Package 'anndataR'

November 11, 2026

Title AnnData interoperability in R

Version 1.2.3

Description Bring the power and flexibility of AnnData to the R ecosystem, allowing you to effortlessly manipulate and analyse 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://anndataR.scverse.org/, https://github.com/scverse/anndataR

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

Depends R (>= 4.5.0)

Imports cli, lifecycle, Matrix, methods, purrr, R6 (>= 2.4.0), reticulate (>= 1.41.1), rlang, stats

Suggests BiocFileCache, BiocStyle, knitr, processx, rhdf5 (>= 2.52.1), Rarr (>= 1.11.12), rmarkdown, S4Vectors, Seurat, SeuratObject, SingleCellExperiment, spelling, SummarizedExperiment, testthat (>= 3.0.0), vctrs, withr, yaml

VignetteBuilder knitr

biocViews SingleCell, DataImport, DataRepresentation

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

Config/testthat/edition 3

Encoding UTF-8

Roxygen list(markdown = TRUE, r6 = TRUE)

RoxygenNote 7.3.3

Language en-GB

git_url https://git.bioconductor.org/packages/anndataR

git_branch RELEASE_3_23

git_last_commit 89fad23

git_last_commit_date 2026-10-22

Repository Bioconductor 3.23

Date/Publication 2026-11-11

Author Robrecht Cannoodt [aut, cre] (ORCID:
<https://orcid.org/0000-0003-3641-729X>, github: rcannood),
Luke Zappia [aut] (ORCID: <https://orcid.org/0000-0001-7744-8565>,
github: lazappi),
Martin Morgan [aut] (ORCID: <https://orcid.org/0000-0002-5874-8148>,
github: mtmorgan),
Louise Deconinck [aut] (ORCID: <https://orcid.org/0000-0001-8100-6823>,
github: LouiseDck),
Danila Bredikhin [ctb] (ORCID: <https://orcid.org/0000-0001-8089-6983>,
github: gtca),
Isaac Virshup [ctb] (ORCID: <https://orcid.org/0000-0002-1710-8945>,
github: ivirshup),
Brian Schilder [ctb] (ORCID: <https://orcid.org/0000-0001-5949-2191>,
github: bschilder),
Chananchida Sang-aram [ctb] (ORCID:
<https://orcid.org/0000-0002-0922-0822>, github: csangara),
Artür Manukyan [ctb] (ORCID: <https://orcid.org/0000-0002-0441-9517>,
github: Artur-man),
Hugo Gruson [ctb] (ORCID: <https://orcid.org/0000-0002-4094-1476>,
github: Bisaloo),
Data Intuitive [fnd],
Chan Zuckerberg Initiative [fnd],
scverse consortium [spn]

Maintainer Robrecht Cannoodt <rcannood@gmail.com>

Contents

anndataR-package 3
AbstractAnnData 4
AbstractAnnData-s3methods 9
AnnData 11
AnnData-usage 13
AnnDataView 15
as_AnnData 17
as_HDF5AnnData 21
as_InMemoryAnnData 22
as_ReticulateAnnData 23
as_Seurat 24
as_SingleCellExperiment 27
as_ZarrAnnData 29
generate_dataset 30
HDF5AnnData 32
InMemoryAnnData 36

read_h5ad 38
read_zarr 39
reticulate-helpers 40
ReticulateAnnData 42
write_h5ad 44
write_zarr 46
ZarrAnnData 48

Index **52**

---

anndataR-package *anndataR: AnnData interoperability in R*

Description

Bring the power and flexibility of AnnData to the R ecosystem, allowing you to effortlessly manipulate and analyse 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.

Author(s)

Maintainer: Robrecht Cannoodt <rcannood@gmail.com> (ORCID) (rcannood)

Authors:

- Luke Zappia <luke@lazappi.id.au> (ORCID) (lazappi)
- Martin Morgan <mtmorgan.bioc@gmail.com> (ORCID) (mtmorgan)
- Louise Deconinck <louise.deconinck@gmail.com> (ORCID) (LouiseDck)

Other contributors:

- Danila Bredikhin <danila.bredikhin@embl.de> (ORCID) (gtca) [contributor]
- Isaac Virshup (ORCID) (ivirshup) [contributor]
- Brian Schilder <brian_schilder@alumni.brown.edu> (ORCID) (bschilder) [contributor]
- Chananchida Sang-aram (ORCID) (csangara) [contributor]
- Artür Manukyan (ORCID) (Artur-man) [contributor]
- Hugo Gruson (ORCID) (Bisaloo) [contributor]
- Data Intuitive <info@data-intuitive.com> [funder]
- Chan Zuckerberg Initiative [funder]
- scverse consortium [sponsor]

See Also

Useful links:

- https://anndataR.scverse.org/
- https://github.com/scverse/anndataR
- Report bugs at https://github.com/scverse/anndataR/issues

AbstractAnnData *Abstract AnnData class*

Description

This class is an abstract representation of an AnnData object. It is intended to be used as a base class for concrete implementations of AnnData objects, such as InMemoryAnnData or HDF5AnnData.

See AnnData-usage for details on creating and using AnnData objects.

Value

An AbstractAnnData object

Active bindings

X See AnnData-usage

layers See AnnData-usage

obs See AnnData-usage

var See AnnData-usage

obs_names See AnnData-usage

var_names See AnnData-usage

obsm See AnnData-usage

varm See AnnData-usage

obsp See AnnData-usage

varp See AnnData-usage

uns See AnnData-usage

Methods

Public methods:

- AbstractAnnData$print()
- AbstractAnnData$shape()
- AbstractAnnData$n_obs()
- AbstractAnnData$n_vars()
- AbstractAnnData$obs_keys()
- AbstractAnnData$var_keys()
- AbstractAnnData$layers_keys()
- AbstractAnnData$obsm_keys()
- AbstractAnnData$varm_keys()
- AbstractAnnData$obsp_keys()
- AbstractAnnData$varp_keys()
- AbstractAnnData$uns_keys()

- AbstractAnnData$as_SingleCellExperiment()
- AbstractAnnData$as_Seurat()
- AbstractAnnData$as_InMemoryAnnData()
- AbstractAnnData$as_ReticulateAnnData()
- AbstractAnnData$as_HDF5AnnData()
- AbstractAnnData$as_ZarrAnnData()
- AbstractAnnData$write_h5ad()
- AbstractAnnData$write_zarr()
- AbstractAnnData$clone()

Method `print()`**:** See AnnData-usage

Usage:

```
AbstractAnnData$print(...)
```

Arguments:

... Optional arguments to print method

Method `shape()`**:** See AnnData-usage

Usage:

```
AbstractAnnData$shape()
```

Method `n_obs()`**:** See AnnData-usage

Usage:

```
AbstractAnnData$n_obs()
```

Method `n_vars()`**:** See AnnData-usage

Usage:

```
AbstractAnnData$n_vars()
```

Method `obs_keys()`**:** See AnnData-usage

Usage:

```
AbstractAnnData$obs_keys()
```

Method `var_keys()`**:** See AnnData-usage

Usage:

```
AbstractAnnData$var_keys()
```

Method `layers_keys()`**:** See AnnData-usage

Usage:

```
AbstractAnnData$layers_keys()
```

Method `obsm_keys()`**:** See AnnData-usage

Usage:

```
AbstractAnnData$obsm_keys()
```

Method `varm_keys()`**:** See AnnData-usage

Usage:

```
AbstractAnnData$varm_keys()
```

Method `obsp_keys()`**:** See AnnData-usage

Usage:

```
AbstractAnnData$obsp_keys()
```

Method `varp_keys()`**:** See AnnData-usage

Usage:

```
AbstractAnnData$varp_keys()
```

Method `uns_keys()`**:** See AnnData-usage

Usage:

```
AbstractAnnData$uns_keys()
```

Method `as_SingleCellExperiment()`**:** Convert to `SingleCellExperiment`
See `as_SingleCellExperiment()` for more details on the conversion

Usage:

```
AbstractAnnData$as_SingleCellExperiment(
  x_mapping = NULL,
  assays_mapping = TRUE,
  colData_mapping = TRUE,
  rowData_mapping = TRUE,
  reducedDims_mapping = TRUE,
  colPairs_mapping = TRUE,
  rowPairs_mapping = TRUE,
  metadata_mapping = TRUE
)
```

Arguments:

`x_mapping` See `as_SingleCellExperiment()`

`assays_mapping` See `as_SingleCellExperiment()`

`colData_mapping` See `as_SingleCellExperiment()`

`rowData_mapping` See `as_SingleCellExperiment()`

`reducedDims_mapping` See `as_SingleCellExperiment()`

`colPairs_mapping` See `as_SingleCellExperiment()`

`rowPairs_mapping` See `as_SingleCellExperiment()`

`metadata_mapping` See `as_SingleCellExperiment()`

Returns: A `SingleCellExperiment` object

Method `as_Seurat()`**:** Convert to `Seurat`
See `as_Seurat()` for more details on the conversion

Usage:

```
AbstractAnnData$as_Seurat(
  assay_name = "RNA",
  x_mapping = NULL,
  layers_mapping = TRUE,
  object_metadata_mapping = TRUE,
  assay_metadata_mapping = TRUE,
  reduction_mapping = TRUE,
  graph_mapping = TRUE,
  misc_mapping = TRUE
)
```

Arguments:

`assay_name` See `as_Seurat()`

`x_mapping` See `as_Seurat()`

`layers_mapping` See `as_Seurat()`

`object_metadata_mapping` See `as_Seurat()`

`assay_metadata_mapping` See `as_Seurat()`

`reduction_mapping` See `as_Seurat()`

`graph_mapping` See `as_Seurat()`

`misc_mapping` See `as_Seurat()`

Returns: A `Seurat` object

Method `as_InMemoryAnnData()`**:** Convert to an `InMemoryAnnData`
See `as_InMemoryAnnData()` for more details on the conversion

