Package 'osmar'

February 20, 2015

Type Package

Title OpenStreetMap and R

Version 1.1-7

Date 2013-11-16

Author Thomas Schlesinger and Manuel J. A. Eugster

Maintainer Thomas Schlesinger <tho.schlesinger@googlemail.com>

Description This package provides infrastructure to access
OpenStreetMap data from different sources, to work with the data
in common R manner, and to convert data into available
infrastructure provided by existing R packages (e.g., into sp and
igraph objects).

Depends R (>= 2.10), methods, XML, RCurl, geosphere

Suggests igraph, sp (>= 0.9-93)

License GPL-2

LazyLoad yes

URL http://osmar.r-forge.r-project.org/

Collate 'osm-descriptors.R' 'source.R' 'osmar-plotting.R'
'as-osmar-elements.R' 'as-osmar.R' 'as-osm.R' 'as-sp.R' 'get.R'
'osmar-subsetting.R' 'osmar-finding.R' 'osmar.R' 'source-api.R'
'source-file.R' 'source-osmosis.R' 'as-osmar-sp.R'
'as-igraph.R'

Repository CRAN

Repository/R-Forge/Project osmar

Repository/R-Forge/Revision 98

Repository/R-Forge/DateTimeStamp 2013-11-16 18:37:16

Date/Publication 2013-11-21 08:33:54

NeedsCompilation no

R topics documented:

as_igraph 2
as_osm 3
as_osmar 3
as_osmar_bbox 4
as_osmar_bbox.Spatial 5
as_sp 5
c.osmar 6
corner_bbox 7
dim.osmar 8
find 8
find_down 9
find_nearest_node 11
get_osm 12
muc 13
node 14
osmsource_api 15
osmsource_file 16
osmsource_osmosis 17
osm_crs 18
plot.osmar 19
subset.osmar 20
summary.osmar 21
%grep% 22

Index **23**

`as_igraph` *Convert osmar object to igraph*

Description

Convert an osmar object to an igraph (see igraph-package).

Usage

```
as_igraph(obj)
```

Arguments

`obj`	An `osmar` object

Value

An igraph-package `graph` object

Examples

```
file <- system.file("extdata", "kaufstr.xml", package = "osmar")
raw <- readLines(file)
kaufstr <- as_osmar(xmlParse(raw))
kaufstrGraph <- as_igraph(kaufstr)
```

`as_osm` *Convert osmar object to OSM-XML*

Description

Convert an osmar object to an OSM-XML object.

Usage

```
as_osm(obj, ...)
```

Arguments

`obj`	An `osmar` object
`...`	Ignored

Value

An xml object

Note

Not yet implemented!

`as_osmar` *Convert OSM-XML to an osmar object*

Description

Convert a given OSM-XML object (as parsed by `xmlParse`) to an osmar object.

Usage

```
as_osmar(xml)
```

Arguments

`xml`	An OSM-XML object

Value

A list (with class attribute `osmar`) with three elements:

`nodes` A list with two data frames containing the attributes and tags of the nodes.

`ways` A list with three data frames containing the attributes, tags, and references of the ways.

`relations` A list with three data frames containing the attributes, tags, and references of the relations.

Examples

```
file <- system.file("extdata", "kaufstr.xml", package = "osmar")
  raw <- readLines(file)
  kaufstr <- as_osmar(xmlParse(raw))
```

`as_osmar_bbox` *Bounding box converter generic*

Description

Generic function for implementing converters from various objects (e.g., `sp` `Spatial` objects) to osmar `bbox` objects.

Usage

```
as_osmar_bbox(obj, ...)
```

Arguments

`obj`	Object to compute osmar `bbox`
`...`	Additional parameters for underlying functions

See Also

Other as_osmar_bbox: `as_osmar_bbox.Spatial`, `center_bbox`, `corner_bbox`

`as_osmar_bbox.Spatial` *Convert sp object to an osmar object*

Description

Functions to convert a given `sp` object to osmar infrastructure and objects.

