Package 'osmdata'

March 22, 2021

Title Import 'OpenStreetMap' Data as Simple Features or Spatial Objects

Version 0.1.5

Maintainer Mark Padgham <mark.padgham@email.com>

Description Download and import of 'OpenStreetMap' ('OSM') data as 'sf' or 'sp' objects. 'OSM' data are extracted from the 'Overpass' web server (<https://overpass-api.de/>) and processed with very fast 'C++' routines for return to 'R'.

License GPL-3

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

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

Depends R (>= 3.2.4)

Imports curl, httr, jsonlite, lubridate, magrittr, methods, Rcpp (>= 0.12.4), rvest, sp, tibble, utils, xml2

Suggests knitr, raster, rmarkdown, sf, testthat

LinkingTo Rcpp

VignetteBuilder knitr

Encoding UTF-8

LazyData true

NeedsCompilation yes

RoxygenNote 7.1.1

SystemRequirements C++11

X-schema.org-applicationCategory Data Access

X-schema.org-isPartOf https://ropensci.org

X-schema.org-keywords open0street0map, openstreetmap, overpass0API, OSM

Author Mark Padgham [aut, cre],
Bob Rudis [aut],
Robin Lovelace [aut],
Maëlle Salmon [aut],
Andrew Smith [ctb],
James Smith [ctb],
Andrea Gilardi [ctb],
Enrico Spinielli [ctb],
Marcin Kalicinski [ctb, cph] (Author of included RapidXML code),
Finkelstein Noam [ctb, cph] (Author of included stub.R code),
Bartnik Lukasz [ctb, cph] (Author of included stub.R code)

Repository CRAN

Date/Publication 2021-03-22 12:10:02 UTC

R topics documented:

add_osm_feature *Add a feature to an Overpass query*

Description

Add a feature to an Overpass query

Usage

```
add_osm_feature(
  opq,
  key,
  value,
  key_exact = TRUE,
  value_exact = TRUE,
  match_case = TRUE,
  bbox = NULL
)
```

Arguments

opq	An overpass_query object
key	feature key
value	value for feature key; can be negated with an initial exclamation mark, value = "!this", and can also be a vector, value = c ("this","that").
key_exact	If FALSE, key is not interpreted exactly; see https://wiki.openstreetmap.org/wiki/Overpass_API
value_exact	If FALSE, value is not interpreted exactly
match_case	If FALSE, matching for both key and value is not sensitive to case
bbox	optional bounding box for the feature query; must be set if no opq query bbox has been set

Value

opq object

Note

key_exact should generally be TRUE, because OSM uses a reasonably well defined set of possible keys, as returned by available_features. Setting key_exact = FALSE allows matching of regular expressions on OSM keys, as described in Section 6.1.5 of https://wiki.openstreetmap.org/wiki/Overpass_API/Overpass_QL. The actual query submitted to the overpass API can be obtained from opq_string.

References

https://wiki.openstreetmap.org/wiki/Map_Features

Examples

```
## Not run:
q <- opq ("portsmouth usa") %>%
                add_osm_feature(key = "amenity",
                                value = "restaurant") %>%
                add_osm_feature(key = "amenity", value = "pub")
osmdata_sf (q) # all objects that are restaurants AND pubs (there are none!)
q1 <- opq ("portsmouth usa") %>%
                add_osm_feature(key = "amenity",
                                value = "restaurant")
q2 <- opq ("portsmouth usa") %>%
                add_osm_feature(key = "amenity", value = "pub")
c (osmdata_sf (q1), osmdata_sf (q2)) # all restaurants OR pubs
# Use of negation to extract all non-primary highways
q <- opq ("portsmouth uk") %>%
        add_osm_feature (key = "highway", value = "!primary")

## End(Not run)
```

