

Package ‘rstac’

October 31, 2021

Title Client Library for SpatioTemporal Asset Catalog

Version 0.9.1-5

Description Provides functions to access, search and download spacetime earth observation data via SpatioTemporal Asset Catalog (STAC). This package supports the version 1.0.0 of the STAC specification (<https://github.com/radiantearth/stac-spec>).

For further details see Simoes et al. (2021) <[doi:10.1109/IGARSS47720.2021.9553518](https://doi.org/10.1109/IGARSS47720.2021.9553518)>.

License MIT + file LICENSE

URL <https://github.com/brazil-data-cube/rstac>

BugReports <https://github.com/brazil-data-cube/rstac/issues>

Encoding UTF-8

RoxygenNote 7.1.2

Depends R (>= 3.5)

Imports htrr, crayon, utils, magrittr, jsonlite, lifecycle

Suggests covr, testthat, vcr

NeedsCompilation no

Author Brazil Data Cube Team [cre, aut],
National Institute for Space Research (INPE) [cph]

Maintainer Brazil Data Cube Team <brazildatacube@inpe.br>

Repository CRAN

Date/Publication 2021-10-31 21:10:02 UTC

R topics documented:

assets_download	2
assets_function	3
collections	5
doc_query	6
ext_query	6
get_request	7

items	8
items_bands	10
items_functions	11
print	16
rstac	17
RSTACDocument	19
sign_bdc	22
sign_planetary_computer	23
stac	24
stac_search	25
utilities	27
%>%	27

Index	28
--------------	-----------

assets_download	<i>Downloads assets via STAC API</i>
-----------------	--------------------------------------

Description

The `assets_download` function downloads the assets provided by the STAC API.

Usage

```
assets_download(
  items,
  asset_names = NULL,
  output_dir = ".",
  overwrite = FALSE,
  items_max = Inf,
  progress = TRUE,
  fn = NULL,
  ...,
  assets_name = deprecated()
)
```

Arguments

<code>items</code>	a STACItem or STACItemCollection object representing the result of <code>/stac/search</code> , <code>/collections/{collectionId}/items</code> or <code>/collections/{collectionId}/items/{itemId}</code> endpoints.
<code>asset_names</code>	a character with the assets names to be filtered.
<code>output_dir</code>	a character directory in which the assets will be saved.
<code>overwrite</code>	a logical if TRUE will replaced the existing file, if FALSE a warning message is shown.
<code>items_max</code>	a numeric corresponding how many items will be downloaded.
<code>progress</code>	a logical indicating if a progress bar must be shown or not. Defaults to TRUE.

fn	a function to handle the list of assets for each item. Using this function you can change the hrefs for each asset, as well as use another request verb, such as POST.
...	config parameters to be passed to GET or POST methods, such as add_headers or set_cookies .
assets_name	[Deprecated] use asset_names parameter instead.

Value

The same STACItemCollection or STACItem object, with the link of the item pointing to the directory where the assets were saved.

See Also

[stac_search\(\)](#), [items\(\)](#), [get_request\(\)](#)

Examples

```
## Not run:
stac("https://brazildatacube.dpi.inpe.br/stac/") %>%
  stac_search(collections = "CB4_64_16D_STK-1",
             datetime = "2019-06-01/2019-08-01") %>%
  stac_search() %>%
  get_request() %>%
  assets_download(asset_names = "thumbnail", output_dir = ".",
                 overwrite = FALSE)

## End(Not run)
```

assets_function	<i>Assets functions</i>
-----------------	-------------------------

Description

This function returns the date, band and URL fields for each assets of an STACItemCollection object. For the URL you can add the GDAL library drivers for the following schemes: HTTP/HTTPS files, S3 (AWS S3) and GS (Google Cloud Storage).

Usage

```
assets_list(
  items,
  asset_names = NULL,
  sort = TRUE,
  gdal_vsi_resolution = TRUE,
  assets_names = deprecated()
)
```

```

assets_select(items, asset_names)

## S3 method for class 'STACItemCollection'
assets_select(items, asset_names)

## S3 method for class 'STACItem'
assets_select(items, asset_names)

assets_filter(items, ..., fn = NULL)

## S3 method for class 'STACItemCollection'
assets_filter(items, ..., fn = NULL)

## S3 method for class 'STACItem'
assets_filter(items, ..., fn = NULL)

```

Arguments

items	a STACItemCollection object representing the result of /stac/search, /collections/{collectionId}/items
asset_names	a character with the assets names to be filtered. If NULL (default) all assets will be returned..
sort	a logical if true the dates will be sorted in increasing order. By default, the dates are sorted.
gdal_vsi_resolution	a logical if true, gdal drivers are included in the URL of each asset. The following schemes are supported: HTTP/HTTPS files, S3 (AWS S3) and GS (Google Cloud Storage).
assets_names	[Deprecated] use asset_names parameter instead.
...	additional arguments. See details.
fn	a function that will be used to filter the attributes listed in the properties.

Value

a list with the attributes of date, bands and paths.