Usage:

```
AbstractAnnData$as_InMemoryAnnData()
```

Returns: An InMemoryAnnData object

Method `as_ReticulateAnnData()`**:** Convert to a `ReticulateAnnData`
See `as_ReticulateAnnData()` for more details on the conversion

Usage:

```
AbstractAnnData$as_ReticulateAnnData()
```

Returns: A ReticulateAnnData object

Method `as_HDF5AnnData()`**:** Convert to an `HDF5AnnData`
See `as_HDF5AnnData()` for more details on the conversion

Usage:

```
AbstractAnnData$as_HDF5AnnData(
  file,
  compression = c("none", "gzip", "lzf"),
  chunk_size = "auto",
  mode = c("w-", "r", "r+", "a", "w", "x")
)
```

Arguments:

`file` See `as_HDF5AnnData()`

compression See as_HDF5AnnData()

chunk_size See as_HDF5AnnData()

mode See as_HDF5AnnData()

Returns: An HDF5AnnData object

Method as_ZarrAnnData(): Convert to a ZarrAnnData
See as_ZarrAnnData() for more details on the conversion

Usage:

```
AbstractAnnData$as_ZarrAnnData(
  file,
 compression = c("none", "gzip", "blosc", "zstd", "lzma", "bz2", "zlib", "lz4"),
  mode = c("w-", "r", "r+", "a", "w", "x")
)
```

Arguments:

file See as_ZarrAnnData()

compression See as_ZarrAnnData()

mode See as_ZarrAnnData()

Returns: A ZarrAnnData object

Method write_h5ad(): Write the AnnData object to an H5AD file
See write_h5ad() for details

Usage:

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

Arguments:

path See write_h5ad()

compression See write_h5ad()

chunk_size See write_h5ad()

mode See write_h5ad()

Returns: path invisibly

Method write_zarr(): Write the AnnData object to a Zarr file
See write_zarr() for details

Usage:

```
AbstractAnnData$write_zarr(
  path,
 compression = c("none", "gzip", "blosc", "zstd", "lzma", "bz2", "zlib", "lz4"),
  mode = c("w-", "r", "r+", "a", "w", "x")
)
```

Arguments:

`path` See `write_zarr()`

`compression` See `write_zarr()`

`mode` See `write_zarr()`

Returns: `path` invisibly

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
AbstractAnnData$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

AnnData-usage for details on creating and using `AnnData` objects

Other AnnData classes: `AnnDataView`, `HDF5AnnData`, `InMemoryAnnData`, `ReticulateAnnData`, `ZarrAnnData`

`AbstractAnnData-s3methods`

S3 Methods for AbstractAnnData Objects

Description

These S3 methods provide standard R interfaces for AbstractAnnData objects, making them behave like native R objects with familiar syntax.

Usage

```
## S3 method for class 'AbstractAnnData'
dim(x)

## S3 method for class 'AbstractAnnData'
nrow(x)

## S3 method for class 'AbstractAnnData'
ncol(x)

## S3 method for class 'AbstractAnnData'
dimnames(x)

## S3 replacement method for class 'AbstractAnnData'
dimnames(x) <- value

## S3 method for class 'AbstractAnnData'
x[i, j, drop = TRUE, ...]
```

Arguments

x	An AbstractAnnData object
value	For dimnames<-: A list of two character vectors (obs_names, var_names)
i	Row indices (observations). Can be numeric, logical, or character.
j	Column indices (variables). Can be numeric, logical, or character.
drop	Ignored (for compatibility with generic)
...	Additional arguments passed to methods

Details

Subsetting behaviour: The [method supports logical, integer, and character subsetting for both observations (rows) and variables (columns). However, unlike standard R behaviour:

- Logical vectors are **not recycled** and must have the exact same length as the dimension being subset
- **Negative indices are not supported** (R's "exclude these" syntax)

These design choices ensure clear and predictable subsetting behaviour for biological data matrices, avoiding potential confusion from accidental recycling or exclusion patterns.

The following S3 methods are available:

- dim(x): Get dimensions (n_obs, n_vars), equivalent to x$shape()
- nrow(x): Get number of observations, equivalent to x$n_obs()
- ncol(x): Get number of variables, equivalent to x$n_vars()
- dimnames(x): Get dimension names, (obs_names, var_names)
- rownames(x): Get observation names, equivalent to x$obs_names()
- colnames(x): Get variable names, equivalent to x$var_names()
- dimnames(x) <- value: Set dimension names
- rownames(x) <- value: Set observation names, equivalent to x$obs_names() <- ...
- colnames(x) <- value: Set variable names, equivalent to x$var_names() <- ...
- x[i, j]: Subset observations and/or variables

Value

- dim: Numeric vector of length 2 (n_obs, n_vars)
- nrow, ncol: Integer count
- dimnames: List with obs_names and var_names
- rownames, colnames: Character vector
- dimnames<-, rownames<-, colnames<-: The modified object (invisibly)
- [: A AnnDataView object with the specified subset

Examples

```
# Create example data
ad <- generate_dataset(n_obs = 100, n_vars = 50, format = "AnnData")

# Standard R methods work
dim(ad)
nrow(ad)
ncol(ad)
dimnames(ad)
rownames(ad)
colnames(ad)

# Set names using dimnames
dimnames(ad) <- list(
  paste0("cell_", 1:nrow(ad)),
  paste0("gene_", 1:ncol(ad))
)

# Or set names individually (uses dimnames<- internally)
rownames(ad) <- paste0("cell_", 1:nrow(ad))
colnames(ad) <- paste0("gene_", 1:ncol(ad))

# Subsetting creates AnnDataView
subset_ad <- ad[1:10, 1:5]
subset_ad <- ad[rep(c(TRUE, FALSE), length.out = nrow(ad)), ] # logical subsetting (no recycling)
subset_ad <- ad[c("cell_1", "cell_2"), c("gene_1", "gene_2")]  # name subsetting
```

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

Description

For more information on the functionality of an AnnData object, see AnnData-usage

Usage

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

Arguments

X	See the X slot in AnnData-usage
obs	See the obs slot in AnnData-usage
var	See the var slot in AnnData-usage
layers	See the layers slot in AnnData-usage
obsm	See the obsm slot in AnnData-usage
varm	See the varm slot in AnnData-usage
obsp	See the obsp slot in AnnData-usage
varp	See the varp slot in AnnData-usage
uns	See the uns slot in AnnData-usage
shape	Shape tuple (e.g. c(n_obs, n_vars)). Can be provided if both X or obs and var are not provided.

Value

An InMemoryAnnData object

See Also

AnnData-usage for details of AnnData structure and usage

Other AnnData creators: as_AnnData(), read_h5ad(), read_zarr()