Usage

```
## S3 method for class 'Spatial'
as_osmar_bbox(obj, ...)
```

Arguments

`obj`	A `Spatial` object
`...`	Ignored

Value

A `bbox` object

See Also

Other as_osmar_bbox: `as_osmar_bbox`, `center_bbox`, `corner_bbox`

Examples

```
data("muc", package = "osmar")
  muc_points <- as_sp(muc, "points")
  bbox(muc_points)           # sp::bbox object
  as_osmar_bbox(muc_points)  # osmar::bbox object
```

`as_sp` *Convert osmar object to sp object*

Description

Convert an osmar object to a `sp` object.

Usage

```
as_sp(obj, what = c("points", "lines", "polygons"),
  crs = osm_crs(), simplify = TRUE)
```

Arguments

`obj`	An `osmar` object
`what`	A string describing the sp-object; see Details section
`crs`	A valid `CRS` object; default value is given by `osm_crs`-function
`simplify`	Should the result list be simplified to one element if possible?

Details

Depending on the strings given in `what` the `osmar` object will be converted into in a list of objects given by the sp-package:

`what = "points"` the object will be converted in a `SpatialPointsDataFrame`. The data slot is filled with the attrs slot of `obj$nodes`.

`what = "lines"` the object will be converted in a `SpatialLinesDataFrame`. It is build with all possible elements which are in `obj$ways` `obj$relations`. The data slot is filled with elements of both.

`what = "polygons"` the object will be converted in a `SpatialPolygonsDataFrame`. It consists of elements which are in `obj$ways` slot.

Every conversion needs at least a non-empty `obj$nodes$attrs`-slot because spatial information are stored in there.

Value

A list of one or more sp objects; see Details section.

Examples

```
data("muc", package = "osmar")
  muc_points <- as_sp(muc, "points")
  muc_lines <- as_sp(muc, "lines")
  muc_polygons <- as_sp(muc, "polygons")

  bbox(muc_points)
```

`c.osmar` *Combine osmar objects*

Description

Combine two or more `osmar` objects.

Usage

```
 ## S3 method for class 'osmar'
c(...)
```

Arguments

`...`	`osmar` objects to be concatenated

Value

An `osmar` object based on the provided objects

Examples

```
## Not run:
    muc <- get_osm(center_bbox(11.575278, 48.137222, 200, 200))
    o1 <- subset(muc, node_ids = find(muc, node(tags(v == "Marienplatz"))))
    o2 <- subset(muc, ids = find_down(muc, way(c(96619179, 105071000))))

    o1
    o2
    c(o1, o2)

## End(Not run)
```

`corner_bbox` *Get OSM elements*

Description

Utility functions to specify *what* to get from the OSM data source. These are the request elements which work for most sources, see the specific sources for specialized elements.

Usage

```
corner_bbox(left, bottom, right, top)

center_bbox(center_lon, center_lat, width, height)
```

Arguments

`left`	Minimum longitude
`bottom`	Minimum latitude
`right`	Maximum longitude
`top`	Maximum latitutde
`center_lon`	Center longitude
`center_lat`	Center latitude
`width`	Box width
`height`	Box height

See Also

`osm_descriptors`, `get_osm`

Other as_osmar_bbox: `as_osmar_bbox`, `as_osmar_bbox.Spatial`

`dim.osmar` *Dimension of osmar objects*

Description

Dimension of osmar objects

Usage

```
## S3 method for class 'osmar'
dim(x)
```

Arguments

`x`	An `osmar` object

Value

A named vector with the number of nodes, ways and relations.

Examples

```
## Not run: 
    muc <- get_osm(center_bbox(11.575278, 48.137222, 200, 200))
    dim(muc)

## End(Not run)
```

`find` *Find element for a given condition*

Description

Find element for a given condition

Usage

```
find(object, condition)
```

Arguments

`object`	An `osmar` object
`condition`	A condition for the element to find; see details section.

