Package ‘rosm’

April 8, 2017

Type Package
Title Plot Raster Map Tiles from Open Street Map and Other Sources
Version 0.2.2
Date 2017-04-6
Encoding UTF-8
Maintainer Dewey Dunnington <dewey@fishandwhistle.net>
Description Download and plot Open Street Map <http://www.openstreetmap.org/>, Bing Maps <http://www.bing.com/maps> and other tiled map sources in a way that works seamlessly with plotting from the ‘sp’ package. Use to create high-resolution basemaps and add hillshade to vector-based maps.
License GPL-2
LazyData TRUE
Imports curl, abind, jpeg, png, sp, rgdal, rjson, methods, plyr, prettymap
Suggests rcanvec, cartography, raster, testthat
URL https://github.com/paleolimbot/rosm
BugReports https://github.com/paleolimbot/rosm/issues
RoxygenNote 5.0.1
NeedsCompilation no
Author Dewey Dunnington [aut, cre], Timothée Giraud [ctb]
Repository CRAN
Date/Publication 2017-04-07 22:14:30 UTC

R topics documented:

as.tile_source ................................................................. 2
bmaps.plot ................................................................. 3
bmaps.types ................................................................. 4
**Description**

Tile sources define where rosm looks for map tiles. There are a number of built-in types (osm.types), or they can be created using as.tile_source(), registered using register.tile_source for easy access, or passed directly to the osm.plot family of methods.

**Usage**

```r
as.tile_source(x, ...)

is.tile_source(x)

source_from_url_format(url_format, max_zoom = tile.maxzoom.default(),
                        min_zoom = 0, attribution = NULL, ...)
```

**Arguments**

- **x**: An object (usually a name or string format) with which to create a tile source
- **...**: Arguments passed to other methods
- **url_format**: A string in the form `https://tiles.wmflabs.org/bw-mapnik/${z}/${x}/${y}.png`, where `z`, `x`, and `y` are the zoom, xtile, and ytile, respectively. Also valid is `$q`, which will be passed a quadkey.
- **max_zoom**: An integer specifying the maximum zoom to use (default is 19)
- **min_zoom**: An integer specifying the minimum zoom to use (default is 1)
- **attribution**: An attribution string, required by some tile providers.
Details

Passing a name from `osm.types` will return that tile source; passing a name from `register_tile_source` will return that tile source, and passing a URL format in the form `https://tiles.wmflabs.org/bw-mapnik/${z}/${x}/${y}.png` will create a new tile source. Old style function names in the form `tile.url.TYPE` are still supported but are deprecated.

Value

An object of class `tile_source`

Examples

```r
# get builtin tile sources
as.tile_source("osm")

# get custom tile sources
as.tile_source("http://a.basemaps.cartocdn.com/dark_all/${z}/${x}/${y}.png")

# get registered tile sources
register_tile_source(dark = "http://a.basemaps.cartocdn.com/dark_all/${z}/${x}/${y}.png")
as.tile_source("dark")

# create more complex tile sources using source_from_url_format
source_from_url_format("http://a.basemaps.cartocdn.com/dark_all/${z}/${x}/${y}.png",
  attribution = "Tiles by CartoDB")

# test for tile sources
is.tile_source(as.tile_source("osm"))
```

---

### `bmaps.plot`

#### Plot Bing Maps

**Description**


**Usage**

```r
bmaps.plot(bbox, type = "Aerial", key = NULL, ...)
```

**Arguments**

- `bbox` A bounding box as generated by `sp::bbox()` or `prettymapr::searchbbox()`
- `type` Use Aerial, AerialWithLabels, or Road.
- `key` If plotting a large number of images, consider getting your own (free) key at the [Microsoft Website](https://www.bing.com/maps/).
- `...` Arguments passed on to `osm.plot`. 
Examples

library(prettymapr)
bmaps.plot(makebbox(47.2, -59.7, 43.3, -66.4))
bmaps.plot(makebbox(47.2, -59.7, 43.3, -66.4), type="Road")

bmaps.types  

List types of Bing Maps

Description
List types of Bing Maps

Usage
bmaps.types()

Value
A list of valid bing map types

Examples
bmaps.types()

extract_bbox  

Extract a bounding box from an object

Description
This function is used internally by osm.plot, bmaps.plot, and osm.raster to extract a bounding box from their first argument. This allows considerable flexibility when specifying a location to map, in particular with character input (a place name that will be geocoded), and other Spatial*/Raster* objects.

Usage
extract_bbox(x, tolatonin = TRUE, ...)

extract_bbox
Arguments

x A Spatial* object, a Raster* object, a bounding box, or a character string that will be passed to searchbbox() (prettymapr package). Multiple strings will result in a bounding box that contains all of the geocoded bounding boxes. The last resort is calling sp::bbox() on the x.

tolatlon Should the bounding box be un-projected to lat/lon coordinates?