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
```

AnnData-usage *AnnData structure and usage*

Description

The AnnData object stores a data matrix X together with annotations of observations obs (obsm, obsp) and variables var (varm, varp). Additional layers of data can be stored in layers and unstructured annotations in uns.

Back ends:

There are different back ends for AnnData objects that inherit from the abstract AbstractAnnData class and store and access data in different ways:

- InMemoryAnnData stores data in memory
- HDF5AnnData provides an interface to a H5AD file
- ZarrAnnData provides an interface to a Zarr store
- ReticulateAnnData wraps a Python AnnData object via **reticulate**

See the class documentation for details.

Usage:

The items listed as **"Slots"** are elements of the AnnData object that contain data and can be accessed or set. **"Fields"** return information about the AnnData object but cannot be set directly. Both, as well as methods, can be accessed using the $ operator

For example:

- adata$X returns the X matrix
- adata$X <- x sets the X matrix
- adata$method() calls a method

Value

An AnnData object inheriting from AbstractAnnData

Fields

shape Dimensions (observations x variables) of the AnnData object

n_obs Number of observations

n_vars Number of variables

obs_keys Keys (column names) of obs

var_keys Keys (column names) of var

layers_keys Keys (element names) of layers

obsm_keys Keys (element names) of obsm

varm_keys Keys (element names) of varm

obsp_keys Keys (element names) of obsp

varp_keys Keys (element names) of varp

uns_keys Keys (element names) of uns

Slots

`X` The main data matrix. Either `NULL` or an observation x variable matrix (without dimnames) with dimensions consistent with `n_obs` and `n_vars`.

`layers` Additional data layers. Must be `NULL` or a named list of matrices having dimensions consistent with `n_obs` and `n_vars`.

`obs` Observation annotations. A `data.frame` with columns containing information about observations. The number of rows of `obs` defines the observation dimension of the `AnnData` object (`n_obs`). If `NULL`, an `n_obs` × 0 `data.frame` will automatically be generated.

`var` Variable annotations. A `data.frame` with columns containing information about variables. The number of rows of `var` defines the variable dimension of the `AnnData` object (`n_vars`). If `NULL`, an `n_vars` × 0 `data.frame` will automatically be generated.

`obs_names` Observation names. Either `NULL` or a vector of unique identifiers used to identify each row of `obs` and to act as an index into the observation dimension of the `AnnData` object. For compatibility with *R* representations, `obs_names` should be a unique character vector.

`var_names` Variable names. Either `NULL` or a vector of unique identifiers used to identify each row of `var` and to act as an index into the variable dimension of the `AnnData` object. For compatibility with *R* representations, `var_names` should be a unique character vector.

`obsm` Multi-dimensional observation annotation. Must be `NULL` or a named list of array-like elements with number of rows equal to `n_obs`.

`varm` Multi-dimensional variable annotations. Must be `NULL` or a named list of array-like elements with number of rows equal to `n_vars`.

`obsp` Observation pairs. Must be `NULL` or a named list of array-like elements with number of rows and columns equal to `n_obs`.

`varp` Variable pairs. Must be `NULL` or a named list of array-like elements with number of rows and columns equal to `n_vars`.

`uns` Unstructured annotations. Must be `NULL` or a named list.

Methods

Conversion methods::

`as_SingleCellExperiment()` Convert to `SingleCellExperiment::SingleCellExperiment`, see `as_SingleCellExperiment()`

`as_Seurat()` Convert to `SeuratObject::Seurat`, see `as_Seurat()`

`as_InMemoryAnnData()` Convert to `InMemoryAnnData`, as `as_InMemoryAnnData()`

`as_HDF5AnnData()` Convert to `HDF5AnnData`, see `as_HDF5AnnData()`

`as_ZarrAnnData()` Convert to `ZarrAnnData`, see `as_ZarrAnnData()`

`as_ReticulateAnnData()` Convert to `ReticulateAnnData`, see `as_ReticulateAnnData()`

Output methods::

`write_h5ad()` Write the `AnnData` object to an HDF5 file, see `write_h5ad()`

`write_zarr()` Write the `AnnData` object to a Zarr store, see `write_zarr()`

General methods::

`print()` Print a summary of the `AnnData` object

Functions that can be used to create AnnData objects

`AnnData()` Create an InMemoryAnnData object

`read_h5ad()` Read an `AnnData` from a H5AD file

`read_zarr()` Read an `AnnData` from a Zarr store

`as_AnnData()` Convert other objects to an `AnnData` object

See Also

The documentation for the Python `anndata` package https://anndata.readthedocs.io/en/stable/

AbstractAnnData for the abstract class that all `AnnData` objects inherit from

InMemoryAnnData for the in-memory implementation of `AnnData`

HDF5AnnData for the HDF5-backed implementation of `AnnData`

ZarrAnnData for the Zarr-backed implementation of `AnnData`

ReticulateAnnData for the reticulate-based implementation that wraps Python AnnData objects

---

`AnnDataView` *A View of an AnnData Object*

Description

A lazy view of an AnnData object that allows applying subsetting operations without immediately executing them. Subsetting is applied when converting to a concrete AnnData implementation (InMemoryAnnData, HDF5AnnData) or other formats (SingleCellExperiment, Seurat).

Value

A `AnnDataView` object

Super class

`anndataR::AbstractAnnData` -> `AnnDataView`

Active bindings

`X` See AnnData-usage

`layers` See AnnData-usage

`obs` See AnnData-usage

`var` See AnnData-usage

`obs_names` See AnnData-usage

`var_names` See AnnData-usage

`obsm` See AnnData-usage

varm See AnnData-usage

obsp See AnnData-usage

varp See AnnData-usage

uns See AnnData-usage

Methods

Public methods:

- AnnDataView$new()
- AnnDataView$subset()
- AnnDataView$clone()

Method new(): Create a new AnnDataView object

Usage:

```
AnnDataView$new(base_adata, i, j)
```

Arguments:

base_adata An existing AnnData object to create a view of

i Optional initial obs subset (logical, integer, or character vector)

j Optional initial var subset (logical, integer, or character vector)

Returns: A new AnnDataView object Subset the AnnDataView

Method subset():

Usage:

```
AnnDataView$subset(i, j)
```

Arguments:

i Row indices (observations). Can be numeric, logical, or character.

j Column indices (variables). Can be numeric, logical, or character.

Returns: A new AnnDataView object

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
AnnDataView$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

See Also

AnnData-usage for details on creating and using AnnData objects

Other AnnData classes: AbstractAnnData, HDF5AnnData, InMemoryAnnData, ReticulateAnnData, ZarrAnnData

Examples

```
# Create a base AnnData object
ad <- AnnData(
  X = matrix(1:15, 3L, 5L),
  obs = data.frame(row.names = LETTERS[1:3], cell_type = c("A", "B", "A")),
  var = data.frame(row.names = letters[1:5], gene_type = c("X", "Y", "X", "Y", "Z"))
)

# Create a view with lazy subsetting using S3 [ method
view <- ad[ad$obs$cell_type == "A", ad$var$gene_type %in% c("X", "Y")]

# Apply subsetting by converting to a concrete implementation
result <- view$as_InMemoryAnnData()
```

---

as_AnnData *Convert to an* AnnData *object*

Description

Convert other objects to an AnnData object. See the sections below for details on how slots are mapped between objects. For more information on the functionality of an AnnData object, see AnnData-usage.

Usage

```
as_AnnData(
  x,
  x_mapping = NULL,
  layers_mapping = TRUE,
  obs_mapping = TRUE,
  var_mapping = TRUE,
  obsm_mapping = TRUE,
  varm_mapping = TRUE,
  obsp_mapping = TRUE,
  varp_mapping = TRUE,
  uns_mapping = TRUE,
  assay_name = NULL,
 output_class = c("InMemory", "HDF5AnnData", "ZarrAnnData", "ReticulateAnnData"),
  ...
)

## S3 method for class 'SingleCellExperiment'
as_AnnData(
  x,
  x_mapping = NULL,
  layers_mapping = TRUE,
```

```
  obs_mapping = TRUE,
  var_mapping = TRUE,
  obsm_mapping = TRUE,
  varm_mapping = TRUE,
  obsp_mapping = TRUE,
  varp_mapping = TRUE,
  uns_mapping = TRUE,
  assay_name = TRUE,
 output_class = c("InMemory", "HDF5AnnData", "ZarrAnnData", "ReticulateAnnData"),
  ...
)

## S3 method for class 'Seurat'
as_AnnData(
  x,
  x_mapping = NULL,
  layers_mapping = TRUE,
  obs_mapping = TRUE,
  var_mapping = TRUE,
  obsm_mapping = TRUE,
  varm_mapping = TRUE,
  obsp_mapping = TRUE,
  varp_mapping = TRUE,
  uns_mapping = TRUE,
  assay_name = NULL,
 output_class = c("InMemory", "HDF5AnnData", "ZarrAnnData", "ReticulateAnnData"),
  ...
)
```

Arguments

x	The object to convert
x_mapping	A string specifying the data to map to the X slot. If NULL, no data will be copied to the X slot. See below for details.
layers_mapping	A named character vector where the names are keys of layers in the new AnnData object and values are the names of items in the corresponding slot of x. See below for details.
obs_mapping	A named character vector where the names are names of obs columns in the new AnnData object and values are the names of columns in the corresponding slot of x. See below for details.
var_mapping	A named character vector where the names are names of var columns in the new AnnData object and values are the names of columns in the corresponding slot of x. See below for details.
obsm_mapping	A named character vector where the names are keys of obsm in the new AnnData object and values are the names of items in the corresponding slot of x. See below for details.

varm_mapping	A named character vector where the names are keys of varm in the new AnnData object and values are the names of items in the corresponding slot of x. See below for details.
obsp_mapping	A named character vector where the names are keys of obsp in the new AnnData object and values are the names of items in the corresponding slot of x. See below for details.
varp_mapping	A named character vector where the names are keys of varp in the new AnnData object and values are the names of items in the corresponding slot of x. See below for details.
uns_mapping	A named character vector where the names are keys of uns in the new AnnData object and values are the names of items in the corresponding slot of x. See below for details.
assay_name	For SeuratObject::Seurat objects, the name of the assay to be converted. If NULL, the default assay will be used (SeuratObject::DefaultAssay()). This is ignored for other objects.
output_class	The AnnData class to convert to. Must be one of "HDF5AnnData" or "InMemoryAnnData".
...	Additional arguments passed to the generator function for output_class

Value

An AnnData object of the class requested by output_class containing the data specified in the mapping arguments.

Details of mapping arguments

All mapping arguments except for x_mapping expect a named character vector where names are the keys of the slot in the AnnData object and values are the names of items in the corresponding slot of x. If TRUE, the conversion function will guess which items to copy as described in the conversion tables for each object type. In most cases, the default is to copy all items using the same names except where the correspondence between objects is unclear. To avoid copying anything to a slot, set the mapping argument to FALSE. Empty mapping arguments (NULL, c(), list()) will be treated as FALSE with a warning. If an unnamed vector is provided, the values will be used as names.

- TRUE will guess which items to copy as described in the conversion tables for each object type
- c(adata_item = "x_item") will copy x_item from the slot in x to adata_item in the corresponding slot of new AnnData object
- FALSE will avoid copying anything to the slot
- c("x_item") is equivalent to c(x_item = "x_item")

Converting from a SingleCellExperiment object

This table describes how slots in a SingleCellExperiment::SingleCellExperiment object to the new AnnData object.

From SingleCellExperiment	To AnnData	Example mapping argument
assays(x)	adata$X	x_mapping = "counts"

assays(x)	adata$layers	layers_mapping = c(counts = "counts")
colData(x)	adata$obs	obs_mapping = c(n_counts = "n_counts", cell_type = "CellT
rowData(x)	adata$var	var_mapping = c(n_cells = "n_cells", pct_zero = "PctZero"
reducedDims(x)	adata$obsm	obsm_mapping = c(X_pca = "pca")
featureLoadings(reducedDims(x))	adata$varm	varm_mapping = c(PCs = "pca")
colPairs(x)	adata$obsp	obsp_mapping = c(connectivities = "RNA_nn")
rowPairs(x)	adata$varp	varp_mapping = c(similarities = "gene_overlaps")
metadata(x)	adata$uns	uns_mapping = c(metadata = "project_metadata")

Unnamed assays:

If `assayNames(x)` is `NULL` or any assay names are empty they will automatically be named with a warning:

Examples:

- Old names: `NULL` -> New names: `"assay1", "assay2", ...`
- Old names: `"counts"` -> New names: `"counts", "assay2"`

Converting from a `Seurat` object

Only one assay can be converted from a `SeuratObject::Seurat` object to an `AnnData` object at a time. This can be controlled using the `assay_name` argument. By default, the current default assay will be used.

This table describes how slots in a `SeuratObject::Seurat` object to the new `AnnData` object.

From `Seurat`	**To** `AnnData`	**Example mapping argument**	**Defau**
Layers(x)	adata$X	x_mapping = "counts"	Nothi
Layers(x)	adata$layers	layers_mapping = c(counts = "counts")	All it
x[[]]	adata$obs	obs_mapping = c(n_counts = "n_counts", cell_type = "CellType")	All co
x[[assay_name]][[]]	adata$var	var_mapping = c(n_cells = "n_cells", pct_zero = "PctZero")	All co
Reductions(x)	adata$obsm	obsm_mapping = c(X_pca = "pca")	All em
Loadings(x)	adata$varm	varm_mapping = c(PCs = "pca")	All va
Graphs(x)	adata$obsp	obsp_mapping = c(connectivities = "RNA_nn")	All gr
Misc(x)	adata$varp	varp_mapping = c(similarities = "gene_overlaps")	No da
Misc(x)	adata$uns	uns_mapping = c(metadata = "project_metadata")	All it

Graph conversion:

By default, all graphs in a `SeuratObject::Seurat` object that match the assay being converted are copied to the `obsp` slot of the new `AnnData` object. If a graph does not have an associated assay:

- If `assay_name` is the default assay, they will be *converted* with a warning
- if `assay_name` is not the default assay, they will be *skipped* with a warning

To override this behaviour, provide a custom mapping using the `obsp_mapping` argument.

Unexpected dimensions:

A `SeuratObject::Seurat` is more flexible in terms of the dimensions of items that can be stored in various slots. For example, a `Layer` does not have to match the dimensions of the whole object. If an item has unexpected dimensions, it will be skipped with a warning.

See Also

Other AnnData creators: AnnData(), read_h5ad(), read_zarr()

Other object converters: as_HDF5AnnData(), as_InMemoryAnnData(), as_ReticulateAnnData(), as_Seurat(), as_SingleCellExperiment(), as_ZarrAnnData(), reticulate-helpers

Examples

