Package: hubeau (via r-universe)

September 12, 2024

Type Package

Title Get Data from the French National Database on Water 'Hub'Eau'

Version 0.5.0.9000

Date 2024-03-04

Description Collection of functions to help retrieving data from 'Hub'Eau' the free and public French National APIs on water <https://hubeau.eaufrance.fr/>.

License MIT + file LICENSE

URL https://inrae.github.io/hubeau/, https://github.com/inrae/hubeau#readme

BugReports https://github.com/inrae/hubeau/issues

Depends R (>= 3.5.0)

Imports dplyr, httr, magrittr, purrr, tibble, urltools, utils

Suggests ggplot2, Hmisc, knitr, leafpop, lubridate, mapview, rmarkdown, sf, spelling, testthat (>= 3.0.0), tools

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

Language en-US

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

Repository https://inrae.r-universe.dev

RemoteUrl https://github.com/inrae/hubeau

RemoteRef HEAD

RemoteSha 55899d0a2856b0b87165dfd855c655d985debbac

Contents

`convert_list_to_tibble` *Convert list provided by the APIs into a tibble*

Description

Convert list provided by the APIs into a tibble

Usage

```
convert_list_to_tibble(l)
```

Arguments

`l`	a list provided by the API (See doApiQuery)

Details

This function is used internally by all the retrieving data functions for converting data after the call to doApiQuery.

Value

A tibble::tibble with one row by record and one column by field.

Examples

```
# To get the available APIs in the package
list_apis()

# To get the available endpoints in an API
list_endpoints("prelevements")

# To get available parameters in endpoint "chroniques" of the API "prelevements"
list_params(api = "prelevements", endpoint = "chroniques")

# To query the endpoint "chroniques" of the API "prelevements"
# on all devices in the commune of Romilly-sur-Seine in 2018
## Not run:
resp <- doApiQuery(api = "prelevements",
                   endpoint = "chroniques",
                   code_commune_insee = "10323",
                   annee = "2018")
convert_list_to_tibble(resp)

## End(Not run)
```

`doApiQuery` *Main internal functions for querying the Hub'Eau API endpoints*

Description

The function `doQueryApi` is called by all the function querying the API endpoints and return the raw data sent by the endpoint.

Usage

```
doApiQuery(api, endpoint, ..., params)
```

Arguments

`api`	a character name of the API (e.g.: "indicateurs_services", "prelevements"...), see example for available APIs
`endpoint`	a character name of the endpoint, see example for available endpoints in an API
`...`	parameters of the queries and their values in the format `Param1_Name = "Param1 value", Param2_Name` use the function list_params for a list of the available filter parameters for a given API endpoint and see the API documentation for their description
`params`	(deprecated) a list the list of parameters of the queries and their values in the format `list(ParamName = "Param value", ...)`. This parameter is replaced by the parameter `...`

Details

Pagination of the queries is handled automatically and the returned list is the concatenation of all the results sent by the API.

The functions `get_[api]_[endpoint]` call the function `doQueryApi` and parse the response in a tibble::tibble format for the user (See convert_list_to_tibble).

By default the user agent used for the query is "https://github.com/inrae/hubeau". You can redefined the user agent with the global option "hubeau.user_agent": `options(hubeau.user_agent = "My user agent")`.

Value

A list with the concatenated results returned by the API.

Examples

```
# To get the available APIs in the package
list_apis()

# To get the available endpoints in an API
list_endpoints("prelevements")

# To get available parameters in endpoint "chroniques" of the API "prelevements"
list_params(api = "prelevements", endpoint = "chroniques")

# To query the endpoint "chroniques" of the API "prelevements"
# on all devices in the commune of Romilly-sur-Seine in 2018
## Not run:
resp <- doApiQuery(api = "prelevements",
                   endpoint = "chroniques",
                   code_commune_insee = "10323",
                   annee = "2018")
convert_list_to_tibble(resp)

## End(Not run)
```

`get_ecoulement_stations` *Retrieve data from API "Ecoulement des cours d'eau"*

Description

The data originate from the "ONDE" river low waters monitoring network. Available endpoints are:

- `get_ecoulement_stations` retrieves site data and locations
- `get_ecoulement_observations` retrieves flow information
- `get_ecoulement_campagnes` retrieves annual surveys

See the API documentation for available filter parameters: https://hubeau.eaufrance.fr/page/api-ecoulement

Usage

```
get_ecoulement_stations(...)

get_ecoulement_observations(...)

get_ecoulement_campagnes(...)
```

Arguments

...	parameters of the queries and their values in the format `Param1_Name = "Param1 value", Param2_Name` use the function list_params for a list of the available filter parameters for a given API endpoint and see the API documentation for their description

Value

A tibble::tibble with one row by record and one column by field.

