Package 'CDSE'

September 2, 2025

Type Package

Title 'Copernicus Data Space Ecosystem' API Wrapper

Version 0.3.0

Description Provides interface to the 'Copernicus Data Space Ecosystem' API <https://dataspace.copernicus.eu/analyse/apis>, mainly for searching the catalog of available data from Copernicus Sentinel missions and obtaining the images for just the area of interest based on selected spectral bands. The package uses the 'Sentinel Hub' REST API interface <https://dataspace.copernicus.eu/analyse/apis/sentinel-hub> that provides access to various satellite imagery archives. It allows you to access raw satellite data, rendered images, statistical analysis, and other features.
This package is in no way officially related to or endorsed by Copernicus.

Depends R (>= 3.6.0)

Imports geojsonsf, grDevices, httr2, jsonlite, lubridate, lutz, sf, stats, terra, utils

URL https://zivankaraman.github.io/CDSE/, https://github.com/zivankaraman/CDSE

BugReports https://github.com/zivankaraman/CDSE/issues

License AGPL-3

Encoding UTF-8

Config/build/clean-inst-doc FALSE

RoxygenNote 7.3.2

Suggests maps, parallel, rsi, tibble

NeedsCompilation no

Author Zivan Karaman [aut, cre, cph] (ORCID: <https://orcid.org/0000-0002-8933-4589>)

Maintainer Zivan Karaman <zivan.karaman@gmail.com>

Repository CRAN

Date/Publication 2025-09-02 11:00:02 UTC

Contents

CDSE *Package providing interface to the 'Copernicus Data Space Ecosystem' API*

Description

The CDSE package for R was developed to allow access to the 'Copernicus Data Space Ecosystem' https://dataspace.copernicus.eu/ data and services from R. The 'Copernicus Data Space Ecosystem', deployed in 2023, offers access to the EO data collection from the Copernicus missions, with discovery and download capabilities and numerous data processing tools. In particular, the 'Sentinel Hub' API https://documentation.dataspace.copernicus.eu/APIs/SentinelHub.html provides access to the multi-spectral and multi-temporal big data satellite imagery service, capable of fully automated, real-time processing and distribution of remote sensing data and related EO products. Users can use APIs to retrieve satellite data over their AOI and specific time range from full archives in a matter of seconds. When working on the application of EO where the area of interest is relatively small compared to the image tiles distributed by Copernicus (100 x 100 km), it allows to retrieve just the portion of the image of interest rather than downloading the huge tile image file and processing it locally. The goal of the CDSE package is to provide easy access to this functionality from R.

The main functions allow to search the catalog of available imagery from the Sentinel-1, Sentinel-2, Sentinel-3, and Sentinel-5 missions, and to process and download the images of an area of interest and a time range in various formats. Other functions might be added in subsequent releases of the package.

API authentication

Most of the API functions require OAuth2 authentication. The recommended procedure is to obtain an authentication client object from the `GetOAuthClient` function, and to pass it as the `client` argument to the functions requiring the authentication. For more detailed information, you are invited to consult the `"Before you start"` document.

Project homepage

https://zivankaraman.github.io/CDSE/, https://github.com/zivankaraman/CDSE

Issues

For bug reports and feature requests please use the tracker https://github.com/zivankaraman/CDSE/issues

Author(s)

Zivan Karaman <zivan.karaman@gmail.com>

CDSE-deprecated *Deprecated functions in package* **CDSE**.

Description

The functions listed below are deprecated and will be defunct in the near future. When possible, alternative functions with similar functionality are also mentioned. Help pages for deprecated functions are available at `help("<function>-deprecated")`.

`GetArchiveImage`

For `GetArchiveImage`, use `GetImage`.

GetArchiveImage *Get image from the archive (deprecated)*

Description

Retrieves the image for the area of interest using the parameters provided.

Arguments

aoi	sf or sfc object, typically a (multi)polygon, describing the Area of Interest.
bbox	numeric vector of four elements describing the bounding box of interest. Specify with a coordinate pair on two (opposite) vertices of the bounding box rectangle. Coordinates need to be in longitude, latitude. Only one of either aoi or bbox may be specified.
time_range	scalar or vector (Date or character that can be converted to date) defining the time interval.
collection	character indicating which collection to search. Must be one of the collections returned by GetCollections.
script	a length one character string containing the evaluation script or the name of the file containing the script.
mosaicking_order	character indicating the order in which tiles are overlapped from which the output result is mosaicked. Must be one of "mostRecent", "leastRecent", or "leastCC". Default: "mostRecent"
file	name of the file to save the image. If NULL, a SpatRaster object is returned. Default: NULL
format	character indicating the output file format. Must be one of "image/tiff", "image/png", or "image/jpeg". Default: "image/tiff"
pixels	integer scalar or length-two vector indicating the request image width and height. Values must be integers between 1 and 2500.
resolution	numeric scalar or length-two vector indicating the spatial resolution of the request image in horizontal and vertical direction (in meters). Only one of the arguments "pixels" or "resolution" must be set at the same time. If the argument "pixels" or "resolution" is scalar, the same value is used for horizontal and vertical direction (width and height).
buffer	numeric, width of the buffer to retrieve the image of enlarged area. Default: 0
mask	logical indicating if the image should contain only pixels within Area of Interest. Default: FALSE
client	OAuth client object to use for authentication.
token	OAuth token character string to use for authentication. Exactly one of either client or token must be specified. It is recommended to use client.
url	character indicating the process endpoint. Default: Copernicus Data Space Ecosystem process endpoint

Details

If aoi argument is provided, the result is returned in the same coordinate reference system.

