Package ‘zellkonverter’

October 12, 2021

Title Conversion Between scRNA-seq Objects

Version 1.2.1

Date 2021-06-22

Description Provides methods to convert between Python AnnData objects and SingleCellExperiment objects. These are primarily intended for use by downstream Bioconductor packages that wrap Python methods for single-cell data analysis. It also includes functions to read and write H5AD files used for saving AnnData objects to disk.

biocViews SingleCell, DataImport, DataRepresentation

License MIT + file LICENSE

Imports Matrix, basilisk, reticulate, SingleCellExperiment (>= 1.11.6), SummarizedExperiment, DelayedArray, methods, S4Vectors, utils

Suggests covr, spelling, testthat, knitr, rmarkdown, BiocStyle, scRNAseq, HDF5Array, rhdf5, BiocFileCache

URL `https://github.com/theislab/zellkonverter`

BugReports `https://github.com/theislab/zellkonverter/issues`

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.1.1

Language en-GB

StagedInstall no

VignetteBuilder knitr

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

git_branch RELEASE_3_13

git_last_commit a3c4f31

git_last_commit_date 2021-06-22

Date/Publication 2021-10-12

Author Luke Zappia [aut, cre] (<`https://orcid.org/0000-0001-7744-8565`>),
Aaron Lun [aut] (<`https://orcid.org/0000-0002-3564-4813`>)

Maintainer Luke Zappia <`luke@lazappi.id.au`>

R topics documented:

`zellkonverter-package` *zellkonverter: Conversion Between scRNA-seq Objects*

Description

Provides methods to convert between Python AnnData objects and SingleCellExperiment objects. These are primarily intended for use by downstream Bioconductor packages that wrap Python methods for single-cell data analysis. It also includes functions to read and write H5AD files used for saving AnnData objects to disk.

Author(s)

Maintainer: Luke Zappia <`luke@lazappi.id.au`> (ORCID)

Authors:

- Aaron Lun <`infinite.monkeys.with.keyboards@gmail.com`> (ORCID)

See Also

Useful links:

- `https://github.com/theislab/zellkonverter`
- Report bugs at `https://github.com/theislab/zellkonverter/issues`

`AnnData-Conversion` *Convert AnnData between and SingleCellExperiment*

Description

Conversion between Python AnnData objects and SingleCellExperiment objects.

Usage

```
AnnData2SCE(adata, X_name = NULL, skip_assays = FALSE, hdf5_backed = TRUE)

SCE2AnnData(sce, X_name = NULL, skip_assays = FALSE)
```

