Package 'osmplotr'

October 14, 2022

Title Bespoke Images of 'OpenStreetMap' Data

Version 0.3.3

Description Bespoke images of 'OpenStreetMap' ('OSM') data and data visualisation using 'OSM' objects.

License GPL-3

URL https://docs.ropensci.org/osmplotr/, https://github.com/ropensci/osmplotr

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

Depends R (>= 3.2.3)

Imports e1071, ggm, ggplot2, mapproj, methods, osmdata, rgeos, sp, spatstat (>= 2.0-0), spatstat.core, spatstat.geom

Suggests curl, knitr, magrittr, rmarkdown, testthat

VignetteBuilder knitr

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

NeedsCompilation no

Author Mark Padgham [aut, cre],
Richard Beare [aut],
Finkelstein Noam [ctb, cph] (Author of included stub.R code),
Bartnik Lukasz [ctb, cph] (Author of included stub.R code)

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

Repository CRAN

Date/Publication 2021-03-28 00:10:03 UTC

R topics documented:

`add_axes` *add_axes*

Description

Adds axes to the internal region of an OSM plot.

Usage

```
add_axes(
  map,
  colour = "black",
  pos = c(0.02, 0.03),
  alpha = 0.4,
  fontsize = 3,
  fontface,
  fontfamily,
  ...
)
```

Arguments

`map`	A `ggplot2` object to which the axes are to be added.
`colour`	Colour of axis (determines colour of all elements: lines, ticks, and labels).
`pos`	Positions of axes and labels relative to entire plot device.
`alpha`	alpha value for semi-transparent background surrounding axes and labels (lower values increase transparency).
`fontsize`	Size of axis font (in `ggplot2` terms; default=3).
`fontface`	Fontface for axis labels (1:4=plain,bold,italic,bold-italic).

fontfamily	Family of axis font (for example, 'Times').
...	Mechanism to allow many parameters to be passed with alternative names (color for colour and xyz for fontxyz.

Value

Modified version of map with axes added.

See Also

osm_basemap.

Examples

```
bbox <- get_bbox (c (-0.13, 51.5, -0.11, 51.52))
map <- osm_basemap (bbox = bbox, bg = "gray20")
map <- add_osm_objects (map, london$dat_BNR, col = "gray40")
map <- add_axes (map)
print (map)

# Map items are added sequentially, so adding axes prior to objects will
# produce a different result.
map <- osm_basemap (bbox = bbox, bg = "gray20")
map <- add_axes (map)
map <- add_osm_objects (map, london$dat_BNR, col = "gray40")
print_osm_map (map)
```

---

add_colourbar *add_colorbar*

Description

Adds a colourbar to an existing map. Intended to be used in combination with add_osm_surface. At present, only plots on right side of map.

Usage

```
add_colourbar(
  map,
  barwidth = 0.02,
  barlength = 0.7,
  zlims,
  cols,
  vertical = TRUE,
  alpha = 0.4,
  text_col = "black",
  fontsize = 3,
  fontface,
```

```
  fontfamily,
  ...
)
```

Arguments

map	A ggplot2 object to which the colourbar is to be added.
barwidth	Relative width of the bar (perpendicular to its direction), either a single number giving distance from right or upper margin, or two numbers giving left/right or lower/upper limits.
barlength	Relative length of the bar (parallel to its direction), either a single number giving total length of centred bar, or two numbers giving lower/upper or left/right limits.
zlims	Vector of (min,max) values for scale of colourbar. These should be the values returned from add_osm_surface.
cols	Vector of colours.
vertical	If FALSE, colourbar is aligned horizontally instead of default vertical alignment.
alpha	Transparency level of region immediately surrounding colourbar, including behind text. Lower values are more transparent.
text_col	Colour of text, tick marks, and lines on colourbar.
fontsize	Size of text labels (in ggplot2 terms; default=3).
fontface	Fontface for colourbar labels (1:4=plain,bold,italic,bold-italic).
fontfamily	Family of colourbar font (for example, 'Times').
...	Mechanism to allow many parameters to be passed with alternative names (such as xyz for fontxyz).

Value

Modified version of map with colourbar added.

See Also

osm_basemap, add_osm_surface.

