Package 'osmextract'

February 15, 2021

Type Package

Title Download and Read OpenStreetMap Data Extracts

Version 0.2.1

Description Find, download, convert and read Open Street Map data extracts obtained from several providers.

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

URL https://docs.ropensci.org/osmextract/ (website)
https://github.com/ropensci/osmextract

BugReports https://github.com/ropensci/osmextract/issues

Depends R (>= 3.5.0)

Imports sf (>= 0.8.1), utils, tools

Suggests testthat, knitr, rmarkdown, covr

VignetteBuilder knitr

Language en_GB

NeedsCompilation no

Author Andrea Gilardi [aut, cre] (<https://orcid.org/0000-0002-9424-7439>), Robin Lovelace [aut] (<https://orcid.org/0000-0001-5679-6536>), Barry Rowlingson [ctb] (<https://orcid.org/0000-0002-8586-6625>), Salva Fernández [rev] (Salva reviewed the package (v. 0.1) for rOpenSci, see <https://github.com/ropensci/software-review/issues/395>), Nicholas Potter [rev] (Nicholas reviewed the package (v. 0.1) for rOpenSci, see <https://github.com/ropensci/software-review/issues/395>)

Maintainer Andrea Gilardi <andrea.gilardi@unimib.it>

Repository CRAN

Date/Publication 2021-02-15 14:10:05 UTC

R topics documented:

`bbbike_zones` *An sf object of geographical zones taken from bbbike.org*

Description

Start bicycle routing for... everywhere!

Usage

```
bbbike_zones
```

Format

An `sf` object with 235 rows and 11 columns:

name The, usually English, long-form name of the city.

last_modified When was it last modified?

type empty

pbf_file_size Size of the pbf file in bytes.

base_url The base URL for the city.

poly_url The `.poly` file location.

pbf Link to the latest `.osm.pbf` file for this region.

level An integer code always equal to 3 (since the bbbike data represent non-hierarchical geographical zones). This is used only for matching operations in case of spatial input. The oe_* functions will select the geographical area closest to the input place with the highest "level". See geofabrik_zones for an example of a hierarchical structure.

geometry The `sfg` for that geographical region, rectangular.

Details

An sf object containing the URLs, names and file_size of the OSM extracts.

Source

https://download.bbbike.org/osm/

See Also

Other provider's-database: geofabrik_zones, openstreetmap_fr_zones

---

geofabrik_zones *An sf object of geographical zones taken from Geofabrik*

Description

An sf object containing the URLs, names and file-sizes of the OSM extracts stored at https://download.geofabrik.de/. You can read more details about these data at the following link: https://download.geofabrik.de/technical.html.

Usage

```
geofabrik_zones
```

Format

An sf object with 430 rows and 15 columns:

id A unique identifier. It contains letters, numbers and potentially the characters "-" and "/".

name The, usually English, long-form name of the area.

parent The identifier of the next larger excerpts that contains this one, if present.

level An integer code between 1 and 4. If level = 1 then the zone corresponds to one of the continents plus the Russian Federation: Africa, Antarctica, Asia, Australia and Oceania, Central America, Europe, North America, Russian Federation and South America. If level = 2 then the zone corresponds to the continent's subregions (i.e. the countries, such as Italy, Great Britain, Spain, USA, Mexico, Belize, Morocco, Peru and so on). There are also some exceptions that correspond to the Special Sub Regions (according to their Geofabrik definition), which are: South Africa (includes Lesotho), Alps, Britain and Ireland, Germany + Austria + Switzerland, US Midwest, US Northeast, US Pacific, US South, US West and all US states. Level = 3L correspond to the subregions of each state (or each level 2 zone). For example the West Yorkshire, which is a subregion of England, is a level 3 zone. Finally, level = 4L correspond to the subregions of the third level and it is mainly related to some small areas in Germany. This field is used only for matching operations in case of spatial input.

iso3166-1_alpha2 A character vector of two-letter ISO3166-1 codes. This will be set on the smallest extract that still fully (or mostly) contains the entity with that code; e.g. the code "DE" will be given for the Germany extract and not for Europe even though Europe contains Germany. If an extract covers several countries and no per-country extracts are available (e.g. Israel and Palestine), then several ISO codes will be given (such as "PS IL" for "Palestine and Israel").

iso3166_2 A character vector of usually five-character ISO3166-2 codes. The same rules as above apply. Some entities have both an *iso3166-1* and *iso3166-2* code. For example, the *iso3166_2* code of each US State is "US - " plus the code of the state.

pbf Link to the latest `.osm.pbf` file for this region.

bz2 Link to the latest `.osm.bz2` file for this region.

shp Link to the latest shape file for this region.

pbf.internal Link to the latest `.osm.pbf` file with user data for this region (requires OSM login).

history Link to the latest history file for this region (requires OSM login).

taginfo Link to the Geofabrik taginfo instance for this region.

updates Link to the updates directory (append /state.txt for status file).

geometry The sfc for that geographical region. These are not the country boundaries but a buffer around countries.

pbf_file_size Size of the `.pbf` file in bytes.

Source

https://download.geofabrik.de/

See Also

Other provider's-database: `bbbike_zones`, `openstreetmap_fr_zones`

`oe_download` *Download a file given a url*

Description

This function is used to download a file given a URL. It focuses on OSM extracts with `.osm.pbf` format stored by one of the providers implemented in the package. The URL is specified through the parameter `file_url`.

