Package 'VisiumIO'

April 22, 2026

Title Import Visium data from the 10X Space Ranger pipeline

Version 1.7.7

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 (>= 4.5.0), TENxIO

Imports BiocBaseUtils, BiocGenerics, BiocIO (>= 1.15.1), jsonlite, methods, S4Vectors, SingleCellExperiment, SpatialExperiment, SummarizedExperiment

Suggests arrow, BiocStyle, data.table, knitr, readr, rmarkdown, sf, tinytest

biocViews Software, Infrastructure, DataImport, SingleCell, Spatial

VignetteBuilder knitr

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

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

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

Collate 'TENxGeoJSON.R' 'TENxParquet-class.R' 'TENxSpatialCSV.R' 'TENxSpatialList-class.R' 'TENxSpatialParquet.R' 'TENxVisium-class.R' 'TENxVisiumList-class.R' 'TENxVisiumHD-class.R' 'VisiumIO-package.R' 'utilities.R'

Date 2025-12-24

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

git_branch devel

git_last_commit 989b6eb

git_last_commit_date 2026-03-25

Repository Bioconductor 3.24

Date/Publication 2026-04-21

Author Marcel Ramos [aut, cre] (ORCID:
<https://orcid.org/0000-0002-3242-0582>),
Estella YiXing Dong [aut, ctb],
Dario Righelli [aut, ctb],
Helena Crowell [aut, ctb],
NCI [fnd] (GrantNo.: U24CA289073)

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

Contents

`VisiumIO-package` *VisiumIO: Import Visium data from the 10X Space Ranger pipeline*

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.

Author(s)

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

Authors:

- Estella YiXing Dong [contributor]
- Dario Righelli [contributor]
- Helena Crowell [contributor]

See Also

Useful links:

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

compareBarcodes *Compare barcodes between raw and filtered data*

Description

This function compares the barcodes between raw and filtered data **depending** on the order of `processing`. Typically, the "raw" barcodes are compared to the "filtered" ones. The presence of raw barcodes in the filtered data are marked as `TRUE` in the resulting `data.frame`.

Usage

```
compareBarcodes(
  from_resource,
  to_resource,
  spacerangerOut,
  format = c("mtx", "h5"),
  processing = c("raw", "filtered"),
  ...
)
```

Arguments

`from_resource`	`character(1)` The path to the resource file whose barcodes are used as the basis of the comparison; typically, the "raw" feature barcodes are used.
`to_resource`	`character(1)` The path to the resource file whose barcodes are compared to the `from_resource`; typically, the "filtered" feature barcodes.
`spacerangerOut`	`character(1)` A single string specifying the path to the directory where the output of `spaceranger count` is located; typically (but not necessarily), this is the `outs` directory. The directory must contain the `(processing)_feature_bc_matrix` and `spatial` sub directories.
`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.
`processing`	`character(2)` A vector of length 2 that corresponds to the processing type. The processing types are typically "raw" and "filtered". These are the prefixes of the folder names `raw_feature_bc_matrix` and `filtered_feature_bc_matrix`. The order of the vector determines the comparison. By default, `processing = c("raw", "filtered")`, which means barcodes in the raw data are compared to the filtered data.
`...`	Additional arguments passed to `TENxH5` or `TENxFileList`.

Value

A `data.frame` with barcodes of the first element in the `processing` data type as the first column and a logical vector indicating whether the barcodes are found in the second element in `processing`. For example, if processing is `c("raw", "filtered")`, then the first column will be the barcodes in the `raw` data and the second column will be a logical vector indicating whether the barcodes are found in the `filtered` data.

Examples

```
if (interactive()) {
    compareBarcodes(
        from_resource = "V1_Adult_Mouse_Brain_raw_feature_bc_matrix.tar.gz",
        to_resource =
            "V1_Adult_Mouse_Brain_filtered_feature_bc_matrix.tar.gz",
    )

    compareBarcodes(
        from_resource =
            "V1_Adult_Mouse_Brain_raw_feature_bc_matrix.h5",
        to_resource =
            "V1_Adult_Mouse_Brain_filtered_feature_bc_matrix.h5"
    )

    compareBarcodes(spacerangerOut = "~/data/outs", format = "h5")

    compareBarcodes(
        spacerangerOut = "~/data/feature_bc_matrix", format = "mtx"
    )

    compareBarcodes(
        spacerangerOut = "~/data/folder_feature_bc_matrix", format = "mtx"
    )
}
```

---

st_invert_y *Flip the Y-axis of cell or nucleus segmentations to align with H&E image*

Description

This function flips the Y-axis of cell or nucleus segmentations stored in an `sf` object to align with the H&E image. The Y-axis flipping is necessary because the origin (0,0) in image coordinates is at the top-left corner, while in Cartesian coordinates, the origin is at the bottom-left corner. The function takes into account the image height and scaling factor to accurately flip the Y-coordinates of the segmentations.

