Package ‘OtsuFire’

July 21, 2025

Type Package

Title Fire Scars, Severity and Regeneration Mapping Using 'Otsu' Thresholding

Version 0.1.4

Description Tools to segment fire scars and assess severity and vegetation regeneration using 'Otsu' thresholding on Relative Burn Ratio (RBR) and differenced Normalized Burn Ratio (dNBR) image composites.
Includes support for mosaic handling, polygon metrics, post-fire regeneration detection, day-of-year flagging,
and validation against reference datasets. Designed for analysis of fire history in the Iberian Peninsula.
Input Landsat composites follow the methodology described in Quintero et al. (2025) <doi:10.2139/ssrn.4929831>.

License GPL-3

Encoding UTF-8

URL `https://github.com/olgaviedma/OtsuFire`

BugReports `https://github.com/olgaviedma/OtsuFire/issues`

Depends R (>= 4.1.0)

Imports data.table, dplyr, gdalUtilities, glue, purrr, raster, sf, stringr, terra, magrittr, tidyr, rlang, OtsuSeg

Suggests testthat (>= 3.0.0)

RoxygenNote 7.3.2

Config/testthat/edition 3

NeedsCompilation no

Author Natalia Quintero [aut],
Olga Viedma [aut, cre],
Hammadi Achour [ctb],
Jose Manuel Moreno [ctb]

Maintainer Olga Viedma <`olga.viedma@uclm.es`>

Repository CRAN

Date/Publication 2025-06-14 09:00:06 UTC

Contents

`calculate_doy_flags` *Calculate DOY-Based Flags and Summary Statistics from Raster and Fire Polygons*

Description

This function extracts Day-of-Year (DOY) values from a raster (e.g., from a Landsat composite), masked by burned area polygons, and computes per-polygon statistics: mode, median, and optionally percentiles (e.g., 5th, 25th, 95th). These DOY values are converted to calendar dates, and their components (day, month, year) are also included.

Optionally, the function can threshold the DOY band around the median and/or mode of each polygon using user-defined DOY windows (e.g., \pm10 days), create binary raster masks, and vectorize them using GDAL.

Use this function to assign a representative burning date (DOY) to each fire event and to discard events with unusually high internal variation in DOY among their pixels.

The function supports either a single shapefile (original burned areas) or a named list of shapefiles or ‘sf‘ objects generated with different Otsu thresholds.

Arguments

`raster`	A multi-band ‘SpatRaster‘ object. One of the bands must contain DOY values.
`doy_band`	Integer. The index of the DOY band in the raster (default is 2).
`polygons_sf`	A single shapefile path, an ‘sf‘ object, or a named list of shapefiles or ‘sf‘ objects. Names are used to label outputs (e.g., ‘"ge50"‘, ‘"ge100"‘).
`output_dir`	Directory where output raster and shapefile files will be saved.
`year`	Calendar year (numeric or character) used for DOY-to-date conversion and filenames.
`doy_thresholds`	Numeric vector of DOY windows (e.g., ‘c(10, 15)‘) to apply around the selected statistic(s).
`stats`	Statistic(s) to use for thresholding. One of ‘"median"‘, ‘"mode"‘, or ‘"both"‘ (default ‘"both"‘).

`calc_percentiles`	Logical. If ‘TRUE‘, compute additional DOY percentiles. Default is ‘TRUE‘.
`percentiles`	Numeric vector of percentiles to calculate (default: ‘c(0.05, 0.25, 0.95)‘).
`polygonize`	Logical. If ‘TRUE‘, vectorize the DOY binary masks using GDAL (‘gdal_polygonize.py‘). Default is ‘TRUE‘.
`python_exe`	Path to the Python executable. Required if ‘polygonize = TRUE‘.
`gdal_polygonize_script`	Path to ‘gdal_polygonize.py‘. Required if ‘polygonize = TRUE‘.
`get_mode`	Optional custom function to calculate the mode. Default function provided.
`get_median`	Optional custom function to calculate the median. Default function provided.
`get_quantile`	Optional custom function to calculate percentiles. Default function provided.
`keep_all_polygons`	Logical. If ‘TRUE‘, all polygons are kept and a flag column is added indicating which polygons meet the threshold condition. If ‘FALSE‘, only polygons meeting the condition are saved. Default: ‘TRUE‘.

Value

A named list with two elements:

sf_objects Named list of ‘sf‘ outputs containing DOY statistics and date components for each input polygon set.

shp_paths Named list of output shapefile paths, including DOY summary shapefiles and any polygonized threshold layers.

Note

Examples require large external raster files (hosted on Zenodo) and depend on external software (Python, GDAL). Therefore, they are wrapped in dontrun to avoid errors during R CMD check and to ensure portability.

The DOY raster band should be derived from a Landsat composite generated in Google Earth Engine (GEE), following the approach described by Quintero et al. (2025).

References

Quintero, N., Viedma, O., Veraverbeke, S., & Moreno, J. M. (2025). *Optimising Regional Fire Severity Mapping Using Pixel-Based Image Compositing*. SSRN 4929831.

Examples

```
## Not run:
# Case 1: Multiple Otsu-based burned area shapefiles
rast <- terra::rast("rbr_doy_stack_1987.tif")
polys <- list(
  ge50 = "burned_areas_1987_otsu_ge50.shp",
  ge100 = "burned_areas_1987_otsu_ge100.shp"
)
calculate_doy_flags(
  raster = rast,
  doy_band = 2,
  polygons_sf = polys,
  output_dir = "output/",
  year = 1987,
  stats = "both",
  doy_thresholds = c(10, 15),
  calc_percentiles = TRUE,
  percentiles = c(0.05, 0.25, 0.95),
  polygonize = TRUE,
  python_exe = "/usr/bin/python",
  gdal_polygonize_script = "/usr/bin/gdal_polygonize.py"
)

# Case 2: Single shapefile without polygonization
calculate_doy_flags(
  raster = rast,
  polygons_sf = "burned_areas_1987_origval.shp",
  output_dir = "output/",
  year = 1987,
  stats = "median",
  doy_thresholds = 10,
  calc_percentiles = FALSE,
  polygonize = FALSE
)

## End(Not run)
```

`calculate_polygon_metrics`
Calculate Geometric Metrics and Filter Fire Polygons (Batch Mode)

Description

Calculates geometric descriptors for fire-affected polygons from a list of input shapefiles. Metrics include area, perimeter, bounding box dimensions, rotated bounding box, and shape ratios. Optionally applies spatial filters and performs attribute joins to enrich outputs.

