Package: ineapir (via r-universe)

May 18, 2026

Title Obtaining Data Published by the National Statistics Institute

Version 0.2.5

Description Get open statistical data and metadata disseminated by the National Statistics Institute of Spain (INE). The functions return data frames with the requested information thanks to calls to the 'INE' API <https://www.ine.es/dyngs/DAB/index.htm?cid=1100>.

License EUPL (== 1.2)

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Imports httr, jsonlite

Config/Needs/website rmarkdown

URL https://github.com/es-ine/ineapir, https://es-ine.github.io/ineapir/

BugReports https://github.com/es-ine/ineapir/issues

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Config/pak/sysreqs libssl-dev

Repository https://es-ine.r-universe.dev

Date/Publication 2025-09-10 07:20:22 UTC

RemoteUrl https://github.com/es-ine/ineapir

RemoteRef HEAD

RemoteSha 6676cb693857841c338c86d47d286363704af6e9

Contents

`get_data_series` *Get data from a specific series*

Description

Retrieve data from series published by INE calling the API

Usage

```
get_data_series(
  codSeries = NULL,
  nlast = 1,
  dateStart = NULL,
  dateEnd = NULL,
  det = 0,
  tip = NULL,
  lang = "ES",
  validate = TRUE,
  verbose = FALSE,
  unnest = FALSE
)
```

Arguments

codSeries	(string): Code of the series. For further information about codes click this link.
nlast	(int): number of periods to retrieve. By default is set to 1 period.
dateStart	(string): the initial date of the requested data. The required format is yyyy/mm/dd. Additionally, dateStart can be a vector of dates, where each date represents the start date of individual ranges where the end date should be found at the same position in the dateEnd vector. If dateStart and dateEnd are equal, the specified dates are retrieved. If no end date is entered, all dates will be queried, from the corresponding start date to the last available period.
dateEnd	(string): the end date of the requested data. The required format is yyyy/mm/dd. Additionally, dateEnd can be a vector of dates, where each date represents the end date of individual ranges where the initial date should be found at the same position in the dateStart vector. The length of the dateEnd vector must be less than or equal to the length of the dateStart vector.
det	(int): level of detail. Valid values: 0, 1 or 2.
tip	(string): set to 'A' for friendly output (e.g. readable dates), set to 'M' to include metadata or set to 'AM' for both.
lang	(string): language of the retrieved data. Set to 'ES' for Spanish or set to 'EN' for English.
validate	(logical): validate input parameters.
verbose	(logical): print additional information, including the URL to call the API service.
unnest	(logical): set to TRUE to obtain a single data frame of data

Value

Data frame with data of a series according to the code specified in the function

Examples

```
df <- get_data_series(codSeries = "IPC251856", nlast = 5)
head(df)

# Get data for an open range date
df <- get_data_series(codSeries = "IPC251856", dateStart = "2024/01/01")
head(df)

# Get data for a single range data
df <- get_data_series(codSeries = "IPC251856",
dateStart = "2023/01/01", dateEnd = "2023/05/01")
head(df)

# Get data for specific dates
df <- get_data_series(codSeries = "IPC251856",
dateStart = c("2023/01/01","2024/01/01"),
dateEnd = c("2023/01/01","2024/01/01"))
head(df)
```

```
# Get data for multiple date ranges
df <- get_data_series(codSeries = "IPC251856",
dateStart = c("2023/01/01","2024/01/01"),
dateEnd = c("2023/03/01","2024/03/01"))
head(df)
```

`get_data_series_filter` *Get data from series for a specific operation given a filter*

Description

Get data from series for a specific operation given a filter

Usage

```
get_data_series_filter(
  operation = NULL,
  filter = NULL,
  periodicity = NULL,
  nlast = 1,
  dateStart = NULL,
  dateEnd = NULL,
  det = 0,
  tip = NULL,
  lang = "ES",
  page = 1,
  validate = TRUE,
  verbose = FALSE,
  unnest = FALSE
)
```

Arguments

`operation` (string): Code of the operation. To obtain a list of available operations see `get_metadata_operations()`.

`filter` (list): list of variables and values.

Filtering data from series:
When we request data from series there is the possibility of filtering data on the fly using metadata information about the variables and their values that define the series. To get variables for a given operation see `get_metadata_variables()` and to get values for a specific variable see `get_metadata_values()`. See also `get_metadata_series_varval()` to get all the values at once.

Filter format:
The format is `list(id_variable1 = id_value1, id_variable2 = id_value2)`. Besides:

- A variable can take more than one value: `list(id_variable1 = c(id_value11, id_value12), id_variable2 = id_value2)`.
- A variable can take a empty character "" to get all its possible values: `list(id_variable1 = id_value1, id_variable2 = "")`.

Using shortcuts:
Additionally, shortcuts can be used to filter. They simplify the filtering approach by using standardized names for variable IDs and therefore simplify their use. The format is: `list(shortcut_variable1 = name1, shortcut_variable2 = name2)`. Besides, the *values* wrapper can also be used: `list(values = c(name1, name2))`. To see a list of all available shortcuts, see `get_filter_shortcuts()` function. Let's also remark that for better performance is recommended to use numeric codes.