Examples

```
bbox <- get_bbox (c (-0.13, 51.5, -0.11, 51.52))
map <- osm_basemap (bbox = bbox, bg = "gray20")
# Align volcano data to lat-lon range of bbox
dv <- dim (volcano)
x <- seq (bbox [1,1], bbox [1,2], length.out = dv [1])
y <- seq (bbox [2,1], bbox [2,2], length.out = dv [2])
dat <- data.frame (
                   x = rep (x, dv [2]),
                   y = rep (y, each = dv [1]),
                   z = as.numeric (volcano)
                   )
```

```
map <- add_osm_surface (map, obj = london$dat_BNR, dat = dat,
                        cols = heat.colors (30))
map <- add_axes (map)
# Note colours of colourbar can be artibrarily set, and need not equal those
# passed to 'add_osm_surface'
map <- add_colourbar (map, zlims = range (volcano), cols = heat.colors(100),
                      text_col = "black")
print_osm_map (map)

# Horizontal colourbar shifted away from margins:
map <- osm_basemap (bbox = bbox, bg = "gray20")
map <- add_osm_surface (map, obj = london$dat_BNR, dat = dat,
                        cols = heat.colors (30))
map <- add_colourbar (map, zlims = range (volcano), cols = heat.colors(100),
                      barwidth = c(0.1,0.15), barlength = c(0.5, 0.9),
                      vertical = FALSE)
print_osm_map (map)
```

`add_osm_groups` *add_osm_groups*

Description

Plots spatially distinct groups of OSM objects in different colours.

Usage

```
add_osm_groups(
  map,
  obj,
  groups,
  cols,
  bg,
  make_hull = FALSE,
  boundary = -1,
  size,
  shape,
  border_width = 1,
  colmat,
  rotate
)
```

Arguments

`map`	A `ggplot2` object to which the grouped objects are to be added.
`obj`	An `sp` `SpatialPointsDataFrame`, `SpatialPolygonsDataFrame`, or `SpatialLinesDataFrame` (list of polygons or lines) returned by `extract_osm_objects`.

groups	A list of spatial points objects, each of which contains the coordinates of points defining one group.
cols	Either a vector of >= 4 colours passed to colour_mat (if colmat = TRUE) to arrange as a 2-D map of visually distinct colours (default uses rainbow colours), or (if colmat = FALSE), a vector of the same length as groups specifying individual colours for each.
bg	If given, then any objects not within groups are coloured this colour, otherwise (if not given) they are assigned to nearest group and coloured accordingly (boundary has no effect in this latter case).
make_hull	Either a single boolean value or a vector of same length as groups specifying whether convex hulls should be constructed around all groups (TRUE), or whether the group already defines a hull (convex or otherwise; FALSE).
boundary	(negative, 0, positive) values define whether the boundary of groups should (exclude, bisect, include) objects which straddle the precise boundary. (Has no effect if bg is given).
size	Size argument passed to ggplot2 (polygon, path, point) functions: determines width of lines for (polygon, line), and sizes of points. Respective defaults are (0, 0.5, 0.5).
shape	Shape of points or lines (the latter passed as linetype); see shape.
border_width	If given, draws convex hull borders around entire groups in same colours as groups (try values around 1-2).
colmat	If TRUE generates colours according to colour_mat, otherwise the colours of groups are specified directly by the vector of cols.
rotate	Passed to colour_mat to rotate colours by the specified number of degrees clockwise.

Value

Modified version of map with groups added.

Note

Any group that is entirely contained within any other group is assumed to represent a hole, such that points internal to the smaller contained group are *excluded* from the group, while those outside the smaller yet inside the bigger group are included.

See Also

colour_mat, add_osm_objects.

Examples

```
bbox <- get_bbox (c (-0.13, 51.5, -0.11, 51.52))
# Download data using 'extract_osm_objects'
## Not run:
dat_HP <- extract_osm_objects (key = 'highway',
                               value = 'primary',
```

```
                                bbox = bbox)
dat_T <- extract_osm_objects (key = 'tree', bbox = bbox)
dat_BNR <- extract_osm_objects (key = 'building', value = '!residential',
bbox = bbox)

## End(Not run)
# These data are also provided in
dat_HP <- london$dat_HP
dat_T <- london$dat_T
dat_BNR <- london$dat_BNR

# Define a function to easily generate a basemap
bmap <- function ()
{
    map <- osm_basemap (bbox = bbox, bg = "gray20")
    map <- add_osm_objects (map, dat_HP, col = "gray70", size = 1)
    add_osm_objects (map, dat_T, col = "green")
}

# Highlight a single region using all objects lying partially inside the
# boundary (via the boundary = 1 argument)
pts <- sp::SpatialPoints (cbind (c (-0.115, -0.125, -0.125, -0.115),
                                 c (51.505, 51.505, 51.515, 51.515)))
## Not run:
dat_H <- extract_osm_objects (key = 'highway', bbox = bbox) # all highways
map <- bmap ()
map <- add_osm_groups (map, dat_BNR, groups = pts, cols = "gray90",
                       bg = "gray40", boundary = 1)
map <- add_osm_groups (map, dat_H, groups = pts, cols = "gray80",
                       bg = "gray30", boundary = 1)
print_osm_map (map)

## End(Not run)

# Generate random points to serve as group centres
set.seed (2)
ngroups <- 6
x <- bbox [1,1] + runif (ngroups) * diff (bbox [1,])
y <- bbox [2,1] + runif (ngroups) * diff (bbox [2,])
groups <- cbind (x, y)
groups <- apply (groups, 1, function (i)
                 sp::SpatialPoints (
                     matrix (i, nrow = 1, ncol = 2)))
# plot a basemap and add groups
map <- bmap ()
cols <- rainbow (length (groups))
## Not run:
map <- add_osm_groups (map,
                       obj = london$dat_BNR,
                       group = groups,
                       cols = cols)
cols <- adjust_colours (cols, -0.2)
map <- add_osm_groups (map, obj = london$dat_H, groups = groups, cols = cols)
```

```
print_osm_map (map)

# Highlight convex hulls containing groups:
map <- bmap ()
map <- add_osm_groups (map,
                       obj = london$dat_BNR,
                       group = groups,
                       cols = cols,
                       border_width = 2)
print_osm_map (map)

## End(Not run)
```

