Package: anndataR (via r-universe)

September 14, 2024

Title AnnData interoperability in R

Version 0.99.0

Description Bring the power and flexibility of AnnData to the R ecosystem, allowing you to effortlessly manipulate and analyze your single-cell data. This package lets you work with backed h5ad and zarr files, directly access various slots (e.g. X, obs, var), or convert the data into SingleCellExperiment and Seurat objects.

License MIT + file LICENSE

URL https://scverse.org/anndataR, https://github.com/scverse/anndataR

BugReports https://github.com/scverse/anndataR/issues

Depends R (>= 4.0.0)

Imports Matrix, methods, R6

Suggests anndata, BiocStyle, knitr, reticulate (>= 1.36.1), hdf5r (>= 1.3.11), rmarkdown, S4Vectors, SeuratObject, SingleCellExperiment, SummarizedExperiment, testthat (>= 3.0.0), withr

VignetteBuilder knitr

Config/Needs/website pkgdown, tibble, knitr, rprojroot, stringr, readr, purrr, dplyr, tidyr

Config/testthat/edition 3

Encoding UTF-8

Roxygen list(markdown = TRUE, r6 = TRUE)

RoxygenNote 7.3.2

biocViews SingleCell, DataImport, DataRepresentation

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

RemoteUrl https://github.com/scverse/anndataR

RemoteRef HEAD

RemoteSha dbc6897492d129bc59bf8de51bdaafdf90bcd22a

Contents

`AnnData` *Create an in-memory AnnData object.*

Description

For more information on the functionality of an AnnData object, see anndataR-package.

Usage

```
AnnData(
  X = NULL,
  obs = NULL,
  var = NULL,
  layers = NULL,
  obsm = NULL,
  varm = NULL,
  obsp = NULL,
  varp = NULL,
  uns = NULL,
  shape = shape
)
```

Arguments

`X`	Either `NULL` or a observation × variable matrix with dimensions consistent with `obs` and `var`.
`obs`	Either `NULL` or a `data.frame` with columns containing information about observations. If `NULL`, an `n_obs`×0 data frame will automatically be generated.
`var`	Either `NULL` or a `data.frame` with columns containing information about variables. If `NULL`, an `n_vars`×0 data frame will automatically be generated.
`layers`	Either `NULL` or a named list, where each element is an observation × variable matrix with dimensions consistent with `obs` and `var`.
`obsm`	The obsm slot is used to store multi-dimensional annotation arrays. It must be either `NULL` or a named list, where each element is a matrix with `n_obs` rows and an arbitrary number of columns.

varm	The varm slot is used to store multi-dimensional annotation arrays. It must be either NULL or a named list, where each element is a matrix with n_vars rows and an arbitrary number of columns.
obsp	The obsp slot is used to store sparse multi-dimensional annotation arrays. It must be either NULL or a named list, where each element is a sparse matrix where each dimension has length n_obs.
varp	The varp slot is used to store sparse multi-dimensional annotation arrays. It must be either NULL or a named list, where each element is a sparse matrix where each dimension has length n_vars.
uns	The uns slot is used to store unstructured annotation. It must be either NULL or a named list.
shape	Shape tuple (#observations, #variables). Can be provided if X or obs and var are not provided.

Value

An in-memory AnnData object.

See Also

anndataR-package

Examples

```
adata <- AnnData(
  X = matrix(1:12, nrow = 3, ncol = 4),
  obs = data.frame(
    row.names = paste0("obs", 1:3),
    n_counts = c(1, 2, 3),
    n_cells = c(1, 2, 3)
  ),
  var = data.frame(
    row.names = paste0("var", 1:4),
    n_cells = c(1, 2, 3, 4)
  )
)

adata
```

---

from_Seurat *Convert a Seurat object to an AnnData object*

Description

from_Seurat() converts a Seurat object to an AnnData object. Only one assay can be converted at a time.

