Package: CuratedAtlasQueryR (via r-universe)

November 7, 2026

Type Package

Title Queries the Human Cell Atlas

Version 1.10.0

Description Provides access to a copy of the Human Cell Atlas, but with harmonised metadata. This allows for uniform querying across numerous datasets within the Atlas using common fields such as cell type, tissue type, and patient ethnicity. Usage involves first querying the metadata table for cells of interest, and then downloading the corresponding cells into a SingleCellExperiment object.

License GPL-3

Depends R (>= 4.2.0)

Imports dplyr, SummarizedExperiment, SingleCellExperiment, purrr (>= 1.0.0), BiocGenerics, glue, HDF5Array, DBI, tools, httr, cli, assertthat, SeuratObject, Seurat, methods, rlang, stats, S4Vectors, tibble, utils, dbplyr (>= 2.3.0), duckdb, stringr

Suggests zellkonverter, rmarkdown, knitr, testthat, basilisk, arrow, reticulate, spelling, forcats, ggplot2, tidySingleCellExperiment, rprojroot

Biarch true

biocViews AssayDomain, Infrastructure, RNASeq, DifferentialExpression, GeneExpression, Normalization, Clustering, QualityControl, Sequencing, Transcription, Transcriptomics

Encoding UTF-8

RoxygenNote 7.2.3

LazyDataCompression xz

URL https://github.com/stemangiola/CuratedAtlasQueryR

BugReports https://github.com/stemangiola/CuratedAtlasQueryR/issues

VignetteBuilder knitr

Roxygen list(markdown = TRUE)

Collate 'utils.R' 'counts.R' 'dev.R' 'metadata.R' 'seurat.R' 'unharmonised.R' 'zzz.R'

Language en-US

PackageStatus Deprecated

Config/pak/sysreqs cmake libglpk-dev make libicu-dev libpng-dev libuv1-dev libxml2-dev libssl-dev python3 zlib1g-dev

Repository Bioconductor 3.23

Date/Publication 2026-04-28 13:01:11 UTC

RemoteUrl https://github.com/bioc/CuratedAtlasQueryR

RemoteRef RELEASE_3_23

RemoteSha 553d9afbb96a328a235f491078a3b237d042f9cd

Contents

DATABASE_URL 2
get_metadata 3
get_seurat 5
get_single_cell_experiment 6
get_SingleCellExperiment 7
get_unharmonised_dataset 8
get_unharmonised_metadata 9
SAMPLE_DATABASE_URL 10

Index **11**

`DATABASE_URL` *URL pointing to the full metadata file*

Description

URL pointing to the full metadata file

Usage

```
DATABASE_URL
```

Format

An object of class `character` of length 1.

Value

A character scalar consisting of the URL

Examples

```
get_metadata(remote_url = DATABASE_URL)
```

`get_metadata` *Gets the Curated Atlas metadata as a data frame.*

Description

Downloads a parquet database of the Human Cell Atlas metadata to a local cache, and then opens it as a data frame. It can then be filtered and passed into `get_single_cell_experiment()` to obtain a `SingleCellExperiment::SingleCellExperiment`

Usage

```
get_metadata(
  remote_url = DATABASE_URL,
  cache_directory = get_default_cache_dir(),
  use_cache = TRUE
)
```

Arguments

`remote_url`	Optional character vector of length 1. An HTTP URL pointing to the location of the parquet database.
`cache_directory`	Optional character vector of length 1. A file path on your local system to a directory (not a file) that will be used to store `metadata.parquet`
`use_cache`	Optional logical scalar. If `TRUE` (the default), and this function has been called before with the same parameters, then a cached reference to the table will be returned. If `FALSE`, a new connection will be created no matter what.

Details

The metadata was collected from the Bioconductor package `cellxgenedp`. it's vignette `using_cellxgenedp` provides an overview of the columns in the metadata. The data for which the column `organism_name` included "Homo sapiens" was collected collected from `cellxgenedp`.

The columns `dataset_id` and `file_id` link the datasets explorable through `CuratedAtlasQueryR` and `cellxgenedp`to the CELLxGENE portal.

Our representation, harmonises the metadata at dataset, sample and cell levels, in a unique coherent database table.

Dataset-specific columns (definitions available at cellxgene.cziscience.com) `cell_count`, `collection_id`, `created_at.x`, `created_at.y`, `dataset_deployments`, `dataset_id`, `file_id`, `filename`, `filetype`, `is_primary_data.y`, `is_valid`, `linked_genesets`, `mean_genes_per_cell`, `name`, `published`, `published_at`, `revised_at`, `revision`, `s3_uri`, `schema_version`, `tombstone`, `updated_at.x`, `updated_at.y`, `user_submitted`, `x_normalization`

Sample-specific columns (definitions available at cellxgene.cziscience.com)

sample_, .sample_name, age_days, assay, assay_ontology_term_id, development_stage, development_stage_ontol
ethnicity, ethnicity_ontology_term_id, experiment___, organism, organism_ontology_term_id,
sample_placeholder, sex, sex_ontology_term_id, tissue, tissue_harmonised, tissue_ontology_term_id,
disease, disease_ontology_term_id, is_primary_data.x

