

Package ‘dtwSat’

October 13, 2022

Type Package

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

Version 0.2.8

Date 2022-10-11

Description Provides an implementation of the Time-Weighted Dynamic Time Warping (TWDTW) method for land cover mapping. TWDTW computes the similarity between satellite image time series with a set of known temporal patterns (e.g. phenological cycles of the vegetation). ‘dtwSat’ offers the user methods to create temporal patterns for land cover types, perform TWDTW analysis for satellite datasets, visualize the results of the analysis, produce land cover maps, create temporal plots for land cover change, and compute accuracy metrics.

Depends R (>= 3.5.0), zoo, raster, ggplot2

Imports methods, rgdal, dtw, proxy, scales, reshape2, grDevices, RColorBrewer, plyr, stats, sp, lubridate, caret, mgcv, xtable, Rdpack, data.table, foreach

Suggests gridExtra, grid, png, Hmisc, rbenchmark, doParallel, knitr, rmarkdown

License GPL (>= 3) | file LICENSE

URL <https://www.victor-maus.com/dtwSat/>,
<https://github.com/vwmaus/dtwSat/>

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

Maintainer Victor Maus <vwmaus1@gmail.com>

LazyData true

VignetteBuilder knitr

Encoding UTF-8

RoxygenNote 7.2.1

Collate 'class-crossValidation.R' 'class-twdtwRaster.R'
 'class-twdtwAssessment.R' 'class-twdtwTimeSeries.R'
 'class-twdtwMatches.R' 'createPatterns.R' 'data.R' 'dtw.R'
 'dwtSat.R' 'getInternals.R' 'getMatchingDates.R'
 'getTimeSeries.R' 'linearWeight.R' 'logisticWeight.R'
 'methods.R' 'miscellaneous.R' 'plot.R' 'plotAccuracy.R'
 'plotAdjustedArea.R' 'plotAlignments.R' 'plotArea.R'
 'plotChanges.R' 'plotClassification.R' 'plotCostMatrix.R'
 'plotDistance.R' 'plotMapSamples.R' 'plotMaps.R'
 'plotMatches.R' 'plotPaths.R' 'plotPatterns.R'
 'plotTimeSeries.R' 'resampleTimeSeries.R' 'subset.R' 'twdtw.R'
 'twdtwApply.R' 'twdtwAssess.R' 'twdtwClassify.R'
 'twdtwCrossValidate.R' 'twdtwDist.R' 'twdtwXtable.R'
 'twdtw_reduce_time.R' 'utils.R' 'zzz.R'

RdMacros Rdpack

NeedsCompilation yes

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 2022-10-11 16:02:32 UTC

R topics documented:

createPatterns	3
dtwSat	5
get	6
getDatesFromDOY	7
getTimeSeries	8
linearWeight	10
logisticWeight	11
MOD13Q1.MT.yearly.patterns	12
MOD13Q1.patterns.list	13
MOD13Q1.ts	14
MOD13Q1.ts.labels	15
MOD13Q1.ts.list	15
plot	16
plotAccuracy	17
plotAdjustedArea	19
plotAlignments	20
plotArea	21
plotChanges	23
plotClassification	25
plotCostMatrix	27
plotDistance	28

plotMaps	29
plotMapSamples	31
plotMatches	34
plotPaths	35
plotPatterns	36
plotTimeSeries	37
resampleTimeSeries	39
shiftDates	40
subset	41
twdtwApply	43
twdtwAssess	48
twdtwAssessment-class	53
twdtwClassify	55
twdtwCrossValidate	66
twdtwCrossValidation-class	67
twdtwMatches-class	69
twdtwRaster-class	72
twdtwReduceTime	75
twdtwTimeSeries-class	77
twdtwXtable	80

Index**83**

createPatterns	<i>Create patterns</i>
----------------	------------------------

Description

Create temporal patterns from objects of class `twdtwTimeSeries`.

Usage

```
createPatterns(
  x,
  from = NULL,
  to = NULL,
  freq = 1,
  attr = NULL,
  split = TRUE,
  formula,
  ...
)

## S4 method for signature 'twdtwTimeSeries'
createPatterns(
  x,
  from = NULL,
  to = NULL,
```

```

    freq = 1,
    attr = NULL,
    split = TRUE,
    formula,
    ...
  )

```

Arguments

x	an object of class <code>twdtwTimeSeries</code> .
from	A character or <code>Dates</code> object in the format "yyyy-mm-dd". If not provided it is equal to the smallest date of the first element in x. See details.
to	A <code>character</code> or <code>Dates</code> object in the format "yyyy-mm-dd". If not provided it is equal to the greatest date of the first element in x. See details.
freq	An integer. The sampling frequency of the output patterns.
attr	A vector character or numeric. The attributes in x to be used. If not declared the function uses all attributes.
split	A logical. If TRUE the samples are split by label. If FALSE all samples are set to the same label.
formula	A formula. Argument to pass to <code>gam</code> .
...	other arguments to pass to the function <code>gam</code> in the package <code>mgcv</code> .

Details

The hidden assumption is that the temporal pattern is a cycle the repeats itself within a given time interval. Therefore, all time series samples in x are aligned with each other, keeping their respective sequence of days of the year. The function fits a Generalized Additive Model (GAM) to the aligned set of samples.

Value

an object of class `twdtwTimeSeries`

Author(s)

Victor Maus, <vwmaus1@gmail.com>

References

- Maus V, Camara G, Appel M, Pebesma E (2019). "dtwSat: Time-Weighted Dynamic Time Warping for Satellite Image Time Series Analysis in R." *Journal of Statistical Software*, **88**(5), 1–31. doi:10.18637/jss.v088.i05.
- Maus V, Camara G, Cartaxo R, Sanchez A, Ramos FM, de Queiroz GR (2016). "A Time-Weighted Dynamic Time Warping Method for Land-Use and Land-Cover Mapping." *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, **9**(8), 3729–3739. doi:10.1109/JSTARS.2016.2517118.

