Package ‘epair’

February 6, 2021

Title Grabs data from EPA API, simplifies getting pollutant data

Version 0.1.0

Description A package to aid the user in mak-
ing queries to the EPA API site found at https://aqs.epa.gov/aqsweb/documents/data_api.
It combines API calling methods from various web scraping packages with specific strings to re-
trieve data from the EPA API. It also contains
easy to use loaded variables that help a user navigate services offered by the API and aid the user in
determining the appropriate way to make a an API call.

Depends R (>= 3.3.3)

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

Imports xml2 (>= 1.1.1),
rvest (>= 0.3.5),
httr (>= 1.4.1),
jsonlite (>= 1.6.1)

Suggests testthat

BugReports `https://github.com/GLOrozcoM/epair/issues`

URL `https://github.com/GLOrozcoM/epair`

R topics documented:

`add.variable` *Add a variable to a call*

Description

Add a variable to a call

Usage

```
add.variable(query, variable, name = deparse(substitute(variable)))
```

Arguments

`query`	A URL containing authentication for EPA API
`variable`	A variable for a call. Consult VARIABLE.TYPES for possible variables.
`name`	Default argument should be left as is. Will take the name used for variable above to create the final URL.

Value

A URL containing query + variable.

Examples

```
## Not run:
endpoint <- 'dailyData/byState'
state <- "37"
call <- epair:::create.base.call(endpoint)
call <- add.variable(call, state)
call     # Call requires more variables before being placed

## End(Not run)
```

---

add.variables   *Add variables to a query*

Description

Add variables to a query

Usage

```
add.variables(query, variables, name = NA)
```

Arguments

`query`	A URL containing authentication for the EPA API site.
`variables`	A list of variables. Each variable should be declared with the appropriate name. Consult VARIABLE.TYPES for the right names.
`name`	A list containing names of API variables.

Value

A URL consisting of query + variables.

Examples

```
## Not run:
endpoint <- "dailyData/byState"
variable.list <- list("state" = '37',
                      "bdate" = '20200101',
                      "edate" = '20200102',
                      "param" = '44201')
call <- epair:::create.base.call(endpoint)
call <- add.variables(call, variable.list)
call

## End(Not run)
```

`assign.description.to.services` *Assign a description to each service*

Description

Assign a description to each service

Usage

```
assign.description.to.services(services)
```

Arguments

`services`	A list of services offered by the EPA API.

Value

The list of services with descriptions for each service.

Examples

```
## Not run:
services <- assign.description.to.services(services)
services[[1]]$Description

## End(Not run)
```

`change.classes.filter` *Update name to parameter class entry*

Description

As of this package release, a service in the API was incorrectly described in the original website. This function gives the services variable the proper information describing usage of the service.

Usage

```
change.classes.filter(services)
```

Arguments

`services`	List of services offered by the API.

Value

Services with corrected name of filter for parameter classes.

Examples

```
## Not run:
services <- get.services()
services <- change.classes.filter(services)
services$List$Filter$`Parameter Classes (groups of parameters, like criteria or all)`

## End(Not run)
```

`create.authentication` *Generate the string authentication needed for EPA API*

Description

Generate the string authentication needed for EPA API

Usage

```
create.authentication(email, key)
```

Arguments

`email`	Email registered with EPA API
`key`	Key obtained from EPA API. Register your email for a key here https://aqs.epa.gov/aqsweb/document

Value

A string with authentication info. It looks like '&email=user_email&key=user_key'.

Examples

```
auth <- create.authentication("myemail@domain.com", "myapikey")
auth
```

`create.base.call` *Make the first call when forming a query.*

Description

Make the first call when forming a query.

Usage

```
create.base.call(endpoint)
```

Arguments

`endpoint`	Endpoint for forming a query. See ENDPOINTS for all available endpoints. See SERVICES if you know the service but not the endpoint.

Value

A URL string containing authentication for the call.