Cell-specific columns (definitions available at cellxgene.cziscience.com)

cell_, cell_type, cell_type_ontology_term_idm, cell_type_harmonised, confidence_class,
cell_annotation_azimuth_l2, cell_annotation_blueprint_singler

Through harmonisation and curation we introduced custom column, not present in the original CELLxGENE metadata

- tissue_harmonised: a coarser tissue name for better filtering
- age_days: the number of days corresponding to the age
- cell_type_harmonised: the consensus call identity (for immune cells) using the original and three novel annotations using Seurat Azimuth and SingleR
- confidence_class: an ordinal class of how confident cell_type_harmonised is. 1 is complete consensus, 2 is 3 out of four and so on.
- cell_annotation_azimuth_l2: Azimuth cell annotation
- cell_annotation_blueprint_singler: SingleR cell annotation using Blueprint reference
- cell_annotation_blueprint_monaco: SingleR cell annotation using Monaco reference
- sample_id_db: Sample subdivision for internal use
- file_id_db: File subdivision for internal use
- sample_: Sample ID
- .sample_name: How samples were defined

Possible cache path issues

If your default R cache path includes non-standard characters (e.g. dash because of your user or organisation name), the following error can manifest

Error in db_query_fields.DBIConnection(): ! Can't query fields. Caused by error: ! Parser Error: syntax error at or near "/" LINE 2: FROM /Users/bob/Library/Caches...

The solution is to choose a different cache, for example

get_metadata(cache_directory = path.expand('~'))

Value

A lazy data.frame subclass containing the metadata. You can interact with this object using most standard dplyr functions. For string matching, it is recommended that you use stringr::str_like to filter character columns, as stringr::str_match will not work.

Examples

```
library(dplyr)
filtered_metadata <- get_metadata() |>
    filter(
        ethnicity == "African" &
            assay %LIKE% "%10x%" &
            tissue == "lung parenchyma" &
            cell_type %LIKE% "%CD4%"
    )
```

`get_seurat` *Given a data frame of HCA metadata, returns a Seurat object corresponding to the samples in that data frame*

Description

Given a data frame of HCA metadata, returns a Seurat object corresponding to the samples in that data frame

Usage

```
get_seurat(...)
```

Arguments

`...`	Arguments passed on to `get_single_cell_experiment` `data` A data frame containing, at minimum, a `sample_` column, which corresponds to a single cell sample ID. This can be obtained from the `get_metadata()` function. `assays` A character vector whose elements must be either "counts" and/or "cpm", representing the corresponding assay(s) you want to request. By default only the count assay is downloaded. If you are interested in comparing a limited amount of genes, the "cpm" assay is more appropriate. `repository` A character vector of length one. If provided, it should be an HTTP URL pointing to the location where the single cell data is stored. `cache_directory` An optional character vector of length one. If provided, it should indicate a local file path where any remotely accessed files should be copied. `features` An optional character vector of features (ie genes) to return the counts for. By default counts for all features will be returned.

Value

A Seurat object containing the same data as a call to `get_single_cell_experiment()`

Examples

```
meta <- get_metadata() |> head(2)
seurat <- get_seurat(meta)
```

`get_single_cell_experiment` *Gets a SingleCellExperiment from curated metadata*

Description

Given a data frame of Curated Atlas metadata obtained from `get_metadata()`, returns a `SingleCellExperiment::SingleC` object corresponding to the samples in that data frame

Usage

```
get_single_cell_experiment(
  data,
  assays = "counts",
  cache_directory = get_default_cache_dir(),
  repository = COUNTS_URL,
  features = NULL
)
```

Arguments

`data`	A data frame containing, at minimum, a `sample_` column, which corresponds to a single cell sample ID. This can be obtained from the `get_metadata()` function.
`assays`	A character vector whose elements must be either "counts" and/or "cpm", representing the corresponding assay(s) you want to request. By default only the count assay is downloaded. If you are interested in comparing a limited amount of genes, the "cpm" assay is more appropriate.
`cache_directory`	An optional character vector of length one. If provided, it should indicate a local file path where any remotely accessed files should be copied.
`repository`	A character vector of length one. If provided, it should be an HTTP URL pointing to the location where the single cell data is stored.
`features`	An optional character vector of features (ie genes) to return the counts for. By default counts for all features will be returned.

