Package 'rstac'

December 14, 2020

Title Client Library for SpatioTemporal Asset Catalog

Version 0.9.0

Description Provides functions to access, search and download spacetime earth observation data via SpatioTemporal Asset Catalog (STAC). This package supports the version 0.8.1 or higher of the STAC specification (<http://stacspec.org>).

License MIT + file LICENSE

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

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

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Depends R (>= 3.2)

Imports httr, crayon, utils, magrittr, jsonlite

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 2020-12-14 09:10:03 UTC

R topics documented:

`assets_download` *Downloads assets via STAC API*

Description

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

Usage

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

Arguments

`items`	a `STACItem` or `STACItemCollection` object representing the result of `/stac/search`, `/collections/{collectionId}/items` or `/collections/{collectionId}/items/{itemId}` endpoints.
`assets_name`	a `character` with the assets names to be filtered.
`output_dir`	a `character` directory in which the assets will be saved.
`overwrite`	a `logical` if TRUE will replaced the existing file, if FALSE a warning message is shown.
`items_max`	a `numeric` corresponding how many items will be downloaded.
`progress`	a `logical` indicating if a progress bar must be shown or not. Defaults to `TRUE`.
`...`	config parameters to be passed to GET or POST methods, such as add_headers or set_cookies.

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("http://brazildatacube.dpi.inpe.br/stac/") %>%
  stac_search(collections = "CB4_64_16D_STK-1") %>%
  stac_search(limit = 2) %>%
  get_request() %>%
  assets_download(assets_name = "thumbnail", output_dir = ".",
  overwrite = FALSE)

## End(Not run)
```

---

assets_list *Utility functions*

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, assets_names, sort = TRUE, gdal_vsi_resolution = TRUE)
```

Arguments

items	a STACItemCollection object representing the result of /stac/search, /collections/{collectionId
assets_names	a character with the assets names to be filtered.
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).

Value

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

Examples

```
# STACItemCollection object
stac_item <- stac("http://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 (v0.8.0, v0.8.1 and v0.9.0).

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("http://brazildatacube.dpi.inpe.br/stac/") %>%
  collections() %>%
  get_request()

stac("http://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	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("http://brazildatacube.dpi.inpe.br/stac/") %>%
  stac_search(collections = "CB4_64_16D_STK-1") %>%
  ext_query("bdc:tile" %in% c("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("http://brazildatacube.dpi.inpe.br/stac/") %>%
 get_request()

stac("http://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}/ite endpoints (v0.8.0, v0.8.1 and v0.9.0).

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 datetime 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("http://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("http://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_fields *Utility functions*

Description

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

Usage

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

Arguments

items	a STACItemCollection object representing the result of /stac/search, /collections/{collectionId
...	a named way to provide field names to get the subfields values from the RSTACDocument objects.
field	a character with the names of the field to get the subfields values from the RSTACDocument objects.

Value

a character with the subfields of the feature field.

Examples

```
# STACItemCollection object
stac_item <- stac("http://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"))
```

---

items_functions *STACItemCollection functions*

Description

The items_length() function shows how many items there are in the STACItemCollection object. The items_matched() function shows how many items matched the search criteria. It support search:metadata (v0.8.0) and context (v0.9.0) STAC API extensions. The items_fetch() function request all STAC Items through pagination. The items_datetime() function retrieves a the datetime field in properties from STACItemCollection and STACItem objects. The items_bbox() function retrieves a the bbox field of a STACItemCollection or an STACItem object. The get_assets_name() function returns the assets name from STACItemCollection and STACItem objects.

Usage

```
items_length(items)

items_matched(items)

items_fetch(items, ..., progress = TRUE)

items_datetime(items)
```

```
items_bbox(items)

items_bands(items)
```

Arguments

items	a STACItemCollection object.
...	config parameters to be passed to GET or POST methods, such as add_headers or set_cookies.
progress	a logical indicating if a progress bar must be shown or not. Defaults to TRUE.

Value

The items_length() returns an integer value. The items_matched() returns an integer value. If STAC web server does not support this extension, returns NULL. The items_fetch() returns an STACItemCollection with all matched items. The items_datetime() returns a list of all items' datetime. The items_bbox() returns a list with all items' bounding boxes.

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_length()
x %>% items_matched()
x %>% items_datetime()
x %>% items_bbox()

## End(Not run)

## Not run:
x <-
  stac("http://brazildatacube.dpi.inpe.br/stac") %>%
  stac_search(collections = "CB4_64_16D_STK-1") %>%
  stac_search(limit = 500) %>%
  get_request()

x %>% items_fetch()

## End(Not run)
```

items_group *Utility functions*

Description

This function groups the items contained within the STACItemCollection object according to some specified fields. Each index in the returned list contains items belonging to the same group.