Details

The basis of an `osmar` object are `data.frames`; therefore the `condition` principally follows the rules for `subset`: logical expression indicating elements or rows to keep.

Furthermore, one has to define on which element and which data of the `osmar` object the condition applies: `element(data(condition))`, see `osm_descriptors`.

Value

The ID of the the element

See Also

binary_grep

Other finding: `find_down`, `find_nearest_node`, `find_up`

Examples

```
data("muc", package = "osmar")
  find(muc, node(tags(v == "Marienplatz")))
  find(muc, node(tags(v %agrep% "marienplatz")))
  find(muc, node(attrs(id == 19475890)))
  find(muc, way(tags(k == "highway" & v == "pedestrian")))
```

`find_down` *Find all elements related to an ID*

Description

For a given ID these functions return all IDs of related elements.

Usage

```
find_down(object, ids)

find_up(object, ids)
```

Arguments

`object`	An `osmar` object
`ids`	A vector of IDs tagged whether they are `node`, `way`, or `relation`

Details

`find_down` finds all elements downwards the hierarchy:

node	->	node
way	->	way + node
relation	->	relation + way + node

`find_up` finds all elements upwards the hierarchy:

node	->	node + way + relation
way	->	way + relation
relation	->	relation

Value

A list with the three elements `node_ids`, `way_ids`, `relation_ids`

See Also

Other finding: `find`, `find_nearest_node`

Examples

```
data("muc", package = "osmar")
  o1 <- find(muc, way(tags(k == "highway" & v == "pedestrian")))

  find_down(muc, way(o1))
  find_up(muc, way(o1))
```

`find_nearest_node` *Find nearest node with given conditions*

Description

For a given ID, find nearest node (geographical distance) with given conditions.

Usage

```
find_nearest_node(object, id, condition)
```

Arguments

`object`	An `osmar` object
`id`	An node ID
`condition`	Condition for the element to find; see `find`

Value

A node ID or `NA`

See Also

Other finding: find, find_down, find_up

Examples

```
data("muc", package = "osmar")
  id <- find(muc, node(tags(v == "Marienplatz")))[1]

  find_nearest_node(muc, id, way(tags(k == "highway" & v == "pedestrian")))
```

---

get_osm *Get OSM data*

Description

Get OSM data as osmar object from different sources by providing a bounding box.

Usage

```
get_osm(x, source = osmsource_api(), ...)
```

Arguments

x	Data identifier, e.g., bounding box or specific element; see the help page of the used OSM source for a detailed list on the supported identifiers
source	OSM source, e.g., osmsource_api
...	Additional arguments suppported by the specific OSM source; see corresponding source help page for a detailed list

Value

An osmar object

See Also

bbox, osm_descriptors, osmsource_api, osmsource_osmosis

Examples

```
## Not run:
  api <- osmsource_api()

  box <- corner_bbox(11.579341, 48.15102, 11.582852, 48.1530)
  gschw <- get_osm(box, source = api)

  kaufstr <- get_osm(way(3810479))
  kaufstr_full <- get_osm(way(3810479), full = TRUE)

## End(Not run)
```

`muc` *Object of class osmar from central Munich*

Description

Data retrieved with `get_osm(center_bbox(11.575278, 48.137222, 200, 200))`.