Examples

```
## Not run:
# Retrieve 2022 observation campaigns in the Jura French department
get_ecoulement_campagnes(
  list(code_departement = "39",
       date_campagne_min = "2022-01-01",
       date_campagne_max = "2022-12-31")
)

# Retrieve river stations
stations_39 <- get_ecoulement_stations(
  list(code_departement = "39",
       fields = "code_station,libelle_cours_eau,libelle_commune")
)
stations_39

# Get the query parameters for the requested API/endpoint
list_params(api = "ecoulement",
                    endpoint = "observations")

# Retrieve the river flow data in the Jura departement in 2022 with
# a selection of the fields
onde_39 <- get_ecoulement_observations(
  list(code_departement = "39",
       date_observation_min = "2022-01-01",
       date_observation_max = "2022-12-31",
       fields = "code_station,libelle_station,date_observation,libelle_ecoulement")
)
onde_39

## End(Not run)
```

`get_hydrobio_stations_hydrobio` *Retrieve data from API "Hydrobiologie"*

Description

The data originate from the "NAIADES" database. Available endpoints are:

- `get_hydrobio_stations_hydrobio` retrieves site data and locations
- `get_hydrobio_indices` retrieves bioassessment indices values
- `get_hydrobio_taxons` retrieves taxa data

See the API documentation for available filter parameters: https://hubeau.eaufrance.fr/page/api-hydrobiologie

Usage

```
get_hydrobio_stations_hydrobio(...)

get_hydrobio_indices(...)

get_hydrobio_taxons(...)
```

Arguments

`...`	parameters of the queries and their values in the format `Param1_Name = "Param1 value", Param2_Name` use the function list_params for a list of the available filter parameters for a given API endpoint and see the API documentation for their description

Value

A tibble::tibble with one row by record and one column by field.

Examples

```
## Not run:
# Retrieve the hydrobiology monitoring sites in the Pays-de-Loire region
list_params(api = "hydrobio",
            endpoint = "stations_hydrobio")

get_hydrobio_stations_hydrobio(code_region = 52)

# Retrieve the hydrobiological bioassessment indices in the city of Rennes
list_params(api = "hydrobio",
            endpoint = "indices")

get_hydrobio_indices(code_commune = 35051)
```

```
# species records in the city of Rennes
list_params(api = "hydrobio",
            endpoint = "taxons")

get_hydrobio_taxons(code_commune = 35051)

## End(Not run)
```

`get_hydrometrie_obs_elab` *Retrieve data from API "Hydrométrie"*

Description

Available endpoints are:

- `get_hydrometrie_obs_elab` retrieves hydrometric elaborate observations (daily/monthly mean flow)
- `get_hydrometrie_observations_tr` retrieves hydrometric "real time" observations ()
- `get_hydrometrie_sites` retrieves hydrometric sites
- `get_hydrometrie_stations` retrieves hydrometric stations

See the API documentation of each endpoint for available filter parameters: https://hubeau.eaufrance.fr/page/api-hydrometrie

Usage

```
get_hydrometrie_obs_elab(...)

get_hydrometrie_observations_tr(..., entities = "station")

get_hydrometrie_sites(..., unique_site = TRUE)

get_hydrometrie_stations(..., code_sandre_reseau_station = FALSE)
```

Arguments

`...`	parameters of the queries and their values in the format `Param1_Name = "Param1 value", Param2_Name` use the function list_params for a list of the available filter parameters for a given API endpoint and see the API documentation for their description
`entities`	1-length character string filtering the rows of the returned value, possible values are: "station" for filtering on station rows, "site" for filtering on site rows, "both" for keeping all the rows
`unique_site`	optional logical, if set to `FALSE` sites with several different locations produce one row by different location otherwise the first location found is used for fields `code_commune_site`, `libelle_commune`, `code_departement`, `code_region`, `libelle_region`, `libelle_departement`

code_sandre_reseau_station
: optional logical indicating if code_sandre_reseau_station field is included in the result; if so, one line is added by item and other fields are repeated

Value

A tibble::tibble with one row by record and one column by field.

