

# Package ‘BiocSklern’

February 6, 2023

**Title** interface to python sklearn via Rstudio reticulate

**Description** This package provides interfaces to selected sklearn elements, and demonstrates fault tolerant use of python modules requiring extensive iteration.

**Version** 1.20.1

**Suggests** testthat, restfulSE, HDF5Array, BiocStyle, rmarkdown, knitr

**Depends** R (>= 4.0), reticulate, methods, SummarizedExperiment

**Imports** basilisk

**License** Artistic-2.0

**LazyLoad** yes

**biocViews** StatisticalMethod, DimensionReduction, Infrastructure

**RoxygenNote** 7.2.1

**VignetteBuilder** knitr

**SystemRequirements** python (>= 2.7), sklearn, numpy, pandas, h5py

**Roxygen** list(markdown=TRUE)

**Encoding** UTF-8

**git\_url** <https://git.bioconductor.org/packages/BiocSklern>

**git\_branch** RELEASE\_3\_16

**git\_last\_commit** 7dd118f

**git\_last\_commit\_date** 2022-12-26

**Date/Publication** 2023-02-06

**Author** Vince Carey [cre, aut]

**Maintainer** Vince Carey <stvjc@channing.harvard.edu>

## R topics documented:

|                          |   |
|--------------------------|---|
| h5mat . . . . .          | 2 |
| H5matref . . . . .       | 3 |
| SkDecomp . . . . .       | 4 |
| SkDecomp-class . . . . . | 4 |

|                             |    |
|-----------------------------|----|
| skIncrPartialPCA . . . . .  | 5  |
| skIncrPCA . . . . .         | 6  |
| skIncrPCA_h5 . . . . .      | 6  |
| skIncrPPCA . . . . .        | 7  |
| skKMeans . . . . .          | 9  |
| skPartialPCA_step . . . . . | 10 |
| skPCA . . . . .             | 11 |
| skPWD . . . . .             | 11 |

|              |           |
|--------------|-----------|
| <b>Index</b> | <b>13</b> |
|--------------|-----------|

---

|       |  |
|-------|--|
| h5mat | <i>create a file connection to HDF5 matrix</i> |
|-------|--|

---

### Description

create a file connection to HDF5 matrix

### Usage

```
h5mat(infile, mode = "r", ...)
```

### Arguments

|        |   |
|--------|---|
| infile | a pathname to an HDF5 file  |
| mode   | character(1) defaults to "r", see <code>py_help</code> for <code>h5py.File</code> |
| ...    | unused  |

### Value

instance of (S3) `h5py._hl.files.File`

### Note

The result of this function must be used with `basiliskRun` with the `env` argument set to `bsklenv`, or there is a risk of inconsistent python modules being invoked. This should only be used with the persistent environment discipline of `basilisk`.

### Examples

```
if (interactive()) { # not clear why
fn = system.file("ban_6_17/assays.h5", package="BiocSklern")
proc = basilisk::basiliskStart(BiocSklern::bsklenv)
basilisk::basiliskRun(proc, function(infile, mode="r") {
  h5py = reticulate::import("h5py")
  hh = h5py$File( infile, mode=mode )
  cat("File reference:\n ")
  print(hh)
  cat("File attributes in python:\n ")
})
```

```

print(head(names(hh)))
cat("File keys in python:\n ")
print(hh$keys())
cat("HDF5 dataset in python:\n ")
print(hh['assay001'])
}, infile=fn, mode="r")
basilisk::basiliskStop(proc)
}

```

---

H5matref

*obtain an HDF5 dataset reference suitable for handling as numpy matrix*


---

### Description

obtain an HDF5 dataset reference suitable for handling as numpy matrix

### Usage

```
H5matref(filename, dsname = "assay001")
```

### Arguments

|          |   |
|----------|---|
| filename | a pathname to an HDF5 file                                  |
| dsname   | internal name of HDF5 matrix to use, defaults to 'assay001' |

### Value

instance of (S3) "h5py.\_hl.dataset.Dataset"

### Note

This should only be used with persistent environment discipline of basilisk. Additional support is planned in Bioc 3.12.