Value

A bounding box in the form of sp::bbox()

Examples

library(prettymapr)
ns <- makebbox(47.2, -59.7, 43.3, -66.4)
stopifnot(identical(ns, extract_bbox(ns)))

# downloads data, takes a long time to test
ns <- extract_bbox("nova scotia")

---

**osm.image**

*Get Open Street Map Tiles As A RasterStack*

Description

Get Open Street Map tiles as RasterStack object (requires package raster to be installed).

Usage

```r
osm.image(x, zoomin = 0, zoom = NULL, type = NULL,
  forcedownload = FALSE, cachefiles = NULL, progress = c("text", "none"),
  quiet = TRUE)
``` 

```r
osm.raster(x, zoomin = 0, zoom = NULL, type = "osm",
  forcedownload = FALSE, cachefiles = NULL, progress = c("text", "none"),
  quiet = TRUE, projection = NULL, crop = FALSE, filename = NULL, ...)
``` 

Arguments

x A bounding box as generated by sp::bbox() or prettymapr::searchbbox(). Must be in lon/lat (epsg:4326)! Alternatively, pass a Spatial* object to use the bounding box of that
The amount by which to adjust the automatically calculated zoom (or manually specified if the zoom parameter is passed). Use +1 to zoom in, or -1 to zoom out.

Manually specify the zoom level (not recommended; adjust zoomin instead).

A map type; one of that returned by osm.types. User defined types are possible by defining tile.url.TYPENAME <-function(xtile, ytile, zoom){ and passing TYPENAME as the type argument.

TRUE if cached tiles should be re-downloaded. Useful if some tiles are corrupted.

The directory in which tiles should be cached. Defaults to getwd()/rosm.cache.

A progress bar to use, or "none" to suppress progress updates

Pass FALSE to see more error messages, particularly if your tiles do not download/load properly.

A map projection in which to reproject the RasterStack as generated by CRS() or Spatial@proj4string. If a Spatial object is passed as the first argument, this argument will be ignored.

TRUE if results should be cropped to the specified bounding box (see x), FALSE otherwise.

A filename to which the raster should be written (see raster::writeRaster()). Use a ".tif" extension to write as a GeoTIFF.

Arguments passed on to raster::writeRaster() if filename is specified.

A projected RasterStack of the fused tiles.

Examples

library(cartography)
library(raster)
library(prettymap)

ns <- makebbox(47.2, -59.7, 43.3, -66.4)
x <- osm.raster(ns, projection=CRS("+init=epsg:26920"), crop=TRUE)
# plot using plotRGB (from the raster package)
plotRGB(x)

# use a Spatial* object as the first argument to automatically set the bounding
# box and projection
data(nuts2006)
spdf <- nuts0.spdf[nuts0.spdf$idd="DE",]
x <- osm.raster(spdf, type="thunderforestlandscape")
plotRGB(x)

# write to disk by passing a filename argument (use .tif extension to write GeoTIFF)
osm.raster(ns, projection=CRS("+init=epsg:26920"), crop=TRUE, filename="ns.tif")

# can also write Raster* objects using osm.raster
osm.raster(x, filename="germany.tif")
osm.lines

Overlay lines on an OSM plot

Description
Plot lines on a plot created by osm.plot. This is a simple wrapper around points().

Usage
osm.lines(x, y = NULL, epsg = 4326, toepsg = 3857, ...)

Arguments
- x: X coordinate vector or object as parsed by xy.coords
- y: Y coordinate vector
- epsg: EPSG code of the supplied coordinates
- toepsg: EPSG code of the projected coordinates to be ploted
- ...: Args passed on to lines

Examples
library(rosm)
library(prettymapr)
locs <- geocode(c("wolfville, ns", "kentville, ns", "halifax, ns"))
prettymap({
  osm.plot(searchbbox("nova scotia"))
  osm.lines(locs$lon, locs$lat, lwd=2)
})