Examples

```

# STACItemCollection object
stac_item <- stac("https://brazildatacube.dpi.inpe.br/stac/") %>%
  stac_search(collections = "CB4_64_16D_STK-1", limit = 100,
             datetime = "2017-08-01/2018-03-01",
             bbox = c(-48.206, -14.195, -45.067, -12.272)) %>%
  get_request() %>% items_fetch(progress = FALSE)

stac_item %>% assets_list(assets_names = c("EVI", "NDVI"))

```

`collections`*Endpoint functions*

Description

The `collections` function implements the WFS3 `/collections` and `/collections/{collectionId}` endpoints.

Each endpoint retrieves specific STAC objects:

- `/collections`: Returns a list of STAC Collection published in the STAC service
- `/collections/{collectionId}`: Returns a single STAC Collection object

Usage

```
collections(q, collection_id = NULL)
```

Arguments

`q` a `RSTACQuery` object expressing a STAC query criteria.
`collection_id` a character collection id to be retrieved.

Value

A `RSTACQuery` object with the subclass `collections` for `/collections/` endpoint, or a `collection_id` subclass for `/collections/{collection_id}` endpoint, containing all search field parameters to be provided to STAC API web service.

See Also

[get_request\(\)](#), [post_request\(\)](#), [items\(\)](#)

Examples

```
stac("https://brazildatacube.dpi.inpe.br/stac/") %>%  
  collections() %>%  
  get_request()  
  
stac("https://brazildatacube.dpi.inpe.br/stac/") %>%  
  collections(collection_id = "CB4_64_16D_STK-1") %>%  
  get_request()
```

doc_query *Document utils functions*

Description

Document utils functions

Usage

doc_query(d)

Arguments

d an RSTACDocument object

Value

a RSTACQuery object with the predecessor subclass with the fields used in the request.

ext_query *Extension functions*

Description

The `ext_query()` is the *exported function* of the STAC API query extension. It can be used after a call to `stac_search()` function. It allows that additional fields and operators other than those defined in `stac_search()` function be used to make a complex filter.

The function accepts multiple filter criteria. Each filter entry is an expression formed by `<field> <operator> <value>`, where `<field>` refers to a valid item property. Supported `<fields>` depends on STAC API service implementation. The users must rely on service providers' documentation to know which properties can be used by this extension.

The `ext_query()` function allows the following `<operators>`

- `==` corresponds to `'eq'`
- `!=` corresponds to `'neq'`
- `<` corresponds to `'lt'`
- `<=` corresponds to `'lte'`
- `>` corresponds to `'gt'`
- `>=` corresponds to `'gte'`
- `\%startsWith\%` corresponds to `'startsWith'` and implements a string prefix search operator.
- `\%endsWith\%` corresponds to `'endsWith'` and implements a string suffix search operator.
- `\%contains\%:` corresponds to `'contains'` and implements a string infix search operator.
- `\%in\%:` corresponds to `'in'` and implements a vector search operator.

Besides this function, the following S3 generic methods were implemented to get things done for this extension:

- The `endpoint()` for subclass `ext_query`
- The `before_request()` for subclass `ext_query`
- The `after_response()` for subclass `ext_query`

See source file `ext_query.R` for an example on how implement new extensions.

Usage

```
ext_query(q, ...)
```

Arguments

`q` a RSTACQuery object expressing a STAC query criteria.
`...` entries with format `<field> <operator> <value>`.

Value

A RSTACQuery object with the subclass `ext_query` containing all request parameters to be passed to `post_request()` function.

See Also

[stac_search\(\)](#), [post_request\(\)](#), [endpoint\(\)](#), [before_request\(\)](#), [after_response\(\)](#), [content_response\(\)](#)

Examples

```
stac("https://brazildatacube.dpi.inpe.br/stac/") %>%
  stac_search(collections = "CB4_64_16D_STK-1") %>%
  ext_query("bdc:tile" %in% "022024") %>%
  post_request()
```

get_request

STAC API request functions

Description

The `get_request` is function that makes HTTP GET requests to STAC web services, retrieves, and parse the data.

The `post_request` is function that makes HTTP POST requests to STAC web services, retrieves, and parse the data.

Usage

```
get_request(q, ...)
```

```
post_request(q, ..., encode = c("json", "multipart", "form"))
```

Arguments

q a RSTACQuery object expressing a STAC query criteria.

... config parameters to be passed to **GET** or **POST** methods, such as [add_headers](#) or [set_cookies](#).

encode a character informing the request body Content-Type. Accepted types are 'json' ('application/json'), 'form' ('application/x-www-form-urlencoded'), and 'multipart' ('multipart/form-data'). Defaults to 'json'.

Value

Either a STACCatalog, STACCollection, STACCollectionList, STACItemCollection or STACItem object depending on the subclass and search fields parameters of q argument.

See Also

[stac\(\)](#) [stac_search\(\)](#) [collections\(\)](#) [items\(\)](#)

Examples

```
stac("https://brazildatacube.dpi.inpe.br/stac/") %>%
  get_request()
```

```
stac("https://brazildatacube.dpi.inpe.br/stac/") %>%
  stac_search(collections = "CB4_64_16D_STK-1") %>%
  post_request()
```

items

Endpoint functions

Description

The `items` function implements WFS3 `/collections/{collectionId}/items`, and `/collections/{collectionId}/items/{itemId}` endpoints.

Each endpoint retrieves specific STAC objects:

- `/collections/{collectionId}/items`: Returns a STAC Items collection (GeoJSON)
- `/collections/{collectionId}/items/{itemId}`: Returns a STAC Item (GeoJSON Feature)

The endpoint `/collections/{collectionId}/items` accepts the same filters parameters of [stac_search\(\)](#) function.

