Package: RecoverMetricsR (via r-universe)

October 15, 2026

Type Package

Title Quantifying Post-Fire Spectral Recovery Metrics

Version 0.1.0

Description Computes pixel-wise metrics describing post-disturbance spectral recovery from a yearly spectral-index time series. Metrics include disturbance magnitude (DM), an ordinary least squares (OLS) recovery slope, early- and late-period recovery rates, recovery-rate change, Relative Recovery Indicator (RRI), Ratio of Eighty Percent (R80P), and Year-on-Year recovery (YrYr). The RRI, R80P, and YrYr metrics are based on approaches described by Frazier et al. (2018) <doi:10.1016/j.rse.2017.11.007>. Threshold-based recovery-time approaches have been used to quantify post-fire vegetation recovery from spectral time series by Chompuchan and Lin (2017) <doi:10.1016/j.ecolind.2017.04.038>. Exponential recovery formulations have also been applied to post-fire vegetation time series by Gouveia et al. (2010) <doi:10.5194/nhess-10-673-2010>. Parametric recovery models, including exponential, logistic, and Gompertz formulations, are provided to characterize nonlinear post-disturbance recovery trajectories.

License GPL-3

Encoding UTF-8

Depends R (>= 4.1.0)

Imports dplyr, ggplot2, grid, rlang, stats, terra (>= 1.8)

Suggests knitr, rmarkdown, sf, testthat (>= 3.0.0)

VignetteBuilder knitr

RoxygenNote 7.3.2

NeedsCompilation no

Author Hammadi Achour [aut, cre] (ORCID: <https://orcid.org/0000-0002-6364-3789>), Amal Daboussi [aut], Wahbi Jaouadi [aut]

Maintainer Hammadi Achour <hammadi.achour@gmail.com>

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

Date/Publication 2026-10-15 16:10:21 UTC

RemoteUrl https://github.com/cran/RecoverMetricsR

RemoteRef HEAD

RemoteSha 0d70cb68f11817a4647048b8c8b332ec6ab6562a

Contents

classify_disturbance_severity 2
classify_recovery_typology 5
compute_recovery_metrics 8
compute_recovery_time 12
compute_recovery_time_uncertainty 14
detect_disturbance 16
fit_recovery_model 19
fit_recovery_model_core 21
predict_recovery_raster 23
preprocess_raster 25

Index **28**

`classify_disturbance_severity`
Classify Spectral Disturbance and Recovery Across Post-Disturbance Years

---

Description

Classifies the magnitude and direction of spectral change between a pre-disturbance reference layer and multiple post-disturbance layers. Positive values of the pre-minus-post difference indicate a decrease in the spectral index, whereas negative values indicate an increase.

Usage

```
classify_disturbance_severity(
  raster_stack,
  pre_year,
  event_year,
  post_years,
  mask_layer = NULL,
  plot = TRUE,
  combine = TRUE
)
```

Arguments

`raster_stack`	A `terra::SpatRaster` containing a spectral index time series.
`pre_year`	Layer index, layer name, or year corresponding to the pre-disturbance reference condition.
`event_year`	Layer index, layer name, or year corresponding to the disturbance year. The argument is validated for workflow consistency but the event layer itself is not used in the post-disturbance classification.
`post_years`	Vector of layer indices, layer names, or years corresponding to post-disturbance observations. The event year is normally excluded so that all layers describe spectral change relative to the pre-disturbance reference.
`mask_layer`	Optional spatial mask. Can be a `terra::SpatRaster`, `terra::SpatVector`, or `sf` object used to restrict the analysis spatially.
`plot`	Logical. If `TRUE`, produces a plot of spectral-change class distributions when `combine = TRUE`.
`combine`	Logical. If `TRUE`, combines statistics from all post-disturbance years into a single data frame and plot.

Details

For each post-disturbance layer, spectral change is calculated as:

$$S_{pre} - S_{post}$$

where `S_pre` is the pre-disturbance spectral index value and `S_post` is the value observed in the corresponding post-disturbance year.

The resulting difference is assigned to seven classes ranging from high spectral disturbance to high spectral recovery. The classes describe the magnitude and direction of spectral change relative to the pre-disturbance reference.

A near-zero difference indicates a spectral condition close to the pre-fire reference. Positive values indicate that the post-disturbance spectral index remains below the pre-fire reference, whereas negative values indicate that the post-disturbance spectral index exceeds the pre-fire reference.

The classification thresholds are:

- High spectral disturbance: difference > 0.66
- High-moderate spectral disturbance: 0.44 < difference <= 0.66
- Low-moderate spectral disturbance: 0.27 < difference <= 0.44
- Low spectral disturbance: 0.10 < difference <= 0.27
- Near pre-fire spectral condition: -0.10 < difference <= 0.10
- Low spectral recovery: -0.25 < difference <= -0.10
- High spectral recovery: difference <= -0.25

The `Near pre-fire spectral condition` class represents a spectral state close to the pre-fire reference, within approximately +/- 0.10 spectral-index units. It does not imply complete ecological or structural recovery.

The `Low spectral recovery` and `High spectral recovery` classes indicate that the post-disturbance spectral index exceeds the pre-fire reference. They describe spectral recovery relative to the pre-fire reference and should not be interpreted as direct measurements of ecological regeneration or structural recovery.