Usage

```
from_Seurat(
  seurat_obj,
  output_class = c("InMemoryAnnData", "HDF5AnnData"),
  assay = NULL,
  X = "counts",
  ...
)
```

Arguments

seurat_obj	An object inheriting from Seurat.
output_class	Name of the AnnData class. Must be one of "HDF5AnnData" or "InMemoryAnnData".
assay	Assay to be converted. If NULL, DefaultAssay() is used.
X	Which of 'counts', 'data', or 'scale.data' will be used for X. By default, 'counts' will be used (if it is not empty), followed by 'data', then 'scale.data'. The remaining non-empty slots will be stored in different layers.
...	Additional arguments passed to the generator function.

Details

For more information on the functionality of an AnnData object, see anndataR-package.

See Also

anndataR-package

---

from_SingleCellExperiment

Convert a SingleCellExperiment object to an AnnData object

Description

from_SingleCellExperiment() converts a SingleCellExperiment to an AnnData object.

Usage

```
from_SingleCellExperiment(
  sce,
  output_class = c("InMemory", "HDF5AnnData"),
  ...
)
```

Arguments

sce	An object inheriting from SingleCellExperiment.
output_class	Name of the AnnData class. Must be one of "HDF5AnnData" or "InMemoryAnnData".
...	Additional arguments passed to the generator function. See the "Details" section for more information on which parameters

Value

from_SingleCellExperiment() returns an AnnData object (e.g., InMemoryAnnData) representing the content of sce.

Examples

```
## construct an AnnData object from a SingleCellExperiment
library(SingleCellExperiment)
sce <- SingleCellExperiment(
  assays = list(counts = matrix(1:5, 5L, 3L)),
  colData = DataFrame(cell = 1:3),
  rowData = DataFrame(gene = 1:5)
)
from_SingleCellExperiment(sce, "InMemory")
```