For each input shapefile, the function: - Computes metrics per polygon and saves a new shapefile. - Applies spatial filters and saves a filtered version. - Optionally joins metrics back to the original input polygons (if ‘join_attributes = TRUE‘).

If ‘dissolve = TRUE‘, adjacent polygons are merged to reconstruct contiguous burned areas. A numeric ‘burned_id‘ is then assigned to each resulting polygon and used in subsequent joins.

## Metrics computed per polygon: - ‘area_ha‘: Area in hectares. - ‘bbox_wx‘: Width of axis-aligned bounding box (x). - ‘bbox_hy‘: Height of axis-aligned bounding box (y). - ‘perim_m‘: Perimeter in meters. - ‘p_w_ratio‘: Perimeter-to-width ratio. - ‘h_w_ratio‘: Height-to-width ratio. - ‘burned_id‘: Polygon ID if ‘dissolve = TRUE‘.

## Filter behavior: All thresholds are optional. If a threshold is ‘NULL‘, the corresponding filter is skipped. If ‘filter_logic = "AND"‘, all active filters must be satisfied. If ‘"OR"‘, any filter match is sufficient.

## Spatial join (‘join_attributes = TRUE‘): - Attributes from the original input shapefile are preserved via intersection-based joins. - Only input polygons with area ? ‘min_input_area_ha‘ are used. - The join selects, for each input polygon, the intersecting output polygon with the largest shared area. - Only polygons that intersect filtered outputs are retained in the joined outputs. - Optionally, you can restrict which variables from the original shapefile to retain using ‘columns_to_keep‘.

Arguments

`shapefile_paths`	Character vector. Paths to input shapefiles.
`output_dir`	Optional. Directory to save outputs. Defaults to the input file’s folder.
`area_min_ha`	Minimum area in hectares (‘area_ha‘). Default: 10.
`bbox_h_min`	Minimum height of axis-aligned bounding box (‘bbox_hy‘). Default: 630.
`mnbbx_wd_min`	Minimum width of rotated bounding box. Default: 800.
`p_w_ratio_min`	Minimum perimeter-to-width ratio. Default: 4.0.
`h_w_ratio_min`	Minimum height-to-width ratio. Default: 0.35.
`output_format`	Output format for saved files: ‘"shp"‘ or ‘"geojson"‘. Default: ‘"shp"‘.
`filter_logic`	Logical combination of filters: ‘"AND"‘ (default) or ‘"OR"‘.
`dissolve`	Logical. If ‘TRUE‘, dissolve adjacent polygons before computing metrics. Default: TRUE.
`join_attributes`	Logical. If ‘TRUE‘, joins filtered/dissolved polygons back to the original shapefile. Default: TRUE.
`min_input_area_ha`	Minimum area (in hectares) for input polygons used in spatial joins. Default: 10.
`overlay_polygons_path`	Optional. ‘sf‘ object or path to shapefile to be used for join instead of the input shapefile. Default: ‘NULL‘.
`columns_to_keep`	Optional. Character vector of variable names to retain from the original shapefile during spatial join. Default: ‘NULL‘ (keep all).

Value

A named list (one element per input file), each containing:

metrics Path to shapefile with all polygons and computed metrics.

filtered Path to shapefile with only polygons passing the filters.

joined_metrics (If ‘join_attributes = TRUE‘) Path to joined shapefile with original attributes + metrics.

joined_filtered (If ‘join_attributes = TRUE‘) Path to joined filtered shapefile.

polygons_all ‘sf‘ object of all polygons with computed metrics.

polygons_filtered ‘sf‘ object of polygons that passed the filters.

Note

Examples require large external raster files (hosted on Zenodo). Therefore, they are wrapped#' in dontrun to avoid errors during R CMD check and to ensure portability.

Examples

```
## Not run:
burned_files <- list.files("path/to/shapefiles", pattern = "\\.shp$", full.names = TRUE)

results <- calculate_polygon_metrics(
  shapefile_paths = burned_files,
  output_dir = "outputs/",
  area_min_ha = 10,
  bbox_h_min = 600,
  mnbbx_wd_min = 800,
  p_w_ratio_min = 4.0,
  h_w_ratio_min = 0.25,
  output_format = "geojson",
  filter_logic = "AND",
  dissolve = TRUE,
  join_attributes = TRUE,
  min_input_area_ha = 5,
  columns_to_keep = c("CLC_CODE", "ECOR_REGION")
)

## End(Not run)
```

`flag_otsu_regenera` *Flag Burned Polygons Based on Regeneration Overlap*

Description

This function assigns regeneration flags to burned area polygons based on their spatial overlap with post-fire regeneration polygons (e.g., from years 1, 2, or more after fire). A polygon is flagged as ‘"regenera"‘ if the total intersected area with regeneration polygons from a specific post-fire year exceeds a user-defined minimum threshold (‘min_regen_ratio‘) relative to its own area.

Filenames of the regeneration layers must contain ‘"P1"‘, ‘"P2"‘, etc., to indicate the number of years after fire. These period labels are automatically extracted and used to name the output columns (e.g., ‘regen_ratio_P1‘, ‘regenera_flag_P1‘).

For each detected regeneration year, the function adds:

- ‘regen_ratio_Px‘: proportion of the burned polygon area overlapping with regeneration polygons.
- ‘regenera_flag_Px‘: ‘"regenera"‘ if the threshold is met, ‘"no_regenera"‘ otherwise.