Usage

```
oe_download(
  file_url,
  provider = NULL,
  file_basename = basename(file_url),
  download_directory = oe_download_directory(),
  file_size = NA,
```

```
  force_download = FALSE,
  max_file_size = 5e+08,
  quiet = FALSE
)
```

Arguments

file_url	A URL pointing to a .osm.pbf file that should be downloaded.
provider	Which provider stores the file? If NULL (the default), it may be inferred from the URL, but it must be specified for non-standard cases. See details and examples.
file_basename	The basename of the file. The default behaviour is to auto-generate it from the URL using basename().
download_directory	Where to download the file containing the OSM data? By default this is equal to oe_download_directory(), which is equal to tempdir() and it changes each time you restart R. You can set a persistent download_directory by adding the following to your .Renviron file (e.g. with edit_r_environ function in usethis package): OSMEXT_DOWNLOAD_DIRECTORY=/path/to/osm/data.
file_size	How big is the file? Optional. NA by default. If it's bigger than max_file_size and the function is run in interactive mode, then an interactive menu is displayed, asking for permission for downloading the file.
force_download	Should the .osm.pbf file be updated if it has already been downloaded? FALSE by default. This parameter is used to update old .osm.pbf files.
max_file_size	The maximum file size to download without asking in interactive mode. Default: 5e+8, half a gigabyte.
quiet	Boolean. If FALSE, the function prints informative messages. Starting from sf version 0.9.6, if quiet is equal to FALSE, then vectortranslate operations will display a progress bar.

Details

This function runs several checks before actually downloading a new file to avoid overloading the OSM providers. The first step is the definition of the file's path associated to the input file_url. The path is created by pasting together the download_directory, the name of chosen provider (which may be inferred from the URL) and the basename() of the URL. For example, if file_url is equal to "https://download.geofabrik.de/europe/italy-latest.osm.pbf", and download_directory = "/tmp", then the path is built as "/tmp/geofabrik_italy-latest.osm.pbf". Thereafter, the function checks the existence of that file and, if it founds it, then it returns the path. The parameter force_download is used to modify this behaviour. If there is no file associated with the new path, then the function downloads a new file using download.file() with mode = "wb", and, again, it returns the path.

Value

A character string representing the file's path.

Examples

```
its_match = oe_match("ITS Leeds", provider = "test", quiet = TRUE)
# ITS Leeds data are stored on github, which is not a standard provider.
# So we need to specify the provider parameter.
oe_download(
  file_url = its_match$url,
  file_size = its_match$file_size,
  provider = "test"
)
## Not run:
iow_details = oe_match("Isle of Wight")
oe_download(
  file_url = iow_details$url,
  file_size = iow_details$file_size
)
Sucre_details = oe_match("Sucre", provider = "bbbike")
oe_download(
  file_url = Sucre_details$url,
  file_size = Sucre_details$file_size,
  download_directory = tempdir()
)
## End(Not run)
```

---

oe_download_directory *Return the download directory used by the package*

Description

By default, the download directory is equal to `tempdir()`. You can set a persistent download directory by adding the following command to your `.Renviron` file (e.g. with `edit_r_environ` function in `usethis` package: OSMEXT_DOWNLOAD_DIRECTORY=/path/to/osm/data.

Usage

```
oe_download_directory()
```

Value

A character vector representing the path for the download directory used by the package.

Examples

```
oe_download_directory()
```

oe_find *Get the location of files*

Description

This function takes a `place` name and it returns the path of `.pbf` and `.gpkg` files associated with it.

Usage

```
oe_find(
  place,
  provider = "geofabrik",
  download_directory = oe_download_directory(),
  download_if_missing = FALSE,
  ...
)
```

Arguments

`place`	Description of the geographical area that should be matched with a `.osm.pbf` file through the chosen `provider`. Can be either a length-1 character vector, an `sf` or `sfc` object, or a numeric vector of coordinates with length 2. In the last case, it is assumed that the EPSG code is 4326 specified as c(LON, LAT), while you can use any CRS with an `sf` or `sfc` object. See Details and examples in `oe_match()`.
`provider`	Which provider should be used to download the data? Available providers can be found with the following command: `oe_providers()`. For `oe_get()` and `oe_match()`, if `place` is equal to ITS Leeds, then `provider` is set equal to `test`. This is just for simple examples and internal tests.
`download_directory`	Directory where the files downloaded by osmextract are stored. By default it is equal to `oe_download_directory()`.
`download_if_missing`	Attempt to download the file if it cannot be found? `FALSE` by default.
`...`	Extra arguments that are passed to `oe_match()` and `oe_get()`. Please note that you cannot modify the argument `download_only`.

Details

The matching between the existing files (saved in a directory specified by `download_directory` parameter) and the input `place` is performed using `list.files`, setting a pattern equal to the basename of the URL associated to the input `place`. For example, if you specify `place = "Isle of Wight"`, then the input `place` is matched with a URL of a `.osm.pbf` file (via `oe_match()`) and the matching is performed setting a pattern equal to the basename of that URL.

If there is no file in download_directory that can be matched with the basename and download_if_missing parameter is equal to TRUE, then the function tries to download and translate a new file from the chosen provider (geofabrik is the default provider). If download_if_missing parameter is equal to FALSE (default value), then the function stops with an error.

Value

A character vector of length one (or two) representing the path(s) of the corresponding .pbf (and .gpkg) files.

Examples

```
res = oe_get("ITS Leeds", provider = "test")
oe_find("ITS Leeds", provider = "test")
## Not run:
oe_find("Isle of Wight")
oe_find("Isle of Wight", download_if_missing = TRUE)
oe_find("Leeds", provider = "bbbike", download_if_missing = TRUE)
## End(Not run)
```

---

oe_get *Find, download, translate and read OSM extracts from several providers*

Description

This function is used to find, download, translate and read OSM extracts obtained from several providers. It is a wrapper around oe_match() and oe_read(). Check the introductory vignette, the examples and the help pages of the wrapped functions to understand the details behind all parameters.

