

Package: cytome (via r-universe)

September 29, 2026

Type Package

Title Native R Reader and Streaming Interface for the Cytome
Single-Cell Store

Version 0.1.3

Description Reads .cytome files - a SQLite-backed single-cell store -
natively in R with zero Python dependency. A .cytome is a plain
SQLite database, so the structural data (cells / genes / peaks,
embeddings, graphs, fragments) is read with DBI; the chunked,
compressed CSR count matrices are decoded via a small Rcpp
layer (lz4-block / zstd / zlib). Constructs a
SingleCellExperiment or a Seurat object, and provides
cytome-native chunk streaming for out-of-core analysis without
loading the full matrix. Part of the PIASO toolkit for
single-cell omics (<<https://piaso.org>>).

License MIT + file LICENSE

URL <https://github.com/genecell/cytome-r>, <https://piaso.org>

BugReports <https://github.com/genecell/cytome-r/issues>

Encoding UTF-8

Depends R (>= 4.1)

Imports DBI, RSQLite, Matrix, Rcpp, methods

LinkingTo Rcpp

Suggests SingleCellExperiment, SummarizedExperiment, DelayedArray,
S4Arrays, SparseArray, S4Vectors, SeuratObject, testthat (>=
3.0.0)

SystemRequirements liblz4, libzstd, zlib

Config/roxygen2/version 8.1.0

Config/pak/sysreqs libzstd-dev zlib1g-dev

Repository <https://genecell.r-universe.dev>

Date/Publication 2026-09-29 01:48:15 UTC

RemoteUrl <https://github.com/genecell/cytome-r>

RemoteRef HEAD
RemoteSha 04563025b39997e34c502aecc5ea0df3977ecbd2

Contents

cytome	2
cytome_close	3
cytome_delayed	3
cytome_embedding	4
cytome_embeddings	5
cytome_fragments	5
cytome_graph	6
cytome_graphs	6
cytome_info	7
cytome_matrices	7
cytome_obs	8
cytome_open	8
cytome_stream	9
cytome_var	10
read_cytome	10
read_cytome_matrix	11
write_cytome	12
write_cytome.Seurat	13
write_cytome.SingleCellExperiment	14
Index	16

cytome	<i>cytome: native R reader for the Cytome single-cell store</i>
--------	---

Description

A ‘cytome’ file is a plain SQLite database. ‘cytome’ opens it read-only with DBI, reads the relational tables (cells / genes / peaks, embeddings, graphs, fragments) directly, and decodes the chunked compressed CSR count matrices via a small Rcpp layer (lz4-block / zstd / zlib) — with zero Python dependency.

cytome_close	<i>Close a cytome handle</i>
--------------	------------------------------

Description

Close a cytome handle

Usage

```
cytome_close(x)
```

Arguments

x	A ‘cytome’ handle from [cytome_open()].
---	---

cytome_delayed	<i>A ‘cytome’ matrix as a DelayedArray</i>
----------------	--

Description

Returns a [DelayedArray::DelayedArray] backed by the file: nothing is read until something asks for values, and block-processing code (‘DelayedMatrixStats’, ‘scuttle’, ‘scran’, ...) then works on a matrix that never has to be held whole.

Usage

```
cytome_delayed(path, matrix_name)
```

Arguments

path	Path to a ‘cytome’ file.
matrix_name	Matrix to wrap, e.g. “RNA_counts”.

Details

Extraction is **chunk-aligned**: only the storage chunks overlapping the requested cells are read, and ‘chunkdim()’ reports the storage geometry so DelayedArray’s block grid lands on chunk boundaries rather than straddling them. Peak memory is therefore set by the block size, not by the size of the matrix.

For data that fits comfortably in memory, ‘delayed = FALSE’ is faster – the reason to use this is data that does not fit, and code you did not write (‘scran’, ‘scuttle’, ‘DelayedMatrixStats’) that must run on it.

Requires the suggested ‘DelayedArray’, ‘S4Arrays’ and ‘SparseArray’ packages.

Value

A ‘DelayedArray’, features x cells.

