Package ‘TENxIO’

April 3, 2025

Type Package

Title Import methods for 10X Genomics files

Version 1.9.3

Depends R (>= 4.2.0), SingleCellExperiment, SummarizedExperiment

Imports BiocBaseUtils, BiocGenerics, BiocIO, GenomeInfoDb, GenomicRanges, HDF5Array, Matrix, MatrixGenerics, methods, RCurl, readr, rhdf5, R.utils, S4Vectors, utils

Suggests BiocStyle, DropletTestFiles, ExperimentHub, knitr, RaggedExperiment (>= 1.29.5), rmarkdown, Rsamtools, tinytest

Description Provides a structured S4 approach to importing data files from the 10X pipelines. It mainly supports Single Cell Multiome ATAC + Gene Expression data among other data types. The main Bioconductor data representations used are SingleCellExperiment and RaggedExperiment.

biocViews Software, Infrastructure, DataImport, SingleCell

VignetteBuilder knitr

License Artistic-2.0

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

BugReports `https://github.com/waldronlab/TENxIO/issues`

URL `https://github.com/waldronlab/TENxIO`

Collate 'TENxFile-class.R' 'TENxFileList-class.R' 'TENxFragments-class.R' 'TENxH5-class.R' 'TENxIO-package.R' 'TENxMTX-class.R' 'TENxPeaks-class.R' 'TENxTSV-class.R' 'utils.R'

Date 2024-11-13

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

git_branch devel

git_last_commit c6c5a73

git_last_commit_date 2024-11-21

Repository Bioconductor 3.21

Date/Publication 2025-04-02

Author Marcel Ramos [aut, cre] (ORCID:
<https://orcid.org/0000-0002-3242-0582>)

Maintainer Marcel Ramos <`marcel.ramos@sph.cuny.edu`>

Contents

`TENxFile` *TENxFile constructor function*

Description

The `TENxFile` constructor function serves as the auto-recognizer function for 10X files. It can import several different file extensions, namely:

```
* H5 - on-disk HDF5
* MTX - matrix market
* .tar.gz - compressed tarball
```

Usage

```
TENxFile(resource, extension, ...)
```

Arguments

`resource`	character(1) The path to the file
`extension`	character(1) The file extension for the given resource. It can usually be obtained from the file path. An override can be provided especially for `ExperimentHub` resources where the file extension is removed.
`...`	Additional inputs to the low level class generator functions

Details

Note that the example below includes the use of a large ~ 4 GB `ExperimentHub` resource obtained from the 10X website.

Value

A subclass of `TENxFile` according to the input file extension

Examples

```
if (interactive()) {

    ## from ExperimentHub
    hub <- ExperimentHub::ExperimentHub()
    fname <- hub[["EH1039"]]
    TENxFile(fname, extension = "h5", group = "mm10", version = "2")
    TENxFile(fname, extension = "h5", group = "mm10", version = "2") |>
        metadata()

}
```

`TENxFile-class` *TENxFile: General purpose class for 10X files*

Description

The `TENxFile` class is the default representation for unrecognized subclasses. It inherits from the BiocFile class and adds a few additional slots. The constructor function can handle typical 10X file types. For more details, see the constructor function documentation.

Usage

```
## S4 method for signature 'TENxFile'
metadata(x, ...)
```

Arguments

`x`	An object of class `TENxFile`, `TENxFileList`, `TENxMTX`, `TENxH5`, `TENxPeaks`, `TENxTSV`, or derivatives
`...`	Additional arguments (not used)

Value

A list of metadata for the given object

Functions

- `metadata(TENxFile)`: `metadata` method for `TENxFile` objects

Slots

`extension` character(1) The file extension as extracted from the file path or overridden via the `ext` argument in the constructor function.

