

# Package: osmdata (via r-universe)

August 24, 2024

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

**Version** 0.2.5.019

**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, httr2, lubridate, magrittr, methods, Rcpp (>= 0.12.4), reproj, rvest, tibble, utils, xml2

**Suggests** httptest2, jsonlite, knitr, markdown, raster, rmarkdown, sf, sp, testthat

**LinkingTo** Rcpp

**VignetteBuilder** knitr

**Encoding** UTF-8

**NeedsCompilation** yes

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.3.1

**X-schema.org-applicationCategory** Data Access

**X-schema.org-isPartOf** <https://ropensci.org>

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

**Repository** <https://ropensci.r-universe.dev>

**RemoteUrl** <https://github.com/ropensci/osmdata>

**RemoteRef** main  
**RemoteSha** c4bd595756774049c14998a7f1cf175430b04aec

**Contents**

add_osm_feature . . . . .	2
add_osm_features . . . . .	4
available_features . . . . .	6
available_tags . . . . .	7
bbox_to_string . . . . .	8
getbb . . . . .	9
get_overpass_url . . . . .	11
opq . . . . .	11
opq_around . . . . .	14
opq_csv . . . . .	15
opq_enclosing . . . . .	16
opq_osm_id . . . . .	17
opq_string . . . . .	19
osmdata . . . . .	20
osmdata_data_frame . . . . .	22
osmdata_sc . . . . .	23
osmdata_sf . . . . .	24
osmdata_sp . . . . .	25
osmdata_xml . . . . .	26
osm_elevation . . . . .	27
osm_lines . . . . .	27
osm_multilines . . . . .	28
osm_multipolygons . . . . .	29
osm_points . . . . .	30
osm_poly2line . . . . .	31
osm_polygons . . . . .	32
overpass_status . . . . .	33
set_overpass_url . . . . .	33
trim_osmdata . . . . .	34
unique_osmdata . . . . .	35
unname_osmdata_sf . . . . .	36

<b>Index</b>	<b>37</b>
--------------	-----------

---

add_osm_feature	<i>Add a feature to an Overpass query</i>
-----------------	---

---

**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; can be negated with an initial exclamation mark, key = "!this", and can also be a vector if value is missing.
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 <a href="https://wiki.openstreetmap.org/wiki/Overpass_API">https://wiki.openstreetmap.org/wiki/Overpass_API</a>
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

**add\_osm\_feature vs add\_osm\_features**

Features defined within an [add\\_osm\\_features](#) call are combined with a logical OR.

Chained calls to either [add\\_osm\\_feature](#) or `add_osm_features()` combines features from these calls in a logical AND; this is analagous to chaining `dplyr::filter()` on a data frame.

`add_osm_features()` with only one feature is logically equivalent to `add_osm_feature()`.

**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](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](https://wiki.openstreetmap.org/wiki/Map_Features)

**See Also**

[add\\_osm\\_features](#)

Other queries: [add\\_osm\\_features\(\)](#), [bbox\\_to\\_string\(\)](#), [getbb\(\)](#), [opq\(\)](#), [opq\\_around\(\)](#), [opq\\_csv\(\)](#), [opq\\_enclosing\(\)](#), [opq\\_osm\\_id\(\)](#), [opq\\_string\(\)](#), [overpass\\_status\(\)](#)

**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")

# key negation without warnings
q3 <- opq ("Vinçà", osm_type="node") %>%
  add_osm_feature (key = c("name", "!name:ca"))
q4 <- opq ("el Carxe", osm_type="node") %>%
  add_osm_feature (key = "natural", value = "peak") %>%
  add_osm_feature (key = "!ele")

## End(Not run)
```

---

add\_osm\_features

*Add multiple features to an Overpass query*

---

**Description**

Alternative version of [add\\_osm\\_feature](#) for creating single queries with multiple features. Key-value matching may be controlled by using the filter symbols described in [https://wiki.openstreetmap.org/wiki/Overpass\\_API/Overpass\\_QL#By\\_tag\\_.28has-kv.29](https://wiki.openstreetmap.org/wiki/Overpass_API/Overpass_QL#By_tag_.28has-kv.29).

**Usage**

