Package: lakeRS (via r-universe)

June 26, 2026

Title Remote Sensing Lake Data

Version 0.1.0

Description Download and process ESA Sentinel-2 L2A data to obtain trophic state information or other satellite based indexes.

License MIT + file LICENSE

URL https://github.com/KWB-R/lakeRS

BugReports https://github.com/KWB-R/lakeRS/issues

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Suggests covr,

Imports tidyr, geosphere, htmltools, htmlwidgets, leaflet, openeo, terra

Depends R (>= 4.0.0)

LazyData true

LazyDataCompression bzip2

Config/pak/sysreqs libabsl-dev cmake libgdal-dev gdal-bin libgeos-dev make libicu-dev libpng-dev libuv1-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev

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

Date/Publication 2026-05-10 17:16:29 UTC

RemoteUrl https://github.com/KWB-R/lakeRS

RemoteRef HEAD

RemoteSha c47de2bfe199c96942de3437a5abc8018fecda18

Contents

add_lakeID *Add lake name and identifier to a lake index result*

Description

Adds a human-readable lake name and a stable lake identifier to a list created by `seasonal_index_per_lake()`. If no identifier is supplied, an identifier is generated from the mean x and y coordinates of the lake grid.

Usage

```
add_lakeID(lakeIndex, lakeName = NULL, lakeID = NULL)
```

Arguments

`lakeIndex`	A list containing lake-level index results and at least the coordinate vectors `x` and `y`. Typically the list is produced internally by `seasonal_index_per_lake()`.
`lakeName`	Optional character scalar with the lake name. If `NULL`, the output element `Name` is set to `NA`.
`lakeID`	Optional character scalar with a unique lake identifier. If `NULL`, the identifier is generated as `"<mean(x)>_<mean(y)>"`.

Value

The input list with two additional elements: `Name` and `ID`.

bbox_from_point *Create a bounding box around a point*

Description

Calculates the north, east, south, and west limits of a square-like bounding box around a point in WGS 84 coordinates. The limits are computed by moving a fixed geodesic distance from the point in the four cardinal directions.

Usage

```
bbox_from_point(lat, lon, meters = 20)
```

Arguments

`lat`	Numeric scalar. Latitude of the center point in WGS 84 (`EPSG:4326`).
`lon`	Numeric scalar. Longitude of the center point in WGS 84 (`EPSG:4326`).
`meters`	Numeric scalar greater than zero. Distance in meters from the point to each side of the bounding box. Default is `20`.

Value

A named numeric vector with elements `north`, `east`, `south`, and `west`, suitable for `start_openEO_job()` or `display_geometry()`.

`bbox_from_rectangle` *Create a buffered bounding box around a rectangle*

Description

Expands a latitude-longitude rectangle by a geodesic buffer distance and returns the resulting north, east, south, and west limits.

Usage

```
bbox_from_rectangle(top_lat, bottom_lat, left_lon, right_lon, meters = 20)
```

Arguments

`top_lat`	Numeric scalar. Northern latitude of the rectangle in WGS 84 (`EPSG:4326`). Must be greater than `bottom_lat`.
`bottom_lat`	Numeric scalar. Southern latitude of the rectangle in WGS 84 (`EPSG:4326`).
`left_lon`	Numeric scalar. Western longitude of the rectangle in WGS 84 (`EPSG:4326`). Must be smaller than `right_lon`.
`right_lon`	Numeric scalar. Eastern longitude of the rectangle in WGS 84 (`EPSG:4326`).
`meters`	Numeric scalar greater than zero. Buffer distance in meters. Default is `20`.

Value

A named numeric vector with elements `north`, `east`, `south`, and `west`.

`best_nk` *Estimate a suitable number of k-means clusters*

Description

Runs k-means clustering for increasing numbers of clusters and recommends the largest number that still satisfies a minimum cluster-size criterion. The decision is based on the proportion of variance explained by the clustering.

Usage

```
best_nk(
  pixelDynamic,
  kMax = 10,
  minimum_clusterSize = 0.01,
  correlate_first = FALSE
)
```

Arguments

pixelDynamic	A list or data-frame-like object of pixel-level annual dynamics, usually pixelDynamics from dynamic_per_pixel(). Each pixel is expected to contain a 365-day moving-average series.
kMax	Integer. Maximum number of clusters to test. Default is 10.
minimum_clusterSize	Numeric. Minimum allowed cluster size expressed as a proportion of all clustered pixels. Default is 0.01.
correlate_first	Logical. If TRUE, pixels are clustered by their correlation profiles rather than by raw index values.

Details

At most 10,000 pixels are sampled before testing cluster counts. If correlate_first = TRUE, the selected pixels are correlated with at most 1,000 reference pixels. See prepare_for_clustering() for details on NA handling and optional correlation pre-processing.

Value

Integer giving the recommended number of clusters. The function also draws a diagnostic plot of explained variance by number of clusters.

---

combine_years_dynamic *Combine whole-lake dynamics across years*

Description

Combines multiple annual outputs from dynamic_per_pixel() into summary curves across years. For each day of year, the function averages the annual median lake dynamics and calculates the standard deviation across those annual medians.

