Package 'lstar'

June 22, 2026

Title Uniform Data Model and 'Zarr' Interchange for Single-Cell Omics

Version 0.1.0

Description A lightweight interchange layer for single-cell and spatial omics data, built on the L-star model of labelled axes and typed fields over them, serialized to the 'Zarr' format. Provides bidirectional converters (``profiles'') for 'Seurat', 'SingleCellExperiment', 'Conos', and 'pagoda2' objects, including collections of heterogeneous samples, via a shared C++ core ('libstar') so the same store is readable from R, 'Python', and C++.

License MIT + file LICENSE

Encoding UTF-8

URL https://github.com/kharchenkolab/lstar

BugReports https://github.com/kharchenkolab/lstar/issues

LinkingTo cpp11

Imports Matrix, methods, stats, utils

Suggests SeuratObject, Seurat, SingleCellExperiment, SummarizedExperiment, S4Vectors, GenomicRanges, igraph, conos, pagoda2, BPCells, HDF5Array, testthat (>= 3.0.0), knitr, rmarkdown

VignetteBuilder knitr

Additional_repositories https://bnprks.r-universe.dev

SystemRequirements C++17, zlib

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

NeedsCompilation yes

Author Peter Kharchenko [aut, cre]

Maintainer Peter Kharchenko <pk.restricted@gmail.com>

Repository CRAN

Date/Publication 2026-06-22 15:50:07 UTC

Contents

`collection_from` *Assemble a collection of heterogeneous samples from per-sample objects.*

Description

A *collection* keeps each sample's own data — per-sample `cells.<s>`/`genes.<s>` axes and `<field>.<s>` fields, over gene sets that may overlap, differ, or be entirely **disjoint** across samples — alongside a `samples` axis and a *union* `cells` axis carrying the joint analysis (a shared embedding, clusters, a graph). This is lstar's "a collection is not one aligned tensor" model, built from any list of separately-processed samples rather than hand-assembled.

Usage

```
collection_from(
  samples,
  joint = NULL,
  sample_field = "sample",
  prefix_cells = TRUE
)
```

Arguments

samples	a named list of per-sample lstar_datasets (or Seurat / SingleCellExperiment objects, read via the profiles). Each must have cells and genes axes.
joint	optional named list of fields over the **union** cells (the integration outputs): a matrix becomes a joint embedding; a factor/character vector a clustering (inducing a factor axis); a (cells x cells) sparse matrix a graph relation.
sample_field	name of the design label recording each union cell's sample (default "sample").
prefix_cells	prefix each cell label with its sample name so union labels are unique (default TRUE).

Value

an lstar_dataset of kind "collection".

See Also

write_conos(), read_seurat()

Examples

```
mk <- function(s, nc, genes) {
  ds <- list(kind = "sample", axes = list(), fields = list())
  ds$axes$cells <- list(labels = paste0("c", seq_len(nc)),
                        origin = "observed", role = "observation")
  ds$axes$genes <- list(labels = genes, origin = "observed", role = "feature")
 m <- as(Matrix::Matrix(matrix(rpois(nc * length(genes), 2), nc), sparse = TRUE), "CsparseMatrix")
 ds$fields$counts <- list(role = "measure", span = c("cells", "genes"), state = "raw", values = m)
  class(ds) <- "lstar_dataset"; ds
}
col <- collection_from(list(A = mk("A", 5, paste0("g", 1:8)),
                            B = mk("B", 7, paste0("g", 3:12))))   # divergent gene sets
col$kind
```

---

col_sum_by_group *Per-(group, gene) sufficient stats over a CSC measure (the shared libstar kernel).*

Description

For a cells x genes sparse matrix and a per-cell group assignment, returns each group's sum, sum-of-squares, and number of expressing cells per gene, computed over log1p(m) (or raw). This is the reduction cluster stats and marker tables are built from; the same C++ core backs the WASM and Python bindings.

Usage

```
col_sum_by_group(m, code, ngroups, lognorm = TRUE)
```

Arguments

m	a cells x genes sparse matrix (coerced to CSC).
code	length-nrow(m) integer group assignment in [0, ngroups) (0-based).
ngroups	number of groups.
lognorm	compute over log1p(m) (default TRUE).

Value

a list with sum, sumsq, n_expr, each a flat row-major (group, gene) numeric vector of length ngroups * ncol(m).

---

field_value *Accessor: a field's value by name.*

Description

Accessor: a field's value by name.