These classes describe spectral change and should not be interpreted as direct measurements of ecological, structural, or combustion severity without independent validation.

Layer selection accepts three forms: a numeric index in `1..nlyr(raster_stack)`, an exact layer name, or a year value that appears within the layer names. Year matching uses a digit-boundary search, so a year like `2000` will match `"NBR_2000"` but not `"NBR_20000"`.

By convention, the event year is excluded from `post_years` so that all layers describe post-fire spectral change relative to the pre-disturbance reference. Users who wish to map immediate fire severity relative to the pre-disturbance reference can include the event-year layer in `post_years` explicitly. In that case, the resulting classes describe the immediate spectral disturbance associated with the event.

Value

A list containing:

severity_stack A `terra::SpatRaster` with one layer per value in `post_years`, named after the corresponding year and containing integer spectral-change classes.

stats_by_year A named list of data frames containing pixel counts and percentages for each class and post-disturbance year. Names match the layer names in `severity_stack`.

combined_stats A data frame combining spectral-change class statistics across post-disturbance years.

plot A `ggplot2` object showing the distribution of spectral-change classes, or `NULL` if plotting is disabled.

Examples

```
library(terra)

# Small toy example -- runs in well under 5 seconds
r <- rast(
  nrows = 2,
  ncols = 2,
  nlyrs = 5
)

values(r) <- rep(
  c(0.70, 0.30, 0.50, 0.55, 0.60),
  each = 4
)

names(r) <- paste0(
  "NBR_",
  2020:2024
)
```

```
# Event year excluded from post_years
result <- classify_disturbance_severity(
  raster_stack = r,
  pre_year = "NBR_2020",
  event_year = "NBR_2021",
  post_years = c(
    "NBR_2022",
    "NBR_2023",
    "NBR_2024"
  ),
  plot = FALSE
)

result$combined_stats

# Years are also accepted
result2 <- classify_disturbance_severity(
  raster_stack = r,
  pre_year = 2020,
  event_year = 2021,
  post_years = 2022:2024,
  plot = FALSE
)

identical(
  result$combined_stats$Count,
  result2$combined_stats$Count
)




# Combined statistics plot
result_plot <- classify_disturbance_severity(
  raster_stack = r,
  pre_year = "NBR_2020",
  event_year = "NBR_2021",
  post_years = c(
    "NBR_2022",
    "NBR_2023",
    "NBR_2024"
  ),
  plot = TRUE,
  combine = TRUE
)

result_plot$plot
```

classify_recovery_typology

Classify Recovery Typology

Description

Classifies post-disturbance recovery into four typological classes according to mean time-to-recovery (TTR) and sensitivity of TTR to the recovery threshold.

Usage

```
classify_recovery_typology(
  ttr_stack,
  fast_TTR = 3,
  unc_threshold = 2,
  plot_map = TRUE,
  plot_stats = TRUE,
  area = TRUE
)
```

Arguments

`ttr_stack`	A `terra::SpatRaster` produced by `compute_recovery_time_uncertainty`. It must contain at least two threshold-specific TTR layers and a `TTR_Threshold_Range` layer.
`fast_TTR`	Numeric. Maximum TTR, in years, defining fast recovery.
`unc_threshold`	Numeric. Maximum threshold range, in years, defining low sensitivity of the recovery-time estimate to the selected threshold.
`plot_map`	Logical. If `TRUE`, plots the recovery typology map.
`plot_stats`	Logical. If `TRUE`, plots the recovery typology statistics.
`area`	Logical. If `TRUE`, computes the area of each recovery type in hectares.

Details

Mean recovery time is calculated across all threshold-specific TTR layers. The threshold range is obtained from the `TTR_Threshold_Range` layer and represents the sensitivity of the recovery-time estimate to the choice of recovery threshold.

The typology combines two dimensions:

- recovery speed, based on whether mean TTR is less than or equal to `fast_TTR`;
- threshold sensitivity, based on whether the TTR range is less than or equal to `unc_threshold`.

Four classes are therefore produced:

- 1: Fast and stable recovery;
- 2: Fast but threshold-sensitive recovery;
- 3: Slow and stable recovery;

- 4: Slow and threshold-sensitive recovery.

The terms "stable" and "threshold-sensitive" refer specifically to the sensitivity of TTR to the selected recovery threshold. They do not imply ecological stability or statistical uncertainty.

The area calculation assumes the raster is in a projected coordinate system with meters as the linear unit. For rasters in geographic coordinates (degrees), project to a suitable CRS with `terra::project()` first.

Value

A list containing:

TTR_Mean Raster containing mean TTR across recovery thresholds.

TTR_Combined Raster containing mean TTR and threshold-range layers.

Recovery_Type Integer raster containing the four recovery typology classes.

Stats Data frame containing pixel counts, percentages, and, optionally, area in hectares for each recovery type.

Stats_Plot A `ggplot2` object showing recovery typology percentages, or `NULL` if `plot_stats = FALSE`.

