

Package: tiledbsoma (via r-universe)

August 22, 2024

Type Package

Title TileDB SOMA

Description Interface for working with 'TileDB'-based Stack of Matrices, Annotated ('SOMA'): an open data model for representing annotated matrices, like those commonly used for single cell data analysis.

Version 0.99.4

URL <https://github.com/single-cell-data/TileDB-SOMA>

BugReports <https://github.com/single-cell-data/TileDB-SOMA/issues>

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 4.0.0)

Imports R6, methods, Matrix, stats, bit64, tiledb (>= 0.20.2), arrow, utils, fs, glue, urltools, Rcpp, data.table, spd, rlang, tools

LinkingTo Rcpp, RcppSpdlog

Additional_repositories <https://ghrr.github.io/drat>

Remotes bioc::release/SingleCellExperiment,
github::tiledb-inc/TileDB-R@0.20.2

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

Suggests rmarkdown, knitr, withr, testthat (>= 3.0.0), pbmc3k.tiledb, SeuratObject (>= 4.1.0), datasets, SingleCellExperiment (>= 1.20.0), SummarizedExperiment, S4Vectors

VignetteBuilder knitr

Config/testthat/edition 2

OS_type unix

SystemRequirements cmake, git

Repository <https://aaronwolen.r-universe.dev>

RemoteUrl <https://github.com/single-cell-data/TileDB-SOMA>

RemoteRef aaronwolen/use-remotes-for-tiledb-0.20.2

RemoteSha 05fbf9effea1be30cc9fab80103de7e1189c69a0

Contents

ConfigList	3
example-datasets	4
matrixZeroBasedView	5
PlatformConfig	7
show_package_versions	9
SOMAAxisQuery	9
SOMAAxisQueryResult	10
SOMACollection	11
SOMACollectionCreate	11
SOMACollectionOpen	12
SOMADataFrame	13
SOMADataFrameCreate	14
SOMADataFrameOpen	15
SOMADenseNDArray	16
SOMADenseNDArrayCreate	18
SOMADenseNDArrayOpen	18
SOMAExperiment	19
SOMAExperimentAxisQuery	20
SOMAExperimentCreate	26
SOMAExperimentOpen	27
SOMAMeasurement	27
SOMAMeasurementCreate	28
SOMAMeasurementOpen	29
SOMAOpen	30
SOMASparseNDArray	30
SOMASparseNDArrayCreate	32
SOMASparseNDArrayOpen	33
SOMATileDBContext	33
SparseReadIter	35
TableReadIter	36
TileDBObject	37
tiledbsoma_stats	39
write_soma	39
write_soma.Seurat	40

Index

42

ConfigList

A Configuration List

Description

An R6 mapping type for configuring various “parameters”. Essentially, serves as a nested map where the inner map is a [ScalarMap](#): {<param>: {<key>: <value>}}

Super class

`tiledb::MappingBase -> ConfigList`

Methods

Public methods:

- `ConfigList$get()`
- `ConfigList$set()`
- `ConfigList$setv()`
- `ConfigList$clone()`

Method `get()`:

Usage:

```
ConfigList$get(param, key = NULL, default = quote(expr = ))
```

Arguments:

`param` Outer key or “parameter” to fetch

`key` Inner key to fetch; pass NULL to return the [map](#) for `param`

`default` Default value to fetch if key is not found; defaults to NULL

Returns: The value of key for `param` in the map, or `default` if key is not found

Method `set()`:

Usage:

```
ConfigList$set(param, key, value)
```

Arguments:

`param` Outer key or “parameter” to set

`key` Inner key to set

`value` Value to add for key, or NULL to remove the entry for key; optionally provide only `param` and `value` as a [ScalarMap](#) to update `param` with the keys and values from `value`

Returns: `\[chainable\]` Invisibly returns `self` with `value` added for key in `param`

Method `setv()`:

Usage:

```
ConfigList$setv(...)
```

Arguments:

... Ignored

Returns: Nothing; setv() is disabled for ConfigList objects

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
ConfigList$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

example-datasets

SOMA Example Datasets

Description

Access example SOMA objects bundled with the tiledbsoma package.

Use list_datasets() to list the available datasets and load_dataset() to load a dataset into memory using the appropriate SOMA class. The extract_dataset() method returns the path to the extracted dataset without loading it into memory.

Usage

```
list_datasets()
```

```
extract_dataset(name, dir = tempdir())
```

```
load_dataset(name, dir = tempdir(), tiledbsoma_ctx = NULL)
```

Arguments

name The name of the dataset.

dir The directory where the dataset will be extracted to (default: tempdir()).

tiledbsoma_ctx Optional TileDB ‘Context’ object, defaults to NULL

Details

The SOMA objects are stored as tar.gz files in the package’s extdata directory. Calling load_dataset() extracts the tar.gz file to the specified dir, inspects its metadata to determine the appropriate SOMA class to instantiate, and returns the SOMA object.

Value

- list_datasets() returns a character vector of the available datasets.
- extract_dataset() returns the path to the extracted dataset.
- load_dataset() returns a SOMA object.

Examples

```
soma_pbmc_small <- load_dataset("soma-exp-pbmc-small")
```

`matrixZeroBasedView` *matrixZeroBasedView* is a wrapper shim for a matrix or `Matrix::sparseMatrix` providing

Description

`matrixZeroBasedView` is a class that allows elemental matrix access using zero-based indices.

Methods

Public methods:

- `matrixZeroBasedView$new()`
- `matrixZeroBasedView$take()`
- `matrixZeroBasedView$dim()`
- `matrixZeroBasedView$nrow()`
- `matrixZeroBasedView$ncol()`
- `matrixZeroBasedView$get_one_based_matrix()`
- `matrixZeroBasedView$sum()`
- `matrixZeroBasedView$print()`
- `matrixZeroBasedView$clone()`

Method `new()`: Initialize (lifecycle: experimental)

Usage:

```
matrixZeroBasedView$new(x)
```

Arguments:

x `matrix` or `Matrix::sparseMatrix` or `Matrix::Matrix`

Method `take()`: Zero-based matrix element access

Usage:

```
matrixZeroBasedView$take(i = NULL, j = NULL)
```

Arguments:

i Row index (zero-based).

j Column index (zero-based).

Returns: The specified matrix slice as another `matrixZeroBasedView`

Method `dim()`: dim

Usage:

```
matrixZeroBasedView$dim()
```

Returns: The dimensions of the matrix.

Method nrow(): nrow

Usage:

```
matrixZeroBasedView$nrow()
```

Returns: Matrix row count.

Method ncol(): ncol

Usage:

```
matrixZeroBasedView$ncol()
```

Returns: Matrix column count.

Method get_one_based_matrix(): Get the one-based R matrix with its original class

Usage:

```
matrixZeroBasedView$get_one_based_matrix()
```

Returns: One-based matrix

Method sum(): Perform arithmetic sum between this linkmatrixZeroBasedView and another linkmatrixZeroBasedView.

Usage:

```
matrixZeroBasedView$sum(x)
```

Arguments:

x the linkmatrixZeroBasedView to sum.

Returns: The result of the sum as a [matrixZeroBasedView](#).