Examples

```
## Not run:
# Retrieve the hydrometric sites in the department of Aube
get_hydrometrie_sites(code_departement = "10")

# The same operation returning 2 rows for the site 'H0203020' which has 2 different locations
get_hydrometrie_sites(code_departement = "10", unique_site = FALSE)

# Retrieve the hydrometric stations in the department of Aube
get_hydrometrie_stations(code_departement = "10")

# Which parameters are available for endpoint "obs_elab" of API "hydrometrie"?
list_params("hydrometrie", "obs_elab")

# Retrieve the hydrometric monthly mean flow at site 'H0203020'
get_hydrometrie_obs_elab(code_entite = "H0203020", grandeur_hydro_elab = "QmM")

# Retrieve the hydrometric daily mean flow at site 'H0203020' of the last 30 days
get_hydrometrie_obs_elab(code_entite = "H0203020",
                         date_debut_obs_elab = format(Sys.Date() -30, "%Y-%m-%d"),
                         grandeur_hydro_elab = "QmJ")

## End(Not run)
```

---

get_indicateurs_services_communes

Retrieve data from API "Indicateurs des services"

Description

Retrieve performance indicators collected in drinking water and sanitation services in France.

See list_params and the API documentation for available filter parameters: https://hubeau.eaufrance.fr/page/api-indicateurs-services

Usage

```
get_indicateurs_services_communes(...)

get_indicateurs_services_indicateurs(...)

get_indicateurs_services_services(...)
```

Arguments

...	parameters of the queries and their values in the format `Param1_Name = "Param1 value", Param2_Name` use the function list_params for a list of the available filter parameters for a given API endpoint and see the API documentation for their description

Value

`get_indicateurs_services_communes` returns a tibble::tibble with one row by commune, by service and by year and the following columns:

- "code_commune_insee": character identifier of the commune
- "nom_commune": character name of the commune
- "codes_service": integer identifier of the drinking water supply and/or sanitation service
- "annee": integer year of the data
- The following columns are the performance indicators flagged by their respective codes. The documentation of these codes can be found at this URL: https://www.services.eaufrance.fr/indicateurs/indicateurs.

`get_indicateurs_services_indicateurs` returns a tibble::tibble with one row by service and by year and the following columns:

- "code_service": character identifier of the service
- "nom_service": character name of the service
- "numero_siren ": character SIREN identifier of the service
- "type_collectivite": character kind of community
- "mode_gestion": character management mechanism of the service
- "annee": integer year of the data
- "indicateur": value of the indicator
- "uri_indicateur": the link to the indicator documentation

`get_indicateurs_services_services` returns a tibble::tibble with one row by commune, by service and by year and the following columns:

- "code_service": character identifier of the service
- "nom_service": character name of the service
- "code_commune_insee": character identifier of the commune
- "nom_commune": character name of the commune
- "numero_siren ": character SIREN identifier of the service
- "type_collectivite": character kind of community
- "mode_gestion": character management mechanism of the service
- "annee": integer year of the data
- The following columns are the performance indicators flagged by their respective codes. The documentation of these codes can be found at this URL: https://www.services.eaufrance.fr/indicateurs/indicateurs.

Examples

```
## Not run:
# Retrieve performance indicator time series in the commune of Romilly-sur-Seine
get_indicateurs_services_communes(code_commune = "10323")

# Retrieve the drinking water withdrawal indicators of the year 2012 for all services
get_indicateurs_services_indicateurs(code_indicateur = "D102.0", annee = "2012")

# Retrieve performance indicator time series of Romilly-sur-Seine with service details
get_indicateurs_services_services(code_commune = "10323")

## End(Not run)
```

---

get_niveaux_nappes_stations

Retrieve data from API "Piézométrie"

Description

The available endpoints are:

- `get_niveaux_nappes_stations` retrieves list of piezometric stations
- `get_nappes_chroniques` retrieves piezometric archived time series
- `get_nappes_chroniques_tr` retrieves piezometric "real time" data

See the API documentation for available filter parameters: https://hubeau.eaufrance.fr/page/api-piezometrie

Usage

```
get_niveaux_nappes_stations(...)

get_niveaux_nappes_chroniques(...)

get_niveaux_nappes_chroniques_tr(...)
```

Arguments

`...`	parameters of the queries and their values in the format `Param1_Name = "Param1 value", Param2_Name` use the function list_params for a list of the available filter parameters for a given API endpoint and see the API documentation for their description

Value

A tibble::tibble with one row by record and one column by field.