Usage

```
combine_years_dynamic(indexDynamicList, years = NULL)
```

Arguments

indexDynamicList	A named or unnamed list of annual dynamic objects created by dynamic_per_pixel(). Each element must contain lakeDynamic with column q_0.5 and days_around_ma.
years	Optional numeric vector of years to include. If NULL, all list elements are used. If the list is named, matching is performed on the names.

Details

All selected dynamics must have been computed with the same days_around_ma; otherwise the function stops because the smoothing windows are not comparable.

Value

A list with mean_of_median and sd_over_median, each a numeric vector with one value per day of year.

---

combine_years_lakeData

Combine lake-level index values across years

Description

Converts a list of annual lake index objects into a wide table with one row per lake and one index column per year.

Usage

```
combine_years_lakeData(lakeIndexList, aggregationType = "modus")
```

Arguments

lakeIndexList	A list of lists produced by seasonal_index_per_lake(). Each element must contain year, Name, ID, and the requested index statistic.
aggregationType	Character scalar. Use "modus" to extract IndexModusBest or "median" to extract IndexMedian.

Details

The function warns if lake names or IDs are not unique in the output.

Value

A data.frame with lake identifiers (name, id) and one wide year_<year> column per available year.

combine_years_pixelData *Combine annual lake index results into a harmonized pixel-wise data structure*

Description

This function merges multiple annual lake index maps (each produced by `seasonal_index_per_lake()`) into a unified per-pixel dataset. It ensures that all years are represented on a common spatial grid. If differences in CRS or grid alignment between years exist, the rasters are resampled to match the most recent year's configuration.

Usage

```
combine_years_pixelData(lakeIndexList)
```

Arguments

`lakeIndexList`	A list of yearly lake index lists created by `seasonal_index_per_lake()`. Each element must contain (at minimum) `IndexPixel`, `x`, `y`, `crs`, and `year` components.

Details

The function performs the following steps:

1. Validates input to ensure `lakeIndexList` is a list of lists.
2. Extracts available years and their corresponding CRS values.
3. Selects the most recent year to define the target grid.
4. For each year, checks grid compatibility; if needed, resamples to the target resolution and CRS using bilinear interpolation.
5. Merges all years' pixel tables into a single data frame by coordinates.

Value

A list with the following elements:

`yearsIndex` A `data.frame` where each row corresponds to a pixel position (`x`, `y`, `i_row`, `i_col`) and each `yYYYY` column stores the pixel values of that year.

`crs` A character string giving the CRS of the final combined grid.

See Also

`seasonal_index_per_lake()`, `terra::resample()`, `terra::rast()`

`determine_trend` *Calculate adjusted linear trends over recent years*

Description

Calculate adjusted linear trends over recent years

Usage

```
determine_trend(yearly_spread = yearly_spread, tyears = 3, ttype = "short")
```

Arguments

`yearly_spread`	A data frame of lakes or pixels as row and the yearly index values as columns
`tyears`	The number of years used for the trend analysis.
`ttype`	Character scalar, either `"short"` or `"long"`. Used to name the output columns.

Details

In order to rule out systematic errors based on weather conditions in a year, different hardware or algorithms, the difference between the overall median value of all lakes (or all pixels) and years and the yearly median value of all lakes(or all pixels) is added to the Index of a lake (or pixel) before the trends are calculated. Median values are based on lakes or pixels for which data is available in all considered years. Subsequantly, the trend is calculated as linear regression between previous years and adjusted NDTrI values. For a reliable trend assessment the number of lakes needs to be large enough (n>10). Only rows with complete data in all year columns are used to estimate the global and yearly medians for the bias correction. Individual row trends are returned as `NA` if any value in the selected trend window is missing.

Value

A list with `trends`, `periodYear`, `overallMedian`, and `yearlyMedians`. `trends` contains the slope and the standard error of the slope for each row.

`discreteClassAssessment`

Classify yearly index values into discrete classes

Description

Assigns each yearly index value to one of `nClass` classes. Break points are computed separately for each year using the observed minimum and maximum as outer bounds and equally spaced inner breaks between lower and upper quantiles.

Usage

```
discreteClassAssessment(yearly_spread, nClass = 10, proportionExtreme = 0.05)
```

Arguments

`yearly_spread`	Data frame with one or more columns whose names start with `year_`.
`nClass`	Integer. Number of classes to create per year. Default is `10`.
`proportionExtreme`	Numeric between 0 and 1. Initial proportion used for the lower and upper quantiles that define the inner range. Default is `0.05`.

Details

If the requested extreme proportion would create class sizes smaller than one class share, it is reduced to `1 / nClass`. #' Breaks are constructed using actual data min/max for outer bounds and creates `nClass - 2` **equally spaced** classes between the proportionExtreme quantiles (`1 - proportionExtreme` and `0 + proportionExtreme`).

Value

A list with `assessment`, `breaks`, `nClass`, and `proportionExtreme`. `assessment` is the input data frame with additional `<year>_class` columns. `breaks` contains the class boundaries for each year.