Usage

```
data(muc)
```

Format

The format is: List of 3 $ nodes :List of 2 ..$ attrs:'data.frame': 975 obs. of 9 variables: .. ..$ id : num [1:975] 1955016 17780035 18929510 18929515 18929522 ... .$ lat : num [1:975] 48.1 48.1 48.1 48.1 48.1 ... .$ lon : num [1:975] 11.6 11.6 11.6 11.6 11.6 ... .$ user : Factor w/ 36 levels "chan","ckol",..: 26 24 13 12 21 6 6 13 26 21 ... .$ uid : Factor w/ 36 levels "107037","109029",..: 6 29 14 34 15 10 10 14 6 15 ... .$ visible : Factor w/ 1 level "true": 1 1 1 1 1 1 1 1 1 1 ... .$ version : num [1:975] 3 35 3 3 6 6 6 3 3 3 ... .$ changeset: num [1:975] 10239803 10484152 6909578 1460631 10162612 ... .$ timestamp: POSIXlt[1:975], format: "2011-12-29 21:07:53" "2012-01-24 12:51:04" ... ..$ tags :'data.frame': 662 obs. of 3 variables: .. ..$ id: num [1:662] 17780035 17780035 17780035 17780035 17780035 ... .$ k : Factor w/ 109 levels "addr:city","addr:country",..: 18 35 36 37 42 43 44 45 46 47 ... .$ v : Factor w/ 291 levels "-0.5","-1","-2",..: 43 196 211 102 194 167 288 170 194 196 ... ..- attr(*, "class")= chr [1:3] "nodes" "osmar_element" "list" $ ways :List of 3 ..$ attrs:'data.frame': 214 obs. of 7 variables: .. ..$ id : num [1:214] 9.66e+07 8.58e+07 8.58e+07 1.05e+08 1.05e+08 ... .$ user : Factor w/ 26 levels "FK270673","FloSch",..: 22 7 7 23 22 8 23 19 21 11 ... .$ uid : Factor w/ 26 levels "109029","130472",..: 24 10 10 15 24 6 15 22 3 9 ... .$ visible : Factor w/ 1 level "true": 1 1 1 1 1 1 1 1 1 1 ... .$ version : num [1:214] 2 1 1 2 1 3 2 2 2 3 ... .$ changeset: num [1:214] 7622488 6411339 6411339 7857000 7622488 ... .$ timestamp: POSIXlt[1:214], format: "2011-03-20 22:13:47" "2010-11-20 00:18:02" ... ..$ tags :'data.frame': 607 obs. of 3 variables: .. ..$ id: num [1:607] 9.66e+07 9.66e+07 8.58e+07 8.58e+07 1.05e+08 ... .$ k : Factor w/ 52 levels "addr:city","addr:country",..: 22 26 11 11 25 26 22 26 11 25 ... .$ v : Factor w/ 88 levels "-0.5","-1","-2",..: 64 6 88 88 4 4 44 4 88 4 ... ..$ refs :'data.frame': 1262 obs. of 2 variables: .. ..$ id : num [1:1262] 96619179 96619179 85765758 85765758 85765758 ... .$ ref: num [1:1262] 1.12e+09 3.40e+08 9.96e+08 9.96e+08 9.96e+08 ... ..- attr(*, "class")= chr [1:3] "ways" "osmar_element" "list" $ relations:List of 3 ..$ attrs:'data.frame': 56 obs. of 7 variables: .. ..$ id : num [1:56] 1773072 1796136 1843975 1792663 30479 ... .$ user : Factor w/ 20 levels "Andreas Binder",..: 15 8 8 8 12 10 12 9 15 2 ... .$ uid : Factor w/ 20 levels "109029","13832",..: 1 16 16 16 2 14 2 8 1 6 ... .$ visible : Factor w/ 1 level "true": 1 1 1 1 1 1 1 1 1 1 ... .$ version : num [1:56] 14 5 6 13 91 4 48 7 15 62 ... .$ changeset: num [1:56] 10510995 10210507 10210507 10210507 10393071 ... .$ timestamp: POSIXlt[1:56], format: "2012-01-27 10:36:50" "2011-12-26 19:56:40" ... ..$ tags :'data.frame': 425 obs. of 3 variables: .. ..$ id: num [1:425] 1773072 1773072 1773072 1773072 1773072 ... .$ k : Factor w/ 47 levels "admin_level",..: 10 14 24 25 26 33 35 43 44 46 ... .$ v : Factor w/ 150 levels "-3","0","09",..: 93 106 91 70 40 125 139 61 100 72 ... ..$ refs :'data.frame': 6119 obs. of 4 variables: .. ..$ id : num [1:6119] 1773072 1773072 1773072 1773072 1773072 ... .$ type: Factor w/ 3 levels "node","relation",..: 1 3 1 3 1