`periodicity` (int): id of the periodicity of the series. Common periodicities: 1 (monthly), 3 (quarterly), 6 (bi-annual), 12 (annual). To obtain a list of periodicities see `get_metadata_periodicity()`.

`nlast` (int): number of periods to retrieve. By default is set to 1 period.

`dateStart` (string): the initial date of the requested data. The required format is yyyy/mm/dd. Additionally, dateStart can be a vector of dates, where each date represents the start date of individual ranges where the end date should be found at the same position in the dateEnd vector. If dateStart and dateEnd are equal, the specified dates are retrieved. If no end date is entered, all dates will be queried, from the corresponding start date to the last available period.

`dateEnd` (string): the end date of the requested data. The required format is yyyy/mm/dd. Additionally, dateEnd can be a vector of dates, where each date represents the end date of individual ranges where the initial date should be found at the same position in the dateStart vector. The length of the dateEnd vector must be less than or equal to the length of the dateStart vector.

`det` (int): level of detail. Valid values: 0, 1 or 2.

`tip` (string): set to 'A' for friendly output (e.g. readable dates), set to 'M' to include metadata or set to 'AM' for both.

`lang` (string): language of the retrieved data. Set to 'ES' for Spanish or set to 'EN' for English.

`page` (int): page number. The retrieved result of the query is paginated. Default value is set to 1.

`validate` (logical): validate the input parameters. A FALSE value means fewer API calls. Therefore, it is recommended to set it to FALSE when there is no doubt about the validity of the input parameters, including the filter.

`verbose` (logical): print additional information, including the URL to call the API service.

`unnest` (logical): obtain a single data frame of data

Value

Data frame with data of series according to the operation and filter specified in the function

Examples

```
# Get last five periods and filter data of time series from "IPC" operation
filter <- list("115"= "28", "3" = "84", "762" = "304092")
df <- get_data_series_filter(operation = "IPC", periodicity = 1,
nlast = 5, unnest = TRUE, validate = FALSE, filter = filter)
head(df)
```

`get_data_table` *Get data from a specific table*

Description

Get data from a specific table

Usage

```
get_data_table(
  idTable = NULL,
  filter = NULL,
  nlast = NULL,
  dateStart = NULL,
  dateEnd = NULL,
  det = 0,
  tip = NULL,
  lang = "ES",
  validate = TRUE,
  verbose = FALSE,
  unnest = FALSE,
  metanames = FALSE,
  metacodes = FALSE
)
```

Arguments

`idTable` (int): id of the table. For further information about ids click this link.

`filter` (list): list of variables and values.

Filtering data from tables:
When we request data from tables there is the possibility of filtering data on the fly using metadata information about the variables and their values that define the series. See `get_metadata_table_varval()` to get all the values at once. There are different approaches to build the filter depending on the table type.

Case one: tempus table:
URL example. For a tempus table the filter is based on ids. The format is `list(id_variable1 = id_value1, id_variable2 = id_value2)`. Besides:

- A variable can take more than one value: `list(id_variable1 = c(id_value11, id_value12), id_variable2 = id_value2)`.
- A variable can take a empty character "" to get all its possible values: `list(id_variable1 = id_value1, id_variable2 = "")`.

Case two: px tables:
URL example. For a px table the filter is based on codes. The format is `list(cod_variable1 = cod_value1, cod_variable2 = cod_value2)`. Besides:

- A variable can take more than one value: `list(cod_variable1 = c(cod_value11, cod_value12), id_variable2 = cod_value2)`.
- A variable can take a empty character "" to get all its possible values: `list(cod_variable1 = cod_value1, cod_variable2 = "")`.

Case three: tpx table:
URL example. For a tpx table the filter is based on codes. The format is `list(cod_variable1 = cod_value1, cod_variable2 = cod_value2)`. Besides:

- A variable can take more than one value: `list(cod_variable1 = c(cod_value11, cod_value12), id_variable2 = cod_value2)`.
- A variable can take a empty character "" to get all its possible values: `list(cod_variable1 = cod_value1, cod_variable2 = "")`.

URL example. There are tpx tables that contain variable ids and value ids. In this case, we can use the ids instead of the codes to build the filter. To do this we add the alias *~id* at the end of each id: `list(id_variable1~id = id_value1~id, id_variable2`

Using shortcuts:
Additionally, shortcuts can be used to filter. They simplify the filtering approach by using standardized names for variable IDs and therefore simplify their use. The format for a tempus table is: `list(shortcut_variable1 = name1, shortcut_variable2 = name2)`. However, for px and tpx tables the format is: `list(values = c(name1, name2)`. The *values* wrapper can also be used with tempus tables. To see a list of all available shortcuts, see `get_filter_shortcuts()` function. Let's also remark that for better performance is recommended to use numeric ids for tempus tables and alphanumeric codes for px and tpx tables.

`nlast` (int): number of periods to retrieve. By default it retrieves all available periods.

