Package: gedi2 (via r-universe)

May 19, 2026

Type Package

Title Gene Expression Decomposition and Integration

Version 2.3.4

Date 2026-05-09

Description A memory-efficient implementation for integrating gene expression data from single-cell RNA sequencing experiments. Uses a C++ backend with thin R wrappers to enable analysis of large-scale single-cell datasets. The package supports multiple data modalities including count matrices, paired data (splicing, RNA velocity, CITE-seq), and binary indicators. It implements a latent variable model with block coordinate descent optimization for dimensionality reduction and batch effect correction. Core algorithms are described in Madrigal et al. (2024) <doi:10.1038/s41467-024-50963-0>.

License MIT + file LICENSE

URL https://github.com/csglab/gedi2

BugReports https://github.com/csglab/gedi2/issues

Depends R (>= 4.0.0)

Imports Rcpp (>= 1.0.0), R6 (>= 2.5.0), Matrix (>= 1.3.0), ggplot2, scales, methods, stats, utils

LinkingTo Rcpp, RcppEigen

Suggests hdf5r, uwot, digest, glmnet, Seurat, SeuratObject, SingleCellExperiment, testthat (>= 3.0.0)

Config/testthat/edition 3

SystemRequirements GNU make

Encoding UTF-8

RoxygenNote 7.3.3

NeedsCompilation yes

Author Arsham Mikaeili Namini [aut, cre], Hamed S.Najafabadi [aut]

Maintainer Arsham Mikaeili Namini

<arsham.mikaeilinamini@mail.mcgill.ca>

Config/pak/sysreqs make

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

Date/Publication 2026-05-19 08:41:47 UTC

RemoteUrl https://github.com/cran/gedi2

RemoteRef HEAD

RemoteSha b9c69389c9da780d083e5ec2cae576b5a07c5283

Contents

`check_optional_dependencies` *Check GEDI Optional Dependencies*

Description

Check which optional dependencies are installed and display their status.

Usage

```
check_optional_dependencies()
```

Value

Named logical vector indicating which optional packages are installed

Examples

```
check_optional_dependencies()
```

`CreateGEDIObject` *Create GEDI Object*

Description

Creates and configures a GEDI (Gene Expression Data Integration) model object. This implementation uses Option 2 memory optimization: C++ computes Yi from M, eliminating duplicate Yi storage in R.

Usage

```
CreateGEDIObject(
  Samples,
  M = NULL,
  Y = NULL,
  X = NULL,
  colData = NULL,
  C = NULL,
  H = NULL,
  K = 40,
  mode = "Bsphere",
  adjustD = TRUE,
  orthoZ = TRUE,
  Z_shrinkage = 1,
  A_shrinkage = 1,
  Qi_shrinkage = 1,
  Rk_shrinkage = 1,
  oi_shrinkage = 1,
  o_shrinkage = 1,
  si_shrinkage = 1,
  fixed_si = NA,
  rsvd_p = 10,
  rsvd_sdist = "normal",
  verbose = 1,
  num_threads = 1
)
```

Arguments

`Samples`	Factor or character vector indicating sample of origin for each cell
`M`	Raw count matrix (sparse or dense), or list of two matrices for paired data. C++ will compute Yi = log(M+1) internally - no Yi copy stored in R!

Y	Log-transformed expression matrix (optional if M provided)
X	Binary indicator matrix (optional if M or Y provided)
colData	Optional data.frame with cell metadata
C	Gene-level prior matrix (genes x pathways)
H	Sample-level covariate matrix (covariates x samples)
K	Number of latent factors (default: 10)
mode	Normalization mode: "Bl2" or "Bsphere" (default: "Bl2")
adjustD	Whether to adjust D based on B row norms (default: TRUE)
orthoZ	Whether Z columns should be orthogonal (default: TRUE)
Z_shrinkage, A_shrinkage, Qi_shrinkage, Rk_shrinkage, oi_shrinkage, o_shrinkage, si_shrinkage	Regularization strengths (default: 1)
fixed_si	Fix cell library sizes at this value, or NA to optimize (default: NA)
rsvd_p	Oversampling parameter for randomized SVD (default: 10)
rsvd_sdist	Random distribution for rSVD: "normal", "unif", or "rademacher" (default: "normal")
verbose	Verbosity level: 0 (silent), 1 (info), 2 (debug) (default: 1)
num_threads	Number of OpenMP threads (default: 0 = auto)