See Also

[twdtwMatches-class](#), [twdtwTimeSeries-class](#), [getTimeSeries](#), and [twdtwApply](#)

Examples

```
# Creating patterns from objects of class twdtwTimeSeries
evi = brick(system.file("lucc_MT/data/evi.tif", package="dtwSat"))
ndvi = brick(system.file("lucc_MT/data/ndvi.tif", package="dtwSat"))
timeline = scan(system.file("lucc_MT/data/timeline", package="dtwSat"), what="date")
rts = twdtwRaster(evi, ndvi, timeline=timeline)

# Read field samples
## Not run:
field_samples = read.csv(system.file("lucc_MT/data/samples.csv", package="dtwSat"))
prj_string = scan(system.file("lucc_MT/data/samples_projection", package="dtwSat"),
                  what = "character")

# Extract time series
ts = getTimeSeries(rts, y = field_samples, proj4string = prj_string)

# Create temporal patterns
patt = createPatterns(x=ts, from="2005-09-01", to="2006-09-01", freq=8, formula = y~s(x))

# Plot patterns
autoplot(patt[[1]], facets = NULL) + xlab("Time") + ylab("Value")

## End(Not run)
```

dtwSat

Time-Weighted Dynamic Time Warping for Satellite Image Time Series

Description

Provides an implementation of the Time-Weighted Dynamic Time Warping (TWDTW) method for land use and land cover mapping using satellite image time series (Maus et al. 2016; Maus et al. 2019). TWDTW is based on the Dynamic Time Warping technique and has achieved high accuracy for land use and land cover classification using satellite data. The method is based on comparing unclassified satellite image time series with a set of known temporal patterns (e.g. phenological cycles associated with the vegetation). Using 'dtwSat' the user can build temporal patterns for land cover types, apply the TWDTW analysis for satellite datasets, visualize the results of the time series analysis, produce land cover maps, and create temporal plots for land cover change analysis.

Author(s)

Victor Maus, <vwmaus1@gmail.com>

References

Maus V, Camara G, Appel M, Pebesma E (2019). “dtwSat: Time-Weighted Dynamic Time Warping for Satellite Image Time Series Analysis in R.” *Journal of Statistical Software*, **88**(5), 1–31. doi:10.18637/jss.v088.i05.

Maus V, Camara G, Cartaxo R, Sanchez A, Ramos FM, de Queiroz GR (2016). “A Time-Weighted Dynamic Time Warping Method for Land-Use and Land-Cover Mapping.” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, **9**(8), 3729–3739. doi:10.1109/JSTARS.2016.2517118.

See Also

[twdtwApply](#)

get

Get elements from twdtwMatches objects

Description

Get elements from [twdtwMatches-class](#) objects.

Usage

```
## S4 method for signature 'twdtwMatches'
getAlignments(object, timeseries.labels = NULL, patterns.labels = NULL)
```

```
## S4 method for signature 'twdtwMatches'
getInternals(object, timeseries.labels = NULL, patterns.labels = NULL)
```

```
## S4 method for signature 'twdtwMatches'
getMatches(object, timeseries.labels = NULL, patterns.labels = NULL)
```

Arguments

`object` an object of class `twdtwMatches`.
`timeseries.labels` a vector with labels of the time series.
`patterns.labels` a vector with labels of the patterns.

Value

a list with TWDTW results or an object [twdtwTimeSeries-class](#).

Author(s)

Victor Maus, <vwmaus1@gmail.com>

References

Maus V, Camara G, Appel M, Pebesma E (2019). “dtwSat: Time-Weighted Dynamic Time Warping for Satellite Image Time Series Analysis in R.” *Journal of Statistical Software*, **88**(5), 1–31. doi:10.18637/jss.v088.i05.

Maus V, Camara G, Cartaxo R, Sanchez A, Ramos FM, de Queiroz GR (2016). “A Time-Weighted Dynamic Time Warping Method for Land-Use and Land-Cover Mapping.” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, **9**(8), 3729–3739. doi:10.1109/JSTARS.2016.2517118.

See Also

[twdtwMatches-class](#), and [twdtwApply](#)

Examples

```
# Getting patterns from objects of class twdtwMatches
patt = twdtwTimeSeries(MOD13Q1.patterns.list)
ts = twdtwTimeSeries(MOD13Q1.ts.list)
mat = twdtwApply(x=ts, y=patt, weight.fun=logisticWeight(-0.1,100),
                keep=TRUE, legacy = TRUE)
getPatterns(mat)
getTimeSeries(mat)
getAlignments(mat)
getMatches(mat)
getInternals(mat)
```

getDatesFromDOY

Get dates from year and day of the year

Description

This function retrieves the date corresponding to the given year and day of the year.

Usage

```
getDatesFromDOY(year, doy)
```

Arguments

year A vector with the years.

doy A vector with the day of the year. It must have the same length as year.

Value

A [Dates](#) object.

Author(s)

Victor Maus, <vwmaus1@gmail.com>

References

Maus V, Camara G, Appel M, Pebesma E (2019). “dtwSat: Time-Weighted Dynamic Time Warping for Satellite Image Time Series Analysis in R.” *Journal of Statistical Software*, **88**(5), 1–31. doi:10.18637/jss.v088.i05.

Maus V, Camara G, Cartaxo R, Sanchez A, Ramos FM, de Queiroz GR (2016). “A Time-Weighted Dynamic Time Warping Method for Land-Use and Land-Cover Mapping.” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, **9**(8), 3729–3739. doi:10.1109/JSTARS.2016.2517118.