Usage

```
items(q, feature_id = NULL, datetime = NULL, bbox = NULL, limit = NULL)
```

Arguments

- q** a RSTACQuery object expressing a STAC query criteria.
- feature_id** a character with item id to be fetched. Only works if the `collection_id` is informed. This is equivalent to the endpoint `/collections/{collectionId}/items/{featureId}`.
- datetime** a character with a date-time or an interval. Date and time strings needs to conform RFC 3339. Intervals are expressed by separating two date-time strings by '/' character. Open intervals are expressed by using '..' in place of date-time.
- Examples:
- A date-time: "2018-02-12T23:20:50Z"
 - A closed interval: "2018-02-12T00:00:00Z/2018-03-18T12:31:12Z"
 - Open intervals: "2018-02-12T00:00:00Z/.." or "../2018-03-18T12:31:12Z"
- Only features that have a `datetime` property that intersects the interval or date-time informed in `datetime` are selected.
- bbox** a numeric vector with only features that have a geometry that intersects the bounding box are selected. The bounding box is provided as four or six numbers, depending on whether the coordinate reference system includes a vertical axis (elevation or depth):
- Lower left corner, coordinate axis 1
 - Lower left corner, coordinate axis 2
 - Lower left corner, coordinate axis 3 (optional)
 - Upper right corner, coordinate axis 1
 - Upper right corner, coordinate axis 2
 - Upper right corner, coordinate axis 3 (optional)
- The coordinate reference system of the values is WGS84 longitude/latitude (<http://www.opengis.net/def/crs/OGC/1.3/CRS84>). The values are in most cases the sequence of minimum longitude, minimum latitude, maximum longitude and maximum latitude. However, in cases where the box spans the antimeridian the first value (west-most box edge) is larger than the third value (east-most box edge).
- limit** an integer defining the maximum number of results to return. If not informed it defaults to the service implementation.

Value

A RSTACQuery object with the subclass `items` for `/collections/{collection_id}/items` endpoint, or a `item_id` subclass for `/collections/{collection_id}/items/{feature_id}` endpoint, containing all search field parameters to be provided to STAC API web service.

See Also

[get_request\(\)](#), [post_request\(\)](#), [collections\(\)](#)

Examples

```
stac("https://brazildatacube.dpi.inpe.br/stac/") %>%
  collections("CB4_64_16D_STK-1") %>%
  items(bbox = c(-47.02148, -12.98314, -42.53906, -17.35063)) %>%
  get_request()

stac("https://brazildatacube.dpi.inpe.br/stac/") %>%
  collections("CB4_64_16D_STK-1") %>%
  items("CB4_64_16D_STK_v001_022023_2020-07-11_2020-07-26") %>%
  get_request()
```

items_bands

STACItemCollection function

Description

[Deprecated] Use `items_assets()` function instead.

Usage

```
items_bands(items)
```

Arguments

`items` a `STACItemCollection` or `STACItem` object.

Examples

```
## Not run:

x <- stac("http://brazildatacube.dpi.inpe.br/stac") %>%
  stac_search(collections = "CB4_64_16D_STK-1") %>%
  stac_search() %>%
  get_request()

x %>% items_assets()

## End(Not run)
```

items_functions *STACItemCollection functions*

Description

These functions provide support to work with `STACItemCollection` and `STACItem` objects.

- `items_length()`: shows how many items there are in the `STACItemCollection` object.
- `items_matched()`: shows how many items matched the search criteria. It supports `search:metadata` (v0.8.0), `context` (v0.9.0), and `numberMatched` (OGC WFS3 core spec).
- `items_fetch()`: request all STAC Items through pagination.
- `items_next()`: fetches a new page from STAC service.
- `items_datetime()`: retrieves a the datetime field in properties from `STACItemCollection` and `STACItem` objects.
- `items_bbox()`: retrieves a the bbox field of a `STACItemCollection` or an `STACItem` object.
- `item_assets()`: returns the assets name from `STACItemCollection` and `STACItem` objects.
- `items_filter()`: selects only items that match some criteria.
- `items_reap()`: extract key values by traversing all items in an `STACItemCollection` object.
- `items_fields()`: lists field names inside an item.
- `items_group()`: organizes items as elements of a list using some criteria.
- `items_sign()`: allow access assets by preparing its url.

This function returns the subfields of the feature field of a `STACItemCollection` object.