Examples

```
library(terra)

# Small toy example -- runs in well under 5 seconds
r <- rast(nrows = 2, ncols = 2, nlyrs = 4)
values(r) <- c(
  2, 3, 5, 7,
  3, 4, 6, 8,
  2, 3, 5, 7,
  3, 4, 6, 8
)
names(r) <- c(
  "TTR_70", "TTR_80", "TTR_90",
  "TTR_Threshold_Range"
)

result <- classify_recovery_typology(
  ttr_stack = r,
  fast_TTR = 3,
  unc_threshold = 2,
  plot_map = FALSE,
  plot_stats = FALSE,
  area = TRUE
)

result$Stats
```

```
# Statistics plot
result_plot <- classify_recovery_typology(
  ttr_stack = r,
  fast_TTR = 3,
  unc_threshold = 2,
  plot_map = FALSE,
  plot_stats = TRUE,
  area = FALSE
)

result_plot$Stats_Plot
```

`compute_recovery_metrics`

Compute Pixel-wise Spectral Recovery Metrics

Description

Computes pixel-wise spectral recovery metrics from a disturbance-year raster time series. Metrics include disturbance magnitude (DM), OLS recovery slope, early and late recovery rates, recovery-rate change, Relative Recovery Indicator (RRI), Ratio of Eighty Percent (R80P), and Year-on-Year Average (YrYr).

Usage

```
compute_recovery_metrics(
  raster_stack,
  years,
  event_year,
  pre_window = 2,
  post_window = 5,
  mask_layer = NULL,
 metrics = c("DM", "Slope", "YrYr_early", "YrYr_late", "Recovery_Rate_Change", "RRI",
    "R80P", "YrYr"),
  clamp = TRUE,
 clamp_ranges = list(DM = c(-1, 1), Slope = c(-0.2, 0.2), YrYr_early = c(-0.5, 0.5),
  YrYr_late = c(-0.5, 0.5), Recovery_Rate_Change = c(-1, 1), RRI = c(-2, 3), R80P =
    c(-1, 4), YrYr = c(-0.5, 0.5)),
  eps = 1e-06
)
```

Arguments

`raster_stack`	A `terra::SpatRaster` containing the spectral index time series. Layers must correspond to the years supplied in `years`.

`years`	Numeric vector giving the year corresponding to each raster layer. Must be sorted in strictly increasing order.
`event_year`	Numeric value specifying the disturbance year.
`pre_window`	Positive integer specifying the number of observations before the disturbance used to calculate the pre-disturbance baseline.
`post_window`	Positive integer specifying the number of observations after the disturbance used for recovery analysis.
`mask_layer`	Optional spatial mask. Can be a `terra::SpatVector`, `terra::SpatRaster`, or an `sf` object. Raster masks are resampled to `raster_stack` if their CRS, resolution, or extent differ.
`metrics`	Character vector specifying the metrics to return. Available metrics are `"DM"`, `"Slope"`, `"YrYr_early"`, `"YrYr_late"`, `"Recovery_Rate_Change"`, `"RRI"`, `"R80P"`, and `"YrYr"`.
`clamp`	Logical. If `TRUE`, values outside the ranges specified in `clamp_ranges` are replaced by `NA`.
`clamp_ranges`	Named list defining the lower and upper bounds for individual metrics. Names must correspond to metric names and each element must be a length-2 numeric vector with `lower < upper`. Metrics not listed are not clamped. Defaults are tuned for NBR-like indices.
`eps`	Numeric. Small positive threshold below which denominators in ratio metrics are treated as zero, producing `NA`. Default `1e-6`.

Details

The RRI, R80P, and YrYr metrics are based on methods described by Frazier et al. (2018), with adaptations for a flexible number of post-disturbance observations. The recovery-rate change metric is a package-specific extension comparing early and late recovery rates.

The pre-disturbance baseline is calculated as the mean of the specified pre-disturbance layers. The disturbance value corresponds to the layer matching `event_year`.

The disturbance magnitude is calculated as:

$$DM = Index_{post,final} - Index_{event}$$

The OLS recovery slope is estimated from all available post-disturbance observations using the actual year offsets from the disturbance year.

`YrYr` represents the average annual recovery from the disturbance year to the final post-disturbance observation:

$$YrYr = \frac{Index_{post,final} - Index_{event}}{t_{final} - t_{event}}$$

where the denominator represents the actual elapsed number of years.

The early and late recovery rates are calculated from the mean spectral values during the early and late portions of the post-disturbance trajectory, respectively, relative to their mean year offsets.

The recovery-rate change is defined as:

$$Recovery_Rate_Change = YrYr_{early} - YrYr_{late}$$

Positive values indicate that the recovery rate decreases from the early to the late period, whereas negative values indicate an increase in the recovery rate.

The Relative Recovery Indicator is calculated as:

$$RRI = \frac{Index_{post,max} - Index_{event}}{Index_{pre} - Index_{event}}$$

The Ratio of Eighty Percent is calculated as:

$$R80P = \frac{Index_{post,max}}{0.8 \times Index_{pre}}$$

These formulations are based on methods described by Frazier et al. (2018), with adaptations to the flexible post-disturbance window used here.

The `post_window` argument specifies the number of raster layers used after the disturbance. If the supplied `years` contain gaps, the actual elapsed time is used when calculating `YrYr` and the OLS slope, but layer selection still follows the supplied layer order.

The default bounds in `clamp_ranges` are tuned for NBR-like indices that fall roughly within `[-1, 1]`. Users working with other spectral indices should supply their own bounds. Clamping replaces values outside the range with `NA`, which is useful for removing unphysical fits but produces additional `NA` values that would otherwise be finite. Set `clamp = FALSE` to inspect raw values.