Usage

```
oe_get(
  place,
  layer = "lines",
  ...,
  provider = "geofabrik",
  match_by = "name",
  max_string_dist = 1,
  level = NULL,
  download_directory = oe_download_directory(),
  force_download = FALSE,
  max_file_size = 5e+08,
  vectortranslate_options = NULL,
  osmconf_ini = NULL,
  extra_tags = NULL,
  force_vectortranslate = FALSE,
```

```
  download_only = FALSE,
  skip_vectortranslate = FALSE,
  never_skip_vectortranslate = FALSE,
  quiet = FALSE
)
```

Arguments

place	Description of the geographical area that should be matched with a .osm.pbf file through the chosen provider. Can be either a length-1 character vector, an sf or sfc object, or a numeric vector of coordinates with length 2. In the last case, it is assumed that the EPSG code is 4326 specified as c(LON, LAT), while you can use any CRS with an sf or sfc object. See Details and examples in oe_match().
layer	Which layer should be read in? Typically points, lines (the default), multilinestrings, multipolygons or other_relations. If you specify an ad-hoc query using the argument query (see introductory vignette and examples), then oe_get() and oe_read() will read the layer specified in the query and ignore layer. See also #122.
...	Arguments that will be passed to sf::st_read(), like query, wkt_filter or stringsAsFactors. Check the introductory vignette to understand how to create your own (SQL-like) queries.
provider	Which provider should be used to download the data? Available providers can be found with the following command: oe_providers(). For oe_get() and oe_match(), if place is equal to ITS Leeds, then provider is set equal to test. This is just for simple examples and internal tests.
match_by	Which column of the provider's database should be used for matching the input place with a .osm.pbf file? The default is "name". Check Details and Examples in oe_match() to understand how this parameter works. Ignored if place is not a character vector since the matching is performed through a spatial operation.
max_string_dist	Numerical value greater or equal than 0. What is the maximum distance in fuzzy matching (i.e. Approximate String Distance, see adist()) between input place and match_by column to tolerate before testing alternative providers or looking for geographical matching with Nominatim API? This parameter is set equal to 0 if match_by is equal to iso3166_1_alpha2 or iso3166_2. Check Details and Examples in oe_match() to understand why this parameter is important. Ignored if place is not a character vector since the matching is performed through a spatial operation.
level	An integer representing the desired hierarchical level in case of spatial matching. For the geofabrik provider, for example, 1 corresponds with continent-level datasets, 2 for countries, 3 corresponds to regions and 4 to subregions. Hence, we could approximately say that smaller administrative units correspond to bigger levels. If NULL, the default, the oe_* functions will select the highest available level. See Details and Examples in oe_match().

`download_directory`
: Where to download the file containing the OSM data? By default this is equal to `oe_download_directory()`, which is equal to `tempdir()` and it changes each time you restart R. You can set a persistent `download_directory` by adding the following to your `.Renviron` file (e.g. with `edit_r_environ` function in `usethis` package): OSMEXT_DOWNLOAD_DIRECTORY=/path/to/osm/data.

`force_download`
: Should the `.osm.pbf` file be updated if it has already been downloaded? `FALSE` by default. This parameter is used to update old `.osm.pbf` files.

`max_file_size`
: The maximum file size to download without asking in interactive mode. Default: `5e+8`, half a gigabyte.

`vectortranslate_options`
: Options passed to the `sf::gdal_utils()` argument `options`. Set by default. Check details in the introductory vignette and the help page of `oe_vectortranslate()`.

`osmconf_ini`
: The configuration file. See documentation at gdal.org. Check details in the introductory vignette and the help page of `oe_vectortranslate()`. Set by default.

`extra_tags`
: Which additional columns, corresponding to OSM tags, should be in the resulting dataset? `NULL` by default. Check the introductory vignette and the help pages of `oe_vectortranslate()` and `oe_get_keys()`. Ignored when `osmconf_ini` is not `NULL`.

`force_vectortranslate`
: Boolean. Force the original `.pbf` file to be translated into a `.gpkg` file, even if a `.gpkg` with the same name already exists? `FALSE` by default. If tags in `extra_tags` match data in previously translated `.gpkg` files no translation occurs (see #173 for details). Check the introductory vignette and the help page of `oe_vectortranslate()`.

`download_only`
: Boolean. If `TRUE`, then the function only returns the path where the matched file is stored, instead of reading it. `FALSE` by default.

`skip_vectortranslate`
: Boolean. If `TRUE`, then the function skips all vectortranslate operations and it reads (or simply returns the path) of the `.osm.pbf` file. `FALSE` by default.

`never_skip_vectortranslate`
: Boolean. This is used in case the user passed its own `.ini` file or vectortranslate options (since, in those case, it's too difficult to determine if an existing `.gpkg` file was generated following the same options.)

`quiet`
: Boolean. If `FALSE`, the function prints informative messages. Starting from `sf` version 0.9.6, if `quiet` is equal to `FALSE`, then vectortranslate operations will display a progress bar.

Details

The algorithm that we use for importing an OSM extract data into R is divided into 4 steps: 1) match the input `place` with the url of a `.pbf` file; 2) download the `.pbf` file; 3) convert it into `.gpkg` format and 4) read-in the `.gpkg` file. The function `oe_match()` is used to perform the first operation and the function `oe_read()` (which is a wrapper around `oe_download()`, `oe_vectortranslate()` and `sf::st_read()`) performs the other three operations.

Value

An sf object.

See Also

oe_match(), oe_download(), oe_vectortranslate(), and oe_read().