`dateStart` (string): the initial date of the requested data. The required format is yyyy/mm/dd. Additionally, dateStart can be a vector of dates, where each date represents the start date of individual ranges where the end date should be found at the same position in the dateEnd vector. If dateStart and dateEnd are equal, the specified dates are retrieved. If no end date is entered, all dates will be queried, from the corresponding start date to the last available period.

`dateEnd` (string): the end date of the requested data. The required format is yyyy/mm/dd. Additionally, dateEnd can be a vector of dates, where each date represents the end date of individual ranges where the initial date should be found at the same position in the dateStart vector. The length of the dateEnd vector must be less than or equal to the length of the dateStart vector.

`det` (int): level of detail. Valid values: 0, 1 or 2.

tip	(string): set to 'A' for friendly output (e.g. readable dates), set to 'M' to include metadata or set to 'AM' for both.
lang	(string): language of the retrieved data. Set to 'ES' for Spanish or set to 'EN' for English.
validate	(logical): validate the input parameters. A FALSE value means fewer API calls. Therefore, it is recommended to set it to FALSE when there is no doubt about the validity of the input parameters, including the filter.
verbose	(logical): print additional information, including the URL to call the API service.
unnest	(logical): set to TRUE to obtain a single data frame of data.
metanames	(logical): set to TRUE to extract the name of the values that defined the table. The names are extracted from the metadata information (it is mandatory to include 'M' in the tip parameter). Several columns are created corresponding to the values of the different variables.
metacodes	(logical): set to TRUE to extract the codes and ids of the values that defined the table. The codes and ids are extracted from the metadata information (it is mandatory to include 'M' in the tip parameter). Several columns are created corresponding to the values of the different variables.

Value

Data frame with data of a table according to the id specified in the function

Examples

```
# Obtaining the last two periods and filter data
filter <- list("3" = "74", "762" = "304092")
df <-get_data_table(idTable = 50902, nlast = 2, unnest = TRUE,
filter = filter, validate = FALSE)
head(df)

# Get data for an open range date
df <- get_data_table(idTable = 50902, unnest = TRUE, tip= "A",
filter = filter, validate = FALSE,
dateStart = "2025/01/01")
head(df)

# Get data for a single range data
df <- get_data_table(idTable = 50902, unnest = TRUE, tip= "A",
filter = filter, validate = FALSE,
dateStart = "2023/01/01", dateEnd = "2023/05/01")
head(df)

# Get data for specific dates
df <- get_data_table(idTable = 50902, unnest = TRUE, tip= "A",
filter = filter, validate = FALSE,
dateStart = c("2023/01/01","2024/01/01"),
dateEnd = c("2023/01/01","2024/01/01"))
head(df)
```

```
# Get data for multiple date ranges
df <- get_data_table(idTable = 50902, unnest = TRUE, tip= "A",
filter = filter, validate = FALSE,
dateStart = c("2023/01/01","2024/01/01"),
dateEnd = c("2023/03/01","2024/03/01"))
head(df)

# Get medatada as well
df <- get_data_table(idTable = 50902, nlast = 2, unnest = TRUE,
filter = filter, validate = FALSE,
metanames = TRUE, metacodes = TRUE, tip = "M")
head(df)
```

`get_filter_shortcuts` *Get all available filter shortcuts*

Description

Get all available filter shortcuts

Usage

```
get_filter_shortcuts(lang = "ES", validate = TRUE, verbose = FALSE)
```

Arguments

`lang`	(string): language. Set to 'ES' for the Spanish version of the shortcuts or set to 'EN' for the English version of the shortcuts.
`validate`	(logical): validate input parameters.
`verbose`	(logical): print additional information.

Value

Data frame with information of the available filter shortcuts

Examples

```
# Shortcuts in spanish
df <- get_filter_shortcuts()
head(df)

# Shortcuts in english
df <- get_filter_shortcuts(lang = "EN")
head(df)
```

`get_metadata_classifications` *Get all available classifications*

Description

Get all available classifications

Usage

```
get_metadata_classifications(
  operation = NULL,
  lang = "ES",
  validate = TRUE,
  verbose = FALSE
)
```

Arguments

`operation`	(string): Code of the operation. Provide code to get all the classifications for the given operation. To obtain a list of available operations see `get_metadata_operations()`. If no operation is specified then all the classifications will be shown.
`lang`	(string): language of the retrieved data. Set to 'ES' for Spanish or set to 'EN' for English.
`validate`	(logical): validate input parameters. A FALSE value means fewer API calls.
`verbose`	(logical): print additional information, including the URL to call the API service.

Value

Data frame with information of the available classifications

Examples

```
# Get all classifications
df <- get_metadata_classifications()
head(df)

# Get classifications for a specific operation
df <- get_metadata_classifications(operation = "IPC", validate = FALSE)
head(df)
```

`get_metadata_operation_table` *Get the operation for a given table*

Description

Get the operation for a given table

Usage

```
get_metadata_operation_table(
  idTable = NULL,
  lang = "ES",
  validate = TRUE,
  verbose = FALSE
)
```

Arguments

`idTable`	(int): id of the table. For further information about ids click this link.
`lang`	(string): language of the retrieved data. Set to 'ES' for Spanish or set to 'EN' for English.
`validate`	(logical): validate input parameters.
`verbose`	(logical): print additional information, including the URL to call the API service.