Value

SpatRaster object (from the package terra) of the requested image (if file is NULL), or the (invisible) name of the file created.

Source

https://documentation.dataspace.copernicus.eu/APIs/SentinelHub/Process.html

See Also

GetCollections, SearchCatalog

CDSE-deprecated

Examples

```
## Not run:
dsn <- system.file("extdata", "centralpark.geojson", package = "CDSE")
aoi <- sf::read_sf(dsn, as_tibble = FALSE)
script_file <- system.file("scripts", "NDVI_uint8.js", package = "CDSE")
day <- "2023-07-11"
ras <- GetArchiveImage(aoi = aoi, time_range = day, script = script_file,
       collection = "sentinel-2-l2a",format = "image/tiff",
       mosaicking_order = "leastCC", resolution = 10, client = OAuthClient)

## End(Not run)
```

GetCollections *List available collections*

Description

Retrieves the list of available imagery collections.

Usage

```
GetCollections(as_data_frame = TRUE, url = getOption("CDSE.catalog_url"))
```

Arguments

as_data_frame	logical indicating if the result should be returned as data frame. Default: TRUE
url	character indicating the STAC catalog search endpoint. Default: Copernicus Data Space Ecosystem STAC endpoint

Details

This function doesn't require authentication.

Value

A `list` or a `data.frame` of all available imagery collections and their attributes.

Source

https://documentation.dataspace.copernicus.eu/APIs/SentinelHub/Catalog.html

See Also

GetImage, SearchCatalog

Examples

```
## Not run:
GetCollections(as_data_frame = TRUE)

## End(Not run)
```

GetImage *Get image from the archive*

Description

Retrieves the image for the area of interest using the parameters provided.

Usage

```
GetImage(
  aoi,
  bbox,
  time_range,
  collection,
  script,
  file = NULL,
  format = c("image/tiff", "image/png", "image/jpeg"),
  mosaicking_order = c("mostRecent", "leastRecent", "leastCC"),
  pixels,
  resolution,
  buffer = 0,
  mask = FALSE,
  client,
  token,
  url = getOption("CDSE.process_url")
)
```

Arguments

aoi	sf or sfc object, typically a (multi)polygon, describing the Area of Interest.
bbox	numeric vector of four elements describing the bounding box of interest. Specify with a coordinate pair on two (opposite) vertices of the bounding box rectangle. Coordinates need to be in longitude, latitude. Only one of either aoi or bbox may be specified.
time_range	scalar or vector (Date or character that can be converted to date) defining the time interval.

`collection` character indicating which collection to search. Must be one of the collections returned by `GetCollections`.

`script` a length one character string containing the evaluation script or the name of the file containing the script.

`file` name of the file to save the image. If NULL, a `SpatRaster` object is returned. Default: NULL

`format` character indicating the output file format. Must be one of "image/tiff", "image/png", or "image/jpeg". Default: "image/tiff"

`mosaicking_order` character indicating the order in which tiles are overlapped from which the output result is mosaicked. Must be one of "mostRecent", "leastRecent", or "leastCC". Default: "mostRecent"

`pixels` integer scalar or length-two vector indicating the request image width and height. Values must be integers between 1 and 2500.

`resolution` numeric scalar or length-two vector indicating the spatial resolution of the request image in horizontal and vertical direction (in meters).

Only one of the arguments "pixels" or "resolution" must be set at the same time. If the argument "pixels" or "resolution" is scalar, the same value is used for horizontal and vertical direction (width and height).

`buffer` numeric, width of the buffer to retrieve the image of enlarged area. Default: 0

`mask` logical indicating if the image should contain only pixels within Area of Interest. Default: FALSE

`client` OAuth client object to use for authentication.

`token` OAuth token character string to use for authentication.

Exactly one of either `client` or `token` must be specified. It is recommended to use `client`.

`url` character indicating the process endpoint. Default: Copernicus Data Space Ecosystem process endpoint

Details

If `aoi` argument is provided, the result is returned in the same coordinate reference system.

Value

`SpatRaster` object (from the package `terra`) of the requested image (if `file` is `NULL`), or the (invisible) name of the file created.

Source

https://documentation.dataspace.copernicus.eu/APIs/SentinelHub/Process.html

See Also

`GetCollections`, `SearchCatalog`

Examples

```
## Not run:
dsn <- system.file("extdata", "centralpark.geojson", package = "CDSE")
aoi <- sf::read_sf(dsn, as_tibble = FALSE)
script_file <- system.file("scripts", "NDVI_uint8.js", package = "CDSE")
day <- "2023-07-11"
ras <- GetImage(aoi = aoi, time_range = day, script = script_file,
       collection = "sentinel-2-l2a",format = "image/tiff",
       mosaicking_order = "leastCC", resolution = 10, client = OAuthClient)

## End(Not run)
```

GetImageBy... *Get image from the archive (vectorization ready)*

Description

These functions retrieve the image for the area of interest using the parameters provided. They are simple wrappers around the 'GetImage' function with arguments organized in a way that facilitates calling the function in a vectorized manner (using 'lapply' or similar function) and thus potentially also the parallelization.