Usage

```
field_value(ds, name)
```

Arguments

ds	an lstar_dataset
name	field name

Value

the field's values (a vector, matrix, or sparse Matrix), or NULL if absent.

---

lstar_markers *Tidy marker table for a factor's DE bundle.*

Description

Gathers the de.<factor>.<stat> fields (score/lfc/pval/padj over the factor x genes axes) into a long-form data frame: one row per (group, gene) with whichever statistics are present.

Usage

```
lstar_markers(ds, factor, top = NULL, sort_by = "score", descending = TRUE)
```

Arguments

ds	an lstar_dataset (e.g. from lstar_read()).
factor	the factor-axis name the DE was computed over (e.g. "leiden").
top	optional: keep only the top-N genes per group (by sort_by).
sort_by	statistic to rank within a group (default "score").
descending	sort descending (default TRUE).

Value

a data frame with columns group, gene, and the available statistics.

---

lstar_read *Read an L* Zarr store into an R dataset.*

Description

Read an L* Zarr store into an R dataset.

Usage

```
lstar_read(path)
```

Arguments

path	path to a *.lstar.zarr store

Value

an lstar_dataset: a list with axes and fields, each field's values assembled as a base vector, matrix, or Matrix sparse matrix.

Examples

```
p <- tempfile(fileext = ".lstar.zarr")
ds <- list(kind = "sample", axes = list(), fields = list())
ds$axes$cells <- list(labels = paste0("c", 1:3), origin = "observed", role = "observation")
ds$fields$depth <- list(role = "measure", span = "cells", values = c(1, 2, 3))
class(ds) <- "lstar_dataset"
lstar_write(ds, p)
ds2 <- lstar_read(p)
field_value(ds2, "depth")
```

`lstar_read_block` *Read a contiguous gene (column) range of a CSC measure from an L* store, bounded-memory.*

Description

Reads genes `[g_lo, g_hi)` (0-based, half-open) of a CSC measure as a `dgCMatrix` (cells x genes), decoding only the store chunks that overlap the range. The general block-read primitive a consumer drives to build out-of-core reductions over an L* store without implementing them in lstar.

Usage

```
lstar_read_block(path, field, g_lo, g_hi, cell_names = NULL, gene_names = NULL)
```

Arguments

`path`	path to a `*.lstar.zarr` store.
`field`	name of a `(cells, genes)` CSC measure field in the store.
`g_lo, g_hi`	0-based, half-open gene (column) range `[g_lo, g_hi)` to read.
`cell_names, gene_names`	optional dimnames for the returned matrix (default `NULL`).

Value

a `dgCMatrix` (cells x genes) holding the requested gene columns.

`lstar_read_genes` *Read an arbitrary set of gene columns of a CSC measure, returning cells x genes.*

Description

Gathers the requested gene columns from a chunked CSC store, decoding each touched chunk **at most once** (an ascending sweep over sorted-unique columns), then restores the caller's order. Efficient for scattered subsets (e.g. overdispersed genes for PCA) – unlike a per-column read it does not re-decode a chunk once per gene it contains.

Usage

```
lstar_read_genes(path, field, genes, all_genes, cell_names = NULL)
```

Arguments

path	path to a *.lstar.zarr store.
field	name of a (cells, genes) CSC measure field in the store.
genes	the gene columns to gather, as names (matched against all_genes) or 1-based indices.
all_genes	the field's full gene-label vector, used to resolve genes to column positions.
cell_names	optional row names (cells) for the returned matrix (default NULL).

Value

a dgCMatrix (cells x length(genes)) in the caller's requested gene order.

---

lstar_stream_col_sum_by_group

Per-(group, gene) sums of a CSC measure in a store, streamed and bounded-memory.

Description

Streaming, fused pseudobulk: per cell-group sums of each gene, computed in one threaded pass over a chunked CSC store, with the same optional depth-normalized log1p view as stream_col_stats() (the counterpart of pagoda2's colSumByFacView). group is a per-cell integer bucket in [0, ngroups) in store row order (out-of-range cells are skipped; pass NA cells as 0 for an explicit <NA> row).

Usage

```
lstar_stream_col_sum_by_group(
  path,
  field,
  group,
  ngroups,
  lognorm = FALSE,
  depth = NULL,
  depthScale = 1,
  block = 4096L,
  n_threads = 1L
)
```

Arguments

path	path to a *.lstar.zarr store.
field	name of a (cells, genes) CSC measure field in the store.
group	integer per-cell group bucket in [0, ngroups), in store row order.
ngroups	number of groups (the result has one row per group).

lognorm	compute over the log1p view of the measure (default FALSE).
depth	optional per-cell depth vector (length = n cells, store row order) for the normalized view.
depthScale	depth scaling factor used with depth (default 1).
block	streaming block size in cells per pass (default 4096).
n_threads	threads per block reduction: 1 = serial (default), N = N threads, 0 = all cores.