See Also

[shiftDates](#)

Examples

```
year = c(2000, 2001)
doy = c(366, 365)
dates = getDatesFromDOY(year, doy)
dates
```

getTimeSeries

Get time series from twdtw objects*

Description

Get time series from objects of class twdtw*.

Usage

```
## S4 method for signature 'twdtwTimeSeries'
getTimeSeries(object, labels = NULL)

## S4 method for signature 'twdtwMatches'
getTimeSeries(object, labels = NULL)

## S4 method for signature 'twdtwMatches'
getPatterns(object, labels = NULL)

## S4 method for signature 'twdtwRaster'
getTimeSeries(object, y, labels = NULL, proj4string = NULL, id.labels = NULL)
```


Arguments

object	an object of class <code>twdtw*</code> .
labels	character vector with time series labels. For signature <code>twdtwRaster</code> this argument can be used to set the labels for each sample in <code>y</code> , or it can be combined with <code>id.labels</code> to select samples with a specific label.
y	a <code>data.frame</code> whose attributes are: longitude, latitude, the start "from" and the end "to" of the time interval for each sample. This can also be a <code>SpatialPointsDataFrame</code> whose attributes are the start "from" and the end "to" of the time interval. If missing "from" and/or "to", they are set to the time range of the object.
proj4string	projection string, see <code>CRS-class</code> . Used if <code>y</code> is a <code>data.frame</code> .
id.labels	a numeric or character with a column name from <code>y</code> to be used as sample labels. Optional.

Value

An object of class `twdtwTimeSeries`.

a list with TWDTW results or an object `twdtwTimeSeries-class`.

Author(s)

Victor Maus, <vwmaus1@gmail.com>

References

Maus V, Camara G, Appel M, Pebesma E (2019). "dtwSat: Time-Weighted Dynamic Time Warping for Satellite Image Time Series Analysis in R." *Journal of Statistical Software*, **88**(5), 1–31. doi:10.18637/jss.v088.i05.

Maus V, Camara G, Cartaxo R, Sanchez A, Ramos FM, de Queiroz GR (2016). "A Time-Weighted Dynamic Time Warping Method for Land-Use and Land-Cover Mapping." *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, **9**(8), 3729–3739. doi:10.1109/JSTARS.2016.2517118.

See Also

`twdtwRaster-class`, `twdtwTimeSeries-class`, and `twdtwMatches-class`

Examples

```
# Getting time series from objects of class twdtwTimeSeries
ts = twdtwTimeSeries(MOD13Q1.ts.list)
getTimeSeries(ts, 2)
# Getting time series from objects of class twdtwTimeSeries
ts = twdtwTimeSeries(MOD13Q1.ts.list)
patt = twdtwTimeSeries(MOD13Q1.patterns.list)
mat = twdtwApply(x=ts, y=patt, keep=TRUE, legacy=TRUE)
getTimeSeries(mat, 2)

## This example creates a twdtwRaster object and extract time series from it.
```

```

# Creating objects of class twdtwRaster with evi and ndvi time series
evi = brick(system.file("lucc_MT/data/evi.tif", package="dtwSat"))
ndvi = brick(system.file("lucc_MT/data/ndvi.tif", package="dtwSat"))
timeline = scan(system.file("lucc_MT/data/timeline", package="dtwSat"), what="date")
rts = twdtwRaster(evi, ndvi, timeline=timeline)

# Location and time range
ts_location = data.frame(longitude = -55.96957, latitude = -12.03864,
                          from = "2007-09-01", to = "2013-09-01")
prj_string = "+proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0"

# Extract time series
ts = getTimeSeries(rts, y = ts_location, proj4string = prj_string)

autoplot(ts[[1]], facets = NULL) + xlab("Time") + ylab("Value")

```

linearWeight

Linear weight function

Description

Builds a linear time weight function to compute the TWDTW local cost matrix [1].

Usage

```
linearWeight(a, b = 0)
```

Arguments

a	numeric. The slop of the line.
b	numeric. The intercept of the line.

Details

The linear `linearWeight` and `logisticWeight` weight functions can be passed to `twdtwApply` through the argument `weight.fun`. This will add a time-weight to the dynamic time warping analysis. The time weight creates a global constraint useful to analyse time series with phenological cycles of vegetation that are usually bound to seasons. In previous studies by Maus et al. (2016) the logistic weight had better results than the linear for land cover classification. See Maus et al. (2016) and Maus et al. (2019).

Value

A `function` object.

Author(s)

Victor Maus, <vwmaus1@gmail.com>

References

Maus V, Camara G, Appel M, Pebesma E (2019). “dtwSat: Time-Weighted Dynamic Time Warping for Satellite Image Time Series Analysis in R.” *Journal of Statistical Software*, **88**(5), 1–31. doi:10.18637/jss.v088.i05.

Maus V, Camara G, Cartaxo R, Sanchez A, Ramos FM, de Queiroz GR (2016). “A Time-Weighted Dynamic Time Warping Method for Land-Use and Land-Cover Mapping.” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, **9**(8), 3729–3739. doi:10.1109/JSTARS.2016.2517118.

See Also

[twdtwApply](#)

Examples

```
lin_fun = linearWeight(a=0.1)
lin_fun
```

logisticWeight	<i>Logistic weight function</i>
----------------	---------------------------------

Description

Builds a logistic time weight function to compute the TWDTW local cost matrix [1].

Usage

```
logisticWeight(alpha, beta)
```

Arguments

alpha	numeric. The steepness of logistic weight.
beta	numeric. The midpoint of logistic weight.

