Package 'dtwSat'

September 22, 2023

Type Package

Title Time-Weighted Dynamic Time Warping for Satellite Image Time Series Analysis

Version 1.0.0

Date 2023-09-22

Description Provides a robust approach to land use mapping using multi-dimensional (multi-band) satellite image time series. By leveraging the Time-Weighted Dynamic Time Warping (TWDTW) distance metric in tandem with a 1 Nearest-Neighbor (1-NN) Classifier, this package offers functions to produce land use maps based on distinct seasonality patterns, commonly observed in the phenological cycles of vegetation. The approach is described in Maus et al. (2016) <doi:10.1109/JSTARS.2016.2517118> and Maus et al. (2019) <doi:10.18637/jss.v088.i05>. A primary advantage of TWDTW is its capability to handle irregularly sampled and noisy time series,
while also requiring minimal training sets. The package includes tools for training the 1-NN-TWDTW model,
visualizing temporal patterns, producing land use maps, and visualizing the results.

License GPL (>= 3)

URL https://github.com/vwmaus/dtwSat/

BugReports https://github.com/vwmaus/dtwSat/issues/

Maintainer Victor Maus <vwmaus1@gmail.com>

VignetteBuilder knitr

Encoding UTF-8

RoxygenNote 7.2.3

Depends twdtw, sf, stars, ggplot2

Imports mgcv, stats, tidyr, proxy

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

Collate 'plot.R' 'predict.R' 'prepare_time_series.R' 'shift_dates.R' 'train.R' 'zzz.R'

NeedsCompilation no

Author Victor Maus [aut, cre] (<https://orcid.org/0000-0002-7385-4723>),
Marius Appel [ctb] (<https://orcid.org/0000-0001-5281-3896>),
Nikolas Kuschnig [ctb] (<https://orcid.org/0000-0002-6642-2543>),
Toni Giorgino [ctb] (<https://orcid.org/0000-0002-6642-2543>)

Repository CRAN

Date/Publication 2023-09-22 07:20:09 UTC

R topics documented:

`get_time_series_freq` *Compute the Most Common Sampling Frequency across all observations*

Description

This function calculates the most common difference between consecutive time points. This can be useful for determining the aproximate sampling frequency of the time series data.

Usage

```
get_time_series_freq(x)
```

Arguments

x	A data frame including a column called ‘observations“ with the time series

Value

A difftime object representing the most common time difference between consecutive samples.

plot.twdtw_knn1 *Plot Patterns from twdtw-knn1 model*

Description

This function visualizes time series patterns from the "twdtw_knn1" model. It produces a multi-faceted plot, where each facet represents a different time series label from the model's data. Within each facet, different bands or indices (attributes) are plotted as distinct lines, differentiated by color.

Usage

```
## S3 method for class 'twdtw_knn1'
plot(x, bands = NULL, ...)
```

Arguments

x	A model of class "twdtw_knn1".
bands	A character vector specifying the bands or indices to plot. If NULL (default), all available bands or indices in the data will be plotted.
...	Additional arguments passed to ggplot. Currently not used.

Value

A ggplot object displaying the time series patterns.

See Also

twdtw_knn1

Examples

```
## Not run:

# Read training samples
samples_path <-
  system.file("mato_grosso_brazil/samples.gpkg", package = "dtwSat")

samples <- st_read(samples_path, quiet = TRUE)

# Get satellite image time sereis files
tif_path <- system.file("mato_grosso_brazil", package = "dtwSat")
tif_files <- dir(tif_path, pattern = "\\.tif$", full.names = TRUE)

# Get acquisition dates
acquisition_date <- regmatches(tif_files, regexpr("[0-9]{8}", tif_files))
acquisition_date <- as.Date(acquisition_date, format = "%Y%m%d")

# Create a 3D datacube
```

```
dc <- read_stars(tif_files,
                 proxy = FALSE,
                 along = list(time = acquisition_date),
                 RasterIO = list(bands = 1:6))
dc <- st_set_dimensions(dc, 3, c("EVI", "NDVI", "RED", "BLUE", "NIR", "MIR"))
dc <- split(dc, c("band"))

# Create a knn1-twdtw model
m <- twdtw_knn1(x = dc,
                y = samples,
                cycle_length = 'year',
                time_scale = 'day',
                time_weight = c(steepness = 0.1, midpoint = 50),
                formula = band ~ s(time))

print(m)

# Visualize model patterns
plot(m)

# Classify satellite images
system.time(lu <- predict(dc, model = m))

# Visualise land use classification
ggplot() +
  geom_stars(data = lu) +
  theme_minimal()


## End(Not run)
```

`predict.twdtw_knn1` *Predict using the twdtw_knn1 model*

Description

This function predicts the classes of new data using the Time Warped Dynamic Time Warping (TWDTW) method with a 1-nearest neighbor approach. The prediction is based on the minimum TWDTW distance to the known patterns stored in the `twdtw_knn1` model.

