

Package: cudacellr (via r-universe)

October 27, 2026

Title GPU-Aware Single-Cell Analysis Workflows

Version 0.4.0

Description Provides single-cell analysis primitives for sparse count matrices, including library-size normalization, log-normalization, highly variable feature selection, principal component analysis, and nearest-neighbour construction. Integrates the 'cudaverse' numerical and sparse layers while preserving matrix identifiers and recording the computation backend for each stage. Optional adapters preserve native 'SingleCellExperiment' and 'SeuratObject' version 5 object structures.

License MIT + file LICENSE

URL <https://cudaverse.github.io/cudacellr/>,
<https://github.com/cudaverse/cudacellr>

BugReports <https://github.com/cudaverse/cudacellr/issues>

Encoding UTF-8

Imports cudaverse (>= 0.4.1), Matrix, methods

Suggests DelayedArray, knitr, rmarkdown, S4Vectors, SeuratObject (>= 5.0.0), SingleCellExperiment, SummarizedExperiment, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Yaoxiang Li [aut, cre]

Maintainer Yaoxiang Li <liyaoxiang@outlook.com>

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

Date/Publication 2026-10-27 18:48:44 UTC

RemoteUrl <https://github.com/cran/cudacellr>

RemoteRef HEAD

RemoteSha 4c94e8e4aaa41ecce2e2b3d3384e6e8a58d246f7

Contents

cuda_cell_neighbors	2
cuda_cell_pca	3
cuda_hvg	4
cuda_normalize_counts	4
cuda_provenance-methods	5
cudacell_sce	6
cudacell_seurat	8
cudacell_workflow	10
Index	12

cuda_cell_neighbors	<i>Construct a nearest-neighbour representation</i>
---------------------	---

Description

Construct a nearest-neighbour representation

Usage

```
cuda_cell_neighbors(  
  embedding,  
  k = 15L,  
  metric = c("euclidean", "cosine"),  
  device = c("auto", "cuda", "cpu"),  
  batch_size = 256L  
)
```

Arguments

embedding	Cell-by-component matrix or cuda_pca result.
k	Number of neighbours.
metric	Distance metric.
device	Computation device.
batch_size	Maximum query rows per exact kNN distance block. Smaller values reduce peak distance memory.

Value

A cuda_knn object retaining cell names from the embedding.

Examples

```

cuda_cell_neighbors(
  matrix(rnorm(60), 20, 3),
  k = 5,
  batch_size = 4,
  device = "cpu"
)

```

 cuda_cell_pca

Run PCA on highly variable single-cell features

Description

Run PCA on highly variable single-cell features

Usage

```

cuda_cell_pca(
  counts,
  n_components = 30L,
  n_hvg = 2000L,
  scale. = TRUE,
  device = c("auto", "cuda", "cpu"),
  scale_factor = 10000,
  log1p = TRUE,
  min_mean = 0
)

```

Arguments

counts	Feature-by-cell matrix.
n_components	Number of principal components.
n_hvg	Number of highly variable features.
scale.	Whether to scale selected features before PCA.
device	Device passed to <code>cudaverse::cuda_pca()</code> .
scale_factor	Target library size used during normalization.
log1p	Whether to apply <code>log1p()</code> after library-size normalization.
min_mean	Minimum feature mean used during highly variable feature selection.

Value

A `cuda_pca` object with an additional `features` element. Cell names label PCA score rows and selected feature names label loadings.

Examples

```
set.seed(2)
counts <- matrix(rpois(30 * 20, lambda = 2), 30, 20)
cuda_cell_pca(counts, n_components = 3, n_hvg = 10, device = "cpu")
```

cuda_hvg	Select highly variable features
----------	---------------------------------

Description

Select highly variable features

Usage

```
cuda_hvg(counts, n_top = 2000L, min_mean = 0)
```

Arguments

counts	Feature-by-cell count or normalized expression matrix.
n_top	Number of features to select.
min_mean	Minimum feature mean.

Value

A data frame ordered by dispersion with feature statistics and a selected indicator. The index column retains the original feature position even when feature names are duplicated.

Examples

```
set.seed(1)
counts <- Matrix::rsparsematrix(20, 10, density = 0.3)
counts@x <- abs(round(counts@x * 5))
cuda_hvg(counts, n_top = 5)
```

cuda_normalize_counts	Normalize a single-cell count matrix
-----------------------	--------------------------------------

Description

Counts are expected with features in rows and cells in columns.

Usage

```
cuda_normalize_counts(counts, scale_factor = 10000, log1p = TRUE)
```

Arguments

counts	A base matrix, Matrix sparse matrix, or cudaspase.
scale_factor	Target library size.
log1p	Whether to apply log1p() to non-zero normalized values.

Value

A sparse Matrix::dgCMatrix with the input feature and cell names.

Examples

```
counts <- Matrix::Matrix(matrix(c(1, 0, 3, 2, 4, 1), 3), sparse = TRUE)
cuda_normalize_counts(counts)
```

cuda_provenance-methods

Inspect provenance stored on native single-cell containers

Description

These methods expose the shared `cudaverse::cuda_provenance()` contract for SingleCellExperiment and Seurat objects produced by cudacellr.

