Package ‘OtsuSeg’

July 21, 2025

Type Package

Title Raster Thresholding Using Otsu´s Algorithm

Version 0.1.0

Description Provides tools to process raster data and apply Otsu-based thresholding for burned area mapping and other image segmentation tasks. Implements the method described by Otsu (1979) <doi:10.1109/TSMC.1979.4310076>, a data-driven technique that determines an optimal threshold by maximizing the inter-class variance of pixel intensities. It includes validation functions to assess segmentation accuracy against reference data using standard accuracy metrics such as precision, recall, and F1-score.

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

Encoding UTF-8

License GPL-3

Depends R (>= 3.0.0)

Imports raster, zoo, sf

Suggests testthat (>= 3.0.0), curl

RoxygenNote 7.3.2

NeedsCompilation no

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

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

Config/testthat/edition 3

Author Hammadi, Achour [aut],
Olga Viedma [ctb, cre],
Zina Soltani [ctb],
Imene Habibi [ctb],
Wahbi Jaouadi [ctb]

Repository CRAN

Date/Publication 2025-05-12 08:20:13 UTC

Contents

`binarize_raster` *Binarize a Raster Using Otsu's Thresholding (with Inter-Class and Intra-Class Variance)*

Description

This function computes deltaNBR (difference between post-fire and pre-fire NBR), rescales it, applies a smoothed histogram, and uses Otsu's thresholding to create a binary raster representing burn scars. It also generates and saves plots of the smoothed histogram, inter-class variance curve, and the inter-class and intra-class variance curves on separate plots.

Usage

```
binarize_raster(
  x,
  y,
  output_shapefile = TRUE,
  shapefile_path = "binary_raster.shp"
)
```

Arguments

`x`	RasterLayer. A raster layer object representing pre-fire NBR (e.g., 'raster::RasterLayer').
`y`	RasterLayer. A raster layer object representing post-fire NBR (e.g., 'raster::RasterLayer').
`output_shapefile`	Logical. If TRUE, saves the binary raster as a shapefile. Default is TRUE.
`shapefile_path`	Character. Path to save the shapefile. Used only if output_shapefile is TRUE.

Value

A list containing:

`best_threshold`	Numeric. The computed Otsu threshold value.
`area_hectares`	Numeric. The estimated burned area in hectares.
`binary_raster_smoothed`	RasterLayer. The binary raster created using the Otsu threshold.
`binary_shapefile`	sf object. The binary shapefile created, if output_shapefile is TRUE.
`shapefile_path`	Character. Path where the shapefile was saved, if output_shapefile is TRUE.

Examples

```
#For CRAN checks, a temporary directory is used to avoid leaving files.
#For permanent use, specify a path like "results/binary_raster.shp"
  pre_fire <- get_external_data("NBRpre.tif", load = TRUE)
  post_fire <- get_external_data("NBRpost.tif", load = TRUE)
  shapefile_path <- file.path(tempdir(), "binary_raster.shp")
  result <- binarize_raster(pre_fire, post_fire, shapefile_path = shapefile_path)
  print(result$area_hectares)
  # Clean up (optional)
 unlink(list.files(tempdir(), pattern = "binary_raster\\.(shp|shx|dbf|prj)$", full.names = TRUE))
```

`get_external_data` *Download and load example data for OtsuSeg*

Description

Downloads a ZIP file with example rasters and shapefiles from GitHub releases and optionally loads a specific file.

Usage

```
get_external_data(filename = NULL, path = tempdir(), load = FALSE)
```

Arguments

`filename`	Optional. The name of the file to return (after extraction).
`path`	Local directory to download and extract files. Default is a temp folder.
`load`	Logical. If TRUE, returns a loaded object (RasterLayer for .tif, sf for .shp). Default is FALSE.

Value

File path or loaded object if ‘load = TRUE‘.

Examples

```
# Download example data and list contents
data_dir <- get_external_data(path = tempdir())
list.files(data_dir)

# Load a specific raster file (.tif)
pre_fire <- get_external_data("NBRpre.tif", path = tempdir(), load = TRUE)
print(pre_fire)

# Load a specific shapefile (.shp)
shape_data <- get_external_data("shapefile_reference.shp", path = tempdir(), load = TRUE)
print(shape_data)
```

`Quality_control` *Quantitative Comparison of Binary Shape with Reference Shape*

Description

This function calculates various metrics (e.g., Precision, Recall, F1 Score) to quantitatively compare a binary raster (in shapefile format) with a reference vector shape. It provides insights into how well the detected (binary) shape matches the reference shape.

Usage

```
Quality_control(binary_shape, reference_shape, metrics = NULL)
```

Arguments

`binary_shape`	A shapefile (sf object) representing the binary raster (e.g., burn scars).
`reference_shape`	A shapefile (sf object) representing the reference vector shape (e.g., actual burn scars).
`metrics`	A vector of metric names to calculate. If NULL, all available metrics are computed. Available metrics are: - "Precision": The proportion of the detected shape that correctly overlaps the reference. - "Recall": The proportion of the reference shape that is correctly detected. - "F1_Score": The harmonic mean of Precision and Recall, balancing both. - "IoU" (Intersection over Union): The ratio of the intersection area to the union area. - "OS" (Omission Error): The proportion of the reference shape that was missed (1 - Recall). - "US" (Commission Error): The proportion of the detected shape that is false (1 - Precision). - "E" (Overall Error): The combined error of omission and commission. - "SimSize" (Size Similarity): The relative similarity in size between the two shapes. - "Loc" (Location Error): The Euclidean distance between the centroids of the two shapes. - "AFI" (Area Fit Index): The ratio of the intersection area to the difference between total areas. Default is NULL, which computes all metrics.

Value

A data frame containing the computed metrics and their values.

Examples

```
library(sf)
  # Create a simple binary shape (square)
  binary_shape <- st_as_sf(st_sfc(st_polygon(list(rbind(
    c(0, 0), c(1, 0), c(1, 1), c(0, 1), c(0, 0)
  )))))

  # Create a reference shape (slightly shifted square)
  reference_shape <- st_as_sf(st_sfc(st_polygon(list(rbind(
    c(0.1, 0.1), c(1.1, 0.1), c(1.1, 1.1), c(0.1, 1.1), c(0.1, 0.1)
  )))))

  # Apply Quality Control
  result <- Quality_control(binary_shape, reference_shape)
  print(result)
```

Index