osm.plot
Plot Open Street Map Tiles

Description
Plot Open Street Map tiles using rasterImage and sp::plot. Define your own tile sources by creating a tile url function in the global environment, although most OSM listed servers are included. See osm.types for types options. By default tiles are plotted in the Spherical Mercator projection (epsg:3857); pass project=FALSE to keep lat/lon coordinates.
Usage

```
osm.plot(bbox, zoomin = 0, zoom = NULL, type = NULL,
  forcedownload = FALSE, stoponlargerequest = TRUE, fusetiles = TRUE,
  cachedir = NULL, res = 150, project = TRUE, progress = c("text",
    "none"), quiet = TRUE, ...)```

Arguments

**bbox**  
A bounding box as generated by `sp::bbox()` or `prettymapr::searchbbox()`.

**zoomin**  
The amount by which to adjust the automatically calculated zoom (or manually specified if the zoom parameter is passed). Use +1 to zoom in, or -1 to zoom out.

**zoom**  
Manually specify the zoom level (not recommended; adjust zoomin or res instead).

**type**  
A map type; one of that returned by `osm.types`. User defined types are possible by defining `tile.url.TYPENAME <- function(xtile, ytile, zoom){}` and passing TYPENAME as the type argument.

**forcedownload**  
TRUE if cached tiles should be re-downloaded. Useful if some tiles are corrupted.

**stoponlargerequest**  
By default `osm.plot` will only load 32 tiles at a time. If plotting at a higher resolution it may be necessary to pass `true` here.

**fusetiles**  
TRUE if tiles should be fused into a single image. This is the default because white lines appear between tiles if it is set to FALSE. PDFs appear not to have this problem, so when plotting large, high resolution PDFs it may be faster (and more memory efficient) to use fusetiles=FALSE.

**cachedir**  
The directory in which tiles should be cached. Defaults to `getwd()/rosm.cache`.

**res**  
The resolution used to calculate scale.

**project**  
TRUE if tiles should be projected to a pseudo-mercator projection. FALSE if lat/lon should be maintained. Because `sp::plot` adjusts the aspect according to latitude for lat/lon coordinates, this makes little difference at high zoom and may make plotting overlays more convenient. Defaults to TRUE.

**progress**  
A progress bar to use, or "none" to suppress progress updates

**quiet**  
Pass FALSE to see more error messages, particularly if your tiles do not download/load properly.

...  
Additional parameters to be passed on to the first call to `sp::plot`

Examples

```
library(prettymapr)
ns <- makebbox(47.2, -59.7, 43.3, -66.4)
osm.plot(ns)
osm.plot(ns, type="stamenbw")
prettymap(osm.plot(ns), scale.style="ticks", scale.tick.cex=0)
```
osm.points

Overlay points on an OSM plot

Description
Plot points on a plot created by `osm.plot`. This is a simple wrapper around `points()`.

Usage

```r
osm.points(x, y = NULL, epsg = 4326, toepsg = 3857, ...)
```

Arguments
- `x`: X coordinate vector or object as parsed by `xy.coords`
- `y`: Y coordinate vector
- `epsg`: EPSG code of the supplied coordinates
- `toepsg`: EPSG code of the projected coordinates to be plotted
- `...`: Args passed on to `points`

Examples

```r
library(rosm)
library(prettymap)
locs <- geocode(c("wolfville, ns", "kentville, ns", "halifax, ns"))
prettymap({
  osm.plot(searchbbox("nova scotia"))
  osm.points(locs$lon, locs$lat, pch=18, cex=0.7)
})
```

---

osm.polygon

Overlay a polygon on an OSM plot

Description
Plot a polygon on a plot created by `osm.plot`. This is a simple wrapper around `polygon()`.

Usage

```r
osm.polygon(x, y = NULL, epsg = 4326, toepsg = 3857, ...)
```
Arguments

\(x\)
X coordinate vector or object as parsed by \textit{xy.coords}

\(y\)
Y coordinate vector

\(epsg\)
EPSG code of the supplied coordinates

\(toepsg\)
EPSG code of the projected coordinates to be plotted

\(...\)
Args passed on to \textit{polygon}

Examples

library(rosm)
library(prettymap)
locs <- geocode(c("wolfville, ns", "kentville, ns", "halifax, ns"))
prettymap({
osm.plot(searchbbox("nova scotia"))
osm.polygon(locs$lon, locs$lat)
})

\texttt{osm.segments \hspace{1cm} \textit{Overlay segments on an OSM plot}}

Description

Plot segments on a plot created by \texttt{osm.plot}. This is a simple wrapper around \texttt{segments()}.

Usage

\texttt{osm.segments(x0, y0, x1 = x0, y1 = y0, epsg = 4326, toepsg = 3857, ...)}

Arguments

\(x0\)
X1 coordinate vector

\(y0\)
Y1 coordinate vector

\(x1\)
X2 coordinate vector

\(y1\)
Y2 coordinate vector

\(epsg\)
EPSG code of the supplied coordinates

\(toepsg\)
EPSG code of the projected coordinates to be plotted

\(...\)
Args passed on to \textit{points}
Examples

```r
library(rosm)
library(prettymap)
locs <- geocode(c("wolfville, ns", "kentville, ns", "halifax, ns"))
prettymap(
  osm.plot(searchbbox("nova scotia"))
  osm.segments(locs$lon[1:2], locs$lat[1:2], locs$lon[2:3], locs$lat[2:3])
)
```

---

**osm.text**  
*Overlay text on an OSM plot*

**Description**

Plot text on a plot created by `osm.plot`.

**Usage**