Examples

```
# Download OSM extracts associated to a simple test.
its = oe_get("ITS Leeds", quiet = FALSE)
class(its)
unique(sf::st_geometry_type(its))

# Get another layer from the test extract
its_points = oe_get("ITS Leeds", layer = "points")
unique(sf::st_geometry_type(its_points))

# Get the .osm.pbf and .gpkg file path
oe_get("ITS Leeds", download_only = TRUE)
oe_get("ITS Leeds", download_only = TRUE, skip_vectortranslate = TRUE)
# See also ?oe_find()

# Add additional tags
its_with_oneway = oe_get("ITS Leeds", extra_tags = "oneway", quiet = FALSE)
names(its_with_oneway)
table(its_with_oneway$oneway, useNA = "ifany")

# Use the query argument to get only oneway streets:
q = "SELECT * FROM 'lines' WHERE oneway IN ('yes')"
its_oneway = oe_get("ITS Leeds", query = q)
its_oneway

## Not run:
# A more complex example
west_yorkshire = oe_get("West Yorkshire", quiet = FALSE)
# If you run it again, the function will not download the file
# or convert it again
west_yorkshire = oe_get("West Yorkshire", quiet = FALSE)
# Match with place name
oe_get("Milan") # Warning: the .pbf file is 400MB
oe_get("Vatican City")
oe_get("Zurich")

# Match with coordinates (any EPSG)
milan_duomo = sf::st_sfc(sf::st_point(c(1514924, 5034552)), crs = 3003)
oe_get(milan_duomo, quiet = FALSE) # Warning: the .pbf file is 400MB
# Match with numeric coordinates (EPSG = 4326)
oe_match(c(9.1916, 45.4650), quiet = FALSE)
# Alternative providers
baku = oe_get(place = "Baku", provider = "bbbike", quiet = FALSE)
```

```
# Other examples:
oe_get("RU", match_by = "iso3166_1_alpha2", quiet = FALSE)
# The following example mimics read_sf
oe_get("Andora", stringsAsFactors = FALSE, quiet = TRUE, as_tibble = TRUE)
## End(Not run)
```

`oe_get_keys` *Return all keys stored in "other_tags" column*

Description

This function is used to return the names of all keys that are stored in "other_tags" column since they were not explicitly included in the file. See Details.

Usage

```
oe_get_keys(zone, layer = "lines")

## Default S3 method:
oe_get_keys(zone, layer = "lines")

## S3 method for class 'character'
oe_get_keys(zone, layer = "lines")

## S3 method for class 'sf'
oe_get_keys(zone, layer = "lines")
```

Arguments

`zone`	An `sf` object (typically read-in with `oe_read()` or `oe_get()`) with an `other_tags` field, or a character vector (of length 1) that points to a `.gpkg` file (typically created using `oe_vectortranslate()` or `oe_get()`).
`layer`	Which `layer` should be read in? Typically `points`, `lines` (the default), `multilinestrings`, `multipolygons` or `other_relations`. If you specify an ad-hoc query using the argument `query` (see introductory vignette and examples), then `oe_get()` and `oe_read()` will read the layer specified in the query and ignore `layer`. See also #122.

Details

OSM data are typically documented using several `tags`. A `tag` is a pair of two items, namely a `key` and a `value`. As we documented in `oe_vectortranslate()`, the conversion between `.osm.pbf` and `.gpkg` formats is governed by a CONFIG file that indicates which tags are explicitly added to the `.gpkg` file. All the other keys stored in the `.osm.pbf` file are automatically appended using an "other_tags" field, with a syntax compatible with the PostgreSQL HSTORE type. This function is used to display the names of all keys stored in the "other_tags" column.

You can also use the `hstore_get_value()` function from GDAL to extract one particular tag from an existing `.gpkg` file. Check the introductory vignette and see examples.

The definition of a generic S3 implementation started in osmextract/issues/138.

Value

A character vector indicating the name of all keys stored in "other_tags" field.

See Also

`oe_vectortranslate()` and osmextract/issues/107.

Examples

```
itsleeds_gpkg_path = oe_get("ITS Leeds", download_only = TRUE)
oe_get_keys(itsleeds_gpkg_path)

itsleeds = oe_get("ITS Leeds")
oe_get_keys(itsleeds)

# Add an extra key to an existing .gpkg file without vectortranslate
names(oe_read(
  itsleeds_gpkg_path,
  query = "SELECT *,  hstore_get_value(other_tags, 'oneway')  AS oneway FROM lines"
))
```

---

oe_match *Match input place with a url*

Description

This function is used to match an input `place` with the URL of a `.osm.pbf` file (and its file-size, if present). The URLs are stored in several provider's databases. See `oe_providers()` and examples.

Usage

```
oe_match(place, ...)

## Default S3 method:
oe_match(place, ...)

## S3 method for class 'sf'
oe_match(place, ...)

## S3 method for class 'sfc'
oe_match(place, provider = "geofabrik", level = NULL, quiet = FALSE, ...)

## S3 method for class 'numeric'
```

```
oe_match(place, provider = "geofabrik", quiet = FALSE, ...)

## S3 method for class 'character'
oe_match(
  place,
  provider = "geofabrik",
  quiet = FALSE,
  match_by = "name",
  max_string_dist = 1,
  ...
)
```

Arguments

place	Description of the geographical area that should be matched with a .osm.pbf file through the chosen provider. Can be either a length-1 character vector, an sf or sfc object, or a numeric vector of coordinates with length 2. In the last case, it is assumed that the EPSG code is 4326 specified as c(LON, LAT), while you can use any CRS with an sf or sfc object. See Details and examples in oe_match().
...	arguments passed to other methods
provider	Which provider should be used to download the data? Available providers can be found with the following command: oe_providers(). For oe_get() and oe_match(), if place is equal to ITS Leeds, then provider is set equal to test. This is just for simple examples and internal tests.
level	An integer representing the desired hierarchical level in case of spatial matching. For the geofabrik provider, for example, 1 corresponds with continent-level datasets, 2 for countries, 3 corresponds to regions and 4 to subregions. Hence, we could approximately say that smaller administrative units correspond to bigger levels. If NULL, the default, the oe_* functions will select the highest available level. See Details and Examples in oe_match().
quiet	Boolean. If FALSE, the function prints informative messages. Starting from sf version 0.9.6, if quiet is equal to FALSE, then vectortranslate operations will display a progress bar.
match_by	Which column of the provider's database should be used for matching the input place with a .osm.pbf file? The default is "name". Check Details and Examples in oe_match() to understand how this parameter works. Ignored if place is not a character vector since the matching is performed through a spatial operation.
max_string_dist	Numerical value greater or equal than 0. What is the maximum distance in fuzzy matching (i.e. Approximate String Distance, see adist()) between input place and match_by column to tolerate before testing alternative providers or looking for geographical matching with Nominatim API? This parameter is set equal to 0 if match_by is equal to iso3166_1_alpha2 or iso3166_2. Check Details and Examples in oe_match() to understand why this parameter is important. Ignored if place is not a character vector since the matching is performed through a spatial operation.