Examples

```
## Not run:
# Retrieve the hydrometric stations in the department of Aube
get_niveaux_nappes_stations(code_departement = "10")

# Retrieve the archived observed piezometric level at station '07548X0009/F' (old BSS identifier)
# for the year 2020
df <- get_niveaux_nappes_chroniques(code_bss = "07548X0009/F",
                                    date_debut_mesure = "2020-01-01",
                                    date_fin_mesure = "2020-12-31")

# Plot the water elevation (NGF)
plot(as.POSIXct(df$date_mesure), df$niveau_nappe_eau, type = "l")

# For retrieving the last real time observed piezometric level
# at station 'BSS001VZGZ' (new BSS identifier)
df <- get_niveaux_nappes_chroniques_tr(bss_id = "BSS001VZGZ")

# Plot the water elevation (NGF)
plot(as.POSIXct(df$date_mesure), df$niveau_eau_ngf, type = "l")

## End(Not run)
```

`get_poisson_observations`

Retrieve data from API "Poisson"

Description

Available endpoint:

- `get_poisson_observations` retrieves data of scientific fishery operations

See the API documentation for available filter parameters: https://hubeau.eaufrance.fr/page/api-poisson

Usage

```
get_poisson_observations(...)

get_poisson_indicateurs(...)

get_poisson_operations(...)

get_poisson_stations(...)
```

Arguments

`...`	parameters of the queries and their values in the format `Param1_Name = "Param1 value", Param2_Name` use the function list_params for a list of the available filter parameters for a given API endpoint and see the API documentation for their description

Value

A tibble::tibble with one row by record and one column by field.

Examples

```
## Not run:
# Get the endpoints available for the API "Poisson"
list_endpoints(api = "poisson")

# List the stations available in Brest
get_poisson_stations(libelle_commune = "Brest")

# List the operations available in Brest
get_poisson_operations(libelle_commune = "Brest")

# List the indicators available in Brest
get_poisson_indicateurs(libelle_commune = "Brest")

# Get the query parameters for the requested API/endpoint
list_params(api = "poisson",
                     endpoint = "observations")

# Retrieve selected fields on a river fish sampled in Brest
library(dplyr)
fields <- c("code_operation",
                "date_operation",
                "libelle_point_prelevement_aspe",
                "effectif_lot",
                "code_alternatif_taxon")

brest_fishes <- get_poisson_observations(
  list(
    libelle_commune = "Brest",
    fields = fields
    )
 ) %>%
 group_by_at(vars(-effectif_lot)) %>%
   summarise(nb_individals = sum(effectif_lot))

brest_fishes

## End(Not run)
```

`get_prelevements_points_prelevement`

Retrieve data from API "Prélèvements en eau"

Description

Available endpoints are:

- `get_prelevements_points_prelevement` retrieves withdrawal points
- `get_prelevements_ouvrages` retrieves withdrawal devices
- `get_prelevements_chroniques` retrieves time series of withdrawals

See the API documentation for available filter parameters: https://hubeau.eaufrance.fr/page/api-prelevements-eau

Usage

```
get_prelevements_points_prelevement(...)

get_prelevements_ouvrages(...)

get_prelevements_chroniques(...)
```

Arguments

`...`	parameters of the queries and their values in the format `Param1_Name = "Param1 value", Param2_Name` use the function list_params for a list of the available filter parameters for a given API endpoint and see the API documentation for their description

Value

A tibble::tibble with one row by record and one column by field.

Examples

```
## Not run:
# Retrieve the withdrawal points located in Romilly-sur-Seine
get_prelevements_points_prelevement(code_commune_insee = "10323")

# Retrieve the withdrawal devices located in Romilly-sur-Seine
get_prelevements_ouvrages(code_commune_insee = "10323")

# Retrieve the withdrawal time series of the devices located in Romilly-sur-Seine
get_prelevements_chroniques(code_commune_insee = "10323")

## End(Not run)
```

`get_qualite_eau_potable_communes_udi` *Retrieve data from API "Qualité de l'eau potable"*

Description

Results of the sanitary control of the distributed water commune by commune: samples and results of the analyses carried out within the framework of the regulatory sanitary control on the distribution units or the installations directly upstream, and links between communes and distribution units. The elements made available in this dataset correspond to a compilation of analysis bulletins published online, commune by commune, on the website of the Ministry of Health: https://sante.gouv.fr/sante-et-environnement/eaux/eau