---

generate_dataset *Generate a dataset*

Description

Generate a dataset with different types of columns and layers

Usage

```
generate_dataset(
  n_obs = 10L,
  n_vars = 20L,
  x_type = "numeric_matrix",
  layer_types = c("numeric_matrix", "numeric_dense", "numeric_csparse",
    "numeric_rsparse", "numeric_matrix_with_nas", "numeric_dense_with_nas",
    "numeric_csparse_with_nas", "numeric_rsparse_with_nas", "integer_matrix",
   "integer_dense", "integer_csparse", "integer_rsparse", "integer_matrix_with_nas",
   "integer_dense_with_nas", "integer_csparse_with_nas", "integer_rsparse_with_nas"),
  obs_types = c("character", "integer", "factor", "factor_ordered", "logical", "numeric",
    "character_with_nas", "integer_with_nas", "factor_with_nas",
    "factor_ordered_with_nas", "logical_with_nas", "numeric_with_nas"),
  var_types = c("character", "integer", "factor", "factor_ordered", "logical", "numeric",
    "character_with_nas", "integer_with_nas", "factor_with_nas",
    "factor_ordered_with_nas", "logical_with_nas", "numeric_with_nas"),
```

```
obsm_types = c("numeric_matrix", "numeric_dense", "numeric_csparse", "numeric_rsparse",
 "numeric_matrix_with_nas", "numeric_dense_with_nas", "numeric_csparse_with_nas",
 "numeric_rsparse_with_nas", "integer_matrix", "integer_dense", "integer_csparse",
  "integer_rsparse", "integer_matrix_with_nas", "integer_dense_with_nas",
 "integer_csparse_with_nas", "integer_rsparse_with_nas", "character", "integer",
  "factor", "factor_ordered", "logical", "numeric", "character_with_nas",
  "integer_with_nas", "factor_with_nas",
   "factor_ordered_with_nas",
  "logical_with_nas", "numeric_with_nas"),
varm_types = c("numeric_matrix", "numeric_dense", "numeric_csparse", "numeric_rsparse",
 "numeric_matrix_with_nas", "numeric_dense_with_nas", "numeric_csparse_with_nas",
 "numeric_rsparse_with_nas", "integer_matrix", "integer_dense", "integer_csparse",
  "integer_rsparse", "integer_matrix_with_nas", "integer_dense_with_nas",
 "integer_csparse_with_nas", "integer_rsparse_with_nas", "character", "integer",
  "factor", "factor_ordered", "logical", "numeric", "character_with_nas",
  "integer_with_nas", "factor_with_nas",
   "factor_ordered_with_nas",
  "logical_with_nas", "numeric_with_nas"),
obsp_types = c("numeric_matrix", "numeric_dense", "numeric_csparse", "numeric_rsparse",
 "numeric_matrix_with_nas", "numeric_dense_with_nas", "numeric_csparse_with_nas",
 "numeric_rsparse_with_nas", "integer_matrix", "integer_dense", "integer_csparse",
  "integer_rsparse", "integer_matrix_with_nas", "integer_dense_with_nas",
  "integer_csparse_with_nas", "integer_rsparse_with_nas"),
varp_types = c("numeric_matrix", "numeric_dense", "numeric_csparse", "numeric_rsparse",
 "numeric_matrix_with_nas", "numeric_dense_with_nas", "numeric_csparse_with_nas",
 "numeric_rsparse_with_nas", "integer_matrix", "integer_dense", "integer_csparse",
  "integer_rsparse", "integer_matrix_with_nas", "integer_dense_with_nas",
  "integer_csparse_with_nas", "integer_rsparse_with_nas"),
 uns_types = c("scalar_character", "scalar_integer", "scalar_factor",
  "scalar_factor_ordered", "scalar_logical", "scalar_numeric",
 "scalar_character_with_nas", "scalar_integer_with_nas", "scalar_factor_with_nas",
  "scalar_factor_ordered_with_nas", "scalar_logical_with_nas",
  "scalar_numeric_with_nas", "vec_character", "vec_integer", "vec_factor",
 "vec_factor_ordered", "vec_logical", "vec_numeric", "vec_character_with_nas",
 "vec_integer_with_nas", "vec_factor_with_nas", "vec_factor_ordered_with_nas",
  "vec_logical_with_nas",
   "vec_numeric_with_nas", "df_character", "df_integer",
  "df_factor", "df_factor_ordered", "df_logical", "df_numeric",
  "df_character_with_nas", "df_integer_with_nas", "df_factor_with_nas",
  "df_factor_ordered_with_nas", "df_logical_with_nas", "df_numeric_with_nas",
  "mat_numeric_matrix", "mat_numeric_dense", "mat_numeric_csparse",
 "mat_numeric_rsparse", "mat_numeric_matrix_with_nas", "mat_numeric_dense_with_nas",
 "mat_numeric_csparse_with_nas", "mat_numeric_rsparse_with_nas", "mat_integer_matrix",

   "mat_integer_dense", "mat_integer_csparse", "mat_integer_rsparse",
  "mat_integer_matrix_with_nas", "mat_integer_dense_with_nas",
  "mat_integer_csparse_with_nas", "mat_integer_rsparse_with_nas", "list"),
 example = FALSE,
```

```
  format = c("list", "AnnData", "SingleCellExperiment", "Seurat")
)
```

Arguments

n_obs	Number of observations to generate
n_vars	Number of variables to generate
x_type	Type of matrix to generate for X
layer_types	Types of matrices to generate for layers
obs_types	Types of vectors to generate for obs
var_types	Types of vectors to generate for var
obsm_types	Types of matrices to generate for obsm
varm_types	Types of matrices to generate for varm
obsp_types	Types of matrices to generate for obsp
varp_types	Types of matrices to generate for varp
uns_types	Types of objects to generate for uns
example	If TRUE, the types will be overridden to a small set of types. This is useful for documentations.
format	Object type to output, one of "list", "AnnData", "SingleCellExperiment", or "Seurat".