### Examples

```

fn = system.file("ban_6_17/assays.h5", package="BiocSklern")
ban = H5matref(fn)
ban
proc = basilisk::basiliskStart(BiocSklern::bsklenv)
fullpca = basilisk::basiliskRun(proc, function() {
  np = import("numpy", convert=FALSE) # ensure
  print(ban$shape)
  print(np$take(ban, 0:3, 0L))
  fullpca = skPCA(ban)
  tx = getTransformed(fullpca)
  print(dim(tx))
  fullpca
})

```

```

    })
    basilisk::basiliskStop(proc)
    # project samples
    np = reticulate::import("numpy", convert=FALSE, delay_load=TRUE)
    np$take(ban, 0:20, 0L)$shape
    st = skPartialPCA_step(np$take(ban, 0:20, 0L), n_comp=4L)
    st = skPartialPCA_step(np$take(ban, 21:40, 0L), n_comp=4L, obj=st)
    st = skPartialPCA_step(np$take(ban, 41:63, 0L), n_comp=4L, obj=st)
    oo = st$transform(ban)
    dim(oo)
    cor(oo[,1:4], getTransformed(fullpca)[,1:4])

```

---

|          |                                 |
|----------|---------------------------------|
| SkDecomp | <i>constructor for SkDecomp</i> |
|----------|---------------------------------|

---

### Description

constructor for SkDecomp

### Usage

```
SkDecomp(transform, method)
```

### Arguments

|           |   |
|-----------|---|
| transform | typically a numerical matrix representing a projection of the input |
| method    | character(1) arbitrary tag describing the decomposition             |

---

|                |   |
|----------------|---|
| SkDecomp-class | <i>container for sklearn objects and transforms</i> |
|----------------|---|

---

### Description

container for sklearn objects and transforms

### Usage

```
## S4 method for signature 'SkDecomp'
getTransformed(x)
```

### Arguments

|   |                      |
|---|----------------------|
| x | instance of SkDecomp |
|---|----------------------|

### Value

the getTransformed method returns a matrix

**Slots**

transform stored as R matrix  
method string identifying method

**Note**

In Bioc 3.11, the object slot is removed. This is a consequence of adoption of basilisk discipline for acquiring and using python resources, which greatly increases reliability, at the expense of added complication in handling python objects interactively in R. We are working on restoring this functionality but it will take time.

---

skIncrPartialPCA      *use basilisk discipline to perform partial (n\_components) incremental (chunk.size) PCA with scikit.decomposition*

---

**Description**

use basilisk discipline to perform partial (n\_components) incremental (chunk.size) PCA with scikit.decomposition

**Usage**

```
skIncrPartialPCA(mat, n_components, chunk.size = 10)
```

**Arguments**

mat                    a matrix  
n\_components        integer(1) number of PCs to compute  
chunk.size            integer(1) number of rows to use each step

**Note**

A good source for capabilities and examples is at the [sklearn doc site](#).

**Examples**

```
lk = skIncrPartialPCA(iris[,1:4], n_components=3L)  
lk  
head(getTransformed(lk))
```

---

skIncrPCA                    *use sklearn IncrementalPCA procedure*

---

### Description

use sklearn IncrementalPCA procedure

### Usage

```
skIncrPCA(mat, n_components = 2L, batch_size = 5L, ...)
```

### Arguments

|              |   |
|--------------|---|
| mat          | a matrix – can be R matrix or numpy.ndarray, if the latter it must be set up in a basilisk persistent environment, and that is not currently demonstrated for this package. |
| n_components | number of PCA to retrieve   |
| batch_size   | number of records to use at each iteration  |
| ...          | passed to python IncrementalPCA   |

### Value

matrix with rotation

### Examples

```
dem = skIncrPCA(iris[,1:4], batch_size=25L)
dem2 = skIncrPCA(iris[,1:4], batch_size=25L, n_components=2L)
dem
dem2
```

---

skIncrPCA\_h5                    *demo of HDF5 processing with incremental PCA/batch\_size/fit\_transform*

---

### Description

demo of HDF5 processing with incremental PCA/batch\_size/fit\_transform

### Usage

```
skIncrPCA_h5(fn, dsname = "assay001", n_components, chunk.size = 10L)
```

**Arguments**

|              |  |
|--------------|--|
| fn           | character(1) path to HDF5 file   |
| dsname       | character(1) name of dataset within HDF5 file, assumed to be 2-dimensional array |
| n_components | numeric(1) passed to IncrementalPCA  |
| chunk.size   | numeric(1) passed to IncrementalPCA as batch_size                                |

**Note**

Here we use `IncrementalPCA$fit_transform` and let python take care of chunk retrieval. `skIncrPartialPCA` acquires chunks from R matrix and uses `IncrementalPCA$partial_fit`.

