Package 'immundata'

April 8, 2025

Title A Unified Data Layer for Single-Cell, Spatial and Bulk Immunomics

Version 0.0.1

Description Provides a unified data layer for single-cell, spatial and bulk T-cell and B-cell immune receptor repertoire data, integrating diverse data formats such as AIRR and raw sequencing files. Includes utilities for receptor aggregation, metadata normalization, and clonotype filtering.

License Apache License (>= 2)

Encoding UTF-8

RoxygenNote 7.3.2

Depends R (>= 4.1.0)

Imports cli, R6, checkmate, dplyr, duckplyr, glue, lifecycle, readr, rlang

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation no

Author Vadim I. Nazarov [aut, cre] (<https://orcid.org/0000-0003-3659-2709>)

Maintainer Vadim I. Nazarov <support@immunomind.com>

Repository CRAN

Date/Publication 2025-04-08 15:40:07 UTC

Contents

imd_files *Get Immundata Default File Names*

Description

Returns the standardized default filenames for storing receptor-level and annotation-level data as used in `load_repertoires()` and related Immundata I/O functions.

Usage

```
imd_files()
```

Value

A named list of file names (e.g., `receptors.parquet`, `annotations.parquet`).

imd_schema *Get Immundata Internal Schema Field Names*

Description

Returns the standardized field names used across Immundata objects and processing functions, as defined in `IMD_GLOBALS$schema`. These include column names for barcodes, receptors, repertoires, and related metadata.

Usage

```
imd_schema()
```

Value

A named list of schema field names.

ImmunData *ImmunData: A Unified Structure for Immune Receptor Repertoire Data*

Description

ImmunData is an abstract R6 class for managing and transforming immune receptor repertoire data. It supports flexible backends (e.g., Arrow, DuckDB, dbplyr) and lazy evaluation, and provides tools for filtering, aggregation, and receptor-to-repertoire mapping.

Public fields

`.receptors` A receptor-level table containing immune receptor features (e.g., CDR3, V/J gene, clonotype ID, counts). This table is typically aggregated and is used for quantitative analysis of immune repertoire signatures. It can be a local tibble, Arrow Table, DuckDB table, or any other dplyr-compatible backend (including lazy data sources).

`.annotations` A barcode-level table that links each barcode (i.e., cell ID) to a receptor in `.receptors`. It can also store cell-level metadata such as sample ID, donor, or tissue source. This table is **not aggregated** and typically contains one row per barcode.

`schema_receptor` A named list describing how to interpret receptor-level data. This includes the fields used for aggregation (e.g., `CDR3`, `V_gene`, `J_gene`), and optionally unique identifiers for each receptor row. Used to ensure consistency across processing steps.

`schema_repertoire` A named list defining how barcodes or annotations should be grouped into repertoires. This may include sample-level metadata (e.g., `sample_id`, `donor_id`) used to define unique repertoires.

Active bindings

`receptors` Accessor for the receptor-level table (`.receptors`).

`annotations` Accessor for the annotation-level table (`.annotations`).

Methods

Public methods:

- `ImmunData$new()`
- `ImmunData$build_repertoires()`
- `ImmunData$print()`
- `ImmunData$filter_receptors()`
- `ImmunData$filter_annotations()`
- `ImmunData$filter_barcodes()`
- `ImmunData$clone()`

Method `new()`**:** Creates a new ImmunData object. This constructor expects receptor-level and barcode-level data, along with a receptor schema defining aggregation and identity fields.

Usage:

```
ImmunData$new(receptors, annotations, schema)
```

Arguments:

`receptors` A receptor-level dataset (e.g., grouped by CDR3/V/J).

`annotations` A cell/barcode-level dataset mapping barcodes to receptor rows.

`schema` A named list specifying the receptor schema (e.g., aggregate fields, ID columns).

Method `build_repertoires()`**:** Defines repertoires by concatenating selected annotation columns.

Usage:

```
ImmunData$build_repertoires(schema = "repertoire_id", sep = "-")
```

Arguments:

`schema` A character vector of column names in `.annotations` used to define repertoire grouping.

`sep` A string separator used to concatenate multiple columns into a single repertoire ID.

Method `print()`**:** Prints class information for the `ImmunData` object.

Usage:

```
ImmunData$print()
```

Method `filter_receptors()`**:** Filters the receptor-level data using tidyverse filter syntax, and then updates the annotation table to include only linked barcodes.

Usage:

```
ImmunData$filter_receptors(...)
```

Arguments:

`...` Filtering conditions applied to the receptor-level table.

Returns: A new `ImmunData` object with filtered receptor and annotation tables.

Method `filter_annotations()`**:** Filters the annotation-level table using tidyverse filter syntax, and updates the receptor table to include only matching receptor entries.

Usage:

```
ImmunData$filter_annotations(...)
```

Arguments:

`...` Filtering conditions applied to the annotations table.

Returns: A new `ImmunData` object with filtered annotation and receptor tables.