Value

Data frame with information of the operation according to the table specified in the function

Examples

```
# Get the operation of the table with identification code "50902"
df <- get_metadata_operation_table(idTable = 50902)
head(df)
```

get_metadata_operations *Get all available operations*

Description

Get all available operations

Usage

```
get_metadata_operations(
  operation = NULL,
  lang = "ES",
  geo = NULL,
  page = 0,
  validate = TRUE,
  verbose = FALSE
)
```

Arguments

operation	(string): code of the operation. To obtain a list of available operations see get_metadata_operations(). If no operation is specified then all the operations will be shown
lang	(string): language of the retrieved data. Set to 'ES' for Spanish or set to 'EN' for English.
geo	(int): set to 0 for operations with national data or set to 1 for operations with data with a greater level of disaggregation.
page	(int): page number. The retrieved result of the query is paginated (page=0 retrieves all pages).
validate	(logical): validate input parameters. A FALSE value means fewer API calls.
verbose	(logical): print additional information, including the URL to call the API service.

Value

Data frame with information of the available operations

Examples

```
# Get all operations
df <- get_metadata_operations()
head(df)

# Get a specific operation
df <- get_metadata_operations(operation = "IPC", validate = FALSE)
```

```
head(df)

# Get operations with territorial disaggregation
df <- get_metadata_operations(geo = 1)
head(df)
```

`get_metadata_periodicity` *Get all available periodicities*

Description

Get all available periodicities

Usage

```
get_metadata_periodicity(
  operation = NULL,
  lang = "ES",
  validate = TRUE,
  verbose = FALSE
)
```

Arguments

`operation`	(string): Code of the operation. Provide code to get all the periodicities for the given operation. To obtain a list of available operations see `get_metadata_operations()`. If no operation is specified then all the periodicities will be shown.
`lang`	(string): language of the retrieved data. Set to 'ES' for Spanish or set to 'EN' for English.
`validate`	(logical): validate input parameters. A FALSE value means fewer API calls.
`verbose`	(logical): print additional information, including the URL to call the API service.

Value

Data frame with information of the available periodicities

Examples

```
# Get all periodicities
df <- get_metadata_periodicity()
head(df)

# Get periodicities for a specific operation
df <- get_metadata_periodicity(operation = "IPC", validate = FALSE)
head(df)
```

`get_metadata_publication_dates` *Get the dates of a publication*

Description

Get the dates of a publication

Usage

```
get_metadata_publication_dates(
  publication = NULL,
  det = 0,
  tip = NULL,
  lang = "ES",
  page = 0,
  validate = TRUE,
  verbose = FALSE
)
```

Arguments

`publication`	(int): id of the publication. To obtain a list of available publications see `get_metadata_publications()`.
`det`	(int): level of detail. Valid values: 0, 1 or 2.
`tip`	(string): set to 'A' for friendly output (e.g. readable dates), set to 'M' to include metadata or set to 'AM' for both.
`lang`	(string): language of the retrieved data. Set to 'ES' for Spanish or set to 'EN' for English.
`page`	(int): page number. The retrieved result of the query is paginated (page=0 retrieves all pages).
`validate`	(logical): validate input parameters. A FALSE value means fewer API calls.
`verbose`	(logical): print additional information, including the URL to call the API service.

Value

Data frame with information of the dates of the publication specified in the function

Examples

```
# Get the dates of a publication
df <- get_metadata_publication_dates(publication = 8, validate = FALSE)
head(df)
```

get_metadata_publications *Get all publications*

Description

Get all publications

Usage

```
get_metadata_publications(
  operation = NULL,
  det = 0,
  lang = "ES",
  page = 0,
  validate = TRUE,
  verbose = FALSE
)
```

Arguments

operation	(string): code of the operation. Provide code to get all the publications for the given operation. To obtain a list of available operations see get_metadata_operations(). If no operation is specified then all the publications will be shown.
det	(int): level of detail. Valid values: 0, 1 or 2.
lang	(string): language of the retrieved data. Set to 'ES' for Spanish or set to 'EN' for English.
page	(int): page number. The retrieved result of the query is paginated (page=0 retrieves all pages).
validate	(logical): validate input parameters. A FALSE value means fewer API calls.
verbose	(logical): print additional information, including the URL to call the API service.