Usage

```
## S3 method for class 'twdtw_knn1'
predict(object, newdata, ...)
```

Arguments

`object`	A `twdtw_knn1` model object generated by the `twdtw_knn1` function.
`newdata`	A data frame or similar object containing the new observations (time series data) to be predicted.

... Additional arguments passed to the twdtw function. If provided, they will overwrite twdtw arguments previously passed to twdtw_knn1.

Value

A vector of predicted classes for the newdata.

See Also

twdtw_knn1

Examples

```
## Not run:

# Read training samples
samples_path <-
  system.file("mato_grosso_brazil/samples.gpkg", package = "dtwSat")

samples <- st_read(samples_path, quiet = TRUE)

# Get satellite image time sereis files
tif_path <- system.file("mato_grosso_brazil", package = "dtwSat")
tif_files <- dir(tif_path, pattern = "\\.tif$", full.names = TRUE)

# Get acquisition dates
acquisition_date <- regmatches(tif_files, regexpr("[0-9]{8}", tif_files))
acquisition_date <- as.Date(acquisition_date, format = "%Y%m%d")

# Create a 3D datacube
dc <- read_stars(tif_files,
                 proxy = FALSE,
                 along = list(time = acquisition_date),
                 RasterIO = list(bands = 1:6))
dc <- st_set_dimensions(dc, 3, c("EVI", "NDVI", "RED", "BLUE", "NIR", "MIR"))
dc <- split(dc, c("band"))

# Create a knn1-twdtw model
m <- twdtw_knn1(x = dc,
                y = samples,
                cycle_length = 'year',
                time_scale = 'day',
                time_weight = c(steepness = 0.1, midpoint = 50),
                formula = band ~ s(time))

print(m)

# Visualize model patterns
plot(m)

# Classify satellite images
system.time(lu <- predict(dc, model = m))
```

```
# Visualise land use classification
ggplot() +
  geom_stars(data = lu) +
  theme_minimal()


## End(Not run)
```

`prepare_time_series` *Prepare a Time Series Tibble from a 2D stars Object with Bands and Time Attributes*

Description

This function reshapes a data frame, which has been converted from a stars object, into a nested wide tibble format. The stars object conversion often results in columns named in formats like "band.YYYY.MM.DD", "XYYYY.MM.DD.band", or "YYYY.MM.DD.band".

Usage

```
prepare_time_series(x)
```

Arguments

`x`	A data frame derived from a stars object containing time series data in wide format. The column names should adhere to one of the following formats: "band.YYYY.MM.DD", "XYYYY.MM.DD.band", or "YYYY.MM.DD.band".

Value

A nested tibble in wide format. Each row of the tibble corresponds to a unique 'ts_id' that maintains the order from the original stars object. The nested structure contains observations (time series) for each 'ts_id', including the 'time' of each observation, and individual bands are presented as separate columns.

`pretty_arguments` *Print Pretty Arguments*

Description

Display a list of arguments of a given function in a human-readable format.

Usage

```
pretty_arguments(args)
```

Arguments

args	A list of named arguments to display.

Value

Invisible NULL. The function is mainly used for its side effect of printing.

Examples

```
## Not run:
pretty_arguments(formals(twdtw_knn1))

## End(Not run)
```

---

print.twdtw_knn1 *Print method for objects of class twdtw_knn1*

Description

This method provides a structured printout of the important components of a twdtw_knn1 object.

Usage

```
## S3 method for class 'twdtw_knn1'
print(x, ...)
```

Arguments

x	An object of class twdtw_knn1.
...	ignored

Value

Invisible twdtw_knn1 object.

shift_dates *Shift Dates to Start on a Specified Origin Year*

Description

Shifts a vector of dates to start on the same day-of-year in a specified origin year while preserving the relative difference in days among the observations. This way the temporal pattern (e.g., seasonality) inherent to the original dates will also be preserved in the shifted dates.