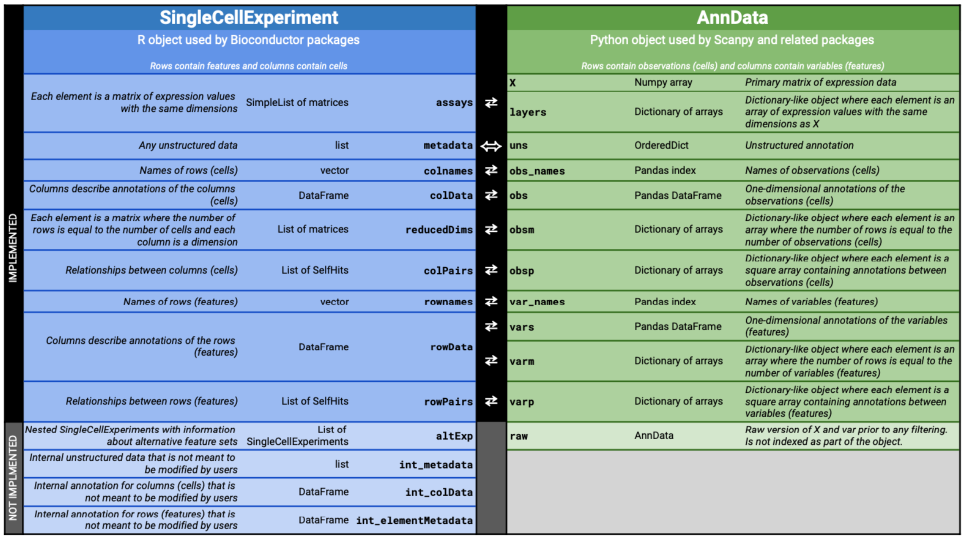
Arguments

`adata`	A **reticulate** reference to a Python AnnData object.
`X_name`	For `SCE2AnnData()` name of the assay to use as the primary matrix (`X`) of the AnnData object. If `NULL`, the first assay of `sce` will be used by default. For `AnnData2SCE()` name used when saving `X` as an assay. If `NULL` looks for an `X_name` value in `uns`, otherwise uses `"X"`.
`skip_assays`	Logical scalar indicating whether to skip conversion of any assays in `sce` or `adata`, replacing them with empty sparse matrices instead.
`hdf5_backed`	Logical scalar indicating whether HDF5-backed matrices in `adata` should be represented as HDF5Array objects. This assumes that `adata` is created with `backed="r"`.
`sce`	A SingleCellExperiment object.

Details

These functions assume that an appropriate Python environment has already been loaded. As such, they are largely intended for developer use, most typically inside a **basilisk** context.

The conversion is not entirely lossless. The current mapping is shown below (also at `https://tinyurl.com/AnnData2SCE`):



	SingleCellExperiment R object used by Bioconductor packages Rows contain features and columns contain cells				AnnData Python object used by Scanpy and related packages Rows contain observations (cells) and columns contain variables (features)		
IMPLEMENTED	*Each element is a matrix of expression values with the same dimensions*	SimpleList of matrices	**`assays`**	⇄	**`X`**	Numpy array	*Primary matrix of expression data*
					`layers`	Dictionary of arrays	*Dictionary-like object where each element is an array of expression values with the same dimensions as X*
	Any unstructured data	list	**`metadata`**	⇔	**`uns`**	OrderedDict	*Unstructured annotation*
	Names of rows (cells)	vector	**`colnames`**	⇄	**`obs_names`**	Pandas index	*Names of observations (cells)*
	Columns describe annotations of the columns (cells)	DataFrame	**`colData`**	⇄	**`obs`**	Pandas DataFrame	*One-dimensional annotations of the observations (cells)*
	Each element is a matrix where the number of rows is equal to the number of cells and each column is a dimension	List of matrices	**`reducedDims`**	⇄	**`obsm`**	Dictionary of arrays	*Dictionary-like object where each element is an array where the number of rows is equal to the number of observations (cells)*
	Relationships between columns (cells)	List of SelfHits	**`colPairs`**	⇄	**`obsp`**	Dictionary of arrays	*Dictionary-like object where each element is a square array containing annotations between observations (cells)*
	Names of rows (features)	vector	**`rownames`**	⇄	**`var_names`**	Pandas index	*Names of variables (features)*
	Columns describe annotations of the rows (features)	DataFrame	**`rowData`**	⇄	**`vars`**	Pandas DataFrame	*One-dimensional annotations of the variables (features)*
				⇄	**`varm`**	Dictionary of arrays	*Dictionary-like object where each element is an array where the number of rows is equal to the number of variables (features)*
	Relationships between rows (features)	List of SelfHits	**`rowPairs`**	⇄	**`varp`**	Dictionary of arrays	*Dictionary-like object where each element is a square array containing annotations between variables (features)*
NOT IMPLMENTED	*Nested SingleCellExperiments with information about alternative feature sets*	List of SingleCellExperiments	**`altExp`**		**`raw`**	AnnData	*Raw version of X and var prior to any filtering. Is not indexed as part of the object.*
	Internal unstructured data that is not meant to be modified by users	list	**`int_metadata`**				
	Internal annotation for columns (cells) that is not meant to be modified by users	DataFrame	**`int_colData`**				
	Internal annotation for rows (features) that is not meant to be modified by users	DataFrame	**`int_elementMetadata`**				

In `SCE2AnnData()`, matrices are converted to a **numpy**-friendly format. Sparse matrices are converted to dgCMatrix objects while all other matrices are converted into ordinary matrices. If `skip_assays = TRUE`, empty sparse matrices are created instead and the user is expected to fill in the assays on the Python side.