Value

Data frame with information about publications

Examples

```
# Get all publications
df <- get_metadata_publications()
head(df)

# Get publications for a specific operation
df <- get_metadata_publications(operation = "IPC", validate = FALSE)
head(df)
```

`get_metadata_series` *Get information for a specific series*

Description

Get information for a specific series

Usage

```
get_metadata_series(
  codSeries = NULL,
  det = 0,
  tip = NULL,
  lang = "ES",
  validate = TRUE,
  verbose = FALSE
)
```

Arguments

`codSeries`	(string): code of the series. For further information about codes click this link.
`det`	(int): level of detail. Valid values: 0, 1 or 2.
`tip`	(string): set to 'A' for friendly output (e.g. readable dates), set to 'M' to include metadata or set to 'AM' for both.
`lang`	(string): language of the retrieved data. Set to 'ES' for Spanish or set to 'EN' for English.
`validate`	(logical): validate input parameters.
`verbose`	(logical): print additional information, including the URL to call the API service.

Value

Data frame with information of a series according to the code specified in the function

Examples

```
# Get information of time series with code "IPC206449"
df <- get_metadata_series(codSeries = "IPC206449")
head(df)
```

`get_metadata_series_filter` *Get all the series for a specific operation given a filter*

Description

Get all the series for a specific operation given a filter

Usage

```
get_metadata_series_filter(
  operation = NULL,
  filter = NULL,
  periodicity = NULL,
  det = 0,
  tip = NULL,
  lang = "ES",
  page = 1,
  validate = TRUE,
  verbose = FALSE
)
```

Arguments

`operation` (string): code of the operation. To obtain a list of available operations see `get_metadata_operations()`.

`filter` (list): list of variables and values.

Filtering data from series:
When we request data from series there is the possibility of filtering data on the fly using metadata information about the variables and their values that define the series. To get variables for a given operation see `get_metadata_variables()` and to get values for a specific variable see `get_metadata_values()`. See also `get_metadata_series_varval()` to get all the values at once.

Filter format:
The format is `list(id_variable1 = id_value1, id_variable2 = id_value2)`. Besides:

- A variable can take more than one value: `list(id_variable1 = c(id_value11, id_value12), id_variable2 = id_value2)`.
- A variable can take a empty character "" to get all its possible values: `list(id_variable1 = id_value1, id_variable2 = "")`.

Using shortcuts:
Additionally, shortcuts can be used to filter. They simplify the filtering approach by using standardized names for variable IDs and therefore simplify their use. The format is: `list(shortcut_variable1 = name1, shortcut_variable2 = name2)`. Besides, the *values* wrapper can also be used: `list(values = c(name1, name2)`.

To see a list of all available shortcuts, see `get_filter_shortcuts()` function. Let's also remark that for better performance is recommended to use numeric codes.

`periodicity` (int): id of the periodicity of the series. Common periodicities: 1 (monthly), 3 (quarterly), 6 (bi-annual), 12 (annual). To obtain a list of periodicities see `get_metadata_periodicity()`.

`det` (int): level of detail. Valid values: 0, 1 or 2.

`tip` (string): set to 'A' for friendly output (e.g. readable dates), set to 'M' to include metadata or set to 'AM' for both.

`lang` (string): language of the retrieved data. Set to 'ES' for Spanish or set to 'EN' for English.

`page` (int): page number. The retrieved result of the query is paginated. Default value is set to 1.

`validate` (logical): validate input parameters. A FALSE value means fewer API calls. Therefore, it is recommended to set it to FALSE when there is no doubt about the validity of the input parameters, including the filter.

`verbose` (logical): print additional information, including the URL to call the API service.

Value

Data frame with information of the series according to the operation and filter specified in the function

Examples

```
# Get time series from "IPC" operation that include the filter variables
# and values
df <- get_metadata_series_filter(operation = "IPC", periodicity = 1,
filter = list("115"= "29", "3" = "84", "762" = ""), validate = FALSE)
head(df)
```

`get_metadata_series_operation`

Get all the series for a specific operation

Description

Get all the series for a specific operation

Usage

```
get_metadata_series_operation(
  operation = NULL,
  det = 0,
  tip = NULL,
  lang = "ES",
  page = 1,
  validate = TRUE,
  verbose = FALSE
)
```

Arguments

operation	(string): code of the operation. To obtain a list of available operations see get_metadata_operations().
det	(int): level of detail. Valid values: 0, 1 or 2.
tip	(string): set to 'A' for friendly output (e.g. readable dates), set to 'M' to include metadata or set to 'AM' for both.
lang	(string): language of the retrieved data. Set to 'ES' for Spanish or set to 'EN' for English.
page	(int): page number. The retrieved result of the query is paginated. Default value is set to 1.
validate	(logical): validate input parameters. A FALSE value means fewer API calls.
verbose	(logical): print additional information, including the URL to call the API service.

Value

Data frame with information of the series belonging to an operation.

Examples

```
# Get metadata of time series from "IPC" operation
# Retrieve page 1
df <- get_metadata_series_operation(operation = "IPC", validate = FALSE)
nrow(df)

# Retrieve page 2
df <- get_metadata_series_operation(operation = "IPC", validate = FALSE,
page = 2)
nrow(df)
```

`get_metadata_series_table` *Get all the series for a given table*

Description

Get all the series for a given table

Usage

```
get_metadata_series_table(
  idTable = NULL,
  filter = NULL,
  det = 0,
  tip = NULL,
  lang = "ES",
  validate = TRUE,
  verbose = FALSE,
  metanames = FALSE,
  metacodes = FALSE
)
```

Arguments

`idTable` (int): id of the table. For further information about ids click this link.