```
add_osm_features(
  opq,
  features,
  bbox = NULL,
  key_exact = TRUE,
  value_exact = TRUE
)
```

**Arguments**

opq	An overpass_query object
features	A named list or vector with the format <code>list("&lt;key&gt;" = "&lt;value&gt;")</code> or <code>c("&lt;key&gt;" = "&lt;value&gt;")</code> or a character vector of key-value pairs with keys and values enclosed in escape-formatted quotations. See examples for details.
bbox	optional bounding box for the feature query; must be set if no opq query bbox has been set.
key_exact	If FALSE, key is not interpreted exactly; see <a href="https://wiki.openstreetmap.org/wiki/Overpass_API">https://wiki.openstreetmap.org/wiki/Overpass_API</a>
value_exact	If FALSE, value is not interpreted exactly

**Value**

[opq](#) object

**add\_osm\_feature vs add\_osm\_features**

Features defined within an [add\\_osm\\_features](#) call are combined with a logical OR.

Chained calls to either [add\\_osm\\_feature](#) or `add_osm_features()` combines features from these calls in a logical AND; this is analagous to chaining `dplyr::filter()` on a data frame.

`add_osm_features()` with only one feature is logically equivalent to `add_osm_feature()`.

**References**

[https://wiki.openstreetmap.org/wiki/Map\\_Features](https://wiki.openstreetmap.org/wiki/Map_Features)

**See Also**

[add\\_osm\\_feature](#)

Other queries: [add\\_osm\\_feature\(\)](#), [bbox\\_to\\_string\(\)](#), [getbb\(\)](#), [opq\(\)](#), [opq\\_around\(\)](#), [opq\\_csv\(\)](#), [opq\\_enclosing\(\)](#), [opq\\_osm\\_id\(\)](#), [opq\\_string\(\)](#), [overpass\\_status\(\)](#)

**Examples**

```
## Not run:
q <- opq ("portsmouth usa") %>%
  add_osm_features (features = list (
    "amenity" = "restaurant",
    "amenity" = "pub"
  ))

q <- opq ("portsmouth usa") %>%
  add_osm_features (features = c (
    "\"amenity\"=\"restaurant\"",
    "\"amenity\"=\"pub\""
  ))

# This extracts in a single query the same result as the following:
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

## End(Not run)
```

---

available_features	<i>List recognized features in OSM</i>
--------------------	--

---

**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](https://wiki.openstreetmap.org/wiki/Map_Features)

**See Also**

Other osminfo: [available\\_tags\(\)](#)

**Examples**

```
## Not run:  
available_features ()  
  
## End(Not run)
```

---

available_tags	<i>List tags associated with a feature</i>
----------------	--

---

**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](https://wiki.openstreetmap.org/wiki/Map_Features)

**See Also**

Other osminfo: [available\\_features\(\)](#)

**Examples**

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

---

bbox_to_string	<i>Convert a named matrix or a named or unnamed vector or data.frame to a string</i>
----------------	--

---

## 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, vector or a data.frame with <code>osm_type</code> and <code>osm_id</code> columns. If character, the bbox will be found (geocoded) and extracted with <a href="#">getbb</a> . 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: "15.3152361,76.4406446,15.3552361,76.4406446" is the bounding box for Hampi, India. For data.frames with OSM objects, a character string representing a set of OSM objects in overpass query language. For example: "relation(id:11747082)" represents the area of the Catalan Countries. A set of objects can also be represented for multirow data.frames (e.g. "relation(id:11747082,307833); way(id:22422490)").

## See Also

Other queries: [add\\_osm\\_feature\(\)](#), [add\\_osm\\_features\(\)](#), [getbb\(\)](#), [opq\(\)](#), [opq\\_around\(\)](#), [opq\\_csv\(\)](#), [opq\\_enclosing\(\)](#), [opq\\_osm\\_id\(\)](#), [opq\\_string\(\)](#), [overpass\\_status\(\)](#)

## Examples

```
## Not run:
bbox_to_string (getbb ("València"))
bbox_to_string (getbb ("València", format_out = "data.frame"))

## End(Not run)
```



---

getbb	<i>Get bounding box for a given place name</i>
-------	--

---

### 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 <a href="https://wiki.openstreetmap.org/wiki/Nominatim">https://wiki.openstreetmap.org/wiki/Nominatim</a>
viewbox	The bounds in which you're searching
format_out	Character string indicating output format: matrix (default), string (see <a href="#">bbox_to_string()</a> ), data.frame (all 'hits' returned by Nominatim), sf_polygon (for polygons that work with the sf package), polygon (full polygonal bounding boxes for each match) or osm_type_id (string for querying inside defined OSM areas <a href="#">bbox_to_string()</a> ).
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 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.

For `format_out = "osm_type_id"`, a character string representing an OSM object in overpass query language. For example: `"relation(id:11747082)"` represents the area of the Catalan Countries. If one exact match exists with potentially multiple polygonal boundaries, only the first relation or way is returned. A set of objects can also be represented for multiple results (e.g. `relation(id:11747082,307833); way(id:22422490)`). See examples below for illustration. The OSM objects that can be used as **areas in overpass queries** *must be closed rings* (ways or relations).

## 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"`.

## See Also

Other queries: `add_osm_feature()`, `add_osm_features()`, `bbox_to_string()`, `opq()`, `opq_around()`, `opq_csv()`, `opq_enclosing()`, `opq_osm_id()`, `opq_string()`, `overpass_status()`

## Examples