The `RRI` and `R80P` metrics are ratio-based and become undefined when their denominators are close to zero. Specifically, `RRI` requires `Index_pre - Index_event != 0`, and `R80P` requires `Index_pre != 0`. Pixels where the denominator falls below `eps` are returned as `NA`. This occurs most often in pixels where the pre-disturbance and disturbance-year values are similar (e.g., unburned pixels inside a fire perimeter), and results in `RRI` and `R80P` containing more `NA` values than the difference-based metrics.

When the input raster includes large areas of unburned vegetation, the spatial distribution of the metrics will be dominated by pixels with little spectral change. For fire-specific analyses, restrict the analysis to the disturbance area using `mask_layer` or by cropping `raster_stack` before calling this function. This produces more interpretable distributions and reduces the fraction of pixels affected by clamping.

Value

A `terra::SpatRaster` containing the requested metrics.

References

Frazier, R. J., Coops, N. C., Wulder, M. A., Kennedy, R. E., & Hogg, E. H. (2018). Analyzing spatial and temporal variability in short-term rates of post-fire vegetation return from Landsat time series. *Remote Sensing of Environment*, 205, 32–45. doi:10.1016/j.rse.2017.11.007

Storey, E. A., Stow, D. A., O'Leary, J. F., & Riggan, P. J. (2016). Evaluating post-fire recovery of chaparral vegetation using Landsat time series. *Remote Sensing of Environment*, 183, 53–64. doi:10.1016/j.rse.2016.05.018

Examples

```
library(terra)

# Small toy example -- runs in well under 5 seconds
r <- rast(nrows = 2, ncols = 2, nlyrs = 6)
values(r) <- rep(c(0.60, 0.20, 0.30, 0.38, 0.45, 0.50), each = 4)
names(r) <- paste0("Index_", 2020:2025)

# Compute all available metrics
result <- compute_recovery_metrics(
  raster_stack = r,
  years = 2020:2025,
  event_year = 2021,
  pre_window = 1,
  post_window = 4
)
result

# Select a subset of metrics
selected <- compute_recovery_metrics(
  raster_stack = r,
  years = 2020:2025,
  event_year = 2021,
  pre_window = 1,
  post_window = 4,
  metrics = c("RRI", "R80P", "YrYr", "Recovery_Rate_Change")
)
selected




# Example with a spatial mask (SpatVector)
v <- vect(
  "POLYGON ((0 0, 1 0, 1 1, 0 1, 0 0))",
  crs = crs(r)
)

masked <- compute_recovery_metrics(
  raster_stack = r,
  years = 2020:2025,
  event_year = 2021,
  pre_window = 1,
  post_window = 4,
  mask_layer = v,
  metrics = c("RRI", "YrYr")
)
masked
```

compute_recovery_time *Compute Pixel-wise Recovery Time*

Description

Computes the number of years required for each pixel to reach a given proportion of its pre-disturbance spectral value and remain above it for a specified number of consecutive years.

Usage

```
compute_recovery_time(
  raster_stack,
  years,
  event_year,
  proportion = 0.8,
  pre_window = 2,
  persistence = 1,
  mask_layer = NULL,
  return_year = FALSE
)
```

Arguments

raster_stack	A terra::SpatRaster of yearly spectral index layers.
years	Numeric vector of years corresponding to raster layers. Its length must equal the number of raster layers.
event_year	Numeric. Disturbance year.
proportion	Numeric. Proportion of the pre-disturbance value used as the recovery threshold. Default is 0.8.
pre_window	Integer. Number of pre-disturbance observations used to compute the baseline.
persistence	Integer. Number of consecutive years the pixel must remain above the threshold to be considered recovered. Default is 1.
mask_layer	Optional spatial mask. Can be a terra::SpatRaster, terra::SpatVector, or sf object.
return_year	Logical. If TRUE, return the calendar year of recovery instead of the elapsed time. Default is FALSE.

Details

The pre-disturbance baseline is computed as the mean of the pre_window observations before event_year. The recovery threshold is proportion * baseline. For each pixel, the function finds the first run of at least persistence consecutive post-disturbance observations at or above the threshold.

Missing values are treated as "not recovered" for the purposes of run detection. The reported recovery time is the number of years between the disturbance and the first year of the qualifying run.

`pre_window` and `persistence` count raster layers (observations), not calendar years.

When the input raster includes unburned areas, those pixels will typically be classified as recovered at the first post-disturbance year, because their spectral values never dropped below the threshold. This produces a spike at TTR = 1 in the unmasked distribution. To obtain meaningful time-to-recovery distributions, restrict the analysis to the disturbance area using `mask_layer` or by cropping `raster_stack` before calling this function.

Value

A `terra::SpatRaster` with a single layer. If `return_year = FALSE`, the layer holds elapsed years since the disturbance. If `return_year = TRUE`, the layer holds the calendar year of recovery. Cells that never meet the recovery criterion are `NA`.