Details

If the input place is specified as a spatial object (either `sf` or `sfc`), then the function will return a geographical area that completely contains the object (or an error). The argument `level` (which must be specified as an integer between 1 and 4, extreme values included) is used to select between multiple geographically nested areas. We could roughly say that smaller administrative units correspond to higher levels. Check the help page of the chosen provider for more details on `level` field. By default, `level = NULL`, which means that `oe_match()` will return the area corresponding to the highest available level. If there is no geographical area at the desired level, then the function will return an error. If there are multiple areas at the same `level` intersecting the input place, then the function will return the area whose centroid is closest to the input place.

If the input place is specified as a character vector and there are multiple plausible matches between the input place and the `match_by` column, then the function will return a warning and it will select the first match. See Examples. On the other hand, if the approximate string distance between the input `place` and the best match in `match_by` column is greater than `max_string_dist`, then the function will look for exact matches (i.e. `max_string_dist = 0`) in the other supported providers. If it finds an exact match, then it will return the corresponding URL. Otherwise, if `match_by` is equal to `"name"`, then it will try to geolocate the input `place` using the Nominatim API, and then it will perform a spatial matching operation (see Examples and introductory vignette), while, if `match_by != "name"`, then it will return an error.

The fields `iso3166_1_alpha2` and `iso3166_2` are used by Geofabrik provider to perform matching operations using ISO 3166-1 alpha-2 and ISO 3166-2 codes. See geofabrik_zones for more details.

Value

A list with two elements, named `url` and `file_size`. The first element is the URL of the `.osm.pbf` file associated with the input `place`, while the second element is the size of the file in bytes (which may be `NULL` or `NA`)

See Also

`oe_providers()` and `oe_match_pattern()`.

Examples

```
# The simplest example:
oe_match("Italy")

# The default provider is "geofabrik", but we can change that:
oe_match("Leeds", provider = "bbbike")

# By default, the matching operations are performed through the column
# "name" in the provider's database but this can be a problem. Hence,
# you can perform the matching operations using other columns:
oe_match("RU", match_by = "iso3166_1_alpha2")
# Run oe_providers() for reading a short description of all providers and
# check the help pages of the corresponding databases to learn which fields
# are present.

# You can always increase the max_string_dist argument, but it can be
```

```
# dangerous:
oe_match("London", max_string_dist = 3, quiet = FALSE)

# Match the input zone using an sfc object:
milan_duomo = sf::st_sfc(sf::st_point(c(1514924, 5034552)), crs = 3003)
oe_match(milan_duomo, quiet = FALSE)
leeds = sf::st_sfc(sf::st_point(c(430147.8, 433551.5)), crs = 27700)
oe_match(leeds, provider = "bbbike")

# If you specify more than one sfg object, then oe_match will select the OSM
# extract that covers all areas
milan_leeds = sf::st_sfc(
  sf::st_point(c(9.190544, 45.46416)), # Milan
  sf::st_point(c(-1.543789, 53.7974)), # Leeds
  crs = 4326
)
oe_match(milan_leeds)

# Match the input zone using a numeric vector of coordinates
# (in which case crs = 4326 is assumed)
oe_match(c(9.1916, 45.4650)) # Milan, Duomo using CRS = 4326

# The following returns a warning since Berin is matched both
# with Benin and Berlin
oe_match("Berin", quiet = FALSE)

# If the input place does not match any zone in the chosen provider, then the
# function will test the other providers:
oe_match("Leeds")

# If the input place cannot be exactly matched with any zone in any provider,
# then the function will try to geolocate the input and then it will perform a
# spatial match:
oe_match("Milan")

# The level parameter can be used to select smaller or bigger geographical
# areas during spatial matching
yak = c(-120.51084, 46.60156)
## Not run:
oe_match(yak, level = 3) # error
## End(Not run)
oe_match(yak, level = 2) # by default, level is equal to the maximum value
oe_match(yak, level = 1)
```

`oe_match_pattern` *Check patterns in the provider's databases*

Description

This function is used to explore the provider's databases and look for patterns. This function can be useful in combination with `oe_match()` and `oe_get()` for an easy match. See Examples.

Usage

```
oe_match_pattern(
  pattern,
  provider = "geofabrik",
  match_by = "name",
  full_row = FALSE
)
```

Arguments

pattern	Character string representing the pattern that should be explored.
provider	Which provider should be used? Check a summary of all available providers with oe_providers().
match_by	Column name of the provider's database that will be used to find the match.
full_row	Boolean. Return all columns for the matching rows? FALSE by default.

Value

A character vector or a subset of the provider's database.

Examples

```
## Not run:
oe_match("Yorkshire", quiet = FALSE)
## End(Not run)
oe_match_pattern("Yorkshire")

res = oe_match_pattern("Yorkshire", full_row = TRUE)
sf::st_drop_geometry(res)[1:3]
```

---

oe_providers *Summary of available providers*

Description

This function is used to display a short summary of the major characteristics of the databases associated to all available providers.

Usage

```
oe_providers(quiet = FALSE)
```

Arguments

quiet	Boolean. If FALSE, the function prints informative messages. Starting from sf version 0.9.6, if quiet is equal to FALSE, then vectortranslate operations will display a progress bar.

Value

A `data.frame` with 4 columns representing the name of each available provider, the name of the corresponding database and the number of features and fields.

Examples

```
oe_providers()
```

`oe_read` *Read a .pbf or .gpkg object from file or url*

Description

This function is used to read a `.pbf` or `.gpkg` object from file or URL. It is a wrapper around `oe_download()`, `oe_vectortranslate()`, and `sf::st_read()`, creating an easy way to download, convert, and read a `.pbf` or `.gpkg` file. Check the introductory vignette and the help pages of the wrapped function for more details.