See Also

[read_cytome()] with ‘delayed = TRUE’, [cytome_stream()]

Examples

```
ref <- system.file("extdata", "reference.cytome", package = "cytome")
if (nzchar(ref) && requireNamespace("DelayedArray", quietly = TRUE)) {
  m <- cytome_delayed(ref, "RNA_counts")
  dim(m)
}
```

cytome_embedding

Read a dense embedding (e.g. X_umap, X_svd) into a matrix

Description

Embeddings are stored as dense, row-major, compressed chunks (‘dense_chunks’).

Usage

```
cytome_embedding(x, name)
```

Arguments

x	A ‘cytome’ handle.
name	Embedding name, e.g. “X_umap” (see [cytome_embeddings()]).

Value

A numeric matrix (cells x n_cols) with cell barcodes as row names when available.

cytome_embeddings	<i>List embeddings stored in a cytome</i>
-------------------	---

Description

List embeddings stored in a cytome

Usage

```
cytome_embeddings(x)
```

Arguments

x A ‘cytome’ handle.

Value

Character vector of embedding/array names (from ‘embedding_meta’).

cytome_fragments	<i>Query ATAC fragments overlapping a genomic interval</i>
------------------	--

Description

Fragments are stored per chromosome in ‘fragments_<chrom>’ tables (columns ‘start, end_, cell_idx, dup_count’) with an SQLite R*-tree index, so interval queries are fast and native — no Python.

Usage

```
cytome_fragments(x, chrom, start = NULL, end = NULL, with_barcode = TRUE)
```

Arguments

x A ‘cytome’ handle.
chrom Chromosome, e.g. “chr1”.
start, end Optional 0-based half-open interval ‘[start, end)’. If both ‘NULL’, returns all fragments on ‘chrom’.
with_barcode If ‘TRUE’ (default), join ‘cell_idx’ to the cell ‘barcode’.

Value

A data.frame: chrom, start, end, cell_idx, [barcode], dup_count.

cytome_graph	<i>Read one graph as a sparse matrix</i>
--------------	--

Description

Read one graph as a sparse matrix

Usage

```
cytome_graph(x, name, n = NULL)
```

Arguments

x	A 'cytome' handle.
name	Graph name, from [cytome_graphs()].
n	Square dimension; defaults to the stored shape, or the cell count for a graph stored one row per edge.

Value

A [Matrix::dgCMatrix-class], cells x cells.

cytome_graphs	<i>Graphs stored in a cytome</i>
---------------	----------------------------------

Description

Lists the cell-by-cell graphs of a store, in either form the Python package writes: compressed row chunks ('graph_meta' / 'graph_chunks', cytome 0.3.6 and later), and one row per edge ('graph_edges', earlier versions and this package's own writer).

Usage

```
cytome_graphs(x)
```

Arguments

x	A 'cytome' handle.
---	--------------------

Value

A character vector of graph names.

cytome_info	<i>High-level summary of a cytome</i>
-------------	---------------------------------------

Description

High-level summary of a cytome

Usage

cytome_info(x)

Arguments

x A ‘cytome’ handle.

cytome_matrices	<i>Matrix catalogue (from ‘matrix_meta’)</i>
-----------------	--

Description

Matrix catalogue (from ‘matrix_meta’)

Usage

cytome_matrices(x)

Arguments

x A ‘cytome’ handle.

Value

A data.frame: matrix_name, n_rows, n_cols, dtype, row_entity, col_entity.

cytome_obs	<i>Read an entity table (cells / genes / peaks / tiles / samples) as a data.frame</i>
------------	---

Description

Read an entity table (cells / genes / peaks / tiles / samples) as a data.frame

Usage

```
cytome_obs(x, entity = "cells")
```

Arguments

x	A ‘cytome’ handle.
entity	Entity table name. Default “cells” (obs). Use “genes”/“peaks” for var.

Value

A data.frame.

cytome_open	<i>Open a .cytome file (read-only)</i>
-------------	--

Description

Open a .cytome file (read-only)

Usage

```
cytome_open(path)
```

Arguments

path	Path to a ‘cytome’ SQLite file.
------	---------------------------------

Value

A ‘cytome’ handle (a list with the open DBI connection). Close with [cytome_close()].

cytome_stream	<i>Stream a cytome matrix chunk-by-chunk (cytome-native, out-of-core)</i>
---------------	---

Description

Iterates the matrix's stored chunks, decoding **one chunk at a time** into a sparse sub-matrix and passing it to 'FUN' — so you can compute over a matrix far larger than RAM without ever materialising the whole thing, directly from the cytome format (no BPCells or other backend required). This is the foundation for R-side streaming single-cell analysis.

