Package: gfwr (via r-universe)

November 1, 2024

Title Access data from Global Fishing Watch APIs

Version 2.0.0

Description This package connects to several Global Fishing Watch APIs
to get vessel and events information in an R-friendly format.

License Apache License (>= 2)

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

Imports dplyr, geojsonsf, httr2, jsonlite, magrittr, purrr, readr,
rjson, rlang, tibble, tidyr, methods, class

Suggests devtools, ggplot2, glue, knitr, qpdf, rmarkdown,
rnaturalearth, rnaturalearthdata, scales, sf

URL `https://github.com/GlobalFishingWatch/gfwr`,
`https://globalfishingwatch.github.io/gfwr/`

BugReports `https://github.com/GlobalFishingWatch/gfwr/issues`

Depends R (>= 2.10)

LazyData true

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

RemoteUrl https://github.com/GlobalFishingWatch/gfwr

RemoteRef HEAD

RemoteSha a8f404153a85817c863ed1aa8f66ca86a0c9a09a

Contents

`get_event` *Base function to get events from API and convert response to data frame*

Description

Base function to get events from API and convert response to data frame

Usage

```
get_event(
  event_type,
  encounter_types = NULL,
  vessels = NULL,
  flags = NULL,
  vessel_types = NULL,
  start_date = "2012-01-01",
  end_date = "2024-12-31",
  region = NULL,
  region_source = NULL,
  gap_intentional_disabling = NULL,
  duration = 1,
  confidences = c(2, 3, 4),
  limit = 99999,
  offset = 0,
  sort = "+start",
  key = gfw_auth(),
  quiet = FALSE,
  print_request = FALSE,
  ...
)
```

Arguments

`event_type`	Type of event to get data of. A vector with any combination of "ENCOUNTER", "FISHING", "GAP", "LOITERING", "PORT_VISIT"
`encounter_types`	Filters for types of vessels during the encounter. A vector with any combination of: "CARRIER-FISHING", "FISHING-CARRIER", "FISHING-SUPPORT", "SUPPORT-FISHING"
`vessels`	A vector of vesselIds, obtained via the `get_vessel_info()` function

`flags`	ISO3 code for the flag of the vessels. Null by default.
`vessel_types`	A vector of vessel types, any combination of: "FISHING", "CARRIER", "SUPPORT", "PASSENGER", "OTHER_NON_FISHING", "SEISMIC_VESSEL", "BUNKER_OR_TANKER", "CARGO"
`start_date`	Start of date range to search events, in YYYY-MM-DD format and including this date
`end_date`	End of date range to search events, in YYYY-MM-DD format and excluding this date
`region`	sf shape to filter raster or GFW region code (such as an EEZ code). See details about formatting the geojson
`region_source`	source of the region ('EEZ','MPA', 'RFMO' or 'USER_SHAPEFILE')
`gap_intentional_disabling`	Logical. Whether the Gap events are intentional, according to Global Fishing Watch algorithms
`duration`	duration, in minutes, of the event, ex. 30
`confidences`	Confidence levels (1-4) of events (port visits only)
`limit`	Limit
`offset`	Offset
`sort`	How to sort the events. By default, +start, which sorts the events in ascending order (+) of the start dates of the events. Other possible values are -start, +end, -end.
`key`	Authorization token. Can be obtained with gfw_auth() function
`quiet`	Boolean. Whether to print the number of events returned by the request
`print_request`	Boolean. Whether to print the request, for debugging purposes. When contacting the GFW team it will be useful to send this string
`...`	Other arguments

Details

There are currently four available event types and these events are provided for three vessel types - fishing, carrier, and support vessels. Fishing events (`event_type = "FISHING"`) are specific to fishing vessels and loitering events (`event_type = "LOITERING"`) are specific to carrier vessels. Port visits (`event_type = "PORT_VISIT"`) and encounters (`event_type = "ENCOUNTER"`) are available for all vessel types. For more details about the various event types, see the GFW API documentation.

Encounter events involve multiple vessels and one row is returned for each vessel involved in an encounter. For example, an encounter between a carrier and fishing vessel (`CARRIER-FISHING`) will have one row for the fishing vessel and one for the carrier vessel. The `id` field for encounter events has two components separated by a `.`. The first component is the unique id for the encounter event and will be the same for all vessels involved in the encounter. The second component is an integer used to distinguish between different vessels in the encounter.