Details

The linear `linearWeight` and `logisticWeight` weight functions can be passed to `twdtwApply` through the argument `weight.fun`. This will add a time-weight to the dynamic time warping analysis. The time weight creates a global constraint useful to analyze time series with phenological cycles of vegetation that are usually bound to seasons. In previous studies by Maus et al. (2016) the logistic weight had better results than the linear for land cover classification. See Maus et al. (2016) and Maus et al. (2019).

Value

A `function` object.

Author(s)

Victor Maus, <vwmaus1@gmail.com>

References

Maus V, Camara G, Appel M, Pebesma E (2019). “dtwSat: Time-Weighted Dynamic Time Warping for Satellite Image Time Series Analysis in R.” *Journal of Statistical Software*, **88**(5), 1–31. doi:10.18637/jss.v088.i05.

Maus V, Camara G, Cartaxo R, Sanchez A, Ramos FM, de Queiroz GR (2016). “A Time-Weighted Dynamic Time Warping Method for Land-Use and Land-Cover Mapping.” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, **9**(8), 3729–3739. doi:10.1109/JSTARS.2016.2517118.

See Also

[twdtwApply](#)

Examples

```
log_fun = logisticWeight(alpha=-0.1, beta=100)
log_fun
```

MOD13Q1.MT.yearly.patterns

Data: Pattern time series

Description

This dataset has a list of patterns with the phenological cycle of: Water, Cotton-Fallow, Forest, Low vegetation, Pasture, Soybean-Cotton, Soybean-Maize, Soybean-Millet, Soybean-Sunflower, and Wetland. These time series are based on the MODIS product MOD13Q1 250 m 16 days (Didan 2015). The patterns were built from ground truth samples of each crop using Generalized Additive Models (GAM), see [createPatterns](#).

Usage

```
MOD13Q1.MT.yearly.patterns
```

Format

A [twdtwTimeSeries](#) object.

Author(s)

Victor Maus, <vwmaus1@gmail.com>

References

- Didan K (2015). “MOD13Q1 MODIS/Terra Vegetation Indices 16-Day L3 Global 250m SIN Grid V006 [Data set], NASA EOSDIS LP DAAC.” doi:10.5067/MODIS/MOD13Q1.006.
- Maus V, Camara G, Appel M, Pebesma E (2019). “dtwSat: Time-Weighted Dynamic Time Warping for Satellite Image Time Series Analysis in R.” *Journal of Statistical Software*, **88**(5), 1–31. doi:10.18637/jss.v088.i05.
- Maus V, Camara G, Cartaxo R, Sanchez A, Ramos FM, de Queiroz GR (2016). “A Time-Weighted Dynamic Time Warping Method for Land-Use and Land-Cover Mapping.” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, **9**(8), 3729–3739. doi:10.1109/JSTARS.2016.2517118.

See Also

For details about MOD13Q1 see (Didan 2015).

MOD13Q1.patterns.list *Data: patterns time series*

Description

This dataset has a list of patterns with the phenological cycle of: Soybean, Cotton, and Maize. These time series are based on the MODIS product MOD13Q1 250 m 16 days (Didan 2015). The patterns were built from ground truth samples of each crop using Generalized Additive Models (GAM), see [createPatterns](#).

Usage

```
MOD13Q1.patterns.list
```

Format

A named list of three [zoo](#) objects, “Soybean”, “Cotton”, and “Maize”, whose indices are [Dates](#) in the format “yyyy-mm-dd”. Each node has 6 attributes: “ndvi”, “evi”, “red”, “nir”, “blue”, and “mir”.

Author(s)

Victor Maus, <vwmaus1@gmail.com>

References

- Didan K (2015). “MOD13Q1 MODIS/Terra Vegetation Indices 16-Day L3 Global 250m SIN Grid V006 [Data set], NASA EOSDIS LP DAAC.” doi:10.5067/MODIS/MOD13Q1.006.
- Maus V, Camara G, Appel M, Pebesma E (2019). “dtwSat: Time-Weighted Dynamic Time Warping for Satellite Image Time Series Analysis in R.” *Journal of Statistical Software*, **88**(5), 1–31. doi:10.18637/jss.v088.i05.

Maus V, Camara G, Cartaxo R, Sanchez A, Ramos FM, de Queiroz GR (2016). “A Time-Weighted Dynamic Time Warping Method for Land-Use and Land-Cover Mapping.” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, **9**(8), 3729–3739. doi:10.1109/JSTARS.2016.2517118.

See Also

[MOD13Q1.ts](#), [MOD13Q1.ts.list](#), and [createPatterns](#).

For details about MOD13Q1 see (Didan 2015).

MOD13Q1.ts

Data: An example of satellite time series

Description

This dataset has a time series based on the MODIS product MOD13Q1 250 m 16 days (Didan 2015). It is an irregularly sampled time series using the real date of each pixel from “2009-08-05” to “2013-07-31”.

Usage

MOD13Q1.ts

Format

A `zoo` object, whose indices are `Dates` in the format “yyyy-mm-dd”. Each node has 6 attributes: “ndvi”, “evi”, “red”, “nir”, “blue”, and “mir”.

Author(s)

Victor Maus, <vwmaus1@gmail.com>

References

Didan K (2015). “MOD13Q1 MODIS/Terra Vegetation Indices 16-Day L3 Global 250m SIN Grid V006 [Data set], NASA EOSDIS LP DAAC.” doi:10.5067/MODIS/MOD13Q1.006.

Maus V, Camara G, Appel M, Pebesma E (2019). “dtwSat: Time-Weighted Dynamic Time Warping for Satellite Image Time Series Analysis in R.” *Journal of Statistical Software*, **88**(5), 1–31. doi:10.18637/jss.v088.i05.

Maus V, Camara G, Cartaxo R, Sanchez A, Ramos FM, de Queiroz GR (2016). “A Time-Weighted Dynamic Time Warping Method for Land-Use and Land-Cover Mapping.” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, **9**(8), 3729–3739. doi:10.1109/JSTARS.2016.2517118.