Value

GEDI R6 object with memory-efficient architecture

Examples

```
# Load example data
pbmc_small <- SeuratObject::pbmc_small

# Basic usage - memory efficient!
model <- CreateGEDIObject(
  Samples = pbmc_small@meta.data$orig.ident,
  M = pbmc_small@assays$RNA@counts, # Only M stored in R; Yi computed in C++
  K = 10,
  num_threads = 1
)

# Train the model
model$train(iterations = 50)

# Access results via active bindings
Z <- model$Z
params <- model$params

# Access projections (computed on demand, cached)
zdb <- model$projections$ZDB
db <- model$projections$DB
```

```
# Access imputed expression
Y_imputed <- model$imputed$Y() # No M needed!
Y_var <- model$imputed$variance(pbmc_small@assays$RNA@counts) # M required
```

---

gedi_to_seurat *Convert GEDI Model to Seurat Object*

Description

Creates a Seurat object from a trained GEDI model, including imputed data, projections, and embeddings.

Usage

```
gedi_to_seurat(
  model,
  M = NULL,
  project = "GEDI",
  assay = "RNA",
  use_imputed = TRUE,
  add_projections = TRUE,
  add_embeddings = TRUE,
  min_cells = 0,
  min_features = 0,
  verbose = TRUE
)
```

Arguments

model	GEDI model object (trained)
M	Original count matrix (optional). If not provided, will use back-transformed imputed values as approximate counts.
project	Character, project name for Seurat object (default: "GEDI")
assay	Character, name for the main assay (default: "RNA")
use_imputed	Logical, whether to add imputed data as separate assay (default: TRUE)
add_projections	Logical, whether to add ZDB and DB projections as separate assays (default: TRUE)
add_embeddings	Logical, whether to add UMAP and PCA embeddings if available (default: TRUE)
min_cells	Integer, filter genes with counts in < min_cells (default: 0)
min_features	Integer, filter cells with < min_features genes (default: 0)
verbose	Logical, whether to print progress messages (default: TRUE)

Value

Seurat object with:

- RNA assay: Original or back-transformed counts
- imputed assay: GEDI imputed expression (if use_imputed = TRUE)
- ZDB/DB/ADB assays: GEDI projections (if add_projections = TRUE)
  - ZDB: Batch-corrected gene expression (genes x cells)
  - DB: Cell embeddings in latent factor space (K x cells)
  - ADB: Pathway activities (pathways x cells) - only if C matrix provided
- umap/pca reductions: Embeddings (if add_embeddings = TRUE and cached)
- meta.data: Sample labels and colData from GEDI model

Examples

```
# Load example data
pbmc <- SeuratObject::pbmc_small

# Train GEDI model
gedi_model <- seurat_to_gedi(pbmc, K = 15)
gedi_model$train(iterations = 10)

# Convert back to Seurat
seurat_obj <- gedi_to_seurat(
  gedi_model,
  use_imputed = TRUE,
  add_projections = TRUE,
  add_embeddings = TRUE
)

# Now can use Seurat functions
library(Seurat)
DefaultAssay(seurat_obj) <- "imputed"
seurat_obj <- FindVariableFeatures(seurat_obj)
```

---

install_optional_dependencies

List Optional GEDI Dependencies

Description

Reports which optional packages are needed and provides install commands. Does **not** install anything automatically.

Usage

```
install_optional_dependencies(which = "all", verbose = TRUE)
```

Arguments

which	Character vector specifying which dependency groups to query. Options: "h5" (hdf5r), "umap" (uwot), "validation" (digest), or "all" (default).
verbose	Logical, whether to print messages (default: TRUE)

Value

A named logical vector indicating which packages are installed (invisibly).