```
## Not run:
getbb ("Salzburg")
# select based on display_name, print query url
getbb ("Hereford", display_name_contains = "United States", 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

area <- getbb ("València", format_out = "osm_type_id")
# select multiple areas with format_out = "osm_type_id"
areas <- getbb ("València", format_out = "data.frame")
bbox_to_string (areas [areas$osm_type != "node", ])
```

```

# 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	<i>get_overpass_url</i>
------------------	-------------------------

---

### 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\(\)](#)

Other overpass: [set\\_overpass\\_url\(\)](#)

---

opq	<i>Build an Overpass query</i>
-----	--------------------------------

---

### Description

Build an Overpass query

## Usage

```
opq(
  bbox = NULL,
  nodes_only = FALSE,
  osm_types = c("node", "way", "relation"),
  out = c("body", "tags", "meta", "skel", "tags center", "ids"),
  datetime = NULL,
  datetime2 = NULL,
  adiff = FALSE,
  timeout = 25,
  memsize
)
```

## Arguments

bbox	Either (i) four numeric values specifying the maximal and minimal longitudes and latitudes, in the form <code>c(xmin, ymin, xmax, ymax)</code> or (ii) a character string in the form <code>xmin,ymin,xmax,ymax</code> . These will be passed to <a href="#">getbb</a> to be converted to a numerical bounding box. Can also be (iii) a matrix representing a bounding polygon as returned from <code>getbb(..., format_out = "polygon")</code> . To search in an area, (iv) a character string with a relation or a (closed) way id in the format <code>"way(id:1)", "relation(id:1, 2)"</code> or <code>"relation(id:1, 2, 3); way(id:2)"</code> as returned by <code>getbb(..., format_out = "osm_type_id")</code> or <a href="#">bbox_to_string</a> with a <code>data.frame</code> from <code>getbb(..., format_out = "data.frame")</code> to select all areas combined (relations and ways).
nodes_only	WARNING: this parameter is equivalent to <code>osm_types = "node"</code> and will be DEPRECATED. If TRUE, query OSM nodes only. Some OSM structures such as <code>place = "city"</code> or <code>highway = "traffic_signals"</code> 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 <code>osm_points</code> list item.
osm_types	A character vector with several OSM types to query: <code>node</code> , <code>way</code> and <code>relation</code> is the default. <code>nwr</code> , <code>nw</code> , <code>wr</code> , <code>nr</code> and <code>rel</code> are also valid types. Ignored if <code>nodes_only = TRUE</code> . <code>osm_types = "node"</code> is equivalent to <code>nodes_only = TRUE</code> .
out	The level of verbosity of the overpass result: <code>body</code> (geometries and tags, the default), <code>tags</code> (tags without geometry), <code>meta</code> (like <code>body</code> + <code>Timestamp</code> , <code>Version</code> , <code>Changeset</code> , <code>User</code> , <code>User ID</code> of the last edition), <code>skel</code> (geometries only), <code>tags_center</code> (tags without geometry + the coordinates of the center of the bounding box) and <code>ids</code> (type and id of the objects only).
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 <a href="https://wiki.openstreetmap.org/wiki/Overpass_API/Overpass_QL#date">https://wiki.openstreetmap.org/wiki/Overpass_API/Overpass_QL#date</a> . This <i>must</i> be in ISO8601 format ("YYYY-MM-DDThh:mm:ssZ"), where both the "T" and "Z" characters must be present.
datetime2	If specified, return the <i>difference</i> in the OSM database between <code>datetime</code> and <code>datetime2</code> , where <code>datetime2 &gt; datetime</code> . See <a href="https://wiki.openstreetmap.org/wiki/Overpass_API/Overpass_QL#date">https://wiki.openstreetmap.org/wiki/Overpass_API/Overpass_QL#date</a> .

	<a href="https://wiki.openstreetmap.org/wiki/Overpass_API/Overpass_QL#Difference_between_two_dates_(diff).">org/wiki/Overpass_API/Overpass_QL#Difference_between_two_dates_(diff).</a>
adiff	If TRUE, query for <b>augmented difference</b> . The result indicates what happened to the modified and deleted OSM objects. Requires <code>datetime(2)*</code> .
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 <i>bytes</i> ; may need to be increased in order to handle large queries.