Method print(): print

Usage:

```
matrixZeroBasedView$print()
```

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
matrixZeroBasedView$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

PlatformConfig	Platform Configuration
----------------	------------------------

Description

An R6 mapping type for configuring various “parameters” for multiple “platforms”, essentially serves a multi-nested map where the inner map is a [ScalarMap](#) contained within a [ConfigList](#) (middle map): {platform: {param: {key: value}}}

Super class

[tiledbsoma::MappingBase](#) -> PlatformConfig

Methods

Public methods:

- [PlatformConfig\\$platforms\(\)](#)
- [PlatformConfig\\$params\(\)](#)
- [PlatformConfig\\$get\(\)](#)
- [PlatformConfig\\$get_params\(\)](#)
- [PlatformConfig\\$set\(\)](#)
- [PlatformConfig\\$setv\(\)](#)
- [PlatformConfig\\$clone\(\)](#)

Method platforms():

Usage:

```
PlatformConfig$platforms()
```

Returns: The names of the “platforms” (outer keys)

Method params():

Usage:

```
PlatformConfig$params(platform = NULL)
```

Arguments:

platform The “platform” to pull parameter names (middle keys) for; pass TRUE to return all possible parameter names

Returns: The parameter names (middle keys) for platform

Method get():

Usage:

```
PlatformConfig$get(  
  platform,  
  param = NULL,  
  key = NULL,  
  default = quote(expr = )  
)
```

Arguments:

platform The name of the “platform” (outer key) to fetch

param The name of the “parameters” of platform to fetch; if NULL, returns the [configuration](#) for platform

key The “key” (inner key) for param in platform to fetch; if NULL and param is passed, returns the [map](#) for param in platform

default Default value to fetch if key is not found; defaults to null

Returns: The value of key for param in platform in the map, or default if key is not found

Method get_params():*Usage:*

PlatformConfig\$get_params(platform)

Arguments:

platform The name of the “platform” (outer key) to fetch

Returns: The [ConfigList](#) for platform

Method set():*Usage:*

PlatformConfig\$set(platform, param, key, value)

Arguments:

platform The name of the “platform” (outer key) to set

param Name of the “parameter” (middle key) in platform to set

key Inner key to set

value Value to add for key, or NULL to remove the entry for key; optionally provide only platform, param, and value as a [ScalarMap](#) to update param for platform with the keys and values from value

Returns: `\[chainable\]` Invisibly returns self with value added for key in param for platform

Method setv():*Usage:*

PlatformConfig\$setv(...)

Arguments:

... Ignored

Returns: Nothing; setv() is disabled for PlatformConfig objects

Method clone(): The objects of this class are cloneable with this method.*Usage:*

PlatformConfig\$clone(deep = FALSE)

Arguments:

deep Whether to make a deep clone.

show_package_versions *Display package versions*

Description

Print version information for **tiledb** (R package), libtiledb_soma, and TileDB embedded, suitable for assisting with bug reports.

Usage

```
show_package_versions()
```

SOMAAxisQuery *SOMA Axis Query*

Description

Construct a single-axis query object with a combination of coordinates and/or value filters for use with [SOMAXperimentAxisQuery](#). (lifecycle: experimental)

Per dimension, the SOMAAxisQuery can have value of:

- None (i.e., coords = NULL and value_filter = NULL) - read all values
- Coordinates - a set of coordinates on the axis dataframe index, expressed in any type or format supported by [SOMADataFrame](#)'s read() method.
- A SOMA value_filter across columns in the axis dataframe, expressed as string
- Or, a combination of coordinates and value filter.

Public fields

coords The coordinates for the query.

value_filter The value filter for the query.

Methods

Public methods:

- [SOMAAxisQuery\\$new\(\)](#)
- [SOMAAxisQuery\\$clone\(\)](#)

Method new(): Create a new SOMAAxisQuery object.

Usage:

```
SOMAAxisQuery$new(value_filter = NULL, coords = NULL)
```

Arguments:

value_filter Optional string containing a logical expression that is used to filter the returned values.

coords Optional indices specifying the rows to read: either a vector of the appropriate type or a named list of vectors corresponding to each dimension.

Method clone(): The objects of this class are cloneable with this method.

Usage:

SOMAAxisQuery\$clone(deep = FALSE)

Arguments:

deep Whether to make a deep clone.

See Also

[tiledb::parse_query_condition\(\)](#) for more information about valid value filters.

SOMAAxisQueryResult SOMAExperiment Axis Query Result

Description

Access [SOMAExperimentAxisQuery](#) results.

Active bindings

obs [arrow::Table](#) containing obs query slice.

var [arrow::Table](#) containing var query slice. measurement_name.

X_layers named list of [arrow::Tables](#) for each X layer.

Methods

Public methods:

- [SOMAAxisQueryResult\\$new\(\)](#)
- [SOMAAxisQueryResult\\$clone\(\)](#)

Method new(): Create a new SOMAAxisQueryResult object.

Usage:

SOMAAxisQueryResult\$new(obs, var, X_layers)

Arguments:

obs, var [arrow::Table](#) containing obs or var query slice.

X_layers named list of [arrow::Tables](#), one for each X layer.

Method clone(): The objects of this class are cloneable with this method.

Usage:

SOMAAxisQueryResult\$clone(deep = FALSE)

Arguments:

deep Whether to make a deep clone.

SOMACollection	<i>SOMA Collection</i>
----------------	------------------------

Description

Contains a key-value mapping where the keys are string names and the values are any SOMA-defined foundational or composed type, including [SOMACollection](#), [SOMADataFrame](#), [SOMADenseNDArray](#), [SOMASparseNDArray](#), or [SOMAExperiment](#). (lifecycle: experimental)

Super classes

```

tiledbsoma::TileDBObject -> tiledbsoma::TileDBGGroup -> tiledbsoma::SOMACollectionBase
-> SOMACollection

```

Methods**Public methods:**

- [SOMACollection\\$clone\(\)](#)

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
SOMACollection$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

SOMACollectionCreate	<i>Create SOMA Collection</i>
----------------------	-------------------------------

Description

Factory function to create a SOMADataFrame for writing, (lifecycle: experimental)

Usage

```

SOMACollectionCreate(
  uri,
  platform_config = NULL,
  tiledbsoma_ctx = NULL,
  tiledb_timestamp = NULL
)

```

Arguments

uri	URI for the TileDB object
platform_config	Optional platform configuration
tiledb_soma_ctx	Optional SOMATileDBContext
tiledb_timestamp	Optional Datetime (POSIXct) for TileDB timestamp

SOMACollectionOpen *Open SOMA Collection*

Description

Factory function to open a SOMACollection for reading, (lifecycle: experimental)

Usage

```
SOMACollectionOpen(
  uri,
  mode = "READ",
  platform_config = NULL,
  tiledb_soma_ctx = NULL,
  tiledb_timestamp = NULL
)
```

Arguments

uri	URI for the TileDB object
mode	One of "READ" or "WRITE"
platform_config	Optional platform configuration
tiledb_soma_ctx	optional SOMATileDBContext
tiledb_timestamp	Optional Datetime (POSIXct) for TileDB timestamp. In READ mode, defaults to the current time. If non-NULL, then all members accessed through the collection object inherit the timestamp.