---

add_osm_objects *add_osm_objects*

Description

Adds layers of spatial objects (polygons, lines, or points generated by `extract_osm_objects`) to a graphics object initialised with `osm_basemap`.

Usage

```
add_osm_objects(map, obj, col = "gray40", border = NA, hcol, size, shape)
```

Arguments

`map`	A `ggplot2` object to which the objects are to be added.
`obj`	A spatial (`sp`) data frame of polygons, lines, or points, typically as returned by `extract_osm_objects`.
`col`	Colour of lines or points; fill colour of polygons.
`border`	Border colour of polygons.
`hcol`	(Multipolygons only) Vector of fill colours for holes
`size`	Size argument passed to `ggplot2` (polygon, path, point) functions: determines width of lines for (polygon, line), and sizes of points. Respective defaults are (0, 0.5, 0.5).
`shape`	Shape of points or lines (the latter passed as `linetype`); see `shape`.

Value

modified version of `map` to which objects have been added.

See Also

`osm_basemap`, `extract_osm_objects`.

Examples

```
bbox <- get_bbox (c (-0.13, 51.5, -0.11, 51.52))
map <- osm_basemap (bbox = bbox, bg = "gray20")

## Not run:
# The 'london' data used below were downloaded as:
dat_BNR <- extract_osm_objects (bbox = bbox, key = 'building',
                                value = '!residential')
dat_HP <- extract_osm_objects (bbox = bbox, key = 'highway',
                               value = 'primary')
dat_T <- extract_osm_objects (bbox = bbox, key = 'tree')

## End(Not run)
map <- add_osm_objects (map, obj = london$dat_BNR,
                        col = "gray40", border = "yellow")
map <- add_osm_objects (map, obj = london$dat_HP, col = "gray80",
                        size = 1, shape = 2)
map <- add_osm_objects (map, london$dat_T, col = "green",
                        size = 2, shape = 1)
print_osm_map (map)

# Polygons with different coloured borders
map <- osm_basemap (bbox = bbox, bg = "gray20")
map <- add_osm_objects (map, obj = london$dat_HP, col = "gray80")
map <- add_osm_objects (map, london$dat_T, col = "green")
map <- add_osm_objects (map, obj = london$dat_BNR, col = "gray40",
                        border = "yellow", size = 0.5)
print_osm_map (map)
```

`add_osm_surface` *add_osm_surface*

Description

Adds a colour-coded surface of spatial objects (polygons, lines, or points generated by `extract_osm_objects` to a graphics object initialised with `osm_basemap`. The surface is spatially interpolated between the values given in `dat`, which has to be a matrix of `data.frame` of 3 columns (x, y, z), where (x,y) are (longitude, latitude), and z are the values to be interpolated. Interpolation uses `spatstat.core::Smooth.ppp`, which applies a Gaussian kernel smoother optimised to the given data, and is effectively non-parametric.

Usage

```
add_osm_surface(
  map,
  obj,
  dat,
  method = "idw",
  grid_size = 100,
```

```
  cols = heat.colors(30),
  bg,
  size,
  shape
)
```

Arguments

map	A ggplot2 object to which the surface are to be added
obj	An sp SpatialPolygonsDataFrame or SpatialLinesDataFrame (list of polygons or lines) returned by extract_osm_objects
dat	A matrix or data frame of 3 columns (x, y, z), where (x, y) are (longitude, latitude), and z are the values to be interpolated
method	Either idw (Inverse Distance Weighting as spatstat.core::idw; default), Gaussian for kernel smoothing (as spatstat.core::Smooth.ppp), or any other value to avoid interpolation. In this case, dat must be regularly spaced in x and y.
grid_size	size of interpolation grid
cols	Vector of colours for shading z-values (for example, terrain.colors(30))
bg	If specified, OSM objects outside the convex hull surrounding dat are plotted in this colour, otherwise they are included in the interpolation (which will generally be inaccurate for peripheral values)
size	Size argument passed to ggplot2 (polygon, path, point) functions: determines width of lines for (polygon, line), and sizes of points. Respective defaults are (0, 0.5, 0.5). If bg is provided and size has 2 elements, the second determines the size of the background objects.
shape	Shape of lines or points, for details of which see ?ggplot::shape. If bg is provided and shape has 2 elements, the second determines the shape of the background objects.