Usage

```
shift_dates(x, origin = "1970-01-01")
```

Arguments

x	A vector of date strings or Date objects representing the sequence to shift.
origin	A date string or Date object specifying the desired origin year for the shifted dates. Default is "1970-01-01".

Details

The primary goal of this function is to align a sequence of dates based on the day-of-year in a desired origin year. This can be particularly useful for comparing or visualizing two or more time series with different absolute dates but aiming to align them based on the day-of-year or another relative metric.

Value

A vector of Date objects with the shifted dates starting on the same day-of-year in the specified origin year.

Examples

```
x <- c("2011-09-14", "2011-09-30", "2011-10-16", "2011-11-01")

shift_dates(x)
```

`twdtw_knn1` *Train a KNN-1 TWDTW model with optional GAM resampling*

Description

This function prepares a KNN-1 model with the Time Warp Dynamic Time Warping (TWDTW) algorithm. If a formula is provided, the training samples are resampled using Generalized Additive Models (GAM).

Usage

```
twdtw_knn1(
  x,
  y,
  time_weight,
  cycle_length,
  time_scale,
  formula = NULL,
  start_column = "start_date",
  end_column = "end_date",
  label_colum = "label",
  sampling_freq = NULL,
  ...
)
```

Arguments

`x`	A three-dimensional stars object (x, y, time) with bands as attributes.
`y`	An sf object with the coordinates of the training points.
`time_weight`	A numeric vector with length two (steepness and midpoint of logistic weight) or a function. See details in twdtw.
`cycle_length`	The length of the cycle, e.g. phenological cycles. Details in twdtw.
`time_scale`	Specifies the time scale for the observations. Details in twdtw.
`formula`	Either NULL or a formula to reduce samples of the same label using Generalized Additive Models (GAM). Default is `band ~ s(time)`. See details.
`start_column`	Name of the column in y that indicates the start date. Default is 'start_date'.
`end_column`	Name of the column in y that indicates the end date. Default is 'end_date'.
`label_colum`	Name of the column in y containing land use labels. Default is 'label'.
`sampling_freq`	The time frequency for sampling, including the unit (e.g., '16 day'). If NULL, the function will infer the frequency. This parameter is only used if a formula is provided.
`...`	Additional arguments passed to the gam function and to twdtw function.

Details

If `formula` is NULL, the KNN-1 model will retain all training samples. If a formula is passed (e.g., `band ~ s(time)`), then samples of the same label (land cover class) will be resampled using GAM. Resampling can significantly reduce prediction processing time.

Value

A 'twdtw_knn1' model containing the trained model information and the data used.

Examples

```
## Not run:

# Read training samples
samples_path <-
  system.file("mato_grosso_brazil/samples.gpkg", package = "dtwSat")

samples <- st_read(samples_path, quiet = TRUE)

# Get satellite image time sereis files
tif_path <- system.file("mato_grosso_brazil", package = "dtwSat")
tif_files <- dir(tif_path, pattern = "\\.tif$", full.names = TRUE)

# Get acquisition dates
acquisition_date <- regmatches(tif_files, regexpr("[0-9]{8}", tif_files))
acquisition_date <- as.Date(acquisition_date, format = "%Y%m%d")

# Create a 3D datacube
dc <- read_stars(tif_files,
                 proxy = FALSE,
                 along = list(time = acquisition_date),
                 RasterIO = list(bands = 1:6))
dc <- st_set_dimensions(dc, 3, c("EVI", "NDVI", "RED", "BLUE", "NIR", "MIR"))
dc <- split(dc, c("band"))

# Create a knn1-twdtw model
m <- twdtw_knn1(x = dc,
                y = samples,
                cycle_length = 'year',
                time_scale = 'day',
                time_weight = c(steepness = 0.1, midpoint = 50),
                formula = band ~ s(time))

print(m)

# Visualize model patterns
plot(m)

# Classify satellite images
system.time(lu <- predict(dc, model = m))

# Visualise land use classification
```

```
ggplot() +
  geom_stars(data = lu) +
  theme_minimal()


## End(Not run)
```

Index