3 1 3 3 1 ... .$ ref : num [1:6119] 1.45e+09 1.32e+08 6.00e+07 5.59e+07 6.00e+07 ... .$ role: Factor w/ 11 levels "","admin_centre",..: 11 8 11 8 11 8 11 9 10 11 ... ..- attr(*, "class")= chr [1:3] "relations" "osmar_element" "list" - attr(*, "class")= chr [1:2] "osmar" "list"

Source

http://www.openstreetmap.org/, downloaded 10 February 2012.

See Also

find, as_sp

Examples

```
data("muc", package = "osmar")
```

---

node *Element descriptors*

Description

For getting OSM data and finding elements in an osmar object one needs to describe the data—here we provide a simple description language.

Usage

```
 node(object)

 way(object)

 relation(object)

 ## Default S3 method:
node(object)

 ## Default S3 method:
way(object)

 ## Default S3 method:
relation(object)

 attrs(condition)

 tags(condition)

 refs(condition)

 ## S3 method for class 'condition'
```

```
relation(object)

 ## S3 method for class 'condition'
relation(object)

 ## S3 method for class 'condition'
relation(object)
```

Arguments

`object`	The descriptor; see details
`condition`	Condition to describe the object

See Also

bbox

Examples

```
## Description by ID (*.default):
  node(1)
way(1)
relation(1)
## Description by condition (*.condition):
  node(tags(v == "Marienplatz"))
## Description by condition (*.condition):
  way(attrs(id == 17458))
```

`osmsource_api` *API OSM data source*

Description

OSM API version 0.6 data source; see `http://wiki.openstreetmap.org/wiki/API_v0.6`.

Usage

```
osmsource_api(url = "http://api.openstreetmap.org/api/0.6/")
```

Arguments

`url`	URL of the API

Supported request elements

Bounding box: Use `corner_bbox` or `center_bbox` to retrieve:

- all nodes that are inside a given bounding box and any relations that reference them;
- all ways that reference at least one node that is inside a given bounding box, any relations that reference them [the ways], and any nodes outside the bounding box that the ways may reference;
- all relations that reference one of the nodes or ways included due to the above rules (does not apply recursively);

Basic request elements: Use `node`, `way`, `relation` to retrieve an element by its ID.

Use `full = TRUE` as additional argument to the `get_osm` function. This means that all members of the specified elements are retrieved as well:

- For a way, it will return the way specified plus all nodes referenced by the way.
- For a relation, it will return: (1) the relation itself; (2) all nodes, ways, and relations that are members of the relation; and (3) all nodes used by ways from the previous step.

References

http://wiki.openstreetmap.org/wiki/API_v0.6

See Also

`get_osm`, `bbox`, `osm_descriptors`

Other osmsource: `osmsource_file`, `osmsource_osmosis`

Examples

```
## Not run:
    api <- osmsource_api()

    box <- corner_bbox(11.579341, 48.15102, 11.582852, 48.1530)
    gschw <- get_osm(box, source = api)

    kaufstr <- get_osm(way(3810479))
    kaufstr_full <- get_osm(way(3810479), full = TRUE)

## End(Not run)
```

`osmsource_file` *OSM file data source*

Description

Imports the complete OSM file.

Usage

```
osmsource_file(file)
```

Arguments

`file`	The file name (and path) of the osm file

Supported request elements

Dummy request element: Use the function `compete_file` as dummy description for all elements

See Also

`get_osm`, `bbox`, `osm_descriptors`

Other osmsource: `osmsource_api`, `osmsource_osmosis`

Examples

```
## Not run:
    get_osm(complete_file(), source = osmsource_file("muc.osm"))

## End(Not run)
```

`osmsource_osmosis` *Osmosis OSM data source*

Description

Planet dumps as OSM data source through the osmosis command line Java application.