Value

modified version of map to which surface has been added

Note

Points beyond the spatial boundary of dat are included in the surface if bg is not given. In such cases, values for these points may exceed the range of provided data because the surface will be extrapolated beyond its domain. Actual plotted values are therefore restricted to the range of given values, so any extrapolated points greater or less than the range of dat are simply set to the respective maximum or minimum values. This allows the limits of dat to be used precisely when adding colourbars with add_colourbar.

See Also

osm_basemap, add_colourbar.

Examples

```
# Get some data
bbox <- get_bbox (c (-0.13, 51.5, -0.11, 51.52))
# dat_B <- extract_osm_objects (key = 'building', bbox = bbox)
# These data are also provided in
dat_B <- london$dat_BNR # actuall non-residential buildings
# Make a data surface across the map coordinates, and remove periphery
n <- 5
x <- seq (bbox [1,1], bbox [1,2], length.out = n)
y <- seq (bbox [2,1], bbox [2,2], length.out = n)
dat <- data.frame (
    x = as.vector (array (x, dim = c(n, n))),
    y = as.vector (t (array (y, dim = c(n, n)))),
    z = x * y
    )
## Not run:
map <- osm_basemap (bbox = bbox, bg = 'gray20')
map <- add_osm_surface (map, dat_B, dat = dat, cols = heat.colors (30))
print_osm_map (map)

## End(Not run)

# If data do not cover the entire map region, then the peripheral remainder
# can be plotted by specifying the 'bg' colour. First remove periphery from
# 'dat':
d <- sqrt ((dat$x - mean (dat$x)) ^ 2 + (dat$y - mean (dat$y)) ^ 2)
dat <- dat [which (d < 0.01),]
## Not run:
map <- osm_basemap (bbox = bbox, bg = 'gray20')
map <- add_osm_surface (map, dat_B, dat = dat,
                        cols = heat.colors (30), bg = 'gray40')
print_osm_map (map)

## End(Not run)

# Polygons and (lines/points) can be overlaid as data surfaces with different
# colour schemes.
# dat_HP <- extract_osm_objects (key = 'highway',
#                                value = 'primary',
#                                bbox = bbox)
# These data are also provided in
dat_HP <- london$dat_HP
cols <- adjust_colours (heat.colors (30), adj = -0.2) # darken by 20%
## Not run:
map <- add_osm_surface (map, dat_HP, dat, cols = cols,
                        bg = 'gray60', size = c(1.5,0.5))
print_osm_map (map)

## End(Not run)

# Adding multiple surfaces of either polygons or (lines/points) produces a
# 'ggplot2' warning, and forces the colour gradient to revert to the last
```

```
# given value.
dat_T <- london$dat_T # trees
## Not run:
map <- osm_basemap (bbox = bbox, bg = 'gray20')
map <- add_osm_surface (map, dat_B, dat = dat,
                        cols = heat.colors (30), bg = 'gray40')
map <- add_osm_surface (map, dat_HP, dat,
                        cols = heat.colors (30), bg = 'gray60',
                        size = c(1.5,0.5))
map <- add_osm_surface (map, dat_T, dat, cols = topo.colors (30),
                        bg = 'gray70', size = c(5,2), shape = c(8, 1))
print_osm_map (map) # 'dat_HP' is in 'topo.colors' not 'heat.colors'

## End(Not run)

# Add axes and colourbar
## Not run:
map <- add_axes (map)
map <- add_colourbar (map, cols = heat.colors (100), zlims = range (dat$z),
                      barwidth = c(0.02), barlength = c(0.6,0.99),
                      vertical = TRUE)
print_osm_map (map)

## End(Not run)
```

`adjust_colours` *adjust_colours*

Description

Adjusts a given colour by lightening or darkening it by the specified amount (relative scale of -1 to 1). Adjustments are made in RGB space, for limitations of which see `?convertColor`

Usage

```
adjust_colours(cols, adj = 0, plot = FALSE)
```

Arguments

`cols`	A vector of R colours (for allowable formats of which, see `?col2rgb`).
`adj`	A number between -1 and 1 determining how much to lighten (positive values) or darken (negative values) the colours.
`plot`	If `TRUE`, generates a plot to allow visual comparison of original and adjusted colours.