Examples

```
## Not run:
library(gfwr)
# port visits
get_event(event_type = "PORT_VISIT",
          vessels = c("e0c9823749264a129d6b47a7aabce377",
          "8c7304226-6c71-edbe-0b63-c246734b3c01"),
          start_date = "2017-01-26",
          end_date = "2017-12-31",
          confidence = c(3, 4), # only for port visits
          key = gfw_auth())
 #encounters
 get_event(event_type = "ENCOUNTER",
 vessels = c("e0c9823749264a129d6b47a7aabce377",
  "8c7304226-6c71-edbe-0b63-c246734b3c01"),
  start_date = "2018-01-30",
  end_date = "2023-02-04",
  key = gfw_auth())
 # fishing
 get_event(event_type = "FISHING",
 vessels = c("8c7304226-6c71-edbe-0b63-c246734b3c01"),
  start_date = "2017-01-26",
  end_date = "2023-02-04",
  key = gfw_auth())
 # GAPS
 get_event(event_type = "GAP",
 vessels = c("e0c9823749264a129d6b47a7aabce377",
  "8c7304226-6c71-edbe-0b63-c246734b3c01"),
  start_date = "2017-01-26",
  end_date = "2023-02-04",
  key = gfw_auth())
 # loitering
 get_event(event_type = "LOITERING",
 vessels = c("e0c9823749264a129d6b47a7aabce377",
  "8c7304226-6c71-edbe-0b63-c246734b3c01"),
  start_date = "2017-01-26",
  end_date = "2023-02-04",
  key = gfw_auth())
 # encounter type
 get_event(event_type = "ENCOUNTER",
 encounter_types = "CARRIER-FISHING",
 start_date = "2020-01-01",
 end_date = "2020-01-31",
 key = gfw_auth())
 # vessel types
 get_event(event_type = "ENCOUNTER",
 vessel_types = c("CARRIER", "FISHING"),
 start_date = "2020-01-01",
 end_date = "2020-01-31",
 key = gfw_auth())
# fishing events in Senegal EEZ
get_event(event_type = 'FISHING',
              start_date = "2020-10-01",
              end_date = "2020-12-31",
              region = 8371,
              region_source = 'EEZ',
              flags = 'CHN',
              key = gfw_auth())

# fishing events in user shapefile
test_polygon <- sf::st_bbox(c(xmin = -70, xmax = -40, ymin = -10, ymax = 5),
 crs = 4326) |>
 sf::st_as_sfc() |>
 sf::st_as_sf()
get_event(event_type = 'FISHING',
              start_date = "2020-10-01",
              end_date = "2020-12-31",
              region = test_polygon,
              region_source = 'USER_SHAPEFILE',
              key = gfw_auth())

## End(Not run)
```

get_event_stats — *Base function to get events stats from API and convert response to data frame*

Description

Base function to get events stats from API and convert response to data frame

Usage

```
get_event_stats(
  event_type,
  encounter_types = NULL,
  vessels = NULL,
  vessel_types = NULL,
  start_date = "2012-01-01",
  end_date = "2024-12-31",
  region_source = NULL,
  region = NULL,
  interval = NULL,
  duration = 1,
  confidences = c(2, 3, 4),
  key = gfw_auth(),
  quiet = FALSE,
  print_request = FALSE,
  ...
)
```

Arguments

`event_type`	Type of event to get data of. A vector with any combination of "ENCOUNTER", "FISHING", "GAP", "LOITERING", "PORT_VISIT"
`encounter_types`	Filters for types of vessels during the encounter. A vector with any combination of: "CARRIER-FISHING", "FISHING-CARRIER", "FISHING-SUPPORT", "SUPPORT-FISHING"
`vessels`	A vector of vesselIds, obtained via the `get_vessel_info()` function
`vessel_types`	Optional. A vector of vessel types, any combination of: "FISHING", "CARRIER", "SUPPORT", "PASSENGER", "OTHER_NON_FISHING", "SEISMIC_VESSEL", "BUNKER_OR_TANKER", "CARGO"
`start_date`	Start of date range to search events, in YYYY-MM-DD format and including this date
`end_date`	End of date range to search events, in YYYY-MM-DD format and excluding this date
`region_source`	Optional but mandatory if using the argument region. Source of the region. If 'EEZ','MPA', 'RFMO', then the value for the argument region must be the code for that region. If 'USER_SHAPEFILE', then region has to be an sf object
`region`	GFW region code (such as an EEZ, MPA or RFMO code) or a formatted geojson shape. See Details about formatting the geojson.
`interval`	Time series granularity. Must be a string. Possible values: 'HOUR', 'DAY', 'MONTH', 'YEAR'.
`duration`	duration, in minutes, of the event, ex. 30
`confidences`	Confidence levels (1-4) of events (port visits only)
`key`	Authorization token. Can be obtained with gfw_auth() function
`quiet`	Boolean. Whether to print the number of events returned by the request
`print_request`	Boolean. Whether to print the request, for debugging purposes. When contacting the GFW team it will be useful to send this string
`...`	Other arguments

Details

There are currently four available event types and these events are provided for three vessel types - fishing, carrier, and support vessels. Fishing events (`event_type = "FISHING"`) are specific to fishing vessels and loitering events (`event_type = "LOITERING"`) are specific to carrier vessels. Port visits (`event_type = "PORT_VISIT"`) and encounters (`event_type = "ENCOUNTER"`) are available for all vessel types. For more details about the various event types, see the GFW API documentation.