## Details

The `out` statement for tags, tags center and id, do not return geometries. Neither `out = "meta"` nor `adiff = TRUE` options are implemented for all `osmdata_*` functions yet. Use [osmdata\\_xml](#) or [osmdata\\_data\\_frame](#) to get the result of these queries. See the documentation of the **out statement** and **augmented difference** for more details about these options.

## Value

An `overpass_query` object

## Note

See [https://wiki.openstreetmap.org/wiki/Overpass\\_API#Resource\\_management\\_options\\_.28osm-script.29](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.

## See Also

Other queries: [add\\_osm\\_feature\(\)](#), [add\\_osm\\_features\(\)](#), [bbox\\_to\\_string\(\)](#), [getbb\(\)](#), [opq\\_around\(\)](#), [opq\\_csv\(\)](#), [opq\\_enclosing\(\)](#), [opq\\_osm\\_id\(\)](#), [opq\\_string\(\)](#), [overpass\\_status\(\)](#)

## 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)

# Filter by a search area
qa1 <- getbb ("Catalan Countries", format_out = "osm_type_id") %>%
  opq (nodes_only = TRUE) %>%
  add_osm_feature (key = "capital", value = "4")
opqa1 <- osmdata_sf (qa1)
# Filter by a multiple search areas
bb <- getbb ("Vilafranca", format_out = "data.frame")
qa2 <- bbox_to_string (bb [bb$osm_type != "node", ]) %>%
  opq (nodes_only = TRUE) %>%
  add_osm_feature (key = "place")
opqa2 <- osmdata_sf (qa2)

## End(Not run)

```

---

opq\_around

opq\_around

---

## Description

Find all features around 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\\_around](#) -> [osmdata\\_xml](#) (or other extraction function). See examples for how to use.

## Usage

```
opq_around(lon, lat, radius = 15, key = NULL, value = NULL, timeout = 25)
```

## Arguments

lon	Longitude of desired point
lat	Latitude of desired point
radius	Radius in metres around the point for which data should be extracted. Queries with large values for this parameter may fail.
key	(Optional) OSM key of enclosing data
value	(Optional) OSM value matching 'key' of enclosing data
timeout	It may be necessary to increase this value for large queries, because the server may time out before all data are delivered.

## See Also

Other queries: [add\\_osm\\_feature\(\)](#), [add\\_osm\\_features\(\)](#), [bbox\\_to\\_string\(\)](#), [getbb\(\)](#), [opq\(\)](#), [opq\\_csv\(\)](#), [opq\\_enclosing\(\)](#), [opq\\_osm\\_id\(\)](#), [opq\\_string\(\)](#), [overpass\\_status\(\)](#)

## Examples

```
## Not run:
# Get all benches ("amenity=bench") within 100m of a particular point
lat <- 53.94542
lon <- -2.52017
key <- "amenity"
value <- "bench"
radius <- 100
x <- opq_around(lon, lat, radius, key, value) %>%
  osmdata_sf()

## End(Not run)
```

---

opq\_csv

*Transform an Overpass query to return the result in a csv format*


---

## Description

Transform an Overpass query to return the result in a csv format

## Usage

```
opq_csv(q, fields, header = TRUE)
```

## Arguments

q	A opq string or an object of class <code>overpass_query</code> constructed with <a href="#">opq</a> or alternative opq builders (+ <a href="#">add_osm_feature/s</a> ).
fields	a character vector with the field names.
header	if FALSE, do not ask for column names.

## Details

The output format csv, ask for results in csv. See [CSV output mode](#) for details. To get the data, use [osmdata\\_data\\_frame](#).

## Value

The `overpass_query` or string with the prefix changed to return a csv.

## Note

csv queries that reach the timeout will return a 0 row data.frame without any warning. Increase timeout in q if you don't see the expected result.

## See Also

Other queries: [add\\_osm\\_feature\(\)](#), [add\\_osm\\_features\(\)](#), [bbox\\_to\\_string\(\)](#), [getbb\(\)](#), [opq\(\)](#), [opq\\_around\(\)](#), [opq\\_enclosing\(\)](#), [opq\\_osm\\_id\(\)](#), [opq\\_string\(\)](#), [overpass\\_status\(\)](#)

## Examples

```
## Not run:
q <- getbb ("Catalan Countries", format_out = "osm_type_id") %>%
  opq (out = "tags center", osm_type = "relation", timeout = 100) %>%
  add_osm_feature ("admin_level", "7") %>%
  add_osm_feature ("boundary", "administrative") %>%
  opq_csv (fields = c("name", "::type", "::id", "::lat", "::lon"))
comarques <- osmdata_data_frame (q) # without timeout parameter, 0 rows

qid<- opq_osm_id (
  type = "relation",
  id = c ("341530", "1809102", "1664395", "343124"),
  out = "tags"
) %>%
  opq_csv (fields = c ("name", "name:ca"))
cities <- osmdata_data_frame (qid)

## 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 = NULL,
  lat = NULL,
  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.