Usage

```
oe_read(
  file_path,
  layer = "lines",
  ...,
  provider = NULL,
  download_directory = oe_download_directory(),
  file_size = NULL,
  force_download = FALSE,
  max_file_size = 5e+08,
  download_only = FALSE,
  skip_vectortranslate = FALSE,
  vectortranslate_options = NULL,
  osmconf_ini = NULL,
  extra_tags = NULL,
  force_vectortranslate = FALSE,
  never_skip_vectortranslate = FALSE,
  quiet = FALSE
)
```

Arguments

`file_path`	A URL or the path of a `.pbf` or `.gpkg` file. If a URL, then it must be specified using HTTP/HTTPS protocol.
`layer`	Which `layer` should be read in? Typically `points`, `lines` (the default), `multilinestrings`, `multipolygons` or `other_relations`. If you specify an ad-hoc query using the argument `query` (see introductory vignette and examples), then `oe_get()` and `oe_read()` will read the layer specified in the query and ignore `layer`. See also #122.

`...` Arguments that will be passed to `sf::st_read()`, like `query`, `wkt_filter` or `stringsAsFactors`. Check the introductory vignette to understand how to create your own (SQL-like) queries.

`provider` Which provider should be used to download the data? Available providers can be found with the following command: `oe_providers()`. For `oe_get()` and `oe_match()`, if `place` is equal to ITS Leeds, then `provider` is set equal to `test`. This is just for simple examples and internal tests.

`download_directory` Where to download the file containing the OSM data? By default this is equal to `oe_download_directory()`, which is equal to `tempdir()` and it changes each time you restart R. You can set a persistent `download_directory` by adding the following to your `.Renviron` file (e.g. with `edit_r_environ` function in `usethis` package): OSMEXT_DOWNLOAD_DIRECTORY=/path/to/osm/data.

`file_size` How big is the file? Optional. `NA` by default. If it's bigger than `max_file_size` and the function is run in interactive mode, then an interactive menu is displayed, asking for permission to download the file.

`force_download` Should the `.osm.pbf` file be updated if it has already been downloaded? `FALSE` by default. This parameter is used to update old `.osm.pbf` files.

`max_file_size` The maximum file size to download without asking in interactive mode. Default: `5e+8`, half a gigabyte.

`download_only` Boolean. If `TRUE`, then the function only returns the path where the matched file is stored, instead of reading it. `FALSE` by default.

`skip_vectortranslate` Boolean. If `TRUE`, then the function skips all vectortranslate operations and it reads (or simply returns the path) of the `.osm.pbf` file. `FALSE` by default.

`vectortranslate_options` Options passed to the `sf::gdal_utils()` argument `options`. Set by default. Check details in the introductory vignette and the help page of `oe_vectortranslate()`.

`osmconf_ini` The configuration file. See documentation at gdal.org. Check details in the introductory vignette and the help page of `oe_vectortranslate()`. Set by default.

`extra_tags` Which additional columns, corresponding to OSM tags, should be in the resulting dataset? `NULL` by default. Check the introductory vignette and the help pages of `oe_vectortranslate()` and `oe_get_keys()`. Ignored when `osmconf_ini` is not `NULL`.

`force_vectortranslate` Boolean. Force the original `.pbf` file to be translated into a `.gpkg` file, even if a `.gpkg` with the same name already exists? `FALSE` by default. If tags in `extra_tags` match data in previously translated `.gpkg` files no translation occurs (see #173 for details). Check the introductory vignette and the help page of `oe_vectortranslate()`.

`never_skip_vectortranslate` Boolean. This is used in case the user passed its own `.ini` file or vectortranslate options (since, in those case, it's too difficult to determine if an existing `.gpkg` file was generated following the same options.)

`quiet` Boolean. If `FALSE`, the function prints informative messages. Starting from `sf` version 0.9.6, if `quiet` is equal to `FALSE`, then vectortranslate operations will display a progress bar.

Details

The arguments `provider`, `download_directory`, `file_size`, `force_download`, and `max_file_size` are ignored if `file_path` points to an existing `.pbf` or `.gpkg` file.

You cannot add any field or layer to an existing `.gpkg` file (unless you have the `.pbf` file and you convert it again with a different configuration), but you can extract some of the tags in `other_tags` field. Check examples and `oe_get_keys()` for more details.

Value

An `sf` object.

Examples

```
# Read an existing .pbf file. First we need to copy a .pbf file into a
# temporary directory
file.copy(
  from = system.file("its-example.osm.pbf", package = "osmextract"),
  to = file.path(tempdir(), "its-example.osm.pbf")
)
my_pbf = file.path(tempdir(), "its-example.osm.pbf")
oe_read(my_pbf)
oe_read(my_pbf, layer = "points") # Read a new layer
# The following example shows how to add new tags
names(oe_read(my_pbf, extra_tags = c("oneway", "ref"), quiet = FALSE))

# Read an existing .gpkg file. This file was created by oe_read
my_gpkg = file.path(tempdir(), "its-example.gpkg")
oe_read(my_gpkg)
# You cannot add any layer to an existing .gpkg file but you can extract some
# of the tags in other_tags. Check oe_get_keys() for more details.
names(oe_read(my_gpkg, extra_tags = c("maxspeed"))) # doesn't work
# Instead, use the query argument
names(oe_read(
  my_gpkg,
  quiet = TRUE,
  query =
  "SELECT *,
  hstore_get_value(other_tags, 'maxspeed') AS maxspeed
  FROM lines
  "
))

# Read from a URL
my_url = "https://github.com/ropensci/osmextract/raw/master/inst/its-example.osm.pbf"
# Please note that if you read from a URL which is not linked to one of the
# supported providers, you need to specify the provider parameter:
## Not run:
oe_read(my_url, provider = "test", quiet = FALSE)

## End(Not run)
```

oe_search *Search for a place and return an sf data frame locating it*

Description

This (at the moment internal) function provides a simple interface to the nominatim service for finding the geographical location of place names.