The user-defined geojson has to be surrounded by a geojson tag, that can be created using a simple paste:

```
geojson_tagged <- paste0('{"geojson":', your_geojson,'}').
```

If you have an **sf** shapefile, you can also use function `sf_to_geojson()` to obtain the correctly-formatted geojson.

Examples

```
## Not run:
library(gfwr)
 # stats for encounters involving Russian carriers in given time range
get_event_stats(event_type = 'ENCOUNTER',
encounter_types = c("CARRIER-FISHING","FISHING-CARRIER"),
vessel_types = 'CARRIER',
start_date = "2018-01-01",
end_date = "2023-01-31",
flags = 'RUS',
duration = 60,
interval = "YEAR",
key = gfw_auth())
 # port visits stats in a region (Senegal)
 get_event_stats(event_type = 'PORT_VISIT',
start_date = "2018-01-01",
end_date = "2019-01-31",
confidences = c('3','4'),
region = 8371,
region_source = 'EEZ',
interval = "YEAR")

## End(Not run)
```

`get_last_report` *Base function to get status of last report generated*

Description

Function to check the status of the last API request sent with get_raster().

Usage

```
get_last_report(key = gfw_auth())
```

Arguments

`key`	Authorization token. Can be obtained with `gfw_auth()` function

Details

The `get_last_report()` function will tell you if the APIs are still processing your request and will download the results if the request has finished successfully. You will receive an error message if the request finished but resulted in an error or if it's been >30 minutes since the last report was generated using `get_raster()`.

For more information, see the https://globalfishingwatch.org/our-apis/documentation#get-last-report-generated.

Examples

```
## Not run:
get_last_report(key = gfw_auth())

## End(Not run)
```

get_raster *Base function to get raster from API and convert response to data frame*

Description

Base function to get raster from API and convert response to data frame

Usage

```
get_raster(
  spatial_resolution = NULL,
  temporal_resolution = NULL,
  group_by = NULL,
  filter_by = NULL,
  start_date = NULL,
  end_date = NULL,
  region = NULL,
  region_source = NULL,
  key = gfw_auth(),
  print_request = FALSE
)
```

Arguments

`spatial_resolution`	raster spatial resolution. Can be "LOW" = 0.1 degree or "HIGH" = 0.01 degree
`temporal_resolution`	raster temporal resolution. Can be 'HOURLY', 'DAILY', 'MONTHLY', 'YEARLY'
`group_by`	parameter to group by. Can be 'VESSEL_ID', 'FLAG', 'GEARTYPE', 'FLAGANDGEARTYPE' or 'MMSI'. Optional.
`filter_by`	parameter to filter by.
`start_date`	Start of date range to search events, in YYYY-MM-DD format and including this date
`end_date`	End of date range to search events, in YYYY-MM-DD format and excluding this date
`region`	sf shape to filter raster or GFW region code (such as a Marine Regions Geographic Identifier or EEZ code).

`region_source`	source of the region ('EEZ','MPA', 'RFMO' or 'USER_SHAPEFILE')
`key`	Authorization token. Can be obtained with `gfw_auth()` function
`print_request`	Boolean. Whether to print the request, for debugging purposes. When contacting the GFW team it will be useful to send this string

Examples

```
## Not run:
library(gfwr)
# using region codes
code_eez <- get_region_id(region_name = 'CIV', region_source = 'EEZ',
key = gfw_auth())
get_raster(spatial_resolution = 'LOW',
           temporal_resolution = 'YEARLY',
           group_by = 'FLAG',
           start_date = "2021-01-01",
           end_date = "2021-10-01",
           region = code_eez$id,
           region_source = 'EEZ',
           key = gfw_auth(),
           print_request = TRUE)
#using a sf from disk /loading a test sf object
data(test_shape)
get_raster(spatial_resolution = 'LOW',
            temporal_resolution = 'YEARLY',
            start_date = '2021-01-01',
            end_date = '2021-10-01',
            region = test_shape,
            region_source = 'USER_SHAPEFILE',
            key = gfw_auth(),
            print_request = TRUE)

## End(Not run)
```