**Examples**

```
if (interactive()) {  
  fn = system.file("hdf5/irmatt.h5", package="BiocSkllearn") # 'transposed' relative to R iris  
  dem = skIncrPCA_h5(fn, n_components=3L, dsname="tquants")  
  dem  
  head(getTransformed(dem))  
}
```

---

|            |  |
|------------|--|
| skIncrPPCA | <i>optionally fault tolerant incremental partial PCA for projection of samples from SummarizedExperiment</i> |
|------------|--|

---

**Description**

optionally fault tolerant incremental partial PCA for projection of samples from SummarizedExperiment

**Usage**

```
skIncrPPCA(  
  se,  
  chunksize,  
  n_components,  
  assayind = 1,  
  picklePath = "./skIdump.pkl",  
  matTx = force,  
  ...  
)
```

**Arguments**

|                           |   |
|---------------------------|---|
| <code>se</code>           | instance of SummarizedExperiment  |
| <code>chunksize</code>    | integer number of samples per step  |
| <code>n_components</code> | integer number of PCs to compute  |
| <code>assayind</code>     | not used, assumed set to 1  |
| <code>picklePath</code>   | if non-null, incremental results saved here via <code>joblib.dump</code> , for each chunk. If NULL, no saving of incremental results.                       |
| <code>matTx</code>        | a function defaulting to <code>force()</code> that accepts a matrix and returns a matrix with identical dimensions, e.g., <code>function(x) log(x+1)</code> |
| <code>...</code>          | not used  |

**Value**

python instance of `sklearn.decomposition.incremental_pca.IncrementalPCA`

**Note**

Will treat samples as records and all features (rows) as attributes, projecting. to an `n_components`-dimensional space. Method will acquire chunk of assay data and transpose before computing PCA contributions. In case of crash, restore from `picklePath` using `joblib$load` after loading `reticulate`. You can use the `n_samples_seen_` component of the restored python reference to determine where to restart. You can manage resumption using `skPartialPCA_step`.

**Examples**

```
# demo SE made with TENxGenomics:
# mm = matrixSummarizedExperiment(h5path, 1:27998, 1:750)
# saveHDF5SummarizedExperiment(mm, "tenx_750")
#
if (FALSE) {
if (requireNamespace("HDF5Array")) {
  se750 = HDF5Array::loadHDF5SummarizedExperiment(
    system.file("hdf5/tenx_750", package="BiocSklern"))
  lit = skIncrPPCA(se750[, 1:50], chunksize=5, n_components=4)
  round(cor(pypc <- lit$transform(dat <- t(as.matrix(assay(se750[,1:50]))))),3)
  rpc = prcomp(dat)
  round(cor(rpc$x[,1:4], pypc), 3)
} # this has to be made basilisk-compliant
} # and is blocked until then
```



---

skKMeans                      *interface to sklearn.cluster.KMeans using basilisk discipline*

---

## Description

interface to sklearn.cluster.KMeans using basilisk discipline

## Usage

```
skKMeans(mat, ...)
```

## Arguments

|     |  |
|-----|--|
| mat | a matrix-like datum or reference to such |
| ... | arguments to sklearn.cluster.KMeans      |

## Value

a list with cluster assignments (integers starting with zero) and asserted cluster centers.

## Note

This is a demonstrative interface to the resources of sklearn.cluster. In this particular interface, we are using sklearn.cluster.k\_means\_.KMeans. There are many other possibilities in sklearn.cluster: *\_dbscan\_inner*, *feature\_agglomeration*, *hierarchical*, *k\_means*, *k\_means\_elkan*, *affinity\_propagation*, *bicluster*, *birch*, *dbscan*, *hierarchical*, *k\_means*, *mean\_shift*, *setup*, *spectral*.

Basilisk discipline has not been used for this function, 1 June 2022.

## Examples

```
irloc = system.file("csv/iris.csv", package="BiocSklern")
np = reticulate::import("numpy", delay_load=TRUE, convert=FALSE)
h5py = reticulate::import("h5py", delay_load=TRUE)
irismat = np$genfromtxt(irloc, delimiter=',')
ans = skKMeans(irismat, n_clusters=2L)
names(ans) # names of available result components
table(iris$Species, ans$labels)
# now use an HDF5 reference
irh5 = system.file("hdf5/irmat.h5", package="BiocSklern")
fref = h5py$File(irh5)
ds = fref$`__getitem__`("quants")
ans2 = skKMeans(np$array(ds)$T, n_clusters=2L) # HDF5 matrix is transposed relative to python array layout! Is the
table(ans$labels, ans2$labels)
ans3 = skKMeans(np$array(ds)$T,
  n_clusters=8L, max_iter=200L,
  algorithm="full", random_state=20L)
dem = skKMeans(iris[,1:4], n_clusters=3L, max_iter=100L, algorithm="full",
  random_state=20L)
```

```

str(dem)
tab = table(iris$Species, dem$labels)
tab
plot(iris[,1], iris[,3], col=as.numeric(factor(iris$Species)))
points(dem$centers[,1], dem$centers[,3], pch=19, col=apply(tab,2,which.max))

