number, average, median, minimum, and maximum of the pathogenicity scores at each amino acid position.
- aa_pathogenicity_mode: the modal am_class at the amino acid position, as a factor. Tied mode is assigned to lower pathogenicity.

Examples

```
P35557 <-
    am_data("hg38") |>
    filter(uniprot_id %in% "P35557")

am_aa_pos(P35557)

am_aa_pos(P35557) |>
    select(
        uniprot_id, POS, REF, ALT, protein_variant,
        starts_with("aa_"), am_pathogenicity, am_class
    ) |>
    arrange(aa_pos)


am_aa_pathogenicity(P35557)
```

---

clinvar_data *Integrate ClinVar Labels with AlphaMissense Pathogenicity Scores*

Description

clinvar_data() loads ClinVar information from the supplemental table of the AlphaMissense [2023] paper.

clinvar_plot() integrates ClinVar classifications with AlphaMissense predicted scores derived from am_data("aa_substitutions") and returns a ggplot object for visualization.

Usage

```
clinvar_data(record = ALPHAMISSENSE_RECORD, bfc = BiocFileCache())

clinvar_plot(uniprotId, alphamissense_table, clinvar_table)
```

Arguments

record	character(1) Zenodo record for the AlphaMissense data resources.
bfc	an object returned by BiocFileCache() representing the location of the AlphaMissenseR database. The default is the 'global' BiocFileCache.
uniprotId	character() a valid UniProt accession identifier.
alphamissense_table	a table containing AlphaMissense predictions for protein variants. By default, the table is derived from am_data("aa_substitution"). Alternatively, a user-defined tibble::tbl_df or data.frame can be supplied.
clinvar_table	a table containing ClinVar information. By default, the table is derived from the supplemental data of the AlphaMissense paper. Alternatively, a user-defined tibble::tbl_df or data.frame can be supplied.

Details

For clinvar_plot(), alphamissense_table columns must include:

- uniprot_id: UniProt accession identifier.
- protein_variant: variant identifier string, with protein position in the middle, and the reference and mutant amino acid residues to the left and right of the position, respectively.
- am_class: AlphaMissense classification of either "benign", "ambiguous", or "pathogenic".
- am_pathogenicity: AlphaMissense predicted score.

clinvar_table columns must include:

- uniprot_id: UniProt accession identifier, matching alphamissense_table.
- protein_variant: variant identifier string, matching alphamissense_table format.
- cv_class: binary ClinVar classification of "benign" or "pathogenic".

Value

clinvar_data() returns a tbl with 82872 rows and 5 variables:

- variant_id: ClinVar variant identifier.
- transcript_id: Ensembl transcript identifier.
- protein_variant: UniProt accession:protein variant identifier.
- AlphaMissense: AlphaMissense pathogenicity score.
- label: Binary ClinVar class, either "benign" or "pathogenic".

clinvar_plot() returns a ggplot object which overlays ClinVar classifications onto AlphaMissense predicted scores. Blue, gray, and red colors represent pathogenicity classifications for "likely benign", "ambiguous", or "likely pathogenic", respectively. Large, bolded points are ClinVar variants colored according to their clinical classification, while smaller points in the background are AlphaMissense predictions.

References

Cheng et al., Accurate proteome-wide missense variant effect prediction with AlphaMissense. *Science* 381, eadg7492. DOI:10.1126/science.adg7492.

Examples

```
clinvar_data()

alphamissense_table <- am_data("aa_substitutions")

clinvar_plot(
    uniprotId = "P37023",
    alphamissense_table = alphamissense_table
)
```

---

db_connect *Manipulate the Database of Missense Mutations*

Description

`db_connect()` manages connections to AlphaMissense record-specific databases. By default, connections are created once and reused.

`db_tables()` queries for the names of temporary and regular tables defined in the database.

`db_temporary_table()` creates a temporary (for the duration of the duckdb connection) table from a tibble.

`db_range_join()` performs a range join, finding all positions in `key` within ranges defined by `join`. The result is stored in table `to`.

`db_disconnect()` disconnects the duckdb database and shuts down the DuckDB server associated with the connection. Temporary tables are lost.

`db_disconnect_all()` disconnects all managed duckdb database connection.

Usage

```
db_connect(
    record = ALPHAMISSENSE_RECORD,
    bfc = BiocFileCache(),
    read_only = TRUE,
    managed = read_only
)

db_tables(db = db_connect())

db_temporary_table(db, value, to)

db_range_join(db, key, join, to)
```

```
db_disconnect(db = db_connect())

db_disconnect_all()
```

Arguments

record	character(1) Zenodo record for the AlphaMissense data resources.
bfc	an object returned by BiocFileCache() representing the location where downloaded files and the parsed database will be stored. The default is the 'global' BiocFileCache.
read_only	logical(1) open the connection 'read only'. TRUE protects against overwriting existing data and is the default.
managed	logical(1) when TRUE, re-use an existing managed connection to the same database.
db	duckdb_connection object, returned by db_connect().
value	a data.frame / tibble containing data to be placed in a temporary table, e.g., from a GenomicRanges object to be used in a range join.
to	the character(1) name of the table to be created
key	a character(1) table name in db containing missense mutation coordinates.
join	a character(1) table name in db containing ranges to be used for joining with (filtering) key.

Details

For db_connect(), set managed = FALSE when, for instance, accessing a database in a separate process. Remember to capture the database connection db_unmanaged <- db_connect(managed = FALSE) and disconnect when done 'db_disconnect(db_unmanaged). Connections are managed by default.

db_temporary_table() **overwrites** an existing table with name to.

db_range_join() **overwrites** an existing table to. The table key is usually "hg19" or "hg38" and must have CHROM and POS columns. The table join must have columns CHROM, start and end. Following *Bioconductor* convention and as reported in am_browse(), coordinates are 1-based and ranges defined by start and end are closed. All columns from both key and join are included, so column names (other than CHROM) cannot be duplicated.

db_disconnect() should be called on each unmanaged connection, and once (to free the default managed connection) at the end of a session.