A final column ‘regenera_flag_all‘ is also created, indicating ‘"regenera"‘ only if all selected years in ‘remove_condition‘ meet their respective thresholds.

If ‘replace_by_P1 = TRUE‘, the function will optionally replace burned polygons in the filtered output by overlapping P1 regeneration polygons that are larger in area. Optionally, a maximum area ratio (‘max_area_ratio_p1‘) can be defined to restrict replacements to only those P1 polygons not exceeding a given multiple of the burned area.

You can also choose to export polygons flagged as ‘"no_regenera"‘ using ‘save_no_regenera‘. These can be optionally filtered by a minimum area threshold (‘min_area_no_regenera‘).

If ‘validate_geometries = TRUE‘, geometries are checked and corrected using ‘sf::st_make_valid()‘, which is more robust but significantly slower. Set to ‘FALSE‘ by default for performance.

Arguments

`burned_files`	Character vector with paths to burned area shapefiles (.shp).
`regenera_files`	Character vector with paths to regeneration shapefiles (.shp). Filenames must contain ‘"P1"‘, ‘"P2"‘, etc., to indicate the year after fire.
`min_regen_ratio`	Named numeric vector with minimum area ratio thresholds (between 0 and 1) for each regeneration year. Names must correspond to period labels (e.g., ‘c(P1 = 0.05, P2 = 0.10)‘). These values are used: (1) when no classwise_regen_thresholds are provided, (2) or as fallback when a polygon has no match or NA thresholds.
`remove_no_regenera`	Logical. If ‘TRUE‘, polygons that do not meet all specified thresholds (see ‘remove_condition‘) will be excluded from the filtered output.
`remove_condition`	Character vector indicating the post-fire years (e.g., ‘"P1"‘, ‘"P2"‘, ‘"P3"‘) that must meet their thresholds in ‘min_regen_ratio‘ in order for a polygon to be retained. Used to identify false fire detections due to lack of regeneration.
`output_dir`	Optional output directory. If ‘NULL‘, output files are saved in the same folder as the burned shapefiles.
`replace_by_P1`	Logical. If ‘TRUE‘ (default), filtered burned polygons will be replaced by overlapping P1 regeneration polygons when the latter are larger in area.
`max_area_ratio_p1`	Optional. Maximum allowed ratio between the area of a P1 regeneration polygon and the intersected burned polygon. If specified, only regenerated polygons with area greater than the burned area and not exceeding `max_area_ratio_p1` times that area will be used to replace burned polygons. If `NULL` (default), only the condition `area_ha` > `burn_area_ha` is applied.
`save_no_regenera`	Character. Options are ‘"all"‘ (save all ‘"no_regenera"‘ polygons), ‘"area_filter"‘ (save only those with area ? ‘min_area_no_regenera‘), or ‘"none"‘ (do not save them).
`min_area_no_regenera`	Numeric. Minimum area in hectares to retain a ‘"no_regenera"‘ polygon when ‘save_no_regenera = "area_filter"‘. Default is ‘0.1‘ ha.
`validate_geometries`	Logical. If ‘TRUE‘, applies ‘st_make_valid()‘ to correct invalid geometries before processing. Default is ‘FALSE‘ for faster performance.

`output_format` Character. Output format for saved files: ‘"shp"‘ for ESRI Shapefile (default) or ‘"geojson"‘ for GeoJSON. @param classwise_regen_thresholds Optional ‘data.frame‘ with per-class regeneration thresholds. Must include:

- `class_field`: field name in the burned polygons (e.g., ‘"CORINE_CLA"‘, ‘"eco_id"‘).
- `class_value`: corresponding class value.
- One or more columns named `P1`, `P2`, etc., with numeric thresholds.

If provided, these thresholds will override `min_regen_ratio` for polygons matching each class. Polygons without a match will fall back to the values in `min_regen_ratio`.

Value

Saves up to four shapefiles per burned area file:

Main shapefile (unfiltered) All polygons with ‘regen_ratio_Px‘, ‘regenera_flag_Px‘, and ‘regenera_flag_all‘.

Filtered shapefile Only polygons that satisfy all thresholds defined in ‘remove_condition‘. File ends in ‘_filter.shp‘.

Final shapefile with replacements (if ‘replace_by_P1 = TRUE‘) Filtered polygons with some replaced by larger P1 regeneration polygons. File ends in ‘_new.shp‘.

Shapefile of no_regenera polygons (if ‘save_no_regenera != "none"‘) Contains polygons not meeting regeneration thresholds. File ends in ‘_no_selected.shp‘.

Note

Examples require large external raster files (hosted on Zenodo). Therefore, they are wrapped in dontrun to avoid errors during R CMD check and to ensure portability.

Examples

```
## Not run:
# Apply per-period regeneration thresholds and filter non-regenerating polygons
burned_files <- list.files("data/burned", pattern = "\\.shp$", full.names = TRUE)
regenera_files <- list.files("data/regenera", pattern = "\\.shp$", full.names = TRUE)

flag_otsu_regenera(
  burned_files = burned_files,
  regenera_files = regenera_files,
  min_regen_ratio = c(P1 = 0.05, P2 = 0.10),
  remove_no_regenera = TRUE,
  remove_condition = c("P1", "P2"),
  output_dir = "results/",
  replace_by_P1 = TRUE,
  max_area_ratio_p1 = 3,
  save_no_regenera = "area_filter",
  min_area_no_regenera = 100,
  validate_geometries = FALSE,
  output_format = c("geojson")
)
```

```
# Define class-specific regeneration thresholds
class_thresholds <- data.frame(
  class_field = "CORINE_CLA",
  class_value = c(1, 2),
  P1 = c(0.15, 0.15),
  P2 = c(0.0, 0.05)
)

# Run flag_otsu_regenera1 using classwise thresholds and fallback values
result <- flag_otsu_regenera(
  burned_files = burned_list,
  regenera_files = regenera_list,
  min_regen_ratio = c(P1 = 0.05, P2 = 0.15),  # fallback for classes not in the table
  classwise_regen_thresholds = class_thresholds,
  group_field = "CORINE_CLA",
  remove_no_regenera = TRUE,
  remove_condition = c("P1", "P2"),
  replace_by_P1 = TRUE,
  max_area_ratio_p1 = 3,
  save_no_regenera = TRUE,
  min_area_no_regenera = 500,
  validate_geometries = FALSE,
  output_format = "shp",
  output_dir = burned_dir
)

## End(Not run)
```