Examples

```
library(terra)

r <- rast(nrows = 4, ncols = 4, nlyrs = 6)
values(r) <- rep(c(0.60, 0.20, 0.30, 0.38, 0.45, 0.50), each = 16)
names(r) <- paste0("Index_", 2020:2025)

# Elapsed years since disturbance
ttr <- compute_recovery_time(
  raster_stack = r,
  years = 2020:2025,
  event_year = 2021,
  proportion = 0.8,
  pre_window = 1,
  persistence = 1
)

ttr




# Return the calendar year of recovery instead
rec_year <- compute_recovery_time(
  raster_stack = r,
  years = 2020:2025,
  event_year = 2021,
  proportion = 0.8,
  pre_window = 1,
  persistence = 2,
  return_year = TRUE
)

rec_year
```

compute_recovery_time_uncertainty

Assess Recovery-Time Threshold Sensitivity

Description

Computes time-to-recovery (TTR) for multiple recovery thresholds and quantifies the sensitivity of TTR to the selected threshold.

Usage

```
compute_recovery_time_uncertainty(
  raster_stack,
  years,
  event_year,
  thresholds = c(0.7, 0.8, 0.9),
  pre_window = 2,
  persistence = 1,
  mask_layer = NULL,
  return_band = TRUE
)
```

Arguments

raster_stack	A terra::SpatRaster containing yearly spectral index layers.
years	Numeric vector of years corresponding to the raster layers.
event_year	Numeric disturbance year.
thresholds	Numeric vector of recovery proportions. Values must be in the interval (0, 1].
pre_window	Number of observations before the disturbance used to calculate the pre-disturbance baseline.
persistence	Number of consecutive post-disturbance observations that must meet or exceed the recovery threshold.
mask_layer	Optional spatial mask. Can be a terra::SpatRaster, terra::SpatVector, or sf object, passed to compute_recovery_time.
return_band	Logical. If TRUE, also returns the range of TTR values obtained across the specified thresholds. Default is TRUE.

Details

Recovery time is computed independently for each threshold using `compute_recovery_time`. The threshold range is calculated as the difference between the maximum and minimum TTR across thresholds:

$$TTR_{range} = \max(TTR_p) - \min(TTR_p)$$

where p denotes the selected recovery proportions.

The resulting `TTR_Threshold_Range` represents sensitivity of the estimated recovery time to the choice of recovery threshold. It should not be interpreted as a statistical confidence interval or a formal measure of uncertainty.

Threshold-based recovery times have been used in post-fire vegetation recovery studies, including definitions based on a specified proportion of an undisturbed reference condition (Chompuchan and Lin, 2017).

Value

A `terra::SpatRaster` containing one layer for each recovery threshold. Layer names are formatted as `"TTR_70"`, `"TTR_80"`, etc. If `return_band = TRUE` and more than one threshold is supplied, an additional layer named `"TTR_Threshold_Range"` is returned.

References

Chompuchan, C., & Lin, C.-Y. (2017). Assessment of forest recovery at the deciduous forest in a complex terrain using Landsat time series. *Ecological Indicators*, 79, 249–259. doi:10.1016/j.ecolind.2017.04.038

Examples

```
library(terra)

# Small toy example -- runs in well under 5 seconds
r <- rast(nrows = 2, ncols = 2, nlyrs = 6)
values(r) <- rep(c(0.60, 0.20, 0.35, 0.42, 0.50, 0.56), each = 4)
names(r) <- paste0("NBR_", 2020:2025)

result <- compute_recovery_time_uncertainty(
  raster_stack = r,
  years = 2020:2025,
  event_year = 2021,
  thresholds = c(0.7, 0.8, 0.9),
  pre_window = 1,
  persistence = 1
)

result




# Without the sensitivity band, and with a spatial mask
```

```
library(sf)

p <- st_sfc(
  st_polygon(list(rbind(
    c(0, 0), c(1, 0), c(1, 1), c(0, 1), c(0, 0)
  ))),
  crs = crs(r)
)
p <- st_sf(id = 1, geometry = p)

result_masked <- compute_recovery_time_uncertainty(
  raster_stack = r,
  years = 2020:2025,
  event_year = 2021,
  thresholds = c(0.6, 0.7, 0.8, 0.9),
  pre_window = 1,
  persistence = 2,
  mask_layer = p,
  return_band = FALSE
)

result_masked
```

`detect_disturbance` *Detect Disturbance Year from Spectral Time Series*

Description

Detects a disturbance year from a raster spectral time series using either a year-on-year difference rule or a LandTrendr-style local OLS slope approach. A user-provided disturbance year can also be supplied, in which case automatic detection is skipped.

Usage

```
detect_disturbance(
  raster_stack,
  years,
  disturbance_year = NULL,
  mask_layer = NULL,
  detect = c("difference", "landtrendr")
)
```

Arguments

`raster_stack`	A `terra::SpatRaster` containing temporal spectral index layers.
`years`	Numeric vector giving the year corresponding to each raster layer. Its length must equal the number of raster layers.

`disturbance_year`
: Optional numeric value specifying a known disturbance year. If supplied, automatic detection is skipped.

`mask_layer`
: Optional spatial mask. Can be a `terra::SpatRaster`, `terra::SpatVector`, or `sf` polygon identifying the disturbed area. The mask is used only when `disturbance_year` is `NULL`. Raster masks are resampled to `raster_stack` if their CRS, resolution, or extent differ. Vector masks are reprojected to the raster CRS if needed.

`detect`
: Character string specifying the automatic detection method. Must be either `"difference"` or `"landtrendr"`. The argument is ignored when `disturbance_year` is supplied.