---

available_features *List recognized features in OSM*

Description

List recognized features in OSM

Usage

```
available_features()
```

Value

character vector of all known features

Note

requires internet access

References

https://wiki.openstreetmap.org/wiki/Map_Features

Examples

```
## Not run:
available_features()

## End(Not run)
```

`available_tags` *List tags associated with a feature*

Description

List tags associated with a feature

Usage

```
available_tags(feature)
```

Arguments

`feature`	feature to retrieve

Value

character vector of all known tags for a feature

Note

requires internet access

References

https://wiki.openstreetmap.org/wiki/Map_Features

Examples

```
## Not run:
available_tags("aerialway")

## End(Not run)
```

bbox_to_string *Convert a named matrix or a named or unnamed vector to a string*

Description

This function converts a bounding box into a string for use in web apis

Usage

```
bbox_to_string(bbox)
```

Arguments

bbox	bounding box as character, matrix or vector. If character, the bbox will be found (geocoded) and extracted with getbb. Unnamed vectors will be sorted appropriately and must merely be in the order (x, y, x, y).

Value

A character string representing min x, min y, max x, and max y bounds. For example: "15.3152361,76.4406446,15.355236 is the bounding box for Hampi, India.

Examples

```
## Not run:
bbox_to_string (getbb ("hampi india"))

## End(Not run)
```

getbb *Get bounding box for a given place name*

Description

This function uses the free Nominatim API provided by OpenStreetMap to find the bounding box (bb) associated with place names.

Usage

```
getbb(
  place_name,
  display_name_contains = NULL,
  viewbox = NULL,
  format_out = "matrix",
  base_url = "https://nominatim.openstreetmap.org",
  featuretype = "settlement",
```

```
  limit = 10,
  key = NULL,
  silent = TRUE
)
```

Arguments

place_name	The name of the place you're searching for
display_name_contains	Text string to match with display_name field returned by https://wiki.openstreetmap.org/wiki/Nominatim
viewbox	The bounds in which you're searching
format_out	Character string indicating output format: matrix (default), string (see bbox_to_string()), data.frame (all 'hits' returned by Nominatim), sf_polygon (for polygons that work with the sf package) or polygon (full polygonal bounding boxes for each match).
base_url	Base website from where data is queried
featuretype	The type of OSM feature (settlement is default; see Note)
limit	How many results should the API return?
key	The API key to use for services that require it
silent	Should the API be printed to screen? TRUE by default

Details

It was inspired by the functions bbox from the **sp** package, bb from the **tmaptools** package and bb_lookup from the github package **nominatim** package, which can be found at https://github.com/hrbrmstr/nominatim.

See https://wiki.openstreetmap.org/wiki/Nominatim for details.

Value

Defaults to a matrix in the form: `min max x ... ... y ... ...` If `format_out = "polygon"`, one or more two-columns matrices of polygonal longitude-latitude points. Where multiple `place_name` occurrences are found within `nominatim`, each item of the list of coordinates may itself contain multiple coordinate matrices where multiple exact matches exist. If one one exact match exists with potentially multiple polygonal boundaries (for example, "london uk" is an exact match, but can mean either greater London or the City of London), only the first is returned. See examples below for illustration.

Note

Specific values of `featuretype` include "street", "city", https://wiki.openstreetmap.org/wiki/Nominatim for details). The default `featuretype = "settlement"` combines results from all intermediate levels below "country" and above "streets". If the bounding box or polygon of a city is desired, better results will usually be obtained with `featuretype = "city"`.

Examples

```
## Not run:
getbb("Salzburg")
# select based on display_name, print query url
getbb("Hereford", display_name_contains = "USA", silent = FALSE)
# top 3 matches as data frame
getbb("Hereford", format_out = "data.frame", limit = 3)
# Examples of polygonal boundaries
bb <- getbb ("london uk", format_out = "polygon") # single match
dim(bb[[1]][[1]]) # matrix of longitude/latitude pairs
bb_sf = getbb("kathmandu", format_out = "sf_polygon")
# sf:::plot.sf(bb_sf) # can be plotted if sf is installed
getbb("london", format_out = "sf_polygon")
getbb("accra", format_out = "sf_polygon") # rectangular bb
# Using an alternative service (locationiq requires an API key)
# add LOCATIONIQ=type_your_api_key_here to .Renviron:
key <- Sys.getenv("LOCATIONIQ")
if(nchar(key) ==  32) {
  getbb(place_name,
        base_url = "https://locationiq.org/v1/search.php",
        key = key)
}

## End(Not run)
```