`generate_burnable_mask` *Generate Burnable Mask from CORINE Rasters*

Description

This function reclassifies CORINE land cover rasters into binary burnable masks using a predefined set of vegetation-related land cover classes. It masks the rasters using an Iberian Peninsula shapefile, optionally reprojects them to EPSG:3035 and/or EPSG:4326, and polygonizes the result using GDAL tools. The raster masks can be used in GEE for masking the Landsat composites.

The reclassification uses the following CORINE classes (corine_code / description): - 244: Agroforestry areas (22) - 311: Broad-leaved forest (23) - 312: Coniferous forest (24) - 313: Mixed forest (25) - 321: Natural grasslands (26) - 322: Moors and heathland (27) - 323: Sclerophyllous vegetation (28) - 324: Transitional woodland-shrub (29) - 333: Sparsely vegetated areas (32) - 334: Burnt areas (33)

These classes are assigned a gridcode from 22 to 33 in the final output shapefile. The .csv file with gridcode and CORINE classes correspondences is provided in the README folder (clc_legend_gridcode.csv) (if ‘vectorize = TRUE‘).

Usage

```
generate_burnable_mask(
  corine_rasters,
  peninsula_shapefile,
  output_raster_dir,
  output_vector_dir,
  gdalwarp_path,
  python_exe,
  gdal_polygonize_script,
  reproject = TRUE,
  res = 90,
  to_wgs84 = TRUE,
  vectorize = TRUE
)
```

Arguments

`corine_rasters`	Named list of CORINE raster file paths (e.g., one per year).
`peninsula_shapefile`	Path to the Iberian Peninsula shapefile used to mask the extent.
`output_raster_dir`	Directory where binary and class rasters will be written.
`output_vector_dir`	Directory where vector shapefiles will be saved (if ‘vectorize = TRUE‘).
`gdalwarp_path`	Full path to the GDAL ‘gdalwarp‘ executable.
`python_exe`	Full path to the Python executable.
`gdal_polygonize_script`	Full path to ‘gdal_polygonize.py‘.
`reproject`	Logical. Whether to reproject rasters to EPSG:3035. Default is TRUE.
`res`	Numeric. Output resolution in meters (used when reprojecting). Default is 90.
`to_wgs84`	Logical. If TRUE, generates an additional raster version reprojected to EPSG:4326.
`vectorize`	Logical. Whether to convert the reclassified raster to polygon shapefile. Default is TRUE.

Value

No R object is returned, but the following files are written: - Binary raster mask (‘burneable_mask_binary_<year>_ETRS89.ti - Optional EPSG:4326 version (‘..._wgs84.tif‘) - Multiclass raster with selected CORINE codes (‘burneable_classes_def1_<year>_ETRS89.tif‘) - Polygon shapefile with selected CORINE classes (‘burneable_classes_def1_<year>_ETRS89.shp‘)

Note

Examples require large external raster files (hosted on Zenodo) and depend on external software (Python, GDAL). Therefore, they are wrapped in dontrun to avoid errors during R CMD check and to ensure portability.

Examples

```
## Not run:
corine_rasters <- list(
  "corine06" = "data/CORINE_2006.tif"
)

generate_burnable_mask(
  corine_rasters = corine_rasters,
  peninsula_shapefile = "data/peninsula_ib.shp",
  output_raster_dir = "output/rasters",
  output_vector_dir = "output/vectors",
  gdalwarp_path = "C:/ProgramData/anaconda3/Library/bin/gdalwarp.exe",
  python_exe = "C:/ProgramData/anaconda3/python.exe",
  gdal_polygonize_script =
  "C:/ProgramData/anaconda3/Lib/site-packages/osgeo/scripts/gdal_polygonize.py",
  reproject = TRUE,
  res = 90,
  to_wgs84 = TRUE,
  vectorize = TRUE
)

## End(Not run)
```

---

mask_to_mosaic *Apply a Binary Raster Mask (and Optional Geographic Clip) to a Two-Band Mosaic*

Description

This function applies a binary raster mask to a 2-band burned area mosaic (RBR and DOY). All pixels in the mosaic where the mask does not match ‘valid_mask_value‘ are assigned ‘NA‘. Optionally, a shapefile can be provided to crop and mask the mosaic to a specific geographic area (e.g., the Iberian Peninsula).

The function ensures that the mask raster is projected and aligned to the mosaic before applying. The output is saved with the same resolution as the input mosaic.

Usage

```
mask_to_mosaic(
  mosaic_path,
  mask_raster_path,
  output_path = NULL,
  valid_mask_value = 1,
  crs_target = 3035,
  shapefile_clip = NULL
)
```

Arguments

`mosaic_path`	Character. Path to the input 2-band raster mosaic (e.g., with RBR and DOY).
`mask_raster_path`	Character. Path to the binary mask raster (e.g., burnable areas).
`output_path`	Character (optional). Output destination. There are three behaviors: • If 'NULL', the output is saved in the same folder as the input mosaic, using the same base name with suffix '_mosaic_masked_res90m.tif'. • If a directory path is provided, the output file will be saved inside that folder, using the same base name as the input mosaic, with suffix '_mosaic_masked_res90m.tif'. • If a full file path ending in '.tif' is provided, that name is used directly as the output.
`valid_mask_value`	Numeric. Value in the mask that defines valid pixels (default is 1).
`crs_target`	Integer or character. EPSG code for reprojection (only used if needed). Default: 3035.
`shapefile_clip`	Character (optional). Path to a shapefile to further crop and mask the mosaic.

