Package ‘VisiumIO’

May 18, 2024

Title Import Visium data from the 10X Space Ranger pipeline

Version 1.0.0

Description The package allows users to readily import spatial data obtained from either the 10X website or from the Space Ranger pipeline. Supported formats include tar.gz, h5, and mtx files. Multiple files can be imported at once with *List type of functions. The package represents data mainly as SpatialExperiment objects.

License Artistic-2.0

Depends R (>= 4.4.0), TENxIO

Imports BiocBaseUtils, BiocGenerics, BiocIO, jsonlite, methods, S4Vectors, SpatialExperiment, SummarizedExperiment

Suggests BiocStyle, knitr, rmarkdown, tinytest

biocViews Software, Infrastructure, DataImport, SingleCell, Spatial

VignetteBuilder knitr

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

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

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

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

git_branch RELEASE_3_19

git_last_commit cc0c5de

git_last_commit_date 2024-04-30

Repository Bioconductor 3.19

Date/Publication 2024-05-17

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

Maintainer Marcel Ramos <`marcel.ramos@roswellpark.org`>

Contents

`TENxSpatialCSV-class` *Represent and import spatial CSV data from 10X Genomics*

Description

`TENxSpatialCSV` is a class to represent and import spatial CSV files with specific column names. It is a composed class of TENxIO::TENxFile and contains additional slots for the column names and whether the CSV is a list-type of file.

Usage

```
TENxSpatialCSV(resource, colnames = .TISSUE_POS_COLS)

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

Arguments

`resource`	character(1) The path to the file
`colnames`	`character()` A vector specifying the column names of the CSV, defaults to `c("barcode", "in_tissue", "array_row", "array_col", "pxl_row_in_fullres", "pxl_col_in_fullres")`.
`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.
`...`	Additional inputs to the low level class generator functions

Details

Typically, the user will not create an object of this class directly but rather use the `TENxVisium()` constructor function to create an object of this class in the background. The column names are set to the default values of `c("barcode", "in_tissue", "array_row", "array_col", "pxl_row_in_fullres", "pxl_col_in_fullres")`. The column names can be changed by specifying the `colnames` argument in the constructor function.

Value

TENxSpatialCSV: An object of class TENxSpatialCSV

import-method: A `DataFrame` object containing the data from the CSV file

Slots

`isList` `logical(1)` A scalar specifying whether the CSV is a list-type of file

`colnames` `character()` A vector specifying the column names of the CSV

Examples

```
sample_dir <- system.file(
    file.path("extdata", "10xVisium", "section1"),
    package = "SpatialExperiment"
)
spatial_dir <- Filter(
  function(x) endsWith(x, "spatial"), list.dirs(sample_dir)
)
csvresource <- file.path(spatial_dir, "tissue_positions_list.csv")
TENxSpatialCSV(csvresource)
head(import(TENxSpatialCSV(csvresource)), 4)
```

`TENxSpatialList-class` *A class to represent and import spatial Visium data*

Description

This class is a composed class of TENxFileList, which can contain a list of TENxFile objects, and a TENxSpatialList object. It is meant to handle spatial Visium data from 10X Genomics.

Usage

```
TENxSpatialList(
  resources,
  sample_id = "sample01",
  images = c("lowres", "hires", "detected", "aligned"),
  jsonFile = .SCALE_JSON_FILE,
  tissuePattern = "tissue_positions.*\\.csv",
```

```
  ...
)

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

Arguments

`resources`	A TENxFileList object or a file path to the tarball containing the matrix / assay data resources.
`sample_id`	`character(1)` A single string specifying the sample ID.
`images`	`character()` A vector specifying the images to be imported; can be one or multiple of "lowres", "hires", "detected", "aligned".
`jsonFile`	`character(1)` A single string specifying the name of the JSON file containing the scale factors.
`tissuePattern`	`character(1)` A single string specifying the pattern to match the tissue positions file.
`...`	Parameters to pass to the format-specific method.
`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.

Details

Typically, the user will not create an object of this class directly but rather use the `TENxVisium()` constructor function to create an object of this class.

Value

A `SpatialExperiment` object

Methods (by generic)

- `import(con = TENxSpatialList, format = ANY, text = ANY)`: Import a `TENxSpatialList` object

Examples

```
spatial_dir <- system.file(
    file.path("extdata", "10xVisium", "section1", "outs", "spatial"),
    package = "SpatialExperiment"
)

TENxSpatialList(resources = spatial_dir)
```

`TENxVisium-class` *A class to represent and import a single Visium Sample*

Description

This class is a composed class of TENxFileList which can contain a list of TENxFile objects and a TENxSpatialList object. It is meant to handle a single Visium sample from 10X Genomics.