---

get_overpass_url *get_overpass_url*

Description

Return the URL of the specified overpass API. Default is https://overpass-api.de/api/interpreter/.

Usage

```
get_overpass_url()
```

Value

The overpass API URL

See Also

set_overpass_url()

opq *Build an Overpass query*

Description

Build an Overpass query

Usage

```
opq(
  bbox = NULL,
  nodes_only = FALSE,
  datetime = NULL,
  datetime2 = NULL,
  timeout = 25,
  memsize
)
```

Arguments

bbox	Either (i) four numeric values specifying the maximal and minimal longitudes and latitudes, in the form `c(xmin,ymin,xmax,ymax)` or (ii) a character string in the form `xmin,ymin,xmax,ymax`. These will be passed to getbb to be converted to a numerical bounding box. Can also be (iii) a matrix representing a bounding polygon as returned from `getbb(...,format_out = "polygon")`.
nodes_only	If `TRUE`, query OSM nodes only. Some OSM structures such as `place = "city"` or `highway = "traffic_signals"` are represented by nodes only. Queries are built by default to return all nodes, ways, and relation, but this can be very inefficient for node-only queries. Setting this value to `TRUE` for such cases makes queries more efficient, with data returned in the `osm_points` list item.
datetime	If specified, a date and time to extract data from the OSM database as it was up to the specified date and time, as described at https://wiki.openstreetmap.org/wiki/Overpass_API/Overpass_QL#date. This *must* be in ISO8601 format ("YYYY-MM-DDThh:mm:ssZ"), where both the "T" and "Z" characters must be present.
datetime2	If specified, return the *difference* in the OSM database between `datetime` and `datetime2`, where `datetime2 > datetime`. See https://wiki.openstreetmap.org/wiki/Overpass_API/Overpass_QL#Delta_between_two_dates_.28.22diff.22.29.
timeout	It may be necessary to increase this value for large queries, because the server may time out before all data are delivered.
memsize	The default memory size for the 'overpass' server in *bytes*; may need to be increased in order to handle large queries.

Value

An `overpass_query` object

Note

See https://wiki.openstreetmap.org/wiki/Overpass_API#Resource_management_options_.28osm-script.29 for explanation of timeout and memsize (or maxsize in overpass terms). Note in particular the comment that queries with arbitrarily large memsize are likely to be rejected.

Examples

```
## Not run:
q <- getbb ("portsmouth", display_name_contains = "United States") %>%
            opq () %>%
            add_osm_feature("amenity", "restaurant") %>%
            add_osm_feature("amenity", "pub")
osmdata_sf (q) # all objects that are restaurants AND pubs (there are none!)
q1 <- getbb ("portsmouth", display_name_contains = "United States") %>%
                opq () %>%
                add_osm_feature("amenity", "restaurant")
q2 <- getbb ("portsmouth", display_name_contains = "United States") %>%
                opq () %>%
                add_osm_feature("amenity", "pub")
c (osmdata_sf (q1), osmdata_sf (q2)) # all restaurants OR pubs

# Use nodes_only to retrieve single point data only, such as for central
# locations of cities.
opq <- opq (bbox, nodes_only = TRUE) %>%
    add_osm_feature (key = "place", value = "city") %>%
    osmdata_sf (quiet = FALSE)

## End(Not run)
```

---

opq_enclosing *opq_enclosing*

Description

Find all features which enclose a given point, and optionally match specific 'key'-'value' pairs. This function is *not* intended to be combined with add_osm_feature, rather is only to be used in the sequence opq_enclosing -> opq_string -> osmdata_xml (or other extraction function). See examples for how to use.