Details

When `disturbance_year` is `NULL`, the function calculates a spatial mean time series from `raster_stack`, optionally restricted to `mask_layer`.

Two automatic detection methods are available:

- `"difference"`: identifies the year following the largest negative year-on-year change in the spatial mean spectral index.
- `"landtrendr"`: identifies the year following the most negative local OLS slope calculated over a short moving window. The window contains up to four consecutive observations (three-year span), following a LandTrendr-style local trend approach. This option is not an implementation of the LandTrendr segmentation algorithm.

If `mask_layer` is supplied and masking produces only `NA` values, a warning is issued and the original unmasked raster is used for detection.

The detected year corresponds to the observation immediately following the largest negative change or the most negative local slope.

The LandTrendr-style slope is computed against positions in the moving window, not calendar years. For uniformly spaced annual data (the intended use), the two are equivalent.

The user-provided `disturbance_year` is returned as-is. Callers are responsible for ensuring that it matches a value in `years` if subsequent functions depend on that.

Value

A list with two elements:

event_year Detected or user-provided disturbance year.

mask The supplied disturbance mask, if any.

References

Kennedy, R. E., Yang, Z., & Cohen, W. B. (2010). Detecting trends in forest disturbance and recovery using yearly Landsat time series: 1. LandTrendr – Temporal segmentation algorithms. *Remote Sensing of Environment*, 114(12), 2897–2910. doi:10.1016/j.rse.2010.07.008

Examples

```
library(terra)

# Small synthetic time series with a clear decrease in 2021
r <- rast(
  nrows = 2,
  ncols = 2,
  nlyrs = 6
)

values(r) <- rep(
  c(0.60, 0.20, 0.30, 0.38, 0.45, 0.50),
  each = 4
)

names(r) <- paste0(
  "Index_",
  2020:2025
)

# Automatic detection using year-on-year differences
det_diff <- detect_disturbance(
  raster_stack = r,
  years = 2020:2025,
  detect = "difference"
)

det_diff

# User-provided disturbance year
det_user <- detect_disturbance(
  raster_stack = r,
  years = 2020:2025,
  disturbance_year = 2021
)

det_user




# LandTrendr-style local slope detection
det_lt <- detect_disturbance(
  raster_stack = r,
  years = 2020:2025,
  detect = "landtrendr"
)

det_lt

# Detection restricted to an sf polygon
library(sf)
```

```
p <- st_sfc(
  st_polygon(list(rbind(
    c(0, 0), c(1, 0), c(1, 1), c(0, 1), c(0, 0)
  ))),
  crs = crs(r)
)

p <- st_sf(id = 1, geometry = p)

det_masked <- detect_disturbance(
  raster_stack = r,
  years = 2020:2025,
  mask_layer = p,
  detect = "difference"
)

det_masked
```

`fit_recovery_model` *Fit Spectral Recovery Model with Optional Model Fit*

Description

Plots mean spectral recovery trajectories following disturbance. Optionally fits parametric recovery models and overlays fitted curves.

Usage

```
fit_recovery_model(
  raster_stack,
  years,
  event_year,
  mask_layer = NULL,
  index_name = "Spectral index",
  ci = "sd",
  model = NULL
)
```

Arguments

`raster_stack`	A `terra::SpatRaster` of yearly spectral index layers.
`years`	Numeric vector of years. Its length must equal the number of raster layers.
`event_year`	Numeric disturbance year.
`mask_layer`	Optional spatial mask. Can be a `terra::SpatRaster`, `terra::SpatVector`, or an `sf` object defining regions. If an `sf` or `SpatVector` object with multiple features is supplied, a separate trajectory is computed for each feature.

index_name	Character. Y-axis label. Default "Spectral index".
ci	Either "sd" to show plus/minus one standard deviation around the mean, or FALSE to suppress the error bars.
model	NULL to skip model fitting, a single model name ("exponential", "logistic", or "gompertz"), or "compare" to fit all three and rank them by AIC.

Details

Model fitting pools all mask regions into a single mean trajectory. To fit separate models per region, call this function once per region.

When model is a single model name, the fitted curve is overlaid on the plot and an equation label is added. When model = "compare", all three models are fitted, but no curve is overlaid; the metrics table includes AIC and BIC for ranking. Models that fail to converge are skipped with a warning.

The fitted trajectory consists of the layers strictly after the disturbance year, consistent with the other functions in the package.

When mask_layer = NULL, the standard deviation shown by the error bars reflects the spatial spread of all pixels in the raster, including unburned areas. To plot the spread within a specific region, supply mask_layer.

Value

A list with two elements:

plot A ggplot2 object.

metrics A data frame of fit statistics, or an empty data frame when model = NULL.