Details

This function is typically used after mosaicking burned area products derived from satellite imagery. It filters areas outside a burnable mask (e.g., based on land cover) and optionally restricts output to a defined geographic region (e.g., the Iberian Peninsula).

Input mosaic must have: - Band 1: Relative Burn Ratio (RBR) - Band 2: Day of Year (DOY)

Output raster is written in GeoTIFF format with LZW compression.

Value

Character. Full path to the final masked (and optionally clipped) raster.

Note

Examples require large external raster files (hosted on Zenodo). Therefore, they are wrapped in dontrun to avoid errors during R CMD check and to ensure portability.

Examples

```
## Not run:
# Case 1: Let the function generate the output name in the same folder
mask_to_mosaic(
  mosaic_path = "data/IBERIAN_MinMin_all_year_2012_mosaic_res90m.tif",
  mask_raster_path = "data/burneable_mask_corine_ETRS89.tif"
)

# Case 2: Save the output to a specific folder, name generated automatically
mask_to_mosaic(
  mosaic_path = "data/IBERIAN_MinMin_all_year_2012_mosaic_res90m.tif",
  mask_raster_path = "data/burneable_mask_corine_ETRS89.tif",
  output_path = "outputs/"
)

# Case 3: Define the full output file name explicitly
mask_to_mosaic(
  mosaic_path = "data/IBERIAN_MinMin_all_year_2012_mosaic_res90m.tif",
  mask_raster_path = "data/burneable_mask_corine_ETRS89.tif",
  output_path = "outputs/custom_masked_output_2012.tif"
)

## End(Not run)
```

---

mosaic_reproject_resample

Mosaic, Reproject, and Resample Burned Area Tiles

Description

This function creates a seamless mosaic from multiple raster tiles (e.g., RBR and DOY composites), masks them with a shapefile (e.g., Iberian Peninsula), reprojects the mosaic to a specified CRS (default: EPSG:3035), and resamples it to a target resolution (default: 90m).

It assumes the input rasters are two-band GeoTIFFs: - Band 1: RBR (Relative Burn Ratio) - Band 2: DOY (Day of Year of fire)

Usage

```
mosaic_reproject_resample(
  folder_path,
  mask_path,
  year = NULL,
  raster_pattern = "IBERIAN_MinMin_all_year_*.tif",
  crs_target = "EPSG:3035",
  res_target = 90,
  nodata_value = -9999
)
```

Arguments

`folder_path`	Character. Path to the folder containing the raster tiles.
`mask_path`	Character. Path to a polygon shapefile used to mask the rasters (e.g., burnable area mask).
`year`	Integer or character. Year label used for naming output files. If NULL, uses folder name.
`raster_pattern`	Character. Regex pattern to identify input raster tiles. Default: 'IBERIAN_MinMin_all_year_*.tif'.
`crs_target`	Character. EPSG code string for the target CRS to reproject (default: "EPSG:3035").

`res_target`	Numeric. Target spatial resolution in meters. Default: 90.
`nodata_value`	Numeric. NoData value to assign to missing values. Default: -9999.

Details

The function uses ‘gdalbuildvrt‘ and ‘gdalwarp‘ for efficient VRT-based mosaicking and reprojection. Steps: 1. Apply spatial mask to each raster tile. 2. Create VRT mosaic. 3. Reproject and resample the mosaic with ‘gdalwarp‘. 4. Clean NoData values and assign band names (’rbr’, ’doy’). 5. Save compressed GeoTIFF output.

Value

Character. Full path to the final resampled GeoTIFF mosaic.

Note

Examples require large external raster files (hosted on Zenodo) and depend on external software (Python, GDAL). Therefore, they are wrapped in dontrun to avoid errors during R CMD check and to ensure portability.

Examples

```
## Not run:
mosaic_reproject_resample(
  folder_path = "ZENODO/exdata",
  mask_path = "ZENODO/exdata/iberian_peninsula_proj_final.shp",
  year = 2012,
  raster_pattern = "IBERIAN_MinMin_all_year_2012_*.tif",
  crs_target = "EPSG:3035",
  res_target = 90
)

## End(Not run)
```

`process_otsu_rasters` *Process Burned Area Rasters Using Otsu Thresholding or Percentile Clipping*

Description

This function computes binary burned area masks from a severity index raster (RBR or dNBR) using Otsu’s thresholding or percentile clipping. The segmentation can be applied to the full raster or stratified by: - CORINE land cover classes (‘corine_raster_path‘) - WWF ecoregions (‘ecoregion_shapefile_path‘) - or the intersection of both (CORINE ? Ecoregion)

## Key features: - Supports Otsu thresholding after pre-filtering by minimum index values (‘otsu_thresholds‘) or using original values - Supports percentile-based thresholding via ‘trim_percentiles‘ - Enforces

a minimum threshold (‘min_otsu_threshold_value‘) when Otsu yields low values - Allows stratification by CORINE, ecoregions, or their intersection - Enables CORINE reclassification using a custom ‘reclass_matrix‘ - Automatically computes RBR or dNBR if ‘nbr_pre_path‘ and ‘nbr_post_path‘ are provided - Outputs: - Burned area binary rasters - Polygon shapefiles (dissolved by threshold label) - Histogram and inter-class variance plots per threshold - Threshold log files