Value

db_connect() returns an open duckdb_connection to the AlphaMissense record-specific database.

db_tables() returns a character vector of database table names.

db_temporary_table() returns the temporary table as a dbplyr tibble.

db_range_join() returns to (the temporary table created from the join) as a dbplyr tibble.

db_disconnect() returns FALSE if the connection has already been closed or is not valid (via dbIsValid()) or TRUE if disconnection is successful. Values are returned invisibly.

db_disconnect_all() returns the db_disconnect() value for each connection, invisibly.

Examples

```
db_connect()          # default 'read-only' connection

db_rw <- db_connect(read_only = FALSE)

am_data("hg38")       # uses the default, 'read-only' connection
db_tables()           # connections initially share the same tables
db_tables(db_rw)

## ranges of interest -- the first 200000 bases on chromsomes 1-4.
ranges <- tibble(
    CHROM = paste0("chr", 1:4),
    start = rep(1, 4),
    end = rep(200000, 4)
)
db_temporary_table(db_rw, ranges, "ranges")

db_tables(db_rw)      # temporary table available to the db_rw connection...
db_tables()           # ...but not to the read-only connection

rng <- db_range_join(db_rw, "hg38", "ranges", "ranges_overlaps")
rng
rng |>
    count(CHROM) |>
    arrange(CHROM)

db_disconnect(db_rw)  # explicit read-write connection
db_disconnect()       # implicit read-only connection

db_disconnect_all()
```

`gosling_plot` *Plot a GPos or GRanges Object Using Shiny and Gosling*

Description

`plot_granges()` creates a Shiny app that displays a Gosling plot of a given GRanges object. It visualizes genomic ranges with both rectangle and point representations, and allows for customization of the plot title and subtitle.

Usage

```
gosling_plot(
  granges,
  title = "GRanges Plot",
  subtitle = "Stacked nucleotide example",
  plot_type = c("bar", "lollipop"),
  create_app = interactive()
)
```

Arguments

granges	A GenomicRanges::GRanges (e.g., GenomicRanges::GPos) object containing the genomic ranges to be plotted.
title	character(1) Title of the plot. Default is "GRanges Plot".
subtitle	character(1) The subtitle of the plot. Default is "Stacked nucleotide example".
plot_type	character(1) Select the type of gosling plot. Default is "bar"; available types are described in Details.
create_app	logical(1) Produce a Shiny app for Gosling visualization. Default TRUE when used interactively.

Details

The function supports 2 types of plots as selected through the plot_type argument: (1) "bar": a barplot-like view of the pathogenicity score at each position, similar to a sequencing coverage plot, and (2) "lollipop": a lollipop plot focusing on the pathogenicity classification (ambiguous, benign, pathogenic) at each position. It requires that the GenomicRanges::GRanges object has the following metadata columns: 'am_class' (effect classification),'am_pathogenicity' (pathogenicity score), 'ALT' (alternative allele) and 'REF' (reference allele).

Value

gosling_plot() with create_app = TRUE returns a shinyApp object that, when run, displays the Gosling plot. When create_app = FALSE, the (invisible) return value represents the gosling object to be used in Shiny apps.

Note

This function requires the shiny, shiny.gosling, and GenomicRanges packages to be installed.

Examples

```
## Create a sample GRanges object from AlphamissenseR
gpos <-
    am_data("hg38") |>
    filter(uniprot_id == "Q1W6H9") |>
    to_GPos()

## Plot `gpos` using shiny.gosling
if (requireNamespace("shiny.gosling", quietly = TRUE)) {
   gosling_plot(
       gpos, title = "Q1W6H_track", subtitle = "bar plot example",
       plot_type = "bar", create_app = FALSE
   )
}
```

to_GPos *Create GenomicRanges Objects from AlphaMissense Annotations*

Description

to_GPos() coerces a tibble derived from am_data("hg19") or am_data("hg38") resources to GenomicRanges GPos objects.

Usage

```
to_GPos(tbl)
```

Arguments

tbl	a tibble derived from am_data("hg19") or am_data("hg38"). The tibble must have columns CHROM, POS, and genome.

Value

to_GPos() returns a GPos object, which can be used in the same was a GRanges object for range-based filtering and annotation

Examples

```
am_data("hg38") |>
    filter(CHROM == "chr2", POS < 10000000, REF == "G") |>
    select(-REF) |>
    to_GPos()
```

Index

∗ **datasets**
ALPHAMISSENSE_RECORD, 5

af_colorfunc_by_position
(af_predictions), 2
af_prediction_view (af_predictions), 2
af_predictions, 2
ALPHAMISSENSE_RECORD, 5
am_aa_pathogenicity (am_aa_pos), 7
am_aa_pos, 7
am_available (ALPHAMISSENSE_RECORD), 5
am_browse (ALPHAMISSENSE_RECORD), 5
am_data (ALPHAMISSENSE_RECORD), 5

clinvar_data, 8
clinvar_plot (clinvar_data), 8

data.frame, *9*
db_connect, 10
db_disconnect (db_connect), 10
db_disconnect_all (db_connect), 10
db_range_join (db_connect), 10
db_tables (db_connect), 10
db_temporary_table (db_connect), 10

GenomicRanges::GPos, *13*
GenomicRanges::GRanges, *13*
gosling_plot, 12

tibble::tbl_df, *9*
to_GPos, 14