Examples

```
# Show which optional packages are missing
install_optional_dependencies()
```

---

list_h5_structure *List structure of H5 or H5AD file*

Description

Helper function to explore H5/H5AD file structure.

Usage

```
list_h5_structure(file_path, recursive = TRUE)
```

Arguments

file_path	Character. Path to the H5 or H5AD file.
recursive	Logical. List all nested groups. Default TRUE.

Value

data.frame with file structure information

Examples

```
# Round-trip: write a tiny H5AD via write_h5ad() then list its structure.
if (requireNamespace("hdf5r", quietly = TRUE) &&
    requireNamespace("SeuratObject", quietly = TRUE)) {
  pbmc_small <- SeuratObject::pbmc_small
  model <- CreateGEDIObject(
    Samples = pbmc_small@meta.data$orig.ident,
    M       = pbmc_small@assays$RNA@counts,
    K       = 3,
    verbose = 0
  )
  model$train(iterations = 5)
```

```
  tmp <- tempfile(fileext = ".h5ad")
  write_h5ad(model, tmp)
  list_h5_structure(tmp)
  unlink(tmp)
}
```

`plot_convergence` *Plot Training Convergence*

Description

Visualizes convergence of model parameters during training. Supports multiple layout styles for different use cases.

Usage

```
plot_convergence(
  model,
  layout = c("faceted", "separate", "compact"),
  params = NULL,
  log_scale = TRUE,
  smooth = FALSE,
  title = "Training Convergence"
)
```

Arguments

`model`	GEDI model object
`layout`	Character, layout style: • "faceted": Single plot with facet_wrap (default, best for reports) • "separate": List of individual plots (for interactive exploration) • "compact": Two-panel plot (global vs sample-specific)
`params`	Character vector, which parameters to include. Options: "Z", "A", "o", "Bi", "Qi", "oi", "si", "Rk", "Ro", "sigma2". NULL means all available.
`log_scale`	Logical, use log10 scale for y-axis
`smooth`	Logical, add smooth trend line
`title`	Character, plot title

Value

ggplot2 object (for "faceted" or "compact") or list of ggplot2 objects (for "separate")

Examples

```
if (requireNamespace("SeuratObject", quietly = TRUE)) {
  pbmc_small <- SeuratObject::pbmc_small
  model <- CreateGEDIObject(
    Samples = pbmc_small@meta.data$orig.ident,
    M       = pbmc_small@assays$RNA@counts,
    K       = 3,
    verbose = 0
  )
  model$train(iterations = 10, track_interval = 2)
  plot_convergence(model)
  plots <- plot_convergence(model, layout = "separate")
}
```

`plot_dispersion` *Plot Dispersion Analysis*

Description

Visualizes the relationship between expected and observed variance for count data. Useful for assessing model fit quality.

Usage

```
plot_dispersion(
  dispersion_df,
  show_identity = TRUE,
  point_size = 0.1,
  alpha = 0.5,
  title = "Dispersion Analysis"
)
```

Arguments

`dispersion_df`	Data frame from compute_dispersion() with columns: Expected_Var, Observed_Var, Sample, n, bin
`show_identity`	Logical, whether to show y=x identity line
`point_size`	Numeric, size of points
`alpha`	Numeric, transparency of points
`title`	Character, plot title

Value

ggplot2 object

Examples

```
if (requireNamespace("SeuratObject", quietly = TRUE)) {
  pbmc_small <- SeuratObject::pbmc_small
  M <- pbmc_small@assays$RNA@counts
  model <- CreateGEDIObject(
    Samples = pbmc_small@meta.data$orig.ident,
    M       = M,
    K       = 3,
    verbose = 0
  )
  model$train(iterations = 5)
  disp <- model$imputed$dispersion(M)
  plot_dispersion(disp)
}
```

---

plot_embedding *Plot Embedding with Improved API*

Description

Simplified interface for plotting embeddings with automatic caching. Model is the first argument, and color_by handles metadata/genes automatically.