Value

Corresponding vector of adjusted colours (as hexadecimal strings).

See Also

osm_structures, ?col2rgb.

Examples

```
cols <- adjust_colours (cols = heat.colors (10), adj = -0.2, plot = TRUE)

# 'adjust_colours' also offers an easy way to adjust the default colour
# schemes provided by 'osm_structures'. The following lines darken the
# highway colour of the 'light' colour scheme by 20%
structures <- osm_structures (structures = c("building", "highway", "park"),
                              col_scheme = "light")
structures$cols [2] <- adjust_colours (structures$cols [2], adj = -0.2)
# Plot these structures:
bbox <- get_bbox (c (-0.13, 51.5, -0.11, 51.52))
## Not run:
dat_B <- extract_osm_objects (key = "building", bbox = bbox)
dat_H <- extract_osm_objects (key = "highway", bbox = bbox)
dat_P <- extract_osm_objects (key = "park", bbox = bbox)

## End(Not run)
# These data are also included in the 'london' data of 'osmplotr'
osm_data <- list (dat_B = london$dat_BNR,
                  dat_H = london$dat_HP,
                  dat_P = london$dat_P)
dat <- make_osm_map (structures = structures,
                     osm_data = osm_data,
                     bbox = bbox)
print_osm_map (dat$map)
```

---

colour_mat *colour_mat*

Description

Generates a 2D matrix of graduated colours by interpolating between the given colours specifying the four corners.

Usage

```
colour_mat(cols, n = c(10, 10), rotate, plot = FALSE)
```

Arguments

cols	vector of length >= 4 of colors (example, default = rainbow (4), or RColorBrewer::brewer.pal (4, 'Set1')). cols are wrapped clockwise around the corners from top left to bottom left.
n	number of rows and columns of colour matrix (default = 10; if length 2, then dimensions of rectangle).

rotate	rotates the entire colour matrix by the specified angle (in degrees).
plot	plots the colour matrix.

Value

Matrix of colours.

See Also

add_osm_groups.

Examples

```
cm <- colour_mat (n = 5, cols = rainbow(4), rotate = 90, plot = TRUE)

# 'colour_mat' is intended primarily for use in colouring groups added with
# 'add_osm_groups' using the 'colmat = TRUE' option:
bbox <- get_bbox (c (-0.13, 51.5, -0.11, 51.52))
# Generate random points to serve as group centres
set.seed (2)
ngroups <- 6
x <- bbox [1,1] + runif (ngroups) * diff (bbox [1,])
y <- bbox [2,1] + runif (ngroups) * diff (bbox [2,])
groups <- cbind (x, y)
groups <- apply (groups, 1, function (i)
                 sp::SpatialPoints (matrix (i, nrow = 1, ncol = 2)))
# plot a basemap and add groups
map <- osm_basemap (bbox = bbox, bg = "gray20")
map <- add_osm_groups (map, obj = london$dat_BNR, group = groups,
                       cols = rainbow (4), colmat = TRUE, rotate = 90)
print_osm_map (map)
```

---

connect_highways *connect_highways*

Description

Takes a list of highways names which must enclose an internal area, and returns a SpatialLines object containing a sequence of OSM nodes which cyclically connect all highways. Will fail if the streets do not form a cycle.

Usage

```
connect_highways(highways, bbox, plot = FALSE)
```

Arguments

highways	A vector of highway names passed directly to the Overpass API. Wildcards and whitespaces are '.'; for other options see online help for the overpass API.
bbox	the bounding box for the map. A 2-by-2 matrix of 4 elements with columns of min and max values, and rows of x and y values.
plot	If TRUE, then all OSM data for each highway is plotted and the final cycle overlaid.

Value

A single set of SpatialPoints containing the lat-lon coordinates of the cyclic line connecting all given streets.

Note

1. connect_highways is primarily intended to provide a means to define boundaries of groups which can then be highlighted using add_osm_groups.
2. This function can not be guaranteed failsafe owing both to the inherently unpredictable nature of OpenStreetMap, as well as to the unknown relationships between named highways. The plot option enables problematic cases to be examined and hopefully resolved. The function is still experimental, so please help further improvements by reporting any problems!

See Also

add_osm_groups.