Value

Object containing the generated dataset as defined by output

Examples

```
dummy <- generate_dataset()
## Not run:
dummy <- generate_dataset(format = "AnnData")
dummy <- generate_dataset(format = "SingleCellExperiment")
dummy <- generate_dataset(format = "Seurat")

## End(Not run)
```

---

read_h5ad *Read H5AD*

Description

Read data from a H5AD file

Usage

```
read_h5ad(
  path,
  to = c("InMemoryAnnData", "HDF5AnnData", "SingleCellExperiment", "Seurat"),
  mode = c("r", "r+", "a", "w", "w-", "x"),
  ...
)
```

Arguments

path	Path to the H5AD file to read
to	The type of object to return. Must be one of: "InMemoryAnnData", "HDF5AnnData", "SingleCellExperiment", "Seurat"
mode	The mode to open the HDF5 file. • a creates a new file or opens an existing one for read/write. • r opens an existing file for reading. • r+ opens an existing file for read/write. • w creates a file, truncating any existing ones. • w-/x are synonyms, creating a file and failing if it already exists.
...	Extra arguments provided to adata$to_SingleCellExperiment() or adata$to_Seurat(). See AnnData() for more information on the arguments of these functions. Note: update this documentation when r-lib/roxygen2#955 is resolved.

Value

The object specified by to

Examples

```
h5ad_file <- system.file("extdata", "example.h5ad", package = "anndataR")

# Read the H5AD as a SingleCellExperiment object
if (requireNamespace("SingleCellExperiment", quietly = TRUE)) {
  sce <- read_h5ad(h5ad_file, to = "SingleCellExperiment")
}

# Read the H5AD as a Seurat object
if (requireNamespace("SeuratObject", quietly = TRUE)) {
  seurat <- read_h5ad(h5ad_file, to = "Seurat")
}
```

write_h5ad *Write H5AD*

Description

Write an H5AD file

Usage

```
write_h5ad(
  object,
  path,
  compression = c("none", "gzip", "lzf"),
  mode = c("w-", "r", "r+", "a", "w", "x")
)
```

Arguments

object	The object to write, either a "SingleCellExperiment" or a "Seurat" object
path	Path of the file to write to
compression	The compression algorithm to use when writing the HDF5 file. Can be one of "none", "gzip" or "lzf". Defaults to "none".
mode	The mode to open the HDF5 file. • a creates a new file or opens an existing one for read/write. • r+ opens an existing file for read/write. • w creates a file, truncating any existing ones • w-/x are synonyms creating a file and failing if it already exists.

Value

path invisibly

Examples

```
adata <- AnnData(
  X = matrix(1:5, 3L, 5L),
  layers = list(
    A = matrix(5:1, 3L, 5L),
    B = matrix(letters[1:5], 3L, 5L)
  ),
  obs = data.frame(row.names = LETTERS[1:3], cell = 1:3),
  var = data.frame(row.names = letters[1:5], gene = 1:5)
)
h5ad_file <- tempfile(fileext = ".h5ad")
write_h5ad(adata, h5ad_file)

# Write a SingleCellExperiment as an H5AD
```

```
if (requireNamespace("SingleCellExperiment", quietly = TRUE)) {
  ncells <- 100
  counts <- matrix(rpois(20000, 5), ncol = ncells)
  logcounts <- log2(counts + 1)

  pca <- matrix(runif(ncells * 5), ncells)
  tsne <- matrix(rnorm(ncells * 2), ncells)

  sce <- SingleCellExperiment::SingleCellExperiment(
    assays = list(counts = counts, logcounts = logcounts),
    reducedDims = list(PCA = pca, tSNE = tsne)
  )

  h5ad_file <- tempfile(fileext = ".h5ad")
  write_h5ad(sce, h5ad_file)
}

# Write a Seurat as a H5AD
if (requireNamespace("SeuratObject", quietly = TRUE)) {
  # TODO: uncomment this code when the seurat converter is fixed
  # counts <- matrix(1:15, 3L, 5L)
  # dimnames(counts) <- list(
  #   letters[1:3],
  #   LETTERS[1:5]
  # )
  # gene.metadata <- data.frame(
  #   row.names = LETTERS[1:5],
  #   gene = 1:5
  # )
  # obj <- SeuratObject::CreateSeuratObject(counts, meta.data = gene.metadata)
  # cell.metadata <- data.frame(
  #   row.names = letters[1:3],
  #   cell = 1:3
  # )
  # obj <- SeuratObject::AddMetaData(obj, cell.metadata)
  #
  # h5ad_file <- tempfile(fileext = ".h5ad")
  # write_h5ad(obj, h5ad_file)
}
```

Index