Usage

```
GetImageByTimerange(
  time_range,
  aoi,
  bbox,
  collection,
  script,
  file = NULL,
  format = c("image/tiff", "image/png", "image/jpeg"),
  mosaicking_order = c("mostRecent", "leastRecent", "leastCC"),
  pixels,
  resolution,
  buffer = 0,
  mask = FALSE,
  client,
  token,
  url = getOption("CDSE.process_url")
)

GetImageByAOI(
  aoi,
  time_range,
  collection,
  script,
```

```
  file = NULL,
  format = c("image/tiff", "image/png", "image/jpeg"),
  mosaicking_order = c("mostRecent", "leastRecent", "leastCC"),
  pixels,
  resolution,
  buffer = 0,
  mask = FALSE,
  client,
  token,
  url = getOption("CDSE.process_url")
)

GetImageByBbox(
  bbox,
  time_range,
  collection,
  script,
  file = NULL,
  format = c("image/tiff", "image/png", "image/jpeg"),
  mosaicking_order = c("mostRecent", "leastRecent", "leastCC"),
  pixels,
  resolution,
  buffer = 0,
  mask = FALSE,
  client,
  token,
  url = getOption("CDSE.process_url")
)
```

Arguments

time_range	scalar or vector (Date or character that can be converted to date) defining the time interval.
aoi	sf or sfc object, typically a (multi)polygon, describing the Area of Interest.
bbox	numeric vector of four elements describing the bounding box of interest. Specify with a coordinate pair on two (opposite) vertices of the bounding box rectangle. Coordinates need to be in longitude, latitude. Only one of either aoi or bbox may be specified.
collection	character indicating which collection to search. Must be one of the collections returned by GetCollections.
script	a length one character string containing the evaluation script or the name of the file containing the script.
file	name of the file to save the image. If NULL, a SpatRaster object is returned. Default: NULL
format	character indicating the output file format. Must be one of "image/tiff", "image/png", or "image/jpeg". Default: "image/tiff"

`mosaicking_order`
: character indicating the order in which tiles are overlapped from which the output result is mosaicked. Must be one of "mostRecent", "leastRecent", or "leastCC". Default: "mostRecent"

`pixels`
: integer scalar or length-two vector indicating the request image width and height. Values must be integers between 1 and 2500.

`resolution`
: numeric scalar or length-two vector indicating the spatial resolution of the request image in horizontal and vertical direction (in meters).

  Only one of the arguments "pixels" or "resolution" must be set at the same time. If the argument "pixels" or "resolution" is scalar, the same value is used for horizontal and vertical direction (width and height).

`buffer`
: numeric, width of the buffer to retrieve the image of enlarged area. Default: 0

`mask`
: logical indicating if the image should contain only pixels within Area of Interest. Default: FALSE

`client`
: OAuth client object to use for authentication.

`token`
: OAuth token character string to use for authentication.

  Exactly one of either `client` or `token` must be specified. It is recommended to use `client`.

`url`
: character indicating the process endpoint. Default: Copernicus Data Space Ecosystem process endpoint

Details

If `aoi` argument is provided, the result is returned in the same coordinate reference system.

`GetImageByTimerange` is arranged for vectorization on time_range (time_range is the first argument).

`GetImageByAOI` is arranged for vectorization on aoi (aoi is the first argument).

`GetImageByBbox` is arranged for vectorization on bbox (bbox is the first argument).

Value

`SpatRaster` object (from the package `terra`) of the requested image (if `file` is `NULL`), or the (invisible) name of the file created.

Source

https://documentation.dataspace.copernicus.eu/APIs/SentinelHub/Process.html

See Also

`GetImage`

Examples

```
## Not run:
dsn <- system.file("extdata", "centralpark.geojson", package = "CDSE")
aoi <- sf::read_sf(dsn, as_tibble = FALSE)
cloudless_images <- SearchCatalog(aoi = aoi, from = "2023-01-01", to = "2023-12-31",
                    collection = "sentinel-2-l2a", with_geometry = TRUE,
                    filter = "eo:cloud_cover < 0.8", client = OAuthClient)
script_file <- system.file("scripts", "NDVI_float32.js", package = "CDSE")
days <- rev(cloudless_images$acquisitionDate)
lstRast <- lapply(days, GetImageByTimerange, aoi = aoi, collection = "sentinel-2-l2a",
   script = script_file, file = NULL, format = "image/tiff", mosaicking_order = "mostRecent",
    resolution = 10, buffer = 0, mask = TRUE, client = OAuthClient,
    url = getOption("CDSE.process_url"))
par(mfrow = c(3, 4))
sapply(seq_along(days), FUN = function(i) {
     ras <- lstRast[[i]]
     day <- days[i]
     ras[ras < 0] <- 0
     terra::plot(ras, main = paste("Central Park NDVI on", day), range = c(0, 1),
            cex.main = 0.7, pax = list(cex.axis = 0.5), plg = list(cex = 0.5),
            col = colorRampPalette(c("darkred", "yellow", "darkgreen"))(99))
     })

## End(Not run)
```

`GetOAuthClient` *Get OAuth client*

Description

Gets an OAuth authentication client (`httr2` OAuth client object)

Usage

```
GetOAuthClient(id, secret, url = getOption("CDSE.auth_url"))
```

Arguments

`id`	character, user OAuth client id
`secret`	character, user OAuth client secret
`url`	character, endpoint for requesting tokens. Default: Copernicus Data Space Ecosystem OAuth endpoint

Details

The client can be used in queries requiring the authentication.