`display_geometry` *Display a bounding box or polygon on a leaflet map*

Description

Creates an interactive leaflet map for quickly checking the spatial extent that will be used for an openEO request.

Usage

```
display_geometry(geom, zoom = 15)
```

Arguments

`geom`	Either a named numeric bounding box vector in the order `north`, `east`, `south`, `west`, or a list of latitude-longitude coordinate pairs representing a polygon.
`zoom`	Numeric scalar. Initial leaflet zoom level. Default is `15`.

Details

Polygon coordinates are expected as latitude-longitude pairs, matching common copy/paste formats from map tools. They are internally switched to longitude-latitude order for leaflet. Bounding Box as a named vector or c("north" = lat1, "east" = lon1, "south" = lat2, "west" = lon2) or a list of pairs list(c(lat1, lng2), c(lat2, lon2), c(lat3, lon3))

Value

A `leaflet` map object with either a rectangle or polygon overlay.

`download_openEO_job` *Download results from a finished openEO job*

Description

Searches for a finished openEO job by title or ID and downloads its results to a local folder. The function assumes that the user is already connected and authenticated with openEO.

Usage

```
download_openEO_job(
  title,
  path,
  recentJob = TRUE,
  ID = NULL,
  overwrite = FALSE
)
```

Arguments

`title`	Character scalar. Job title to search for and folder name used under `path`. If `ID` is supplied, `title` is still used as the local folder name.
`path`	Character scalar. Directory into which the job folder is written.
`recentJob`	Logical. If `TRUE`, only the ten most recent jobs are searched when `ID` is not supplied.
`ID`	Optional openEO job identifier or list index. If supplied, this is used instead of title matching.
`overwrite`	Logical. If `FALSE`, the function stops when the target directory already exists. Default is `FALSE`.

Value

Invisibly returns `NULL`. On success, job assets are written to `file.path(path, title)`. If the job is not finished, a message is printed and no files are downloaded.

dynamic_per_pixel *Calculate day-of-year moving-average dynamics per pixel*

Description

Converts image-wise remote-sensing index matrices into pixel-wise time series and calculates smoothed 365-day dynamics. The output includes lake-level summary statistics and, optionally, the individual pixel dynamics.

Usage

```
dynamic_per_pixel(
  imageIndex,
  nc,
  water_scenes_only = TRUE,
  days_around_ma = 20,
  maxPixels = 1000,
  pixelFilter = NULL,
  pixelQualityThreshold = 0.8,
  maxDataPoints = 5e+06,
  returnSinglePixels = TRUE
)
```

Arguments

imageIndex	A list created by ndi_per_image() containing RSindex, t_date, t, x, y, and crs.
nc	A netCDF list returned by open_netcdf(). The SCL band is loaded from this object for scene filtering.
water_scenes_only	Logical. If TRUE, only pixels with SCL class 6 (water) are retained before averaging. If FALSE, cloudy pixels (SCL 8,9,10,11) are removed before averaging.
days_around_ma	Integer or NULL. Half-window size, in days, around each day of year. (i.e. 10 means 10 days before and ten days after the actual day are used for averaging). If NULL, it is derived from the median number of valid observations per pixel.
maxPixels	Numeric. Maximum number of pixels for which individual dynamics are calculated. Values <= 1 are interpreted as a proportion of valid pixels; Inf uses all valid pixels. Ignored when pixelFilter is set.
pixelFilter	Optional integer vector of pixel positions in the original matrix order. If supplied, exactly these pixels are processed.
pixelQualityThreshold	Optional numeric scalar between 0 and 1. Minimum water-scene proportion required for a pixel. If "byMedian", the median number of valid observations among non-empty pixels is used to select pixels.

`maxDataPoints` Numeric. Maximum number of matrix cells processed in one block. Larger datasets are split to reduce memory pressure.

`returnSinglePixels`
Logical. If `TRUE`, returns `pixelDynamics` and the selected pixel coordinates. If `FALSE`, only lake-level summaries are returned.

Details

For each day of year, all images within `days_around_ma` days before and after the target day are averaged. A moving average is only calculated if more than one image falls into the window. Long periods without image availability are set to `NA` to avoid excessive temporal extrapolation.

Value

A list with `lakeDynamic`, `crs`, `year`, `days_around_ma`, and `dataAvailabilityThreshold`. If `returnSinglePixels = TRUE`, the list also contains `x`, `y`, and `pixelDynamics`.

`find_trends` *Rank lakes by short- or long-term trend*

Description

Orders lakes or rows from a `numericAssessment()` result according to the selected trend column.

Usage

```
find_trends(
  numericAssessment,
  trendType,
  bestTrendFirst = FALSE,
  outputVector = "lakeID"
)
```

Arguments

`numericAssessment`
A list created by `numericAssessment()`.

`trendType` Character scalar, usually `"short"` or `"long"`, selecting the column `trend_<trendType>`.

`bestTrendFirst` Logical. Passed to the ordering step. A good Trend is a negative trend, indicating a decrease in eutrophication.

`outputVector` Character scalar. One of `"lakeID"`, `"lakeName"`, or `"rowNumber"`. Determines what is returned.