Examples

```
bbox <- get_bbox (c (-0.13, 51.5, -0.11, 51.52))
## Not run:
highways <- c ("Monmouth.St", "Short.?s.Gardens", "Endell.St", "Long.Acre",
               "Upper.Saint.Martin")
# Note that dots signify "anything", including whitespace and apostrophes,
# and that '?' denotes optional previous character and so here matches
# both "Shorts Gardens" and "Short's Gardens"
highways1 <- connect_highways (highways = highways, bbox = bbox, plot = TRUE)
highways <- c ("Endell.St", "High.Holborn", "Drury.Lane", "Long.Acre")
highways2 <- connect_highways (highways = highways, bbox = bbox, plot = TRUE)

# Use of 'connect_highways' to highlight a region on a map
map <- osm_basemap (bbox = bbox, bg = "gray20")
# dat_B <- extract_osm_data (key = "building",
#                            value = "!residential",
#                            bbox = bbox)
# Those data are part of 'osmplotr':
dat_BNR <- london$dat_BNR # Non-residential buildings
groups <- list (highways1, highways2)
map <- add_osm_groups (map, obj = dat_BNR, groups = groups,
                       cols = c("red", "blue"), bg = "gray40")
print_osm_map (map)
```

```
## End(Not run)
```

`extract_osm_objects` *extract_osm_objects*

Description

Downloads OSM XML objects and converts to sp objects (`SpatialPointsDataFrame`, `SpatialLinesDataFrame`, or `SpatialPolygonsDataFrame`).

Usage

```
extract_osm_objects(
  bbox,
  key,
  value,
  extra_pairs,
  return_type,
  sf = TRUE,
  geom_only = FALSE,
  quiet = FALSE
)
```

Arguments

`bbox`	the bounding box within which all key-value objects should be downloaded. A 2-by-2 matrix of 4 elements with columns of min and max values, and rows of x and y values.
`key`	OSM key to search for. Useful keys include `building`, `waterway`, `natural`, `grass`, `park`, `amenity`, `shop`, `boundary`, and `highway`. Others will be passed directly to the overpass API and may not necessarily return results.
`value`	OSM value to match to key. If `NULL`, all keys will be returned. Negation is specified by `!value`.
`extra_pairs`	A list of additional `key-value` pairs to be passed to the overpass API.
`return_type`	If specified, force return of spatial (`point`, `line`, `polygon`, `multiline`, `multipolygon`) objects. `return_type = 'line'` will, for example, always return a SpatialLinesDataFrame. If not specified, defaults to 'sensible' values (for example, `lines` for highways, `points` for trees, `polygons` for buildings).
`sf`	If `TRUE`, return Simple Features (`sf`) objects; otherwise Spatial (`sp`) objects.
`geom_only`	If `TRUE`, return only those OSM data describing the geometric object; otherwise return all data describing each object.
`quiet`	If `FALSE`, provides notification of progress.

Value

Either a SpatialPointsDataFrame, SpatialLinesDataFrame, or SpatialPolygonsDataFrame.

See Also

add_osm_objects.

Examples

```
## Not run:
bbox <- get_bbox (c(-0.13,51.50,-0.11,51.52))
dat_B <- extract_osm_objects (key = 'building', bbox = bbox)
dat_H <- extract_osm_objects (key = 'highway', bbox = bbox)
dat_BR <- extract_osm_objects (key = 'building',
                               value = 'residential',
                               bbox = bbox)
dat_HP <- extract_osm_objects (key = 'highway',
                               value = 'primary',
                               bbox = bbox)
dat_HNP <- extract_osm_objects (key = 'highway',
                                value = '!primary',
                                bbox = bbox)
extra_pairs <- c ('name', 'Royal.Festival.Hall')
dat <- extract_osm_objects (key = 'building', extra_pairs = extra_pairs,
                            bbox = bbox)

## End(Not run)
```

---

get_bbox *get_bbox*

Description

Converts a string of latitudes and longitudes into a square matrix to be passed as a bbox argument (to extract_osm_objects, osm_basemap, or make_osm_map).

Usage

```
get_bbox(latlon)
```

Arguments

latlon	A vector of (longitude, latitude, longitude, latitude) values.

Value

A 2-by-2 matrix of 4 elements with columns of min and max values, and rows of x and y values.

Examples

```
bbox <- get_bbox (c (-0.15, 51.5, -0.1, 51.52))
```

---

london *london*

Description

A list of Simple Features (sf) data.frame objects containing OpenStreetMap polygons, lines, and points for various OpenStreetMap structures in a small part of central London, U.K. (bbox = -0.13, 51.51, -0.11, 51.52). The list includes:

1. dat_H: 974 non-primary highways as linestrings
2. dat_HP: 159 primary highways as linestrings
3. dat_BNR: 1,716 non-residential buildings as polygons
4. dat_BR: 43 residential buildings as polygons
5. dat_BC: 67 commerical buildings as polygons
6. dat_A: 372 amenities as polygons
7. dat_P: 13 parks as polygons
8. dat_T: 688 trees as points
9. dat_RFH: 1 polygon representing Royal Festival Hall
10. dat_ST: 1 polygon representing 150 Stamford Street

Format

A list of spatial objects

Details

The vignette basic-maps details how these data were downloaded. Note that these internal versions have had all descriptive data removed other than their names, geometries, and their OSM identification numbers.

make_osm_map *make_osm_map*

Description

Makes an entire OSM map for the given bbox using the submitted data, or by downloading data if none submitted. This is a convenience function enabling an entire map to be produced according to the graphical format specified with the `structures` argument.