Usage

```
plot_embedding(
  model,
  embedding = NULL,
  color_by = NULL,
  color = NULL,
  projection = "zdb",
  color_limits = NULL,
  palette = c("blue", "lightgrey", "red"),
  randomize = TRUE,
  point_size = 0.3,
  alpha = 0.9,
  raster = FALSE,
  xlab = "Dim 1",
  ylab = "Dim 2",
  title = NULL,
  legend_title = NULL,
  verbose = TRUE
)
```

Arguments

model	GEDI model object (or embedding matrix for backwards compatibility)
embedding	Character ("umap", "pca") or Nx2 matrix
color_by	Character: "sample", metadata column, gene name, or NULL
color	Vector for manual coloring (overrides color_by)
projection	Character: "zdb" or "db" for gene expression projection
color_limits	Numeric vector c(low, high) or NULL for auto
palette	Character vector of colors for continuous scale
randomize	Logical, randomize point order
point_size	Numeric, size of points
alpha	Numeric, transparency (0-1)
raster	Logical, use rasterization for large datasets
xlab	Character, x-axis label
ylab	Character, y-axis label
title	Character, plot title
legend_title	Character, legend title
verbose	Logical, print computation messages

Value

ggplot2 object

Examples

```
if (requireNamespace("SeuratObject", quietly = TRUE)) {
  pbmc_small <- SeuratObject::pbmc_small
  model <- CreateGEDIObject(
    Samples = pbmc_small@meta.data$orig.ident,
    M       = pbmc_small@assays$RNA@counts,
    K       = 3,
    verbose = 0
  )
  model$train(iterations = 5)
  plot_embedding(model, embedding = "pca", color_by = "sample")
}
```

`plot_feature_ratio` *Plot Two-Feature Comparison*

Description

Compares two features by computing their difference or correlation in the projected space. Mathematically grounded for GEDI's log-space representation.

Usage

```
plot_feature_ratio(
  model,
  gene1,
  gene2,
  comparison = "difference",
  embedding = "umap",
  projection = "zdb",
  color_limits = NULL,
  randomize = TRUE,
  point_size = 0.3,
  alpha = 0.9,
  title = NULL
)
```

Arguments

`model`	GEDI model object
`gene1`	Character or integer, first gene name or index
`gene2`	Character or integer, second gene name or index
`comparison`	Character, type of comparison ("difference" or "correlation")
`embedding`	Character specifying embedding type ("umap", "pca") or a custom N x 2 matrix
`projection`	Character, type of projection ("zdb" or "db")
`color_limits`	Numeric vector c(low, high) or NULL for auto-compute
`randomize`	Logical, whether to randomize point order
`point_size`	Numeric, size of points
`alpha`	Numeric, transparency of points
`title`	Character, plot title

Details

For comparison = "difference": Computes `(Z[gene1,] - Z[gene2,]) * D * B`, equivalent to `ZDB[gene1,] - ZDB[gene2,]`. In log-space, this represents log(gene1/gene2) in the original count space. Positive values indicate gene1 > gene2, negative indicates gene2 > gene1.

Value

ggplot2 object

Examples

```
if (requireNamespace("SeuratObject", quietly = TRUE)) {
  pbmc_small <- SeuratObject::pbmc_small
  model <- CreateGEDIObject(
    Samples = pbmc_small@meta.data$orig.ident,
    M       = pbmc_small@assays$RNA@counts,
    K       = 3,
    verbose = 0
  )
  model$train(iterations = 5)
  gene1 <- rownames(pbmc_small)[1]
  gene2 <- rownames(pbmc_small)[2]
  plot_feature_ratio(model, gene1, gene2, comparison = "difference",
                     embedding = "pca")
}
```

---

plot_features *Plot Multiple Features on Embedding*

Description

Efficiently plots multiple gene features on a 2D embedding using faceting. Computes projections on-the-fly without storing full ZDB matrix.