Examples

```
library(terra)

r <- rast(nrows = 4, ncols = 4, nlyrs = 6)
values(r) <- rep(c(0.60, 0.20, 0.30, 0.38, 0.45, 0.50), each = 16)
names(r) <- paste0("Index_", 2020:2025)

# Trajectory only, no model fit
out <- fit_recovery_model(
  raster_stack = r,
  years = 2020:2025,
  event_year = 2021
)

out$plot




# Fit a single exponential recovery model
fit_exp <- fit_recovery_model(
  raster_stack = r,
```

```
  years = 2020:2025,
  event_year = 2021,
  model = "exponential"
)

fit_exp$plot
fit_exp$metrics

# Compare exponential, logistic, and Gompertz
fit_cmp <- fit_recovery_model(
  raster_stack = r,
  years = 2020:2025,
  event_year = 2021,
  model = "compare"
)

fit_cmp$metrics
```

`fit_recovery_model_core`

Fit Core Recovery Model

Description

Fits a parametric recovery model (exponential, logistic, or Gompertz) to a mean post-disturbance spectral trajectory using bounded nonlinear least squares. Returns `NULL` if the model fails to converge or if the fitted parameters are invalid.

Usage

```
fit_recovery_model_core(t, y, model = c("exponential", "logistic", "gompertz"))
```

Arguments

`t`	Numeric vector of time since disturbance.
`y`	Numeric vector of mean spectral index values corresponding to `t`.
`model`	Character. One of `"exponential"`, `"logistic"`, or `"gompertz"`.

Details

The exponential model represents recovery as an asymptotic process:

$$y(t) = A(1 - exp(-kt))$$

where `A` is the asymptotic recovery level and `k` is the recovery-rate parameter. Exponential formulations have been used to characterize post-fire vegetation recovery from satellite time series (Gouveia et al., 2010) and post-fire NBR recovery (Chompuchan and Lin, 2017).

The logistic model is defined as:

$$y(t) = A/(1 + exp(-k(t - x0)))$$

where `A` is the asymptotic level, `k` is the rate parameter, and `x0` is the inflection point.

The Gompertz model is defined as:

$$y(t) = Aexp(-exp(-k(t - x0)))$$

where `A` is the asymptotic level, `k` is the rate parameter, and `x0` is the inflection point. Gompertz functions have been used to characterize vegetation biomass recovery following fire.

The logistic and Gompertz formulations are provided as alternative nonlinear representations of recovery trajectories. Model selection should be based on the characteristics and fit of the observed recovery trajectory.

Bounds on fitted parameters are `A` in `[0, 2]`, `k` in `[0.001, 5]`, and `x0` in `[0, 30]`. These are appropriate for normalized difference spectral indices such as NBR or NDVI in `[-1, 1]`. If the input `y` falls outside these bounds, the fit will fail and `NULL` is returned.

The function returns `NULL` (rather than raising an error) when the model cannot be fitted or when fitted parameters fall outside plausible ranges, so that it can be used safely inside loops. Callers should check `is.null(fit)` before accessing list elements.

The `$fit` element of the returned list is the raw `nls` object, which supports `predict()` and `summary()`. Convenience extracts (`$coefficients`, `$vcov`, `$sigma`) are also provided.

Value

A list containing:

model The fitted model name.

coefficients Estimated model coefficients.

sigma Residual standard error estimated from the fitted model.

vcov Variance-covariance matrix of the estimated coefficients, when available.

data Data used for model fitting after removal of non-finite observations.

fit The fitted `nls` model object.

If the model cannot be fitted or the fitted parameters are invalid, `NULL` is returned.

References

Chompuchan, C., & Lin, C.-Y. (2017). Assessment of forest recovery at the deciduous forest in a complex terrain using Landsat time series. *Ecological Indicators*, 79, 249–259. doi:10.1016/j.ecolind.2017.04.038

Gouveia, C., DaCamara, C. C., & Trigo, R. M. (2010). Post-fire vegetation recovery in Portugal based on spot/vegetation data. *Natural Hazards and Earth System Sciences*, 10(4), 673–684. doi:10.5194/nhess106732010

Examples

```
# Simulated exponential recovery trajectory
t <- 0:5
y <- 0.70 * (1 - exp(-0.6 * t))

fit <- fit_recovery_model_core(
  t = t,
  y = y,
  model = "exponential"
)

fit$coefficients




# Logistic and Gompertz alternatives
t2 <- 0:6
y2 <- 0.70 / (1 + exp(-0.8 * (t2 - 2)))

fit_log <- fit_recovery_model_core(
  t = t2,
  y = y2,
  model = "logistic"
)

fit_log$coefficients

y3 <- 0.70 * exp(-exp(-0.8 * (t2 - 2)))

fit_gom <- fit_recovery_model_core(
  t = t2,
  y = y3,
  model = "gompertz"
)

fit_gom$coefficients
```

`predict_recovery_raster`

Predict Spectral Recovery Raster

Description

Predicts post-disturbance spectral recovery pixel by pixel using a fitted recovery model. For each pixel, a parametric model is fitted to the post-disturbance trajectory, then extrapolated over a user-defined horizon.

Usage

```
predict_recovery_raster(
  raster_stack,
  years,
  event_year,
  model = c("exponential", "logistic", "gompertz"),
  horizon = 5,
  mask_layer = NULL,
  max_abs = 2
)
```

Arguments

raster_stack	A terra::SpatRaster of yearly spectral index layers.
years	Numeric vector of calendar years corresponding to raster layers. Its length must equal the number of raster layers.
event_year	Numeric disturbance year.
model	Character. One of "exponential", "logistic", or "gompertz".
horizon	Integer. Number of years to predict after the last observation.
mask_layer	Optional spatial mask. Can be a terra::SpatRaster, terra::SpatVector, or an sf object defining pixels to include. Raster masks are resampled to raster_stack if their CRS, resolution, or extent differ. Vector masks are reprojected to the raster CRS if needed.
max_abs	Numeric. Maximum allowed absolute predicted value. Predictions exceeding this threshold are treated as failed fits and returned as NA. Default is 2.