Usage

```
items_length(items)

## S3 method for class 'STACItem'
items_length(items)

## S3 method for class 'STACItemCollection'
items_length(items)

items_matched(items, ...)

## S3 method for class 'STACItem'
items_matched(items, ...)

## S3 method for class 'STACItemCollection'
items_matched(items, ..., matched_field = NULL)

items_fetch(items, ...)
```

```
## S3 method for class 'STACItem'
items_fetch(items, ...)

## S3 method for class 'STACItemCollection'
items_fetch(items, ..., progress = TRUE, matched_field = NULL)

items_next(items, ...)

## S3 method for class 'STACItem'
items_next(items, ...)

## S3 method for class 'STACItemCollection'
items_next(items, ...)

items_datetime(items)

## S3 method for class 'STACItem'
items_datetime(items)

## S3 method for class 'STACItemCollection'
items_datetime(items)

items_bbox(items)

## S3 method for class 'STACItem'
items_bbox(items)

## S3 method for class 'STACItemCollection'
items_bbox(items)

items_assets(items, ...)

## S3 method for class 'STACItem'
items_assets(items, ...)

## S3 method for class 'STACItemCollection'
items_assets(items, ..., simplify = FALSE)

items_filter(items, ..., filter_fn = NULL)

items_reap(items, ..., field = NULL)

items_fields(items, ..., field = NULL)

## S3 method for class 'STACItemCollection'
items_fields(items, ..., field = NULL)

## S3 method for class 'STACItem'
```

```

items_fields(items, ..., field = NULL)

items_group(items, ..., field = NULL, index = NULL)

items_sign(items, ..., sign_fn = NULL)

## S3 method for class 'STACItemCollection'
items_sign(items, ..., sign_fn = NULL)

## S3 method for class 'STACItem'
items_sign(items, ..., sign_fn = NULL)

```

Arguments

items	a STACItemCollection object representing the result of /stac/search, /collections/{collectionId}/i
...	a named way to provide field names to get the subfields values from the RSTACDocument objects.
matched_field	a character vector with the path where the number of items returned in the named list is located starting from the initial node of the list. For example, if the information is at position items\$meta\$found of the object, it must be passed as the following parameter c("meta", "found").
progress	a logical indicating if a progress bar must be shown or not. Defaults to TRUE.
simplify	a logical should return only the assets name of the first item? if not a list with all assets name will be returned. Default is FALSE.
filter_fn	a function that receives an item that should evaluate a logical value.
field	a character with the names of the field to get the subfields values from the RSTACDocument objects.
index	an atomic vector with values as group index
sign_fn	a function that receives an item as parameter and returns an item signed.

Details

Ellipsis argument (...) appears in different items function and has distinct purposes:

- items_matched() and items_assets(): ellipsis is not used.
- items_fetch() and items_next(): ellipsis is used to pass additional http options to [GET](#) or [POST](#) methods, such as [add_headers](#) or [set_cookies](#).
- items_filter(): ellipsis is used to pass logical expressions using keys in properties field as filter criteria.
- items_reap(), items_fields(), and items_group(): ellipsis can be used to provide fields names to get the subfields values from the STACItemCollection objects.
- items_sign(): in a near future, ellipsis will be used to append key value pairs to url query string of an asset.

items_sign() has sign_fn parameter that must be a function that receives as argument an item and returns an signed item. rstac provides sign_bdc() and sign_planetary_computer() functions to access Brazil Data Cube products and Microsoft Planetary Computer catalogs, respectively.

Value

- `items_length()`: an integer value.
- `items_matched()`: returns an integer value if STAC web server does support this extension, otherwise returns NULL.
- `items_fetch()`: an `STACItemCollection` with all matched items.
- `items_next()`: fetches a new page from STAC service.
- `items_datetime()`: a list of all items' datetime.
- `items_bbox()`: returns a list with all items' bounding boxes.
- `item_assets()`: if `simplify` is TRUE, returns a character value with all assets names of the first item. Otherwise, returns a list with assets name for each item.
- `items_filter()`: a `STACItemCollection` object.
- `items_reap()`: a vector if the supplied field is atomic, otherwise or a list.
- `items_fields()`: a character vector.
- `items_group()`: a list of `STACItemCollection` objects.
- `items_sign()`: a `STACItemCollection` object with signed assets url

A character with the subfields of the feature field.

Examples

```
## Not run:

x <- stac("https://brazildatacube.dpi.inpe.br/stac") %>%
  stac_search(collections = "CB4_64_16D_STK-1") %>%
  stac_search(limit = 500) %>%
  get_request()

x %>% items_length()
x %>% items_matched()
x %>% items_datetime()
x %>% items_bbox()
x %>% items_fetch()

## End(Not run)

## Not run:

# Defining BDC token
Sys.setenv("BDC_ACCESS_KEY" = <your_bdc_access_key>)

# STACItem object
stac("https://brazildatacube.dpi.inpe.br/stac/") %>%
  stac_search(collections = "CB4_64_16D_STK-1", limit = 100,
    datetime = "2017-08-01/2018-03-01",
    bbox = c(-48.206,-14.195,-45.067,-12.272)) %>%
  get_request() %>% items_sign(sign_fn = sign_bdc())
```

```

## End(Not run)

## Not run:

# STACItemCollection object
stac("https://brazildatacube.dpi.inpe.br/stac/") %>%
  stac_search(collections = "CB4_64_16D_STK-1", limit = 100,
             datetime = "2017-08-01/2018-03-01",
             bbox = c(-48.206,-14.195,-45.067,-12.272)) %>%
  get_request() %>% items_filter(`eo:cloud_cover` < 10)

# Example with AWS STAC
stac("https://earth-search.aws.element84.com/v0") %>%
  stac_search(collections = "sentinel-s2-l2a-cogs",
             bbox = c(-48.206,-14.195,-45.067,-12.272),
             datetime = "2018-06-01/2018-06-30",
             limit = 500) %>%
  post_request() %>%
  items_filter(filter_fn = function(x) {x[["eo:cloud_cover"]] < 10})

## End(Not run)

## Not run:
# STACItemCollection object
stac_item <- stac("https://brazildatacube.dpi.inpe.br/stac/") %>%
  stac_search(collections = "CB4_64_16D_STK-1", limit = 100,
             datetime = "2017-08-01/2018-03-01",
             bbox = c(-48.206,-14.195,-45.067,-12.272)) %>%
  get_request() %>% items_fetch(progress = FALSE)

stac_item %>% items_reap(field = c("properties", "datetime"))

## End(Not run)

## Not run:
# STACItemCollection object
stac_item <- stac("https://brazildatacube.dpi.inpe.br/stac/") %>%
  stac_search(collections = "CB4_64_16D_STK-1", limit = 100,
             datetime = "2017-08-01/2018-03-01",
             bbox = c(-48.206,-14.195,-45.067,-12.272)) %>%
  get_request() %>% items_fetch(progress = FALSE)

stac_item %>% items_group(., field = c("properties", "bdc:tiles"))

## End(Not run)

# STACItemCollection object
stac_item <- stac("https://brazildatacube.dpi.inpe.br/stac/") %>%
  stac_search(collections = "CB4_64_16D_STK-1", limit = 10,
             datetime = "2017-08-01/2018-03-01") %>%
  get_request()