SOMADataFrame

*SOMADataFrame***Description**

SOMADataFrame is a multi-column table that must contain a column called `soma_joinid` of type `int64`, which contains a unique value for each row and is intended to act as a join key for other objects, such as [SOMASparseNDArray](#). (lifecycle: experimental)

Super classes

```

tiledbsoma::TileDBObject -> tiledbsoma::TileDBArray -> tiledbsoma::SOMAArrayBase -
> SOMADataFrame

```

Methods**Public methods:**

- [SOMADataFrame\\$create\(\)](#)
- [SOMADataFrame\\$write\(\)](#)
- [SOMADataFrame\\$read\(\)](#)
- [SOMADataFrame\\$clone\(\)](#)

Method `create()`: Create (lifecycle: experimental)

Usage:

```

SOMADataFrame$create(
  schema,
  index_column_names = c("soma_joinid"),
  platform_config = NULL,
  internal_use_only = NULL
)

```

Arguments:

`schema` an [arrow::schema](#).

`index_column_names` A vector of column names to use as user-defined index columns. All named columns must exist in the schema, and at least one index column name is required.

`platform_config` A [platform configuration](#) object

`internal_use_only` Character value to signal this is a 'permitted' call, as `create()` is considered internal and should not be called directly.

Method `write()`: Write (lifecycle: experimental)

Usage:

```

SOMADataFrame$write(values)

```

Arguments:

`values` An [arrow::Table](#) or [arrow::RecordBatch](#) containing all columns, including any index columns. The schema for `values` must match the schema for the `SOMADataFrame`.

Method `read()`: Read (lifecycle: experimental) Read a user-defined subset of data, addressed by the dataframe indexing column, and optionally filtered.

Usage:

```
SOMADataFrame$read(
  coords = NULL,
  column_names = NULL,
  value_filter = NULL,
  result_order = "auto",
  iterated = FALSE,
  log_level = "auto"
)
```

Arguments:

`coords` Optional named list of indices specifying the rows to read; each (named) list element corresponds to a dimension of the same name.

`column_names` Optional character vector of column names to return.

`value_filter` Optional string containing a logical expression that is used to filter the returned values. See [tiledb::parse_query_condition](#) for more information.

`result_order` Optional order of read results. This can be one of either "ROW_MAJOR", "COL_MAJOR", or "auto" (default).

`iterated` Option boolean indicated whether data is read in call (when FALSE, the default value) or in several iterated steps.

`log_level` Optional logging level with default value of "warn".

Returns: `arrow::Table` or `TableReadIter`

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
SOMADataFrame$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

SOMADataFrameCreate *Create SOMA DataFrame*

Description

Factory function to create a SOMADataFrame for writing, (lifecycle: experimental)

Usage

```
SOMADataFrameCreate(
  uri,
  schema,
  index_column_names = c("soma_joinid"),
  platform_config = NULL,
  tiledbsoma_ctx = NULL,
  tiledb_timestamp = NULL
)
```

Arguments

uri	URI for the TileDB object
schema	schema Arrow schema argument passed on to DataFrame\$create()
index_column_names	Index column names passed on to DataFrame\$create()
platform_config	Optional platform configuration
tiledbsoma_ctx	Optional SOMATileDBContext
tiledb_timestamp	Optional Datetime (POSIXct) for TileDB timestamp

SOMADataFrameOpen	<i>Open SOMA DataFrame</i>
-------------------	----------------------------

Description

Factory function to open a SOMADataFrame for reading, (lifecycle: experimental)

Usage

```
SOMADataFrameOpen(
  uri,
  mode = "READ",
  platform_config = NULL,
  tiledbsoma_ctx = NULL,
  tiledb_timestamp = NULL
)
```

Arguments

uri	URI for the TileDB object
mode	One of "READ" or "WRITE"
platform_config	Optional platform configuration
tiledbsoma_ctx	Optional SOMATileDBContext
tiledb_timestamp	Optional Datetime (POSIXct) for TileDB timestamp

SOMADenseNDArray	<i>SOMADenseNDArray</i>
------------------	-------------------------

Description

SOMADenseNDArray is a dense, N-dimensional array of primitive type, with offset (zero-based) int64 integer indexing on each dimension with domain $[0, \text{maxInt64})$. The SOMADenseNDArray has a user-defined schema, which includes:

- **type**: a primitive type, expressed as an Arrow type (e.g., int64, float32, etc), indicating the type of data contained within the array
- **shape**: the shape of the array, i.e., number and length of each dimension

All dimensions must have a positive, non-zero length, and there must be 1 or more dimensions.

The default "fill" value for SOMADenseNDArray is the zero or null value of the array type (e.g., Arrow.float32 defaults to 0.0).

The write method is currently limited to writing from 2-d matrices. (lifecycle: experimental)

Super classes

```

tildebsoma::TileDBObject -> tildebsoma::TileDBArray -> tildebsoma::SOMAArrayBase -
> tildebsoma::SOMANDArrayBase -> SOMADenseNDArray

```

Methods

Public methods:

- SOMADenseNDArray\$read_arrow_table()
- SOMADenseNDArray\$read_dense_matrix()
- SOMADenseNDArray\$write()
- SOMADenseNDArray\$clone()

Method read_arrow_table(): Read as an 'arrow::Table' (lifecycle: experimental)

Usage:

```

SOMADenseNDArray$read_arrow_table(
  coords = NULL,
  result_order = "auto",
  log_level = "warn"
)

```

Arguments:

coords Optional list of integer vectors, one for each dimension, with a length equal to the number of values to read. If NULL, all values are read. List elements can be named when specifying a subset of dimensions.

result_order Optional order of read results. This can be one of either "ROW_MAJOR", "COL_MAJOR", or "auto" (default).

`result_order` Optional order of read results. This can be one of either "ROW_MAJOR", "COL_MAJOR", or "auto" (default).

`log_level` Optional logging level with default value of "warn".

Returns: An `arrow::Table`.

Method `read_dense_matrix()`: Read as a dense matrix (lifecycle: experimental)

Usage:

```
SOMADenseNDArray$read_dense_matrix(
  coords = NULL,
  result_order = "ROW_MAJOR",
  log_level = "warn"
)
```

Arguments:

`coords` Optional list of integer vectors, one for each dimension, with a length equal to the number of values to read. If NULL, all values are read. List elements can be named when specifying a subset of dimensions.

`result_order` Optional order of read results. This can be one of either "ROW_MAJOR", "COL_MAJOR", or "auto" (default).

`result_order` Optional order of read results. This can be one of either "ROW_MAJOR", "COL_MAJOR", or "auto" (default).

`log_level` Optional logging level with default value of "warn".

Returns: A matrix object

Method `write()`: Write matrix data to the array. (lifecycle: experimental)

More general write methods for higher-dimensional array could be added.

Usage:

```
SOMADenseNDArray$write(values, coords = NULL)
```

Arguments:

`values` A matrix. Character dimension names are ignored because SOMANDArray's use integer indexing.

`coords` A list of integer vectors, one for each dimension, with a length equal to the number of values to write. If NULL, the default, the values are taken from the row and column names of values.