Usage

```
st_invert_y(sf, type = c("POINT", "POLYGON"), img_height, scalef)
```

Arguments

`sf`	`sf` an `sf` class object read from a `.geojson` file.
`type`	`character(1)` "POINT" for cell centroid, or "POLYGON" for cell segmentation mask. Default is "POINT".
`img_height`	`numeric(1)` The total length along the Y axis of the image. Obtained by reading in `hires` or `lowres` `.png` under `/spatial` folder with `magick::image_read()`.
`scalef`	`numeric(1)` The scaling factor from a `/spatial/scalefactors_json.json` file

Value

an sf object with Y-axis of the points or polygons flipped

Author(s)

Estella YiXing Dong

Examples

```
geojson_file <- system.file(
    file.path("extdata", "segmented_outputs", "cell_segmentations.geojson"),
    mustWork = TRUE, package = "VisiumIO"
)
geo_data <- sf::st_read(geojson_file, quiet = TRUE)
st_invert_y(
    sf = geo_data, type = "POLYGON", img_height = 3886, scalef = 0.079
)
```

TENxGeoJSON-class *Import 10X Genomics GeoJSON files*

Description

TENxGeoJSON is a class to represent and import GeoJSON files from 10X Genomics. It is a composed class of TENxIO::TENxFile.

Usage

```
TENxGeoJSON(resource)

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

Arguments

resource	character(1) The path to the file
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.

Value

`TENxGeoJSON()`: An object of class TENxGeoJSON

import-method: An `sf` and `data.frame` with the GeoJSON data

See Also

https://www.10xgenomics.com/support/software/xenium-ranger/3.0/analysis/segmentation-inputs

Examples

```
segout_folder <- system.file(
    file.path("extdata", "segmented_outputs"),
    package = "VisiumIO"
)

## import cell boundaries
cellsegs <- file.path(segout_folder, "cell_segmentations.geojson")

TENxGeoJSON(cellsegs)

TENxGeoJSON(cellsegs) |>
    import()

## import nucleus boundaries
nucsegs <- file.path(segout_folder, "nucleus_segmentations.geojson")

TENxGeoJSON(nucsegs) |>
    import()
```

TENxParquet-class *Represent and import Parquet data from 10X Genomics*

Description

`TENxParquet` is a virtual class to represent and import Parquet files from 10X Genomics. It is a composed class of TENxIO::TENxFile.

`TENxSpatialParquet` is a class to represent and import spatial Parquet files from 10X Genomics. It is a composed class of TENxIO::TENxFile.

Usage

```
TENxParquet(resource, type = c("spatial", "mapping"))

TENxMappingParquet(resource, colnames = .MAPPING_POS_COLS)

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

Arguments

resource	character(1) The path to the file
type	character(1) A scalar specifying the type of Parquet file to import, either "spatial" or "mapping". If not provided, the type is determined by whether or not there is a "mapping" keyword in the file name.
colnames	character() A vector specifying the column names of the Parquet, defaults to all columns in the dataset.
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 TENxParquet constructor function to create an object of either TENxSpatialParquet or TENxMappingParquet class in the background.

Value

TENxParquet(): An object of class TENxSpatialParquet or TENxMappingParquet

TENxMappingParquet(): An object of class TENxMappingParquet

Examples

```
sample_dir <- system.file(
    file.path("extdata", "binned_outputs", "square_002um", "spatial"),
    package = "VisiumIO"
)
spatial_dir <- Filter(
    function(x) endsWith(x, "spatial"), list.dirs(sample_dir)
)
parquetres <- file.path(spatial_dir, "tissue_positions.parquet")
TENxParquet(parquetres)
parq_file <- system.file(
    "extdata", "barcode_mappings.parquet",
    package = "VisiumIO", mustWork = TRUE
)
TENxMappingParquet(parq_file)
```

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"). Mainly used for the "positions" CSV type of file which does not include column names in the file.
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.