```

```
stac_item %>% items_fields(field = c("properties"))
```

```
print
```

Printing functions

Description

The print function covers all objects in the rstac package:

- `stac()`: returns a STACCatalog document from /stac (v0.8.0) or / (v0.9.0 or v1.0.0) endpoint.
- `stac_search()`: returns a STACItemCollection document from /stac/search (v0.8.0) or /search (v0.9.0 or v1.0.0) endpoint containing all Items that match the provided search predicates.
- `collections()`: implements the /collections and /collections/{collectionId} endpoints. The former returns a STACCollectionList document that lists all collections published by the server, and the later returns a single STACCollection document that describes a unique collection.
- `items()`: retrieves a STACItemCollection document from /collections/{collectionId}/items endpoint and a STACItem document from /collections/{collectionId}/items/{itemId} endpoints.

The rstac package objects visualization is based on markdown, a lightweight markup language. You can paste the output into any markdown editor for a better visualization.

Call `print()` function to print the rstac's objects. You can determine how many items will be printed using `n` parameter.

Usage

```
## S3 method for class 'RSTACQuery'
print(x, ...)

## S3 method for class 'STACCatalog'
print(x, ...)

## S3 method for class 'STACCollectionList'
print(x, n = 10, ...)

## S3 method for class 'STACCollection'
print(x, ...)

## S3 method for class 'STACItemCollection'
print(x, n = 10, ..., tail = FALSE)

## S3 method for class 'STACItem'
print(x, ...)
```


Arguments

<code>x</code>	either a <code>RSTACQuery</code> object expressing a STAC query criteria or any <code>RSTACDocument</code> .
<code>...</code>	other parameters passed in the functions.
<code>n</code>	number of entries to print. Each object has its own rule of truncation: the <code>STACCollection</code> objects will print 10 links by default. If the object has less than 20 collections, all collections will be shown. In <code>STACItemCollection</code> , 10 features will be printed by default. To show all entries, use <code>n = Inf</code> .
<code>tail</code>	A logical value indicating if last features in <code>STACItemCollection</code> object must be show.

See Also

[stac\(\)](#) [stac_search\(\)](#) [collections\(\)](#) [items\(\)](#)

Examples

```
# STACItemCollection object
stac_item_collection <-
  stac("https://brazildatacube.dpi.inpe.br/stac/") %>%
  stac_search(collections = "CB4_64_16D_STK-1",
             bbox = c(-47.02148, -12.98314, -42.53906, -17.35063),
             limit = 15) %>%
  get_request()

print(stac_item_collection, n = 10)

# STACCollectionList object
stac_collection <-
  stac("https://brazildatacube.dpi.inpe.br/stac/") %>%
  collections() %>%
  get_request()

print(stac_collection, n = 5)

# RSTACQuery object
obj_rstac <- stac("https://brazildatacube.dpi.inpe.br/stac/")

print(obj_rstac)
```

Description

Provides functions to access, search and download spacetime earth observation data via SpatioTemporal Asset Catalog (STAC). This package supports the version 1.0.0 of the STAC specification (<<https://github.com/radiantearth/stac-spec>>). For further details see Simoes et al. (2021) <[doi:10.1109/IGARSS47720.2021](https://doi.org/10.1109/IGARSS47720.2021)>.

The rstac functions

The rstac package provides two categories of functions: API endpoints and data access and organization.

STAC API endpoints functions

- `stac()`: implements STAC `/stac` endpoint for version 0.8.1 or below, and `/` for versions 0.9.0 or higher.
- `collections()`: implements `/collections` and `/collections/{collectionId}` WFS3 endpoints.
- `items()`: implements `/collections/{collectionId}/items` and `/collections/{collectionId}/items/{featureId}` WFS3 endpoints.
- `stac_search()`: implements STAC `/stac/search` endpoint for version 0.8.1 or below, and `/search` endpoint for versions 0.9.0 or higher.

Data access and organization functions

- `get_request()`: makes HTTP GET requests to STAC web service.
- `post_request()`: makes HTTP POST requests to STAC web service.
- `items_matched()`: returns how many items matched the search criteria.
- `items_length()`: informs how many items are stored locally.
- `items_fetch()`: fetches all matched items from service.
- `assets_download()`: download all assets in batch.