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
SOMADenseNDArray$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

SOMADenseNDArrayCreate

Create SOMA Dense Nd Array

Description

Factory function to create a SOMADenseNDArray for writing, (lifecycle: experimental)

Usage

```
SOMADenseNDArrayCreate(
    uri,
    type,
    shape,
    platform_config = NULL,
    tiledbsoma_ctx = NULL,
    tiledb_timestamp = NULL
)
```

Arguments

uri	URI for the TileDB object
type	An Arrow type defining the type of each element in the array.
shape	A vector of integers defining the shape of the array.
platform_config	Optional platform configuration
tiledbsoma_ctx	Optional SOMATileDBContext
tiledb_timestamp	Optional Datetime (POSIXct) for TileDB timestamp

SOMADenseNDArrayOpen *Open SOMA Dense Nd Array*

Description

Factory function to open a SOMADenseNDArray for reading, (lifecycle: experimental)

Usage

```
SOMADenseNDArrayOpen(
    uri,
    mode = "READ",
    platform_config = NULL,
    tiledbsoma_ctx = NULL,
    tiledb_timestamp = NULL
)
```

Arguments

uri	URI for the TileDB object
mode	One of "READ" or "WRITE"
platform_config	Optional platform configuration
tiledbsoma_ctx	Optional SOMATileDBContext
tiledb_timestamp	Optional Datetime (POSIXct) for TileDB timestamp

SOMAEperiment	<i>SOMA Experiment</i>
---------------	------------------------

Description

SOMAEperiment is a specialized [SOMACollection](#), representing one or more modes of measurement across a single collection of cells (aka a "multimodal dataset") with pre-defined fields: obs and ms (see *Active Bindings* below for details). (lifecycle: experimental)

Adding new objects to a collection

The [SOMAEperiment](#) class provides a number of type-specific methods for adding new a object to the collection, such as `add_new_sparse_ndarray()` and `add_new_dataframe()`. These methods will create the new object and add it as member of the SOMAEperiment. The new object will always inherit the parent context (see [SOMATileDBContext](#)) and, by default, its platform configuration (see [PlatformConfig](#)). However, the user can override the default platform configuration by passing a custom configuration to the `platform_config` argument.

Super classes

```
tiledbsoma::TileDBObject -> tiledbsoma::TileDBGroup -> tiledbsoma::SOMACollectionBase
-> SOMAEperiment
```

Active bindings

obs a [SOMADataFrame](#) containing primary annotations on the observation axis. The contents of the `soma_joinid` column define the observation index domain, `obs_id`. All observations for the SOMAEperiment must be defined in this dataframe.

ms a [SOMACollection](#) of named [SOMAMeasurements](#).

Methods**Public methods:**

- [SOMAEperiment\\$axis_query\(\)](#)
- [SOMAEperiment\\$clone\(\)](#)

Method `axis_query()`: Subset and extract data from a [SOMAMeasurement](#) by querying the obs/var axes.

Usage:

```
SOMAXperiment$axis_query(measurement_name, obs_query = NULL, var_query = NULL)
```

Arguments:

`measurement_name` The name of the measurement to query.

`obs_query`, `var_query` An [SOMAAxisQuery](#) object for the obs/var axis.

Returns: A [SOMAXperimentAxisQuery](#) object.

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
SOMAXperiment$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

SOMAXperimentAxisQuery

SOMAXperiment Axis Query

Description

Perform an axis-based query against a [SOMAXperiment](#).

[SOMAXperimentAxisQuery](#) allows easy selection and extraction of data from a single [SOMAMeasurement](#) in a [SOMAXperiment](#), by obs/var (axis) coordinates and/or value filter. The primary use for this class is slicing [SOMAXperiment](#) X layers by obs or var value and/or coordinates. (lifecycle: experimental)

X Layer Support:

Slicing on [SOMASparseNDArray](#) X matrices is supported; slicing on [SOMADenseNDArray](#) is not supported at this time.

Result Size:

[SOMAXperimentAxisQuery](#) query class assumes it can store the full result of both axis dataframe queries in memory, and only provides incremental access to the underlying X NDArray. Accessors such as `n_obs` and `n_vars` codify this in the class.

Active bindings

`experiment` The parent [SOMAXperiment](#) object.

`indexer` The [SOMAAxisIndexer](#) object.

`obs_query` The obs [SOMAAxisQuery](#) object.

`var_query` The var [SOMAAxisQuery](#) object.

`n_obs` The number of obs axis query results.

n_vars The number of var axis query results.
 obs_df The obs [SOMADataFrame](#) object.
 var_df The var [SOMADataFrame](#) object for the specified measurement_name.
 ms The [SOMAMeasurement](#) object for the specified measurement_name.

Methods

Public methods:

- [SOMAExperimentAxisQuery\\$new\(\)](#)
- [SOMAExperimentAxisQuery\\$obs\(\)](#)
- [SOMAExperimentAxisQuery\\$var\(\)](#)
- [SOMAExperimentAxisQuery\\$obs_joinids\(\)](#)
- [SOMAExperimentAxisQuery\\$var_joinids\(\)](#)
- [SOMAExperimentAxisQuery\\$X\(\)](#)
- [SOMAExperimentAxisQuery\\$read\(\)](#)
- [SOMAExperimentAxisQuery\\$to_sparse_matrix\(\)](#)
- [SOMAExperimentAxisQuery\\$to_seurat\(\)](#)
- [SOMAExperimentAxisQuery\\$to_seurat_assay\(\)](#)
- [SOMAExperimentAxisQuery\\$to_seurat_reduction\(\)](#)
- [SOMAExperimentAxisQuery\\$to_seurat_graph\(\)](#)
- [SOMAExperimentAxisQuery\\$to_single_cell_experiment\(\)](#)
- [SOMAExperimentAxisQuery\\$clone\(\)](#)

Method [new\(\)](#): Create a new [SOMAExperimentAxisQuery](#) object.

Usage:

```
SOMAExperimentAxisQuery$new(
  experiment,
  measurement_name,
  obs_query = NULL,
  var_query = NULL
)
```

Arguments:

experiment A [SOMAExperiment](#) object.
 measurement_name The name of the measurement to query.
 obs_query, var_query An [SOMAAxisQuery](#) object for the obs/var axis.

Method [obs\(\)](#): Retrieve obs [TableReadIter](#)

Usage:

```
SOMAExperimentAxisQuery$obs(column_names = NULL)
```

Arguments:

column_names A character vector of column names to retrieve

Method [var\(\)](#): Retrieve var [arrow::Table](#)

Usage:

```
SOMAExperimentAxisQuery$var(column_names = NULL)
```

Arguments:

`column_names` A character vector of column names to retrieve

Method `obs_joinids()`: Retrieve `soma_joinids` as an `arrow::Array` for `obs`.

Usage:

```
SOMAExperimentAxisQuery$obs_joinids()
```

Method `var_joinids()`: Retrieve `soma_joinids` as an `arrow::Array` for `var`.

Usage:

```
SOMAExperimentAxisQuery$var_joinids()
```

Method `X()`: Retrieves an X layer as a `linkSOMASparseNDArrayRead`

Usage:

```
SOMAExperimentAxisQuery$X(layer_name)
```

Arguments:

`layer_name` The name of the layer to retrieve.

Method `read()`: Reads the entire query result as a list of `arrow::Tables`. This is a low-level routine intended to be used by loaders for other in-core formats, such as Seurat, which can be created from the resulting Tables.

Usage:

```
SOMAExperimentAxisQuery$read(
  X_layers = NULL,
  obs_column_names = NULL,
  var_column_names = NULL
)
```

Arguments:

`X_layers` The name(s) of the X layer(s) to read and return.

`obs_column_names`, `var_column_names` Specify which column names in `var` and `obs` dataframes to read and return.

Method `to_sparse_matrix()`: Retrieve a collection layer as a sparse matrix with named dimensions.

Load any layer from the `X`, `obsm`, `varm`, `obsp`, or `varp` collections as a [sparse matrix](#).