### See Also

Other queries: [add\\_osm\\_feature\(\)](#), [add\\_osm\\_features\(\)](#), [bbox\\_to\\_string\(\)](#), [getbb\(\)](#), [opq\(\)](#), [opq\\_around\(\)](#), [opq\\_csv\(\)](#), [opq\\_osm\\_id\(\)](#), [opq\\_string\(\)](#), [overpass\\_status\(\)](#)

### 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,
  out = "body",
  datetime = NULL,
  datetime2 = NULL,
  adiff = FALSE,
  timeout = 25,
  memsize
)
```

**Arguments**

id	One or more official OSM identifiers (long-form integers), which must be entered as either a character or <i>numeric</i> value (because R does not support long-form integers). id can also be a character string prefixed with the id type, e.g. "relation/11158003"
type	Type of objects (recycled); must be either node, way, or relation. Optional if id is prefixed with the type.
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.
out	The level of verbosity of the overpass result: body (geometries and tags, the default), tags (tags without geometry), meta (like body + Timestamp, Version, Changeset, User, User ID of the last edition), skel (geometries only), tags_center (tags without geometry + the coordinates of the center of the bounding box) and ids (type and id of the objects only).
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 <a href="https://wiki.openstreetmap.org/wiki/Overpass_API/Overpass_QL#date">https://wiki.openstreetmap.org/wiki/Overpass_API/Overpass_QL#date</a> . This <i>must</i> be in ISO8601 format ("YYYY-MM-DDThh:mm:ssZ"), where both the "T" and "Z" characters must be present.
datetime2	If specified, return the <i>difference</i> in the OSM database between datetime and datetime2, where datetime2 > datetime. See <a href="https://wiki.openstreetmap.org/wiki/Overpass_API/Overpass_QL#Difference_between_two_dates_(diff)">https://wiki.openstreetmap.org/wiki/Overpass_API/Overpass_QL#Difference_between_two_dates_(diff)</a> .
adiff	If TRUE, query for <b>augmented difference</b> . The result indicates what happened to the modified and deleted OSM objects. Requires datetime(2)*.
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 <i>bytes</i> ; may need to be increased in order to handle large queries.

**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 nevertheless be combined with the **c** operation of [osmdata](#).

**References**

[https://wiki.openstreetmap.org/wiki/Overpass\\_API/Overpass\\_QL#By\\_element\\_id](https://wiki.openstreetmap.org/wiki/Overpass_API/Overpass_QL#By_element_id)

**See Also**

Other queries: [add\\_osm\\_feature\(\)](#), [add\\_osm\\_features\(\)](#), [bbox\\_to\\_string\(\)](#), [getbb\(\)](#), [opq\(\)](#), [opq\\_around\(\)](#), [opq\\_csv\(\)](#), [opq\\_enclosing\(\)](#), [opq\\_string\(\)](#), [overpass\\_status\(\)](#)

**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
# All in one (same result as dat)
id <- c (1489221200, 1489221321, 1489221491, 136190595, 136190596)
type <- c ("node", "node", "node", "way", "way")
datAi0 <- opq_osm_id (id = id, type = type) %>%
  opq_string () %>%
  osmdata_sf ()

## 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

**See Also**

Other queries: [add\\_osm\\_feature\(\)](#), [add\\_osm\\_features\(\)](#), [bbox\\_to\\_string\(\)](#), [getbb\(\)](#), [opq\(\)](#), [opq\\_around\(\)](#), [opq\\_csv\(\)](#), [opq\\_enclosing\(\)](#), [opq\\_osm\\_id\(\)](#), [overpass\\_status\(\)](#)

## 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 'sf', 'SC', 'sp', 'data.frame' or 'xml\_document' 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 <b>sf</b> Simple Features Collection of points or <b>sp</b> SpatialPoints-DataFrame
osm_lines	OSM ways <b>sf</b> Simple Features Collection of linestrings or <b>sp</b> SpatialLines-DataFrame
osm_polygons	OSM ways as <b>sf</b> Simple Features Collection of polygons or <b>sp</b> SpatialPolygons-DataFrame
osm_multilines	OSM relations as <b>sf</b> Simple Features Collection of multilinestrings or <b>sp</b> 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\\_data\\_frame](#): Return OSM data in [data.frame](#) format
- [osmdata\\_sc](#): Return OSM data in **silicate** format
- [osmdata\\_sf](#): Return OSM data in **sf** format
- [osmdata\\_sp](#): Return OSM data in **sp** format
- [osmdata\\_xml](#): Return OSM data in **xml2** 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, Joan Maspons

**See Also**

Useful links:

- [https://docs.ropensci.org/osmdata/\(website\)https://github.com/ropensci/osmdata/\(devel\)](https://docs.ropensci.org/osmdata/(website)https://github.com/ropensci/osmdata/(devel))
- Report bugs at <https://github.com/ropensci/osmdata/issues>

---

osmdata_data_frame	<i>Return an OSM Overpass query as a <a href="#">data.frame</a> object.</i>
--------------------	---

---

**Description**

Return an OSM Overpass query as a [data.frame](#) object.

**Usage**

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

**Arguments**

q	An object of class <code>overpass_query</code> constructed with <a href="#">opq</a> and <a href="#">add_osm_feature</a> . May be omitted, in which case the attributes of the <a href="#">data.frame</a> will not include the query.
doc	If missing, <code>doc</code> is obtained by issuing the overpass query, <code>q</code> , otherwise either the name of a file from which to read data, or an object of class <b>xm12</b> returned from <a href="#">osmdata_xml</a> .
quiet	suppress status messages.
stringsAsFactors	Should character strings in the 'data.frame' be coerced to factors?

**Details**

If you are not interested in the geometries of the results, it's a good option to query for objects that match the features only and forget about members of the ways and relations. You can achieve this by passing the parameter `body = "tags"` to [opq](#).

**Value**

A `data.frame` with `id`, `type` and `tags` of the the objects from the query.

**See Also**

Other extract: [osmdata\\_sc\(\)](#), [osmdata\\_sf\(\)](#), [osmdata\\_sp\(\)](#), [osmdata\\_xml\(\)](#)

## Examples

