Package: mapscanner (via r-universe)

November 22, 2024

Title Print Maps, Draw on Them, Scan Them Back in

Version 0.1.1.001

Description Enables preparation of maps to be printed and drawn on. Modified maps can then be scanned back in, and hand-drawn marks converted to spatial objects.

License GPL-3 | BSD_2_clause + file LICENSE

URL https://github.com/ropensci/mapscanner

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

Depends R (>= 3.5.0)

Imports cli, curl, fs, glue, magick, magrittr, memoise, pdftools, png, purrr, raster, Rcpp, reproj, RNiftyReg, sf, slippymath, tibble

Suggests dplyr, ggplot2, gibble, jpeg, knitr, lwgeom, mapview, mmand, osmdata, polyclip, rmarkdown, spelling, testthat

LinkingTo Rcpp

VignetteBuilder knitr

Encoding UTF-8

Language en-GB

LazyData true

NeedsCompilation yes

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Config/pak/sysreqs libgdal-dev gdal-bin libgeos-dev make libpng-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev

Repository https://ropensci.r-universe.dev

RemoteUrl https://github.com/ropensci/mapscanner

RemoteRef main

RemoteSha 517ad17b981acff309df1b9bcd79351f486739cf

Contents

`ms_aggregate_polys` *Aggregate disparate polygons*

Description

Planar partition from disparate polygon inputs. Overlaps aggregate to `n`.

Usage

```
ms_aggregate_polys(p)
```

Arguments

`p`	input (multi-)polygons (assumed to be overlapping)

Details

Input is a single simple features polygon data frame. No attribute data is considered.

Value

Set of **sf**-format polygons with additional column, `n`, denoting number of overlaps contributing to each of the resultant polygons.

Examples

```
g <- sf::st_sfc (list (
    sf::st_point (cbind (0, 0)),
    sf::st_point (cbind (0, 1)),
    sf::st_point (cbind (1, 0))
))
pts <- sf::st_sf (a = 1:3, geometry = g)
overlapping_polys <- sf::st_buffer (pts, 0.75)

## decompose and count space-filling from overlapping polygons
x <- ms_aggregate_polys (overlapping_polys)
plot (x)
## Not run:
library (ggplot2)
ggplot (x) +
```

```
    geom_sf () +
    facet_wrap (~n)

## End(Not run)

library (sf)
set.seed (6)
pts <- expand.grid (x = 1:8, y = 1:10) %>% st_as_sf (coords = c ("x", "y"))
xsf <- sf::st_buffer (pts, runif (nrow (pts), 0.2, 1.5))
## Not run:
out <- ms_aggregate_polys (xsf)

## End(Not run)
```

`ms_generate_map` *Generate maps for 'mapscanner' use*

Description

Generate a map image for a specified area or bounding box, by downloading tiles from https://www.mapbox.com/. Map is automatically saved in both `.pdf` and `.png` formats, by default in current working directory, or alternative location when `mapname` includes the full path.

Usage

```
ms_generate_map(
  bbox,
  max_tiles = 16L,
  mapname = NULL,
  bw = TRUE,
  style = "light",
  raster_brick = NULL
)
```

Arguments

`bbox`	Either a string specifying the location, or a numeric bounding box as a single vector of (xmin, ymin, xmax, ymax), or a 2-by-2 matrix with columns of (min, max) and rows of (x, y), respectively.
`max_tiles`	Maximum number of tiles to use to create map
`mapname`	Name of map to be produced, optionally including full path. Extension will be ignored.
`bw`	If `FALSE`, print maps in colour, otherwise black-and-white. Note that the default `style = "light"` is monochrome, and that this parameter only has effect for `style` values of `"streets"` or `"outdoors"`.
`style`	The style of the map to be generated; one of 'light', 'streets', or 'outdoors', rendered in black and white. See https://docs.mapbox.com/api/maps/#styles/ for examples.

raster_brick Instead of automatically downloading tiles within a given bbox, a pre-downloaded raster::rasterBrick object may be submitted and used to generate the .pdf and .png equivalents.

Value

Invisibly returns a rasterBrick object from the **raster** package containing all data used to generate the map.

Examples

```
## Not run:
# code used to generate internal files for a portion of Omaha:
bb <- osmdata::getbb ("omaha nebraska")
shrink <- 0.3 # shrink that bb to 30% size
bb <- t (apply (bb, 1, function (i) {
    mean (i) + c (-shrink, shrink) * diff (i) / 2
}))
ms_generate_map (bb, max_tiles = 16L, mapname = "omaha")

## End(Not run)
```

---

ms_rectify_map *Rectify one map to another*

Description

Rectify two previously scanned-in pdf or png maps with RNiftyReg, and return the modifications in map_modified as spatial objects in **sf** format.