See Also

[MOD13Q1.ts.list](#), [MOD13Q1.patterns.list](#).

For details about MOD13Q1 see (Didan 2015).

MOD13Q1.ts.labels *Data: Labels of the satellite time series in MOD13Q1.ts*

Description

These labels are based on field work.

Usage

MOD13Q1.ts.labels

Format

An object of class `data.frame`, whose attributes are: the label of the crop class "label", the start of the crop period "from", and the end of the crop period "to". The dates are in the format "yyyy-mm-dd".

Author(s)

Victor Maus, <vwmaus1@gmail.com>

See Also

[MOD13Q1.ts](#).

MOD13Q1.ts.list *Data: A list of satellite time series*

Description

This dataset has a list of time series based on the MODIS product MOD13Q1 250 m 16 days (Didan 2015). It is an irregularly sampled time series using the real date of each pixel from "2009-08-05" to "2013-07-31".

Usage

MOD13Q1.ts.list

Format

A `zoo` object, whose indices are `Dates` in the format "yyyy-mm-dd". Each node has 6 attributes: "ndvi", "evi", "red", "nir", "blue", and "mir".

Author(s)

Victor Maus, <vwmaus1@gmail.com>

References

Didan K (2015). “MOD13Q1 MODIS/Terra Vegetation Indices 16-Day L3 Global 250m SIN Grid V006 [Data set], NASA EOSDIS LP DAAC.” doi:[10.5067/MODIS/MOD13Q1.006](https://doi.org/10.5067/MODIS/MOD13Q1.006).

Maus V, Camara G, Appel M, Pebesma E (2019). “dtwSat: Time-Weighted Dynamic Time Warping for Satellite Image Time Series Analysis in R.” *Journal of Statistical Software*, **88**(5), 1–31. doi:[10.18637/jss.v088.i05](https://doi.org/10.18637/jss.v088.i05).

Maus V, Camara G, Cartaxo R, Sanchez A, Ramos FM, de Queiroz GR (2016). “A Time-Weighted Dynamic Time Warping Method for Land-Use and Land-Cover Mapping.” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, **9**(8), 3729–3739. doi:[10.1109/JSTARS.2016.2517118](https://doi.org/10.1109/JSTARS.2016.2517118).

See Also

[MOD13Q1.ts](#), and [MOD13Q1.patterns.list](#).

For details about MOD13Q1 see (Didan 2015).

plot	<i>Plotting twdtw* objects</i>
------	--------------------------------

Description

Methods for plotting objects of class twdtw*.

Usage

```
## S4 method for signature 'twdtwAssessment,ANY'
plot(x, type = "area", ...)
```

```
## S4 method for signature 'twdtwCrossValidation,ANY'
plot(x, type = "crossvalidation", ...)
```

```
## S4 method for signature 'twdtwTimeSeries,ANY'
plot(x, type = "timeseries", ...)
```

```
## S4 method for signature 'twdtwMatches,ANY'
plot(x, type = "alignments", ...)
```

```
## S4 method for signature 'twdtwRaster,ANY'
plot(x, type = "maps", ...)
```

Arguments

x	An object of class twdtw*.
type	A character for the plot type: "paths", "matches", "alignments", "classification", "cost", "patterns", "timeseries", "maps", "area", "changes", and "distance".

... additional arguments to pass to plotting functions. [plotPaths](#), [plotCostMatrix](#), [plotAlignments](#), [plotMatches](#), [plotClassification](#), [plotPatterns](#), [plotTimeSeries](#), [plotMaps](#), [plotArea](#), or [plotChanges](#).

Details

Plot types :

paths: Method for plotting the minimum paths in the cost matrix of TWDTW.

matches: Method for plotting the matching points from TWDTW analysis.

alignments: Method for plotting the alignments and respective TWDTW dissimilarity measures.

classification: Method for plotting the classification of each subinterval of the time series based on TWDTW analysis.

cost: Method for plotting the internal matrices used during the TWDTW computation.

patterns: Method for plotting the temporal patterns.

timeseries: Method for plotting the temporal patterns.

Value

A [ggplot](#) object.

Author(s)

Victor Maus, <vwmaus1@gmail.com>

plotAccuracy

Plotting accuracy assessment

Description

Method for plotting accuracy assessment results.

Usage

```
plotAccuracy(  
  x,  
  perc = TRUE,  
  conf.int = 0.95,  
  time.labels = NULL,  
  category.name = NULL,  
  category.type = NULL  
)
```

Arguments

x	An object of class <code>twdtwAssessment</code> or <code>twdtwCrossValidation</code> .
perc	if TRUE shows the results in percent of area. Otherwise shows the area in the map units or km2 for no project raster. Default is TRUE.
conf.int	confidence level (0-1) for interval estimation of the population mean. For details see <code>smean.cl.normal</code> . Used if x is <code>twdtwCrossValidation</code> .
time.labels	a character or numeric for the time periods or NULL to aggregate all classified periods in the same plot. Default is NULL. Used if x is <code>twdtwAssessment</code> .
category.name	a character vector defining the class names. If NULL the class names in the object x are used. Default is NULL.
category.type	a character defining the categories type "numeric" or "letter", if NULL the class names are used. Default is NULL.

Value

A `ggplot` object.

Author(s)

Victor Maus, <vwmaus1@gmail.com>

References

Maus V, Camara G, Appel M, Pebesma E (2019). "dtwSat: Time-Weighted Dynamic Time Warping for Satellite Image Time Series Analysis in R." *Journal of Statistical Software*, **88**(5), 1–31. doi:10.18637/jss.v088.i05.