```
# Convert a Seurat object to an AnnData object
library(Seurat)

counts <- matrix(rbinom(20000, 1000, .001), nrow = 100)
obj <- CreateSeuratObject(counts = counts)
obj <- NormalizeData(obj)
obj <- FindVariableFeatures(obj)
obj <- ScaleData(obj)
obj <- RunPCA(obj, npcs = 10L)
obj <- FindNeighbors(obj)
obj <- RunUMAP(obj, dims = 1:10)

as_AnnData(obj)


# Convert a SingleCellExperiment object to an AnnData object
library(SingleCellExperiment)

sce <- SingleCellExperiment(
  assays = list(counts = matrix(1:5, 5L, 3L)),
  colData = DataFrame(cell = 1:3, row.names = paste0("Cell", 1:3)),
  rowData = DataFrame(gene = 1:5, row.names = paste0("Gene", 1:5))
)

as_AnnData(sce)
```

---

as_HDF5AnnData *Convert an* AnnData *to an* HDF5AnnData

Description

Convert another AnnData object to an HDF5AnnData object

Usage

```
as_HDF5AnnData(
  adata,
  file,
  compression = c("none", "gzip", "lzf"),
  chunk_size = "auto",
  mode = c("w-", "r", "r+", "a", "w", "x")
)
```

Arguments

adata	An AnnData object to be converted to HDF5AnnData
file	The file name (character) of the .h5ad file
compression	The compression algorithm to use when writing the HDF5 file. Can be one of "none", "gzip" or "lzf". Defaults to "none".
chunk_size	The target chunk size in bytes to use when writing HDF5 datasets. When "auto" (default), the chunk size is determined automatically using an algorithm that mimics h5py's auto-chunking behaviour. Set to NULL to disable chunking (contiguous storage, the rhdf5 default), or a number to use a specific target size in bytes.
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

Value

An HDF5AnnData object with the same data as the input AnnData object.

See Also

Other object converters: as_AnnData(), as_InMemoryAnnData(), as_ReticulateAnnData(), as_Seurat(), as_SingleCellExperiment(), as_ZarrAnnData(), reticulate-helpers

---

as_InMemoryAnnData *Convert an* AnnData *to an* InMemoryAnnData

---

Description

Convert another AnnData object to an InMemoryAnnData object

Usage

```
as_InMemoryAnnData(adata)
```

Arguments

adata	An AnnData object to be converted to InMemoryAnnData

Value

An InMemoryAnnData object with the same data as the input AnnData object

See Also

Other object converters: as_AnnData(), as_HDF5AnnData(), as_ReticulateAnnData(), as_Seurat(), as_SingleCellExperiment(), as_ZarrAnnData(), reticulate-helpers

Examples

```
ad <- 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)
)
ad$as_InMemoryAnnData()
```

---

as_ReticulateAnnData *Convert an* AnnData *to a* ReticulateAnnData

---

Description

Convert another AnnData object to a ReticulateAnnData object

Usage

```
as_ReticulateAnnData(adata)
```

Arguments

adata	An AnnData object to be converted to ReticulateAnnData

Value

A ReticulateAnnData object with the same data as the input AnnData object

See Also

Other object converters: as_AnnData(), as_HDF5AnnData(), as_InMemoryAnnData(), as_Seurat(), as_SingleCellExperiment(), as_ZarrAnnData(), reticulate-helpers

Examples

```
# Requires Python anndata to be installed
if (requireNamespace("reticulate", quietly = TRUE) &&
      reticulate::py_module_available("anndata")) {
  ad <- 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)
  )
  ad$as_ReticulateAnnData()
}
```

`as_Seurat` *Convert an* `AnnData` *to a* `Seurat`

Description

Convert an `AnnData` object to a `Seurat` object

Usage

```
as_Seurat(
  adata,
  assay_name = "RNA",
  x_mapping = NULL,
  layers_mapping = TRUE,
  object_metadata_mapping = TRUE,
  assay_metadata_mapping = TRUE,
  reduction_mapping = TRUE,
  graph_mapping = TRUE,
  misc_mapping = TRUE
)
```

Arguments

`adata`	The `AnnData` object to convert.
`assay_name`	Name of the assay to be created in the new `Seurat` object
`x_mapping`	A string specifying the name of the layer in the resulting `Seurat` object where the data in the `X` slot of `adata` will be mapped to
`layers_mapping`	A named vector where names are names of `Layers` in the resulting `Seurat` object and values are keys of `layers` in `adata`. See below for details.
`object_metadata_mapping`	A named vector where names are cell metadata columns in the resulting `Seurat` object and values are columns of `obs` in `adata`. See below for details.
`assay_metadata_mapping`	A named vector where names are variable metadata columns in the assay of the resulting `Seurat` object and values are columns of `var` in `adata`. See below for details.

`reduction_mapping`
: A named vector where names are names of `Embeddings` in the resulting `Seurat` object and values are keys of `obsm` in `adata`. Alternatively, a named list where names are names of `Embeddings` in the resulting `Seurat` object and values are vectors with the items `"key"`, `"embeddings"` and (optionally) `"loadings"`. See below for details.

`graph_mapping`
: A named vector where names are names of `Graphs` in the resulting `Seurat` object and values are keys of `obsp` in `adata`. See below for details.

`misc_mapping`
: A named vector where names are names of `Misc` in the resulting `Seurat` object and values are keys of `uns` in `adata`. See below for details.

Details

Mapping arguments:

All mapping arguments expect a named character vector where names are the names of the slot in the `Seurat` object and values are the keys of the corresponding slot of `adata`. If `TRUE`, the conversion function will guess which items to copy as described in the conversion table below. In most cases, the default is to copy all items using the same names except where the correspondence between objects is unclear. The `reduction_mapping` argument can also accept a more complex list format, see below for details. To avoid copying anything to a slot, set the mapping argument to `FALSE`. Empty mapping arguments (`NULL`, `c()`, `list()`) will be treated as `FALSE` with a warning. If an unnamed vector is provided, the values will be used as names.

Examples::

- `TRUE` will guess which items to copy as described in the conversion table
- `c(seurat_item = "adata_item")` will copy `adata_item` from the slot in `adata` to `seurat_item` in the corresponding slot of the new `Seurat` object
- `FALSE` will avoid copying anything to the slot
- `c("adata_item")` is equivalent to `c(adata_item = "adata_item")`

Conversion table:

From `AnnData`	**To** `Seurat`	**Example mapping argument**
`adata$X`	`Layers(seurat)`	`x_mapping = "counts"` *OR* `layers_mapping = c(counts = NA)`
`adata$layers`	`Layers(seurat)`	`layers_mapping = c(counts = "counts")`
`adata$obs`	`seurat[[]]`	`object_metadata_mapping = c(n_counts = "n_counts", cell_type = "`
`adata$var`	`seurat[[assay_name]][[]]`	`assay_metadata_mapping = c(n_cells = "n_cells", pct_zero = "PctZ`
`adata$obsm`	`Reductions(seurat)`	`reduction_mapping = c(pca = "X_pca")` **OR** `reduction_mapping = lis`
`adata$obsp`	`Graphs(seurat)`	`graph_mapping = c(nn = "connectivities")`
`adata$varp`	*NA*	*NA*
`adata$uns`	`Misc(seurat)`	`misc_mapping = c(project_metadata = "metadata")`

The `reduction_mapping` argument:

For the simpler named vector format, the names should be the names of `Embeddings` in the resulting `Seurat` object, and the values should be the keys of `obsm` in `adata`. A key will created from the `obsm` key.

For more advanced mapping, use the list format where each item is a vector with the following names defining arguments to `SeuratObject::CreateDimReducObject()`:

- key: the key of the resulting SeuratObject::DimReduc object, passed to the key argument after processing with SeuratObject::Key()
- embeddings: a key of the obsm slot in adata, adata$obsm[[embeddings]] is passed to the embeddings argument
- loadings: a key of the varm slot in adata (optional), adata$varm[[loadings]] is passed to the loadings argument

The x_mapping and layers_mapping arguments:

In order to specify where the data in adata$X will be stored in the Layers(seurat) slot of the resulting object, you can use either the x_mapping argument or the layers_mapping argument. If you use x_mapping, it should be a string specifying the name of the layer in Layers(seurat) where the data in adata$X will be stored. If you use layers_mapping, it should be a named vector where names are names of Layers(seurat) and values are keys of layers in adata. In order to indicate the adata$X slot, you use NA as the value in the vector. The name you provide for x_mapping may not be a name in layers_mapping.

Value

A Seurat object containing the requested data from adata

See Also

Other object converters: as_AnnData(), as_HDF5AnnData(), as_InMemoryAnnData(), as_ReticulateAnnData(), as_SingleCellExperiment(), as_ZarrAnnData(), reticulate-helpers

Examples