Usage

```
plot_features(
  model,
  features,
  embedding = "umap",
  projection = "zdb",
  color_limits = "global",
  ncol = NULL,
  randomize = TRUE,
  point_size = 0.2,
  alpha = 0.9,
  title = NULL
)
```

Arguments

model	GEDI model object
features	Character vector of gene names or integer indices
embedding	Character specifying embedding type ("umap", "pca") or a custom N x 2 matrix
projection	Character, type of projection to compute ("zdb" or "db")
color_limits	Character ("global" for shared scale, "individual" for per-facet scale) or numeric vector c(low, high)
ncol	Integer, number of columns in facet layout
randomize	Logical, whether to randomize point order
point_size	Numeric, size of points
alpha	Numeric, transparency of points
title	Character, plot title

Value

ggplot2 object with faceted features

Examples

```
if (requireNamespace("SeuratObject", quietly = TRUE) &&
    requireNamespace("uwot", quietly = TRUE)) {
  pbmc_small <- SeuratObject::pbmc_small
  model <- CreateGEDIObject(
    Samples = pbmc_small@meta.data$orig.ident,
    M       = pbmc_small@assays$RNA@counts,
    K       = 3,
    verbose = 0
  )
  model$train(iterations = 5)
  plot_features(model, c(1, 2), embedding = "pca")
}
```

---

plot_vector_field *Plot Vector Field from Dynamics Analysis*

Description

Visualizes vector fields showing cell state transitions. Uses binned aggregation for cleaner visualization without overplotting.

Usage

```
plot_vector_field(
  dynamics_svd,
  color = NULL,
  alpha = 1,
  n_bins = 50,
  min_per_bin = 10,
  randomize = TRUE,
  arrow_size = 0.5,
  arrow_length = 0.15,
  arrow_color = "black",
  dims = c(1, 2),
  xlab = NULL,
  ylab = NULL,
  title = NULL
)
```

Arguments

dynamics_svd	Result from model$dynamics$vector_field() or similar
color	Vector of length N for coloring arrows, or NULL
alpha	Vector of length N or scalar for arrow transparency
n_bins	Integer, number of bins per dimension for aggregation
min_per_bin	Integer, minimum observations required per bin
randomize	Logical, whether to randomize data order
arrow_size	Numeric, size of arrow lines
arrow_length	Numeric, length of arrow heads (in cm)
arrow_color	Character, color for arrows (if color is NULL)
dims	Integer vector of length 2, which dimensions to plot
xlab	Character, x-axis label
ylab	Character, y-axis label
title	Character, plot title

Value

ggplot2 object

Examples

```
# Build a tiny multi-sample fixture with a sample-level prior H so that
# model$dynamics$vector_field() is available.
set.seed(1)
n_genes <- 80; n_cells <- 60; n_samples <- 3
M <- Matrix::Matrix(
  matrix(stats::rpois(n_genes * n_cells, 5), n_genes, n_cells),
```

```
  sparse = TRUE
)
rownames(M) <- paste0("G", seq_len(n_genes))
colnames(M) <- paste0("C", seq_len(n_cells))
samples <- factor(rep(paste0("S", seq_len(n_samples)),
                      each = n_cells / n_samples))
H <- matrix(c(1, 0, 0,  0, 1, 0), nrow = 2, byrow = TRUE)
colnames(H) <- paste0("S", seq_len(n_samples))
rownames(H) <- c("cond_a", "cond_b")

model <- CreateGEDIObject(Samples = samples, M = M, K = 3, H = H,
                          verbose = 0)
model$train(iterations = 5)
vf <- model$dynamics$vector_field(start.cond = c(1, 0),
                                  end.cond   = c(0, 1))
plot_vector_field(vf)
```

---

read_h5 *Read 10X Genomics H5 file*

Description

Reads a 10X Genomics HDF5 file (CellRanger v2/v3 format) and converts it to a sparse dgCMatrix suitable for use with gedi R6 object.

Usage

```
read_h5(
  file_path,
  feature_format = "gene_name",
  unique.features = TRUE,
  verbose = FALSE
)
```

Arguments

file_path	Character. Path to the 10X H5 file.
feature_format	Character. Which feature identifier to use for gene names. Options: "gene_name" (default, uses feature names) or "gene_ids" (uses feature IDs).
unique.features	Logical. Make feature names unique. Default TRUE.
verbose	Logical. Print progress messages. Default FALSE.