Usage

```
ms_rectify_map(
  map_original,
  map_modified,
  nitems = NULL,
  non_linear = 1,
  type = "hulls",
  downsample = 10,
  concavity = 0,
  length_threshold = 10,
  quiet = FALSE
)
```

Arguments

map_original	File name of the original map without anything drawn over it (either a .pdf or .png; extension will be ignored).
map_modified	File name of the modified version with drawings (either a .pdf or .png; extension will be ignored).
nitems	Optional parameter to explicitly specify number of distinct items to be extracted from map; if possible, specifying this parameter may improve results.
non_linear	Integer value of 0, 1, or 2 representing degree of non-linearity in modified image - see Note.
type	Currently either "points", "polygons", or "hulls", where "points" simply reduces each distinct object to a single, central point; "polygons" identifies individual groups and returns the polygon representing the outer boundary of each; and "hulls" constructs convex or concave polygons around each group.
downsample	Factor by which to downsample type = "polygons", noting that polygons initially include every outer pixel of image, so can generally be downsampled by at least an order or magnitude (n = 10). Higher values may be used for higher-resolution images; lower values will generally only be necessary for very low lower resolution images.
concavity	For type = "hulls", a value between 0 and 1, with 0 giving convex hulls and 1 giving highly concave hulls.
length_threshold	For type = "hulls", the minimal length of a segment to be made more convex. Low values will produce highly detailed hulls which may cause problems; if in doubt, or if odd results appear, increase this value.
quiet	If FALSE, display progress information on screen

Value

An **sf** object representing the drawn additions to map_modified.

Note

The non-linear parameter should generally set according to how the modified maps were digitised. A value of 0 will give fastest results, and should be used for directly scanned or photocopied images. A value of 1 (the default) still presumes modified images have been linearly translated, and will apply affine transformations (rotations, contractions, dilations). This value should be used when modified images have been photographed (potentially from an oblique angle). A value of 2 should only be used when modified maps have somehow been non-linearly distorted, for example through having been crumpled or screwed up. Rectification with non-linear = 2 will likely take considerably longer than with lower values.

Examples

```
f_orig <- system.file ("extdata", "omaha.png", package = "mapscanner")
f_mod <- system.file ("extdata", "omaha-polygons.png",
    package = "mapscanner"
)
```

```
# reduce file sizes to 1/4 to speed up these examples:
f_orig2 <- file.path (tempdir (), "omaha.png")
f_modified2 <- file.path (tempdir (), "omaha-polygons.png")
magick::image_read (f_orig) %>%
    magick::image_resize ("25%") %>%
    magick::image_write (f_orig2)
magick::image_read (f_mod) %>%
    magick::image_resize ("25%") %>%
    magick::image_write (f_modified2)

# then rectify those files:
## Not run:
xy_hull <- ms_rectify_map (f_orig2, f_modified2, type = "hull")
xy_poly <- ms_rectify_map (f_orig2, f_modified2, type = "polygons")
xy_pts <- ms_rectify_map (f_orig2, f_modified2, type = "points")

## End(Not run)
```

`ms_rotate_map` *Rotate maps*

Description

Display original and modified maps to determine necessary rotation

Usage

```
ms_rotate_map(map_original, map_modified, rotation = 0, apply_rotation = FALSE)
```

Arguments

`map_original`	File name of the original map without anything drawn over it (either a `.pdf` or `.png`; extension will be ignored).
`map_modified`	File name of the modified version with drawings (either a `.pdf` or `.png`; extension will be ignored).
`rotation`	Rotation value to be applied, generally +/- 90
`apply_rotation`	If `FALSE`, display results of rotation without actually applying it; otherwise transform the specified `map_modified` image according to the specified rotation.

Value

No return value. Function either modifies files on disk by rotating images by the specified amount (if `apply_rotation = TRUE`), or displays a rotated version of `map_original` (if `apply_rotation = FALSE`).

Note

If a call to ms_rectify_map detects potential image rotation, that function will stop and suggest that rotation be applied using this function in order to determine the required degree of image rotation. Values for `rotation` can be trialled in order to determine the correct value, following which that value can be entered with `apply_rotation = TRUE` in order to actually apply that rotation to the modified image.

`set_mapbox_token` *Set 'mapbox' token*

Description

Set a mapbox token for use with the ms_generate_map function.

Usage

```
set_mapbox_token(token)
```

Arguments

`token`	Personal mapbox API token, obtained from https://docs.mapbox.com/api/#access-tokens-and-token-scopes.

Value

`TRUE` if the token was able to be set; otherwise `FALSE`.

Index