Usage

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

Arguments

items	a STACItemCollection object representing the result of /stac/search, /collections/{collectionId
...	a named way to provide field names to get the subfields values from the RSTACDocument objects.
field	a character with the names of the field to get the subfields values from the RSTACDocument objects.
index	a character with the indexes to be grouped. It can be used with the function items_reap.

Value

a list in which each index corresponds to a group with its corresponding STACItemCollection objects.

Examples

```
# STACItemCollection object
stac_item <- stac("http://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"))
```

items_reap *Utility functions*

Description

This function returns the values of a field of the STACItemCollections object. If the values of the specified field are not atomic the return will be in list form, if they are, it will be returned in vector form.

Usage

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

Arguments

items	a STACItemCollection object representing the result of /stac/search, /collections/{collectionId
...	a named way to provide fields names to get the subfields values from the RSTACDocument objects.
field	a character with the names of the field to get the subfields values from the RSTACDocument objects.

Value

a vector if the supplied field is atomic, or a list if not.

Examples

```
# STACItemCollection object
stac_item <- stac("http://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"))
```

print *Printing functions*

Description

The print function covers all objects in the rstac package:

- stac: Returns a STACCatalog document in /stac (V0.8.0 or below) or / (V0.9.0 or above) endpoint.
- stac_search: Returns a STACItemCollection document in /stac/search (V0.8.0 or below) or /search (V0.9.0 or above) endpoint with a group of Items matching the provided search predicates.
- collections: Return a STACCollectionList document by listing of collections contained in the catalog in /collections endpoint and in /collections/{collectionId} endpoint return a single STACCollection document.
- items: Return a STACItemCollection document in /collections/{collectionId}/items and a STACItem document in /collections/{collectionId}/items/{itemId} WFS3 endpoints.

The rstac package objects visualization is based on **Markdown**, a lightweight markup language, so you can paste the output into any **Markdown** editor for a better visualization.

For printing use the print() function directly, since the package has a generic implementation for its objects. For console output control, you have the option to determine how many items you want to see through the n in print objects parameters, the following objects have the n parameter:

- items
- collections
- stac

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

x	either a RSTACQuery object expressing a STAC query criteria or any RSTACDocument.
...	other parameters passed in the functions.
n	number of lines to view on each object. Each object has its own type of truncation in lines. In the stac_collection object, by default, 10 links will be shown, but if the object has less than 20 collections, all the collections will be displayed. In STACItemCollection, by default, 10 features will be shown. If you want to show all lines of a rstac object, use 'n = Inf'.
tail	To show the last lines of an object.

See Also

stac stac_search collections items

Examples

```
# STACItemCollection object
stac_item_collection <-
  stac("http://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("http://brazildatacube.dpi.inpe.br/stac/") %>%
    collections() %>%
    get_request()

print(stac_collection, n = 5)

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

print(obj_rstac)
```

---

rstac *R client library for STAC (rstac)*

Description

Provides functions to access, search and download spacetime earth observation data via SpatioTemporal Asset Catalog (STAC). This package supports the version 0.8.1 or higher of the STAC specification (<http://stacspec.org>).

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/{feature` 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`

RSTACDocument *Document development functions*

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)

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

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

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 keeped in the new query. If so, it is recommended do use `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`

`stac` *Endpoint functions*

Description

The `stac` function implements `/stac` API endpoint (>=0.8.0). 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

`base_url`	a `character` informing the base url of a STAC web service.
`force_version`	a `character` providing the version of the stac used. If not provided, the rstac 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("http://brazildatacube.dpi.inpe.br/stac/") %>%
  get_request()
```

`stac_search` *Endpoint functions*

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). 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

`q`	a `RSTACQuery` object expressing a STAC query criteria.
`collections`	a `character` vector of collection IDs to include in the search for items. Only items in one of the provided collections will be searched.
`ids`	a `character` vector with item IDs. All other filter parameters that further restrict the number of search results are ignored.
`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).

`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.

`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("http://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("http://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` *Utility functions*

Description

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

Usage

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

Arguments

`x`	either a `RSTACQuery` object expressing a STAC query criteria or any `RSTACDocument`.
`...`	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.

`%>%` *Pipe*

Description

Magrittr compound assignment pipe-operator.

Arguments

`lhs, rhs`	A visualization and a function to apply to it.

Index