Method `filter_barcodes()`**:** Filters the dataset by a set of barcodes (unique cell IDs). The resulting object contains only rows linked to those barcodes.

Usage:

```
ImmunData$filter_barcodes(barcodes = c())
```

Arguments:

`barcodes` A character vector of barcodes to retain.

Returns: A new `ImmunData` object filtered by barcode.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ImmunData$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`load_immundata` *Load a Saved ImmunData Object from Disk*

Description

Loads an `ImmunData` object from a directory created by Immundata I/O functions (e.g., `load_repertoires()` with file splitting or saving). It reads receptor-level and annotation-level data from standardized Parquet files and reconstructs a new `ImmunData` object with inferred schema.

This function expects two files inside the provided directory:

- `receptors.parquet`: contains the receptor-level table
- `annotations.parquet`: contains the annotation-level table

These filenames are defined in `imd_files()` and follow the Immundata storage convention.

Usage

```
load_immundata(path, verbose = TRUE)
```

Arguments

`path`	Path to the folder containing the saved ImmunData files.
`verbose`	Logical. If `TRUE` (default), prints progress messages to the console.

Value

A new `ImmunData` object containing receptor and annotation data.

See Also

`imd_files()`, ImmunData, `duckplyr::read_parquet_duckdb()`

load_metadata *Load and Validate Metadata Table for Immune Repertoire Files*

Description

#' This function loads a metadata table from either a file path or a data frame, validates the presence of a column with repertoire file paths, and converts all file paths to absolute paths. It is used to support flexible pipelines for loading bulk or single-cell immune repertoire data across samples.

If the input is a file path, the function attempts to read it with `readr::read_delim`. If the input is a data frame, it checks whether file paths are absolute; relative paths are only allowed when metadata is loaded from a file.

It warns the user if many of the files listed in the metadata table are missing, and stops execution if none of the files exist.

The column with file paths is normalized and renamed to match the internal filename schema.

Usage

```
load_metadata(metadata, filename_col = "File", delim = "\t", ...)
```

Arguments

`metadata`	A metadata table. Can be either: • a data frame with metadata, • or a path to a text/TSV/CSV file that can be read with `readr::read_delim`.
`filename_col`	A string specifying the name of the column in the metadata table that contains paths to repertoire files. Defaults to `"File"`.
`delim`	Delimiter used to read the metadata file (if a path is provided). Defaults to `"\t"`.
`...`	Additional arguments passed to `readr::read_delim()` when reading metadata from a file.

Value

A validated and updated metadata data frame with absolute file paths, and an additional column renamed according to `IMD_GLOBALS$schema$filename`.

load_repertoires *Load and Aggregate Immune Receptor Repertoire Data*

Description

This function ingests a repertoire dataset (Parquet, CSV, or TSV), aggregates receptors based on a user-defined schema, and splits the result into receptor-level and annotation-level tables. The resulting data is saved to a designated output folder as two Parquet files (receptors and annotations) and then reloaded to create an ImmunData object.

Usage

```
load_repertoires(
  path,
  schema,
  metadata = NULL,
  barcode_col = NULL,
  count_col = NULL,
  repertoire_schema = NULL,
  output_folder = NULL,
  enforce_schema = TRUE,
  verbose = TRUE
)
```

Arguments

path	Path to an input file. This file may be Parquet, CSV, or TSV. The file extension is automatically detected and handled.
schema	Character vector defining which columns in the input data should be used to identify unique receptor signatures. For example, c("V_gene", "J_gene", "CDR3_nt").
metadata	An optional data frame containing additional metadata to merge into the annotation table. Default is NULL.
barcode_col	An optional character string specifying the column in the input data that represents cell barcodes or other unique identifiers. Default is NULL.
count_col	An optional character string specifying the column in the input data that stores bulk receptor counts. Default is NULL.
repertoire_schema	An optional character vector defining how annotations should be grouped into repertoires (for example, c("sample", "donor")). Currently unused in this function, but reserved for future expansions. Default is NULL.
output_folder	Character string specifying the directory to save the resulting Parquet files. If NULL, a folder named immundata-<basename_of_path> is created in the same directory as path.
enforce_schema	Logical. If TRUE, column names and types must strictly match between files. If FALSE, columns are unioned
verbose	. Logical. Not used – for now.

Details

1. **Reading** – The function automatically detects whether `path` points to a Parquet, CSV, or TSV file, using `read_parquet_duckdb` or `read_csv_duckdb`.
2. **Aggregation** – Receptor uniqueness is determined by the columns named in `schema`, while barcodes or counts are handled depending on which parameters (`barcode_col`, `count_col`) are provided.
3. **Saving** – The final receptor-level and annotation-level tables are written to Parquet files in `output_folder`.
4. **Reloading** – The function calls `load_immundata()` on the newly created folder to return a fully instantiated `ImmunData`.

See Also

`load_immundata()`, ImmunData

Index