Available endpoints are:

- `get_qualite_eau_potable_communes_udi` retrieves links between "UDI" (Distribution units or networks) and communes
- `get_qualite_eau_potable_resultats_dis` retrieves samples, analysis results and sanitary conclusions from the sanitary control of the distributed water commune by commune

See the API documentation for available filter parameters: https://hubeau.eaufrance.fr/page/api-qualite-eau-potable

Usage

```
get_qualite_eau_potable_communes_udi(...)

get_qualite_eau_potable_resultats_dis(...)
```

Arguments

`...`	parameters of the queries and their values in the format `Param1_Name = "Param1 value", Param2_Name` use the function list_params for a list of the available filter parameters for a given API endpoint and see the API documentation for their description

Value

A tibble::tibble with one row by record and one column by field.

Examples

```
## Not run:
# List of available filter parameters on 'get_qualite_eau_potable_communes_udi'
list_params("qualite_eau_potable", "communes_udi")

# List of UDIs available in 2022 at Grabels (INSEE code 34116)
get_qualite_eau_potable_communes_udi(annee = 2022, code_commune = 34116)

# List of available filter parameters on 'get_qualite_eau_potable_resultats_dis'
```

```
list_params("qualite_eau_potable", "resultats_dis")

# Get results of analysis realised at Grabels in 2022
get_qualite_eau_potable_resultats_dis(code_commune = 34116,
                                      date_min_prelevement = "2000-01-01",
                                      date_max_prelevement = "2022-12-31")

## End(Not run)
```

`get_qualite_nappes_analyses` *Retrieve data from API "Qualité des nappes d'eau souterraines"*

Description

The data originate from the "ADES" database (https://ades.eaufrance.fr/). Available endpoints are:

- `get_qualite_nappes_stations` retrieves measuring stations for groundwater quality
- `get_qualite_nappes_analyses` retrieves the result analysis of quality measurement

See the API documentation for available filter parameters: https://hubeau.eaufrance.fr/page/api-qualite-nappes

Usage

```
get_qualite_nappes_analyses(...)

get_qualite_nappes_stations(...)
```

Arguments

`...`	parameters of the queries and their values in the format `Param1_Name = "Param1 value", Param2_Name` use the function list_params for a list of the available filter parameters for a given API endpoint and see the API documentation for their description

Value

A tibble::tibble with one row by record and one column by field.

Examples

```
## Not run:
# List of available filter parameters on 'get_qualite_nappes_stations'
list_params("qualite_nappes", "stations")

# List of stations available in Hérault department
get_qualite_nappes_stations(code_commune = 34116)
```

```
# List of available filter parameters on 'get_qualite_nappes_analyses'
list_params("qualite_nappes", "analyses")

# Get results of analysis realised at Grabels in 2019
get_qualite_nappes_analyses(bss_id = "BSS002GNSA",
                            date_debut_prelevement = "2019-11-12")

## End(Not run)
```

`get_qualite_rivieres_analyse`
Retrieve data from API "Qualité physico-chimique des cours d'eau"

Description

The "Quality of rivers" API data comes from the Naïades database. Available for the whole of France (including the French overseas departments and territories), they relate to the results of measurements of the physico-chemical quality of rivers and water bodies transmitted by the Water Agencies.

Available endpoints are:

- `get_qualite_rivieres_station` retrieves physico-chemical measuring stations
- `get_qualite_rivieres_operation` retrieves physico-chemical operations
- `get_qualite_rivieres_analyse` retrieves physico-chemical analyses
- `get_qualite_rivieres_condition_environnementale` retrieves physico-chemical environmental conditions for analyses (please note that pagination is based on the number of analyses)

See the API documentation for available filter parameters: https://hubeau.eaufrance.fr/page/api-qualite-cours-deau

Usage

```
get_qualite_rivieres_analyse(...)

get_qualite_rivieres_analyse_pc(...)

get_qualite_rivieres_condition_environnementale(...)

get_qualite_rivieres_condition_environnementale_pc(...)

get_qualite_rivieres_operation(...)

get_qualite_rivieres_operation_pc(...)

get_qualite_rivieres_station(...)

get_qualite_rivieres_station_pc(...)
```

Arguments

...	parameters of the queries and their values in the format `Param1_Name = "Param1 value", Param2_Name` use the function list_params for a list of the available filter parameters for a given API endpoint and see the API documentation for their description

Details

Endpoints ending by _pc are aliases of the functions without the suffix _pc.