Arguments

`raster_path`	Path to a single-band RBR or dNBR raster.
`nbr_pre_path, nbr_post_path`	Optional. Paths to pre- and post-fire NBR rasters for RBR/dNBR calculation.
`output_dir`	Directory to save output files.
`year`	Optional year label.
`otsu_thresholds`	Numeric vector of minimum values to filter raster before applying Otsu.
`trim_percentiles`	Optional data.frame with ‘min‘ and ‘max‘ columns to define percentile clipping thresholds.
`use_original`	Logical. Use raw values without filtering (ignored if ‘trim_percentiles‘ is set).
`corine_raster_path`	Path to CORINE raster for stratified segmentation.
`reclassify_corine`	Logical. If TRUE, reclassifies CORINE using ‘reclass_matrix‘.
`reclass_matrix`	Matrix with original and new values for CORINE reclassification.
`peninsula_shapefile`	Shapefile used to crop CORINE before reclassification.
`output_corine_raster_dir`	Directory to save reclassified CORINE raster.
`output_corine_vector_dir`	Directory to save reclassified CORINE shapefile.
`reproject`	Logical. Reproject reclassified CORINE raster to EPSG:3035.
`resolution`	Numeric. Target resolution for reprojected CORINE raster.
`corine_classes`	Optional vector of reclassified CORINE classes to keep.
`ecoregion_shapefile_path`	Path to WWF ecoregions shapefile.
`ecoregion_field`	Column in ecoregion shapefile used for class labels.
`ecoregion_classes`	Optional. Vector of selected ecoregion class names.
`segment_by_intersection`	Logical. If TRUE, intersects CORINE and ecoregions.
`min_otsu_threshold_value`	Minimum acceptable threshold. Used if Otsu value is lower.

`python_exe`	Path to Python executable.
`gdal_polygonize_script`	Path to ‘gdal_polygonize.py‘ script.
`gdalwarp_path`	Path to ‘gdalwarp‘ executable (default = "gdalwarp").
`n_rows`, `n_cols`	Number of rows/columns for raster tiling.
`tile_overlap`	Overlap buffer (in map units) for tiles.
`tile`	Logical. Apply raster tiling before polygonization.
`index_type`	"RBR" or "dNBR". Used if ‘nbr_pre_path‘/‘nbr_post_path‘ are provided.
`output_format`	Output vector file format: "shp" or "geojson".

Details

- Raster values are internally rescaled to [0, 255] before Otsu thresholding. - Histogram smoothing and variance curves are used for enhanced threshold detection. - Output shapefiles can be in ESRI Shapefile or GeoJSON format. - Intermediate tile shapefiles and rasters are cleaned after processing.

Value

Named list of ‘sf‘ polygons grouped by segmentation. Output files saved to ‘output_dir‘.

Note

Examples require large external raster files (hosted on Zenodo) and depend on external software (Python, GDAL). Therefore, they are wrapped in dontrun to avoid errors during R CMD check and to ensure portability.

Examples

```
## Not run:
# Example 1: Basic Otsu thresholds (global)
process_otsu_rasters(
  raster_path = "data/rbr_1985.tif",
  output_dir = "output/1985",
  otsu_thresholds = c(0, 50, 100),
  python_exe = "C:/Python/python.exe",
  gdal_polygonize_script = "C:/Python/Scripts/gdal_polygonize.py"
)

# Example 2: Global with percentile trimming
process_otsu_rasters(
  raster_path = "data/rbr_1985.tif",
  output_dir = "output/1985",
  trim_percentiles = data.frame(min = 0.01, max = 0.99),
  python_exe = "C:/Python/python.exe",
  gdal_polygonize_script = "C:/Python/Scripts/gdal_polygonize.py",
  output_format = "geojson"
)

# Example 3: Stratified by reclassified CORINE
```

```
process_otsu_rasters(
  raster_path = "data/rbr_1985.tif",
  output_dir = "output/1985",
  corine_raster_path = "data/corine_1990_etrs89.tif",
  reclassify_corine = TRUE,
  reclass_matrix = matrix(c(
    22, 1,  # grasslands ? 1
    26, 1,
    32, 1,
    27, 2,  # shrublands ? 2
    29, 2,
    33, 3,  # burnt areas ? 3
    23, 4,  # broadleaved forest ? 4
    25, 5,  # mixed forest ? 5
    24, 6   # coniferous ? 6
  ), ncol = 2, byrow = TRUE),
  corine_classes = c(1, 2, 4, 5, 6),
  peninsula_shapefile = "data/peninsula_clip.shp",
  output_corine_raster_dir = "output/corine",
  output_corine_vector_dir = "output/corine",
  otsu_thresholds = c(50),
  min_otsu_threshold_value = 150,
  python_exe = "C:/Python/python.exe",
  gdal_polygonize_script = "C:/Python/Scripts/gdal_polygonize.py"
)

# Example 4: Stratified by WWF ecoregions (default field: "EnS_name")
process_otsu_rasters(
  raster_path = "data/rbr_1985.tif",
  output_dir = "output/1985",
  ecoregion_shapefile_path = "data/ecoregions_olson.shp",
  otsu_thresholds = c(50),
  min_otsu_threshold_value = 150,
  python_exe = "C:/Python/python.exe",
  gdal_polygonize_script = "C:/Python/Scripts/gdal_polygonize.py"
)

# Example 5: Stratified by WWF ecoregions with custom field
process_otsu_rasters(
  raster_path = "data/rbr_1985.tif",
  output_dir = "output/1985",
  ecoregion_shapefile_path = "data/ecoregions_olson.shp",
  ecoregion_field = "EnZ_name",
  ecoregion_classes = c("Iberian Conifer Forests", "Mediterranean Shrublands"),
  otsu_thresholds = c(50),
  min_otsu_threshold_value = 150,
  python_exe = "C:/Python/python.exe",
  gdal_polygonize_script = "C:/Python/Scripts/gdal_polygonize.py"
)

# Example 6: Stratified by CORINE ? Ecoregion intersection
process_otsu_rasters(
  raster_path = "data/rbr_1985.tif",
  output_dir = "output/1985",
  corine_raster_path = "data/corine_1990_etrs89.tif",
  reclassify_corine = TRUE,
  reclass_matrix = matrix(c(
    22, 1, 26, 1, 32, 1,  # grasslands
    27, 2, 29, 2,         # shrublands
    23, 4, 25, 5, 24, 6   # forests
  ), ncol = 2, byrow = TRUE),
  corine_classes = c(1, 2, 4, 5, 6),
  peninsula_shapefile = "data/peninsula_clip.shp",
  output_corine_raster_dir = "output/corine",
  output_corine_vector_dir = "output/corine",
  ecoregion_shapefile_path = "data/ecoregions_olson.shp",
  ecoregion_field = "EnZ_name",
  segment_by_intersection = TRUE,
  otsu_thresholds = c(50),
  min_otsu_threshold_value = 150,
  python_exe = "C:/Python/python.exe",
  gdal_polygonize_script = "C:/Python/Scripts/gdal_polygonize.py"
)

## End(Not run)
```