By default the matrix dimensions are named using the `soma_joinid` values in the specified layer's dimensions (e.g., `soma_dim_0`). However, dimensions can be named using values from any `obs` or `var` column that uniquely identifies each record by specifying the `obs_index` and `var_index` arguments.

For layers in `obsm` or `varm`, the column axis (the axis not indexed by “`obs`” or “`var`”) is set to the range of values present in “`soma_dim_1`”; this ensures that gaps in this axis are preserved (eg. when a query for “`obs`” that results in selecting entries that are all zero for a given PC)

Usage:

```
SOMAExperimentAxisQuery$to_sparse_matrix(
  collection,
  layer_name,
  obs_index = NULL,
  var_index = NULL
)
```

Arguments:

collection The [SOMACollection](#) containing the layer of interest, either: "X", "obsm", "varm", "obsp", or "varp".

layer_name Name of the layer to retrieve from the collection.

obs_index, var_index Name of the column in obs or var (var_index) containing values that should be used as dimension labels in the resulting matrix. Whether the values are used as row or column labels depends on the selected collection:

Collection	obs_index	var_index
X	row names	column names
obsm	row names	ignored
varm	ignored	row names
obsp	row and column names	ignored
varp	ignored	row and column names

Returns: A [Matrix::sparseMatrix](#)

Method `to_seurat()`: Loads the query as a [Seurat](#) object

Usage:

```
SOMAExperimentAxisQuery$to_seurat(
  X_layers = c(counts = "counts", data = "logcounts"),
  obs_index = NULL,
  var_index = NULL,
  obs_column_names = NULL,
  var_column_names = NULL,
  obsm_layers = NULL,
  varm_layers = NULL,
  obsp_layers = NULL
)
```

Arguments:

X_layers A named character of X layers to add to the Seurat assay where the names are the names of Seurat slots and the values are the names of layers within X; names should be one of:

- "counts" to add the layer as counts
- "data" to add the layer as data
- "scale.data" to add the layer as scale.data

At least one of "counts" or "data" is required

obs_index Name of column in obs to add as cell names; uses `paste0("cell", obs_joinids())` by default

`var_index` Name of column in `var` to add as feature names; uses `paste0("feature", var_joinids())` by default
`obs_column_names` Names of columns in `obs` to add as cell-level meta data; by default, loads all columns
`var_column_names` Names of columns in `var` to add as feature-level meta data; by default, loads all columns
`obsm_layers` Names of arrays in `obsm` to add as the cell embeddings; pass `FALSE` to suppress loading in any dimensional reductions; by default, loads all dimensional reduction information
`varm_layers` Named vector of arrays in `varm` to load in as the feature loadings; names must be names of arrays in `obsm` (eg. `varm_layers = c(X_pca = "PCs")`); pass `FALSE` to suppress loading in any feature loadings; will try to determine `varm_layers` from `obsm_layers`
`obsp_layers` Names of arrays in `obsp` to load in as [Graphs](#); by default, loads all graphs
Returns: A [Seurat](#) object

Method `to_seurat_assay()`: Loads the query as a Seurat [Assay](#)

Usage:

```

SOMAExperimentAxisQuery$to_seurat_assay(
  X_layers = c(counts = "counts", data = "logcounts"),
  obs_index = NULL,
  var_index = NULL,
  var_column_names = NULL
)
  
```

Arguments:

`X_layers` A named character of X layers to add to the Seurat assay where the names are the names of Seurat slots and the values are the names of layers within X; names should be one of:

- “counts” to add the layer as counts
- “data” to add the layer as data
- “scale.data” to add the layer as scale.data

At least one of “counts” or “data” is required

`obs_index` Name of column in `obs` to add as cell names; uses `paste0("cell", obs_joinids())` by default
`var_index` Name of column in `var` to add as feature names; uses `paste0("feature", var_joinids())` by default
`var_column_names` Names of columns in `var` to add as feature-level meta data; by default, loads all columns
Returns: An [Assay](#) object

Method `to_seurat_reduction()`: Loads the query as a Seurat [dimensional reduction](#)

Usage:

```

SOMAExperimentAxisQuery$to_seurat_reduction(
  obsm_layer,
  varm_layer = NULL,
  obs_index = NULL,
  var_index = NULL
)
  
```


Arguments:

`obsm_layer` Name of array in obsm to load as the cell embeddings

`varm_layer` Name of the array in varm to load as the feature loadings; by default, will try to determine `varm_layer` from `obsm_layer`

`obs_index` Name of column in obs to add as cell names; uses `paste0("cell", obs_joinids())` by default

`var_index` Name of column in var to add as feature names; uses `paste0("feature", var_joinids())` by default

Returns: A [DimReduc](#) object

Method `to_seurat_graph()`: Loads the query as a Seurat [graph](#)

Usage:

```
SOMAExperimentAxisQuery$to_seurat_graph(obsp_layer, obs_index = NULL)
```

Arguments:

`obsp_layer` Name of array in obsp to load as the graph

`obs_index` Name of column in obs to add as cell names; uses `paste0("cell", obs_joinids())` by default

Returns: A [Graph](#) object

Method `to_single_cell_experiment()`: Loads the query as a [SingleCellExperiment](#) object

Usage:

```
SOMAExperimentAxisQuery$to_single_cell_experiment(
  X_layers = NULL,
  obs_index = NULL,
  var_index = NULL,
  obs_column_names = NULL,
  var_column_names = NULL,
  obsm_layers = NULL,
  obsp_layers = NULL,
  varp_layers = NULL
)
```

Arguments:

`X_layers` A character vector of X layers to add as assays in the main experiment; may optionally be named to set the name of the resulting assay (eg. `X_layers = c(counts = "raw")` will load in X layer "raw" as assay "counts"); by default, loads in all X layers

`obs_index` Name of column in obs to add as cell names; uses `paste0("cell", obs_joinids())` by default

`var_index` Name of column in var to add as feature names; uses `paste0("feature", var_joinids())` by default

`obs_column_names` Names of columns in obs to add as colData; by default, loads all columns

`var_column_names` Names of columns in var to add as rowData; by default, loads all columns

`obsm_layers` Names of arrays in obsm to add as the reduced dimensions; pass `FALSE` to suppress loading in any reduced dimensions; by default, loads all reduced dimensions

`obsp_layers` Names of arrays in obsp to load in as [SelfHits](#); by default, loads all graphs

varp_layers Names of arrays in varp to load in as [SelfHits](#); by default, loads all networks

Returns: A [SingleCellExperiment](#) object

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
SOMAExperimentAxisQuery$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

SOMAExperimentCreate *Create SOMA Experiment*

Description

Factory function to create a SOMADataFrame for writing, (lifecycle: experimental)

Usage

```
SOMAExperimentCreate(
  uri,
  platform_config = NULL,
  tiledbsoma_ctx = NULL,
  tiledb_timestamp = NULL
)
```

Arguments

uri URI for the TileDB object

platform_config Optional platform configuration

tiledbsoma_ctx Optional SOMATileDBContext

tiledb_timestamp Optional Datetime (POSIXct) for TileDB timestamp

SOMAExperimentOpen *Open SOMA Experiment*

Description

Factory function to open a SOMAExperiment for reading, (lifecycle: experimental)

Usage

```
SOMAExperimentOpen(
    uri,
    mode = "READ",
    platform_config = NULL,
    tiledbsoma_ctx = NULL,
    tiledb_timestamp = NULL
)
```

Arguments

uri	URI for the TileDB object
mode	One of "READ" or "WRITE"
platform_config	Optional platform configuration
tiledbsoma_ctx	optional SOMATileDBContext
tiledb_timestamp	Optional Datetime (POSIXct) for TileDB timestamp. In READ mode, defaults to the current time. If non-NULL, then all members accessed through the collection object inherit the timestamp.