`colidx` integer(1) The column index corresponding to the columns in the file that will subsequently be imported

`rowidx` integer(1) The row index corresponding to rows in the file that will subsequently be imported

`remote` logical(1) Whether the file exists on the web, i.e., the `resource` is a URL

`compressed` logical(1) Whether the file is compressed with, e.g., `.gz`

`TENxFileList` *TENxFileList: Represent groups of files from 10X Genomic*

Description

This constructor function is meant to handle `.tar.gz` tarball files from 10X Genomics.

Usage

```
TENxFileList(..., version, compressed = FALSE)
```

Arguments

`...`	Typically, a file path to a tarball archive. Can be named arguments corresponding to file paths, or a named list of file paths.
`version`	character(1) The version in the tarball. See details.
`compressed`	logical(1) Whether or not the file provided is compressed, usually as `tar.gz` (default FALSE)

Details

These tarballs usually contain three files:

1. `matrix.mtx.gz` - the counts matrix
2. `features.tsv.gz` - row metadata usually represented as `rowData`
3. `barcodes.tsv.gz` - column names corresponding to cell barcode identifiers If all above files are in the tarball, the import method will provide a `SingleCellExperiment`. Otherwise, a simple list of imported data is given. Note that version "3" uses 'features.tsv.gz' and version "2" uses 'genes.tsv.gz'. If known, indicate the `version` argument in the `TENxFileList` constructor function.

Value

Either a `SingleCellExperiment` or a list of imported data

Examples

```
fl <- system.file(
    "extdata", "pbmc_granulocyte_sorted_3k_ff_bc_ex_matrix.tar.gz",
    package = "TENxIO", mustWork = TRUE
)

## Method 1 (tarball)
TENxFileList(fl)

## metadata before import
metadata(TENxFileList(fl))

## import() method
import(TENxFileList(fl))

## metadata after import
import(TENxFileList(fl)) |>
    metadata()

## untar to simulate folder output
dir.create(tdir <- tempfile())
untar(fl, exdir = tdir)

## Method 2 (folder)
TENxFileList(tdir)
import(TENxFileList(tdir))

## Method 3 (list of TENxFile objects)
files <- list.files(tdir, recursive = TRUE, full.names = TRUE)
names(files) <- basename(files)
filelist <- lapply(files, TENxFile)

TENxFileList(filelist, compressed = FALSE)

## Method 4 (SimpleList)
TENxFileList(as(filelist, "SimpleList"), compressed = FALSE)

## Method 5 (named arguments)
TENxFileList(
    barcodes.tsv.gz = TENxFile(files[1]),
    features.tsv.gz = TENxFile(files[2]),
    matrix.mtx.gz = TENxFile(files[3])
)

unlink(tdir, recursive = TRUE)
```

`TENxFileList-class` *TENxFileList: A list-like representation for TENxFiles*

Description

This class was designed to mainly handle tarballs from 10X Genomics. The typical file extension for these tarballs is `.tar.gz`.

Usage

```
## S4 method for signature 'TENxFileList'
path(object, ...)

## S4 method for signature 'TENxFileList'
decompress(manager, con, ...)

## S4 method for signature 'TENxFileList,ANY,ANY'
import(con, format, text, ...)

## S4 method for signature 'TENxFileList'
metadata(x, ...)
```

Arguments

`object`	An object containing paths. Even though it will typically contain a single path, `object` can actually contain an arbitrary number of paths.
`...`	Additional arguments (not used)
`manager`	A `ConnectionManager` internal instance; currently not used.
`con`	The connection from which data is loaded or to which data is saved. If this is a `character` vector, it is assumed to be a file name and a corresponding file connection is created and then closed after exporting the object. If it is a BiocFile derivative, the data is loaded from or saved to the underlying resource. If missing, the function will return the output as a character vector, rather than writing to a connection.
`format`	The format of the output. If missing and con is a file name, the format is derived from the file extension. This argument is unnecessary when con is a derivative of BiocFile.
`text`	If con is missing, this can be a character vector directly providing the string data to import.
`x`	An object of class `TENxFile`, `TENxFileList`, `TENxMTX`, `TENxH5`, `TENxPeaks`, `TENxTSV`, or derivatives

Details

These tarballs usually contain three files:

1. `matrix.mtx.gz` - the counts matrix
2. `features.tsv.gz` - row metadata usually represented as `rowData`
3. `barcodes.tsv.gz` - column names corresponding to cell barcode identifiers Note that version '2' includes `genes.tsv.gz` instead of `features.tsv.gz` in version '3'.