Value

Sparse matrix (dgCMatrix) with genes as rows and cells as columns.

Examples

```
# read_h5() expects a 10x Genomics filtered_feature_bc_matrix.h5 file.
# The example body runs only when such a file is on disk and hdf5r is
# installed; otherwise it is silently skipped.
h5_file <- "filtered_feature_bc_matrix.h5"
if (file.exists(h5_file) &&
    requireNamespace("hdf5r", quietly = TRUE)) {
  expr_matrix <- read_h5(h5_file)
  expr_matrix_ids <- read_h5(h5_file, feature_format = "gene_ids")
}
```

---

read_h5ad *Read H5AD file and convert to sparse matrix*

Description

Reads an H5AD file (AnnData format) and extracts the expression matrix as a sparse dgCMatrix suitable for use with gedi R6 object.

Usage

```
read_h5ad(
  file_path,
  layer = NULL,
  use_raw = FALSE,
  transpose = TRUE,
  return_metadata = FALSE,
  feature_format = "gene_name",
  verbose = FALSE
)
```

Arguments

file_path	Character. Path to the H5AD file.
layer	Character. The layer to extract from the H5AD file. Default is NULL, which reads from X (the main expression matrix). Common alternatives include "counts", "data", "scaled", etc.
use_raw	Logical. If TRUE, reads from the raw.X slot instead of X. Default is FALSE.
transpose	Logical. If TRUE, transposes the matrix so genes are rows and cells are columns (gedi format). Default is TRUE.
return_metadata	Logical. If TRUE, returns a list with the expression matrix, cell metadata (obs), and gene metadata (var). If FALSE, returns only the expression matrix. Default is FALSE.

feature_format	Character. Which feature identifier to use for gene names when return_metadata = FALSE. Options: "gene_name" (default, uses var rownames) or "gene_ids" (uses var$gene_ids column). Default is "gene_name".
verbose	Logical. If TRUE, prints detailed progress messages. Default is FALSE.

Details

This function reads H5AD files, which are the standard format for AnnData objects in Python. Compatible with gedi R6 class for seamless integration.

Value

If return_metadata = FALSE, returns a sparse matrix (dgCMatrix) with genes as rows and cells as columns. If return_metadata = TRUE, returns a list with components:

- X: sparse expression matrix (genes x cells)
- obs: data.frame of cell metadata
- var: data.frame of gene metadata

Examples

```
# Round-trip: write a tiny H5AD via write_h5ad(), then read it back.
if (requireNamespace("hdf5r", quietly = TRUE) &&
    requireNamespace("SeuratObject", quietly = TRUE)) {
  pbmc_small <- SeuratObject::pbmc_small
  model <- CreateGEDIObject(
    Samples = pbmc_small@meta.data$orig.ident,
    M       = pbmc_small@assays$RNA@counts,
    K       = 3,
    verbose = 0
  )
  model$train(iterations = 5)

  tmp <- tempfile(fileext = ".h5ad")
  write_h5ad(model, tmp)
  expr_matrix <- read_h5ad(tmp)
  unlink(tmp)
}
```

---

seurat_to_gedi *Convert Seurat Object to GEDI Model*

Description

Extracts count data from a Seurat object and creates a GEDI model. Automatically validates that the data contains raw counts (not normalized). Handles both Seurat v4 and v5, including split layers in v5.