SOMAMeasurement *SOMA Measurement*

Description

A SOMAMeasurement is a sub-element of a [SOMAExperiment](#), and is otherwise a specialized [SOMACollection](#) with pre-defined fields: X, var, obsm/varm, and obsp/varp (see *Active Bindings* below for details). (lifecycle: experimental)

Adding new objects to a collection

The [SOMAMeasurement](#) class provides a number of type-specific methods for adding new a object to the collection, such as `add_new_sparse_ndarray()` and `add_new_dataframe()`. These methods will create the new object and add it as member of the SOMAMeasurement. The new object will always inherit the parent context (see [SOMATileDBContext](#)) and, by default, its platform configuration (see [PlatformConfig](#)). However, the user can override the default platform configuration by passing a custom configuration to the `platform_config` argument.

Super classes

```

tiledbsoma::TileDBObject -> tiledbsoma::TileDBGroup -> tiledbsoma::SOMACollectionBase
-> SOMAMeasurement

```

Active bindings

var a [SOMADataFrame](#) containing primary annotations on the variable axis, for variables in this measurement (i.e., annotates columns of X). The contents of the `soma_joinid` column define the variable index domain, `var_id`. All variables for this measurement must be defined in this dataframe.

X a [SOMACollection](#) of [SOMASparseNDArrays](#), each contains measured feature values indexed by `[obsid, varid]`.

obsm a [SOMACollection](#) of [SOMADenseNDArrays](#) containing annotations on the observation axis. Each array is indexed by `obsid` and has the same shape as `obs`.

obsp a [SOMACollection](#) of [SOMASparseNDArrays](#) containing pairwise annotations on the observation axis and indexed with `[obsid_1, obsid_2]`.

varm a [SOMACollection](#) of [SOMADenseNDArrays](#) containing annotations on the variable axis. Each array is indexed by `varid` and has the same shape as `var`.

varp a [SOMACollection](#) of [SOMASparseNDArrays](#) containing pairwise annotations on the variable axis and indexed with `[varid_1, varid_2]`.

Methods**Public methods:**

- [SOMAMeasurement\\$clone\(\)](#)

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
SOMAMeasurement$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

SOMAMeasurementCreate *Create SOMA Measurement*

Description

Factory function to create a SOMAMeasurement for writing, (lifecycle: experimental)

Usage

```

SOMAMeasurementCreate(
  uri,
  platform_config = NULL,
  tiledbsoma_ctx = NULL,
  tiledb_timestamp = NULL
)

```

Arguments

uri	URI for the TileDB object
platform_config	Optional platform configuration
tiledb_soma_ctx	Optional SOMATileDBContext
tiledb_timestamp	Optional Datetime (POSIXct) for TileDB timestamp

SOMAMeasurementOpen *Open SOMA Measurement*

Description

Factory function to open a SOMAMeasurement for reading, (lifecycle: experimental)

Usage

```
SOMAMeasurementOpen(
  uri,
  mode = "READ",
  platform_config = NULL,
  tiledb_soma_ctx = NULL,
  tiledb_timestamp = NULL
)
```

Arguments

uri	URI for the TileDB object
mode	One of "READ" or "WRITE"
platform_config	Optional platform configuration
tiledb_soma_ctx	optional SOMATileDBContext
tiledb_timestamp	Optional Datetime (POSIXct) for TileDB timestamp. In READ mode, defaults to the current time. If non-NULL, then all members accessed through the collection object inherit the timestamp.

SOMAOpen	<i>Open a SOMA Object</i>
----------	---------------------------

Description

Utility function to open the corresponding SOMA Object given a URI, (lifecycle: experimental)

Usage

```
SOMAOpen(
  uri,
  mode = "READ",
  platform_config = NULL,
  tiledbsoma_ctx = NULL,
  tiledb_timestamp = NULL
)
```

Arguments

uri	URI for the TileDB object
mode	One of "READ" or "WRITE"
platform_config	Optional platform configuration
tiledbsoma_ctx	Optional SOMATileDBContext
tiledb_timestamp	Optional Datetime (POSIXct) with TileDB timestamp. For SOMACollections, all accessed members inherit the collection opening timestamp, and in READ mode the collection timestamp defaults to the time of opening.

SOMASparseNDArray	<i>SOMASparseNDArray</i>
-------------------	--------------------------

Description

SOMASparseNDArray is a sparse, N-dimensional array with offset (zero-based) integer indexing on each dimension. The SOMASparseNDArray has a user-defined schema, which includes:

- type - a primitive type, expressed as an Arrow type (e.g., int64, float32, etc)
- shape - the shape of the array, i.e., number and length of each dimension

All dimensions must have a positive, non-zero length.

Note - on TileDB this is an sparse array with N int64 dimensions of domain [0, maxInt64), and a single attribute.

Duplicate writes:

As duplicate index values are not allowed, index values already present in the object are overwritten and new index values are added. (lifecycle: experimental)

Super classes

```

 tiledbsoma::TileDBObject -> tiledbsoma::TileDBArray -> tiledbsoma::SOMAArrayBase -
 > tiledbsoma::SOMANDArrayBase -> SOMASparseNDArray

```

Methods**Public methods:**

- [SOMASparseNDArray\\$read\(\)](#)
- [SOMASparseNDArray\\$write\(\)](#)
- [SOMASparseNDArray\\$nnz\(\)](#)
- [SOMASparseNDArray\\$clone\(\)](#)

Method [read\(\)](#): Reads a user-defined slice of the SOMASparseNDArray

Usage:

```

SOMASparseNDArray$read(
  coords = NULL,
  result_order = "auto",
  log_level = "auto"
)

```

Arguments:

coords Optional list of integer vectors, one for each dimension, with a length equal to the number of values to read. If NULL, all values are read. List elements can be named when specifying a subset of dimensions.

result_order Optional order of read results. This can be one of either "ROW_MAJOR", "COL_MAJOR", or "auto" (default).

log_level Optional logging level with default value of "warn".

iterated Option boolean indicated whether data is read in call (when FALSE, the default value) or in several iterated steps.

Returns: [SOMASparseNDArrayRead](#)

Method [write\(\)](#): Write matrix-like data to the array. (lifecycle: experimental)

Usage:

```

SOMASparseNDArray$write(values)

```

Arguments:

values Any matrix-like object coercible to a [TsparseMatrix](#). Character dimension names are ignored because SOMANDArray's use integer indexing.

Method [nnz\(\)](#): Retrieve number of non-zero elements (lifecycle: experimental)

Usage:

```

SOMASparseNDArray$nnz()

```

Returns: A scalar with the number of non-zero elements

Method [clone\(\)](#): The objects of this class are cloneable with this method.

