Package 'grec'

May 8, 2026

Type Package

Title Gradient-Based Recognition of Spatial Patterns in Environmental Data

Version 1.6.3

Date 2025-12-05

URL https://github.com/LuisLauM/grec

BugReports https://github.com/LuisLauM/grec/issues

Maintainer Wencheng Lau-Medrano <luis.laum@gmail.com>

Description Provides algorithms for detection of spatial patterns from oceanographic data using image processing methods based on Gradient Recognition.

License GPL (>= 3)

Depends R (>= 3.2.0)

Imports utils, imagine (>= 2.1.2), raster, terra, abind, lifecycle

LazyData true

RoxygenNote 7.3.3

Encoding UTF-8

NeedsCompilation no

Author Wencheng Lau-Medrano [aut, cre]

Repository CRAN

Date/Publication 2025-12-06 12:30:06 UTC

Contents

chl *Sea Surface Chlorophyll Data*

Description

Surface chlorophyll maps downloaded from ERDDAP for running examples with `grec` functions.

Usage

```
chl
```

Format

A `list` with chlorophyll information from February to April of Aqua MODIS source.

References

ERDDAP website: https://coastwatch.pfeg.noaa.gov/erddap

colPalette *Default color palette most using on environmental representations.*

Description

Vector with 2000 colors generated from `tim.colors` function.

Usage

```
colPalette
```

Format

A vector of 2000 colors in RGB format.

References

`tim.colors` from **fields** package

`detectFronts` *Apply gradient-based methodologies to environmental data*

Description

[Deprecated]

This function empowers users to analyze data from various sources, including numeric `matrix`, `arrays`, XYZ-`lists`, `SpatRasters`, or `RasterLayers`*, by applying gradient-seeking methodologies.

Usage

```
detectFronts(...)
```

Arguments

`...`	Same arguments than getGradients.

Details

Since version 1.6.0, this function has been entirely replaced by getGradients. As of version 2.0.0, `detectFronts` will no longer be available.

`getGradients.RasterLayer`
Apply gradient-based methodologies to environmental data

Description

This function empowers users to analyze data from various sources, including numeric `matrix`, `arrays`, XYZ-`lists`, `SpatRasters`, or `RasterLayers`*, by applying gradient-seeking methodologies.

Usage

```
## S3 method for class 'RasterLayer'
getGradients(x, method = "BelkinOReilly2009", intermediate = FALSE, ...)

## S3 method for class 'SpatRaster'
getGradients(x, method = "BelkinOReilly2009", intermediate = FALSE, ...)

## S3 method for class 'array'
getGradients(x, method = "BelkinOReilly2009", intermediate = FALSE, ...)

## Default S3 method:
```

```
getGradients(
  x,
  method = "BelkinOReilly2009",
  intermediate = FALSE,
  ConvolNormalization = FALSE,
  ...
)

getGradients(
  x,
 method = c("BelkinOReilly2009", "median_filter", "Agenbag2003-1", "Agenbag2003-2"),
  intermediate = FALSE,
  ConvolNormalization = FALSE,
  ...
)

## S3 method for class 'list'
getGradients(x, method = "BelkinOReilly2009", intermediate = FALSE, ...)

## S3 method for class 'matrix'
getGradients(x, method = "BelkinOReilly2009", intermediate = FALSE, ...)
```

Arguments

`x`	An object of class `matrix`, `array`, XYZ `list`, `SpatRaster` or `RasterLayer`*. See 'Details.'
`method`	`character` string indicating the method that will be used. For the available methods, see 'Details'.
`intermediate`	`logical` indicating whether to get the intermediate matrices (`TRUE`) or just the final one (`FALSE`).
`...`	Extra arguments that will depend on the selected method. See Details.
`ConvolNormalization`	`logical` indicating if convolutions will perform a previous normalization (`FALSE` by default). See Details.

Details

The **grec** package collaborates with the **imagine** package to execute and apply image processing algorithms for identifying oceanic gradients. **imagine** furnishes the foundational algorithms, developed efficiently utilizing C++ tools. Conversely, **grec** oversees the utilization of these coding instruments in the context of oceanic gradient recognition and handles the development of input/output methods. In this context, the available methods offered by **grec** are contingent on the installed **grec**-**imagine** versions.

(*) Due to the deprecation of the **raster** package, **grec** will not be supporting the use of `RasterLayer` in future versions. Instead, we encourage the use of SpatRaster-class (from the **terra** package), a more recent and actively developed method for working with raster data. This change will take effect as soon as **raster** is removed from CRAN.