```

---

skPartialPCA\_step      *take a step in sklearn IncrementalPCA partial fit procedure*

---

### Description

take a step in sklearn IncrementalPCA partial fit procedure

### Usage

```
skPartialPCA_step(mat, n_components, obj)
```

### Arguments

|              |   |
|--------------|---|
| mat          | a matrix – can be R matrix or numpy.ndarray   |
| n_components | number of PCA to retrieve                     |
| obj          | sklearn.decomposition.IncrementalPCA instance |

### Value

trained IncrementalPCA reference, to which 'transform' method can be applied to obtain projection for any compliant input

### Note

if obj is missing, the process is initialized with the matrix provided

### Examples

```

# these steps are not basilisk-compliant, you need to acquire references
irloc = system.file("csv/iris.csv", package="BiocSklern")
np = reticulate::import("numpy", delay_load=TRUE, convert=FALSE)
irismat = np$genfromtxt(irloc, delimiter=',')
ta = np$take
ipc = skPartialPCA_step(ta(irismat,0:49,0L))
ipc = skPartialPCA_step(ta(irismat,50:99,0L), obj=ipc)
ipc = skPartialPCA_step(ta(irismat,100:149,0L), obj=ipc)
head(names(ipc))
ipc$transform(ta(irismat,0:5,0L))
fullproj = ipc$transform(irismat)
fullpc = prcomp(data.matrix(iris[,1:4]))$x
round(cor(fullpc,fullproj),3)

```

---

|       |                                  |
|-------|----------------------------------|
| skPCA | <i>use sklearn PCA procedure</i> |
|-------|----------------------------------|

---

**Description**

use sklearn PCA procedure

**Usage**

```
skPCA(mat, ...)
```

**Arguments**

|     |  |
|-----|--|
| mat | a matrix – can be R matrix or numpy.ndarray  |
| ... | additional parameters passed to sklearn.decomposition.PCA, for additional information use <code>py_help()</code> on a reticulate-imported <code>sklearn.decomposition.PCA</code> instance. |

**Value**

matrix with rotation

**Note**

If no additional arguments are passed, all defaults are used.

**Examples**

```
#irloc = system.file("csv/iris.csv", package="BiocSklern")
#irismat = SklearnEls()$np$genfromtxt(irloc, delimiter=',')
#skpi = skPCA(irismat)
#getTransformed(skpi)[1:5,]
chk = skPCA(data.matrix(iris[,1:4]))
chk
head(getTransformed(chk))
head(prcomp(data.matrix(iris[,1:4]))$x)
```

---

|       |   |
|-------|---|
| skPWD | <i>use sklearn pairwise_distances procedure</i> |
|-------|---|

---

**Description**

use sklearn pairwise\_distances procedure

**Usage**

```
skPWD(mat, ...)
```

**Arguments**

`mat` a matrix – can be R matrix or `numpy.ndarray`  
`...` additional parameters passed to `sklearn.metrics.pairwise_distances`, for additional information use `py_help()` on a reticulate-imported `sklearn.metrics.pairwise_distances` instance.

**Value**

matrix with rotation

**Note**

If no additional arguments are passed, all defaults are used.

**Examples**

```
irloc = system.file("csv/iris.csv", package="BiocSklern")
data(iris)
irismat = as.matrix(iris[,1:4])
chk1 = skPWD(irismat)
chk1[1:4,1:5]
chk2 = skPWD(irismat, metric='manhattan')
chk2[1:4,1:5]
```

# Index

`getTransformed (SkDecomp-class)`, [4](#)  
`getTransformed, SkDecomp-method`  
    (`SkDecomp-class`), [4](#)

`h5mat`, [2](#)  
`H5matref`, [3](#)

`SkDecomp`, [4](#)  
`SkDecomp-class`, [4](#)  
`skIncrPartialPCA`, [5](#)  
`skIncrPCA`, [6](#)  
`skIncrPCA_h5`, [6](#)  
`skIncrPPCA`, [7](#)  
`skIncrPPCA, SummarizedExperiment-method`  
    (`skIncrPPCA`), [7](#)

`skKMeans`, [9](#)  
`skPartialPCA_step`, [10](#)  
`skPCA`, [11](#)  
`skPWD`, [11](#)