Value

`httr2` OAuth client object

Source

https://documentation.dataspace.copernicus.eu/APIs/SentinelHub/Overview/Authentication.html

See Also

GetOAuthToken

Examples

```
## Not run:
id <- "..."
secret <- "..."
OAuthClient <- GetOAuthClient(id = id, secret = secret)

## End(Not run)
```

GetOAuthToken *Get OAuth token*

Description

Gets an OAuth authentication token (long character string)

Usage

```
GetOAuthToken(id, secret, url = getOption("CDSE.auth_url"))
```

Arguments

`id`	character, user OAuth client id
`secret`	character, user OAuth client secret
`url`	character, endpoint for requesting tokens. Default: Copernicus Data Space Ecosystem OAuth endpoint

Details

The token can be used in queries requiring the authentication.

Value

Long character string containing the authentication token.

Source

https://documentation.dataspace.copernicus.eu/APIs/SentinelHub/Overview/Authentication.html

See Also

GetOAuthClient

Examples

```
## Not run:
id <- "..."
secret <- "..."
token <- GetOAuthToken(id = id, secret = secret)

## End(Not run)
```

GetQueryables *Get CQL2 parameters for a collection*

Description

Returns a list of variable terms that can be used in CQL2 expressions to filter the collection catalog search.

Usage

```
GetQueryables(
  collection,
  as_data_frame = TRUE,
  client,
  token,
  url = getOption("CDSE.catalog_url")
)
```

Arguments

collection	character indicating the collection for which the parameters are queried. Must be one of the collections returned by GetCollections.
as_data_frame	logical indicating if the result should be returned as data frame. Default: TRUE
client	OAuth client object to use for authentication.
token	OAuth token character string to use for authentication. Exactly one of either client or token must be specified. It is recommended to use client.
url	character indicating the STAC catalog search endpoint. Default: Copernicus Data Space Ecosystem STAC endpoint

Details

If no parameters found, a NULL value or 0-row data.frame is returned.

Value

A list or a data.frame.

Source

https://documentation.dataspace.copernicus.eu/APIs/SentinelHub/ApiReference.html#tag/catalog_collections/operation/getCatalogCollectionQueryables

See Also

GetCollections, SearchCatalog

Examples

```
## Not run:
GetQueryables("sentinel-2-l2a", client = OAuthClient)

## End(Not run)
```

GetStatistics *Get statistical values*

Description

Retrieves the simple statistics for the area of interest calculated based on satellite imagery without having to download images.

Usage

```
GetStatistics(
  aoi,
  bbox,
  time_range,
  collection,
  script,
  mosaicking_order = c("mostRecent", "leastRecent", "leastCC")[1],
  pixels,
  resolution,
  buffer = 0,
  percentiles = NULL,
  aggregation_period = 1L,
  aggregation_unit = c("day", "week", "month", "year")[1],
  lastIntervalBehavior = c("SKIP", "SHORTEN", "EXTEND")[1],
  as_data_frame = TRUE,
```

```
  client,
  token,
  url = getOption("CDSE.statistical_url")
)
```

Arguments

`aoi`	sf or sfc object, typically a (multi)polygon, describing the Area of Interest.
`bbox`	numeric vector of four elements describing the bounding box of interest. Specify with a coordinate pair on two (opposite) vertices of the bounding box rectangle. Coordinates need to be in longitude, latitude. Only one of either `aoi` or `bbox` may be specified.
`time_range`	scalar or vector (Date or character that can be converted to date) defining the time interval.
`collection`	character indicating which collection to search. Must be one of the collections returned by `GetCollections`.
`script`	a length one character string containing the evaluation script or the name of the file containing the script.
`mosaicking_order`	character indicating the order in which tiles are overlapped from which the output result is mosaicked. Must be one of "mostRecent", "leastRecent", or "leastCC". Partial matching is used, that is, only enough initial letters of each string element are needed to guarantee unique recognition. Default: "mostRecent"
`pixels`	integer scalar or length-two vector indicating the request image width and height. Values must be integers between 1 and 2500.
`resolution`	numeric scalar or length-two vector indicating the spatial resolution of the request image in horizontal and vertical direction (in meters). Only one of the arguments `pixels` or `resolution` must be set at the same time. If the argument `pixels` or `resolution` is scalar, the same value is used for horizontal and vertical direction (width and height).
`buffer`	numeric, width of the buffer to retrieve the image of enlarged area. Default: 0
`percentiles`	numeric vector indicating which percentile values should be computed. Default: NULL, don't compute any percentiles.
`aggregation_period`	the length of the aggregation period in `aggregation_unit` (days by default). Default: 1
`aggregation_unit`	character indicating the the unit of the aggregation period, must be one of "day", "week", "month", or "year". Partial matching is used, that is, only enough initial letters of each string element are needed to guarantee unique recognition (here just the first letter is enough). Default: "day"
`lastIntervalBehavior`	character indicating the behavior of the last interval if the given `time_range` isn't divisible by the provided `aggregation_period`. Must be one of:

	SKIP - skip the last interval (default behavior) SHORTEN - shortens the last interval so that it ends at the end of provided time_range EXTEND - extends the last interval over the end of the provided time range so that all intervals are of equal duration Partial matching is used, that is, only enough initial letters of each string element are needed to guarantee unique recognition. Default: "SKIP"
as_data_frame	logical indicating if the result should be returned as data frame. Default: TRUE
client	OAuth client object to use for authentication.
token	OAuth token character string to use for authentication. Exactly one of either client or token must be specified. It is recommended to use client.
url	character indicating the process endpoint. Default: Copernicus Data Space Ecosystem process endpoint

Details

The values are aggregated over the period (number of aggregation_units) given by the aggregation_period argument. The default values provide daily statistics. The statistics are returned only for the aggregation_units (days, weeks, months, years) when the data is available. This can be determined by the days of the satellite overpasses, but also by the calculations done in the evaluation script.