```
## Not run:
hampi_df <- opq ("hampi india") %>%
  add_osm_feature (key = "historic", value = "ruins") %>%
  osmdata_data_frame ()
attr (hampi_df, "bbox")
attr (hampi_df, "overpass_call")
attr (hampi_df, "meta")

## End(Not run)
```

---

osmdata_sc	<i>Return an OSM Overpass query as an <a href="#">osmdata</a> object in silicate (SC) format.</i>
------------	---

---

## 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 <code>overpass_query</code> constructed with <a href="#">opq</a> and <a href="#">add_osm_feature</a> . May be omitted, in which case the <a href="#">osmdata</a> object will not include the query.
doc	If missing, <code>doc</code> is obtained by issuing the overpass query, <code>q</code> , otherwise either the name of a file from which to read data, or an object of class <b>xm12</b> returned from <a href="#">osmdata_xml</a> .
quiet	suppress status messages.

## Value

An object of class `osmdata_sc` 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.

## See Also

Other extract: [osmdata\\_data\\_frame\(\)](#), [osmdata\\_sf\(\)](#), [osmdata\\_sp\(\)](#), [osmdata\\_xml\(\)](#)

## Examples

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

## End(Not run)
```

---

osmdata_sf	<i>Return an OSM Overpass query as an <a href="#">osmdata</a> object in <b>sf</b> format.</i>
------------	---

---

## 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 <code>overpass_query</code> constructed with <a href="#">opq</a> and <a href="#">add_osm_feature</a> . May be omitted, in which case the <a href="#">osmdata</a> 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 <b>xml2</b> returned from <a href="#">osmdata_xml</a> .
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.

## See Also

Other extract: [osmdata\\_data\\_frame\(\)](#), [osmdata\\_sc\(\)](#), [osmdata\\_sp\(\)](#), [osmdata\\_xml\(\)](#)

## Examples

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

## End(Not run)
```



---

osmdata_sp	<i>Return an OSM Overpass query as an <a href="#">osmdata</a> object in <b>sp</b> format.</i>
------------	---

---

## 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 <code>overpass_query</code> constructed with <a href="#">opq</a> and <a href="#">add_osm_feature</a> . May be omitted, in which case the <a href="#">osmdata</a> 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 <b>xml2</b> returned from <a href="#">osmdata_xml</a> .
quiet	suppress status messages.

## Value

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

## See Also

Other extract: [osmdata\\_data\\_frame\(\)](#), [osmdata\\_sc\(\)](#), [osmdata\\_sf\(\)](#), [osmdata\\_xml\(\)](#)

## Examples

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

## End(Not run)
```

---

osmdata_xml	<i>Return an OSM Overpass query in XML format Read an (XML format) OSM Overpass response from a string, a connection, or a raw vector.</i>
-------------	--

---

## 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 <code>overpass_query</code> constructed with <a href="#">opq</a> and <a href="#">add_osm_feature</a> .
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 `xml2::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`.