Value

A tibble::tibble with one row by record and one column by field.

Examples

```
## Not run:
# Tutorial provided by https://hubeau.eaufrance.fr/page/api-qualite-cours-deau-tuto

# List of stations in Longuyon
get_qualite_rivieres_station(libelle_commune="Longuyon")
# ... with field selection
get_qualite_rivieres_station(
  libelle_commune = "Longuyon",
  fields = c("code_station",
             "libelle_station",
             "localisation_precise",
             "code_commune",
             "libelle_commune",
             "code_cours_eau",
             "nom_cours_eau")
)

# Query results of nitrate analyses at the station since 2013
get_qualite_rivieres_analyse(
  code_station = "02115725",
  libelle_parametre = "Nitrates",
  date_debut_prelevement = "2013-01-01",
  code_qualification = 1,
  fields = c("code_station",
             "libelle_station",
             "code_parametre",
             "libelle_parametre",
             "date_prelevement",
             "resultat",
             "symbole_unite",
             "code_remarque",
             "mnemo_remarque",
             "code_statut",
             "mnemo_statut",
             "code_qualification",
             "libelle_qualification")
)
```

```
# dates sorted from most recent to oldest
get_qualite_rivieres_analyse(
  code_station = "02115725",
  libelle_parametre = "Nitrates",
  date_debut_prelevement = "2013-01-01",
  code_qualification = 1,
  fields = c("code_station",
             "libelle_station",
             "code_parametre",
             "libelle_parametre",
             "date_prelevement",
             "resultat",
             "symbole_unite",
             "code_remarque",
             "mnemo_remarque",
             "code_statut",
             "mnemo_statut",
             "code_qualification",
             "libelle_qualification"),
 sort = "desc"
)

# Nitrate analysis results since 1 August 2016 for 5 stations
get_qualite_rivieres_analyse(
  code_station = c("02115725",
                   "02115650",
                   "02115685",
                   "02115700",
                   "02115715"),
  libelle_parametre = "Nitrates",
  date_debut_prelevement = "2016-08-01",
  code_qualification = 1,
  fields = c("code_station",
             "libelle_station",
             "libelle_parametre",
             "date_prelevement",
             "resultat",
             "symbole_unite",
             "code_remarque"),
 sort = "desc"
)

# Geographical search by distance
get_qualite_rivieres_station(longitude = 5.62,
                             latitude = 49.44,
                             distance = 8,
                             fields = c("code_station", "libelle_station"))


## End(Not run)
```

get_temperature_stations *Retrieve data from API "Température des cours d'eau"*

Description

Available endpoints are:

- `get_temperature_stations` retrieves temperature stations
- `get_temperature_chronique` retrieves temperature time series

See the API documentation of each endpoint for available filter parameters: https://hubeau.eaufrance.fr/page/api-temperature-continu

Usage

```
get_temperature_stations(...)

get_temperature_chronique(...)
```

Arguments

`...`	parameters of the queries and their values in the format `Param1_Name = "Param1 value", Param2_Name` use the function list_params for a list of the available filter parameters for a given API endpoint and see the API documentation for their description

Value

A tibble::tibble with one row by record and one column by field.

Examples

```
## Not run:
# Retrieve the temperature stations in the department of Loiret
get_temperature_stations(code_departement = "45")

# Retrieve the temperature from 2012-01-01 to 2012-01-05 at site 04051125

get_temperature_chronique(
  code_station = "04051125",
  date_debut_mesure = "2012-01-01",
  date_fin_mesure="2012-01-05",
 fields = paste("code_station,date_mesure_temp,heure_mesure_temp,resultat,symbole_unite")
)


## End(Not run)
```

hubeau *hubeau: A package for retrieving data on the French databases on water 'Hub'Eau'*

Description

The 'hubeau' package provides functions for 'Hub'Eau' APIs and their related endpoints. These functions are named as follow: `hubeau::get_[API]_[endpoint]`.

Currently available APIs and related endpoints are listed below.