The scripts used for the Statistical API have some additional requirements: the evaluatePixel() function must, in addition to other output, always also return dataMask output. This output defines which pixels are excluded from calculations. For more information please visit the online documentation https://documentation.dataspace.copernicus.eu/APIs/SentinelHub/Statistical.html.

If a time_range is not divisible by an aggregation_period, the last ("not full") time interval will be dismissed by default (SKIP option). The user can instead set the lastIntervalBehavior to SHORTEN (shortens the last interval so that it ends at the end of the provided time range) or EXTEND (extends the last interval over the end of the provided time range so that all the intervals are of equal duration).

If percentiles requested are 25, 50, and 75, the columns are renamed 'q1', 'median', and 'q3'.

Value

data.frame or list with statistical values.

Source

https://documentation.dataspace.copernicus.eu/APIs/SentinelHub/Statistical.html

See Also

GetCollections, SearchCatalog

Examples

```
## Not run:
dsn <- system.file("extdata", "centralpark.geojson", package = "CDSE")
aoi <- sf::read_sf(dsn, as_tibble = FALSE)
script_file <- system.file("scripts", "NDVI_CLOUDS_STAT.js", package = "CDSE")
daily_stats <- GetStatistics(aoi = aoi, time_range = c("2023-07-01", "2023-07-31"),
  collection = "sentinel-2-l2a", script = script_file, mosaicking_order = "leastCC",
  resolution = 100, aggregation_period = 1, client = OAuthClient)
# specify week as 7 days
weekly_stats <- GetStatistics(aoi = aoi, time_range = c("2023-07-01", "2023-07-31"),
  collection = "sentinel-2-l2a", script = script_file,mosaicking_order = "leastCC",
  resolution = 100, aggregation_period = 7, client = OAuthClient)
# specify week as 1 week
weekly_stats_extended <- GetStatistics(aoi = aoi, time_range = c("2023-07-01", "2023-07-31"),
  collection = "sentinel-2-l2a", script = script_file, mosaicking_order = "leastCC",
  resolution = 100, aggregation_period = 1, aggregation_unit = "w",
  lastIntervalBehavior = "EXTEND", client = OAuthClient)

## End(Not run)
```

`GetStatisticsBy...` *Get statistical values (vectorization ready)*

Description

These functions retrieve simple statistics for the area of interest calculated based on satellite imagery without having to download images. They are simple wrappers around the `GetStatistics` function with arguments organized in a way that facilitates calling the function in a vectorized manner (using `lapply` or similar function) and thus potentially also the parallelization.

Usage

```
GetStatisticsByTimerange(
  time_range,
  aoi,
  bbox,
  collection,
  script,
  mosaicking_order = c("mostRecent", "leastRecent", "leastCC")[1],
  pixels,
  resolution,
  buffer = 0,
  percentiles = NULL,
  aggregation_period = 1L,
  aggregation_unit = c("day", "week", "month", "year")[1],
  lastIntervalBehavior = c("SKIP", "SHORTEN", "EXTEND")[1],
  as_data_frame = TRUE,
  client,
```

```
  token,
  url = getOption("CDSE.statistical_url")
)

GetStatisticsByAOI(
  aoi,
  time_range,
  collection,
  script,
  mosaicking_order = c("mostRecent", "leastRecent", "leastCC")[1],
  pixels,
  resolution,
  buffer = 0,
  percentiles = NULL,
  aggregation_period = 1L,
  aggregation_unit = c("day", "week", "month", "year")[1],
  lastIntervalBehavior = c("SKIP", "SHORTEN", "EXTEND")[1],
  as_data_frame = TRUE,
  client,
  token,
  url = getOption("CDSE.statistical_url")
)

GetStatisticsByBbox(
  bbox,
  time_range,
  collection,
  script,
  mosaicking_order = c("mostRecent", "leastRecent", "leastCC")[1],
  pixels,
  resolution,
  buffer = 0,
  percentiles = NULL,
  aggregation_period = 1L,
  aggregation_unit = c("day", "week", "month", "year")[1],
  lastIntervalBehavior = c("SKIP", "SHORTEN", "EXTEND")[1],
  as_data_frame = TRUE,
  client,
  token,
  url = getOption("CDSE.statistical_url")
)
```

Arguments

time_range	scalar or vector (Date or character that can be converted to date) defining the time interval.
aoi	sf or sfc object, typically a (multi)polygon, describing the Area of Interest.
bbox	numeric vector of four elements describing the bounding box of interest. Specify

with a coordinate pair on two (opposite) vertices of the bounding box rectangle. Coordinates need to be in longitude, latitude.

Only one of either `aoi` or `bbox` may be specified.

`collection` character indicating which collection to search. Must be one of the collections returned by `GetCollections`.

`script` a length one character string containing the evaluation script or the name of the file containing the script.

`mosaicking_order`
character indicating the order in which tiles are overlapped from which the output result is mosaicked. Must be one of "mostRecent", "leastRecent", or "leastCC". Partial matching is used, that is, only enough initial letters of each string element are needed to guarantee unique recognition. Default: "mostRecent"

`pixels` integer scalar or length-two vector indicating the request image width and height. Values must be integers between 1 and 2500.

`resolution` numeric scalar or length-two vector indicating the spatial resolution of the request image in horizontal and vertical direction (in meters).