Usage

```
opq_enclosing(
  lon,
  lat,
  key = NULL,
  value = NULL,
  enclosing = "relation",
  timeout = 25
)
```

Arguments

lon	Longitude of desired point
lat	Latitude of desired point
key	(Optional) OSM key of enclosing data
value	(Optional) OSM value matching 'key' of enclosing data
enclosing	Either 'relation' or 'way' for whether to return enclosing objects of those respective types (where generally 'relation' will correspond to multipolygon objects, and 'way' to polygon objects).
timeout	It may be necessary to increase this value for large queries, because the server may time out before all data are delivered.

Examples

```
## Not run:
# Get water body surrounding a particular point:
lat <- 54.33601
lon <- -3.07677
key <- "natural"
value <- "water"
x <- opq_enclosing (lon, lat, key, value) %>%
    opq_string () %>%
    osmdata_sf ()

## End(Not run)
```

---

opq_osm_id *Add a feature specified by OSM ID to an Overpass query*

Description

Add a feature specified by OSM ID to an Overpass query

Usage

```
opq_osm_id(id = NULL, type = NULL, open_url = FALSE)
```

Arguments

id	One or more official OSM identifiers (long-form integers)
type	Type of object; must be either node, way, or relation
open_url	If TRUE, open the OSM page of the specified object in web browser. Multiple objects (id values) will be opened in multiple pages.

Value

opq object

Note

Extracting elements by ID requires explicitly specifying the type of element. Only elements of one of the three given types can be extracted in a single query, but the results of multiple types can neverthelss be combined with the c operation of osmdata.

References

https://wiki.openstreetmap.org/wiki/Overpass_API/Overpass_QL#By_element_id

Examples

```
## Not run:
id <- c (1489221200, 1489221321, 1489221491)
dat1 <- opq_osm_id (type = "node", id = id) %>%
    opq_string () %>%
    osmdata_sf ()
dat1$osm_points # the desired nodes
id <- c (136190595, 136190596)
dat2 <- opq_osm_id (type = "way", id = id) %>%
    opq_string () %>%
    osmdata_sf ()
dat2$osm_lines # the desired ways
dat <- c (dat1, dat2) # The node and way data combined

## End(Not run)
```

---

opq_string *Convert an overpass query into a text string*

Description

Convert an osmdata query of class opq to a character string query to be submitted to the overpass API.

Usage

```
opq_string(opq)
```

Arguments

opq	An overpass_query object

Value

Character string to be submitted to the overpass API

Examples

```
## Not run:
q <- opq ("hampi india")
opq_string (q)

## End(Not run)
```

`osmdata` *osmdata class def*

Description

Imports OpenStreetMap (OSM) data into R as either 'sf' or 'sp' objects. OSM data are extracted from the overpass API and processed with very fast C++ routines for return to R. The package enables simple overpass queries to be constructed without the user necessarily understanding the syntax of the overpass query language, while retaining the ability to handle arbitrarily complex queries. Functions are also provided to enable recursive searching between different kinds of OSM data (for example, to find all lines which intersect a given point).

Usage

```
osmdata(
  bbox = NULL,
  overpass_call = NULL,
  meta = NULL,
  osm_points = NULL,
  osm_lines = NULL,
  osm_polygons = NULL,
  osm_multilines = NULL,
  osm_multipolygons = NULL
)
```

Arguments

`bbox`	bounding box
`overpass_call`	overpass_call
`meta`	metadata of overpass query, including timestamps and version numbers
`osm_points`	OSM nodes as **sf** Simple Features Collection of points or **sp** SpatialPoints-DataFrame
`osm_lines`	OSM ways **sf** Simple Features Collection of linestrings or **sp** SpatialLines-DataFrame
`osm_polygons`	OSM ways as **sf** Simple Features Collection of polygons or **sp** SpatialPolygons-DataFrame
`osm_multilines`	OSM relations as **sf** Simple Features Collection of multilinestrings or **sp** SpatialLinesDataFrame