Set the option "VisiumIO.csvreader" to either "data.table" or "readr" to use the data.table::fread or readr::read_csv functions, respectively. These options are useful when the CSV file is relatively large and the user wants to use faster read-in options. Note that the outputs will still be converted to DataFrame when incorporated to the SpatialExperiment or SingleCellExperiment object.

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

`variant character(1)` A scalar specifying the variant of the CSV file "positions", "cell_boundaries", or "other". The variant is determined by the name of the CSV file within the constructor function. Values include "positions", "cell_boundaries", and "other".

`compressed logical(1)` A scalar specifying whether the CSV is compressed (mainly with a `.gz` file extension).

Examples

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

import(TENxSpatialCSV(csvresource)) |>
    attr("metadata") |>
    lapply(names)
```

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", "aligned_fiducials", "cytassist"),
  jsonFile = .SCALE_JSON_FILE,
  tissuePattern = "tissue_positions.*",
  bin_size = character(0L),
  ...
)

## 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.
bin_size	character(1) The bin size of the images to import. The default is 008. It corresponds to the directory name square_000um where 000 is the bin value.
...	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

Slots

images character() The image name(s) to use with grep and include in the list of files. Can be one of "lowres", "hires", "lowres", "hires", "detected", "aligned", "aligned_fiducials", or "cytassist".

scaleJSON character(1) The file name of the scale factors JSON file, defaults to 'scalefactors_json.json'.

tissuePos character(1) The file name of the tissue positions file; typically a .parquet or .csv file.

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

binSize The bin size of the images to import. The default slot value is character(). It typically corresponds to the directory name square_000um where 000 is the bin value.

Examples

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

TENxSpatialList(resources = spatial_dir, images = "lowres")

TENxSpatialList(resources = spatial_dir, images = "lowres") |>
    metadata() |> lapply(names)
```

TENxSpatialParquet-class

Represent and import spatial Parquet data from 10X Genomics

Description

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

Usage

```
TENxSpatialParquet(resource, colnames)

## S4 method for signature 'TENxSpatialParquet,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 Parquet, defaults to all columns in the dataset.
`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

`TENxSpatialParquet()`: An object of class TENxSpatialParquet

import-method: A `tibble` object containing the data from the Parquet file

Examples

```
sample_dir <- system.file(
    file.path("extdata", "binned_outputs", "square_002um", "spatial"),
    package = "VisiumIO"
)
spatial_dir <- Filter(
    function(x) endsWith(x, "spatial"), list.dirs(sample_dir)
)
parquetres <- file.path(spatial_dir, "tissue_positions.parquet")
TENxSpatialParquet(parquetres)
TENxSpatialParquet(parquetres) |>
    import()

## metadata in attributes
TENxSpatialParquet(parquetres) |>
    import() |>
    attr("metadata") |>
    lapply(names)
```

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"),
  format = c("mtx", "h5"),
  images = c("lowres", "hires", "detected", "aligned", "cytassist"),
  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 directory where the output of spaceranger count is located; typically (but not necessarily), this is the outs directory. The directory must contain the (processing)_feature_bc_matrix and spatial sub directories.
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.
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.
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.
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 or TENxH5 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

```
outs_dir <- system.file(
    file.path("extdata", "10xVisium", "section1", "outs"),
    package = "VisiumIO"
)

## using spaceRangerOut folder
tv <- TENxVisium(
    spaceRangerOut = outs_dir, processing = "raw", images = "lowres"
)

import(tv)

## with TENxFileList spaceRangerOut input
tvfl <- TENxVisium(
    spaceRangerOut = TENxFileList(outs_dir),
    format = "mtx",
    processing = "raw",
    images = "lowres"
)

import(tvfl)

## check metadata of the object
import(tvfl) |>
    metadata() |>
    lapply(names)

## importing h5 format
tvfl <- TENxVisium(
    spaceRangerOut = outs_dir,
    format = "h5",
    processing = "raw",
    images = "lowres"
```

```
)

import(tvfl)

rffolder <- file.path(outs_dir, "raw_feature_bc_matrix")
## using resources and spatialResource inputs
tvfl <- TENxVisium(
    resources = rffolder,
    spatialResource = file.path(dirname(rffolder), "spatial"),
    format = "mtx",
    processing = "raw",
    images = "lowres"
)

import(tvfl)
```

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

Description

This class contains a `SimpleList` of TENxVisiumHD objects each corresponding to one sample. The provided `spaceRangerOut` folder should contain a `binned_outputs` folder where multiple `bin_size` subfolders are present, e.g., `square_002um`.

Usage