Value

A SingleCellExperiment object, with one assay for each value in the assays argument

Examples

```
meta <- get_metadata() |> head(2)
sce <- get_single_cell_experiment(meta)
```

`get_SingleCellExperiment` *Gets a SingleCellExperiment from curated metadata*

Description

Given a data frame of Curated Atlas metadata obtained from `get_metadata()`, returns a `SingleCellExperiment::SingleC` object corresponding to the samples in that data frame

Usage

```
get_SingleCellExperiment(...)
```

Arguments

`...`	Arguments passed on to `get_single_cell_experiment`
	`data` A data frame containing, at minimum, a `sample_` column, which corresponds to a single cell sample ID. This can be obtained from the `get_metadata()` function.
	`assays` A character vector whose elements must be either "counts" and/or "cpm", representing the corresponding assay(s) you want to request. By default only the count assay is downloaded. If you are interested in comparing a limited amount of genes, the "cpm" assay is more appropriate.
	`repository` A character vector of length one. If provided, it should be an HTTP URL pointing to the location where the single cell data is stored.
	`cache_directory` An optional character vector of length one. If provided, it should indicate a local file path where any remotely accessed files should be copied.
	`features` An optional character vector of features (ie genes) to return the counts for. By default counts for all features will be returned.

Value

A SingleCellExperiment object, with one assay for each value in the assays argument

Examples

```
meta <- get_metadata() |> head(2)
sce <- get_single_cell_experiment(meta)
```

`get_unharmonised_dataset` *Returns unharmonised metadata for selected datasets.*

Description

Various metadata fields are *not* common between datasets, so it does not make sense for these to live in the main metadata table. This function is a utility that allows easy fetching of this data if necessary.

Usage

```
get_unharmonised_dataset(
  dataset_id,
  cells = NULL,
  conn = dbConnect(drv = duckdb(), read_only = TRUE),
  remote_url = UNHARMONISED_URL,
  cache_directory = get_default_cache_dir()
)
```

Arguments

`dataset_id`	A character vector, where each entry is a dataset ID obtained from the `$file_id` column of the table returned from `get_metadata()`
`cells`	An optional character vector of cell IDs. If provided, only metadata for those cells will be returned.
`conn`	An optional DuckDB connection object. If provided, it will re-use the existing connection instead of opening a new one.
`remote_url`	Optional character vector of length 1. An HTTP URL pointing to the root URL under which all the unharmonised dataset files are located.
`cache_directory`	Optional character vector of length 1. A file path on your local system to a directory (not a file) that will be used to store the unharmonised metadata files.

Value

A named list, where each name is a dataset file ID, and each value is a "lazy data frame", ie a `tbl`.

Examples

```
dataset <- "838ea006-2369-4e2c-b426-b2a744a2b02b"
harmonised_meta <- get_metadata() |>
    dplyr::filter(file_id == dataset) |> dplyr::collect()
unharmonised_meta <- get_unharmonised_dataset(dataset)
unharmonised_tbl <- dplyr::collect(unharmonised_meta[[dataset]])
dplyr::left_join(harmonised_meta, unharmonised_tbl, by=c("file_id", "cell_"))
```

`get_unharmonised_metadata` *Returns unharmonised metadata for a metadata query*

Description

Various metadata fields are *not* common between datasets, so it does not make sense for these to live in the main metadata table. This function is a utility that allows easy fetching of this data if necessary.

Usage

```
get_unharmonised_metadata(metadata, ...)
```

Arguments

`metadata`	A lazy data frame obtained from `get_metadata()`, filtered down to some cells of interest
`...`	Arguments passed on to `get_unharmonised_dataset` `dataset_id` A character vector, where each entry is a dataset ID obtained from the `$file_id` column of the table returned from `get_metadata()` `cells` An optional character vector of cell IDs. If provided, only metadata for those cells will be returned. `conn` An optional DuckDB connection object. If provided, it will re-use the existing connection instead of opening a new one. `remote_url` Optional character vector of length 1. An HTTP URL pointing to the root URL under which all the unharmonised dataset files are located. `cache_directory` Optional character vector of length 1. A file path on your local system to a directory (not a file) that will be used to store the unharmonised metadata files.

Value

A tibble with two columns:

- `file_id`: the same `file_id` as the main metadata table obtained from `get_metadata()`
- `unharmonised`: a nested tibble, with one row per cell in the input `metadata`, containing unharmonised metadata

Examples

```
harmonised <- dplyr::filter(get_metadata(), tissue == "kidney blood vessel")
unharmonised <- get_unharmonised_metadata(harmonised)
```

SAMPLE_DATABASE_URL *URL pointing to the sample metadata file, which is smaller and for test, demonstration, and vignette purposes only*

Description

URL pointing to the sample metadata file, which is smaller and for test, demonstration, and vignette purposes only

Usage

```
SAMPLE_DATABASE_URL
```

Format

An object of class `character` of length 1.

Value

A character scalar consisting of the URL

Examples

```
get_metadata(remote_url = SAMPLE_DATABASE_URL)
```

Index

∗ **datasets**
DATABASE_URL, 2
SAMPLE_DATABASE_URL, 10

DATABASE_URL, 2

get_metadata, 3
get_metadata(), *5–9*
get_seurat, 5
get_single_cell_experiment, *5*, 6, *7*
get_single_cell_experiment(), *3*, *5*
get_SingleCellExperiment, 7
get_unharmonised_dataset, 8, *9*
get_unharmonised_metadata, 9

SAMPLE_DATABASE_URL, 10
SingleCellExperiment::SingleCellExperiment, *3*, *6*, *7*