Examples

```
## Not run:
endpoint <- "list/states"
call <- epair:::create.base.call(endpoint)
call

## End(Not run)
```

endpoints *Endpoints available in the EPA API*

Description

The endpoints vector contains all endpoints available in the EPA API. To get endpoints directly from the site, use get.endpoints().

find.endpoints.in.tables *Take a list of html tables from api and output all endpoints*

Description

Take a list of html tables from api and output all endpoints

Usage

```
find.endpoints.in.tables(list.tables)
```

Arguments

`list.tables`	List of HTML tables from EPA API

Value

Vector with only endpoints for the API.

Examples

```
## Not run:
API.tables <- epair:::get.all.tables()
endpoints <- epair:::find.endpoints.in.tables(API.tables)
endpoints

## End(Not run)
```

`generate.filter.content` *Generate filter content for an API filter*

Description

Generate filter content for an API filter

Usage

```
generate.filter.content(i, df)
```

Arguments

`i`	Row number at which to get the information for filter.
`df`	The data frame containing filter information.

Value

A list with filter content (endpoint, required variables, etc.)

Examples

```
## Not run:
tbls <- get.all.tables()
single <- tbls[[7]]
content <- generate.filter.content(1, single)

## End(Not run)
```

`generate.filters.list` *Create a list of filters*

Description

Create a list of filters

Usage

```
generate.filters.list(df)
```

Arguments

`df`	A data frame having filter information (e.g. name, required variables).

Value

A list containing filters and respective info.

Examples

```
## Not run:
tbls <- get.all.tables()
single <- tbls[[8]]
generate.filters.list(single)

## End(Not run)
```

`get.all.tables` *Get all the html tables in the API site*

Description

Get all the html tables in the API site

Usage

```
get.all.tables()
```

Value

A list of HTML tables from the EPA API site.

Examples

```
## Not run:
html.tables.list <- get.all.table()
html.tables.list

## End(Not run)
```

`get.endpoints` *Get all endpoints from EPA API*

Description

Get all endpoints from EPA API

Usage

```
get.endpoints()
```

Value

Vector of endpoints from the API

Examples

```
## Not run:
endpoints <- epair:::get.endpoints()
endpoints

## End(Not run)
```

get.first.entries *Get first entries for filter names*

Description

Get first entries for filter names

Usage

```
get.first.entries(df)
```

Arguments

df	Data frame with filters

Value

A vector of indices. These indices are where the first entry for a filter exists in the data frame.

Examples

```
## Not run:
tbls <- get.all.tables()
single <- tbls[[10]]
first.occurrences <- get.first.entries(single)

## End(Not run)
```

get.first.entry.for.filter *Get the first entry for a filter name*

Description

Get the first entry for a filter name

Usage

```
get.first.entry.for.filter(filter.name, df)
```

Arguments

filter.name	Name of the filter in API
df	Data frame containing filter info.

Value

The index for the first occurrence of the filter in the data frame.

Examples

```
## Not run:
tbls <- get.all.tables()
single <- tbls[[11]]
get.first.entry.for.filter("Filter Name", single)

## End(Not run)
```

`get.service.names` *Get service names and descriptions to the services*

Description

Get service names and descriptions to the services

Usage

```
get.service.names()
```

Value

A data frame containing services with names and descriptions offered by the EPA API.

Examples

```
## Not run:
service.names <- epair:::get.service.names()
service.names

## End(Not run)
```

`get.services` *Get a list of services the EPA API offers*

Description

Get a list of services the EPA API offers

Usage

```
get.services()
```

Value

List of services the EPA API offers.

Examples

```
## Not run:
services <- epair:::get.services()
services

## End(Not run)
```

get.table — *Get an HTML table at URL*

Description

Get an HTML table at URL

Usage

```
get.table(url, table.xpath)
```

Arguments

`url`	URL to get table from
`table.xpath`	The X path to the table

Value

A data frame of the HTML table

Examples

```
## Not run:
url <- "https://aqs.epa.gov/aqsweb/documents/data_api.html"
table.path <- '//*[@id="main-content"]/div[2]/div[1]/div/div/table[1]'
df <- epair:::get.table(url, table.path)
df

## End(Not run)
```

get.transpose — *Transpose a data frame*

Description

Transpose a data frame

Usage

```
get.transpose(df)
```

Arguments

`df`	Data frame to be transposed

Value

The transposed data frame. First variable entries become column names.