process_otsu_regenera *Detect Post-Fire Regeneration Using Otsu or Fixed Thresholds*

Description

This function detects vegetation regeneration signals using negative RBR or dNBR values. By default (‘use_fixed_threshold = FALSE‘), the function identifies regeneration areas where RBR or dNBR values are below 0. If ‘use_fixed_threshold = TRUE‘, a fixed threshold value (e.g., -100) is applied instead.

Optionally, percentile-based clipping can be applied before thresholding, and tiling can be used to handle large rasters. The function outputs binary rasters and shapefiles, optionally merged into a single shapefile if ‘bind_all = TRUE‘.

If a historical burn raster (‘rbr_date‘) is provided, it will be used to mask out areas previously burned (i.e., where RBR \ge 0), so that regeneration is only detected in those affected by the most recent fire.

Arguments

`rbr_post`	Path to post-fire RBR or dNBR raster. Alternatively, provide ‘nbr_pre_path‘ and ‘nbr_post_path‘. If a named list is provided, names should match the format ‘"P1"‘, ‘"P2"‘, etc.
`nbr_pre_path`	Path to pre-fire NBR raster (for index computation).
`nbr_post_path`	Path to post-fire NBR raster (for index computation).
`rbr_date`	Optional path to an RBR raster used to mask previously burned areas (RBR \ge 0).

`output_dir`	Directory where outputs (rasters, shapefiles, plots) will be saved.
`python_exe`	Path to the Python executable (used to call GDAL).
`gdal_polygonize_script`	Path to ‘gdal_polygonize.py‘ script.
`n_rows, n_cols`	Number of rows and columns for tiling. Ignored if ‘tile = FALSE‘.
`tile_overlap`	Overlap size (in meters) between tiles. Used for seamless polygonization.
`tile`	Logical. If ‘TRUE‘, raster is tiled before polygonization.
`index_type`	Index type to compute: either ‘"RBR"‘ or ‘"dNBR"‘. Ignored if ‘rbr_post‘ is provided.
`trim_percentiles`	Data frame with columns ‘min‘ and ‘max‘, defining percentile ranges to clip negative RBR values before thresholding. **Note:** This parameter is currently not used unless percentile-based thresholding is implemented.
`bind_all`	Logical. If ‘TRUE‘, all shapefiles from each threshold are merged into one combined shapefile.
`regen_year`	Integer vector specifying the number of years after the fire to consider for regeneration (e.g., ‘c(2)‘).
`fire_year`	Integer indicating the fire year. Used for output naming and labeling.
`use_fixed_threshold`	Logical. If ‘TRUE‘, applies ‘fixed_threshold_value‘ instead of using the default threshold (< 0).
`fixed_threshold_value`	Numeric threshold to use when ‘use_fixed_threshold = TRUE‘. Default is ‘-100‘.
`output_format`	Character. Output format for saved files: ‘"shp"‘ for ESRI Shapefile (default) or ‘"geojson"‘ for GeoJSON.

Value

A named list of results per regeneration year (‘P1‘, ‘P2‘, etc.) with:

‘raster‘ Path to the binary regeneration raster for that year.

‘shapefile‘ Path to the shapefile with polygons derived from the raster tiles for that year.

If ‘bind_all = TRUE‘, an additional item named ‘combined‘ is returned, which contains the path to a single shapefile:

`‘combined‘`	Path to the shapefile with all valid regeneration polygons combined across all years. The filename includes all ‘P‘ labels used (e.g., ‘_P1P2_‘).

Note

Examples require large external raster files (hosted on Zenodo) and depend on external software (Python, GDAL). Therefore, they are wrapped in dontrun to avoid errors during R CMD check and to ensure portability.

Examples

```
## Not run:
# Regeneration detection using RBR raster and default threshold (< 0)
process_otsu_regenera(
  rbr_post = list(P2 = "data/RBR_1986.tif"),
  output_dir = "output/regenera",
  python_exe = "/usr/bin/python3",
  gdal_polygonize_script = "/usr/bin/gdal_polygonize.py",
  fire_year = 1984,
  regen_year = c(2),
  use_fixed_threshold = FALSE,
  output_format = c("geojson")
)

# Same detection but using a fixed threshold of -150
process_otsu_regenera(
  rbr_post = list(P2 = "data/RBR_1986.tif"),
  output_dir = "output/regenera",
  python_exe = "/usr/bin/python3",
  gdal_polygonize_script = "/usr/bin/gdal_polygonize.py",
  fire_year = 1984,
  regen_year = c(2),
  use_fixed_threshold = TRUE,
  fixed_threshold_value = -150,
  bind_all = FALSE,
  n_rows = 2,
  n_cols = 3,
  tile_overlap = 1000,
  tile = TRUE,
  output_format = c("shp")
)

## End(Not run)
```

`validate_fire_maps` *Validate burned area maps against reference polygons*

Description

The ‘validate_fire_maps()‘ function evaluates the spatial accuracy of burned area detection shapefiles by comparing them against independent reference fire polygons (e.g., from Focclim or national databases).