For `AnnData2SCE()`, a warning is raised if there is no corresponding R format for a matrix in the AnnData object, and an empty sparse matrix is created instead as a placeholder. If `skip_assays = NA`, no warning is emitted but variables are created in the `int_metadata()` of the output to specify which assays were skipped.

If `skip_assays = TRUE`, empty sparse matrices are created for all assays, regardless of whether they might be convertible to an R format or not. In both cases, the user is expected to fill in the assays on the R side, see `readH5AD()` for an example.

We attempt to convert between items in the SingleCellExperiment `metadata()` slot and the `AnnData` `uns` slot. If an item cannot be converted a warning will be raised.

Values stored in the `varm` slot of an `AnnData` object are stored in a column of `rowData()` in a SingleCellExperiment as a DataFrame of matrices. If this column is present an attempt is made to transfer this information when converting from SingleCellExperiment to `AnnData`.

Value

`AnnData2SCE()` will return a SingleCellExperiment containing the equivalent data from `adata`.

`SCE2AnnData()` will return a **reticulate** reference to an AnnData object containing the content of `sce`.

Author(s)

Luke Zappia

Aaron Lun

See Also

`writeH5AD()` and `readH5AD()` for dealing directly with H5AD files.

Examples

```
if (requireNamespace("scRNAseq", quietly = TRUE)) {
    library(basilisk)
    library(scRNAseq)
    seger <- SegerstolpePancreasData()

    # These functions are designed to be run inside
    # a specified Python environment
    roundtrip <- basiliskRun(fun = function(sce) {
        # Convert SCE to AnnData:
        adata <- zellkonverter::SCE2AnnData(sce)

        # Maybe do some work in Python on 'adata':
        # BLAH BLAH BLAH

        # Convert back to an SCE:
        zellkonverter::AnnData2SCE(adata)
    }, env = zellkonverterAnnDataEnv, sce = seger)
}
```

`AnnData-Environment` *AnnData environment*

Description

The Python environment used by **zellkonverter** for interfacing with the **anndata** Python library (and H5AD files) is described by the dependencies in `.AnnDataDependencies`. The `zellkonverterAnnDataEnv` variable is the `basilisk::BasiliskEnvironment()` used by **zellkonverter**.

Usage

```
.AnnDataDependencies

zellkonverterAnnDataEnv
```

Format

A character vector containing the pinned versions of all Python packages in `zellkonverterAnnDataEnv`.

A `basilisk::BasiliskEnvironment()` containing **zellkonverter**'s AnnData Python environment.

Details

The `.AnnDataDependencies` variable is exposed for use by other package developers who want an easy way to define the dependencies required for creating a Python environment to work with AnnData objects, most typically within a **basilisk** context. For example, we can simply combine this vector with additional dependencies to create a **basilisk** environment with Python package versions that are consistent with those in **zellkonverter**.

If you want to run code in the exact environment used by **zellkonverter** this can be done using `zellkonverterAnnDataEnv` in combination with `basilisk::basiliskStart()` and/or `basilisk::basiliskRun()`. Please refer to the **basilisk** documentation for more information on using these environments.

Author(s)

Luke Zappia

Aaron Lun

Examples

```
.AnnDataDependencies

cl <- basilisk::basiliskStart(zellkonverterAnnDataEnv)
anndata <- reticulate::import("anndata")
basilisk::basiliskStop(cl)
```

`readH5AD` *Read H5AD*

Description

Reads a H5AD file and returns a SingleCellExperiment object.