```r
osm.text(x, y = NULL, labels = seq_along(x), epsg = 4326, toepsg = 3857, ...)
```

**Arguments**

- `x`  
  X coordinate vector or object as parsed by `xy.coords`

- `y`  
  Y coordinate vector

- `labels`  
  A character vector or expression specifying the text to be written.

- `epsg`  
  EPSG code of the supplied coordinates

- `toepsg`  
  EPSG code of the projected coordinates to be plotted

- `...`  
  Args passed on to `text()`

---

**osm.types**  
*Get List of Valid Tile Sources*

**Description**

Get List of Valid Tile Sources

**Usage**

```r
osm.types()
```
Value

A character vector of valid type parameters.

Examples

```r
osm.types()
```

---

**register_tile_source**  
**Register Tile Sources**

**Description**

Use this function to register tile sources so they can be referred to by name in `osm.plot`. Tile sources will be registered for as long as the namespace is loaded. Use `set_default_tile_source()` to set the default source.

**Usage**

```r
register_tile_source(...)

set_default_tile_source(x, ...)

get_default_tile_source()
```

**Arguments**

- `...` Passed to `as.tile_source` for `set_default_tile_source`, or a named list of tile sources for `register_tile_source`
- `x` The tile source (or coercible string) to use as the default tile source

**Examples**

```r
# set the default tile source
set_default_tile_source("stamenbw")

# register a custom tile source
register_tile_source(dark = "http://a.basemaps.cartocdn.com/dark_all/$(z)/$(x)/$(y).png")

library(prettymapr)
ns <- makebbox(47.2, -59.7, 43.3, -66.4)
prettymap(osm.plot(ns, "dark"))
```
**Plot Raster Map Tiles From Open Street Map and Other Sources**

**Description**

This package provides access and plots Open Street Map and Bing Maps tiles to create high-resolution basemaps and use hillshade tiles to add texture to other maps. Uses the 'sp' package to plot using base graphics. Plot Open Street Map derivative tiles using `osm.plot`, and plot Bing maps (Aerial, Labeled Aerial, Road) using `bmaps.plot`. 16 OSM and 3 Bing sources are included, with the ability to define custom tile sources based on OSM tilex, tiley, and zoom. Use `osm.raster` to get tiles in a RasterStack or write to disk (requires the 'raster' package.)

**Author(s)**

Dewey Dunnington <dewey@fishandwhistle.net>

**References**

Open Street Map tile servers, Bing Maps API documentation

**Examples**

```r
library(rcanvec)
library(prettymapr)
library(sp)

# basic plotting
nsbox <- searchbbox("nova scotia", source="google")
osm.plot(nsbox)
osm.plot(nsbox, type="stamenbw")
bmaps.plot(nsbox)
bmaps.plot(nsbox, type="Road")

# use (prettymap) to add scalebar and north arrow
prettymap(osm.plot(nsbox))
prettymap(bmaps.plot(nsbox, type="Road"))

# increase res argument to plot to file
pdf(height=8, width=10.5)
prettymap(osm.plot(nsbox, type="stamenbw", res=300, stoponlargerequest=FALSE),
         scale.label.col="white", arrow.text.col = "white",
         scale.linecol = "white", arrow.border = "white")
dev.off()

# use osm.raster() to export a RasterStack of tiles
library(raster)
x <- osm.raster(nsbox)
```
set_default_cachedir

Set/Get the Default Tile Cache Location

Description

The default tile cache location is the "rosm.cache" folder in the current working directory, but for a variety of reasons it may be desirable to use one cache directory for all calls in a script. This must be called every time the namespace is loaded.

Usage

set_default_cachedir(cachedir)

get_default_cachedir()

Arguments

cachedir A path to use as the cache directory (relative to the working directory). Use NULL to reset to the default.
set_default_cachedir

Value

The previous cache directory, invisibly.

Examples

```lisp
(set_default_cachedir(tempfile()))
(get_default_cachedir())
(set_default_cachedir(NULL))
```
Index

as.tile_source, 2, 12
bmaps.plot, 3, 4, 13
bmaps.types, 4
extract_bbox, 4
get_default_cachedir
   (set_default_cachedir), 14
get_default_tile_source
   (register_tile_source), 12
is.tile_source (as.tile_source), 2
osm.image, 5
osm.lines, 7
osm.plot, 2–4, 7, 7, 9–13
osm.points, 9
osm.polygon, 9
osm.raster, 4, 13
osm.raster (osm.image), 5
osm.segments, 10
osm.text, 11
osm.types, 2, 3, 6–8, 11
register_tile_source, 2, 3, 12
rosm, 13
rosm-package (rosm), 13
set_defaultCachedir, 14
set_default_tile_source
   (register_tile_source), 12
source_from_url_format
   (as.tile_source), 2