Value

An ordered vector of lake IDs, lake names, or row numbers. Rows with missing trend values are removed from the ranking.

get_clusterLayer *Create a matrix of clusters in the same dimension as Datacube nc*

Description

Create a matrix of clusters in the same dimension as Datacube nc

Usage

```
get_clusterLayer(clusterVector, nc)
```

Arguments

clusterVector	A named vector of clusters per pixel. The names are are made of the index within the datacube xy-matrix combined the "p_" as prefix (Vector is part of the returned output by pixel_clusters())
nc	A netCDF list returned by open_netcdf(). The SCL band is loaded from this object for scene filtering.

load_BandLayer *Load one band from an opened netCDF data cube*

Description

Extracts all time layers of a selected Sentinel-2 band from an open_netcdf() object, optionally restricted to selected months and years.

Usage

```
load_BandLayer(nc, band, monthFilter = NULL, yearFilter = NULL)
```

Arguments

nc	A list returned by open_netcdf() or a compatible object containing SpRast, bands, x, y, t, t_date, and crs.
band	Character scalar with the band name to load, for example "B02", "B05", or "SCL".
monthFilter	Optional numeric vector of months to retain.
yearFilter	Optional numeric vector of years to retain.

Details

Negative band values are set to zero before matrices are returned. This is intended for reflectance bands and follows the Sentinel-2 pre-processing convention to harmonize data (https://docs.sentinel-hub.com/api/latest/data/sentinel-2-l2a/)

Value

A list containing coordinates, time metadata, CRS, and band, a list of matrices with one matrix per retained time step.

`map_layer` *Display a netCDF-aligned matrix as an interactive leaflet layer*

Description

Reprojects a matrix aligned with an `open_netcdf()` object to WGS 84 and adds it as a raster overlay on an interactive leaflet map.

Usage

```
map_layer(
  ncLayer,
  nc,
  lowerLimits = NULL,
  highestValue = NULL,
  classColors = NULL,
  valueRange = NULL,
  legendTitle = NULL,
  plotLegend = TRUE
)
```

Arguments

`ncLayer`	Matrix with the same row and column dimensions as the netCDF grid. Values can be numeric classes, continuous numeric values, or hex color strings.
`nc`	A list returned by `open_netcdf()`. Used for coordinates and CRS.
`lowerLimits`	Optional numeric vector of lower class limits. If supplied, categorical intervals are created with `cut()`.
`highestValue`	Optional numeric upper bound appended to `lowerLimits`. If `NULL`, the maximum layer value is used.
`classColors`	Character vector of colors corresponding to the classes defined by `lowerLimits`.
`valueRange`	Optional numeric vector of length two for continuous color scaling.
`legendTitle`	Optional character scalar used as legend title.
`plotLegend`	Logical. If `TRUE`, a leaflet legend is added when enough information is available.

Details

The function supports a hex-color mode when all values of `ncLayer` are character strings beginning with #.

Value

A `leaflet` map object.

nc_scene_per_image *Calculate scene-class coverage per image*

Description

Calculates the proportion of pixels belonging to a selected Sentinel-2 SCL scene group for each image in a netCDF data cube.

Usage

```
nc_scene_per_image(nc, scene = "clouds")
```

Arguments

nc	A list returned by open_netcdf().
scene	Character scalar. One of "clouds", "water", "snow", or "shadows".

Details

Scene groups are mapped to SCL codes as follows: clouds = 8, 9, 10; water = 6; snow = 11; shadows = 2, 3.

Value

A data.frame with columns date and coverage. coverage is the proportion of pixels in the selected scene group for each image.

ndi_per_image *Calculate a normalized difference index for each image and pixel*

Description

Loads two bands from a netCDF data cube and calculates an pixel-wise normalized difference index for each retained time step.

Usage

```
ndi_per_image(nc, year, bandNames = c("B05", "B02"), monthFilter = NULL)
```

Arguments

nc	A netCDF list created by open_netcdf().
year	Numeric scalar. Year to process. Only one year can be processed at a time.
bandNames	Character vector of length two. The index is calculated as (bandNames[1] - bandNames[2]) / (bandNames[1] + bandNames[2]). The default c("B05", "B02") corresponds to the NDTrI setup used by this package.
monthFilter	Optional numeric vector of months to include. If NULL, all months in year are used.

Details

Band values are loaded through `load_BandLayer()`, which sets negative values to zero before index calculation.

Value

A list containing `x`, `y`, `t`, `t_date`, `crs`, and `RSindex`. The `RSindex` element is a list of matrices, one per image.

NDTrIColors *Colors for the Index*

Description

Colors for the Index

Usage

```
NDTrIColors
```

Format

A dataframe of two columns. "ndtri" listing the values of the noramlized diference trophic index and "color" specifying the corresponding colors

---

numericAssessment *Assess lake status and relative trends from yearly index values*

Description

Adds moving-average status and relative short- and long-term trend estimates to a wide table of yearly index values.