## See Also

Other extract: [osmdata\\_data\\_frame\(\)](#), [osmdata\\_sc\(\)](#), [osmdata\\_sf\(\)](#), [osmdata\\_sp\(\)](#)

## 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	<i>osm_elevation</i>
---------------	----------------------

---

### 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 <a href="#">osmdata_sc</a> .
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.

### See Also

Other transform: [osm\\_poly2line\(\)](#), [trim\\_osmdata\(\)](#), [unique\\_osmdata\(\)](#), [unnname\\_osmdata\\_sf\(\)](#)

---

osm_lines	<i>Extract all osm_lines from an osmdata object</i>
-----------	---

---

### 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 <a href="#">osmdata</a>
id	OSM identification of one or more objects for which lines are to be extracted

**Value**

An **sf** Simple Features Collection of linestrings

**See Also**

Other search: [osm\\_multilines\(\)](#), [osm\\_multipolygons\(\)](#), [osm\\_points\(\)](#), [osm\\_polygons\(\)](#)

**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 <a href="#">osmdata</a>
id	OSM identification of one of more objects for which multilines are to be extracted

**Value**

An **sf** Simple Features Collection of multilines

**See Also**

Other search: [osm\\_lines\(\)](#), [osm\\_multipolygons\(\)](#), [osm\\_points\(\)](#), [osm\\_polygons\(\)](#)

**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	<i>Extract all osm_multipolygons from an osmdata object</i>
-------------------	---

---

**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 <a href="#">osmdata</a>
id	OSM identification of one or more objects for which multipolygons are to be extracted

**Value**

An [sf](#) Simple Features Collection of multipolygons

**See Also**

Other search: [osm\\_lines\(\)](#), [osm\\_multilines\(\)](#), [osm\\_points\(\)](#), [osm\\_polygons\(\)](#)

**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 <a href="#">osmdata</a>
id	OSM identification of one or more objects for which points are to be extracted

**Value**

An [sf](#) Simple Features Collection of points

**See Also**

Other search: [osm\\_lines\(\)](#), [osm\\_multilines\(\)](#), [osm\\_multipolygons\(\)](#), [osm\\_polygons\(\)](#)

**Examples**

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

## End(Not run)
```

---

osm_poly2line	<i>Convert osmdata polygons into lines</i>
---------------	--

---

## 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`.

## See Also

Other transform: `osm_elevation()`, `trim_osmdata()`, `unique_osmdata()`, `unnname_osmdata_sf()`

## 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	<i>Extract all osm_polygons from an osmdata object</i>
--------------	--

---

### 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

<code>dat</code>	An object of class <a href="#">osmdata</a>
<code>id</code>	OSM identification of one or more objects for which polygons are to be extracted

### Value

An `sf` Simple Features Collection of polygons

### See Also

Other search: [osm\\_lines\(\)](#), [osm\\_multilines\(\)](#), [osm\\_multipolygons\(\)](#), [osm\\_points\(\)](#)

### 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	<i>Retrieve status of the Overpass API</i>
-----------------	--

---

**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

**See Also**

Other queries: [add\\_osm\\_feature\(\)](#), [add\\_osm\\_features\(\)](#), [bbox\\_to\\_string\(\)](#), [getbb\(\)](#), [opq\(\)](#), [opq\\_around\(\)](#), [opq\\_csv\(\)](#), [opq\\_enclosing\(\)](#), [opq\\_osm\\_id\(\)](#), [opq\\_string\(\)](#)

---

set_overpass_url	<i>set_overpass_url</i>
------------------	-------------------------

---

**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](https://wiki.openstreetmap.org/wiki/Overpass_API)

**Value**

The overpass API URL

**See Also**

[get\\_overpass\\_url\(\)](#)

Other overpass: [get\\_overpass\\_url\(\)](#)

---

trim_osmdata	<i>trim_osmdata</i>
--------------	---------------------

---

**Description**

Trim an [osmdata](#) object to within a bounding polygon

**Usage**

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

**Arguments**

dat	An <a href="#">osmdata</a> object returned from <a href="#">osmdata_sf</a> or <a href="#">osmdata_sp</a> .
bb_poly	A matrix representing a bounding polygon obtained with <code>getbb(..., format_out = "polygon")</code> (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.

Caution is advised when using polygons obtained from Nominatim via `getbb(..., format_out = "polygon"|"sf_polygon")`. These shapes can be outdated and thus could cause the trimming operation to not give results expected based on the current state of the OSM data.

**See Also**

Other transform: [osm\\_elevation\(\)](#), [osm\\_poly2line\(\)](#), [unique\\_osmdata\(\)](#), [unname\\_osmdata\\_sf\(\)](#)

**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	<i>unique_osmdata</i>
----------------	-----------------------

---

**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

**See Also**

Other transform: [osm\\_elevation\(\)](#), [osm\\_poly2line\(\)](#), [trim\\_osmdata\(\)](#), [unname\\_osmdata\\_sf\(\)](#)

---

unname_osmdata_sf	<i>unname_osmdata_sf</i>
-------------------	--------------------------

---

## 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.

## See Also

Other transform: [osm\\_elevation\(\)](#), [osm\\_poly2line\(\)](#), [trim\\_osmdata\(\)](#), [unique\\_osmdata\(\)](#)

## Examples

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

# All coordinate matrices include rownames with OSM ID values:
head (as.matrix (hampi_sf$osm_lines$geometry [[1]]))
# But 'unname_osmdata_sf' removes both row and column names:
head (as.matrix (hampi_clean$osm_lines$geometry [[1]]))

## End(Not run)
```