`osm_multipolygons`
: OSM relations as **sf** Simple Features Collection of multipolygons or **sp** SpatialPolygonsDataFrame

Functions to Prepare Queries

- getbb: Get bounding box for a given place name
- bbox_to_string: Convert a named matrix or a named vector (or an unnamed vector) return a string
- overpass_status: Retrieve status of the overpass API
- opq: Build an overpass query
- add_osm_feature: Add a feature to an overpass query
- opq_string: Convert an osmdata query to overpass API string

Functions to Get Additional OSM Information

- available_features: List recognised features in OSM
- available_tags: List tags associated with a feature

Functions to Extract OSM Data

- osmdata_sf: Return OSM data in **sf** format
- osmdata_sp: Return OSM data in **sp** format
- osmdata_xml: Return OSM data in **XML** format

Functions to Search Data

- `osm_points`: Extract all `osm_points` objects
- `osm_lines`: Extract all `osm_lines` objects
- `osm_polygons`: Extract all `osm_polygons` objects
- `osm_multilines`: Extract all `osm_multilines` objects
- `osm_multipolygons`: Extract all `osm_multipolygons` objects

Note

Class constructor should never be used directly, and is only exported to provide access to the print method

Author(s)

Mark Padgham, Bob Rudis, Robin Lovelace, Maëlle Salmon

osmdata_sc *Return an OSM Overpass query as an osmdata object in* silicate *(*SC*) format.*

Description

Return an OSM Overpass query as an osmdata object in silicate (SC) format.

Usage

```
osmdata_sc(q, doc, quiet = TRUE)
```

Arguments

q	An object of class overpass_query constructed with opq and add_osm_feature. May be be omitted, in which case the osmdata object will not include the query.
doc	If missing, doc is obtained by issuing the overpass query, q, otherwise either the name of a file from which to read data, or an object of class **XML** returned from osmdata_xml.
quiet	suppress status messages.

Value

An object of class osmdata representing the original OSM hierarchy of nodes, ways, and relations.

Note

The silicate format is currently highly experimental, and recommended for use only if you really know what you're doing.

Examples

```
## Not run:
hampi_sf <- opq ("hampi india") %>%
            add_osm_feature (key="historic", value="ruins") %>%
            osmdata_sc ()

## End(Not run)
```

osmdata_sf — *Return an OSM Overpass query as an osmdata object in* **sf** *format.*

Description

Return an OSM Overpass query as an osmdata object in **sf** format.

Usage

```
osmdata_sf(q, doc, quiet = TRUE, stringsAsFactors = FALSE)
```

Arguments

q	An object of class `overpass_query` constructed with opq and add_osm_feature. May be be omitted, in which case the osmdata object will not include the query.
doc	If missing, doc is obtained by issuing the overpass query, q, otherwise either the name of a file from which to read data, or an object of class **XML** returned from osmdata_xml.
quiet	suppress status messages.
stringsAsFactors	Should character strings in 'sf' 'data.frame' be coerced to factors?

Value

An object of class `osmdata` with the OSM components (points, lines, and polygons) represented in **sf** format.

Examples

```
## Not run:
hampi_sf <- opq ("hampi india") %>%
            add_osm_feature (key="historic", value="ruins") %>%
            osmdata_sf ()

## End(Not run)
```

osmdata_sp — *Return an OSM Overpass query as an osmdata object in* **sp** *format.*

Description

Return an OSM Overpass query as an osmdata object in **sp** format.

Usage

```
osmdata_sp(q, doc, quiet = TRUE)
```

Arguments

q	An object of class overpass_query constructed with opq and add_osm_feature. May be be omitted, in which case the osmdata object will not include the query.
doc	If missing, doc is obtained by issuing the overpass query, q, otherwise either the name of a file from which to read data, or an object of class **XML** returned from osmdata_xml.
quiet	suppress status messages.