Usage

```
numericAssessment(
  yearly_spread,
  statusYears = 3,
  shortTermYears = 3,
  longTermYears = 10
)
```

Arguments

`yearly_spread`	A data frame with lake or pixel identifiers and yearly index columns named `year_<year>`
`statusYears`	Integer. Number of years used for moving-average status. Default is `3`.
`shortTermYears`	Integer. Number of years used for the short-term trend. Default is `3`.
`longTermYears`	Integer. Number of years used for the long-term trend. Default is `10`.

Details

In order to rule out systematic errors based on weather conditions in a year, different hardware or algorithms, the difference between the overall median value of all lakes (or pixels) and years and the yearly median value of all lakes (or pixels) is added to each index per year before the trends are calculated. Thus, the trends of one lake (or pixel) are relative compared to the other lakes (or pixels) of the table. The trend is calculated as linear regression between previous years and adjusted NDI values. For a reliable trend assessment the number of lakes (or pixels) needs to be large enough (n>10).

Value

A list with `assessment` and `periods`. `assessment` is the input data frame extended by status and trend columns where possible. `periods` records the three period lengths used.

`oneLake_df` *Convert annual lake index results for one lake to a data frame*

Description

Combines a list of annual `seasonal_index_per_lake()` outputs for one lake into a long-format summary table.

Usage

```
oneLake_df(lakeIndexList)
```

Arguments

`lakeIndexList`	A list of yearly lake index lists, usually all belonging to the same lake.

Value

A `data.frame` with lake name, lake ID, year, median index, standard deviation, selected modal index, number of valid pixels, and quality threshold.

open_netcdf *Open an openEO netCDF data cube*

Description

Reads a netCDF file into a `terra` SpatRaster and extracts spatial, temporal, band, and CRS metadata used by the lakeRS processing workflow.

Usage

```
open_netcdf(filePath, fileName = "openEO.nc")
```

Arguments

`filePath`	Character scalar. Directory containing the netCDF file.
`fileName`	Character scalar. File name including the `.nc` extension. Default is `"openEO.nc"`.

Details

The function stops if subdatasets in the netCDF file have different row, column, or layer dimensions.

Value

A list with `bands`, `x`, `y`, `t`, `t_date`, `crs`, and `SpRast`. `x` and `y` are pixel-center coordinates, `t` contains numeric time steps, and `t_date` converts these time steps to dates using origin `1990-01-01`.

pixel_clusters *Cluster pixel dynamics with k-means*

Description

Performs k-means clustering on pixel-level moving-average dynamics and returns cluster assignments, cluster centers, and selected k-means diagnostics.

Usage

```
pixel_clusters(
  pixelDynamic,
  k,
  iter.max = 20,
  nstart = 10,
  correlate_first = FALSE,
  whole_dynamic = FALSE
)
```

Arguments

pixelDynamic	A list or data-frame-like object of pixel dynamics, usually pixelDynamics from dynamic_per_pixel().
k	Integer. Number of clusters.
iter.max	Integer. Maximum number of k-means iterations.
nstart	Integer. Number of random starts passed to stats::kmeans().
correlate_first	Logical. If TRUE, clustering uses pixel correlation profiles instead of raw moving-average values.
whole_dynamic	Logical. If FALSE, the number of days used for clustering may be reduced for large pixel sets. If TRUE, all 365 days are used.

Value

A list with clusterVector, clusterCenter, kmeansOut, and days_included.

---

plot_class_assessment *Plot discrete class assessment results for selected lakes*

Description

Draws a class-by-year heatmap for selected lakes from a discreteClassAssessment() result.

Usage

```
plot_class_assessment(
  class_assessment,
  lakeNames = NULL,
  lakeIDs = NULL,
  rowNumbers = NULL
)
```

Arguments

class_assessment	A list created by discreteClassAssessment().
lakeNames	Optional character vector of lake names to plot.
lakeIDs	Optional character vector of lake IDs to plot.
rowNumbers	Optional numeric vector of row numbers in the assessment table to plot.

Details

At least one of lakeNames, lakeIDs, or rowNumbers must identify rows. Colors are generated by rescale_classColors().

Value

No explicit return value. The function draws a base R plot.

`plot_dynamic` *Plot annual dynamics with optional uncertainty intervals*

Description

Draws one or more 365-day time series and optionally overlays 50% and 95% interval polygons. The function is designed for lake-level or cluster-level dynamics derived from `dynamic_per_pixel()` or `combine_years_dynamic()`.

Usage

```
plot_dynamic(
  v_averageList,
  v_sdList = NULL,
  df_q50List = NULL,
  df_q95List = NULL,
  TScolors = lakeRS::tenClusterColors$color,
  TSnames = NULL,
  df_reference = NULL,
  RefName = NULL,
  lakeName = "",
  ylab = "Index",
  ylim = NULL
)
```

Arguments

`v_averageList`	Numeric vector or list of numeric vectors with one value per day of year.
`v_sdList`	Optional numeric vector or list of vectors with standard deviations. If supplied, 50% and 95% intervals are approximated from the standard deviations.
`df_q50List`	Optional data frame or list of data frames with lower and upper 50% interval columns.
`df_q95List`	Optional data frame or list of data frames with lower and upper 95% interval columns.
`TScolors`	Character vector of colors for the time series and polygons.
`TSnames`	Optional character vector of legend names for the time series.
`df_reference`	Optional two-column data frame with reference day and value columns.
`RefName`	Optional character scalar for the reference-line legend.
`lakeName`	Character scalar used as plot title prefix.
`ylab`	Character scalar. Y-axis label.
`ylim`	Optional numeric vector of length two with y-axis limits.