Usage:

```
SOMASparseNDArray$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

SOMASparseNDArrayCreate

Create SOMA Sparse Nd Array

Description

Factory function to create a SOMASparseNDArray for writing, (lifecycle: experimental)

Usage

```
SOMASparseNDArrayCreate(
  uri,
  type,
  shape,
  platform_config = NULL,
  tiledbsoma_ctx = NULL,
  tiledb_timestamp = NULL
)
```

Arguments

uri	URI for the TileDB object
type	An Arrow type defining the type of each element in the array.
shape	A vector of integers defining the shape of the array.
platform_config	Optional platform configuration
tiledbsoma_ctx	Optional SOMATileDBContext
tiledb_timestamp	Optional Datetime (POSIXct) for TileDB timestamp

SOMASparseNDArrayOpen *Open SOMA Sparse Nd Array*

Description

Factory function to open a SOMASparseNDArray for reading, (lifecycle: experimental)

Usage

```
SOMASparseNDArrayOpen(
  uri,
  mode = "READ",
  platform_config = NULL,
  tiledbsoma_ctx = NULL,
  tiledb_timestamp = NULL
)
```

Arguments

uri	URI for the TileDB object
mode	One of "READ" or "WRITE"
platform_config	Optional platform configuration
tiledbsoma_ctx	Optional SOMATileDBContext
tiledb_timestamp	Optional Datetime (POSIXct) for TileDB timestamp

SOMATileDBContext *SOMA TileDB Context*

Description

Context map for TileDB-backed SOMA objects

Super classes

```
tiledbsoma::MappingBase -> tiledbsoma::ScalarMap -> tiledbsoma::SOMAContextBase ->
SOMATileDBContext
```

Methods

Public methods:

- `SOMATileDBContext$new()`
- `SOMATileDBContext$keys()`
- `SOMATileDBContext$items()`
- `SOMATileDBContext$length()`
- `SOMATileDBContext$get()`
- `SOMATileDBContext$set()`
- `SOMATileDBContext$to_tiledb_context()`
- `SOMATileDBContext$context()`
- `SOMATileDBContext$clone()`

Method `new()`:

Usage:

```
SOMATileDBContext$new(config = NULL, cached = TRUE)
```

Arguments:

`config` ...

`cached` Force new creation

Returns: An instantiated SOMATileDBContext object

Method `keys()`:

Usage:

```
SOMATileDBContext$keys()
```

Returns: The keys of the map

Method `items()`:

Usage:

```
SOMATileDBContext$items()
```

Returns: Return the items of the map as a list

Method `length()`:

Usage:

```
SOMATileDBContext$length()
```

Returns: The number of items in the map

Method `get()`:

Usage:

```
SOMATileDBContext$get(key, default = quote(expr = ))
```

Arguments:

`key` Key to fetch

`default` Default value to fetch if key is not found; defaults to NULL

Returns: The value of key in the map, or `default` if key is not found

Method set():

Usage:

```
SOMATileDBContext$set(key, value)
```

Arguments:

key Key to set

value Value to add for key, or NULL to remove the entry for key

Returns: [chainable] Invisibly returns self with value added as key

Method to_tiledb_context():

Usage:

```
SOMATileDBContext$to_tiledb_context()
```

Returns: A [tiledb_ctx](#) object, dynamically constructed. Most useful for the constructor of this class.

Method context():

Usage:

```
SOMATileDBContext$context()
```

Returns: A [tiledb_ctx](#) object, which is a stored (and long-lived) result from `to_tiledb_context`.

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
SOMATileDBContext$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

SparseReadIter

SparseReadIter

Description

SparseReadIter is a class that allows for iteration over a reads on [SOMASparseNDArray](#). Iteration chunks are retrieved as 0-based Views [matrixZeroBasedView](#) of `Matrix::sparseMatrix`.

Super class

[tiledbsoma::ReadIter](#) -> SparseReadIter

Methods

Public methods:

- [SparseReadIter\\$new\(\)](#)
- [SparseReadIter\\$concat\(\)](#)
- [SparseReadIter\\$clone\(\)](#)

Method `new()`: Create (lifecycle: experimental)

Usage:

```
SparseReadIter$new(sr, shape, zero_based = FALSE)
```

Arguments:

sr Soma reader pointer

shape Shape of the full matrix

zero_based Logical, if TRUE will make iterator for [Matrix::dgTMatrix-class](#) otherwise [matrixZeroBasedView](#).

Method `concat()`: Concatenate remainder of iterator.

Usage:

```
SparseReadIter$concat()
```

Returns: [matrixZeroBasedView](#) of [Matrix::sparseMatrix](#)

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
SparseReadIter$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

TableReadIter

SOMA Read Iterator over Arrow Table

Description

TableReadIter is a class that allows for iteration over a reads on [SOMASparseNDArray](#) and [SOMADataFrame](#). Iteration chunks are retrieved as [arrow::Table](#)

Super class

[tiledbsoma::ReadIter](#) -> TableReadIter

Methods**Public methods:**

- [TableReadIter\\$concat\(\)](#)
- [TableReadIter\\$clone\(\)](#)

Method `concat()`: Concatenate remainder of iterator.

Usage:

`TableReadIter$concat()`

Returns: `arrow::Table`

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

`TableReadIter$clone(deep = FALSE)`

Arguments:

`deep` Whether to make a deep clone.

TileDBObject

TileDB Object Base Class

Description

Base class to implement shared functionality across the `TileDBArray` and `TileDBGroup` classes. (lifecycle: experimental)

Active bindings

`platform_config` Platform configuration

`tiledb_soma_ctx` SOMATileDBContext

`uri` The URI of the TileDB object.

Methods**Public methods:**

- [TileDBObject\\$new\(\)](#)
- [TileDBObject\\$class\(\)](#)
- [TileDBObject\\$is_open\(\)](#)
- [TileDBObject\\$mode\(\)](#)
- [TileDBObject\\$print\(\)](#)
- [TileDBObject\\$exists\(\)](#)
- [TileDBObject\\$clone\(\)](#)

Method `new()`: Create a new TileDB object. (lifecycle: experimental)

Usage:

```
TileDBObject$new(
  uri,
  platform_config = NULL,
  tiledbsoma_ctx = NULL,
  tiledb_timestamp = NULL,
  internal_use_only = NULL
)
```

Arguments:

uri URI for the TileDB object
platform_config Optional platform configuration
tiledbsoma_ctx Optional SOMATileDBContext
tiledb_timestamp Optional Datetime (POSIXct) with TileDB timestamp
internal_use_only Character value to signal this is a 'permitted' call, as new() is considered internal and should not be called directly.

Method class(): Print the name of the R6 class.

Usage:

```
TileDBObject$class()
```

Method is_open(): Determine if the object is open for reading or writing

Usage:

```
TileDBObject$is_open()
```

Returns: TRUE if the object is open, otherwise FALSE

Method mode(): Get the mode of the object

Usage:

```
TileDBObject$mode()
```

Returns: If the object is closed, returns "CLOSED"; otherwise returns the mode (eg. "READ") of the object

Method print(): Print-friendly representation of the object.

Usage:

```
TileDBObject$print()
```

Method exists(): Check if the object exists. (lifecycle: experimental)

Usage:

```
TileDBObject$exists()
```

Returns: TRUE if the object exists, FALSE otherwise.

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
TileDBObject$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

tiledbsoma_stats	<i>TileDB SOMA statistics</i>
------------------	-------------------------------

Description

These functions expose the TileDB Core functionality for performance measurements and statistics.

Usage