Usage

```
make_osm_map(
  bbox,
  osm_data,
  structures = osm_structures(),
  dat_prefix = "dat_"
)
```

Arguments

`bbox`	The bounding box for the map. A 2-by-2 matrix of 4 elements with columns of min and max values, and rows of x and y values. If `NULL`, `bbox` is taken from the largest extent of OSM objects in `osm_data`.
`osm_data`	A list of OSM objects as returned from `extract_osm_objects`. These objects may be included in the plot without downloading. These should all be named with the stated `dat_prefix` and have suffixes as given in `structures`.
`structures`	A `data.frame` specifying types of OSM structures as returned from `osm_structures`, and potentially modified to alter lists of structures to be plotted, and their associated colours. Objects are overlaid on plot according to the order given in `structures`.
`dat_prefix`	Prefix for data structures (default `dat_`). Final data structures are created by appending the suffixes from `osm_structures`.

Value

List of two components:

1. List of OSM structures each as `Spatial(Points/Lines/Polygons)DataFrame` and appended to `osm_data` (which is `NULL` by default), and
2. The `map` as a `ggplot2` object

Note

If `osm_data` is not given, then data will be downloaded, which can take some time. Progress is dumped to screen.

See Also

osm_basemap, add_osm_objects.

Examples

```
structures <- c ("highway", "park")
structs <- osm_structures (structures = structures, col_scheme = "light")
# make_osm_map returns potentially modified list of data using the provided
# 'london' data for highways and parks.
dat <- make_osm_map (osm_data = london, structures = structs)
# or download data automatically using a defined bounding boox
bbox <- get_bbox (c(-0.15,51.5,-0.10,51.52))
## Not run:
dat <- make_osm_map (bbox = bbox, structures = structs)
print_osm_map (dat$map)

## End(Not run)
```

---

osmplotr *osmplotr.*

Description

Produces customisable images of OpenStreetMap (OSM) data and enables data visualisation using OSM objects. Extracts data using the overpass API. Contains the following functions, data, and vignettes.

Data Functions

- extract_osm_objects: Download arbitrary OSM objects
- connect_highways: Returns points sequentially connecting list of named highways

Basic Plotting Functions (without data)

- add_axes: Overlay longitudinal and latitudinal axes on plot
- add_osm_objects: Overlay arbitrary OSM objects
- make_osm_map: Automate map production with structures defined in osm_structures
- osm_structures: Define structures and graphics schemes for automating map production
- osm_basemap: Initiate a ggplot2 object for an OSM map
- print_osm_map: Print a map to specified graphics device

Advanced Plotting Functions (with data)

- add_osm_groups: Overlay groups of objects using specified colour scheme
- add_osm_surface: Overlay data surface by interpolating given data
- add_colourbar: Overlay a scaled colourbar for data added with add_osm_surface

Colour Manipulation Functions

- adjust_colours: Lighted or darken given colours by specified amount
- colour_mat: Generate continuous 2D spatial matrix of colours

Other Functions

- get_bbox: return bounding box from input vector

Data

- london: OSM Data from a small portion of central London

Vignettes

- basic-maps: Describes basics of downloading data and making custom maps
- data-maps: Describes how map elements can be coloured according to user-provided data, whether categorical or continuous

---

osm_basemap *osm_basemap*

Description

Generates a base OSM plot ready for polygon, line, and point objects to be overlain with add_osm_objects.

Usage

```
osm_basemap(bbox, structures, bg = "gray20")
```

Arguments

bbox	bounding box (Latitude-longitude range) to be plotted. A 2-by-2 matrix of 4 elements with columns of min and max values, and rows of x and y values. Can also be an object of class sf, for example as returned from extract_osm_objects or the osmdata package, in which case the bounding box will be extracted from the object coordinates.
structures	Data frame returned by osm_structures used here to specify background colour of plot; if missing, the colour is specified by bg.
bg	Background colour of map (default = gray20) only if structs not given).

Value

A ggplot2 object containing the base map.

See Also

add_osm_objects, make_osm_map.