`filter` (list): list of variables and values.

Filtering data from tables:
When we request data from tables there is the possibility of filtering data on the fly using metadata information about the variables and their values that define the series. See `get_metadata_table_varval()` to get all the values at once. There are different approaches to build the filter depending on the table type.

Case one: tempus table:
URL example. For a tempus table the filter is based on ids. The format is `list(id_variable1 = id_value1, id_variable2 = id_value2)`. Besides:

- A variable can take more than one value: `list(id_variable1 = c(id_value11, id_value12), id_variable2 = id_value2)`.
- A variable can take a empty character "" to get all its possible values: `list(id_variable1 = id_value1, id_variable2 = "")`.

Case two: px tables:
URL example. For a px table the filter is based on codes. The format is `list(cod_variable1 = cod_value1, cod_variable2 = cod_value2)`. Besides:

- A variable can take more than one value: `list(cod_variable1 = c(cod_value11, cod_value12), id_variable2 = cod_value2)`.

- A variable can take a empty character "" to get all its possible values: `list(cod_variable1 = cod_value1, cod_variable2 = "")`.

Case three: tpx table:
URL example. For a tpx table the filter is based on codes. The format is `list(cod_variable1 = cod_value1, cod_variable2 = cod_value2)`. Besides:

- A variable can take more than one value: `list(cod_variable1 = c(cod_value11, cod_value12), id_variable2 = cod_value2)`.
- A variable can take a empty character "" to get all its possible values: `list(cod_variable1 = cod_value1, cod_variable2 = "")`.

URL example. There are tpx tables that contain variable ids and value ids. In this case, we can use the ids instead of the codes to build the filter. To do this we add the alias *~id* at the end of each id: `list(id_variable1~id = id_value1~id, id_variable2`

Using shortcuts:
Additionally, shortcuts can be used to filter. They simplify the filtering approach by using standardized names for variable IDs and therefore simplify their use. The format for a tempus table is: `list(shortcut_variable1 = name1, shortcut_variable2 = name2)`. However, for px and tpx tables the format is: `list(values = c(name1, name2)`. The *values* wrapper can also be used with tempus tables. To see a list of all available shortcuts, see `get_filter_shortcuts()` function. Let's also remark that for better performance is recommended to use numeric ids for tempus tables and alphanumeric codes for px and tpx tables.

`det` (int): level of detail. Valid values: 0, 1 or 2.

`tip` (string): set to 'A' for friendly output (e.g. readable dates), set to 'M' to include metadata or set to 'AM' for both.

`lang` (string): language of the retrieved data. Set to 'ES' for Spanish or set to 'EN' for English.

`validate` (logical): validate input parameters. A FALSE value means fewer API calls. Therefore, it is recommended to set it to FALSE when there is no doubt about the validity of the input parameters, including the filter.

`verbose` (logical): print additional information, including the URL to call the API service.

`metanames` (logical): set to TRUE to extract the name of the values that defined the table. The names are extracted from the metadata information (it is mandatory to include 'M' in the tip parameter). Several columns are created corresponding to the values of the different variables.

`metacodes` (logical): set to TRUE to extract the codes and ids of the values that defined the table. The codes and ids are extracted from the metadata information (it is mandatory to include 'M' in the tip parameter). Several columns are created corresponding to the values of the different variables.

Value

Data frame with information of the series for a given table.

Examples

```
# Get time series without data from table with identification code "50902"
filter <- list("3" = "83")
df <- get_metadata_series_table(idTable = 50902, validate = FALSE,
filter = filter)
head(df)

# Get metadata as well
df <- get_metadata_series_table(idTable = 50902, validate = FALSE,
filter = filter, metanames = TRUE, metacodes = TRUE, tip = "M")
head(df)
```

`get_metadata_series_values`

Get all the values for a given series

Description

Get all the values for a given series

Usage

```
get_metadata_series_values(
  codSeries = NULL,
  det = 0,
  tip = NULL,
  lang = "ES",
  validate = TRUE,
  verbose = FALSE
)
```

Arguments

`codSeries`	(string): code of the series. For further information about codes click this link.
`det`	(int): level of detail. Valid values: 0, 1 or 2.
`tip`	(string): set to 'A' for friendly output (e.g. readable dates), set to 'M' to include metadata or set to 'AM' for both.
`lang`	(string): language of the retrieved data. Set to 'ES' for Spanish or set to 'EN' for English.
`validate`	(logical): validate input parameters.
`verbose`	(logical): print additional information, including the URL to call the API service.