It calculates both **pixel-based metrics** and **area-based metrics**, unless a specific mode is selected. Reference polygons are masked to retain only burnable areas based on a CORINE-derived raster and filtered by a minimum area threshold if specified.

Important: If ‘min_area_reference_ha‘ is changed, you must set ‘force_reprocess_ref = TRUE‘ to recalculate the masked reference polygons.

Arguments

`input_shapefile`	Character vector. One or more paths to shapefiles containing the burned polygons to validate.
`ref_shapefile`	Character. Path to the shapefile with reference burned area polygons.
`mask_shapefile`	Character. Path to the shapefile defining the study area boundary.
`burnable_raster`	Character. Path to the raster file indicating burnable areas (binary or categorical).
`year_target`	Numeric. Target year for filtering reference polygons.
`validation_dir`	Character. Output directory to save validation results.
`binary_burnable`	Logical. TRUE if the burnable raster is binary (1 = burnable, 0 = non-burnable). Default is TRUE.
`burnable_classes`	Optional numeric vector of raster values considered burnable if ‘binary_burnable = FALSE‘.
`class_shape`	Optional character. Path to a shapefile containing class information (e.g., CORINE or ecoregions) used for class-wise error breakdown.
`class_field`	Optional character. Name of the field in ‘class_shape‘ to group omission and commission errors by class (e.g., "CORINE", "eco_name", "cor_eco").
`buffer`	Numeric. Optional buffer distance (in meters) applied around reference polygons for pixel-based validation. Default is 0.
`threshold_completely_detected`	Numeric. Minimum percentage (e.g., 90) of a reference polygon area that must be intersected to be considered completely detected. Default is 90.
`min_area_reference_ha`	Numeric. Minimum area (in hectares) to retain reference polygons after masking. Default is NULL (no filtering).
`use_gdal`	Logical. Whether to use external GDAL (via Python) for polygonizing rasters (faster for large datasets). Default is TRUE.
`python_exe`	Character. Path to the Python executable used for GDAL polygonization.
`gdal_polygonize_script`	Character. Path to the ‘gdal_polygonize.py‘ script.
`force_reprocess_ref`	Logical. If TRUE, forces recalculation of masked reference polygons even if cached versions exist. Default is FALSE.
`metrics_type`	Character. Type of metrics to compute. One of ‘"all"‘ (default), ‘"pixel"‘, or ‘"area"‘.

Details

## Pixel-based metrics (when ‘metrics_type = "pixel"‘ or ‘"all"‘): - **True Positives (TP)**: Pixels correctly detected as burned. - **False Positives (FP)**: Pixels wrongly detected as burned. -

False Negatives (FN): Burned pixels missed by the detection. - **True Negatives (TN)**: Pixels correctly identified as unburned.

Derived indicators: - **Precision** = TP / (TP + FP) - **Recall** = TP / (TP + FN) - **F1 Score** = 2 * (Precision * Recall) / (Precision + Recall) - **Intersection over Union (IoU)** = TP / (TP + FP + FN) - **Error Rate** = (FP + FN) / (TP + FP + FN + TN)

## Area-based metrics (when ‘metrics_type = "area"‘ or ‘"all"‘): - **N_Reference_Polygons**: Number of reference polygons after masking and filtering. - **N_Completely_Detected**: Count of reference polygons where detected area ? ‘threshold_completely_detected‘. - **N_Detected_Polygons**: Reference polygons partially or fully detected (>0 - **N_Not_Detected**: Reference polygons without any detected overlap. - **Perc_Detected_Polygons**: Share of reference polygons detected. - **Area_Reference_ha**: Total area of reference polygons (ha). - **Area_Detected_ha**: Total area of detected burned polygons (ha). - **Area_Intersection_ha**: Area of intersection between detection and reference polygons (ha). - **Area_Reference_NotDetected_ha**: Area of reference polygons not intersected (ha). - **Perc_Reference_Area_NotDetected**: Share of reference area missed (- **Recall_Area_percent** = (Area_Intersection_ha / Area_Reference_ha) * 100 - **Precision_Area_percent** = (Area_Intersection_ha / Area_Detected_ha) * 100

## Class-wise error breakdown: If ‘class_shape‘ and ‘class_field‘ are provided and valid: - ‘ref_not_detected‘ and ‘det_not_matched‘ polygons are joined to ‘class_shape‘. - Output CSV files: - ‘omission_by_<class_field>_<input>.csv‘ - ‘commission_by_<class_field>_<input>.csv‘

## Output files: - ‘metrics_summary_<year>.csv‘ and/or ‘polygon_summary_<year>.csv‘ - Shapefiles of undetected and unmatched polygons - Class-wise omission/commission CSVs (if class information is provided)

Value

A list with:

metrics A data.table of pixel-based accuracy metrics, or NULL if ‘metrics_type‘ excludes them.

polygon_summary A data.table of area-based metrics, or NULL if ‘metrics_type‘ excludes them.

Note

Examples require large external raster files (hosted on Zenodo) and depend on external software (Python, GDAL). Therefore, they are wrapped in dontrun to avoid errors during R CMD check and to ensure portability.

Examples

```
## Not run:
validate_fire_maps(
  input_shapefile = list.files("shapefiles", pattern = "\\.shp$", full.names = TRUE),
  ref_shapefile = "ref_polygons_2022.shp",
  mask_shapefile = "mask_region.shp",
  burnable_raster = "burnable.tif",
  year_target = 2022,
  validation_dir = "validation_results",
  binary_burnable = TRUE,
  min_area_reference_ha = 1,
  buffer = 30,
  threshold_completely_detected = 90,
  use_gdal = TRUE,
  python_exe = "C:/Python/python.exe",
  gdal_polygonize_script = "C:/Python/Scripts/gdal_polygonize.py",
  force_reprocess_ref = TRUE,
  metrics_type = "all",
  class_shape = "corine_eco_regions.shp",
  class_field = "cor_eco"
)

## End(Not run)
```

Index