anndata is a commonly used Python package for keeping track of data and learned annotations, and can be used to read from and write to the h5ad file format. It is also the main data format used in the scanpy python package (Wolf, Angerer, and Theis 2018).

However, using scanpy/anndata in R can be a major hassle. When trying to read an h5ad file, R users could approach this problem in one of two ways. A) You could read in the file manually (since it’s an H5 file), but this involves a lot of manual work and a lot of understanding on how the h5ad and H5 file formats work (also, expect major headaches from cryptic hdf5r bugs). Or B) interact with scanpy and anndata through reticulate, but run into issues converting some of the python objects into R.

We recently published anndata on CRAN, which is an R wrapper for the Python package – with some syntax sprinkled on top to make R users feel more at home.

# Installation

Installing anndata is not particularly hard. You do need R and Python to be installed. If you don’t have a working installation of Python, you can let reticulate install Miniconda.

# install the R anndata package install.packages("anndata")

# skip this if you wish to use the local python installation reticulate::install\_miniconda()

# skip this if anndata is already installed anndata::install\_anndata()

# Getting started

The API of anndata is very similar to its Python counterpart. Check out ?anndata for a full list of the functions provided by this package.

AnnData stores a data matrix X together with annotations of observations obs (obsm, obsp), variables var (varm, varp), and unstructured annotations uns.

Here is an example of how to create an AnnData object with 2 observations and 3 variables.

library(anndata) ad <- AnnData(

X = matrix(1:6, nrow = 2),

obs = data.frame(group = c("a", "b"), row.names = c("s1", "s2")), var = data.frame(type = c(1L, 2L, 3L), row.names = c("var1", "var2", "var3")),

layers = list(

spliced = matrix(4:9, nrow = 2), unspliced = matrix(8:13, nrow = 2)

),

obsm = list(

ones = matrix(rep(1L, 10), nrow = 2), rand = matrix(rnorm(6), nrow = 2),

zeros = matrix(rep(0L, 10), nrow = 2)

),

varm = list(

ones = matrix(rep(1L, 12), nrow = 3), rand = matrix(rnorm(6), nrow = 3), zeros = matrix(rep(0L, 12), nrow = 3)

),

uns = list( a = 1,

b = data.frame(i = 1:3, j = 4:6, value = runif(3)), c = list(c.a = 3, c.b = 4)

)

)

ad

## AnnData object with n\_obs × n\_vars = 2 × 3 ## obs: 'group'

## var: 'type'

## uns: 'a', 'b', 'c'

## obsm: 'ones', 'rand', 'zeros'

## varm: 'ones', 'rand', 'zeros' ## layers: 'spliced', 'unspliced'

You can read the information back out using the $ notation.

ad$X

## var1 var2 var3 ## s1 1 3 5

## s2 2 4 6

ad$obs ## group ## s1 a ## s2 b

ad$obsm[["ones"]]

|  |  |  |
| --- | --- | --- |
| ## | [,1] | [,2] [,3] [,4] [,5] |
| ## | [1,] | 1 1 1 1 1 |
| ## | [2,] | 1 1 1 1 1 |

ad$layers[["spliced"]] ## var1 var2 var3

## s1 4 6 8

## s2 5 7 9

ad$uns[["b"]] ## i j value

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ## | 1 | 1 | 4 | 0.1304560 |
| ## | 2 | 2 | 5 | 0.5043473 |
| ## | 3 | 3 | 6 | 0.3199207 |

## Using scanpy

A nice side-effect of loading this package is that it is now super easy to interact with scanpy through these AnnData objects.

library(reticulate) ad$X

## var1 var2 var3 ## s1 1 3 5

## s2 2 4 6

sc <- import("scanpy") sc$pp$normalize\_per\_cell(ad) ad$X

## var1 var2 var3

## s1 1.166667 3.5 5.833333

## s2 1.750000 3.5 5.250000

## Reading / writing AnnData objects

Read from h5ad format:

read\_h5ad("pbmc\_1k\_protein\_v3\_processed.h5ad")

## AnnData object with n\_obs × n\_vars = 713 × 33538

## var: 'gene\_ids', 'feature\_types', 'genome', 'highly\_variable', 'means', 'dispersions', 'dispersions\_norm'

## uns: 'hvgParameters', 'normalizationParameters', 'pca', 'pcaParameters'

## obsm: 'X\_pca' ## varm: 'PCs'

## Creating a view

You can use any of the regular R indexing methods to subset the AnnData object. This will result in a ‘View’ of the underlying data without needing to store the same data twice.

view <- ad[, 2] view

## View of AnnData object with n\_obs × n\_vars = 2 × 1 ## obs: 'group', 'n\_counts'

## var: 'type'

## uns: 'a', 'b', 'c'

## obsm: 'ones', 'rand', 'zeros'

## varm: 'ones', 'rand', 'zeros' ## layers: 'spliced', 'unspliced' view$is\_view

## [1] TRUE

ad[,c("var1", "var2")]

## View of AnnData object with n\_obs × n\_vars = 2 × 2 ## obs: 'group', 'n\_counts'

## var: 'type'

## uns: 'a', 'b', 'c'

## obsm: 'ones', 'rand', 'zeros'

## varm: 'ones', 'rand', 'zeros' ## layers: 'spliced', 'unspliced' ad[-1, ]

## View of AnnData object with n\_obs × n\_vars = 1 × 3 ## obs: 'group', 'n\_counts'

## var: 'type'

## uns: 'a', 'b', 'c'

## obsm: 'ones', 'rand', 'zeros'

## varm: 'ones', 'rand', 'zeros' ## layers: 'spliced', 'unspliced'

## AnnData as a matrix

The X attribute can be used as an R matrix:

ad$X[,c("var1", "var2")] ## var1 var2

## s1 1.166667 3.5

## s2 1.750000 3.5

ad$X[-1, , drop = FALSE] ## var1 var2 var3

## s2 1.75 3.5 5.25

ad$X[, 2] <- 10

You can access a different layer matrix as follows:

ad$layers["unspliced"] ## var1 var2 var3

## s1 8 10 12

## s2 9 11 13

ad$layers["unspliced"][,c("var2", "var3")] ## var2 var3

## s1 10 12

## s2 11 13

## Note on state

If you assign an AnnData object to another variable and modify either, both will be modified:

ad2 <- ad ad$X[,2] <- 10

list(ad = ad$X, ad2 = ad2$X)

|  |  |  |
| --- | --- | --- |
| ##  ## | $ad  var1 var2 var3 |  |
| ## | s1 1.166667 10 | 5.833333 |
| ## | s2 1.750000 10 | 5.250000 |
| ## |  |  |
| ## | $ad2 |  |
| ## | var1 var2 var3 |  |
| ## | s1 1.166667 10 | 5.833333 |
| ## | s2 1.750000 10 | 5.250000 |

This is standard Python behaviour but not R. In order to have two separate copies of an AnnData object, use the $copy() function:

ad3 <- ad$copy() ad$X[,2] <- c(3, 4)

list(ad = ad$X, ad3 = ad3$X) ## $ad

## var1 var2 var3

## s1 1.166667 3 5.833333

## s2 1.750000 4 5.250000

##

## $ad3

## var1 var2 var3

## s1 1.166667 10 5.833333

## s2 1.750000 10 5.250000

# Future work

In some cases, this package may still act more like a Python package rather than an R package. Some more helper functions and helper classes need to be defined in order to fully encapsulate AnnData() objects. Examples are:

ad$chunked\_X(1) ##

Following functionality has not been tested:

ad$rename\_categories(...) ad$strings\_to\_categoricals(...)