Usage

```
readH5AD(file, X_name = NULL, use_hdf5 = FALSE, reader = c("python", "R"))
```

Arguments

`file`	String containing a path to a `.h5ad` file.
`X_name`	Name used when saving `X` as an assay. If `NULL` looks for an `X_name` value in `uns`, otherwise uses `"X"`.
`use_hdf5`	Logical scalar indicating whether assays should be loaded as HDF5-based matrices from the **HDF5Array** package.
`reader`	Which HDF5 reader to use. Either `"python"` for reading with the **anndata** Python package via **reticulate** or `"R"` for **zellkonverter**'s native R reader.

Details

Setting `use_hdf5 = TRUE` allows for very large datasets to be efficiently represented on machines with little memory. However, this comes at the cost of access speed as data needs to be fetched from the HDF5 file upon request.

When first run, this function will instantiate a conda environment containing all of the necessary dependencies. This will not be performed on any subsequent run or if any other **zellkonverter** function has been run prior to this one.

Setting `reader = "R"` will use an experimental native R reader instead of reading the file into Python and converting the result. This avoids the need for a Python environment and some of the issues with conversion but is still under development and is likely to return slightly different output.

Value

A SingleCellExperiment object is returned.

Author(s)

Luke Zappia

Aaron Lun

See Also

`writeH5AD()`, to write a SingleCellExperiment object to a H5AD file.

`AnnData2SCE()`, for developers to convert existing AnnData instances to a SingleCellExperiment.

Examples

```
library(SummarizedExperiment)

file <- system.file("extdata", "krumsiek11.h5ad", package = "zellkonverter")
sce <- readH5AD(file)
class(assay(sce))

sce2 <- readH5AD(file, use_hdf5 = TRUE)
class(assay(sce2))

sce3 <- readH5AD(file, reader = "R")
```

`writeH5AD` *Write H5AD*

Description

Write a H5AD file from a SingleCellExperiment object.

Usage

```
writeH5AD(
  sce,
  file,
  X_name = NULL,
  skip_assays = FALSE,
  compression = c("none", "gzip", "lzf")
)
```

Arguments

`sce`	A SingleCellExperiment object.
`file`	String containing a path to write the new `.h5ad` file.
`X_name`	Name of the assay to use as the primary matrix (`X`) of the AnnData object. If `NULL`, the first assay of `sce` will be used by default.
`skip_assays`	Logical scalar indicating whether assay matrices should be ignored when writing to `file`.
`compression`	Type of compression when writing the new `.h5ad` file.

Details

Environment:

When first run, this function will instantiate a conda environment containing all of the necessary dependencies. This will not be performed on any subsequent run or if any other **zellkonverter** function has been run prior to this one.

Skipping assays:

Setting `skip_assays = TRUE` can occasionally be useful if the matrices in `sce` are stored in a format that is not amenable for efficient conversion to a **numpy**-compatible format. In such cases, it can be better to create an empty placeholder dataset in `file` and fill it in R afterwards.

DelayedArray assays:

If `sce` contains any **DelayedArray** matrices as assays `writeH5AD()` will write them to disk using the **rhdf5** package directly rather than via Python to avoid instantiating them in memory. However there is currently an issue which prevents this being done for sparse **DelayedArray** matrices.

Known conversion issues:

Coercion to factors:
The **anndata** package automatically converts some character vectors to factors when saving `.h5ad` files. This can effect columns of `rowData(sce)` and `colData(sce)` which may change type when the `.h5ad` file is read back into R.

Value

A `NULL` is invisibly returned.

Author(s)

Luke Zappia

Aaron Lun

See Also

`readH5AD()`, to read a SingleCellExperiment file from a H5AD file.

`SCE2AnnData()`, for developers to create an AnnData object from a SingleCellExperiment.

Examples

```
# Using the Zeisel brain dataset
if (requireNamespace("scRNAseq", quietly = TRUE)) {
    library(scRNAseq)
    sce <- ZeiselBrainData()

    # Writing to a H5AD file
    temp <- tempfile(fileext = ".h5ad")
    writeH5AD(sce, temp)
}
```

Index