Usage

```
seurat_to_gedi(
  seurat_object,
  assay = "RNA",
  slot = "counts",
  sample_column = "orig.ident",
  subset_samples = NULL,
  K = 10,
  mode = "Bl2",
  C = NULL,
  H = NULL,
  validate_counts = TRUE,
  use_variable_features = TRUE,
  verbose = TRUE,
  ...
)
```

Arguments

seurat_object	Seurat object
assay	Character, which assay to use (default: "RNA")
slot	Character, which slot/layer to extract (default: "counts"). For Seurat v5 with split layers (e.g., counts.CTRL, counts.STIM), this will automatically detect and combine all matching layers.
sample_column	Character, column name in meta.data for sample labels (default: "orig.ident")
subset_samples	Character vector, subset to specific samples (default: NULL = all)
K	Integer, number of latent factors (default: 10)
mode	Character, normalization mode: "Bl2" or "Bsphere" (default: "Bl2")
C	Gene-level prior matrix (genes x pathways) (default: NULL)
H	Sample-level covariate matrix (covariates x samples) (default: NULL)
validate_counts	Logical, whether to validate data appears to be counts (default: TRUE)
use_variable_features	Logical, whether to subset to highly variable features (default: TRUE). If TRUE, uses genes from VariableFeatures(seurat_object).
verbose	Logical, whether to print progress messages (default: TRUE)
...	Additional arguments passed to CreateGEDIObject()

Value

GEDI R6 object

Examples

```
library(Seurat)

# Load example data
pbmc <- SeuratObject::pbmc_small

# Basic usage
gedi_model <- seurat_to_gedi(
  seurat_object = pbmc,
  sample_column = "orig.ident",
  K = 15
)

# Train the model
gedi_model$train(iterations = 10)
```

`write_h5ad` *Write GEDI model to H5AD file*

Description

Exports a trained GEDI model to H5AD (AnnData) format for interoperability with Python tools like scanpy. The file contains expression data, embeddings, metadata, and GEDI-specific parameters.

Usage

```
write_h5ad(
  model,
  file_path,
  X_slot = c("imputed", "projection", "original"),
  M = NULL,
  include_embeddings = TRUE,
  include_raw = FALSE,
  compression = 6,
  verbose = TRUE
)
```

Arguments

`model`	GEDI R6 object (must be trained)
`file_path`	Character. Path where the H5AD file should be written.
`X_slot`	Character. Which expression data to save in the main X slot: • "imputed": Imputed expression (default, requires M matrix) • "projection": ZDB projection (always available)

- "original": Original M matrix (requires M parameter)

`M` Optional. Original count matrix to save when X_slot="original" or include_raw=TRUE. Must match the dimensions used during model training.

`include_embeddings` Logical. Include PCA/UMAP in obsm if cached. Default TRUE.

`include_raw` Logical. If TRUE, saves original M in raw.X (requires M parameter). Default FALSE.

`compression` Integer. Gzip compression level (0-9). Default 6.

`verbose` Logical. Print progress messages. Default TRUE.

Details

The H5AD file structure contains:

- X: Main expression matrix (based on X_slot parameter)
- obs: Cell metadata (sample IDs, colData)
- var: Gene metadata (gene IDs)
- obsm: Cell embeddings (X_gedi, X_pca, X_umap)
- varm: Gene loadings (gedi_Z, gedi_Q_mean)
- uns: Model parameters and metadata
- raw.X: Original counts (if include_raw=TRUE)

The function handles the technical details of HDF5/AnnData compatibility:

- Writes sparse matrices in CSR format
- Transposes dense matrices for Python's row-major layout
- Creates scalar string attributes (not arrays) for AnnData compatibility
- Validates M matrix identity using fingerprints

Value

Invisibly returns the file path

Examples

```
if (requireNamespace("hdf5r", quietly = TRUE) &&
    requireNamespace("SeuratObject", quietly = TRUE)) {
  pbmc_small <- SeuratObject::pbmc_small
  model <- CreateGEDIObject(
    Samples = pbmc_small@meta.data$orig.ident,
    M       = pbmc_small@assays$RNA@counts,
    K       = 3,
    verbose = 0
  )
  model$train(iterations = 5)

  # Write to a temporary file (CRAN policy: never write to the user's
```

```
  # working directory in examples).
  tmp <- tempfile(fileext = ".h5ad")
  write_h5ad(model, tmp)
  write_h5ad(model, tmp, X_slot = "projection")
  unlink(tmp)
}
```

Index