Examples

```
bbox <- get_bbox (c (-0.13, 51.5, -0.11, 51.52))
map <- osm_basemap (bbox = bbox, bg = "gray20")
map <- add_osm_objects (map, london$dat_BNR, col = "gray40")
print_osm_map (map)
```

`osm_line2poly` *osm_line2poly*

Description

Converts `sf::sfc_LINSTRING` objects to polygons by connecting end points around the given bounding box. This is particularly useful for plotting water and land delineated by coastlines. Coastlines in OpenStreetMap are lines, not polygons, and so there is no directly way to plot ocean water distinct from land. This function enables that by connecting the end points of coastline `LINESTRING` objects to form closed polygons.

Usage

```
osm_line2poly(obj, bbox)
```

Arguments

`obj`	A Simple Features (`sf`) data frame of lines, typically as returned by `extract_osm_objects`, or by `osmdata::osmdata_sf`.
`bbox`	bounding box (Latitude-longitude range) to be plotted. A 2-by-2 matrix of 4 elements with columns of min and max values, and rows of x and y values. Can also be an object of class `sf`, for example as returned from `extract_osm_objects` or the `osmdata` package, in which case the bounding box will be extracted from the object coordinates.

Details

This is a tricky problem for a number of reasons, and the current implementation may not be correct, although it does successfully deal with a few tough situations. Some of the issues are: an osm coastline query returns a mixture of "ways" and polygons.

Polygons correspond to islands, but not all islands are polygons. A "way" is a connected set of points with the land on the left. A piece of coastline in a bounding box may consist of multiple ways, which need to be connected together to create a polygon. Also, ways extend outside the query bounding box, and may join other ways that enter the bounding box (e.g ends of a peninsula). The degree to which this happens depends on the scale of the bounding box. Coastlines may enter at any bounding box edge and exit at any other, including the one they entered from.

Value

A list of three Simple Features (`sf`) data frames, labelled sea islands and land.

Examples

```
# This example uses the \code{osmdata} package to extract data from
# a named bounding box
## Not run:
library (magrittr)
library (osmdata)
bb <- osmdata::getbb ("melbourne, australia")
coast <- extract_osm_objects (bbox = bb,
                              key = "natural",
                              value = "coastline",
                              return_type = "line")
coast <- osm_line2poly (coast, bbox = bb)
# The following map then colours in just the ocean:
map <- osm_basemap (bbox = bb) %>%
    add_osm_objects (coast$sea, col = "lightsteelblue") %>%
    print_osm_map ()

## End(Not run)
```

`osm_structures` *osm_structures*

Description

For the given vector of structure types returns a `data.frame` containing two columns of corresponding OpenStreetMap `key-value` pairs, one column of unambiguous suffixes to be appended to the objects returned by `extract_osm_objects`, and one column specifying colours. This `data.frame` may be subsequently modified as desired, and ultimately passed to `make_osm_map` to automate map production.

Usage

```
osm_structures(
  structures = c("building", "amenity", "waterway", "grass", "natural", "park",
    "highway", "boundary", "tree"),
  col_scheme = "dark"
)
```

Arguments

`structures`	The vector of types of structures (defaults listed in `extract_osm_objects`).
`col_scheme`	Colour scheme for the plot (current options include `dark` and `light`).

Value

`data.frame` of structures, `key-value` pairs, corresponding prefixes, and colours.

See Also

make_osm_map.

Examples

```
# Default structures:
osm_structures ()
# user-defined structures:
structures <- c ("highway", "park", "ameniiy", "tree")
structs <- osm_structures (structures = structures, col_scheme = "light")
# make_osm_map returns potentially modified list of data
## Not run:
dat <- make_osm_map (osm_data = london, structures = structs)
# map contains updated $osm_data and actual map in $map
print_osm_map (dat$map)

## End(Not run)
```

---

print_osm_map *print_osm_map*

Description

Prints an OSM map produced with osmplotr to a specified graphics device.

Usage

```
print_osm_map(
  map,
  width,
  height,
  filename,
  device,
  units = c("in", "cm", "mm", "px"),
  dpi = 300
)
```

Arguments

map	The map to be printed; a ggplot2 object produced by osmplotr.
width	Desired width of graphics device.
height	Desired height of graphics device. Ignored if width specified.
filename	Name of file to which map is to be printed.
device	Type of graphics device (extracted from filename extension if not explicitly provided).
units	Units for height and width of graphics device.
dpi	Resolution of graphics device (dots-per-inch).

See Also

osm_basemap, add_osm_objects, make_osm_map.

Examples

```
bbox <- get_bbox (c (-0.13, 51.5, -0.11, 51.52))
map <- osm_basemap (bbox = bbox, bg = "gray20")
map <- add_osm_objects (map, london$dat_BNR, col = "gray40")
print_osm_map (map, width = 7) # prints to screen device
## Not run:
print_osm_map (map, file = "map.png", width = 500, units = "px")

## End(Not run)
```

Index