Usage

```
cytome_stream(x, name, FUN, features_x_cells = FALSE, ...)
```

Arguments

x	A 'cytome' handle.
name	Matrix name, e.g. "RNA_counts".
FUN	A function called per chunk as 'FUN(mat, row_start, row_end, chunk_idx, ...)', where 'mat' is the chunk's sparse sub-matrix.
features_x_cells	If 'TRUE' each chunk is features x cells; else cells x features (default).
...	Extra arguments passed to 'FUN'.

Value

A list of the per-chunk return values of 'FUN' (reduce as you wish).

Examples

```
## Not run:
x <- cytome_open("data.cytome")
# per-cell total counts without loading the whole matrix:
totals <- cytome_stream(x, "RNA_counts", function(m, s, e, i) Matrix::rowSums(m))
totals <- unlist(totals)

## End(Not run)
```

cytome_var	<i>Read the feature (var) table for a modality's matrix</i>
------------	---

Description

Resolves the matrix's 'col_entity' (e.g. 'RNA_counts' -> 'genes', 'ATAC_counts' -> 'peaks').

Usage

```
cytome_var(x, matrix_name = NULL)
```

Arguments

x	A 'cytome' handle.
matrix_name	Matrix name, e.g. "RNA_counts". If 'NULL', returns the 'genes' table.

Value

A data.frame of features.

read_cytome	<i>Read a '.cytome' file</i>
-------------	------------------------------

Description

One entry point. 'as' chooses what you get back, mirroring 'anndataR::read_h5ad(path, as = ...)':

Usage

```
read_cytome(
  path,
  as = c("SingleCellExperiment", "Seurat", "cytome"),
  main_modality = "RNA",
  alt_modalities = NULL,
  embeddings = TRUE,
  delayed = FALSE,
  ...
)
```

Arguments

path	Path to a ‘cytome’ file.
as	What to return. See above.
main_modality	Modality for the main assay (default “RNA”).
alt_modalities	Modalities to bring across as ‘altExps’ / extra assays. ‘NULL’ (default) = all other ‘_counts’ matrices; ‘character(0)’ = none.
embeddings	If ‘TRUE’ (default) load embeddings.
delayed	If ‘TRUE’, back the main assay with a [DelayedArray::DelayedArray] that reads chunks from the file on demand, so the matrix is never held in memory. ‘SingleCellExperiment’ only, and experimental – see [cytome_delayed()].
...	Passed to the underlying reader.

Details

* “SingleCellExperiment” (default) – main modality as the main assay, other modalities as ‘altExps’, embeddings as ‘reducedDims’. * “Seurat” – main modality as the default assay, other modalities as extra assays, embeddings as ‘DimReduc’s. * “cytome” – the open handle itself, for chunk-wise work with [cytome_stream()]. Close it with [cytome_close()] when done.

Value

A ‘SingleCellExperiment’, a ‘Seurat’ object, or a ‘cytome’ handle.

See Also

[write_cytome()], [cytome_stream()], [cytome_delayed()]

Examples

```
ref <- system.file("extdata", "reference.cytome", package = "cytome")
if (nzchar(ref)) {
  x <- read_cytome(ref, as = "cytome")
  print(x)
  cytome_close(x)
}
```

read_cytome_matrix	<i>Read a cytome count matrix into a sparse Matrix</i>
--------------------	--

Description

Reconstructs the chunked, compressed CSR matrix (stored cells x features) and returns it, by default transposed to **features x cells** (the orientation of an assay in SingleCellExperiment / Seurat).

Usage

```
read_cytome_matrix(x, name, features_x_cells = TRUE)
```

Arguments

x A ‘cytome’ handle.
name Matrix name, e.g. “RNA_counts” (see [cytome_matrices()]).
features_x_cells If ‘TRUE’ (default) return features x cells; else cells x features.

Value

A [Matrix::dgCMatrix-class].

write_cytome	<i>Write an object to a ‘.cytome’ file</i>
--------------	--

Description

Generic. The method is chosen from the class of ‘x’, so the same call works whether you are holding a Seurat object or a SingleCellExperiment – the thing you would otherwise have to look up is the thing R already knows. Third-party packages can add methods for their own classes.