Usage

```
osmsource_osmosis(file, osmosis = "osmosis")
```

Arguments

`file`	The file name (and path) of the planet dump
`osmosis`	The path to the osmosis application

Details

Osmosis is a command line Java application for processing OSM data. It allows, among other things, to extract data inside a bounding box or polygon from so called planet dumps. The usage of this source requires an installed osmosis; see `http://wiki.openstreetmap.org/wiki/Osmosis`.

Supported request elements

Bounding box: Use corner_bbox or center_bbox to retrieve:

- all nodes that are inside a given bounding box and any relations that reference them;
- all ways that reference at least one node that is inside a given bounding box, any relations that reference them [the ways], and any nodes outside the bounding box that the ways may reference;
- all relations that reference one of the nodes or ways included due to the above rules (does not apply recursively);

References

http://wiki.openstreetmap.org/wiki/Osmosis

See Also

get_osm, bbox, osm_descriptors

Other osmsource: osmsource_api, osmsource_file

Examples

```
## Not run:
    ## Download and extract a planet file:
    download.file("http://osmar.r-forge.r-project.org/",
                  "muenchen.osm.gz")
    system("gzip -d muenchen.osm.gz")

    ## Define osmosis source; note that we assume that
    ## osmosis is in our path environment variable (if
    ## not, set osmosis argument to the executable):
    src <- osmsource_osmosis(file = "muenchen.osm")

    ## Get the center of Munich:
    muc_bbox <- center_bbox(11.575278, 48.137222,
                            3000, 3000)
    muc <- get_osm(muc_bbox, src)
    muc

## End(Not run)
```

---

osm_crs *CRS for OpenStreetMap*

Description

Coordinate Reference System used in OpenStreetMap.

Usage

```
osm_crs(crs = "+proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs +towgs84=0,0,0")
```

Arguments

`crs`	A valid proj4 string

Details

The default value is the WGS84 Ellipsoid which is used in GPS, therefore it is used in OpenStreetMap.

Value

A `CRS` object

Examples

```
osm_crs()
  class(osm_crs())
```

`plot.osmar` *Plot osmar object*

Description

Simple plotting functions to visualize `osmar` objects. Note that for more complex plots, we suggest to convert the objects into sp and use their plotting functionality.

Usage

```
## S3 method for class 'osmar'
plot(x,
  way_args = list(col = gray(0.7)),
  node_args = list(pch = 19, cex = 0.1, col = gray(0.3)),
  ...)

plot_nodes(x, add = FALSE, ...)

plot_ways(x, add = FALSE, xlab = "lon", ylab = "lat",
  ...)
```

Arguments

`x`	An `osmar` object
`way_args`	A list of parameters for plotting ways
`node_args`	A list of parameters for plotting nodes
`...`	Ignored
`add`	New plot device or plot on existing onde
`xlab`	A x-axis label
`ylab`	A y-axis label

`subset.osmar` *Subset an osmar object*

Description

Subset an osmar object

Usage

```
  ## S3 method for class 'osmar'
 subset(x, node_ids = NULL,
    way_ids = NULL, relation_ids = NULL,
   ids = list(node_ids = node_ids, way_ids = way_ids, relation_ids = relation_ids),
    ...)
```

Arguments

`x`	An `osmar` object
`node_ids`	Node ID vector
`way_ids`	Way ID vector
`relation_ids`	Relation ID vector
`ids`	A list composed of `node_ids`, `way_ids`, `relation_ids`; for easier usage with results from `find_up` and `find_down`
`...`	Ignored

Value

An `osmar` object containing the specified elements

Examples

```
data("muc", package = "osmar")
  id <- find(muc, node(tags(v == "Marienplatz")))

  subset(muc, node_ids = id)

  subset(muc, ids = find_up(muc, node(id)))
```

summary.osmar *Summarize osmar objects*

Description

Summaries of osmar, nodes, ways, and relations objects. Use these methods to get an overview of the content.