Value

An object of class osmdata with the OSM components (points, lines, and polygons) represented in **sp** format.

Examples

```
## Not run:
hampi_sp <- opq ("hampi india") %>%
            add_osm_feature (key="historic", value="ruins") %>%
            osmdata_sp ()

## End(Not run)
```

---

osmdata_xml *Return an OSM Overpass query in XML format Read an (XML format) OSM Overpass response from a string, a connection, or a raw vector.*

Description

Return an OSM Overpass query in XML format Read an (XML format) OSM Overpass response from a string, a connection, or a raw vector.

Usage

```
osmdata_xml(q, filename, quiet = TRUE, encoding)
```

Arguments

q	An object of class overpass_query constructed with opq and add_osm_feature.
filename	If given, OSM data are saved to the named file
quiet	suppress status messages.
encoding	Unless otherwise specified XML documents are assumed to be encoded as UTF-8 or UTF-16. If the document is not UTF-8/16, and lacks an explicit encoding directive, this allows you to supply a default.

Value

An object of class XML::xml_document containing the result of the overpass API query.

Note

Objects of class `xml_document` can be saved as `.xml` or `.osm` files with `xml2::write_xml`.

Examples

```
## Not run:
q <- opq ("hampi india")
q <- add_osm_feature (q, key="historic", value="ruins")
osmdata_xml (q, filename="hampi.osm")

## End(Not run)
```

`osm_elevation` *osm_elevation*

Description

Add elevation data to a previously-extracted OSM data set, using a pre-downloaded global elevation file from https://srtm.csi.cgiar.org/srtmdata/. Currently only works for `SC`-class objects returned from osmdata_sc.

Usage

```
osm_elevation(dat, elev_file)
```

Arguments

`dat`	An `SC` object produced by osmdata_sc.
`elev_file`	A vector of one or more character strings specifying paths to `.tif` files containing global elevation data.

Value

A modified version of the input `dat` with an additional `z_` column appended to the vertices.

`osm_lines` *Extract all* `osm_lines` *from an osmdata object*

Description

If `id` is of a point object, `osm_lines` will return all lines containing that point. If `id` is of a line or polygon object, `osm_lines` will return all lines which intersect the given line or polygon.

Usage

```
osm_lines(dat, id)
```

Arguments

dat	An object of class osmdata
id	OSM identification of one or more objects for which lines are to be extracted

Value

An **sf** Simple Features Collection of linestrings

Examples

```
## Not run:
dat <- opq ("hengelo nl") %>% add_osm_feature (key="highway") %>%
     osmdata_sf ()
bus <- dat$osm_points [which (dat$osm_points$highway == "bus_stop"),] %>%
        rownames () # all OSM IDs of bus stops
osm_lines (dat, bus) # all highways containing bus stops

# All lines which intersect with Piccadilly Circus in London, UK
dat <- opq ("Fitzrovia London") %>% add_osm_feature (key="highway") %>%
    osmdata_sf ()
i <- which (dat$osm_polygons$name == "Piccadilly Circus")
id <- rownames (dat$osm_polygons [i,])
osm_lines (dat, id)

## End(Not run)
```

---

osm_multilines *Extract all* osm_multilines *from an osmdata object*

Description

id must be of an osm_points or osm_lines object (and can not be the id of an osm_polygons object because multilines by definition contain no polygons. osm_multilines returns any multiline object(s) which contain the object specified by id.