Examples

```
## Not run:
url <- "https://aqs.epa.gov/aqsweb/documents/data_api.html"
table.path <- '//*[@id="main-content"]/div[2]/div[1]/div/div/table[1]'
df <- epair:::get.table(url, table.path)
t.df <- epair:::get.transpose(df)
t.df

## End(Not run)
```

---

get.unique.filters *Get filter names in data frame*

Description

Get filter names in data frame

Usage

```
get.unique.filters(df)
```

Arguments

df	Data frame containing repeated or mixed filter names.

Value

Vector containing only filter names for the API service. Service name and repeated filter names are removed.

Examples

```
## Not run:
tbls <- get.all.tables()
single <- tbls[[6]]
get.unique.filters(single)

## End(Not run)
```

---

get.variables *Populate variables for info on making requests*

Description

Populate variables for info on making requests

Usage

```
get.variables()
```

Value

Data frame containing variables and information about them used in the EPA API.

Examples

```
## Not run:
vars <- epair:::get.variables()
vars$edate

## End(Not run)
```

`is.API.running` *Check if the API is up and running*

Description

Check if the API is up and running

Usage

```
is.API.running()
```

Examples

```
## Not run:
is.API.running()

## End(Not run)
```

`list.remove.escapes.spaces` *Remove tabs, new lines, and empty spaces from entries in a list*

Description

Remove tabs, new lines, and empty spaces from entries in a list

Usage

```
list.remove.escapes.spaces(a.list)
```

Arguments

`a.list`	List to remove entries from.

Value

A list without tabs, new lines, and empty spaces

Examples

```
## Not run:
services <- epair:::get.services()
services <- epair:::list.remove.escapes.spaces(services)
services

## End(Not run)
```

list.string.replacer *Replace every string entry in a list*

Description

Replace every string entry in a list

Usage

```
list.string.replacer(entry.list, pattern, replacement)
```

Arguments

entry.list	List containing character entries
pattern	Pattern to replace
replacement	Replacement for entries following the pattern

Value

A list with entries matching the pattern replaced by replacement

Examples

```
## Not run:
services <- epair:::get.services()
services <- epair:::list.string.replacer(services, "\t", "")
services

## End(Not run)
```

non.endpoint.checker *Check if a string contains characters not seen in endpoints*

Description

Check if a string contains characters not seen in endpoints

Usage

```
non.endpoint.checker(string)
```

Arguments

`string`	A character entry from entries in the data frame of API services

Value

A boolean reflecting presence of endpoint in string.

Examples

```
epair:::non.endpoint.checker("list/states")
epair:::non.endpoint.checker("https://example here")
```

`perform.call` *Perform call and convert data into list*

Description

Perform call and convert data into list

Usage

```
perform.call(endpoint, variables = list(),
  name = deparse(substitute(variables)))
```

Arguments

`endpoint`	An endpoint from the available EPA API endpoints
`variables`	A list of variables or a single variable to filter the EPA API endpoint.
`name`	Specifies the name each variable should have when placed in the URL.

Value

A list containing requested data

Examples

```
## Not run:
endpoint <- 'list/states'
result <- perform.call(endpoint)

## End(Not run)
```

`perform.call.raw` *Perform call and keep original result*

Description

Perform call and keep original result

Usage

```
perform.call.raw(endpoint, variables = list(),
  name = deparse(substitute(variables)))
```

Arguments

`endpoint`	An endpoint from the available EPA API endpoints
`variables`	A list of variables or a single variable to filter the EPA API endpoint.
`name`	Specifies the name each variable should have when placed in the URL. User input is not necessary and should be left in default state.