Value

Data frame with information of the values of a series according to the code specified in the function

Examples

```
# Get metadata of time series with code "IPC206449"
df <- get_metadata_series_values(codSeries = "IPC206449")
head(df)
```

`get_metadata_series_varval` *Get metadata information about the variables and values of series for a given operation*

Description

Get metadata information about the variables and values of series for a given operation

Usage

```
get_metadata_series_varval(
  operation = NULL,
  lang = "ES",
  det = 0,
  validate = TRUE,
  verbose = FALSE
)
```

Arguments

`operation`	(string): code of the operation. To obtain a list of available operations see `get_metadata_operations()`.
`lang`	(string): language of the retrieved data. Set to 'ES' for Spanish or set to 'EN' for English.
`det`	(int): level of detail. Valid values: 0, 1 or 2.
`validate`	(logical): validate input parameters. A FALSE value means fewer API calls.
`verbose`	(logical): print additional information, including the URL to call the API service.

Value

Data frame with information about the variables and values that define the series according to the operation specified in the function

Examples

```
# Get metadata information of time series from "IPC" operation
df <- get_metadata_series_varval(operation = "IPC", validate = FALSE)
head(df)
```

get_metadata_table_groups *Get all groups for a specific a table*

Description

Get all groups for a specific a table

Usage

```
get_metadata_table_groups(
  idTable = NULL,
  lang = "ES",
  validate = TRUE,
  verbose = FALSE
)
```

Arguments

idTable	(int): id of the table. For further information about ids click this link.
lang	(string): language of the retrieved data. Set to 'ES' for Spanish or set to 'EN' for English.
validate	(logical): validate input parameters.
verbose	(logical): print additional information, including the URL to call the API service.

Value

Data frame with information of the groups according to the table specified in the function

Examples

```
# Get the groups of the table with identification code "50902"
df <- get_metadata_table_groups(idTable = 50902)
head(df)
```

get_metadata_table_values *Get all values for a specific table group*

Description

Get all values for a specific table group

Usage

```
get_metadata_table_values(
  idTable = NULL,
  idGroup = NULL,
  det = 0,
  lang = "ES",
  validate = TRUE,
  verbose = FALSE
)
```

Arguments

idTable	(int): id of the table. For further information about ids click this link.
idGroup	(int): id of the group of variables. To get all groups for a specific table see get_metadata_table_groups().
det	(int): level of detail. Valid values: 0, 1 or 2.
lang	(string): language of the retrieved data. Set to 'ES' for Spanish or set to 'EN' for English.
validate	(logical): validate input parameters. A FALSE value means fewer API calls.
verbose	(logical): print additional information, including the URL to call the API service.

Value

Data frame with information of the values of a table group according to the table and group specified in the function

Examples

```
# Get the values of the group "110889" of the table with identification
# code "50902"
df <- get_metadata_table_values(idTable = 50902, idGroup = 110889, validate = FALSE)
head(df)
```

`get_metadata_table_varval` *Get metadata information about the variables and values for a given table*

Description

Get metadata information about the variables and values for a given table

Usage

```
get_metadata_table_varval(
  idTable = NULL,
  det = 0,
  filter = NULL,
  lang = "ES",
  validate = TRUE,
  verbose = FALSE
)
```

Arguments

`idTable`	(int): id of the table. For further information about ids click this link.
`det`	(int): level of detail. Valid values: 0, 1 or 2.
`filter`	(list): list of variables and values.

Filtering data from tables:
When we request data from tables there is the possibility of filtering data on the fly using metadata information about the variables and their values that define the series. There are different approaches to build the filter depending on the table type.

Case one: tempus table:
URL example. For a tempus table the filter is based on ids. The format is `list(id_variable1 = id_value1, id_variable2 = id_value2)`. Besides:

- A variable can take more than one value: `list(id_variable1 = c(id_value11, id_value12), id_variable2 = id_value2)`.
- A variable can take a empty character "" to get all its possible values: `list(id_variable1 = id_value1, id_variable2 = "")`.

Case two: px tables:
URL example. For a px table the filter is based on codes. The format is `list(cod_variable1 = cod_value1, cod_variable2 = cod_value2)`. Besides:

- A variable can take more than one value: `list(cod_variable1 = c(cod_value11, cod_value12), id_variable2 = cod_value2)`.

- A variable can take a empty character "" to get all its possible values: `list(cod_variable1 = cod_value1, cod_variable2 = "")`.

Case three: tpx table:
URL example. For a tpx table the filter is based on codes. The format is `list(cod_variable1 = cod_value1, cod_variable2 = cod_value2)`. Besides:

- A variable can take more than one value: `list(cod_variable1 = c(cod_value11, cod_value12), id_variable2 = cod_value2)`.
- A variable can take a empty character "" to get all its possible values: `list(cod_variable1 = cod_value1, cod_variable2 = "")`.

URL example. There are tpx tables that contain variable ids and value ids. In this case, we can use the ids instead of the codes to build the filter. To do this we add the alias *~id* at the end of each id: `list(id_variable1~id = id_value1~id, id_variable2`

`lang` (string): language of the retrieved data. Set to 'ES' for Spanish or set to 'EN' for English.

`validate` (logical): validate input parameters.A FALSE value means fewer API calls. Therefore, it is recommended to set it to FALSE when there is no doubt about the validity of the input parameters, including the filter.

`verbose` (logical): print additional information, including the URL to call the API service.