Usage

```
## S3 method for class 'SingleCellExperiment'
cuda_provenance(x)
```

```
## S3 method for class 'Seurat'
cuda_provenance(x)
```

Arguments

x	A supported native single-cell container.
---	---

Value

A cuda_provenance data frame.

cudacell_sce

Run a cudacell workflow on a SingleCellExperiment

Description

cudacell_sce() reads one assay, runs [cudacell_workflow\(\)](#), and returns a modified copy of the input object. Existing assays, row and column metadata, reduced dimensions, alternative experiments, pairings, and top-level metadata are retained. New results are stored in native SingleCellExperiment locations:

Usage

```
cudacell_sce(
  x,
  assay = "counts",
  n_hvg = 2000L,
  n_components = 30L,
  k = 15L,
  device = c("auto", "cuda", "cpu"),
  batch_size = 256L,
  scale_factor = 10000,
  log1p = TRUE,
  min_mean = 0,
  scale. = TRUE,
  normalized_assay = NULL,
  reduced_dim = "CUDACELL_PCA",
  col_pair = "CUDACELL_KNN",
  row_data_prefix = "cudacell",
  set_size_factors = FALSE,
  overwrite = FALSE,
  realize = FALSE
)
```

Arguments

x	A SingleCellExperiment.
assay	Exact name of the feature-by-cell count assay.
n_hvg	Number of highly variable features.
n_components	Number of PCA components.
k	Number of neighbours per cell.
device	Computation device.
batch_size	Maximum query rows per exact kNN distance block.
scale_factor	Target library size used during normalization.
log1p	Whether to apply log1p() after library-size normalization.

<code>min_mean</code>	Minimum feature mean used for HVG selection.
<code>scale.</code>	Whether to scale selected features before PCA.
<code>normalized_assay</code>	Name for the added normalized assay. NULL chooses "cudacell_logcounts" when <code>log1p = TRUE</code> and "cudacell_normalized" otherwise.
<code>reduced_dim</code>	Name for the added PCA reduced dimension.
<code>col_pair</code>	Name for the added directed kNN colPair.
<code>row_data_prefix</code>	Prefix for added feature statistics and the namespaced column size factor.
<code>set_size_factors</code>	Whether to also replace the canonical <code>SingleCellExperiment::sizeFactors()</code> values. The namespaced <code><row_data_prefix>_size_factor</code> column is always added to <code>colData</code> .
<code>overwrite</code>	Whether explicitly named cudacellr output fields may be replaced. Other object content is never overwritten.
<code>realize</code>	Whether a delayed assay may be explicitly materialized as a sparse Matrix.

Details

- normalized expression in an assay;
- PCA scores in a reduced dimension;
- highly variable feature statistics in `rowData`;
- directed nearest-neighbour relationships in a `colPair`;
- parameters and compute provenance in `metadata(x)$cudacellr`.

The selected assay must currently be a base matrix or Matrix object. Delayed assays are never realized silently. Set `realize = TRUE` only after confirming that the selected assay fits in memory.

Value

A valid `SingleCellExperiment` of the same class as `x`.

Examples

```
if (requireNamespace("SingleCellExperiment", quietly = TRUE)) {
  counts <- matrix(
    rpois(30 * 12, lambda = 2),
    nrow = 30,
    dimnames = list(paste0("gene_", 1:30), paste0("cell_", 1:12))
  )
  sce <- SingleCellExperiment::SingleCellExperiment(
    assays = list(counts = counts)
  )
  sce <- cudacell_sce(
    sce,
    n_hvg = 10,
    n_components = 3,
    k = 3,
```

```

    device = "cpu"
  )
  SingleCellExperiment::reducedDim(sce, "CUDACELL_PCA")
  cuda_provenance(sce)
}

```

cudacell_seurat

Run a cudacell workflow on a Seurat object

Description

cudacell_seurat() reads one exact assay layer, runs [cudacell_workflow\(\)](#), and returns a modified copy of the input object. The full Seurat package is not required; the object contract uses SeuratObject version 5 or newer.

Usage

```

cudacell_seurat(
  x,
  assay = NULL,
  layer = "counts",
  n_hvg = 2000L,
  n_components = 30L,
  k = 15L,
  device = c("auto", "cuda", "cpu"),
  batch_size = 256L,
  scale_factor = 10000,
  log1p = TRUE,
  min_mean = 0,
  scale. = TRUE,
  output_assay = "CUDACELL",
  reduction = "cudacell_pca",
  reduction_key = "CUDACELLPC_",
  neighbor = "cudacell_knn",
  metadata_prefix = "cudacell",
  overwrite = FALSE,
  realize = FALSE
)

```

Arguments

x	A Seurat object.
assay	Exact input assay name. NULL uses SeuratObject::DefaultAssay() .
layer	Exact feature-by-cell count layer name.
n_hvg	Number of highly variable features.
n_components	Number of PCA components.

