

# Package ‘BiocSklern’

April 11, 2018

**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** 1.0.2

**Author** Vince Carey

**Suggests** testthat, restfulSE

**Depends** R (>= 3.4.0), reticulate, methods, SummarizedExperiment, knitr

**Imports** BBmisc

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

**License** Artistic-2.0

**LazyLoad** yes

**biocViews** StatisticalMethod, DimensionReduction, Infrastructure

**RoxygenNote** 6.0.1.9000

**VignetteBuilder** knitr

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

**NeedsCompilation** no

## R topics documented:

h5mat . . . . .	2
H5matref . . . . .	2
SkDecomp-class . . . . .	3
skIncrPCA . . . . .	3
skIncrPPCA . . . . .	4
SklearnEls . . . . .	5
skPartialPCA_step . . . . .	5
skPCA . . . . .	6
<b>Index</b>	<b>7</b>

---

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

---

**Description**

create a file connection to HDF5 matrix

**Usage**

```
h5mat(file, dsname = "assay001")
```

**Arguments**

file	a pathname to an HDF5 file
dsname	internal name of HDF5 matrix to use

**Examples**

```
fn = system.file("ban_6_17/assays.h5", package="BiocSklern")
h5mat(fn)
```

---

H5matref	<i>obtain an HDF5 dataset reference suitable for handling as numpy matrix</i>
----------	---

---

**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'

**Examples**

```
fn = system.file("ban_6_17/assays.h5", package="BiocSklern")
ban = H5matref(fn)
ban
np = import("numpy", convert=FALSE) # ensure
ban$shape
np$take(ban, 0:3, 0L)
fullpca = skPCA(ban)
dim(getTransformed(fullpca))
ta = np$take
# project samples
ta(ban, 0:20, 0L)$shape
```

```

st = skPartialPCA_step(ta(ban, 0:20, 0L))
st = skPartialPCA_step(ta(ban, 21:40, 0L), obj=st)
st = skPartialPCA_step(ta(ban, 41:63, 0L), obj=st)
oo = st$transform(ban)
dim(oo)
cor(oo[,1:4], getTransformed(fullpca)[,1:4])

```

---

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)

## S4 method for signature 'SkDecomp'
pyobj(x)

```

### Arguments

x                    instance of SkDecomp

### Slots

transform stored as R matrix  
method string identifying method  
object reference to the python object with decomposition components

---

skIncrPCA	<i>use sklearn IncrementalPCA procedure</i>
-----------	---

---

### Description

use sklearn IncrementalPCA procedure

### Usage

```
skIncrPCA(mat, n_components, batch_size)
```

### Arguments

mat                    a matrix – can be R matrix or numpy.ndarray  
n\_components          number of PCA to retrieve  
batch\_size            number of records to use at each iteration

**Value**

matrix with rotation

**Examples**

```
irloc = system.file("csv/iris.csv", package="BiocSklern")
irismat = SklearnEls()$np$genfromtxt(irloc, delimiter=',')
ski = skIncrPPCA(irismat)
ski25 = skIncrPPCA(irismat, batch_size=25L) # non-default
getTransformed(ski)[1:3,]
getTransformed(ski25)[1:3,]
```

---

skIncrPPCA

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

---

**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**

se	instance of SummarizedExperiment
chunksize	integer number of samples per step
n_components	integer number of PCs to compute
assayind	not used, assumed set to 1
picklePath	if non-null, incremental results saved here via sklearn.externals.joblib.dump, for each chunk. If NULL, no saving of incremental results.
matTx	a function defaulting to force() that accepts a matrix and returns a matrix with identical dimensions, e.g., function(x) log(x+1)
...	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 SklearnEls()\$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, 1:1500)
# saveHDF5SummarizedExperiment(mm, "tenx_1500")
#
se1500 = loadHDF5SummarizedExperiment(
  system.file("hdf5/tenx_1500", package="BiocSklearn"))
lit = skIncrPPCA(se1500[, 1:50], chunksize=5, n_components=4)
round(cor(pypc <- lit$transform(dat <- t(as.matrix(assay(se1500[,1:50]))))),3)
rpc = prcomp(dat)
round(cor(rpc$x[,1:4], pypc), 3)
```

SklearnEls

*mediate access to python modules from sklearn.decomposition***Description**

mediate access to python modules from sklearn.decomposition

**Usage**

```
SklearnEls()
```

**Note**

Returns a list with elements np (numpy), pd (pandas), h5py (h5py), skd (sklearn.decomposition), joblib (sklearn.externals.joblib), each referring to python modules.

**Examples**

```
els = SklearnEls()
names(els$skd) # slow at first
# try py_help(els$skd$PCA) # etc.
```

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**

```
irloc = system.file("csv/iris.csv", package="BiocSklern")
irismat = SklearnEls()$np$genfromtxt(irloc, delimiter=',')
ta = SklearnEls()$np$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

*use sklearn PCA procedure*


---

**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(SklearnEls())\$sk\$PCA</code>

**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,]
```

# Index

`getTransformed (SkDecomp-class)`, [3](#)  
`getTransformed, SkDecomp-method`  
    (`SkDecomp-class`), [3](#)

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

`pyobj (SkDecomp-class)`, [3](#)  
`pyobj, SkDecomp-method (SkDecomp-class)`,  
    [3](#)

`SkDecomp-class`, [3](#)  
`skIncrPCA`, [3](#)  
`skIncrPPCA`, [4](#)  
`skIncrPPCA, SummarizedExperiment-method`  
    (`skIncrPPCA`), [4](#)  
`SklearnEls`, [5](#)  
`skPartialPCA_step`, [5](#)  
`skPCA`, [6](#)