Value

Data frame with information about the variables and values that define a table according to the table specified in the function

Examples

```
# Get all the variable and values of the table with identification code "50902"
df <- get_metadata_table_varval(idTable = 50902)
head(df)

# Filter variables and values
df <- get_metadata_table_varval(idTable = 52056,
filter = list(NAC = "00"), validate = FALSE)
head(df)
```

`get_metadata_tables_operation`

Get all tables for a given operation

Description

Get all tables for a given operation

Usage

```
get_metadata_tables_operation(
  operation = NULL,
  det = 0,
  tip = NULL,
  geo = NULL,
  lang = "ES",
  page = 0,
  validate = TRUE,
  verbose = FALSE
)
```

Arguments

operation	(string): code of the operation. To obtain a list of available operations see get_metadata_operations().
det	(int): level of detail. Valid values: 0, 1 or 2.
tip	(string): set to 'A' for friendly output (e.g. readable dates), set to 'M' to include metadata or set to 'AM' for both.
geo	(int): set to 0 for national tables or set to 1 for tables with a greater level of disaggregation.
lang	(string): language of the retrieved data. Set to 'ES' for Spanish or set to 'EN' for English
page	(int): page number. The retrieved result of the query is paginated (page=0 retrieves all pages).
validate	(logical): validate input parameters. A FALSE value means fewer API calls.
verbose	(logical): print additional information, including the URL to call the API service.

Value

Data frame with information of the available tables according to the operation specified in the function

Examples

```
# Get all the tables of the "IPC" operation
df <- get_metadata_tables_operation(operation = "IPC", validate = FALSE)
head(df)
```

`get_metadata_values` *Get all values for a specific variable*

Description

Get all values for a specific variable

Usage

```
get_metadata_values(
  operation = NULL,
  variable = NULL,
  value = NULL,
  det = 0,
  lang = "ES",
  page = 0,
  classification = NULL,
  validate = TRUE,
  verbose = FALSE,
  hierarchy = NULL,
  filter = NULL
)
```

Arguments

`operation`	(string): code of the operation. Provide code to get all the values for the given operation. To obtain a list of available operations see `get_metadata_operations()`.
`variable`	(int): id of a variable. To obtain a list of available variables see `get_metadata_variables()`.
`value`	(int): id of a value. If an id value is specified, the children of the value are requested. To obtain a list of available values for a variable use `get_metadata_values(variable = id_variable)`.
`det`	(int): level of detail. Valid values: 0, 1 or 2.
`lang`	(string): language of the retrieved data. Set to 'ES' for Spanish or set to 'EN' for English.
`page`	(int): page number. The retrieved result of the query is paginated (page=0 retrieves all pages).
`classification`	(int): id of a classification. To obtain a list of available classifications see `get_metadata_classifications()`.
`validate`	(logical): validate input parameters. A FALSE value means fewer API calls.
`verbose`	(logical): print additional information, including the URL to call the API service.
`hierarchy`	(int): depth of the hierarchy tree.

`filter` (list): list of variables and values. When we request the hierarchy tree there is the possibility of filtering using metadata information about the variables and their values that define the series. The format is `list(id_variable1 = id_value1, id_variable2 = id_value2)`. Besides:

- A variable can take more than one value: `list(id_variable1 = c(id_value11, id_value12), id_variable2 = id_value2)`.
- A variable can take a empty character "" to get all its possible values: `list(id_variable1 = id_value1, id_variable2 = "")`.

Value

Data frame with information of the available values for the variable specified in the function

Examples

```
# Get the values of the variable "115"
df <- get_metadata_values(variable = 115)
head(df)

# Get the values of a variable for a specific operation
df <- get_metadata_values(operation = "IPC", variable = 115, validate = FALSE)
head(df)

# Get the children of a value (provinces of Galicia)
# Variable: Autonomous communities (id=70)
# Value: Galicia (id=9008)
df <- get_metadata_values(variable = 70, value = 9008)
head(df)
```

---

get_metadata_variables

Get all available variables

Description

Get all available variables

Usage

```
get_metadata_variables(
  operation = NULL,
  lang = "ES",
  det = 0,
  page = 0,
  validate = TRUE,
  verbose = FALSE
)
```

Arguments

operation	(string): Code of the operation. Provide code to get all the variables for the given operation. To obtain a list of available operations see get_metadata_operations(). If no operation is specified then all the variables will be shown.
lang	(string): language of the retrieved data. Set to 'ES' for Spanish or set to 'EN' for English.
det	(int): level of detail. Valid values: 0, 1 or 2.
page	(int): page number. The retrieved result of the query is paginated (page=0 retrieves all pages).
validate	(logical): validate input parameters. A FALSE value means fewer API calls.
verbose	(logical): print additional information, including the URL to call the API service.

Value

Data frame with information of the available variables

Examples

```
# Gel all variables
df <- get_metadata_variables()
head(df)

# Get variables for a specific operation
df <- get_metadata_variables(operation = "IPC", validate = FALSE)
head(df)
```

Index