Usage

```
write_cytome(x, path, ...)
```

Arguments

x A ‘Seurat’ or ‘SingleCellExperiment’ object.
path Output ‘.cytome’ path.
... Passed to the method.

Value

‘path’, invisibly. The file is readable by Python ‘cytome.open(path)’.

See Also

[read_cytome()]

Examples

```
## Not run:
write_cytome(seurat_obj, "out.cytome")
write_cytome(sce, "out.cytome")

## End(Not run)
```

write_cytome.Seurat	<i>Write a Seurat object to a '.cytome' file</i>
---------------------	--

Description

Write a Seurat object to a '.cytome' file

Usage

```
## S3 method for class 'Seurat'
write_cytome(
  x,
  path,
  assay = NULL,
  layer = "counts",
  alt_assays = NULL,
  layers = FALSE,
  graphs = TRUE,
  dtype = "float32",
  chunk_size = 2000L,
  compression = "zstd",
  embeddings = TRUE,
  ...
)
```

Arguments

x	A 'Seurat' object.
path	Output '.cytome' path.
assay	Assay for the main modality (default the object's default assay).
layer	Layer/slot to store (default "counts").
alt_assays	Other assays to write as extra modalities. 'NULL' (default) writes all of them; 'character(0)' writes none.
layers	Other layers of the main assay ('data', 'scale.data') to carry. 'FALSE' (default), 'TRUE', or name them. See the 'SingleCellExperiment' method for why this is off by default.
graphs	Write the 'graphs' slot to the shared 'graph_edges' table (default 'TRUE').
dtype, chunk_size, compression, embeddings	As for the 'SingleCellExperiment' method.
...	Unused.

Value

'path', invisibly.

```
write_cytome.SingleCellExperiment
```

Write a SingleCellExperiment to a '.cytome' file

Description

Write a SingleCellExperiment to a '.cytome' file

Usage

```
## S3 method for class 'SingleCellExperiment'
write_cytome(
  x,
  path,
  assay = "counts",
  main_modality = "RNA",
  alt_modalities = NULL,
  layers = FALSE,
  graphs = TRUE,
  dtype = "float32",
  chunk_size = 2000L,
  compression = "zstd",
  embeddings = TRUE,
  ...
)
```

Arguments

<code>x</code>	A [SingleCellExperiment::SingleCellExperiment].
<code>path</code>	Output '.cytome' path.
<code>assay</code>	Assay to store as the main counts matrix (default "counts").
<code>main_modality</code>	Modality name for the main assay (default "RNA").
<code>alt_modalities</code>	'altExp' names to write as extra modalities. 'NULL' (default) writes all of them; 'character(0)' writes none.
<code>layers</code>	Additional assays of the main experiment to carry. 'FALSE' (default) writes none, 'TRUE' writes all, or name them. Off by default because normalized values are recomputable from the counts and cost more space than the counts do; turn it on when the point is to hand somebody your object rather than to archive the measurements.
<code>graphs</code>	Write 'colPairs' to the shared 'graph_edges' table (default 'TRUE'). A neighbour graph is the expensive thing to recompute, and carrying it means the recipient reproduces the same clusters.
<code>dtype</code>	"float32" (default) or "float64".
<code>chunk_size</code>	Rows (cells) per storage chunk.

compression	"zstd" (default), "lz4", or "zlib".
embeddings	Store 'reducedDims' (default 'TRUE').
...	Unused.

Value

'path', invisibly.

Index

cytome, [2](#)
cytome_close, [3](#)
cytome_delayed, [3](#)
cytome_embedding, [4](#)
cytome_embeddings, [5](#)
cytome_fragments, [5](#)
cytome_graph, [6](#)
cytome_graphs, [6](#)
cytome_info, [7](#)
cytome_matrices, [7](#)
cytome_obs, [8](#)
cytome_open, [8](#)
cytome_stream, [9](#)
cytome_var, [10](#)

read_cytome, [10](#)
read_cytome_matrix, [11](#)

write_cytome, [12](#)
write_cytome.Seurat, [13](#)
write_cytome.SingleCellExperiment, [14](#)