An additional `ref` argument can be provided when the file contains multiple `feature_type` in the file or "Type" in the `rowData`. By default, the first type reported in `table()` is set as the `mainExpName` in the `SingleCellExperiment` object.

Value

A `TENxFileList` class object

Functions

- `path(TENxFileList)`: Obtain file paths for all files in the object as a vector
- `decompress(TENxFileList)`: An intermediate method for decompressing (via untar) the contents of a `.tar.gz` file list
- `import(con = TENxFileList, format = ANY, text = ANY)`: Recursively import files within a `TENxFileList`
- `metadata(TENxFileList)`: `metadata` method for `TENxFileList` objects

Slots

`listData`	list() The data in list format
`extension`	character() A vector of file extensions for each file
`compressed`	logical(1) Whether the file is compressed as `.tar.gz`
`version`	character(1) The version number of the tarball usually either '2' or '3'

`TENxFragments` *TENxFragments: Import fragments files from 10X*

Description

TENxFragments: Import fragments files from 10X

Usage

```
TENxFragments(resource, yieldSize = 200, which = GRanges(), ...)
```

Arguments

`resource`	character(1) The file path to the fragments resource, usually a compressed tabix file with extension `.tsv.gz`.
`yieldSize`	numeric() The number of records to read by default, 200 records will be imported. A warning will be emitted if not modified.
`which`	GRanges() A GRanges indicating the regions of interest. This get sent to `RSamtools` as the `param` input.
`...`	Further arguments to the class generator function (currently not used)

Value

A `RaggedExperiment` object class

Examples

```
fr <- system.file(
    "extdata", "pbmc_3k_atac_ex_fragments.tsv.gz",
    package = "TENxIO", mustWork = TRUE
)

tfr <- TENxFragments(fr)

fra <- import(tfr)
```

`TENxFragments-class` *TENxFragments: A class to represent fragments data as* `GRanges`

Description

This class is designed to work mainly with `fragments.tsv.gz` files from 10x pipelines.

Usage

```
## S4 method for signature 'TENxFragments,ANY,ANY'
import(con, format, text, ...)
```

Arguments

`con`	The connection from which data is loaded or to which data is saved. If this is a `character` vector, it is assumed to be a file name and a corresponding file connection is created and then closed after exporting the object. If it is a BiocFile derivative, the data is loaded from or saved to the underlying resource. If missing, the function will return the output as a character vector, rather than writing to a connection.

`format`	The format of the output. If missing and `con` is a file name, the format is derived from the file extension. This argument is unnecessary when `con` is a derivative of BiocFile.
`text`	If `con` is missing, this can be a character vector directly providing the string data to import.
`...`	Parameters to pass to the format-specific method.

Details

Fragments data from 10x can be quite large. In order to speed up the initial exploration of the data, we use a default of **200** records for loading. Users can change this default value by specifying a new one via the `yieldSize` argument in the constructor function.

Value

A `TENxFragments` class object

Methods (by generic)

- `import(con = TENxFragments, format = ANY, text = ANY)`: Import method for representing fragments.tsv.gz data from 10x via `Rsamtools` and `RaggedExperiment`

Slots

`which` GRanges() A GRanges indicating the regions of interest. This get sent to `RSamtools` as the `param` input.

`yieldSize` numeric() The number of records to read by default, 200 records will be imported. A warning will be emitted if not modified.