Maus V, Camara G, Cartaxo R, Sanchez A, Ramos FM, de Queiroz GR (2016). "A Time-Weighted Dynamic Time Warping Method for Land-Use and Land-Cover Mapping." *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, **9**(8), 3729–3739. doi:10.1109/JSTARS.2016.2517118.

See Also

`twdtwAssessment` and `twdtwAssess`

Examples

```
## Not run:

# See ?twdtwAssess and ?twdtwCrosValidate

plotAccuracy(x)

plotAccuracy(x, category.type="letter")

## End(Not run)
```

plotAdjustedArea *Plotting area and uncertainty*

Description

Method for plotting area and uncertainty.

Usage

```
plotAdjustedArea(  
  x,  
  perc = TRUE,  
  time.labels = NULL,  
  category.name = NULL,  
  category.type = NULL  
)
```

Arguments

x	An object of class twdtwAssessment or twdtwCrossValidation .
perc	if TRUE shows the results in percent of area. Otherwise shows the area in the map units or km2 for no project raster. Default is TRUE.
time.labels	a character or numeric for the time periods or NULL to aggregate all classified periods in the same plot. Default is NULL. Used if x is twdtwAssessment .
category.name	a character vector defining the class names. If NULL the class names in the object x are used. Default is NULL.
category.type	a character defining the categories type "numeric" or "letter", if NULL the class names are used. Default is NULL.

Value

A [ggplot](#) object.

Author(s)

Victor Maus, <vwmaus1@gmail.com>

References

Maus V, Camara G, Appel M, Pebesma E (2019). "dtwSat: Time-Weighted Dynamic Time Warping for Satellite Image Time Series Analysis in R." *Journal of Statistical Software*, **88**(5), 1–31. [doi:10.18637/jss.v088.i05](https://doi.org/10.18637/jss.v088.i05).

Maus V, Camara G, Cartaxo R, Sanchez A, Ramos FM, de Queiroz GR (2016). "A Time-Weighted Dynamic Time Warping Method for Land-Use and Land-Cover Mapping." *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, **9**(8), 3729–3739. [doi:10.1109/JSTARS.2016.2517118](https://doi.org/10.1109/JSTARS.2016.2517118).

See Also

[twdtwAssessment](#) and [twdtwAssess](#)

Examples

```
## Not run:

# See ?twdtwAssess

plotAdjustedArea(twdtw_assess)

plotAdjustedArea(twdtw_assess, category.type="letter")

## End(Not run)
```

plotAlignments	<i>Plotting alignments</i>
----------------	----------------------------

Description

Method for plotting the alignments and TWDTW dissimilarity measures.

Usage

```
plotAlignments(
  x,
  timeseries.labels = NULL,
  patterns.labels = NULL,
  attr = 1,
  threshold = Inf
)
```

Arguments

x	An object of class twdtwMatches .
timeseries.labels	the label or index of the time series. Default is 1.
patterns.labels	a vector with labels of the patterns. If not declared the function will plot the alignments for all patterns in x.
attr	An integer or character vector indicating the attribute for plotting. Default is 1.
threshold	A number. The TWDTW dissimilarity threshold, <i>i.e.</i> the maximum TWDTW cost for consideration. Default is Inf.

Value

A `ggplot` object.

Author(s)

Victor Maus, <vwmaus1@gmail.com>

References

Maus V, Camara G, Appel M, Pebesma E (2019). “dtwSat: Time-Weighted Dynamic Time Warping for Satellite Image Time Series Analysis in R.” *Journal of Statistical Software*, **88**(5), 1–31. doi:10.18637/jss.v088.i05.

Maus V, Camara G, Cartaxo R, Sanchez A, Ramos FM, de Queiroz GR (2016). “A Time-Weighted Dynamic Time Warping Method for Land-Use and Land-Cover Mapping.” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, **9**(8), 3729–3739. doi:10.1109/JSTARS.2016.2517118.

See Also

[twdtwMatches-class](#), [twdtwApply](#), [plotPaths](#), [plotCostMatrix](#), [plotMatches](#), and [plotClassification](#).

Examples

```
log_fun = logisticWeight(-0.1, 100)
ts = twdtwTimeSeries(MOD13Q1.ts.list)
patt = twdtwTimeSeries(MOD13Q1.patterns.list)
mat1 = twdtwApply(x=ts, y=patt, weight.fun=log_fun, keep=TRUE, legacy=TRUE)

plotAlignments(mat1)

plotAlignments(mat1, attr=c("evi", "ndvi"))
```

plotArea

Plotting accumulated area

Description

Method for plotting time series of accumulated area.

Usage

```
plotArea(
  x,
  time.levels = NULL,
  time.labels = NULL,
  class.levels = NULL,
  class.labels = NULL,
```

```

class.colors = NULL,
perc = TRUE
)

```

Arguments

x	An object of class <code>twdtwRaster</code> .
time.levels	A <code>character</code> or <code>numeric</code> vector with the layers to plot. For plot type "change" the minimum length is two.
time.labels	A <code>character</code> or <code>numeric</code> vector with the labels of the layers. It must have the same length as time.levels. Default is NULL.
class.levels	A <code>character</code> or <code>numeric</code> vector with the levels of the raster values. Default is NULL.
class.labels	A <code>character</code> or <code>numeric</code> vector with the labels of the raster values. It must have the same length as class.levels. Default is NULL.
class.colors	a set of aesthetic values. It must have the same length as class.levels. Default is NULL. See scale_fill_manual for details.
perc	if TRUE shows the results in percent of area. Otherwise shows the area in the map units or km2 for no project raster. Default is TRUE.

Value

A `ggplot` object.