```
ad <- AnnData(
  X = matrix(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)
)

# Default usage
seurat <- ad$as_Seurat(
  assay_name = "RNA",
  x_mapping = "counts",
  layers_mapping = TRUE,
  object_metadata_mapping = TRUE,
  assay_metadata_mapping = TRUE,
  reduction_mapping = TRUE,
  graph_mapping = TRUE,
  misc_mapping = TRUE
)
```

as_SingleCellExperiment *Convert an* AnnData *to a* SingleCellExperiment

Description

Convert an AnnData object to a SingleCellExperiment object

Usage

```
as_SingleCellExperiment(
  adata,
  x_mapping = NULL,
  assays_mapping = TRUE,
  colData_mapping = TRUE,
  rowData_mapping = TRUE,
  reducedDims_mapping = TRUE,
  colPairs_mapping = TRUE,
  rowPairs_mapping = TRUE,
  metadata_mapping = TRUE
)
```

Arguments

adata	The AnnData object to convert
x_mapping	A string specifying the name of the assay in the resulting SingleCellExperiment where the data in the X slot of adata will be mapped to
assays_mapping	A named vector where names are names of assays in the resulting SingleCellExperiment object and values are keys of layers in adata. See below for details.
colData_mapping	A named vector where names are columns of colData in the resulting SingleCellExperiment object and values are columns of obs in adata. See below for details.
rowData_mapping	A named vector where names are columns of rowData in the resulting SingleCellExperiment object and values are columns of var in adata. See below for details.
reducedDims_mapping	A named vector where names are names of reducedDims in the resulting SingleCellExperiment object and values are keys of obsm in adata. Alternatively, a named list where names are names of reducedDims in the resulting SingleCellExperiment object and values are vectors with the items "sampleFactors" and "featureLoadings" and/or "metadata". See below for details.
colPairs_mapping	A named vector where names are names of colPairs in the resulting SingleCellExperiment object and values are keys of obsp in adata. See below for details.

rowPairs_mapping
: A named vector where names are names of rowPairs in the resulting SingleCellExperiment object and values are keys of varp in adata. See below for details.

metadata_mapping
: A named vector where names are names of metadata in the resulting SingleCellExperiment object and values are keys of uns in adata. See below for details.

Details

Mapping arguments:

All mapping arguments expect a named character vector where names are the names of the slot in the SingleCellExperiment object and values are the keys of the corresponding slot of adata. If TRUE, the conversion function will guess which items to copy as described in the conversion tables below. In most cases, the default is to copy all items using the same names except where the correspondence between objects is unclear. The reducedDims_mapping argument can also accept a more complex list format, see below for details. To avoid copying anything to a slot, set the mapping argument to FALSE. Empty mapping arguments (NULL, c(), list()) will be treated as FALSE with a warning. If an unnamed vector is provided, the values will be used as names.

Examples::

- TRUE will guess which items to copy as described in the conversion table
- c(sce_item = "adata_item") will copy adata_item from the slot in adata to sce_item in the corresponding slot of the new SingleCellExperiment object
- FALSE will avoid copying anything to the slot
- c("adata_item") is equivalent to c(adata_item = "adata_item")

Conversion table:

From AnnData	**To** SingleCellExperiment	**Example mapping argument**
adata$X	assays(sce)	x_mapping = "counts"
adata$layers	assays(sce)	assays_mapping = c(counts = "counts")
adata$obs	colData(sce)	colData_mapping = c(n_counts = "n_counts", cell_type = "CellType"
adata$var	rowData(sce)	rowData_mapping = c(n_cells = "n_cells", pct_zero = "PctZero")
adata$obsm	reducedDims(sce)	reducedDims_mapping = c(pca = "X_pca") **OR** reducedDims_mapping = l
adata$obsp	colPairs(sce)	colPairs_mapping = c(nn = "connectivities")
adata$varp	rowPairs(sce)	rowPairs_mapping = c(gene_overlaps = "similarities")
adata$uns	metadata(sce)	uns_mapping = c(project_metadata = "metadata")

The reducedDims_mapping argument:

For the simpler named vector format, the names should be the names of reducedDims in the resulting SingleCellExperiment object, and the values should be the keys of obsm in adata.

For more advanced mapping, use the list format where each item is a vector with the following names used to create a SingleCellExperiment::LinearEmbeddingMatrix (if featureLoadings or metadata is provided):

- sampleFactors: a key of the obsm slot in adata, adata$obsm[[sampleFactors]] is passed to the sampleFactors argument
- featureLoadings: a key of the varm slot in adata (optional), adata$varm[[featureLoadings]] is passed to the featureLoadings argument

- metadata: a key of the uns slot in adata (optional), adata$uns[[metadata]] is passed to the metadata argument

The x_mapping **and** assays_mapping **arguments:**

In order to specify where the data in adata$X will be stored in the assays(sce) slot of the resulting object, you can use either the x_mapping argument or the assays_mapping argument. If you use x_mapping, it should be a string specifying the name of the layer in assays(sce) where the data in adata$X will be stored. If you use assays_mapping, it should be a named vector where names are names of assays(sce) and values are keys of layers in adata. In order to indicate the adata$X slot, you use NA as the value in the vector. The name you provide for x_mapping may not be a name in assays_mapping.

Value

A SingleCellExperiment object containing the requested data from adata

See Also

Other object converters: as_AnnData(), as_HDF5AnnData(), as_InMemoryAnnData(), as_ReticulateAnnData(), as_Seurat(), as_ZarrAnnData(), reticulate-helpers

Examples

```
ad <- 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)
)

# Default usage
sce <- ad$as_SingleCellExperiment(
  assays_mapping = TRUE,
  colData_mapping = TRUE,
  rowData_mapping = TRUE,
  reducedDims_mapping = TRUE,
  colPairs_mapping = TRUE,
  rowPairs_mapping = TRUE,
  metadata_mapping = TRUE
)
```

---

as_ZarrAnnData *Convert an* AnnData *to an* ZarrAnnData

Description

Convert another AnnData object to an ZarrAnnData object

Usage

```
as_ZarrAnnData(
  adata,
  file,
  compression = c("none", "gzip", "blosc", "zstd", "lzma", "bz2", "zlib", "lz4"),
  mode = c("w-", "r", "r+", "a", "w", "x")
)
```

Arguments

adata	An AnnData object to be converted to ZarrAnnData
file	The file name (character) of the .zarr file
compression	The compression algorithm to use when writing the Zarr file. Can be one of "none", "gzip", "blosc", "zstd", "lzma", "bz2", "zlib" or "lz4". Defaults to "none".
mode	The mode to open the Zarr 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

Value

A ZarrAnnData object with the same data as the input AnnData object.

See Also

Other object converters: as_AnnData(), as_HDF5AnnData(), as_InMemoryAnnData(), as_ReticulateAnnData(), as_Seurat(), as_SingleCellExperiment(), reticulate-helpers

---

generate_dataset *Generate a mock dataset*

Description

Generate a mock synthetic dataset with different types of columns and layers. This is primarily designed for use in tests, examples, vignettes and other documentation but is also provided to users for creating reproducible examples.

Use get_generator_types() to get the available types for each slot.

Usage

```
generate_dataset(
  n_obs = 10L,
  n_vars = 20L,
  x_type = "numeric_matrix",
  layer_types = get_generator_types(slot = "layers"),
  obs_types = get_generator_types(slot = "obs"),
  var_types = get_generator_types(slot = "var"),
  obsm_types = get_generator_types(slot = "obsm"),
  varm_types = get_generator_types(slot = "varm"),
  obsp_types = get_generator_types(slot = "obsp"),
  varp_types = get_generator_types(slot = "varp"),
  uns_types = get_generator_types(slot = "uns"),
  example = FALSE,
  format = c("list", "AnnData", "SingleCellExperiment", "Seurat")
)

get_generator_types(example = FALSE, slot = NULL)
```

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 with a small subset of types. This is useful for documentation.
format	Object type to output, one of "list", "AnnData", "SingleCellExperiment", or "Seurat".
slot	Which slot to return types for, if NULL a named list of all slots is returned

Details

To generate no data for a given slot, set the matching type argument to NULL or an empty vector, e.g. obs_types = c() will generate an empty obs data frame.

When generating SingleCellExperiment or Seurat objects, only some of the generated slots will be included in the output object. To generate a more complete object, use format = "AnnData" followed by adata$as_SingleCellExperiment() or adata$as_Seurat().

Use `get_generator_types()` to get a list of the available types for each slot, or for a specific slot by setting `slot`. If `example = TRUE`, only the example types are returned.

Value

For `generate_dataset()`, an object as defined by `output` containing the generated dataset

For `get_generator_types()`, a named list of character vectors or a single character vector if `slot` is not `NULL`

Examples