Value

a ngroups x ngenes numeric matrix (row g = the per-gene sums for group g).

`lstar_write` *Write an R dataset to an L* Zarr store.*

Description

Write an R dataset to an L* Zarr store.

Usage

```
lstar_write(
  ds,
  path,
  chunk_elems = NULL,
  compression = c("none", "gzip", "zlib"),
  level = 5L
)
```

Arguments

ds	an lstar_dataset (as returned by lstar_read() or a profile reader)
path	output store path (a *.lstar.zarr directory)
chunk_elems	if non-NULL, chunk each array along its first axis so each chunk holds about this many elements (e.g. 1e6). This is what lets a reader stream/block-read only the touched chunks (e.g. lstar_read_block(), stream_col_stats()); the default (NULL) writes each array as a single chunk – the portable, byte-identical-to-before default.
compression	chunk codec: "none" (default), "gzip", or "zlib" (numcodecs-compatible; readable by the C++ core and zarr-python).
level	compression level 1-9 (default 5), used when compression is "gzip"/"zlib".

Value

the output path, invisibly.

See Also

lstar_read(), lstar_read_block()

print.lstar_dataset *Print an L* dataset*

Description

Print an L* dataset

Usage

```
## S3 method for class 'lstar_dataset'
print(x, ...)
```

Arguments

x	an lstar_dataset
...	ignored, for S3 compatibility

Value

x, invisibly (called for the side effect of printing a summary of axes and fields).

read_conos *Reconstruct a Conos object from an L* collection*

Description

The inverse of write_conos(): rebuilds the per-sample Pagoda2 objects (raw counts plus the stored PCA reduction) and restores the joint graph, embedding and clustering(s), returning a live conos::Conos object ready for plotting, marker detection and label transfer. Re-running runGraph() will recompute the per-sample variance model (not stored).

Usage

```
read_conos(ds)
```

Arguments

ds	an lstar_dataset of kind "collection" (e.g. from lstar_read()), or a path to an lstar store holding one.

Value

a conos::Conos object.

read_sce — *Read a SingleCellExperiment into an L* dataset.*

Description

Read a SingleCellExperiment into an L* dataset.

Usage

```
read_sce(sce)
```

Arguments

sce	a SingleCellExperiment

Value

an lstar_dataset of kind "sample".

See Also

write_sce()

read_sce_backed — *Open an .h5ad's expression matrix as a disk-backed SingleCellExperiment assay (via HDF5Array).*

Description

Reads the matrix from h5ad as an HDF5Array::H5ADMatrix – a DelayedMatrix that stays on disk (genes x cells, the Bioconductor convention) – and wraps it in a SingleCellExperiment. The matrix is never materialized. Pairs with Python lstar.convert_to_h5ad(store, h5ad).

Usage

```
read_sce_backed(h5ad, layer = NULL, assay_name = "counts")
```

Arguments

h5ad	path to an .h5ad file.
layer	h5ad layer to read; NULL (default) reads X.
assay_name	name for the SCE assay (default inferred: "counts").

Value

a SingleCellExperiment whose assay is a disk-backed DelayedMatrix.

read_seurat — *Read a Seurat object into an L* dataset.*

Description

Handles Seurat v3/v4 (`Assay`) and v5 (`Assay5`); a v5 assay split by sample (`split(assay, f = ...)`) is read as an L* collection. The detected versions are recorded in `ds$profiles`.

Usage

```
read_seurat(so, assay = SeuratObject::DefaultAssay(so))
```

Arguments

`so`	a `Seurat` object
`assay`	assay to read (default: the default assay)

Value

an `lstar_dataset` (of kind `"sample"`, or `"collection"` for a split assay).