Usage

```
TENxVisium(
  resources,
  spatialResource,
  spacerangerOut,
  sample_id = "sample01",
  processing = c("filtered", "raw"),
  images = c("lowres", "hires", "detected", "aligned"),
  jsonFile = .SCALE_JSON_FILE,
  tissuePattern = "tissue_positions.*\\.csv",
  spatialCoordsNames = c("pxl_col_in_fullres", "pxl_row_in_fullres"),
  ...
)

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

Arguments

`resources`	A TENxFileList object or a file path to the tarball containing the matrix / assay data resources.
`spatialResource`	A TENxSpatialList object or a file path to the tarball containing the spatial data.
`spacerangerOut`	`character(1)` A single string specifying the path to the sample directory of `spaceranger count`. The directory must contain the `(processing)_feature_bc_matrix` and `spatial` sub directories in addition to the `outs` folder.
`sample_id`	`character(1)` A single string specifying the sample ID.

`processing`	`character(1)` A single string indicating the processing folder available e.g., "filtered_feature_barcode_matrix" in the `spacerangerOut` folder. It can be either "filtered" or "raw" (default "filtered"). Only used when `spacerangerOut` is specified.
`images`	`character()` A vector specifying the images to be imported; can be one or multiple of "lowres", "hires", "detected", "aligned".
`jsonFile`	`character(1)` A single string specifying the name of the JSON file containing the scale factors.
`tissuePattern`	`character(1)` A single string specifying the pattern to match the tissue positions file.
`spatialCoordsNames`	`character()` A vector of strings specifying the names of the columns in the spatial data containing the spatial coordinates.
`...`	In the constructor, additional arguments passed to `TENxFileList()`; otherwise, 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.

Details

Typically, the user will not create an object of this class directly but rather use TENxVisiumList constructor function for multiple samples. Note that the `images`, `jsonFile`, `tissuePattern`, and `spatialCoordsNames` arguments are only considered when the `spacerangerOut` argument or both the `resources` and `spatialResource` arguments are paths to files.

Value

A SpatialExperiment object

Functions

- `import(con = TENxVisium, format = ANY, text = ANY)`: Import Visium data

Slots

`resources`	A TENxFileList object containing the Visium data.
`spatialList`	A TENxSpatialList object containing the spatial

`coordNames` `character()` A vector specifying the names of the columns in the spatial data containing the spatial coordinates.

`sampleId` `character(1)` A scalar specifying the sample identifier.

See Also

https://support.10xgenomics.com/spatial-gene-expression/software/pipelines/latest/output/overview

Examples

```
sample_dir <- system.file(
    file.path("extdata", "10xVisium", "section1"),
    package = "SpatialExperiment"
)

tv <- TENxVisium(
    spacerangerOut = sample_dir, processing = "raw", images = "lowres"
)

import(tv)
```

`TENxVisiumList-class` *A class to represent and import multiple Visium samples*

Description

This class contains a `SimpleList` of TENxVisium objects each corresponding to one sample.

Usage

```
TENxVisiumList(
  sampleFolders,
  sample_ids,
  processing = c("filtered", "raw"),
  images = c("lowres", "hires", "detected", "aligned"),
  jsonFile = .SCALE_JSON_FILE,
  tissuePattern = "tissue_positions.*\\.csv",
  spatialCoordsNames = c("pxl_col_in_fullres", "pxl_row_in_fullres"),
  ...
)

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

Arguments

`sampleFolders`	`character()` A vector of strings specifying the directories containing the output of the `spaceranger count` command.
`sample_ids`	`character()` A vector of strings specifying the sample IDs. If not provided, the sample IDs will be the names of the `sampleFolders`. Therefore, the `sample_ids` must be the same length as `sampleFolders`.
`processing`	`character(1)` A single string indicating the processing folder available e.g., "filtered_feature_barcode_matrix" in the `spacerangerOut` folder. It can be either "filtered" or "raw" (default "filtered"). Only used when `spacerangerOut` is specified.
`images`	`character()` A vector specifying the images to be imported; can be one or multiple of "lowres", "hires", "detected", "aligned".
`jsonFile`	`character(1)` A single string specifying the name of the JSON file containing the scale factors.
`tissuePattern`	`character(1)` A single string specifying the pattern to match the tissue positions file.
`spatialCoordsNames`	`character()` A vector of strings specifying the names of the columns in the spatial data containing the spatial coordinates.
`...`	In the constructor, additional arguments passed to `TENxFileList()`; otherwise, 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.

Details

Typically, the user will provide a path to a directory containing the output of the `spaceranger count` command. The `spaceranger count` command outputs a folder containing the "raw" or "filtered" `()_feature_bc_matrix`.

Value

A SpatialExperiment object

Functions

- `import(con = TENxVisiumList, format = ANY, text = ANY)`: Import multiple Visium samples

See Also

https://support.10xgenomics.com/spatial-gene-expression/software/pipelines/latest/output/overview

Examples

```
sample_dirs <- list.dirs(
    system.file(
        file.path("extdata", "10xVisium"),
        package = "SpatialExperiment"
    ),
    recursive = FALSE, full.names = TRUE
)

tvl <- TENxVisiumList(
    sampleFolders = sample_dirs,
    sample_ids = c("sample01", "sample02"),
    processing = "raw",
    images = "lowres"
)

import(tvl)
```

Index