Usage

```
oe_search(
  place,
  base_url = "https://nominatim.openstreetmap.org",
  destfile = tempfile(fileext = ".geojson"),
  ...
)
```

Arguments

`place`	Text string containing the name of a place the location of which is to be found, such as `"Leeds"` or `"Milan"`.
`base_url`	The URL of the nominatim server to use. The main open server hosted by OpenStreetMap is the default.
`destfile`	The name of the destination file where the output of the search query, a `.geojson` file, should be saved.
`...`	Extra arguments that are passed to `sf::st_read`.

Value

An `sf` object corresponding to the input place. The `sf` object is read by `sf::st_read()` and it is based on a `geojson` file returned by Nominatim API.

oe_update *Update all the .osm.pbf files saved in a directory*

Description

This function is used to re-download all `.osm.pbf` files stored in `download_directory` that were firstly downloaded through `oe_get()`. See Details.

Usage

```
oe_update(
  download_directory = oe_download_directory(),
  quiet = FALSE,
  delete_gpkg = TRUE,
  max_file_size = 5e+08,
  ...
)
```

Arguments

download_directory	Character string of the path of the directory where the .osm.pbf files are saved.
quiet	Boolean. If FALSE the function prints informative messages. See Details.
delete_gpkg	Boolean. if TRUE the function deletes the old .gpkg files. We added this parameter to minimize the probability of accidentally reading-in old and not-synchronized .gpkg files. See Details. Defaults to TRUE.
max_file_size	The maximum file size to download without asking in interactive mode. Default: 5e+8, half a gigabyte.
...	Additional parameter that will be passed to oe_get() (such as stringsAsFactors or query).

Details

This function is used to re-download .osm.pbf files that are stored in a directory (specified by download_directory param) and that were firstly downloaded through oe_get() . The name of the files must begin with the name of one of the supported providers (see oe_providers()) and it must end with .osm.pbf. All other files in the directory that do not match this format are ignored.

The process for re-downloading the .osm.pbf files is performed using the function oe_get() . The appropriate provider is determined by looking at the first word in the path of the .osm.pbf file. The place is determined by looking at the second word in the file path and the matching is performed through the id column in the provider's database. So, for example, the path geofabrik_italy-latest-update.osm.pbf will be matched with the provider "geofabrik" and the geographical zone italy through the column id in geofabrik_zones.

The parameter delete_gpkg is used to delete all .gpkg files in download_directory. We decided to set its default value to TRUE to minimize the possibility of reading-in old and non-synchronized .gpkg files. If you set delete_gpkg = FALSE, then you need to manually reconvert all files using oe_get() or oe_vectortranslate() .

If you set the parameter quiet to FALSE, then the function will print some useful messages regarding the characteristics of the files before and after updating them. More precisely, it will print the output of the columns size, mtime and ctime from file.info(). Please note that the meaning of mtime and ctime depends on the OS and the file system. Check file.info().

Value

The path(s) of the .osm.pbf file(s) that were updated.

Examples

```
## Not run:
# Set up a fake directory with .pbf and .gpkg files
fake_dir = tempdir()
# Fill the directory
oe_get("andorra", download_directory = fake_dir, download_only = TRUE)
# Check the directory
list.files(fake_dir, pattern = "gpkg|pbf")
# Update all .pbf files and delete .gpkg files
oe_update(fake_dir)
list.files(fake_dir, pattern = "gpkg|pbf")
## End(Not run)
```

`oe_vectortranslate` *Translate a .osm.pbf file into .gpkg format*

Description

This function is used to translate a `.osm.pbf` file into `.gpkg` format. The conversion is performed using ogr2ogr through `vectortranslate` utility in `sf::gdal_utils()` . It was created following the suggestions of the maintainers of GDAL. See Details and examples to understand the basic usage, and check the introductory vignette for more complex use-cases.

Usage

```
oe_vectortranslate(
  file_path,
  layer = "lines",
  vectortranslate_options = NULL,
  osmconf_ini = NULL,
  extra_tags = NULL,
  force_vectortranslate = FALSE,
  never_skip_vectortranslate = FALSE,
  quiet = FALSE
)
```

Arguments

`file_path`	Character string representing the path of the input `.pbf` or `.osm.pbf` file.
`layer`	Which `layer` should be read in? Typically `points`, `lines` (the default), `multilinestrings`, `multipolygons` or `other_relations`. If you specify an ad-hoc query using the argument `query` (see introductory vignette and examples), then `oe_get()` and `oe_read()` will read the layer specified in the query and ignore `layer`. See also #122.
`vectortranslate_options`	Options passed to the `sf::gdal_utils()` argument `options`. Set by default. Check details in the introductory vignette and the help page of `oe_vectortranslate()`.

`osmconf_ini` The configuration file. See documentation at gdal.org. Check details in the introductory vignette and the help page of `oe_vectortranslate()`. Set by default.

`extra_tags` Which additional columns, corresponding to OSM tags, should be in the resulting dataset? `NULL` by default. Check the introductory vignette and the help pages of `oe_vectortranslate()` and `oe_get_keys()`. Ignored when `osmconf_ini` is not `NULL`.

`force_vectortranslate`
Boolean. Force the original `.pbf` file to be translated into a `.gpkg` file, even if a `.gpkg` with the same name already exists? `FALSE` by default. If tags in `extra_tags` match data in previously translated `.gpkg` files no translation occurs (see #173 for details). Check the introductory vignette and the help page of `oe_vectortranslate()`.

`never_skip_vectortranslate`
Boolean. This is used in case the user passed its own `.ini` file or vectortranslate options (since, in those case, it's too difficult to determine if an existing `.gpkg` file was generated following the same options.)

`quiet` Boolean. If `FALSE`, the function prints informative messages. Starting from `sf` version 0.9.6, if `quiet` is equal to `FALSE`, then vectortranslate operations will display a progress bar.