```
tiledbsoma_stats_enable()
tiledbsoma_stats_disable()
tiledbsoma_stats_reset()
tiledbsoma_stats_dump()
tiledbsoma_stats_show()
```

Details

- tiledbsoma_stats_enable()/tiledbsoma_stats_disable(): Enable and disable TileDB's internal statistics.
- tiledbsoma_stats_reset(): Reset all statistics to 0.
- tiledbsoma_stats_dump(): Dump all statistics to a JSON string.
- tiledbsoma_stats_show(): Print all statistics to the console.

write_soma	<i>Write a SOMA Object from an R Object</i>
------------	---

Description

Convert R objects to their appropriate SOMA counterpart function and methods can be written for it to provide a high-level $R \rightarrow$ SOMA interface

Usage

```
write_soma(x, uri, ..., platform_config = NULL, tiledbsoma_ctx = NULL)
```

Arguments

x	An object
uri	URI for resulting SOMA object
...	Arguments passed to other methods
platform_config	Optional platform configuration
tiledbsoma_ctx	Optional SOMATileDBContext

Value

The URI to the resulting [SOMAExperiment](#) generated from the data contained in `x`, returned opened for write

Known methods

- [Writing Seurat objects](#)

write_soma.Seurat	Write a Seurat object to a SOMA, returned opened for write
-------------------	--

Description

Write a [Seurat](#) object to a SOMA, returned opened for write

Usage

```
## S3 method for class 'Seurat'
write_soma(x, uri, ..., platform_config = NULL, tiledbsoma_ctx = NULL)
```

Arguments

<code>x</code>	A Seurat object
<code>uri</code>	URI for resulting SOMA object
<code>...</code>	Arguments passed to other methods
<code>platform_config</code>	Optional platform configuration
<code>tiledbsoma_ctx</code>	Optional SOMATileDBContext

Value

The URI to the resulting [SOMAExperiment](#) generated from the data contained in `x`, returned opened for write

Writing Cell-Level Meta Data

Cell-level meta data is written out as a [data frame](#) called “obs” at the [experiment](#) level

Writing Assays

[Seurat Assay](#) objects are written out as individual [measurements](#):

- the “data” matrix is written out a [sparse matrix](#) called “data” within the “X” group
- the “counts” matrix, if not [empty](#), is written out a [sparse matrix](#) called “counts” within the “X” group

- the “scale.data” matrix, if not `empty`, is written out as a `sparse matrix` called “scale_data” within the “X” group
- feature-level meta data is written out as a `data frame` called “var”

Expression matrices are transposed (cells as rows) prior to writing. All other slots, including results from extended assays (eg. SCTAssay, ChromatinAssay) are lost

Writing DimReducs

Seurat DimReduc objects are written out to the “obsm” and “varm” groups of a `measurement`:

- cell embeddings are written out as a `sparse matrix` in the “obsm” group
- feature loadings, if not `empty`, are written out as a `sparse matrix` in the “varm” groups; loadings are padded with NAs to include all features

Dimensional reduction names are translated to AnnData-style names (eg. “pca” becomes X_{pca} for embeddings and “PCs” for loadings). All other slots, including projected feature loadings and jackstraw information, are lost

Writing Graphs

Seurat Graph objects are written out as `sparse matrices` to the “obsp” group of a `measurement`

Index

Arrow type, [18, 32](#)
arrow::Array, [22](#)
arrow::RecordBatch, [13](#)
arrow::schema, [13](#)
arrow::Table, [10, 13, 17, 21, 22](#)
Assay, [24, 40](#)

ConfigList, [3, 7, 8](#)
configuration, [8](#)

data frame, [40, 41](#)
dgTMatrix-class, [36](#)
dimensional reduction, [24](#)
DimReduc, [25, 41](#)

empty, [40, 41](#)
example-datasets, [4](#)
experiment, [40](#)
extract_dataset (example-datasets), [4](#)

Graph, [24, 25, 41](#)
graph, [25](#)

list_datasets (example-datasets), [4](#)
load_dataset (example-datasets), [4](#)

map, [3, 8](#)
Matrix, [5](#)
matrix, [5](#)
Matrix::sparseMatrix, [23](#)
matrixZeroBasedView, [5, 5, 6, 35, 36](#)
measurement, [41](#)
measurements, [40](#)

platform configuration, [13, 39, 40](#)
PlatformConfig, [7, 19, 27](#)

ScalarMap, [3, 7, 8](#)
SelfHits, [25, 26](#)
Seurat, [23, 24, 40](#)
show_package_versions, [9](#)

SingleCellExperiment, [25, 26](#)
SOMAAxisIndexer, [20](#)
SOMAAxisQuery, [9, 20, 21](#)
SOMAAxisQueryResult, [10](#)
SOMACollection, [11, 11, 19, 23, 27, 28](#)
SOMACollectionCreate, [11](#)
SOMACollectionOpen, [12](#)
SOMADataFrame, [9, 11, 13, 19, 21, 28, 36](#)
SOMADataFrameCreate, [14](#)
SOMADataFrameOpen, [15](#)
SOMADenseNDArray, [11, 16, 20, 28](#)
SOMADenseNDArrayCreate, [18](#)
SOMADenseNDArrayOpen, [18](#)
SOMAExperiment, [11, 19, 19, 20, 21, 27, 40](#)
SOMAExperimentAxisQuery, [9, 10, 20, 20](#)
SOMAExperimentCreate, [26](#)
SOMAExperimentOpen, [27](#)
SOMAMeasurement, [19–21, 27, 27](#)
SOMAMeasurementCreate, [28](#)
SOMAMeasurementOpen, [29](#)
SOMAOpen, [30](#)
SOMASparseNDArray, [11, 13, 20, 28, 30, 35, 36](#)
SOMASparseNDArrayCreate, [32](#)
SOMASparseNDArrayOpen, [33](#)
SOMASparseNDArrayRead, [31](#)
SOMATileDBContext, [19, 27, 33, 39, 40](#)
sparse matrices, [41](#)
sparse matrix, [22, 40, 41](#)
sparseMatrix, [5, 35, 36](#)
SparseReadIter, [35](#)

Table, [14, 36, 37](#)
TableReadIter, [14, 21, 36](#)
tiledb::parse_query_condition, [14](#)
tiledb::parse_query_condition(), [10](#)
tiledb_ctx, [35](#)
TileDBObject, [37](#)
tiledbsoma::MappingBase, [3, 7, 33](#)
tiledbsoma::ReadIter, [35, 36](#)
tiledbsoma::ScalarMap, [33](#)

tiledsoma::SOMAArrayBase, [13](#), [16](#), [31](#)
tiledsoma::SOMACollectionBase, [11](#), [19](#),
[28](#)
tiledsoma::SOMAContextBase, [33](#)
tiledsoma::SOMANDArrayBase, [16](#), [31](#)
tiledsoma::TileDBArray, [13](#), [16](#), [31](#)
tiledsoma::TileDBGroup, [11](#), [19](#), [28](#)
tiledsoma::TileDBObject, [11](#), [13](#), [16](#), [19](#),
[28](#), [31](#)
tiledsoma_stats, [39](#)
tiledsoma_stats_disable
 (tiledsoma_stats), [39](#)
tiledsoma_stats_dump
 (tiledsoma_stats), [39](#)
tiledsoma_stats_enable
 (tiledsoma_stats), [39](#)
tiledsoma_stats_reset
 (tiledsoma_stats), [39](#)
tiledsoma_stats_show
 (tiledsoma_stats), [39](#)
TsparseMatrix, [31](#)

write_soma, [39](#)
write_soma.Seurat, [40](#)
Writing Seurat objects, [40](#)