```
# Generate all types as a list
dummy <- generate_dataset()

# Generate the example types
dummy_example <- generate_dataset(example = TRUE)

# Generate an AnnData
dummy_anndata <- generate_dataset(format = "AnnData", example = TRUE)

# Generate a SingleCellExperiment
if (rlang::is_installed("SingleCellExperiment")) {
  dummy_sce <- generate_dataset(format = "SingleCellExperiment", example = TRUE)
}

# Generate a Seurat object
if (rlang::is_installed("SeuratObject")) {
  dummy_seurat <- generate_dataset(format = "Seurat", example = TRUE)
}

# Get all available generator types
get_generator_types()

# Get generator types for a specific slot
get_generator_types(slot = "obs")

# Get generator types used when example = TRUE
get_generator_types(example = TRUE)
```

`HDF5AnnData` *HDF5AnnData*

Description

Implementation of an HDF5-backed `AnnData` object. This class provides an interface to a H5AD file and minimal data is stored in memory until it is requested by the user. It is primarily designed as an intermediate object when reading/writing H5AD files but can be useful for accessing parts of large files.

See AnnData-usage for details on creating and using `AnnData` objects.

Value

An HDF5AnnData object

Super class

anndataR::AbstractAnnData -> HDF5AnnData

Active bindings

X See AnnData-usage

layers See AnnData-usage

obsm See AnnData-usage

varm See AnnData-usage

obsp See AnnData-usage

varp See AnnData-usage

obs See AnnData-usage

var See AnnData-usage

obs_names See AnnData-usage

var_names See AnnData-usage

uns See AnnData-usage

Methods

Public methods:

- HDF5AnnData$new()
- HDF5AnnData$obs_keys()
- HDF5AnnData$var_keys()
- HDF5AnnData$layers_keys()
- HDF5AnnData$obsm_keys()
- HDF5AnnData$varm_keys()
- HDF5AnnData$obsp_keys()
- HDF5AnnData$varp_keys()
- HDF5AnnData$uns_keys()
- HDF5AnnData$close()
- HDF5AnnData$n_obs()
- HDF5AnnData$n_vars()

Method new(): Close the HDF5 file when the object is garbage collected
HDF5AnnData constructor

Usage:

```
HDF5AnnData$new(
  file,
  X = NULL,
  obs = NULL,
  var = NULL,
  layers = NULL,
  obsm = NULL,
  varm = NULL,
  obsp = NULL,
  varp = NULL,
  uns = NULL,
  shape = NULL,
  mode = c("a", "r", "r+", "w", "w-", "x"),
  compression = c("none", "gzip", "lzf"),
  chunk_size = "auto"
)
```

Arguments:

`file` The file name (character) of the `.h5ad` file. If this file already exits, other arguments must be `NULL`.

`X` See the `X` slot in AnnData-usage

`obs` See the `obs` slot in AnnData-usage

`var` See the `var` slot in AnnData-usage

`layers` See the `layers` slot in AnnData-usage

`obsm` See the `obsm` slot in AnnData-usage

`varm` See the `varm` slot in AnnData-usage

`obsp` See the `obsp` slot in AnnData-usage

`varp` See the `varp` slot in AnnData-usage

`uns` See the `uns` slot in AnnData-usage

`shape` Shape tuple (e.g. `c(n_obs, n_vars)`). Can be provided if both `X` or `obs` and `var` are not provided.

`mode` The mode to open the HDF5 file. See `as_HDF5AnnData()` for details

`compression` The compression algorithm to use. See `as_HDF5AnnData()` for details

`chunk_size` The target chunk size in bytes. See `as_HDF5AnnData()` for details

Details: The constructor creates a new HDF5 `AnnData` interface object. This can either be used to either connect to an existing `.h5ad` file or to create a new one. If any additional slot arguments are set an existing file will be overwritten.

Method `obs_keys()`**:** See AnnData-usage

Usage:

```
HDF5AnnData$obs_keys()
```

Method `var_keys()`**:** See AnnData-usage

Usage:

```
HDF5AnnData$var_keys()
```

Method `layers_keys()`**:** See AnnData-usage

Usage:

```
HDF5AnnData$layers_keys()
```

Method `obsm_keys()`**:** See AnnData-usage

Usage:

```
HDF5AnnData$obsm_keys()
```

Method `varm_keys()`**:** See AnnData-usage

Usage:

```
HDF5AnnData$varm_keys()
```

Method `obsp_keys()`**:** See AnnData-usage

Usage:

```
HDF5AnnData$obsp_keys()
```

Method `varp_keys()`**:** See AnnData-usage

Usage:

```
HDF5AnnData$varp_keys()
```

Method `uns_keys()`**:** See AnnData-usage

Usage:

```
HDF5AnnData$uns_keys()
```

Method `close()`**:** Close the HDF5 file

Usage:

```
HDF5AnnData$close()
```

Method `n_obs()`**:** See the `n_obs` field in AnnData-usage

Usage:

```
HDF5AnnData$n_obs()
```

Method `n_vars()`**:** See the `n_vars` field in AnnData-usage

Usage:

```
HDF5AnnData$n_vars()
```

See Also

AnnData-usage for details on creating and using `AnnData` objects

Other AnnData classes: `AbstractAnnData`, `AnnDataView`, `InMemoryAnnData`, `ReticulateAnnData`, `ZarrAnnData`

InMemoryAnnData *InMemoryAnnData*

Description

Implementation of an in-memory AnnData object where data is stored within the R session. This is the simplest back end and will be most familiar to users. It is want you will want to use in most cases where you want to interact with an AnnData object.

See AnnData-usage for details on creating and using AnnData objects.

Value

An InMemoryAnnData object

Super class

anndataR::AbstractAnnData -> InMemoryAnnData

Active bindings

X See AnnData-usage

layers See AnnData-usage

obs See AnnData-usage

var See AnnData-usage

obs_names See AnnData-usage

var_names See AnnData-usage

obsm See AnnData-usage

varm See AnnData-usage

obsp See AnnData-usage

varp See AnnData-usage

uns See AnnData-usage

Methods

Public methods:

- InMemoryAnnData$new()
- InMemoryAnnData$clone()

Method new(): Creates a new instance of an in-memory AnnData object. Inherits from AbstractAnnData.

Usage:

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

Arguments:

`X` See the `X` slot in AnnData-usage

`obs` See the `obs` slot in AnnData-usage

`var` See the `var` slot in AnnData-usage

`layers` See the `layers` slot in AnnData-usage

`obsm` See the `obsm` slot in AnnData-usage

`varm` See the `varm` slot in AnnData-usage

`obsp` See the `obsp` slot in AnnData-usage

`varp` See the `varp` slot in AnnData-usage

`uns` See the `uns` slot in AnnData-usage

`shape` Shape tuple (e.g. `c(n_obs, n_vars)`). Can be provided if both `X` or `obs` and `var` are not provided.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
InMemoryAnnData$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

AnnData-usage for details on creating and using `AnnData` objects

Other AnnData classes: `AbstractAnnData`, `AnnDataView`, `HDF5AnnData`, `ReticulateAnnData`, `ZarrAnnData`

Examples

```
## complete example
ad <- AnnData(
  X = matrix(1:15, 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)
)
ad

## minimum example
AnnData(
  obs = data.frame(row.names = letters[1:10]),
  var = data.frame(row.names = LETTERS[1:5])
)
```

`read_h5ad` *Read H5AD*

Description

Read data from a H5AD file

Usage

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

Arguments

`path`	Path to the H5AD file to read
`as`	The type of object to return. One of: • `"InMemoryAnnData"`: Read the H5AD file into memory as an `InMemoryAnnData` object • `"HDF5AnnData"`: Read the H5AD file as an `HDF5AnnData` object • `"SingleCellExperiment"`: Read the H5AD file as a `SingleCellExperiment::SingleCellExperi` object • `"Seurat"`: Read the H5AD file as a `SeuratObject::Seurat` object
`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 the `as_*` conversion function for the object specified by `as`

Value

The object specified by as

See Also

Other AnnData creators: AnnData(), as_AnnData(), read_zarr()

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, as = "SingleCellExperiment")
}

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

---

read_zarr *Read Zarr*

Description

Read data from a Zarr store

Usage

```
read_zarr(
  path,
  as = c("InMemoryAnnData", "ZarrAnnData", "SingleCellExperiment", "Seurat"),
  mode = c("r", "r+", "a", "w", "w-", "x"),
  ...
)
```

Arguments

path	Path to the Zarr store to read
as	The type of object to return. One of: • "InMemoryAnnData": Read the Zarr store into memory as an InMemoryAnnData object • "ZarrAnnData": Read the Zarr store as an ZarrAnnData object • "SingleCellExperiment": Read the Zarr store as a SingleCellExperiment::SingleCellExperi object • "Seurat": Read the Zarr store as a SeuratObject::Seurat object

mode — The mode to open the Zarr 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 the as_* conversion function for the object specified by as

Value

The object specified by as

See Also

Other AnnData creators: AnnData(), as_AnnData(), read_h5ad()

Examples

```
# Please use "example_v3.zarr.zip" for AnnData stored as Zarr version 3
zarr_dir <- system.file("extdata", "example_v2.zarr.zip", package = "anndataR")
td <- tempdir(check = TRUE)
unzip(zarr_dir, exdir = td)
zarr_store <- file.path(td, "example_v2.zarr")