Data types

The package implements the follow S3 classes: `STACItemCollection`, `STACItem`, `STACCatalog`, `STACCollectionList` and `STACCollection`. These classes are regular lists representing the corresponding JSON STAC objects.

Author(s)

Maintainer: Brazil Data Cube Team <brazildatacube@inpe.br>

Other contributors:

- National Institute for Space Research (INPE) [copyright holder]

See Also

Useful links:

- <https://github.com/brazil-data-cube/rstac>
- Report bugs at <https://github.com/brazil-data-cube/rstac/issues>

Description

Basically, there are two types of extensions in STAC specification:

1. STAC documents extensions: these extensions can be defined in different elements of the document specification.
2. STAC API extensions: these extensions are associated with the interaction between the client and server through API and may add new elements in the STAC documents or just filter the elements to be returned in the documents.

Here, we will focus on the second type of extension.

To let `rstac` package perform some behavior according to an STAC API extension we need define some functions. These functions can be implemented in three environments:

1. In `rstac` package by including new functions make a GitHub pull request on `rstac` repository (<https://github.com/brazil-data-cube/rstac>)
2. In a new package by using `rstac` as dependent package
3. In a script that loads `rstac` into the environment

All these places may impose specific requirements, however the core logic to implement an extension is the same.

These functions are intended for those who want to implement new STAC API extensions. An extension must define a subclass name and implement all the following S3 generic methods for that subclass:

- `endpoint()`: returns the endpoint value of the extension. Endpoints that vary between STAC API versions can be properly returned by checking the `version` field of `RSTACQuery` object.
- `before_request()`: allows handling query parameters before submit them to the HTTP server;
- `after_request()`: allows to check and parse document received by the HTTP server;

These methods will work 'behind the scenes' when a `RSTACQuery` object representing a user query are passed to a request function (e.g. `get_request()` or `post_request()`). The calling order is:

1. begin of `get_request()` or `post_request()`
2. if STAC API version is not defined, try detect it
3. call `endpoint()`
4. call `before_request()`
5. send HTTP request
6. receive HTTP response
7. `after_response()`
8. end of `get_request()` or `post_request()`

Besides that, the extension must expose a function to receive user parameters and return a RSTACQuery object with a subclass associated with the above S3 methods. This function must accept as its first parameter a RSTACQuery object representing the actual query. To keep the command flow consistency, the function needs to check the subclass of the input query. After that, it must set new or changes the input query parameters according to the user input and, finally, return the new query as a RSTACQuery object.

You can see examples on how to implement an STAC API extension by looking at `stac.R`, `collections.R`, `items.R`, `stac_search.R`, and `ext_query.R` source files. These files implement core STAC API endpoints, as well as the query API extension.

There are also some utility functions described in **Functions** section bellow that can help the extension development.

Usage

```
RSTACDocument(content, q, subclass)

endpoint(q)

before_request(q)

after_response(q, res)

parse_params(q, params)

content_response(res, status_codes, content_types)

check_query_verb(q, verbs, msg = NULL)

check_subclass(x, subclasses)

subclass(x)

omit_query_params(q, names)

RSTACQuery(version = NULL, base_url, params = list(), subclass)
```

Arguments

<code>content</code>	a list data structure representing the JSON file received in HTTP response (see content_response() function)
<code>q</code>	a RSTACQuery object.
<code>subclass</code>	a character corresponding to the subclass of the object to be created.
<code>res</code>	a httr response object.
<code>params</code>	a named list with all URL query parameters to be appended in the URL.
<code>status_codes</code>	a character vector with successful status codes.
<code>content_types</code>	a character vector with all acceptable responses' content type.

verbs	a character vector with allowed HTTP request methods
msg	a character with a personalized error message
x	either a RSTACQuery object expressing a STAC query criteria or any RSTACDocument.
subclasses	a character vector with all allowed S3 subclasses
names	a character vector with the names do omit.
version	a character with the STAC version.
base_url	a character informing the base URL of a STAC web service.

Value

The `RSTACDocument()` function returns a `RSTACDocument` object with subclass defined by `subclass` parameter.

A character endpoint value for `endpoint()` function. A `RSTACQuery` object for `before_request()` and `after_response()` functions.

The `content_response()` function returns a list data structure representing the JSON file received in HTTP response

The `RSTACQuery()` function returns a `STACQuery` object with subclass defined by `subclass` parameter.

Functions

- `RSTACDocument`: The `RSTACDocument()` function is a constructor of STAC documents. Currently, there are five STAC documents defined:
 - `STACCatalog`
 - `STACCollection`
 - `STACCollectionList`
 - `STACItem`
 - `STACItemCollection`

Each document class is associated with STAC API endpoints. As soon as new STAC documents are proposed in the specification, new classes can be created in the `rstac` package.

Let `version` parameter `NULL` to detect version automatically.

- `content_response`: The `content_response` function checks if the request's response is in accordance with the allowed status codes and content-types. It returns the parsed content response.
- `check_query_verb`: The `check_query_verb()` function allows you to define which HTTP verbs are allowed. It is useful for establishing which verbs will be supported by an extension.
- `check_subclass`: The `check_subclass()` function specifies which type of query objects (`RSTACQuery`) or document objects (`RSTACDocument`) are expected in the function extension.
- `subclass`: The `subclass()` function returns a character representing the subclass name of either `RSTACQuery` or `RSTACDocument` S3 classes.
- `omit_query_params`: The `omit_query_params()` function was created to omit the paths that are defined as query parameters to simplify the creation of a query. Therefore, use this method only in endpoints that specify a parameter in their paths.