`get_regions` *List of available regions*

Description

List of available regions

Usage

```
get_regions(region_source = "EEZ", key = gfw_auth())
```

Arguments

`region_source`	string, source of region data ('eez', 'mpa', 'rfmo')
`key`	string, API token

Value

dataframe, all region ids and names for specified region type

---

get_region_id *Function to pull numeric code using region name*

Description

Function to pull numeric code using region name

Usage

```
get_region_id(region_name, region_source = "EEZ", key = gfw_auth())
```

Arguments

`region_name`	string or numeric, EEZ/MPA/RFMO name or id
`region_source`	string, source of region data ('eez', 'mpa', 'rfmo')
`key`	string, API token

Value

dataframe, eez code and EEZ name for matching EEZs

---

get_vessel_info *Base function to get vessel information from API and convert response to data frame*

Description

Base function to get vessel information from API and convert response to data frame

Usage

```
get_vessel_info(
  query = NULL,
  where = NULL,
  ids = NULL,
  includes = c("AUTHORIZATIONS", "OWNERSHIP", "MATCH_CRITERIA"),
  match_fields = NULL,
  registries_info_data = c("ALL"),
  search_type = "search",
  key = gfw_auth(),
  print_request = FALSE,
  ...
)
```

Arguments

`query`	When `search_type = "search"`, a length-1 vector with the identity variable of interest, MMSI, IMO, call sign or ship name.
`where`	When `search_type = "search"`, an SQL expression to find the vessel of interest
`ids`	When `search_type = "id"`, a vector with the vesselId of interest
`includes`	Enhances the response with new information, defaults to include all. `"OWNERSHIP"` returns ownership information `"AUTHORIZATIONS"` lists public authorizations for that vessel `"MATCH_CRITERIA"` adds information about the reason why a vessel is returned
`match_fields`	Optional. Allows to filter by `matchFields` levels. Possible values: `"SEVERAL_FIELDS"`, `"NO_MATCH"`, `"ALL"`. Incompatible with `where`
`registries_info_data`	when `search_type == "id"`, gets all the registry objects, only the delta or the latest. `"NONE"` The API will return the most recent object only `"DELTA"` The API will return only the objects when the vessel changed one or more identity properties `"ALL"` The registryInfo array will return the same number of objects that rows we have in the vessel database
`search_type`	Type of vessel search to perform. Can be `"search"` or `"id"`. (Note:`"advanced"` and `"basic"` are no longer in use as of gfwr 2.0.0.)
`key`	Authorization token. Can be obtained with `gfw_auth()` function
`print_request`	Boolean. Whether to print the request, for debugging purposes. When contacting the GFW team it will be useful to send this string
`...`	Other parameters, such as limit and offset

Details

When `search_type = "search"` the search takes basic identity features like MMSI, IMO, callsign, shipname as inputs, using parameter `"query"`. For more advanced SQL searches, use parameter `"where"`. You can combine logic operators like `AND`, `OR`, `=`, `>=` , `<`, `LIKE` (for fuzzy matching). The `id` search allows the user to search using a GFW vessel id.

Examples

```
## Not run:
library(gfwr)
# Simple searches, using includes
get_vessel_info(query = 224224000, search_type = "search",
key = gfw_auth())
# Advanced search with where instead of query:
get_vessel_info(where = "ssvid = '441618000' OR imo = '9047271'",
search_type = "search", key = gfw_auth())
 # Vessel id search
 get_vessel_info(search_type = "id",
 ids = c("8c7304226-6c71-edbe-0b63-c246734b3c01",
 "6583c51e3-3626-5638-866a-f47c3bc7ef7c"), key = gfw_auth())
 all <- get_vessel_info(search_type = "id",
 ids = c("8c7304226-6c71-edbe-0b63-c246734b3c01"),
 registries_info_data = c("ALL"), key = gfw_auth())
 none <- get_vessel_info(search_type = "id",
 ids = c("8c7304226-6c71-edbe-0b63-c246734b3c01"),
 registries_info_data = c("NONE"), key = gfw_auth())
 delta <- get_vessel_info(search_type = "id",
 ids = c("8c7304226-6c71-edbe-0b63-c246734b3c01"),
 registries_info_data = c("DELTA"), key = gfw_auth())

## End(Not run)
```

`gfw_auth` *Get user API token from .Renviron*

Description

Get user API token from .Renviron

Usage

```
gfw_auth()
```

`test_shape` *A sample shapefile*

Description

An sf shapefile to show as an example of user-defined GeoJSON in get_event() and get_raster()

Usage

```
test_shape
```

Format

A shapefile with a single polygon.

Index