Only one of the arguments "pixels" or "resolution" must be set at the same time. If the argument "pixels" or "resolution" is scalar, the same value is used for horizontal and vertical direction (width and height).

`buffer` numeric, width of the buffer to retrieve the image of enlarged area. Default: 0

`percentiles` numeric vector indicating which percentile values should be computed. Default: NULL, don't compute any percentiles.

`aggregation_period`
the length of the aggregation period in `aggregation_unit` (days by default). Default: 1

`aggregation_unit`
character indicating the the unit of the aggregation period, must be one of "day", "week", "month", or "year". Partial matching is used, that is, only enough initial letters of each string element are needed to guarantee unique recognition (here just the first letter is enough). Default: "day"

`lastIntervalBehavior`
character indicating the behavior of the last interval if the given `time_range` isn't divisible by the provided `aggregation_period`. Must be one of:

`SKIP` - skip the last interval (default behavior)

`SHORTEN` - shortens the last interval so that it ends at the end of provided `time_range`

`EXTEND` - extends the last interval over the end of the provided time range so that all intervals are of equal duration

Partial matching is used, that is, only enough initial letters of each string element are needed to guarantee unique recognition. Default: "SKIP"

`as_data_frame` logical indicating if the result should be returned as data frame. Default: TRUE

`client` OAuth client object to use for authentication.

`token` OAuth token character string to use for authentication.

Exactly one of either `client` or `token` must be specified. It is recommended to use `client`.

`url`	character indicating the process endpoint. Default: Copernicus Data Space Ecosystem process endpoint

Details

The values are aggregated over the period (number of `aggregation_units`) given by the `aggregation_period` argument. The default values provide daily statistics. The statistics are returned only for the `aggregation_units` (days, weeks, months, years) when the data is available. This can be determined by the days of the satellite overpasses, but also by the calculations done in the evaluation script.

The scripts used for the Statistical API have some additional requirements: the `evaluatePixel()` function must, in addition to other output, always also return dataMask output. This output defines which pixels are excluded from calculations. For more information please visit the online documentation https://documentation.dataspace.copernicus.eu/APIs/SentinelHub/Statistical.html.

If a `time_range` is not divisible by an `aggregation_period`, the last ("not full") time interval will be dismissed by default (SKIP option). The user can instead set the `lastIntervalBehavior` to SHORTEN (shortens the last interval so that it ends at the end of the provided time range) or EXTEND (extends the last interval over the end of the provided time range so that all the intervals are of equal duration).

If percentiles requested are 25, 50, and 75, the columns are renamed 'q1', 'median', and 'q3'.

`GetStatisticsByTimerange` is arranged for vectorization on time_range (time_range is the first argument).

`GetStatisticsByAOI` is arranged for vectorization on aoi (aoi is the first argument).

`GetStatisticsByBbox` is arranged for vectorization on bbox (bbox is the first argument).

Value

`data.frame` or `list` with statistical values.

Source

https://documentation.dataspace.copernicus.eu/APIs/SentinelHub/Statistical.html

See Also

`GetStatistics`

Examples

```
## Not run:
dsn <- system.file("extdata", "centralpark.geojson", package = "CDSE")
aoi <- sf::read_sf(dsn, as_tibble = FALSE)
script_file <- "inst/scripts/NDVI_dataMask_float32.js"
seasons <- SeasonalTimerange(from = "2020-06-01", to = "2023-08-31")
lst_stats <- lapply(seasons, GetStatisticsByTimerange, aoi = aoi, collection = "sentinel-2-l2a",
   script = script_file, mosaicking_order = "leastCC", resolution = 100,
   aggregation_period = 7L, client = OAuthClient)
```

```
weekly_stats <- do.call(rbind, lst_stats)
weekly_stats <- weekly_stats[rev(order(weekly_stats$from)), ]
row.names(weekly_stats) <- NULL
head(weekly_stats)

## End(Not run)
```

MakeEvalScript *Create a Sentinel Hub API Evalscript*

Description

Generates a JavaScript evalscript for calculating a spectral index for Sentinel-1 or Sentinel-2 imagery.

Usage

```
MakeEvalScript(x, ...)
```

Arguments

x	A `character` string with the `short_name` of a spectral index from `rsi::spectral_indices()` or a single-row `data.frame` (or `list`) with the elements `bands`, `formula`, `platforms`, and optionally `long_name`.
...	Named arguments providing values for any constants required by the index's formula. The function will stop if a required constant is missing and warn if unused arguments are provided.

Details

This function takes a spectral index definition and creates a JavaScript evalscript compatible with the Sentinel Hub API. The index can be specified by its short name from the `rsi` package's spectral index database (see `rsi::spectral_indices()`) or as a custom `list` or `data.frame`.

The function automatically maps common band names (e.g., 'N', 'R', 'G', 'B') to the corresponding Sentinel-1 or Sentinel-2 band names required by the API. Any constants in the index formula (e.g., the soil adjustment factor 'L' in SAVI) must be provided as named arguments.

Value

A 'character' vector containing the JavaScript evalscript. It can be used as 'script' argument in functions that require one as shown here:

`..., script = paste(MakeEvalScript("NDVI"), collapse = "\n"), ...`.

It can also be saved to a file for later use or further modifications.