API "Ecoulement des cours d'eau":

API documentation: https://hubeau.eaufrance.fr/page/api-ecoulement

Available functions:

- `get_ecoulement_stations()`: site data and locations
- `get_ecoulement_observations()`: flow observations collected during campaigns
- `get_ecoulement_campagnes()`: observation campaign information

API "Hydrométrie":

API documentation: https://hubeau.eaufrance.fr/page/api-hydrometrie

Available functions:

- `get_hydrometrie_sites()`: hydrometry sites (can contain several stations)
- `get_hydrometrie_stations()`: hydrometry stations
- `get_hydrometrie_observations_tr()`: hydrometry water level and discharge time series
- `get_hydrometrie_obs_elab()`: hydrometric elaborate observations (daily/monthly mean flow)

API "Indicateurs des services":

API documentation: https://hubeau.eaufrance.fr/page/api-indicateurs-services

Available functions:

- `get_indicateurs_services_communes()`: performance indicators by commune
- `get_indicateurs_services_indicateurs()`: performance indicators by indicator
- `get_indicateurs_services_services()`: performance indicators by commune for each service

API "Piézométrie":

API documentation: https://hubeau.eaufrance.fr/page/api-piezometrie

Available functions:

- `get_niveaux_nappes_chroniques()`: archived time series of piezometric stations
- `get_niveaux_nappes_chroniques_tr()`: real-time time series of piezometric stations
- `get_niveaux_nappes_stations()`: piezometric stations

API "Poisson":

API documentation: https://hubeau.eaufrance.fr/page/api-poisson

Available function:

- get_poisson_operations(): sampling operations carried out at stations measuring the quality of rivers
- get_poisson_observations(): fish observations made during sampling operations
- get_poisson_indicateurs(): the IPR and IPR+ indicators calculated from fish observations
- get_poisson_stations(): stations measuring the quality of rivers

API "Prélèvements en eau":

API documentation: https://hubeau.eaufrance.fr/page/api-prelevements-eau

Available functions:

- get_prelevements_chroniques(): time series of annual withdrawn volumes by device
- get_prelevements_ouvrages(): withdrawal devices (can contain several withdrawal points)
- get_prelevements_points_prelevement(): withdrawal points

API "Qualité de l'eau potable":

API documentation: https://hubeau.eaufrance.fr/page/api-qualite-eau-potable

Available functions:

- get_qualite_eau_potable_communes_udi(): links between "UDI" (Distribution units or networks) and communes
- get_qualite_eau_potable_resultats_dis(): samples, analysis results and sanitary conclusions from the sanitary control of the distributed water commune by commune

API "Qualité des cours d'eau":

API documentation: https://hubeau.eaufrance.fr/page/api-qualite-cours-deau

Available functions:

- get_qualite_rivieres_station(): stations (measuring points) on rivers or water bodies where water samples have been taken for water quality analyses
- get_qualite_rivieres_operation(): sampling operations carried out at the stations
- get_qualite_rivieres_analyse(): physico-chemical analyses carried out on samples prepared during sampling operations at the stations
- get_qualite_rivieres_condition_environnementale(): environmental conditions (air temperature, presence of leaves, moss, iridescence, etc.) observed during physico-chemical sampling operations

API "Qualité des nappes d'eau souterraines":

API documentation: https://hubeau.eaufrance.fr/page/api-qualite-nappes

Available functions:

- get_qualite_nappes_stations(): measuring stations for groundwater quality
- get_qualite_nappes_analyses(): analysis results of quality measurement

API "Température des cours d'eau":

API Documentation https://hubeau.eaufrance.fr/page/api-temperature-continu

Available functions:

- get_temperature_stations(): temperature stations in French rivers
- get_temperature_chronique(): river temperature time series

`list_apis` *List available Hub'Eau APIs, endpoints and filter parameters*

Description

`list_apis()` returns the list of available APIs in the package.

`list_endpoints()` returns the list of available endpoints for an API.

`list_params()` returns the list of available parameters for an API endpoint.

Usage

```
list_apis()

list_endpoints(api)

list_params(api, endpoint)
```

Arguments

`api`	a character name of the API (e.g.: "indicateurs_services", "prelevements"...), see example for available APIs
`endpoint`	a character name of the endpoint, see example for available endpoints in an API

Details

The listed APIs correspond to the term `[API]` used in the name of the functions `get_[API]_[endpoint]` used for querying the APIs.

Value

character vector of APIs, endpoints or filter parameters

Examples

```
# To get the available APIs in the package
list_apis()

# To get the available endpoints in an API
list_endpoints("prelevements")

# To get available parameters in endpoint "chroniques" of the API "prelevements"
list_params(api = "prelevements", endpoint = "chroniques")
```

Index