Details

Month separators and labels are drawn for a non-leap 365-day year. Missing interval values are removed with `rm_na_for_polygon()` before polygons are drawn.

Value

No explicit return value. The function draws a base R plot.

---

plot_lake_index_histogram

Plot the distribution of pixel index values for one lake-year

---

Description

Draws a histogram and density curve of all valid pixel-level index values for a lake in one year. The selected modal value and median are highlighted.

Usage

```
plot_lake_index_histogram(lakeIndex, lakeName = "", indexName = "")
```

Arguments

`lakeIndex`	A list created by `seasonal_index_per_lake()`.
`lakeName`	Character scalar used in the plot title.
`indexName`	Character scalar used as x-axis label.

Details

The current implementation reads `lakeIndex$nValidePixel`, matching the current spelling in `seasonal_index_per_lake()`.

Value

`NULL` if no valid pixels are available; otherwise no explicit return value. The function draws a base R plot.

plot_layer *Plot a netCDF-aligned matrix as a static map*

Description

Reprojects a matrix aligned with a netCDF grid to WGS 84 and plots it with a distance scale and optional legend using base graphics.

Usage

```
plot_layer(
  ncLayer,
  nc,
  lowerLimits = NULL,
  classColors = NULL,
  highestValue = NULL,
  valueRange = NULL,
  legendTitle = NULL,
  plotLegend = TRUE
)
```

Arguments

ncLayer	Matrix with the same row and column dimensions as the netCDF grid. Values can be numeric or hex color strings.
nc	A list returned by open_netcdf(). Used for coordinates and CRS.
lowerLimits	Optional numeric vector of class lower limits.
classColors	Character vector of colors for classes. In continuous mode without valueRange, defaults internally to c("white", "black").
highestValue	Optional upper bound for the last class.
valueRange	Optional numeric vector of length two for continuous value scaling.
legendTitle	Optional character scalar used as legend title.
plotLegend	Logical. If TRUE, a legend is plotted when class labels are available.

Details

The plot is scaled according to geodesic width and height estimated with geosphere::distGeo(). The helper plot_scale() adjusts the plot region to preserve the map's aspect ratio.

Value

No explicit return value. The function draws a base R plot.

plot_numeric_assessment *Plot numeric status and trend assessment for one lake*

Description

Creates a multi-panel base R plot showing one selected lake's yearly index values, moving-average status, and long- and short-term trend estimates.

Usage

```
plot_numeric_assessment(
  numericData,
  lakeName = NULL,
  lakeID = NULL,
  rowNumber = NULL,
  ylab = "Index"
)
```

Arguments

numericData	A list created by numericAssessment().
lakeName	Optional character scalar identifying the lake by name.
lakeID	Optional character scalar identifying the lake by ID.
rowNumber	Optional numeric row number in numericData$assessment.
ylab	Character scalar. Y-axis label for the index panel.

Details

Exactly one lake should be selected, either by row number, lake name, or lake ID. The first panel shows the distribution of all lakes as quantile bands and overlays the selected lake's annual values and status line. The two trend panels show point estimates with approximate 95% intervals.

Value

No explicit return value. The function opens a graphics device and draws the assessment plot.

plot_scene_coverage *Plot image availability and scene coverage by year*

Description

Visualizes image dates over the day of year, grouped by year, and colors each image according to a scene-coverage proportion.

Usage

```
plot_scene_coverage(
  vDate,
  vCoverage = NULL,
  upperLimits = c(0.05, 0.1, 0.3, 0.5, 0.75, 1)
)
```

Arguments

vDate	Date vector of image acquisition dates.
vCoverage	Optional numeric vector or one-column data frame of scene proportions per image, such as the coverage column from nc_scene_per_image(). If NULL, coverage is shown as missing.
upperLimits	Numeric vector of upper interval limits between 0 and 1. Default is c(0.05, 0.1, 0.3, 0.5, 0.75, 1).

Details

Duplicate images on the same day of year are vertically offset within the year row. The right axis shows the number of images per year.

Value

No explicit return value. The function draws a base R plot.

prepare_for_clustering *Prepare pixel dynamics for clustering*

Description

Cleans and optionally transforms a pixel-by-day moving-average matrix before k-means clustering.

Usage

```
prepare_for_clustering(
  moving_averages_matrix,
  correlate_first = FALSE,
  maxPixels = NULL
)
```

Arguments

moving_averages_matrix	Matrix or data frame with days as rows and pixels as columns. The first column is dropped before processing, matching outputs that include a day-of-year column.
correlate_first	Logical. If TRUE, pixels are represented by their Pearson correlation with a reference set of pixels.
maxPixels	Optional integer. Maximum number of pixels to retain. If not NULL, a reproducible random sample is drawn.