Examples

```
## Not run:

# NDVI
si <- rsi::spectral_indices() # retrieves spectral indices
ndvi <- subset(si, short_name == "NDVI") # creates one-row data.frame
ndvi_script <- MakeEvalScript(ndvi) # generates the script
cat(ndvi_script, sep = "\n")

# SAVI, which requires the constant L
cat(MakeEvalScript("SAVI", L = 0.5), sep = "\n", file = "SAVI.js")

# Using a custom index definition as a data.frame
custom_index <- data.frame(
  short_name = "GNDVI",
  long_name = "Green Normalized Difference Vegetation Index",
  platforms = I(list("Sentinel-2")),
  bands = I(list(c("N", "G"))),
  formula = "(N - G) / (N + G)"
)
cat(MakeEvalScript(custom_index))

## End(Not run)
```

`Point2Bbox` *Create bounding box around a point*

Description

Creates the bounding box (numeric vector of length four) around the input point(s).

Usage

```
Point2Bbox(x, y = NULL, size, crs = 4326)
```

Arguments

`x`	an `sf`, `sfc`, or `SpatialPoints*` object, a numeric indicating the longitude/easting of the point(s), or any input accepted by `xy.coords`
`y`	numeric, the latitude/northing of the point(s). Default: NULL
`size`	numeric indicating the size (in meters) of the bounding box to create
`crs`	coordinate reference system of the input (and the output): object of class `crs`, or input string for `st_crs`. Default: 4326 (WGS 84)

Details

The function assumes that the `crs` units are either degrees or meters, a warning is issued if not, and the result will probably be incorrect.

Value

A bounding box (numeric vector of length four), or a list of bounding boxes if the input is not scalar.

See Also

xy.coords, st_crs

Examples

```
## Not run:
Point2Bbox(x = -73.96557, y = 40.78246, size = 1000, crs = 4326)

## End(Not run)
```

SearchCatalog *Search collection for available images*

Description

Searches the specified collection for available images in the given time interval and intersecting with the bounding box or the area of interest.

Usage

```
SearchCatalog(
  aoi,
  bbox,
  from,
  to,
  collection,
  as_data_frame = TRUE,
  with_geometry = TRUE,
  filter = NULL,
  client,
  token,
  url = getOption("CDSE.catalog_url")
)
```

Arguments

aoi	sf or sfc object, typically a (multi)polygon, describing the Area of Interest.
bbox	numeric vector of four elements describing the bounding box of interest. Specify with a coordinate pair on two (opposite) vertices of the bounding box rectangle. Coordinates need to be in longitude, latitude. Only one of either aoi or bbox may be specified.
from	start of the time interval to search.

to	end of the time interval to search. from and to can be either Date or character that can be converted to date by as.Date. Open interval (one side only) can be obtained by providing the NA or NULL value for the corresponding argument.
collection	character indicating which collection to search. Must be one of the collections returned by GetCollections.
as_data_frame	logical indicating if the result should be returned as data frame. Default: TRUE
with_geometry	logical indicating if the granule geometries should be included in the data.frame. Default: TRUE
filter	character, CQL2 text filter. Use the function GetQueryables to find out which filters can bu used with the collection. Default: NULL (no filtering)
client	OAuth client object to use for authentication.
token	OAuth token character string to use for authentication. Exactly one of either client or token must be specified. It is recommended to use client.
url	character indicating the STAC catalog search endpoint. Default: Copernicus Data Space Ecosystem STAC endpoint

Details

If no images found, a NULL value is returned.

Value

A list, data.frame or a sf object.

Source

https://documentation.dataspace.copernicus.eu/APIs/SentinelHub/Catalog.html

See Also

GetCollections, GetQueryables, GetImage

Examples

```
## Not run:
dsn <- system.file("extdata", "luxembourg.geojson", package = "CDSE")
aoi <- sf::read_sf(dsn, as_tibble = FALSE)
images <- SearchCatalog(aoi = aoi, from = "2023-07-01", to = "2023-07-31",
          collection = "sentinel-2-l2a", with_geometry = TRUE, client = OAuthClient)
images_cloudless <- SearchCatalog(aoi = aoi, from = "2023-07-01", to = "2023-07-31",
          filter = "eo:cloud_cover < 5",
          collection = "sentinel-2-l2a", with_geometry = TRUE, client = OAuthClient)

## End(Not run)
```

SearchCatalogBy... *Search collection for available images (vectorization ready)*

Description

These functions search the specified collection for available images using the parameters provided. They are simple wrappers around the 'SearchCatalog' function with arguments organized in a way that facilitates calling the function in a vectorized manner (using 'lapply' or similar function) and thus potentially also the parallelization. The 'from' and 'to' arguments are combined into a single argument 'time_range'.

Usage

```
SearchCatalogByTimerange(
  time_range,
  aoi,
  bbox,
  collection,
  as_data_frame = TRUE,
  with_geometry = TRUE,
  filter = NULL,
  client,
  token,
  url = getOption("CDSE.catalog_url")
)

SearchCatalogByAOI(
  aoi,
  time_range,
  collection,
  as_data_frame = TRUE,
  with_geometry = TRUE,
  filter = NULL,
  client,
  token,
  url = getOption("CDSE.catalog_url")
)

SearchCatalogByBbox(
  bbox,
  time_range,
  collection,
  as_data_frame = TRUE,
  with_geometry = TRUE,
  filter = NULL,
  client,
  token,
```

```
  url = getOption("CDSE.catalog_url")
)
```

Arguments

time_range	scalar or vector (Date or character that can be converted to date) defining the time interval. Open interval (one side only) can be obtained by providing the NA or NULL value for the corresponding argument.
aoi	sf or sfc object, typically a (multi)polygon, describing the Area of Interest.
bbox	numeric vector of four elements describing the bounding box of interest. Specify with a coordinate pair on two (opposite) vertices of the bounding box rectangle. Coordinates need to be in longitude, latitude. Only one of either aoi or bbox may be specified.
collection	character indicating which collection to search. Must be one of the collections returned by GetCollections.
as_data_frame	logical indicating if the result should be returned as data frame. Default: TRUE
with_geometry	logical indicating if the granule geometries should be included in the data.frame. Default: TRUE
filter	character, CQL2 text filter. Use the function GetQueryables to find out which filters can bu used with the collection. Default: NULL (no filtering)
client	OAuth client object to use for authentication.
token	OAuth token character string to use for authentication. Exactly one of either client or token must be specified. It is recommended to use client.
url	character indicating the STAC catalog search endpoint. Default: Copernicus Data Space Ecosystem STAC endpoint

Details

If no images found, a NULL value is returned.