`TENxH5` *TENxH5: Represent H5 files from 10X*

Description

This constructor function was developed using the PBMC 3K dataset from 10X Genomics (version 3). Other versions are supported and input arguments `version` and `group` can be overridden.

Usage

```
TENxH5(resource, version, group, ranges, rowidx, colidx, ...)
```

Arguments

`resource`	character(1) The path to the file
`version`	character(1) There are currently two recognized versions associated with 10X data, either version "2" or "3". See details for more information.
`group`	character(1) The HDF5 group embedded within the file structure, this is usually either the "matrix" or "outs" group but other groups are supported as well (e.g., "mm10").
`ranges`	character(1) The HDF5 internal folder location embedded within the file that points to the ranged data information, e.g., "/features/interval". Set to `NA_character_` if range information is not present.
`rowidx, colidx`	numeric() A vector of indices corresponding to either rows or columns that will dictate the data imported from the file. The indices will be passed on to the `[` method of the `TENxMatrix` representation.
`...`	Additional inputs to the low level class generator functions

Details

The various `TENxH5` methods including `rowData` and `rowRanges`, provide a snapshot of the data using a length 12 head and tail subset for efficiency. In contrast, methods such as `dimnames` and `dim` give a full view of the dimensions of the data. The `show` method provides relevant information regarding the dimensions of the data including metadata such as `rowData` and "Type" column, if available. The term "projection" refers to the data class that will be provided once the data file is `imported`.

An additional `ref` argument can be provided when the file contains multiple `feature_type` in the file or "Type" in the `rowData`. By default, the first type reported in `table()` is set as the `mainExpName` in the `SingleCellExperiment` object.

For data that do not contain genomic coordinate information, the `TENxH5` will fail to read `"/features/interval"` and will set the `ranges` argument to `NA_character_`.

The data version "3" mainly includes a "matrix" group and "interval" information within the file. Version "2" data does not include ranged-based information and has a different directory structure compared to version "3". See the internal `data.frame`: `TENxIO:::h5.version.map` for a map of fields and their corresponding file locations within the H5 file. This map is used to create the `rowData` structure from the file.

Value

Usually, a `SingleCellExperiment` instance

See Also

`import` section in TENxH5

Examples

```
h5f <- system.file(
    "extdata", "pbmc_granulocyte_ff_bc_ex.h5",
```

```
    package = "TENxIO", mustWork = TRUE
)

TENxH5(h5f)

import(TENxH5(h5f))

h5f <- system.file(
    "extdata", "10k_pbmc_ATACv2_f_bc_ex.h5",
    package = "TENxIO", mustWork = TRUE
)

## Optional ref input, most frequent Type used by default
th5 <- TENxH5(h5f, ranges = "/features/id", ref = "Peaks")
th5
TENxH5(h5f, ranges = "/features/id")
import(th5)
```

`TENxH5-class` *TENxH5: The HDF5 file representation class for 10X Data*

Description

This class is designed to work with 10x Single Cell datasets. It was developed using the PBMC 3k 10X dataset from the CellRanger v2 pipeline.

Usage

```
## S4 method for signature 'TENxH5'
rowData(x, use.names = TRUE, ...)

## S4 method for signature 'TENxH5'
dim(x)

## S4 method for signature 'TENxH5'
dimnames(x)

## S4 method for signature 'TENxH5'
genome(x)

## S4 method for signature 'TENxH5'
rowRanges(x, ...)

## S4 method for signature 'TENxH5,ANY,ANY'
import(con, format, text, ...)

## S4 method for signature 'TENxH5'
show(object)
```