<code>k</code>	Number of neighbours per cell.
<code>device</code>	Computation device.
<code>batch_size</code>	Maximum query rows per exact kNN distance block.
<code>scale_factor</code>	Target library size used during normalization.
<code>log1p</code>	Whether to apply <code>log1p()</code> after library-size normalization.
<code>min_mean</code>	Minimum feature mean used for HVG selection.
<code>scale.</code>	Whether to scale selected features before PCA.
<code>output_assay</code>	Name for the added normalized Assay5. Its data layer contains normalized expression.
<code>reduction</code>	Name for the added PCA DimReduc.
<code>reduction_key</code>	Seurat key for PCA dimensions. It must start with a letter, contain only letters or digits, and end in an underscore.
<code>neighbor</code>	Name for the added exact kNN Neighbor.
<code>metadata_prefix</code>	Prefix for feature statistics and the cell-level size-factor column.
<code>overwrite</code>	Whether the named cudaCellr output fields may be replaced. Other object state is never overwritten.
<code>realize</code>	Whether a non-memory-backed layer may be explicitly materialized as a sparse Matrix.

Details

Results are stored in native, namespaced locations:

- normalized expression and feature statistics in a new Assay5;
- PCA embeddings and loadings in a DimReduc;
- exact kNN indices and distances in a Neighbor;
- library-size factors in cell metadata;
- parameters and compute provenance in the `cudaCell_seurat` tool record.

Existing assays and layers, reductions, graphs, neighbours, identities, cell metadata, images, commands, miscellaneous data, and tool records are retained. Every output collision is checked before device selection or computation. `overwrite = TRUE` replaces only the explicitly named cudaCellr outputs.

Feature metadata use the requested prefix. `<prefix>_hvg_rank` is the one-based global dispersion rank across all input features; `<prefix>_hvg` identifies the selected top `n_hvg` features.

Non-memory-backed layers are never realized silently. Set `realize = TRUE` only after confirming that the selected layer fits in memory. The selected layer must contain every object cell in the same order.

Value

A valid Seurat object. The input object is not modified.

Examples

```

if (requireNamespace("SeuratObject", quietly = TRUE)) {
  counts <- Matrix::Matrix(
    matrix(
      rpois(30 * 12, lambda = 2),
      nrow = 30,
      dimnames = list(
        paste0("gene_", 1:30),
        paste0("cell_", 1:12)
      )
    ),
    sparse = TRUE
  )
  object <- SeuratObject::CreateSeuratObject(counts)
  object <- cudacell_seurat(
    object,
    n_hvg = 10,
    n_components = 3,
    k = 3,
    device = "cpu"
  )
  SeuratObject::Embeddings(object[["cudacell_pca"]])
  cuda_provenance(object)
}

```

cudacell_workflow

Run the initial cudacellr workflow

Description

Run the initial cudacellr workflow

Usage

```

cudacell_workflow(
  counts,
  n_hvg = 2000L,
  n_components = 30L,
  k = 15L,
  device = c("auto", "cuda", "cpu"),
  batch_size = 256L,
  scale_factor = 10000,
  log1p = TRUE,
  min_mean = 0,
  scale. = TRUE
)

```

Arguments

<code>counts</code>	Feature-by-cell count matrix.
<code>n_hvg</code>	Number of variable features.
<code>n_components</code>	PCA components.
<code>k</code>	Neighbours per cell.
<code>device</code>	Computation device.
<code>batch_size</code>	Maximum query rows per exact kNN distance block.
<code>scale_factor</code>	Target library size used during normalization.
<code>log1p</code>	Whether to apply <code>log1p()</code> after library-size normalization.
<code>min_mean</code>	Minimum feature mean used during highly variable feature selection.
<code>scale.</code>	Whether to scale selected features before PCA.

Value

A list containing normalized counts, variable features, PCA, and kNN. Feature and cell identifiers are retained throughout all stages.

Examples

```
set.seed(3)
counts <- matrix(rpois(40 * 25, 2), 40, 25)
cudaCell_workflow(
  counts, n_hvg = 15, n_components = 5, k = 5, device = "cpu"
)
```

Index

`cuda_cell_neighbors`, [2](#)
`cuda_cell_pca`, [3](#)
`cuda_hvg`, [4](#)
`cuda_normalize_counts`, [4](#)
`cuda_provenance-methods`, [5](#)
`cuda_provenance.Seurat`
 (`cuda_provenance-methods`), [5](#)
`cuda_provenance.SingleCellExperiment`
 (`cuda_provenance-methods`), [5](#)
`cuda_cell_sce`, [6](#)
`cuda_cell_seurat`, [8](#)
`cuda_cell_workflow`, [10](#)
`cuda_cell_workflow()`, [6](#), [8](#)
`cudaerse::cuda_pca()`, [3](#)
`cudaerse::cuda_provenance()`, [5](#)

`SeuratObject::DefaultAssay()`, [8](#)
`SingleCellExperiment::sizeFactors()`, [7](#)