Usage

```
osm_multilines(dat, id)
```

Arguments

dat	An object of class osmdata
id	OSM identification of one of more objects for which multilines are to be extracted

Value

An **sf** Simple Features Collection of multilines

Examples

```
## Not run:
dat <- opq ("London UK") %>%
    add_osm_feature (key="name", value="Thames", exact=FALSE) %>%
    osmdata_sf ()
# Get ids of lines called "The Thames":
id <- rownames (dat$osm_lines [which (dat$osm_lines$name == "The Thames"),])
# and find all multilinestring objects which include those lines:
osm_multilines (dat, id)
# Now note that
nrow (dat$osm_multilines) # = 24 multiline objects
nrow (osm_multilines (dat, id)) # = 1 - the recursive search selects the
                                # single multiline containing "The Thames"

## End(Not run)
```

---

osm_multipolygons *Extract all* osm_multipolygons *from an osmdata object*

Description

id must be of an osm_points, osm_lines, or osm_polygons object. osm_multipolygons returns any multipolygon object(s) which contain the object specified by id.

Usage

```
osm_multipolygons(dat, id)
```

Arguments

dat	An object of class osmdata
id	OSM identification of one or more objects for which multipolygons are to be extracted

Value

An **sf** Simple Features Collection of multipolygons

Examples

```
## Not run:
# find all multipolygons which contain the single polygon called
# "Chiswick Eyot" (which is an island).
dat <- opq ("London UK") %>%
    add_osm_feature (key="name", value="Thames", exact=FALSE) %>%
    osmdata_sf ()
index <- which (dat$osm_multipolygons$name == "Chiswick Eyot")
id <- rownames (dat$osm_polygons [id, ])
osm_multipolygons (dat, id)
```

```
# That multipolygon is the Thames itself, but note that
nrow (dat$osm_multipolygons) # = 14 multipolygon objects
nrow (osm_multipolygons (dat, id)) # = 1 - the main Thames multipolygon

## End(Not run)
```

`osm_points` *Extract all* `osm_points` *from an osmdata object*

Description

Extract all `osm_points` from an osmdata object

Usage

```
osm_points(dat, id)
```

Arguments

`dat`	An object of class osmdata
`id`	OSM identification of one or more objects for which points are to be extracted

Value

An **sf** Simple Features Collection of points

Examples

```
## Not run:
tr <- opq ("trentham australia") %>% osmdata_sf ()
coliban <- tr$osm_lines [which (tr$osm_lines$name == "Coliban River"),]
pts <- osm_points (tr, rownames (coliban)) # all points of river
waterfall <- pts [which (pts$waterway == "waterfall"),] # the waterfall point

## End(Not run)
```

`osm_poly2line` *Convert osmdata polygons into lines*

Description

Street networks downloaded with `add_osm_object(key = "highway")` will store any circular highways in `osm_polygons`. this function combines those with the `osm_lines` component to yield a single **sf** `data.frame` of all highways, whether polygonal or not.

Usage

```
osm_poly2line(osmdat)
```

Arguments

osmdat	An osmdata object.

Value

Modified version of same object with all osm_polygons objects merged into osm_lines.

Note

The osm_polygons field is retained, with those features also repeated as LINESTRING objects in osm_lines.

Examples

```
## Not run:
dat <- opq ("colchester uk") %>%
            add_osm_feature (key="highway") %>%
            osmdata_sf ()
# colchester has lots of roundabouts, and these are stored in 'osm_polygons'
# rather than 'osm_lines'. The former can be merged with the latter by:
dat2 <- osm_poly2line (dat)
# 'dat2' will have more lines than 'dat', but the same number of polygons
# (they are left unchanged.)

## End(Not run)
```

---

osm_polygons *Extract all* osm_polygons *from an osmdata object*

Description

If id is of a point object, osm_polygons will return all polygons containing that point. If id is of a line or polygon object, osm_polygons will return all polygons which intersect the given line or polygon.