```
TENxVisiumHD(
  resources,
  spatialResource,
  spaceRangerOut,
  segmented_outputs,
  sample_id = "sample01",
  processing = c("filtered", "raw"),
  format = c("mtx", "h5"),
  images = c("lowres", "hires", "detected", "aligned_fiducials"),
  bin_size = c("008", "016", "002"),
  jsonFile = .SCALE_JSON_FILE,
  tissuePattern = "tissue_positions\\.parquet",
  spatialCoordsNames = c("pxl_col_in_fullres", "pxl_row_in_fullres"),
  mappingPattern = "barcode_mappings\\.parquet",
  boundary = c("cell_segmentations", "nucleus_segmentations", "both"),
  ...
)

## S4 method for signature 'TENxVisiumHD,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 directory where the output of spaceranger count is located; typically (but not necessarily), this is the outs directory. The directory must contain the (processing)_feature_bc_matrix and spatial sub directories.

segmented_outputs
: character(1) The path to the segmented_outputs directory

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.

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.

images
: character() A vector specifying the images to be imported; can be one or multiple of "lowres", "hires", "detected", "aligned".

bin_size
: character(1) The bin size of the images to import. The default is 008. It corresponds to the directory name square_000um where 000 is the bin value.

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.

mappingPattern
: character(1) The pattern used in list.files that identifies the mapping file. The default is "barcode_mappings\\.parquet".

boundary
: character(1) The type of segmentation boundary to use. The options are "cell_segmentations" (default), "nucleus_segmentations", or "both". When "both" is specified, "cell_segmentations" are added to the spatialCoords and the nucleus_segmentations centroids (labeled x.nuc and y.nuc) are added to the colData of the returned object.

...
: 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.

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.

Note that `nucleus_segmentations.geojson` file must be in the same folder as the `cell_segmentations.geojson` file for the nucleus centroids to be imported correctly when selecting the `"both"` for the `boundary` argument.

Value

A SpatialExperiment object

Functions

- `import(con = TENxVisiumHD, format = ANY, text = ANY)`: Import Visium HD data from multiple bin sizes

Author(s)

E. Y. Dong, M. Ramos

Examples

```
vdir <- system.file(
    "extdata", package = "VisiumIO", mustWork = TRUE
)

## with spacerangerOut folder
TENxVisiumHD(spacerangerOut = vdir, bin_size = "002", images = "lowres")

TENxVisiumHD(spacerangerOut = vdir, bin_size = "002", images = "lowres") |>
    import()

## indicate h5 format
TENxVisiumHD(
    spacerangerOut = vdir, bin_size = "002",
    images = "lowres", format = "h5"
)

TENxVisiumHD(
    spacerangerOut = vdir, bin_size = "002",
    images = "lowres", format = "h5"
) |>
    import()

## use resources and spatialResource arguments as file paths
TENxVisiumHD(
    resources = file.path(
        vdir, "binned_outputs", "square_002um",
        "filtered_feature_bc_matrix.h5"
    ),
    spatialResource = file.path(
        vdir, "binned_outputs", "square_002um",
        "spatial"
    ),
    bin_size = "002", processing = "filtered",
    images = "lowres", format = "h5"
) |>
    import()
```

```
## provide the spatialResource argument as a TENxFileList
TENxVisiumHD(
    resources = file.path(
        vdir, "binned_outputs", "square_002um",
        "filtered_feature_bc_matrix.h5"
    ),
    spatialResource = TENxFileList(
        file.path(
            vdir, "binned_outputs", "square_002um",
            "spatial"
        )
    ),
    bin_size = "002", images = "lowres", format = "h5"
) |>
    import()

## with segmented_outputs folder
seg_outs <- system.file(
    "extdata", "segmented_outputs", package = "VisiumIO", mustWork = TRUE
)
TENxVisiumHD(
    segmented_outputs = seg_outs,
    format = "h5",
    images = "lowres"
) |>
    import()
```

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"),
  format = c("mtx", "h5"),
  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".
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.
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.
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 = "VisiumIO"
    ),
    recursive = FALSE, full.names = TRUE
)

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

import(tvl)
```

Index