Details

Values are rounded with signif() before clustering. Depending on the NA pattern, the function removes either rows with missing values or pixels with missing values and reports removals via warnings. If correlation pre-processing is requested, at most 1,000 reference pixels are sampled.

Value

A numeric matrix used as input for clustering. Columns represent pixels when correlate_first = FALSE; otherwise the matrix contains rounded correlation profiles.

---

sameDimProjection *Reproject a raster while preserving row and column counts*

Description

Reprojects a terra input raster to a specified target coordinate reference system (CRS), preserving the original number of rows and columns. The spatial extent of the raster is transformed to the new CRS, and a new raster template is created to match these dimensions before projection.

Usage

```
sameDimProjection(initial_raster, finalCRS = "EPSG:4326")
```

Arguments

initial_raster	A terra::SpatRaster object.
finalCRS	Character string specifying the target CRS, in EPSG or WKT format. Default is "EPSG:4326".

Details

The input extent is projected to the target CRS and used to create a template raster. Projection uses nearest-neighbor resampling (`method = "near"`), which is appropriate for categorical layers and avoids creating interpolated class values.

Value

A `SpatRaster` object reprojected to the specified CRS, maintaining the same number of rows and columns as the input raster.

See Also

`terra::project()`, `terra::rast()`, `terra::ext()`

`save_leaflet` *Save a leaflet map as a self-contained HTML file*

Description

Adds a mobile-friendly viewport meta tag to a leaflet widget and saves it as a self-contained HTML file.

Usage

```
save_leaflet(x, file)
```

Arguments

`x`	A `leaflet` object.
`file`	Character scalar. Output HTML file including path.

Value

The return value of `htmlwidgets::saveWidget()`. The main side effect is writing `file`.

sceneIDs *Sentinel-2 IDs of the SCL Band*

Description

Sentinel-2 IDs of the SCL Band

Usage

```
sceneIDs
```

Format

A data frame of two columsn 1) SCL ID and 2) SCL name

seasonal_index_per_lake *Aggregate seasonal pixel indices to one lake-level index*

Description

Summarizes the pixel-level seasonal index matrix of one lake and year into lake-level statistics, including median, standard deviation, density-based modal candidates, and a selected modal index value.

Usage

```
seasonal_index_per_lake(seasonIndex, lakeName = NULL, lakeID = NULL)
```

Arguments

seasonIndex	A list created by seasonal_index_per_pixel() containing at least RSindex, QualityThreshold, x, y, crs, and year.
lakeName	Optional character scalar with the lake name. If NULL, the output Name is NA.
lakeID	Optional character scalar with a lake identifier. If NULL, an identifier is generated by add_lakeID() from mean coordinates.

Details

If more than one valid pixel is available, potential modes are detected from a kernel density estimate using bw = "SJ" and adjust = 2. Among sufficiently high local maxima, the mode with the highest density is selected as IndexModusBest. If exactly one valid pixel is available, that value is used for both median and mode.

Value

A list with index statistics (`IndexModusBest`, `IndexMedian`, `IndexSD`, `IndexModusPot`, `IndexDensity`, `IndexPixel`), metadata (`QualityThreshold`, `x`, `y`, `crs`, `year`), and lake identifiers (`Name`, `ID`).

`seasonal_index_per_pixel`

Calculate a seasonal index value for each pixel

Description

Filters image-wise index matrices by Sentinel-2 SCL information, averages the remaining index values pixel by pixel, and removes pixels that do not meet a water-scene quality threshold.

Usage

```
seasonal_index_per_pixel(
  imageIndex,
  nc,
  water_scenes_only = TRUE,
  pixelQualityThreshold = 0.8
)
```

Arguments

`imageIndex`	A list created by `ndi_per_image()` containing image-wise `RSindex` matrices, time information, coordinates, and CRS.
`nc`	A netCDF list returned by `open_netcdf()`. Used to load the SCL band for the same year and months as `imageIndex`.
`water_scenes_only`	Logical. If `TRUE`, processing keeps SCL class 6 (water) before averaging. If `FALSE`, the current code calls `scl_mask()` with SCL classes 8 to 11; because `scl_filter()` currently ignores `invert`, this mode should be checked before use.
`pixelQualityThreshold`	Numeric between 0 and 1. Minimum water-scene proportion required for a pixel to be retained in the seasonal index.

Details

Water-scene proportions are calculated by `waterscene_proportion()`. A warning is issued if no pixel meets `pixelQualityThreshold`.

Value

A list with `x`, `y`, `crs`, `year`, `QualityThreshold`, `RSindex`, and `sceneProportions`. `RSindex` is a matrix of seasonal pixel index values; pixels below the water-quality threshold are set to `NA`.

show_pixels *Display pixel locations on an interactive map*

Description

Projects input coordinates to WGS 84 and displays them as circle markers on a leaflet map.

Usage

```
show_pixels(x, y, crs, labels = NULL)
```

Arguments

x	Numeric vector of x coordinates (e.g., easting or longitude in the source coordinate reference system).
y	Numeric vector of y coordinates (e.g., northing or latitude in the source coordinate reference system). Must be the same length as x.
crs	Character string specifying the source coordinate reference system of x and y, in a format understood by terra::project() (e.g., "EPSG:32633").
labels	Optional character vector of labels for the markers. If provided, its length should match the length of x and y.