SearchCatalogByTimerange is arranged for vectorization on time_range (time_range is the first argument).

SearchCatalogByAOI is arranged for vectorization on aoi (aoi is the first argument).

SearchCatalogByBbox is arranged for vectorization on bbox (bbox is the first argument).

Value

A list, data.frame or a sf object.

Source

https://documentation.dataspace.copernicus.eu/APIs/SentinelHub/Catalog.html

See Also

SearchCatalog

Examples

```
## Not run:
dsn <- system.file("extdata", "centralpark.geojson", package = "CDSE")
aoi <- sf::read_sf(dsn, as_tibble = FALSE)
seasons <- SeasonalTimerange(from = "2020-06-01", to = "2023-08-31")
lst_images_cloudless <- lapply(seasons, SearchCatalogByTimerange, aoi = aoi,
    collection = "sentinel-2-l2a", with_geometry = FALSE,
    filter = "eo:cloud_cover < 5", client = OAuthClient)
images_cloudless <- do.call(rbind, lst_images_cloudless)
images_cloudless <- images_cloudless[rev(order(images_cloudless$acquisitionDate)), ]
row.names(images_cloudless) <- NULL
head(images_cloudless[, 1:5])

## End(Not run)
```

`SeasonalFilter` *Filter image catalog for seasonal images*

Description

Filters image catalog entries that fall in the season of interest - dates between `from` day/month and `to` day/month for all years in the `from` - `to` time range.

Usage

```
SeasonalFilter(catalog, from, to)
```

Arguments

`catalog`	`data.frame` or `sf` object as the one produced by a call to `SearchCatalog`
`from`	start of the season of interest.
`to`	end of the season of interest.
	The `from` and `to` arguments can be either Date or character that can be converted to date by `as.Date`. Open intervals are not allowed (both `from` and `to` must be valid dates).

Value

A `data.frame` or a `sf` object, depending on the type of the input.

See Also

`SearchCatalog`, `SeasonalTimerange`

Examples

```
## Not run:
dsn <- system.file("extdata", "centralpark.geojson", package = "CDSE")
aoi <- sf::read_sf(dsn, as_tibble = FALSE)
all_images <- SearchCatalog(aoi = aoi, from = "2021-06-01", to = "2023-08-31",
    collection = "sentinel-2-l2a", with_geometry = TRUE, client = OAuthClient)
sesonal_images <- SeasonalFilter(all_images, from = "2021-06-01", to = "2023-08-31")

## End(Not run)
```

SeasonalTimerange *Create seasonal time range*

Description

Creates list of seasonal filters (one per year) for the season of interest - dates between `from` day/month and `to` day/month for all years in the `from` - `to` time range.

Usage

```
SeasonalTimerange(from, to)
```

Arguments

`from`	start of the season of interest.
`to`	end of the season of interest. The `from` and `to` arguments can be either Date or character that can be converted to date by `as.Date`. Open intervals are not allowed (both `from` and `to` must be valid dates).

Value

A list of time ranges defining the season of interest for each year.

Examples

```
## Not run:
seasons <- SeasonalTimerange(from = "2020-05-01", to = "2023-09-30")
seasons <- SeasonalTimerange(from = "2019-11-01", to = "2023-03-30")

## End(Not run)
```

`UniqueCatalog` *Produce image catalog without multiple entries per date*

Description

Sometimes several images could be available for the given day. It can be useful to have a list where for any given day there is just one row in the list. This unique row can be selected to represent either the least cloud coverage or the biggest coverage of the are of interest.

Usage

```
UniqueCatalog(
  imageCatalog,
  by = c("areaCoverage", "tileCloudCover"),
  keep = names(imageCatalog)
)
```

Arguments

`imageCatalog`	`data.frame` as returned by the `SearchCatalog` function.
`by`	character indicating which attribute is used to select the best image per date. Can be either "areaCoverage" or "tileCloudCover".
`keep`	list of columns to keep in output. Default: all columns in input.

Details

By default, the returned `data.frame` has the same columns as the input catalog. User can specify a subset of columns to include in the output through the `keep` parameter.

Value

`data.frame` with one row per date.

See Also

`SearchCatalog`

Examples

```
## Not run:
dsn <- system.file("extdata", "luxembourg.geojson", package = "CDSE")
aoi <- sf::read_sf(dsn, as_tibble = FALSE)
images <- SearchCatalog(aoi = aoi, from = "2023-07-01", to = "2023-07-31",
          collection = "sentinel-2-l2a", with_geometry = TRUE, client = OAuthClient)
best_daily <- UniqueCatalog(images, by = "areaCoverage",
              keep = c("acquisitionDate", "tileCloudCover", "areaCoverage", "satellite"))

## End(Not run)
```

Index