Arguments

`x`	A `TENxH5` object
`use.names`	For `rowData`: Like `mcols(x)`, by default `rowData(x)` propagates the rownames of `x` to the returned DataFrame object (note that for a SummarizedExperiment object or derivative, the rownames are also the names i.e. `rownames(x)` is always the same as `names(x)`). Setting `use.names=FALSE` suppresses this propagation i.e. it returns a DataFrame object with no rownames. Use this when `rowData(x)` fails, which can happen when the rownames contain NAs (because the rownames of a SummarizedExperiment object or derivative can contain NAs, but the rownames of a DataFrame object cannot). For `combineRows` and `combineCols`: See Combining section below.
`...`	For `assay`, arguments in `...` are forwarded to `assays`. For `rbind`, `cbind`, `...` contains SummarizedExperiment objects (or derivatives) to be combined. For other accessors, ignored.
`con`	The connection from which data is loaded or to which data is saved. If this is a `character` vector, it is assumed to be a file name and a corresponding file connection is created and then closed after exporting the object. If it is a BiocFile derivative, the data is loaded from or saved to the underlying resource. If missing, the function will return the output as a character vector, rather than writing to a connection.
`format`	The format of the output. If missing and `con` is a file name, the format is derived from the file extension. This argument is unnecessary when `con` is a derivative of BiocFile.
`text`	If `con` is missing, this can be a character vector directly providing the string data to import.
`object`	A `TENxH5` class object

Details

The data version "3" mainly includes a "matrix" group and "interval" information within the file. Version "2" data does not include ranged-based information and has a different directory structure compared to version "3". See the internal `data.frame`: `TENxIO:::h5.version.map` for a map of fields and their corresponding file locations within the H5 file. This map is used to create the `rowData` structure from the file.

Value

A `TENxH5` class object

Methods (by generic)

- `rowData(TENxH5)`: Generate the rowData ad hoc from a TENxH5 file
- `dim(TENxH5)`: Get the dimensions of the data as stored in the file
- `dimnames(TENxH5)`: Get the dimension names from the file

- `genome(TENxH5)`: Read genome string from file
- `rowRanges(TENxH5)`: Read interval data and represent as GRanges
- `import(con = TENxH5, format = ANY, text = ANY)`: Import TENxH5 data as a SingleCellExperiment; see section below
- `show(TENxH5)`: Display a snapshot of the contents within a TENxH5 file before import

Slots

`version` character(1) There are currently two recognized versions associated with 10X data, either version "2" or "3". See details for more information.

`group` character(1) The HDF5 group embedded within the file structure, this is usually either the "matrix" or "outs" group but other groups are supported as well.

`ranges` character(1) The HDF5 internal folder location embedded within the file that points to the ranged data information, e.g., "/features/interval".

import

The `import` method uses `DelayedArray::TENxMatrix` to represent matrix data. Generally, version 3 datasets contain associated genomic coordinates. The associated feature data, as displayed by the `rowData` method, is queried for the "Type" column which will indicate that a `splitAltExps` operation is appropriate. If a `ref` input is provided to the constructor function `TENxH5`, it will be used as the main experiment; otherwise, the most frequent category in the "Type" column will be used. For example, the Multiome ATAC + Gene Expression feature data contains both 'Gene Expression' and 'Peaks' labels in the "Type" column.

See Also

TENxH5

`TENxIO` *TENxIO: A Bioconductor package for importing 10X Genomics files*

Description

The package provides file classes based on `BiocIO` for common file extensions found in the 10X Genomics website.

Supported file types

Here is a table of supported file and file extensions and their imported classes:

Extension	Class	Imported as
.h5	TENxH5	SingleCellExperiment w/ TENxMatrix
.mtx / .mtx.gz	TENxMTX	SummarizedExperiment w/ dgCMatrix
.tar.gz	TENxFileList	SingleCellExperiment w/ dgCMatrix
peak_annotation.tsv	TENxPeaks	GRanges

fragments.tsv.gz	TENxFragments	RaggedExperiment
.tsv / .tsv.gz	TENxTSV	tibble

Author(s)

Maintainer: Marcel Ramos <`marcel.ramos@sph.cuny.edu`> (ORCID)