Author(s)

Victor Maus, <vwmaus1@gmail.com>

References

Maus V, Camara G, Appel M, Pebesma E (2019). "dtwSat: Time-Weighted Dynamic Time Warping for Satellite Image Time Series Analysis in R." *Journal of Statistical Software*, **88**(5), 1–31. [doi:10.18637/jss.v088.i05](https://doi.org/10.18637/jss.v088.i05).

Maus V, Camara G, Cartaxo R, Sanchez A, Ramos FM, de Queiroz GR (2016). "A Time-Weighted Dynamic Time Warping Method for Land-Use and Land-Cover Mapping." *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, **9**(8), 3729–3739. [doi:10.1109/JSTARS.2016.2517118](https://doi.org/10.1109/JSTARS.2016.2517118).

See Also

[twdtwRaster-class](#), [twdtwApply](#), [plotMaps](#), [plotChanges](#), and [plotDistance](#).

Examples

```

## Not run:

# Create raster time series
evi = brick(system.file("lucc_MT/data/evi.tif", package="dtwSat"))

```

```

ndvi = brick(system.file("lucc_MT/data/ndvi.tif", package="dtwSat"))
red = brick(system.file("lucc_MT/data/red.tif", package="dtwSat"))
blue = brick(system.file("lucc_MT/data/blue.tif", package="dtwSat"))
nir = brick(system.file("lucc_MT/data/nir.tif", package="dtwSat"))
mir = brick(system.file("lucc_MT/data/mir.tif", package="dtwSat"))
doy = brick(system.file("lucc_MT/data/doy.tif", package="dtwSat"))
timeline = scan(system.file("lucc_MT/data/timeline", package="dtwSat"), what="date")
rts = twdtwRaster(evi, ndvi, red, blue, nir, mir, timeline = timeline, doy = doy)

# Read field samples
field_samples = read.csv(system.file("lucc_MT/data/samples.csv", package="dtwSat"))
proj_str = scan(system.file("lucc_MT/data/samples_projection",
                           package="dtwSat"), what = "character")

# Split samples for training (10%) and validation (90%) using stratified sampling
library(caret)
set.seed(1)
I = unlist(createDataPartition(field_samples$label, p = 0.1))
training_samples = field_samples[I,]
validation_samples = field_samples[-I,]

# Create temporal patterns
training_ts = getTimeSeries(rts, y = training_samples, proj4string = proj_str)
temporal_patterns = createPatterns(training_ts, freq = 8, formula = y ~ s(x))

# Run TWDTW analysis for raster time series
log_fun = weight.fun=logisticWeight(-0.1,50)
r_twdtw = twdtwApply(x=rts, y=temporal_patterns, weight.fun=log_fun, format="GTiff",
                    overwrite=TRUE)

# Classify raster based on the TWDTW analysis
r_lucc = twdtwClassify(r_twdtw, format="GTiff", overwrite=TRUE)

plotArea(r_lucc)

plotArea(r_lucc, perc=FALSE)

## End(Not run)

```

plotChanges

Plotting changes

Description

Method for plotting changes over time.

Usage

```
plotChanges(
```

```

x,
time.levels = NULL,
time.labels = NULL,
class.levels = NULL,
class.labels = NULL,
class.colors = NULL
)

```

Arguments

x	An object of class <code>twdtwRaster</code> .
time.levels	A character or numeric vector with the layers to plot. For plot type "change" the minimum length is two.
time.labels	A character or numeric vector with the labels of the layers. It must have the same length as time.levels. Default is NULL.
class.levels	A character or numeric vector with the levels of the raster values. Default is NULL.
class.labels	A character or numeric vector with the labels of the raster values. It must have the same length as class.levels. Default is NULL.
class.colors	A set of aesthetic values. It must have the same length as class.levels. Default is NULL. See scale_fill_manual for details.

Value

A [ggplot](#) object.

Author(s)

Victor Maus, <vwmaus1@gmail.com>

References

- Maus V, Camara G, Appel M, Pebesma E (2019). "dtwSat: Time-Weighted Dynamic Time Warping for Satellite Image Time Series Analysis in R." *Journal of Statistical Software*, **88**(5), 1–31. [doi:10.18637/jss.v088.i05](https://doi.org/10.18637/jss.v088.i05).
- Maus V, Camara G, Cartaxo R, Sanchez A, Ramos FM, de Queiroz GR (2016). "A Time-Weighted Dynamic Time Warping Method for Land-Use and Land-Cover Mapping." *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, **9**(8), 3729–3739. [doi:10.1109/JSTARS.2016.2517118](https://doi.org/10.1109/JSTARS.2016.2517118).

See Also

[twdtwRaster-class](#), [twdtwApply](#), [plotArea](#), [plotMaps](#), and [plotDistance](#).

Examples

```

## Not run:
# Run TWDTW analysis for raster time series
patt = MOD13Q1.MT.yearly.patterns
evi = brick(system.file("lucc_MT/data/evi.tif", package="dtwSat"))
ndvi = brick(system.file("lucc_MT/data/ndvi.tif", package="dtwSat"))
red = brick(system.file("lucc_MT/data/red.tif", package="dtwSat"))
blue = brick(system.file("lucc_MT/data/blue.tif", package="dtwSat"))
nir = brick(system.file("lucc_MT/data/nir.tif", package="dtwSat"))
mir = brick(system.file("lucc_MT/data/mir.tif", package="dtwSat"))
doy = brick(system.file("lucc_MT/data/doy.tif", package="dtwSat"))
timeline = scan(system.file("lucc_MT/data/timeline", package="dtwSat"), what="date")
rts = twdtwRaster(evi, ndvi, red, blue, nir, mir, timeline = timeline, doy = doy)

time_interval = seq(from=as.Date("2007-09-01"), to=as.Date("2013-09-01"),
                    by="12 month")
log_fun = weight.fun=logisticWeight(-0.1,50)

r_twdtw = twdtwApply(x=rts, y=patt, weight.fun=log_fun, breaks=time_interval,
                    filepath=~"/test_twdtw", overwrite=TRUE, format="GTiff")

r_lucc = twdtwClassify(r_twdtw, format="GTiff", overwrite=TRUE)

plotChanges(r_lucc)

## End(Not run)

```

plotClassification *Plotting subintervals classification*

Description

Method for plotting the classification of each subinterval of the time series based on TWDTW analysis.