Until the current version, `grec` performs four methods:

1. `BelkinOReilly2009` (default): Based on Belkin & O'Reilly (2009) article, it uses a Contextual Median Filter (CMF) for smoothing the original data before the applying of Sobel filters.
2. `median_filter`: it uses a typical median filter (MF) for smoothing the original data. It also allows the user to change the window size for median filter (3 as default).
3. `Agenbag2003-1`: Performs method 1 described on Agenbag et al. (2003) paper, based on the equation:
$$SST_{grad} = \sqrt{(T_{i+1} - T_{i-1})^2 + (T_{j+1} - T_{j-1})^2}$$
4. `Agenbag2003-2`: Performs method 2 described on Agenbag et al. (2003) paper, calculating the the standard deviation of the 3x3 neighbor area for each pixel.

The input data `x` can be represented in various formats to accommodate different data sources. It can be provided as a single numeric `matrix` extracted from an environmental map. Alternatively, it can be represented as a three-dimensional XYZ `list`, where `X` contains a vector of longitudes, `Y` contains a vector of latitudes, and `Z` is a matrix of dimensions `length(x$X) x length(x$Y)`. Additionally, it can be specified as an array, `SpatRaster`, or `RasterLayer*` object. If `x` is an `array`, it must have three dimensions: longitude (lon), latitude (lat), and time. It is not mandatory to define the dimnames. The output will maintain all the attributes of the input data.

`...` allows the (advanced) users to modify some aspects of filter application. Depending on the selected methodology, some parameters can be modified:

times `numeric`. How many times do you want to apply the filtering method?

kernelValues A `numeric` vector that will be used for convolution to detect vertical and horizontal gradients.

radius `numeric`. If **median-filter** method was selected, it allows to change the window size of the filter.

Normalization is a standard practice in convolution to maintain the range of output values consistent with the input data. This is achieved by dividing the convolution output by the absolute value of the kernel. While normalization is recommended to ensure consistent interpretation of results, it is disabled by default and can be enabled by setting the `ConvolNormalization` parameter to `TRUE`.

Finally, Belkin & O'Reilly's work suggests applying a logarithmic transformation to the gradient output. This step is not enabled by default, as it is primarily intended for chlorophyll data. Users are free to apply the transformation manually if it suits their specific needs.

Value

The output class will depend on the input (i.e. the `x` argument). For further details about structure of `SpatRaster` or `RasterLayer`, you can check the specific documentation in terra and raster respectively.

References

Belkin, I. M., & O'Reilly, J. E. (2009). An algorithm for oceanic front detection in chlorophyll and SST satellite imagery. Journal of Marine Systems, 78(3), 319-326 (doi:10.1016/j.jmarsys.2008.11.018).

Agenbag, J.J., A.J. Richardson, H. Demarcq, P. Freon, S. Weeks, and F.A. Shillington. "Estimating Environmental Preferences of South African Pelagic Fish Species Using Catch Size- and Remote Sensing Data". Progress in Oceanography 59, No 2-3 (October 2003): 275-300. (doi:10.1016/j.pocean.2003.07.004).

Examples

```
data(sst)
exampleSSTData <- list(x = sst$longitude,
                       y = sst$latitude,
                       z = sst$sst[,,1])

data(chl)
exampleChlData <- list(x = chl$longitude,
                       y = chl$latitude,
                       z = chl$chlorophyll[,,1])

# Simple application (over a XYZ list)
out_sst <- getGradients(x = exampleSSTData)
out_chl <- getGradients(x = exampleChlData)

# External transformation for chl data
out_chl$z <- log10(out_chl$z)

oldPar <- par(no.readonly = FALSE)

par(mfrow = c(2, 2), mar = rep(0, 4), oma = rep(0, 4))

image(exampleSSTData, col = colPalette, axes = FALSE)
mtext(text = "Original SST", side = 3, line = -2, adj = 0.99, cex = 1.2)

image(out_sst, col = colPalette, axes = FALSE)
mtext(text = "SST gradient", side = 3, line = -2, adj = 0.99, cex = 1.2)

image(exampleChlData, col = colPalette, axes = FALSE)
mtext(text = "Original Chlorophyll", side = 3, line = -2, adj = 0.99, cex = 1.2)

image(out_chl, col = colPalette, axes = FALSE)
mtext(text = "Chlorophyll gradient\n(log scale)", side = 3, line = -4, adj = 0.99,
      cex = 1.2)

par(oldPar)
```

---

sst *Sea Surface Temperature Data*

Description

SST maps downloaded from ERDDAP for running examples with `grec` functions.

Usage

```
sst
```

Format

A `list` with SST information from February to April of Aqua MODIS source.

References

ERDDAP website: https://coastwatch.pfeg.noaa.gov/erddap

Index