See Also

Useful links:

- `https://github.com/waldronlab/TENxIO`
- Report bugs at `https://github.com/waldronlab/TENxIO/issues`

`TENxMTX` *TENxMTX: Represent Matrix Market Format Files from 10X*

Description

This constructor function accepts `.mtx` and `.mtx.gz` compressed formats for eventual importing. It is mainly used with tarball files from 10X Genomics, where more annotation data is included. Importing solely the `.mtx` format will provide users with a `SummarizedExperiment` with an assay of class `dgCMatrix` from the `Matrix` package. Currently, other formats are not supported but if you'd like to request support for a format, please open an issue on GitHub.

Usage

```
TENxMTX(resource, compressed = FALSE, ...)
```

Arguments

`resource`	character(1) The path to the file
`compressed`	logical(1) Whether the resource file is compressed (default FALSE)
`...`	Additional inputs to the low level class generator functions

Value

A `SummarizedExperiment` instance with a `dgCMatrix` in the assay

Examples

```
mtxf <- system.file(
    "extdata", "pbmc_3k_ff_bc_ex.mtx",
    package = "TENxIO", mustWork = TRUE
)

con <- TENxMTX(mtxf)

import(con)
```

`TENxMTX-class` *TENxMTX: The Matrix Market representation class for 10X Data*

Description

This class is designed to work with 10x MTX datasets, particularly from the multiome pipelines.

Usage

```
## S4 method for signature 'TENxMTX,ANY,ANY'
import(con, format, text, ...)
```

Arguments

`con`	The connection from which data is loaded or to which data is saved. If this is a `character` vector, it is assumed to be a file name and a corresponding file connection is created and then closed after exporting the object. If it is a BiocFile derivative, the data is loaded from or saved to the underlying resource. If missing, the function will return the output as a character vector, rather than writing to a connection.
`format`	The format of the output. If missing and con is a file name, the format is derived from the file extension. This argument is unnecessary when con is a derivative of BiocFile.
`text`	If con is missing, this can be a character vector directly providing the string data to import.
`...`	Parameters to pass to the format-specific method.

Details

The `TENxMTX` class is a straightforward implementation that allows the user to import a Matrix Market file format using `Matrix::readMM`. Currently, it returns a `SummarizedExperiment` with an internal `dgCMatrix` assay. To request other formats, please open an issue on GitHub.

Value

A `TENxMTX` class object

Methods (by generic)

- `import(con = TENxMTX, format = ANY, text = ANY)`: Import method mainly for mtx.gz files from 10x

Slots

`compressed` logical(1) Whether or not the file is in compressed format, usually gzipped (`.gz`).

`TENxPeaks` *Import 10x peak annotation files from 10x*

Description

This constructor function is designed to work with the files denoted with "peak_annotation" in the file name. These are usually produced as tab separated value files, i.e., `.tsv`.

Usage

```
TENxPeaks(resource, extension, ...)
```

Arguments

`resource`	character(1) The path to the file
`extension`	character(1) The file extension for the given resource. It can usually be obtained from the file path. An override can be provided especially for `ExperimentHub` resources where the file extension is removed.
`...`	Additional inputs to the low level class generator functions

Details

The output class allows handling of peak data. It can be used in conjunction with the `annotation` method on a `SingleCellExperiment` to add peak information to the experiment. The ranged data is represented as a `GRanges` class object.

Value

A `GRanges` class object of peak locations

Examples

```
fi <- system.file(
    "extdata", "pbmc_granulocyte_sorted_3k_ex_atac_peak_annotation.tsv",
    package = "TENxIO", mustWork = TRUE
)
peak_file <- TENxPeaks(fi)
peak_anno <- import(peak_file)
peak_anno
```

```
example(TENxH5)

## Add peaks to an existing SCE
## First, import the SCE from an example H5 file
h5f <- system.file(
    "extdata", "pbmc_granulocyte_ff_bc_ex.h5",
    package = "TENxIO", mustWork = TRUE
)
con <- TENxH5(h5f)
sce <- import(con)
## auto-import peaks when using annotation<-
annotation(sce, name = "peak_annotation") <- peak_file
annotation(sce)
```

`TENxPeaks-class` *TENxPeaks: The class to represent 10x Peaks files*

Description

This class is designed to work with the files denoted with "peak_annotation" in the file name. These are usually produced as tab separated value files, i.e., `.tsv`.