Usage

```
 ## S3 method for class 'osmar'
summary(object, ...)

 ## S3 method for class 'summary.osmar'
print(x, max.print = 3,
   nchar.value = 20, ...)

 ## S3 method for class 'nodes'
summary(object, ...)

 ## S3 method for class 'summary.nodes'
print(x, max.print = 10,
   nchar.value = 20, ...)

 ## S3 method for class 'ways'
summary(object, ...)

 ## S3 method for class 'summary.ways'
print(x, max.print = 10,
   nchar.value = 20, ...)

 ## S3 method for class 'relations'
summary(object, ...)

 ## S3 method for class 'summary.relations'
print(x, max.print = 10,
   nchar.value = 20, ...)
```

Arguments

`object`	An object (`osmar`, `nodes`, `ways`, or `relations` for which a summary is desired
`...`	Ignored
`x`	The computed summary object to print
`max.print`	Maximum number of shown tags
`nchar.value`	Number of shown characters of the value column

Value

`summary.osmar` returns a list with the summaries for nodes, ways, and relations.

`summary.nodes`, `summary.ways`, `summary.relations` all return a list with

`key` A contingency table of the counts of each key label; in descending order

`val` A contingency table of the counts of each value label; in descending order

`keyval` A contingency table of the counts greater zero of each combination of key and value labels; in descending order

See Also

`osmar`

`%grep%` *Binary operators for grep-like functions*

Description

Binary operators for grep-like functions to use in conditions similar to the "==" operator.

Usage

```
x

x
```

Details

`x, ignore.case = TRUE).`

`x, ignore.case = TRUE)` and converts the index result into a logical vector.

Index

∗Topic **datasets**
  muc, 13
%agrep% (%grep%), 22
%grep%, 22

as_igraph, 2
as_osm, 3
as_osmar, 3
as_osmar_bbox, 4, *5*, *8*
as_osmar_bbox.Spatial, *4*, 5, *8*
as_sp, 5, *14*
attrs (node), 14

bbox, *4, 5*, *12*, *15–18*
bbox (corner_bbox), 7

c.osmar, 6
center_bbox, *4, 5*, *16*, *18*
center_bbox (corner_bbox), 7
complete_file (osmsource_file), 16
corner_bbox, *4, 5*, 7, *16*, *18*
CRS, *6*, *19*

dim.osmar, 8

find, 8, *11, 12*, *14*
find_down, *9*, 9, *12*, *20*
find_nearest_node, *9*, *11*, 11
find_up, *9*, *12*, *20*
find_up (find_down), 9

get_osm, *8*, 12, *16–18*

muc, 13

node, 14, *16*

osm_crs, *6*, 18
osm_descriptors, *8, 9*, *12*, *16–18*
osm_descriptors (node), 14
osmar, *2, 3*, *6–9*, *11, 12*, *14*, *19, 20*, *22*

osmar (as_osmar), 3
osmsource_api, *12*, 15, *17, 18*
osmsource_file, *16*, 16, *18*
osmsource_osmosis, *12*, *16, 17*, 17

plot.osmar, 19
plot_nodes (plot.osmar), 19
plot_ways (plot.osmar), 19
print.summary.nodes (summary.osmar), 21
print.summary.osmar (summary.osmar), 21
print.summary.relations
  (summary.osmar), 21
print.summary.ways (summary.osmar), 21

refs (node), 14
relation, *16*
relation (node), 14

sp, *4–6*
Spatial, *4, 5*
SpatialLinesDataFrame, *6*
SpatialPointsDataFrame, *6*
SpatialPolygonsDataFrame, *6*
subset, *9*
subset.osmar, 20
summary.nodes (summary.osmar), 21
summary.osmar, 21
summary.relations (summary.osmar), 21
summary.ways (summary.osmar), 21

tags (node), 14

way, *16*
way (node), 14

xmlParse, *3*