- **RSTACQuery:** The `RSTACQuery()` function is a constructor of `RSTACQuery` objects. Every extension must implement a subclass of `RSTACQuery` to represent its queries. This is done by informing to the `subclass` parameter the extension's subclass name.

The `params` parameter is a named list where user parameters must be stored. It is important to know if previous query parameters needs to be kept in the new query. If so, it is recommended do use `utils::modifyList()` function to merge the old and new query parameters.

If the `version` parameter is `NULL`, `rstac` will detect STAC API version automatically.

In general, if you are implementing a new subclass, the parameters `version` and `url` will be the same as the previous query. The `params` parameter will be merged with previous query. And `subclass` is the extension's subclass name.

See Also

`ext_query()`

sign_bdc

Signature in hrefs provided by the STAC from the Brazil Data Cube project.

Description

To sign the hrefs with your token you need to store it in an environment variable in `BDC_ACCESS_KEY` or use `access_token` parameter.

Usage

```
sign_bdc(access_token = NULL, ...)
```

Arguments

`access_token` a character with the access token parameter to access Brazil Data Cube assets.
`...` additional parameters can be supplied to the GET function of the `httr` package.

Value

a function that signs each item assets.

Examples

```
## Not run:
# STACItemCollection object
stac_obj <- stac("https://brazildatacube.dpi.inpe.br/stac/") %>%
  stac_search(collections = "CB4_64_16D_STK-1",
             datetime = "2019-06-01/2019-08-01") %>%
  stac_search() %>%
  get_request()

# signing each item href
```

```
stac_obj %>% items_sign(sign_fn = sign_bdc(access_token = "123"))

## End(Not run)
```

```
sign_planetary_computer
```

Signature in hrefs provided by the STAC from Microsoft's Planetary Computer.

Description

To perform the signing of the hrefs a request is sent to Planetary Computer servers and the returned content corresponds to the token that will be used in the href.

Usage

```
sign_planetary_computer(..., token_url = NULL)
```

Arguments

...	additional parameters can be supplied to the GET function of the <code>httr</code> package.
token_url	a character with the URL that generates the tokens in the Microsoft service. By default is used: "https://planetarycomputer.microsoft.com/api/sas/v1/token"

Value

a function that signs each item assets.

Examples

```
## Not run:
# STACItemCollection object
stac_obj <- stac("https://planetarycomputer.microsoft.com/api/stac/v1/") %>%
  stac_search(collections = "sentinel-2-l2a",
             bbox = c(-47.02148, -42.53906, -12.98314, -17.35063)) %>%
  get_request()

# signing each item href
stac_obj %>% items_sign(sign_fn = sign_planetary_computer())

## End(Not run)
```

stac	<i>Endpoint functions</i>
------	---------------------------

Description

The `stac` function implements `/stac` API endpoint ($\geq 0.8.0$), and `/` for versions 0.9.0 or higher. It prepares search fields parameters to be provided to a STAC API web service. This endpoint should return a STAC Catalog document containing all published data catalogs.

Usage

```
stac(base_url, force_version = NULL)
```

Arguments

<code>base_url</code>	a character informing the base url of a STAC web service.
<code>force_version</code>	a character providing the version of the STAC used. If not provided, the <code>rstac</code> package will make requests to try to find the version of STAC used. It is highly recommended that you inform the STAC version you are using.

Value

A `RSTACQuery` object with the subclass `stac` containing all request parameters to be provided to API service.

See Also

[stac_search\(\)](#), [collections\(\)](#), [items\(\)](#), [get_request\(\)](#), [post_request\(\)](#)

Examples

```
stac("https://brazildatacube.dpi.inpe.br/stac/") %>%  
  get_request()
```


Description

(This document is based on STAC specification documentation <https://github.com/radiantearth/stac-spec/> and reproduces some of its parts)

The `stac_search` function implements `/stac/search` API endpoint (v0.8.1) and `/search` (v0.9.0 or v1.0.0). It prepares query parameters used in search API request, a `stac` object with all filter parameters to be provided to `get_request` or `post_request` functions. The GeoJSON content returned by these requests is a `STACItemCollection` object, a regular R list representing a STAC Item Collection document.

Usage

```
stac_search(  
  q,  
  collections = NULL,  
  ids = NULL,  
  bbox = NULL,  
  datetime = NULL,  
  intersects = NULL,  
  limit = NULL  
)
```

Arguments

<code>q</code>	a <code>RSTACQuery</code> object expressing a STAC query criteria.
<code>collections</code>	a character vector of collection IDs to include in the search for items. Only items in one of the provided collections will be searched.
<code>ids</code>	a character vector with item IDs. All other filter parameters that further restrict the number of search results are ignored.
<code>bbox</code>	a numeric vector with only features that have a geometry that intersects the bounding box are selected. The bounding box is provided as four or six numbers, depending on whether the coordinate reference system includes a vertical axis (elevation or depth): <ul style="list-style-type: none">• Lower left corner, coordinate axis 1• Lower left corner, coordinate axis 2• Lower left corner, coordinate axis 3 (optional)• Upper right corner, coordinate axis 1• Upper right corner, coordinate axis 2• Upper right corner, coordinate axis 3 (optional)