Value

A list containing result from query to EPA API

Examples

```
## Not run:
endpoint <- 'list/states'
result <- perform.call.raw(endpoint)

## End(Not run)
```

`place.call` *Place the URL as a call to the EPA API*

Description

Place the URL as a call to the EPA API

Usage

```
place.call(url)
```

Arguments

`url`	A string with a valid URL for the EPA API

Value

Result of query from the API

Examples

```
## Not run:
url <- "user_url"
result <- place.call(url)

## End(Not run)
```

`place.call.raw` *Perform call and maintain JSON Lite structure*

Description

Perform call and maintain JSON Lite structure

Usage

```
place.call.raw(url)
```

Arguments

`url`	URL following structure from EPA API

Value

Results of data request in JSON format

Examples

```
## Not run:
endpoint <- 'list/states'
call <- create.base.call(endpoint)
raw.call <- place.call.raw(call)
raw.call

## End(Not run)
```

`populate.all.services` *Turn tables of API services into a list*

Description

Turn tables of API services into a list

Usage

```
populate.all.services(tables.to.modify)
```

Arguments

`tables.to.modify`	List of tables from API. Each table is a data frame.

Value

A list with each service and filters as chained variables to make for easy calling.

Examples

```
## Not run:
tables.to.modify <- get.all.tables()
services <- populate.all.services(tables.to.modify)
services$List

## End(Not run)
```

`remove.escapes.spaces` *Remove tabs, new lines, and empty spaces from entries in a data frame*

Description

Remove tabs, new lines, and empty spaces from entries in a data frame

Usage

```
remove.escapes.spaces(df)
```

Arguments

`df`	Data frame to remove tabs, new lines, and empty spaces from

Value

Data frame without tabs, new lines, and empty spaces

Examples

```
## Not run:
url <- "https://aqs.epa.gov/aqsweb/documents/data_api.html"
table.path <- '//*[@id="main-content"]/div[2]/div[1]/div/div/table[1]'
df <- epair:::get.table(url, table.path)
df

clean.df <- epair:::remove.escapes.spaces(df)
clean.df

## End(Not run)
```

`service.names` *Names of services offered by the EPA API*

Description

The service.names list contains names of all services offered by the EPA API along with a description of each service.

`services` *Services offered by the EPA API*

Description

The services list contains comprehensive information about all services provided by the EPA API site.

`setup.service` *Make list of single service*

Description

Make list of single service

Usage

```
setup.service(df)
```

Arguments

`df`	Data frame with info to make an API service.

Value

A list with the filter content of a service set to the service name.

Examples

```
## Not run:
tbls <- get.all.tables()
single <- tbls[[8]]
setup.service(single)

## End(Not run)
```

`setup.single.filter` *Create a single filter*

Description

Create a single filter

Usage

```
setup.single.filter(filter.name, i, df)
```

Arguments

`filter.name`	Name of filter in API service
`i`	Row number to use to create filter. Make sure filter information is present at i before hand.
`df`	Data frame with filter information.

Value

A list with filter content given to the filter name.

Examples

```
## Not run:
tbls <- get.all.tables()
single <- tbls[[9]]
filter.name <- "My filter"
setup.single.filter(filter.name, 1, single)

## End(Not run)
```

`string.replacer` *Replace all characters entries in data frame*

Description

Replace all characters entries in data frame

Usage

```
string.replacer(df, pattern, replacement)
```

Arguments

`df`	Data frame containing character entries
`pattern`	Pattern to use for matching
`replacement`	Replacement of entries matching pattern

Value

A data frame with entries following the pattern being replaced by replacement

Examples

```
df <- data.frame(c("1", "2", "3", "4"))
modified.df <- epair:::string.replacer(df, "1", "One")
modified.df
```

`variable.types` *Variable parameter names to use*

Description

The variable.types list contains the listing endpoints for finding out more information in making calls requiring more variables.

`variables` *Variables used for querying in EPA API*

Description

The variables data frame contains information about what variables can be used to build queries in the EPA API.

Index