# Read the Zarr as a SingleCellExperiment object
if (requireNamespace("SingleCellExperiment", quietly = TRUE)) {
  sce <- read_zarr(zarr_store, as = "SingleCellExperiment")
}

# Read the Zarr as a Seurat object
if (requireNamespace("SeuratObject", quietly = TRUE)) {
  seurat <- read_zarr(zarr_store, as = "Seurat")
}
```

---

reticulate-helpers *Reticulate Helper Functions for AnnData Conversion*

Description

This file contains helper functions that enable seamless conversion between R and Python AnnData objects using the reticulate package. These functions provide automatic S3 method dispatch for converting AnnData objects across the R-Python boundary.

Usage

```
## S3 method for class 'collections.abc.Mapping'
py_to_r(x)

## S3 method for class 'anndata._core.anndata.AnnData'
py_to_r(x)

## S3 method for class 'anndata.AnnData'
py_to_r(x)

## S3 method for class 'AbstractAnnData'
r_to_py(x, convert = TRUE)
```

Arguments

x	An AbstractAnnData object (any anndataR implementation)
convert	Whether to convert the result (passed to reticulate)

Details

The main conversion functions include:

- `py_to_r.anndata.AnnData` and `py_to_r.anndata._core.anndata.AnnData`: Convert Python AnnData objects to R ReticulateAnnData objects
- `r_to_py.AbstractAnnData`: Converts R AbstractAnnData objects to Python AnnData objects
- `py_to_r.collections.abc.Mapping`: Converts Python mapping objects to R lists

These functions are automatically registered as S3 methods and are called when using `reticulate::py_to_r()` and `reticulate::r_to_py()` on compatible objects.

Value

A `ReticulateAnnData` object wrapping the Python object

A Python AnnData object

See Also

Other object converters: `as_AnnData()`, `as_HDF5AnnData()`, `as_InMemoryAnnData()`, `as_ReticulateAnnData()`, `as_Seurat()`, `as_SingleCellExperiment()`, `as_ZarrAnnData()`

Examples

```
# Requires Python anndata to be installed
if (requireNamespace("reticulate", quietly = TRUE) &&
      reticulate::py_module_available("anndata")) {

  library(reticulate)
```

```
  # Create Python AnnData object
  ad_py <- import("anndata", convert = FALSE)
  py_adata <- ad_py$AnnData(X = r_to_py(matrix(1:12, 3, 4)))

  # Automatic conversion to R (uses the py_to_r() method for AnnData)
  r_adata <- py_to_r(py_adata)

  # Automatic conversion back to Python (uses r_to_py.AbstractAnnData)
  py_adata2 <- r_to_py(r_adata)
}
```

`ReticulateAnnData` *ReticulateAnnData*

Description

Implementation of an `AnnData` object that wraps a Python **anndata** `AnnData` object using **reticulate**. This allows direct interaction with Python `AnnData` objects while maintaining the R interface. It is useful when you already have a Python `AnnData` or to access functionality that has not yet been implemented in **anndataR**.

See AnnData-usage for details on creating and using `AnnData` objects.

Value

A `ReticulateAnnData` object

Super class

`anndataR::AbstractAnnData` -> `ReticulateAnnData`

Active bindings

`X` See AnnData-usage

`layers` See AnnData-usage

`obs` See AnnData-usage

`var` See AnnData-usage

`obs_names` See AnnData-usage

`var_names` See AnnData-usage

`obsm` See AnnData-usage

`varm` See AnnData-usage

`obsp` See AnnData-usage

`varp` See AnnData-usage

`uns` See AnnData-usage

Methods

Public methods:

- ReticulateAnnData$new()
- ReticulateAnnData$n_obs()
- ReticulateAnnData$n_vars()
- ReticulateAnnData$py_anndata()

Method `new()`**:** `ReticulateAnnData` constructor

Usage:

```
ReticulateAnnData$new(
  py_anndata = NULL,
  X = NULL,
  obs = NULL,
  var = NULL,
  layers = NULL,
  obsm = NULL,
  varm = NULL,
  obsp = NULL,
  varp = NULL,
  uns = NULL,
  shape = NULL
)
```

Arguments:

`py_anndata` A Python AnnData object created using reticulate, or NULL to create a new empty Python AnnData object

`X` See the `X` slot in AnnData-usage (only used if py_anndata is NULL)

`obs` See the `obs` slot in AnnData-usage (only used if py_anndata is NULL)

`var` See the `var` slot in AnnData-usage (only used if py_anndata is NULL)

`layers` See the `layers` slot in AnnData-usage (only used if py_anndata is NULL)

`obsm` See the `obsm` slot in AnnData-usage (only used if py_anndata is NULL)

`varm` See the `varm` slot in AnnData-usage (only used if py_anndata is NULL)

`obsp` See the `obsp` slot in AnnData-usage (only used if py_anndata is NULL)

`varp` See the `varp` slot in AnnData-usage (only used if py_anndata is NULL)

`uns` See the `uns` slot in AnnData-usage (only used if py_anndata is NULL)

`shape` Shape tuple (e.g. `c(n_obs, n_vars)`). Can be provided if both `X` or `obs` and `var` are not provided. (only used if py_anndata is NULL)

Details: The constructor creates a new ReticulateAnnData interface object that wraps a Python AnnData object. If `py_anndata` is provided, it must be a valid Python AnnData object. If NULL, a new Python AnnData object will be created using the other provided arguments.

Method `n_obs()`**:** See the `n_obs` field in AnnData-usage

Usage:

```
ReticulateAnnData$n_obs()
```

Method `n_vars()`**:** See the `n_vars` field in AnnData-usage

Usage:

```
ReticulateAnnData$n_vars()
```

Method `py_anndata()`**:** Get the underlying Python AnnData object

Usage:

```
ReticulateAnnData$py_anndata()
```

Returns: The Python AnnData object wrapped by this ReticulateAnnData

See Also

AnnData-usage for details on creating and using `AnnData` objects

Other AnnData classes: `AbstractAnnData`, `AnnDataView`, `HDF5AnnData`, `InMemoryAnnData`, `ZarrAnnData`

`write_h5ad` *Write H5AD*

Description

Write an H5AD file

Usage

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

Arguments

`object`	The object to write, either a `SingleCellExperiment::SingleCellExperiment` or a `SeuratObject::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"`.
`chunk_size`	The target chunk size in bytes to use when writing HDF5 datasets. When `"auto"` (default), the chunk size is determined automatically using an algorithm that mimics h5py's auto-chunking behaviour. Set to `NULL` to disable chunking (contiguous storage, the rhdf5 default), or a number to use a specific target size in bytes. This only affects array-like datasets; scalar values are unaffected.
`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

`...` Additional arguments passed to `as_AnnData()`

Details

Compression:

Compression is currently not supported for Boolean arrays, they will be written uncompressed.

`NULL` **values:**

For compatibility with changes in Python **anndata** 0.12.0, `NULL` values in `uns` are written to H5AD files as a `NULL` dataset (instead of not being written at all). To disable this behaviour, set `option(anndataR.write_null = FALSE)`. This may be required to allow the file to be read by older versions of Python **anndata**.

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")
adata$write_h5ad(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)
  )

  adata <- as_AnnData(sce)
  h5ad_file <- tempfile(fileext = ".h5ad")
  adata$write_h5ad(h5ad_file)
```

```
}

# Write a Seurat as a H5AD
if (requireNamespace("Seurat", quietly = TRUE)) {
  library(Seurat)

  counts <- matrix(1:15, 5L, 3L)
  dimnames(counts) <- list(
    LETTERS[1:5],
    letters[1:3]
  )
  cell.metadata <- data.frame(
    row.names = letters[1:3],
    cell = 1:3
  )
  obj <- CreateSeuratObject(counts, meta.data = cell.metadata)
  gene.metadata <- data.frame(
    row.names = LETTERS[1:5],
    gene = 1:5
  )
  obj[["RNA"]] <- AddMetaData(GetAssay(obj), gene.metadata)

  adata <- as_AnnData(obj)
  h5ad_file <- tempfile(fileext = ".h5ad")
  adata$write_h5ad(h5ad_file)
}
```

`write_zarr` *Write Zarr*

Description

Write a Zarr file

Usage

```
write_zarr(
  object,
  path,
  compression = c("none", "gzip", "blosc", "zstd", "lzma", "bz2", "zlib", "lz4"),
  mode = c("w-", "r", "r+", "a", "w", "x"),
  ...
)
```

Arguments

`object`	The object to write, either a `SingleCellExperiment::SingleCellExperiment` or a `SeuratObject::Seurat` object
`path`	Path of the file to write to

compression — The compression algorithm to use when writing the Zarr file. Can be one of "none", "gzip", "blosc", "zstd", "lzma", "bz2", "zlib", "lz4". Defaults to "none". See help("compressors", package = "Rarr").

mode — The mode to open the Zarr 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

... — Additional arguments passed to as_AnnData()

Details

NULL values:

For compatibility with changes in Python **anndata** 0.12.0, NULL values in uns are written to Zarr files as a NULL dataset (instead of not being written at all). To disable this behaviour, set option(anndataR.write_null = FALSE). This may be required to allow the file to be read by older versions of Python **anndata**.