# Index

- \* **extract**
  - osmdata\_data\_frame, 22
  - osmdata\_sc, 23
  - osmdata\_sf, 24
  - osmdata\_sp, 25
  - osmdata\_xml, 26
- \* **osminfo**
  - available\_features, 6
  - available\_tags, 7
- \* **overpass**
  - get\_overpass\_url, 11
  - set\_overpass\_url, 33
- \* **package**
  - osmdata, 20
- \* **queries**
  - add\_osm\_feature, 2
  - add\_osm\_features, 4
  - bbox\_to\_string, 8
  - getbb, 9
  - opq, 11
  - opq\_around, 14
  - opq\_csv, 15
  - opq\_enclosing, 16
  - opq\_osm\_id, 17
  - opq\_string, 19
  - overpass\_status, 33
- \* **search**
  - osm\_lines, 27
  - osm\_multilines, 28
  - osm\_multipolygons, 29
  - osm\_points, 30
  - osm\_polygons, 32
- \* **transform**
  - osm\_elevation, 27
  - osm\_poly2line, 31
  - trim\_osmdata, 34
  - unique\_osmdata, 35
  - unname\_osmdata\_sf, 36
- add\_osm\_feature, 2, 3–5, 8, 10, 13–19, 21–26, 33
- add\_osm\_features, 3, 4, 4, 5, 8, 10, 13–15, 17–19, 33
- available\_features, 3, 6, 7, 21
- available\_tags, 7, 7, 21
- bbox\_to\_string, 4, 5, 8, 10, 12–15, 17–19, 21, 33
- bbox\_to\_string(), 9
- c, 18
- data.frame, 21, 22
- get\_overpass\_url, 11, 34
- get\_overpass\_url(), 34
- getbb, 4, 5, 8, 9, 12–15, 17–19, 21, 33
- opq, 3–5, 8, 10, 11, 14, 15, 17–19, 21–26, 33
- opq\_around, 4, 5, 8, 10, 13, 14, 14, 15, 17–19, 33
- opq\_csv, 4, 5, 8, 10, 13, 14, 15, 17–19, 33
- opq\_enclosing, 4, 5, 8, 10, 13–16, 16, 18, 19, 33
- opq\_osm\_id, 4, 5, 8, 10, 13–15, 17, 17, 19, 33
- opq\_string, 3–5, 8, 10, 13–18, 19, 21, 33
- opq\_to\_string(opq\_string), 19
- osm\_elevation, 27, 31, 35, 36
- osm\_lines, 27, 29, 30, 32
- osm\_multilines, 28, 28, 29, 30, 32
- osm\_multipolygons, 28, 29, 29, 30, 32
- osm\_points, 28, 29, 30, 32
- osm\_poly2line, 27, 31, 35, 36
- osm\_polygons, 28–30, 32
- osmdata, 18, 20, 23–25, 27–32, 34, 35
- osmdata-package (osmdata), 20
- osmdata\_data\_frame, 13, 15, 21, 22, 23–26
- osmdata\_sc, 21, 22, 23, 24–27
- osmdata\_sf, 21–23, 24, 25, 26, 34
- osmdata\_sp, 21–24, 25, 26, 34
- osmdata\_xml, 13, 14, 16, 21–25, 26

overpass\_status, [4](#), [5](#), [8](#), [10](#), [13–15](#), [17–19](#),  
[21](#), [33](#)

set\_overpass\_url, [11](#), [33](#)

set\_overpass\_url(), [11](#)

trim\_osmdata, [27](#), [31](#), [34](#), [35](#), [36](#)

unique\_osmdata, [27](#), [31](#), [35](#), [35](#), [36](#)

unname\_osmdata\_sf, [27](#), [31](#), [35](#), [36](#)