See Also

`write_seurat()`

read_seurat_backed — *Open an .h5ad's expression matrix as a disk-backed Seurat v5 assay (via BPCells).*

Description

Reads the matrix from `h5ad` with BPCells (`open_matrix_anndata_hdf5`) so it stays on disk as a streaming `IterableMatrix` – already oriented genes x cells (rownames genes, colnames cells), the Seurat convention – and wraps it in a Seurat v5 `Assay5`. The matrix is never materialized: peak memory is a few megabytes regardless of atlas size. Typical use is the bounded end of an L* conversion, after `lstar.convert_to_h5ad(store, h5ad)` in Python:

Usage

```
read_seurat_backed(h5ad, group = "X", assay = "RNA", project = "lstar")
```

Arguments

`h5ad`	path to an .h5ad file (its `X`/layer stays on disk).
`group`	h5ad group to read as the matrix (default `"X"`; e.g. `"layers/counts"`).
`assay`	name for the Seurat assay (default `"RNA"`).
`project`	Seurat project name.

Details

```
so <- read_seurat_backed("atlas.h5ad")        # counts live on disk (BPCells)
so <- Seurat::NormalizeData(so)               # Seurat v5 ops stream off disk
```

Value

a Seurat object whose assay counts are a disk-backed BPCells matrix.

`stream_col_stats` *Per-gene mean/variance of a CSC measure in a store, read with bounded memory.*

Description

Computes the zero-aware per-gene mean, variance, and number of expressing cells of a `cells x genes` measure directly from an L* store, reading it **block-by-block** so the whole matrix never lands in memory – the C++/R counterpart of Python's `stream_col_stats`. Bounded memory requires a chunked store (one written with `chunk_elems` set, e.g. by a streamed conversion); on an unchunked store it still works but reads the `data` array whole. The field must be CSC (gene-major); use it for HVG selection / variance modeling over an atlas too large to load.

Usage

```
stream_col_stats(
  path,
  field,
  block = 4096L,
  n_threads = 1L,
  lognorm = FALSE,
  depth = NULL,
  depthScale = 1,
  population = FALSE
)
```

Arguments

`path`	path to an L* store (`.lstar.zarr`).
`field`	measure field name (e.g. `"counts"`), stored CSC.
`block`	number of gene columns per streamed block (default 4096).
`n_threads`	threading policy for each block's reduction: 1 = serial (default), N = N threads, `<=0` = all cores. Results are thread-count invariant.
`lognorm`	reduce over `log1p(x)` per nonzero, on the fly, without building the normalized matrix (default `FALSE`).

depth	optional per-cell depth vector (length n cells, in store row order). When given, each nonzero is normalized to `x * depthScale / depth[cell]` before the optional `log1p` – i.e. pagoda2's "plain" analysis view, computed in one streamed pass (the block's row `indices` are read too). `NULL` (default) reduces the raw/log1p values.
depthScale	depth scaling factor (default 1) used with `depth`.
population	if `TRUE`, variance divides by `n` (population, pagoda2's convention) instead of the sample `n-1` (default `FALSE`).

Value

a list with `mean`, `var` (length `ngenes` numeric) and `nnz` (length `ngenes` integer).

---

write_conos *Build an L* dataset from a Conos object (a collection of samples).*

Description

Build an L* dataset from a Conos object (a collection of samples).

Usage

```
write_conos(co, clustering = NULL)
```

Arguments

co	a `Conos` R6 object
clustering	optional name of a joint clustering in `co$clusters` (default: all)

Value

an `lstar_dataset` of kind `collection`

---

write_pagoda2 *Export a Pagoda2 object to an L* (`*.lstar.zarr`) store.*

Description

Writes counts (raw, cell x gene), the embedding, cluster/cell-type/QC labels, and the viewer profile's cluster sufficient stats + marker tables (`viewer@0.1`). Computes the cluster stats with the shared libstar kernel.

Usage

```
write_pagoda2(p2, path = NULL, grouping = "leiden")
```

Arguments

p2	a Pagoda2 (pagoda2.1) object.
path	output store path (*.lstar.zarr); if NULL, only the lstar_dataset is returned.
grouping	a cellMeta column to use as the primary clustering (default "leiden").

Value

an lstar_dataset (invisibly if written).

---

write_sce *Build a SingleCellExperiment from an L* dataset.*

Description

Measures become assays (transposed to genes x cells), embeddings become reducedDims, and arity-1 cell fields become colData.

Usage

```
write_sce(ds)
```

Arguments

ds	an lstar_dataset

Value

a SingleCellExperiment object.

See Also

read_sce()

---

write_seurat *Build a Seurat object from an L* dataset.*

Description

Measures over (cells, genes) become assay layers (transposed to Seurat's genes x cells orientation); embeddings become DimReducs; arity-1 cell fields become meta.data.

Usage

```
write_seurat(ds)
```

Arguments

ds	an lstar_dataset

Value

a Seurat object.

See Also

read_seurat()

Index