Details

The marker colors are currently hard-coded as c("blue", "green", "orange", "purple"). If the number of points exceeds the number of colors, colors will be recycled.

Value

A leaflet map object.

start_openEO_job *Start Job on openEO plattform*

Description

Builds an openEO process graph for a Sentinel-2 collection, spatial and temporal extent, optional property filters, and selected bands. The graph is saved as the requested output format and started as an openEO batch job.

Usage

```
start_openEO_job(
  title,
  geom,
  tBeg,
  tEnd,
  crs = 4326,
  collection = "SENTINEL2_L2A",
  bands = list("B02", "B05", "SCL"),
  relativeOrbitNumber = NULL,
  tileID = NULL,
  outputFormat = "netCDF",
  uniqueTitle = TRUE
)
```

Arguments

`title`	A character defining the job title (can be a lake name, id). It is advised to use a unique name (i.e. including a timestamp) that is used on openEO and as a folder on the local machine
`geom`	Numerical vector defining the bounding box in the coordinate reference system as defined in crs. The order of the values is north, east, south, west. Alternatively, geom can be a list of 3 or more latitude-longitude pairs, each pair is a vector of length 2 in the list to create a polyong (which is actually not bounding box any longer...)
`tBeg, tEnd`	Dates in the format "YYYY-MM-DD" which define the temporal extent of the data to be downloaded. Left side (tBeg) is included, right side (tEnd) is excluded from range.
`crs`	A numerical value defining the coordinate reference system. By default this is 4326 (WGS 84). See here for further information: https://epsg.io/4326
`collection`	One of the available collections on openEO (SENTINEL2_L2A is default). The collections can be listed by `openeo::list_collections()`
`bands`	A list of characters defining The bands to be downloaded. For the normalized difference trophic index bands 2, 5 and the scene classification (SCL) are required. Set to NULL to download all bands.
`relativeOrbitNumber`	Definition of relativeOrbitNumber to filter for as character vector.
`tileID`	Definition of Tiles to filter for as character vector.
`outputFormat`	On of the available output formats (netcdf is default). The formats can be listed by `openeo::list_file_formats()`. Other alternatives for higher dimensional data cubes are "GeoJSON" or "GTiff".
`uniqueTitle`	If TRUE, the previous jobs are checked. If the title has already been used, it is suffixed by an underscore and the lowest possible number

Details

Laitude-Longitude Pairs are switched to longitude-latitude which is the required order of openEO, however providing the data as latitude-longitude is more convient beacuse it is the google maps output format

Value

This function only returns the title of the job that is started on openEO. To download the job use function download_openEO_job

---

tenClassColors *Colors for the Classes*

Description

Colors for the Classes

Usage

```
tenClassColors
```

Format

A data frame of two columns. "class" listing the classes 1 to 10 and "color" specifying the corresponding colors

---

tenClusterColors *Colors for the Clusters*

Description

Colors for the Clusters

Usage

```
tenClusterColors
```

Format

A data frame of two columns. "cluster" listing the clusters 1 to 10 and "color" specifying the corresponding colors

trends_to_classes *Convert numeric trend estimates to significance classes*

Description

Adds categorical trend-significance and trend-strength columns to an assessment table containing trend estimates and standard errors.

Usage

```
trends_to_classes(assessmentTable, trendType = "long")
```

Arguments

assessmentTable	A data frame, typically `numericAssessment(...)$assessment`, containing trend and error columns for the selected `trendType`.
trendType	Character scalar. Either `"long"` or `"short"`; selects `trend_long`/`error_long` or `trend_short`/`error_short`.

Details

Significance classes are `0` if zero lies within ±1 standard error, ±1 if zero lies outside ±1 but inside ±2 standard errors, and ±2 if zero lies outside ±2 standard errors. Positive trend estimates are converted to negative classes, so class sign follows the package's interpretation that a negative numerical trend indicates improvement of the trophic index.

Value

The input data frame extended by `trend_<trendType>_significance` and `trend_<trendType>_strength`.

waterscene_proportion *Calculate water-scene proportions from SCL images*

Description

Computes per-pixel proportions for all Sentinel-2 SCL classes and derives a water-scene proportion after accounting for disturbance classes.

Usage

```
waterscene_proportion(scl_image)
```

Arguments

scl_image	A list of SCL matrices, typically loaded from the SCL band of an `open_netcdf()` object for the same images used in an index calculation.

Details

The water proportion is defined as all pixels that are "correctly" classified as water. Thus, disturbances that occur but are not a false classification like clouds, snow or topographic impacts need to be removed before. After removing those pixels, the water classifiation should be 100% for a perfect pixel

Value

A list with `water`, `NoFalseDisturbance`, and `allScenes`. `water` is the derived water-scene proportion per pixel; `NoFalseDisturbance` is the proportion of pixels assigned to allowed disturbance classes; `allScenes` is a list of per-class proportions.

Index