Usage

```
osm_polygons(dat, id)
```

Arguments

dat	An object of class osmdata
id	OSM identification of one or more objects for which polygons are to be extracted

Value

An **sf** Simple Features Collection of polygons

Examples

```
## Not run:
Extract polygons which intersect Conway Street in London
dat <- opq ("Marylebone London") %>% add_osm_feature (key="highway") %>%
    osmdata_sf ()
conway <- which (dat$osm_lines$name == "Conway Street")
id <- rownames (dat$osm_lines [conway,])
osm_polygons (dat, id)

## End(Not run)
```

---

overpass_status *Retrieve status of the Overpass API*

Description

Retrieve status of the Overpass API

Usage

```
overpass_status(quiet = FALSE)
```

Arguments

quiet	if FALSE display a status message

Value

an invisible list of whether the API is available along with the text of the message from Overpass and the timestamp of the next available slot

---

set_overpass_url *set_overpass_url*

Description

Set the URL of the specified overpass API. Possible APIs with global coverage are:

- "https://overpass-api.de/api/interpreter" (default)
- "https://overpass.kumi.systems/api/interpreter"
- "https://overpass.osm.rambler.ru/cgi/interpreter"
- "https://api.openstreetmap.fr/oapi/interpreter"
- "https://overpass.osm.vi-di.fr/api/interpreter"

Additional APIs with limited local coverage include:

- "https://overpass.osm.ch/api/interpreter" (Switzerland)
- "https://overpass.openstreetmap.ie/api/interpreter" (Ireland)

Usage

```
set_overpass_url(overpass_url)
```

Arguments

`overpass_url`	The desired overpass API URL

Details

For further details, see https://wiki.openstreetmap.org/wiki/Overpass_API

Value

The overpass API URL

See Also

`get_overpass_url()`

`trim_osmdata` *trim_osmdata*

Description

Trim an osmdata object to within a bounding polygon

Usage

```
trim_osmdata(dat, bb_poly, exclude = TRUE)
```

Arguments

dat	An osmdata object returned from osmdata_sf or osmdata_sp.
bb_poly	A matrix representing a bounding polygon obtained with getbb (..., format_out = "polygon") (and possibly selected from resultant list where multiple polygons are returned).
exclude	If TRUE, objects are trimmed exclusively, only retaining those strictly within the bounding polygon; otherwise all objects which partly extend within the bounding polygon are retained.

Value

A trimmed version of dat, reduced only to those components lying within the bounding polygon.

Note

It will generally be necessary to pre-load the **sf** package for this function to work correctly.

Examples

```
## Not run:
dat <- opq ("colchester uk") %>%
            add_osm_feature (key="highway") %>%
            osmdata_sf (quiet = FALSE)
bb <- getbb ("colchester uk", format_out = "polygon")
library (sf) # required for this function to work
dat_tr <- trim_osmdata (dat, bb)
bb <- getbb ("colchester uk", format_out = "sf_polygon")
class (bb) # sf data.frame
dat_tr <- trim_osmdata (dat, bb)
bb <- as (bb, "Spatial")
class (bb) # SpatialPolygonsDataFrame
dat_tr <- trim_osmdata (dat, bb)

## End(Not run)
```

---

unique_osmdata *unique_osmdata*

Description

Reduce the components of an osmdata object to only unique items of each type. That is, reduce $osm_points to only those points not present in other objects (lines, polygons, etc.); reduce $osm_lines to only those lines not present in multiline objects; and reduce $osm_polygons to only those polygons not present in multipolygon objects. This renders an osmdata object more directly compatible with typical output of **sf**.

Usage

```
unique_osmdata(dat)
```

Arguments

dat	An osmdata object

Value

Equivalent object reduced to only unique objects of each type

---

`unname_osmdata_sf` *unname_osmdata_sf*

Description

Remove names from osmdata geometry objects, for cases in which these cause issues, particularly with plotting, such as https://github.com/rstudio/leaflet/issues/631, or https://github.com/r-spatial/sf/issues/1177. Note that removing these names also removes any ability to inter-relate the different components of an osmdata object, so use of this function is only recommended to resolve issues such as those linked to above.

Usage

```
unname_osmdata_sf(x)
```

Arguments

x	An 'osmdata_sf' object returned from function of same name

Value

Same object, yet with no row names on geometry objects.

Index