Usage

```
## S4 method for signature 'TENxPeaks,ANY,ANY'
import(con, format, text, ...)

## S4 replacement method for signature 'SingleCellExperiment,ANY'
annotation(object, ...) <- value

## S4 method for signature 'SingleCellExperiment'
annotation(object, ...)
```

Arguments

`con`	The connection from which data is loaded or to which data is saved. If this is a `character` vector, it is assumed to be a file name and a corresponding file connection is created and then closed after exporting the object. If it is a BiocFile derivative, the data is loaded from or saved to the underlying resource. If missing, the function will return the output as a character vector, rather than writing to a connection.
`format`	The format of the output. If missing and con is a file name, the format is derived from the file extension. This argument is unnecessary when con is a derivative of BiocFile.
`text`	If con is missing, this can be a character vector directly providing the string data to import.

`...`	Parameters to pass to the format-specific method.
`object`	The object to export.
`value`	The annotation information to set on `object`.

Details

This class is a straightforward class for handling peak data. It can be used in conjunction with the `annotation` method on a `SingleCellExperiment` to add peak information to the experiment. The ranged data is represented as a `GRanges` class object.

Value

A `TENxPeaks` class object

Functions

- `import(con = TENxPeaks, format = ANY, text = ANY)`: Import a peaks_annotation file from 10x as a `GRanges` representation
- `annotation(object = SingleCellExperiment) <- value`: Replacement method to add annotation data to a `SingleCellExperiment`
- `annotation(SingleCellExperiment)`: Extraction method to obtain annotation data from a `SingleCellExperiment` representation

---

`TENxTSV-class` *TENxTSV: A class to represent 10x tab separated values files*

Description

This class is general purpose for reading in tabular data from the 10x Genomics website with the `.tsv` file extension. The class also supports compressed files, i.e., those with the `.tsv.gz` extension.

Usage

```
## S4 method for signature 'TENxTSV,ANY,ANY'
import(con, format, text, ...)

TENxTSV(resource, compressed = FALSE, ...)

## S4 method for signature 'TENxTSV'
metadata(x, ...)
```

Arguments

`con`	The connection from which data is loaded or to which data is saved. If this is a `character` vector, it is assumed to be a file name and a corresponding file connection is created and then closed after exporting the object. If it is a BiocFile derivative, the data is loaded from or saved to the underlying resource. If missing, the function will return the output as a character vector, rather than writing to a connection.
`format`	The format of the output. If missing and `con` is a file name, the format is derived from the file extension. This argument is unnecessary when `con` is a derivative of BiocFile.
`text`	If `con` is missing, this can be a character vector directly providing the string data to import.
`...`	Parameters to pass to the format-specific method.
`resource`	character(1) The path to the file
`compressed`	logical(1) Whether the resource file is compressed (default FALSE)
`x`	A `TENxTSV` object

Details

Typical `.tsv` files obtained from the 10X website are compressed and contain information relevant to 'barcodes' and 'features'. Currently, the code only supports files such as `features.tsv.*` and `barcodes.tsv.*`.

Value

A `TENxTSV` class object; a `tibble` for the import method

Functions

- `import(con = TENxTSV, format = ANY, text = ANY)`: General import method for `tsv` files from 10x; using `readr::read_tsv` and returning a `tibble` representation
- `metadata(TENxTSV)`: `metadata` method for `TENxTSV` objects

Index