Details

The new `.gpkg` file is created in the same directory as the input `.osm.pbf` file. The translation process is performed using the `vectortranslate` utility in `sf::gdal_utils()`. This operation can be customized in several ways modifying the parameters `layer`, `extra_tags`, `osmconf_ini`, and `vectortranslate_options`.

The `.osm.pbf` files processed by GDAL are usually categorized into 5 layers, named `points`, `lines`, `multilinestrings`, `multipolygons` and `other_relations`. Check the first paragraphs here for more details. This function can covert only one layer at a time, and the parameter `layer` is used to specify which layer of the `.osm.pbf` file should be converted. Several layers with different names can be stored in the same `.gpkg` file. By default, the function will convert the `lines` layer (which is the most common one according to our experience).

The arguments `osmconf_ini` and `extra_tags` are used to modify how GDAL reads and processes a `.osm.pbf` file. More precisely, several operations that GDAL performs on the input `.osm.pbf` file are governed by a `CONFIG` file, that you can check at the following link. The basic components of OSM data are called *elements* and they are divided into *nodes*, *ways* or *relations*, so, for example, the code at line 7 of that link is used to determine which *ways* are assumed to be polygons (according to the simple-feature definition of polygon) if they are closed. Moreover, OSM data is usually described using several *tags*, i.e a pair of two items: a key and a value. The code at lines 33, 53, 85, 103, and 121 is used to determine, for each layer, which tags should be explicitly reported as fields (while all the other tags are stored in the `other_tags` column, see `oe_get_keys()`). The parameter `extra_tags` is used to determine which extra tags (i.e. key/value pairs) should be added to the `.gpkg` file.

By default, the vectortranslate operations are skipped if the function detects a file having the same path as the input file, `.gpkg` extension and a layer with the same name as the parameter `layer` with all `extra_tags`. In that case the function will simply return the path of the `.gpkg` file. This behaviour can be overwritten by setting `force_vectortranslate = TRUE`. The parameter

osmconf_ini is used to pass your own CONFIG file in case you need more control over the GDAL operations. In that case the vectortranslate operations are never skipped. Check the package introductory vignette for an example. If osmconf_ini is equal to NULL (the default), then the function uses default osmconf.ini file defined by GDAL (but for the extra tags).

The parameter vectortranslate_options is used to control the arguments that are passed to ogr2ogr via sf::gdal_utils() when converting between .pbf and .gpkg formats. ogr2ogr can perform various operations during the conversion process, such as spatial filters or SQL queries. These operations are determined by the vectortranslate_options argument. If NULL (default value), then vectortranslate_options is set equal to

```
c("-f","GPKG","-overwrite","-oo",paste0("CONFIG_FILE=",osmconf_ini),"-lco","GEOMETRY_NAME=geometry"
```

Explanation:

- "-f", "GPKG" says that the output format is GPKG;
- "-overwrite is used to delete an existing layer and recreate it empty;
- "-oo", paste0("CONFIG_FILE=", osmconf_ini) is used to set the Open Options for the .osm.pbf file and change the CONFIG file (in case the user asks for any extra tag or a totally different CONFIG file);
- "-lco", "GEOMETRY_NAME=geometry" is used to change the layer creation options for the .gpkg file and modify the name of the geometry column;
- layer indicates which layer should be converted.

Check the introductory vignette, the help page of sf::gdal_utils() and here for an extensive documentation on all available options.

Value

Character string representing the path of the .gpkg file.

See Also

oe_get_keys()

Examples

```
# First we need to match an input zone with a .osm.pbf file
its_match = oe_match("ITS Leeds", provider = "test")
# The we can download the .osm.pbf files
its_pbf = oe_download(
  file_url = its_match$url,
  file_size = its_match$file_size,
  download_directory = tempdir(),
  provider = "test"
)
# Check that the file was downloaded
list.files(tempdir(), pattern = "pbf|gpkg")
# Convert to gpkg format
its_gpkg = oe_vectortranslate(its_pbf)
# Now there is an extra .gpkg file
list.files(tempdir(), pattern = "pbf|gpkg")
```

```
# Check the layers of the .gpkg file
sf::st_layers(its_gpkg, do_count = TRUE)
# Add points layer
its_gpkg = oe_vectortranslate(its_pbf, layer = "points")
sf::st_layers(its_gpkg, do_count = TRUE)

# Add extra tags to the lines layer. Check original fields
names(sf::st_read(its_gpkg, layer = "lines", quiet = TRUE))
its_gpkg = oe_vectortranslate(
  its_pbf,
  extra_tags = c("oneway", "maxspeed")
)
names(sf::st_read(its_gpkg, layer = "lines", quiet = TRUE))
# Check the introductory vignette for more complex examples.
```

`openstreetmap_fr_zones` *An sf object of geographical zones taken from download.openstreetmap.fr*

Description

An `sf` object containing the URLs, names and file-sizes of the OSM extracts stored at http://download.openstreetmap.fr/.

Usage

```
openstreetmap_fr_zones
```

Format

An `sf` object with 835 rows and 7 columns:

id A unique ID for each area. It is used by `oe_update()`.

name The, usually English, long-form name of the city.

parent The identifier of the next larger excerpts that contains this one, if present.

level An integer code between 1 and 4. Check http://download.openstreetmap.fr/polygons/ to see the hierarchical structure of the zones. 1L correspond to the biggest areas. This is used only for matching operations in case of spatial input.

pbf Link to the latest `.osm.pbf` file for this region.

pbf_file_size Size of the pbf file in bytes.

geometry The `sfg` for that geographical region, rectangular.

Source

https://download.bbbike.org/osm/

See Also

Other provider's-database: bbbike_zones, geofabrik_zones

`test_zones` *An sf object of geographical zones taken from download.openstreetmap.fr*

Description

This object represent a minimal provider's database and it should be used only for examples and tests.

Usage

```
test_zones
```

Format

An object of class `sf` (inherits from `data.frame`) with 2 rows and 7 columns.

Index