The coordinate reference system of the values is WGS84 longitude/latitude (<http://www.opengis.net/def/crs/OGC/1.3/CRS84>). The values are in most cases the sequence of minimum longitude, minimum latitude, maximum longitude and maximum latitude. However, in cases where the box spans the antimeridian the first value (west-most box edge) is larger than the third value (east-most box edge).

datetime	<p>a character with a date-time or an interval. Date and time strings needs to conform RFC 3339. Intervals are expressed by separating two date-time strings by '/' character. Open intervals are expressed by using '..' in place of date-time.</p> <p>Examples:</p> <ul style="list-style-type: none"> • A date-time: "2018-02-12T23:20:50Z" • A closed interval: "2018-02-12T00:00:00Z/2018-03-18T12:31:12Z" • Open intervals: "2018-02-12T00:00:00Z/.." or "../2018-03-18T12:31:12Z" <p>Only features that have a datetime property that intersects the interval or date-time informed in datetime are selected.</p>
intersects	a character value expressing GeoJSON geometries objects as specified in RFC 7946. Only returns items that intersect with the provided polygon.
limit	an integer defining the maximum number of results to return. If not informed it defaults to the service implementation.

Value

A RSTACQuery object with the subclass search containing all search field parameters to be provided to STAC API web service.

See Also

[stac\(\)](#), [ext_query\(\)](#), [get_request\(\)](#), [post_request\(\)](#)

Examples

```
# GET request
stac("https://brazildatacube.dpi.inpe.br/stac/") %>%
  stac_search(collections = "CB4_64_16D_STK-1", limit = 10,
             datetime = "2017-08-01/2018-03-01") %>%
  get_request()

# POST request
stac("https://brazildatacube.dpi.inpe.br/stac/") %>%
  stac_search(collections = "CB4_64_16D_STK-1",
             bbox = c(-47.02148, -12.98314, -42.53906, -17.35063)) %>%
  post_request()
```

utilities	<i>Utility functions</i>
-----------	--------------------------

Description

These function retrieves information about either `rstac` queries (RSTACQuery objects) or `rstac` documents (RSTACDocument objects).

Usage

```
stac_version(x, ...)
```

Arguments

<code>x</code>	either a RSTACQuery object expressing a STAC query criteria or any RSTACDocument.
<code>...</code>	config parameters to be passed to GET method, such as add_headers or set_cookies .

Value

The `stac_version()` function returns a character STAC API version.

<code>%>%</code>	<i>Pipe</i>
---------------------	-------------

Description

Magrittr compound assignment pipe-operator.

Arguments

<code>lhs, rhs</code>	A visualization and a function to apply to it.
-----------------------	--

Index

`%>%`, 27

`add_headers`, 3, 8, 13, 27

`after_response` (RSTACDocument), 19

`after_response()`, 7

`assets_download`, 2

`assets_download()`, 18

`assets_filter` (assets_function), 3

`assets_function`, 3

`assets_list` (assets_function), 3

`assets_select` (assets_function), 3

`before_request` (RSTACDocument), 19

`before_request()`, 7

`check_query_verb` (RSTACDocument), 19

`check_subclass` (RSTACDocument), 19

`collections`, 5

`collections()`, 8, 9, 16–18, 24

`content_response` (RSTACDocument), 19

`content_response()`, 7, 20

`doc_query`, 6

`endpoint` (RSTACDocument), 19

`endpoint()`, 7

`ext_query`, 6

`ext_query()`, 22, 26

`extensions` (RSTACDocument), 19

GET, 3, 8, 13, 27

`get_request`, 7

`get_request()`, 3, 5, 9, 18, 24, 26

`items`, 8

`items()`, 3, 5, 8, 16–18, 24

`items_assets`, 10

`items_assets` (items_functions), 11

`items_bands`, 10

`items_bbox` (items_functions), 11

`items_datetime` (items_functions), 11

`items_fetch` (items_functions), 11

`items_fetch()`, 18

`items_fields` (items_functions), 11

`items_filter` (items_functions), 11

`items_functions`, 11

`items_group` (items_functions), 11

`items_length` (items_functions), 11

`items_length()`, 18

`items_matched` (items_functions), 11

`items_matched()`, 18

`items_next` (items_functions), 11

`items_reap` (items_functions), 11

`items_sign` (items_functions), 11

`omit_query_params` (RSTACDocument), 19

`parse_params` (RSTACDocument), 19

POST, 3, 8, 13

`post_request` (get_request), 7

`post_request()`, 5, 7, 9, 18, 24, 26

`print`, 16

`rstac`, 17

`rstac-package` (rstac), 17

RSTACDocument, 19

RSTACQuery (RSTACDocument), 19

`set_cookies`, 3, 8, 13, 27

`sign_bdc`, 22

`sign_planetary_computer`, 23

`stac`, 24

`stac()`, 8, 16–18, 26

`stac_search`, 25

`stac_search()`, 3, 7, 8, 16–18, 24

`stac_version` (utilities), 27

`subclass` (RSTACDocument), 19

`utilities`, 27

`utils::modifyList()`, 22