Details

The fitted trajectory consists of the layers strictly after the disturbance year. The time axis is defined in years since the disturbance. A pixel is skipped (predicted as NA) if fewer than four finite observations are available, if the model fails to converge, or if the predicted values fall outside the plausible range specified by max_abs.

Multi-layer output from terra::app() requires **terra** >= 1.5-0. The package DESCRIPTION declares terra >= 1.8, so this requirement is always satisfied.

Value

A terra::SpatRaster with one layer per prediction year, named Year_<calendar year>.

Examples

```
library(terra)

r <- rast(nrows = 4, ncols = 4, nlyrs = 6)
values(r) <- rep(c(0.60, 0.20, 0.30, 0.38, 0.45, 0.50), each = 16)
names(r) <- paste0("Index_", 2020:2025)
```

```
pred <- predict_recovery_raster(
  raster_stack = r,
  years = 2020:2025,
  event_year = 2021,
  model = "exponential",
  horizon = 3
)

pred




# Logistic model, prediction horizon of five years
pred_log <- predict_recovery_raster(
  raster_stack = r,
  years = 2020:2025,
  event_year = 2021,
  model = "logistic",
  horizon = 5
)

pred_log
```

---

preprocess_raster *Preprocess Raster Data for Spectral Recovery Metrics*

Description

Aggregates monthly raster stacks to yearly values and optionally applies a spatial mask. The resulting yearly stack is intended as input to `compute_recovery_metrics`.

Usage

```
preprocess_raster(
  raster_stack,
  data_type = c("yearly", "monthly"),
  months_to_aggregate = NULL,
  years,
  mask_layer = NULL,
  fun = "mean"
)
```

Arguments

`raster_stack`	A `terra::SpatRaster`. Yearly, monthly, or pre-aggregated stack.
`data_type`	Character. Either `"yearly"` or `"monthly"`.

`months_to_aggregate`
: Numeric vector of months to aggregate. Used only when `data_type == "monthly"`.

`years`
: Numeric vector of years corresponding to raster layers.

`mask_layer`
: Optional spatial mask. Can be a `terra::SpatRaster`, `terra::SpatVector`, or an `sf` object. Vector masks are reprojected to the raster CRS if needed, and raster masks are resampled to the raster grid if their CRS, resolution, or extent differ.

`fun`
: Character. Aggregation function, either `"mean"` or `"median"`. Default is `"mean"`.

Details

When `data_type = "monthly"`, layer names must contain a year and month in the form `"YYYY_MM"` or `"YYYY-MM"` (for example, `"2020_01"`). Layers matching each requested year and the months in `months_to_aggregate` are aggregated with `fun` to produce a single yearly raster. Years with no matching layers are silently skipped; an error is raised if no year matches at all.

The mask is applied after aggregation. Raster masks are aligned to `raster_stack` by resampling if their grid differs.

Value

A `terra::SpatRaster` containing one layer per year, ready for metric computation.

Examples

```
library(terra)

# Yearly input -- no aggregation required
r <- rast(nrows = 2, ncols = 2, nlyrs = 3)
values(r) <- rep(c(0.5, 0.6, 0.7), each = 4)
names(r) <- c("2020", "2021", "2022")

yearly <- preprocess_raster(
  raster_stack = r,
  data_type = "yearly",
  years = 2020:2022
)
yearly

# Mask with a SpatVector (no sf dependency)
v <- vect(
  "POLYGON ((0 0, 1 0, 1 1, 0 1, 0 0))",
  crs = crs(r)
)
masked <- preprocess_raster(
  raster_stack = r,
  data_type = "yearly",
  years = 2020:2022,
  mask_layer = v
)
masked
```

```
# Monthly input -- aggregate selected months to yearly means
rm <- rast(nrows = 2, ncols = 2, nlyrs = 6)
values(rm) <- rep(c(0.5, 0.6, 0.7, 0.4, 0.5, 0.6), each = 4)
names(rm) <- c(
  "2020_06", "2020_07", "2020_08",
  "2021_06", "2021_07", "2021_08"
)

agg <- preprocess_raster(
  raster_stack = rm,
  data_type = "monthly",
  months_to_aggregate = 6:8,
  years = 2020:2021,
  fun = "mean"
)
agg




# Example with a spatial mask (sf polygon)
library(sf)

p <- st_sfc(
  st_polygon(list(rbind(
    c(0, 0), c(1, 0), c(1, 1), c(0, 1), c(0, 0)
  ))),
  crs = crs(r)
)
p <- st_sf(id = 1, geometry = p)

masked_sf <- preprocess_raster(
  raster_stack = r,
  data_type = "yearly",
  years = 2020:2022,
  mask_layer = p
)
masked_sf
```

Index

classify_disturbance_severity, 2
classify_recovery_typology, 5
compute_recovery_metrics, 8, *25*
compute_recovery_time, 12, *14, 15*
compute_recovery_time_uncertainty, *6*, 14

detect_disturbance, 16

fit_recovery_model, 19
fit_recovery_model_core, 21

predict_recovery_raster, 23
preprocess_raster, 25