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)
)
zarr_store <- tempfile(fileext = ".zarr")
adata$write_zarr(zarr_store)

# Write a SingleCellExperiment as a Zarr store
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)
  )
```

```
  adata <- as_AnnData(sce)
  zarr_store <- tempfile(fileext = ".zarr")
  adata$write_zarr(zarr_store)
}

# Write a Seurat as a Zarr
if (requireNamespace("Seurat", quietly = TRUE)) {
  library(Seurat)

  counts <- matrix(1:15, 5L, 3L)
  dimnames(counts) <- list(
    LETTERS[1:5],
    letters[1:3]
  )
  cell.metadata <- data.frame(
    row.names = letters[1:3],
    cell = 1:3
  )
  obj <- CreateSeuratObject(counts, meta.data = cell.metadata)
  gene.metadata <- data.frame(
    row.names = LETTERS[1:5],
    gene = 1:5
  )
  obj[["RNA"]] <- AddMetaData(GetAssay(obj), gene.metadata)

  adata <- as_AnnData(obj)
  zarr_store <- tempfile(fileext = ".zarr")
  adata$write_zarr(zarr_store)
}
```

ZarrAnnData *ZarrAnnData*

Description

Implementation of a Zarr-backed AnnData object. This class provides an interface to a Zarr file and minimal data is stored in memory until it is requested by the user. It is primarily designed as an intermediate object when reading/writing Zarr files but can be useful for accessing parts of large files.

See AnnData-usage for details on creating and using AnnData objects.

Value

A ZarrAnnData object

Super class

anndataR::AbstractAnnData -> ZarrAnnData

Active bindings

`X` See AnnData-usage

`layers` See AnnData-usage

`obsm` See AnnData-usage

`varm` See AnnData-usage

`obsp` See AnnData-usage

`varp` See AnnData-usage

`obs` See AnnData-usage

`var` See AnnData-usage

`obs_names` See AnnData-usage

`var_names` See AnnData-usage

`uns` See AnnData-usage

Methods

Public methods:

- `ZarrAnnData$new()`
- `ZarrAnnData$n_obs()`
- `ZarrAnnData$n_vars()`
- `ZarrAnnData$obs_keys()`
- `ZarrAnnData$var_keys()`
- `ZarrAnnData$layers_keys()`
- `ZarrAnnData$obsm_keys()`
- `ZarrAnnData$varm_keys()`
- `ZarrAnnData$obsp_keys()`
- `ZarrAnnData$varp_keys()`
- `ZarrAnnData$uns_keys()`

Method `new()`: `ZarrAnnData` constructor

Usage:

```
ZarrAnnData$new(
  file,
  X = NULL,
  obs = NULL,
  var = NULL,
  layers = NULL,
  obsm = NULL,
  varm = NULL,
  obsp = NULL,
  varp = NULL,
  uns = NULL,
  shape = NULL,
  mode = c("a", "r", "r+", "w", "w-", "x"),
 compression = c("none", "gzip", "blosc", "zstd", "lzma", "bz2", "zlib", "lz4")
)
```

Arguments:

`file` The file name (character) of the `.zarr` file. If this file already exits, other arguments must be `NULL`.

`X` See the `X` slot in AnnData-usage

`obs` See the `obs` slot in AnnData-usage

`var` See the `var` slot in AnnData-usage

`layers` See the `layers` slot in AnnData-usage

`obsm` See the `obsm` slot in AnnData-usage

`varm` See the `varm` slot in AnnData-usage

`obsp` See the `obsp` slot in AnnData-usage

`varp` See the `varp` slot in AnnData-usage

`uns` See the `uns` slot in AnnData-usage

`shape` Shape tuple (e.g. `c(n_obs, n_vars)`). Can be provided if both `X` or `obs` and `var` are not provided.

`mode` The mode to open the Zarr file. See `as_ZarrAnnData()` for details

`compression` The compression algorithm to use. See `as_ZarrAnnData()` for details

Details: The constructor creates a new Zarr `AnnData` interface object. This can either be used to either connect to an existing `.zarr` file or to create a new one. If any additional slot arguments are set an existing file will be overwritten.

Method `n_obs()`**:** See the `n_obs` field in AnnData-usage

Usage:

```
ZarrAnnData$n_obs()
```

Method `n_vars()`**:** See the `n_vars` field in AnnData-usage

Usage:

```
ZarrAnnData$n_vars()
```

Method `obs_keys()`**:** See AnnData-usage

Usage:

```
ZarrAnnData$obs_keys()
```

Method `var_keys()`**:** See AnnData-usage

Usage:

```
ZarrAnnData$var_keys()
```

Method `layers_keys()`**:** See AnnData-usage

Usage:

```
ZarrAnnData$layers_keys()
```

Method `obsm_keys()`**:** See AnnData-usage

Usage:

```
ZarrAnnData$obsm_keys()
```

Method `varm_keys()`**:** See AnnData-usage

Usage:

```
ZarrAnnData$varm_keys()
```

Method `obsp_keys()`**:** See AnnData-usage

Usage:

```
ZarrAnnData$obsp_keys()
```

Method `varp_keys()`**:** See AnnData-usage

Usage:

```
ZarrAnnData$varp_keys()
```

Method `uns_keys()`**:** See AnnData-usage

Usage:

```
ZarrAnnData$uns_keys()
```

See Also

AnnData-usage for details on creating and using `AnnData` objects

Other AnnData classes: `AbstractAnnData`, `AnnDataView`, `HDF5AnnData`, `InMemoryAnnData`, `ReticulateAnnData`

Index

∗ **AnnData classes**
  AbstractAnnData, 4
  AnnDataView, 15
  HDF5AnnData, 32
  InMemoryAnnData, 36
  ReticulateAnnData, 42
  ZarrAnnData, 48
∗ **AnnData creators**
  AnnData, 11
  as_AnnData, 17
  read_h5ad, 38
  read_zarr, 39
∗ **internal**
  anndataR-package, 3
  as_HDF5AnnData, 21
  as_InMemoryAnnData, 22
  as_ReticulateAnnData, 23
  as_ZarrAnnData, 29
∗ **object converters**
  as_AnnData, 17
  as_HDF5AnnData, 21
  as_InMemoryAnnData, 22
  as_ReticulateAnnData, 23
  as_Seurat, 24
  as_SingleCellExperiment, 27
  as_ZarrAnnData, 29
  reticulate-helpers, 40
[.AbstractAnnData *(AbstractAnnData-s3methods)*, 9

AbstractAnnData, 4, *13*, *15, 16*, *35–37*, *41*, *44*, *51*
AbstractAnnData-s3methods, 9
AnnData, 11, *21*, *39, 40*
AnnData(), *15*
AnnData-usage, *4–6*, *9*, *11, 12*, 13, *15–17*, *32–37*, *42–44*, *48–51*
anndataR *(anndataR-package)*, 3
anndataR-package, 3
anndataR::AbstractAnnData, *15*, *33*, *36*, *42*, *48*
AnnDataView, *9*, 15, *35*, *37*, *44*, *51*
as_AnnData, *12*, 17, *22, 23*, *26*, *29, 30*, *39–41*
as_AnnData(), *15*, *45*, *47*
as_HDF5AnnData, *21*, 21, *23*, *26*, *29, 30*, *41*
as_HDF5AnnData(), *7, 8*, *14*, *34*
as_InMemoryAnnData, *21, 22*, 22, *23*, *26*, *29, 30*, *41*
as_InMemoryAnnData(), *7*, *14*
as_ReticulateAnnData, *21–23*, 23, *26*, *29, 30*, *41*
as_ReticulateAnnData(), *7*, *14*
as_Seurat, *21–23*, 24, *29, 30*, *41*
as_Seurat(), *6, 7*, *14*
as_SingleCellExperiment, *21–23*, *26*, 27, *30*, *41*
as_SingleCellExperiment(), *6*, *14*
as_ZarrAnnData, *21–23*, *26*, *29*, 29, *41*
as_ZarrAnnData(), *8*, *14*, *50*

dim.AbstractAnnData *(AbstractAnnData-s3methods)*, 9
dimnames.AbstractAnnData *(AbstractAnnData-s3methods)*, 9
dimnames<-.AbstractAnnData *(AbstractAnnData-s3methods)*, 9

generate_dataset, 30
get_generator_types *(generate_dataset)*, 30

HDF5AnnData, *4*, *7–9*, *13–16*, *21, 22*, 32, *37, 38*, *44*, *51*

InMemoryAnnData, *4*, *7*, *9*, *12–16*, *22*, *35*, 36, *38, 39*, *44*, *51*

ncol.AbstractAnnData *(AbstractAnnData-s3methods)*, 9

nrow.AbstractAnnData
(AbstractAnnData-s3methods), 9

py_to_r.anndata._core.anndata.AnnData
(reticulate-helpers), 40
py_to_r.anndata.AnnData
(reticulate-helpers), 40
py_to_r.collections.abc.Mapping
(reticulate-helpers), 40

r_to_py.AbstractAnnData
(reticulate-helpers), 40
read_h5ad, *12*, *21*, 38, *40*
read_h5ad(), *15*
read_zarr, *12*, *21*, *39*, 39
read_zarr(), *15*
reticulate-helpers, 40
ReticulateAnnData, *7*, *9*, *13–16*, *23*, *35*, *37*, *41*, 42, *51*

SeuratObject::CreateDimReducObject(), *25*
SeuratObject::DefaultAssay(), *19*
SeuratObject::DimReduc, *26*
SeuratObject::Key(), *26*
SeuratObject::Seurat, *14*, *19*, *20*, *38*, *39*, *44*, *46*
SingleCellExperiment::LinearEmbeddingMatrix, *20*, *28*
SingleCellExperiment::SingleCellExperiment, *14*, *19*, *38*, *39*, *44*, *46*

write_h5ad, 44
write_h5ad(), *8*, *14*
write_zarr, 46
write_zarr(), *8*, *9*, *14*

ZarrAnnData, *8*, *9*, *13–16*, *29*, *30*, *35*, *37*, *39*, *44*, 48