Usage

```

plotClassification(
  x,
  timeseries.labels = NULL,
  patterns.labels = NULL,
  attr,
  ...
)

```

Arguments

x	An object of class <code>twdtwMatches</code> .
timeseries.labels	The label or index of the time series. Default is 1.
patterns.labels	A vector with labels of the patterns. If not declared the function will plot one alignment for each pattern.
attr	An <code>integer</code> vector or <code>character</code> vector indicating the attribute for plotting. If not declared the function will plot all attributes.
...	Additional arguments passed to <code>twdtwClassify</code> .

Value

A `ggplot` object.

Author(s)

Victor Maus, <vwmaus1@gmail.com>

References

Maus V, Camara G, Appel M, Pebesma E (2019). “dtwSat: Time-Weighted Dynamic Time Warping for Satellite Image Time Series Analysis in R.” *Journal of Statistical Software*, **88**(5), 1–31. doi:10.18637/jss.v088.i05.

Maus V, Camara G, Cartaxo R, Sanchez A, Ramos FM, de Queiroz GR (2016). “A Time-Weighted Dynamic Time Warping Method for Land-Use and Land-Cover Mapping.” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, **9**(8), 3729–3739. doi:10.1109/JSTARS.2016.2517118.

See Also

`twdtwMatches-class`, `twdtwApply`, `twdtwClassify`, `plotAlignments`, `plotPaths`, `plotMatches`, and `plotCostMatrix`.

Examples

```
log_fun = logisticWeight(-0.1, 100)
ts = twdtwTimeSeries(MOD13Q1.ts.list)
patt = twdtwTimeSeries(MOD13Q1.patterns.list)
mat1 = twdtwApply(x=ts, y=patt, weight.fun=log_fun, keep=TRUE, legacy=TRUE)

# Classify interval
from = as.Date("2007-09-01")
to = as.Date("2013-09-01")
by = "6 month"
gp = plotClassification(x=mat1, from=from, to=to, by=by, overlap=.5)
gp
```

plotCostMatrix *Plotting paths*

Description

Method for plotting low cost paths in the TWDTW cost matrix.

Usage

```
plotCostMatrix(  
  x,  
  timeseries.labels = NULL,  
  patterns.labels = NULL,  
  matrix.name = "costMatrix"  
)
```

Arguments

`x` An object of class `twdtwMatches`.

`timeseries.labels` The label or index of the time series. Default is 1.

`patterns.labels` A vector with labels of the patterns. If not declared the function will plot one alignment for each pattern.

`matrix.name` A character. The name of the matrix to plot, "costMatrix" for accumulated cost, "localMatrix" for local cost, or "timeWeight" for time-weight. Default is "costMatrix".

Value

A `ggplot` object.

Author(s)

Victor Maus, <vwmaus1@gmail.com>

References

Maus V, Camara G, Appel M, Pebesma E (2019). "dtwSat: Time-Weighted Dynamic Time Warping for Satellite Image Time Series Analysis in R." *Journal of Statistical Software*, **88**(5), 1–31. doi:10.18637/jss.v088.i05.

Maus V, Camara G, Cartaxo R, Sanchez A, Ramos FM, de Queiroz GR (2016). "A Time-Weighted Dynamic Time Warping Method for Land-Use and Land-Cover Mapping." *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, **9**(8), 3729–3739. doi:10.1109/JSTARS.2016.2517118.

See Also

[twdtwMatches-class](#), [twdtwApply](#), [plotAlignments](#), [plotPaths](#), [plotMatches](#), and [plotClassification](#).

Examples

```
log_fun = logisticWeight(-0.1, 100)
ts = twdtwTimeSeries(MOD13Q1.ts.list)
patt = twdtwTimeSeries(MOD13Q1.patterns.list)
mat1 = twdtwApply(x=ts, y=patt, weight.fun=log_fun, keep=TRUE, legacy=TRUE)

plotCostMatrix(mat1, matrix.name="costMatrix")

plotCostMatrix(mat1, matrix.name="localMatrix")

plotCostMatrix(mat1, matrix.name="timeWeight")
```

plotDistance

Plotting distance maps

Description

Method for plotting TWDTW distance maps.

Usage

```
plotDistance(x, time.levels = 1, time.labels = 1, layers = NULL)
```

Arguments

x	An object of class twdtwRaster .
time.levels	A character or numeric vector with the layers to plot. For plot type "change" the minimum length is two.
time.labels	A character or numeric vector with the labels of the layers. It must have the same length as time.levels. Default is NULL.
layers	A character or numeric vector with the layers/bands of the raster time series.

Value

A [ggplot](#) object.

Author(s)

Victor Maus, <vwmaus1@gmail.com>

References

Maus V, Camara G, Appel M, Pebesma E (2019). “dtwSat: Time-Weighted Dynamic Time Warping for Satellite Image Time Series Analysis in R.” *Journal of Statistical Software*, **88**(5), 1–31. doi:10.18637/jss.v088.i05.

Maus V, Camara G, Cartaxo R, Sanchez A, Ramos FM, de Queiroz GR (2016). “A Time-Weighted Dynamic Time Warping Method for Land-Use and Land-Cover Mapping.” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, **9**(8), 3729–3739. doi:10.1109/JSTARS.2016.2517118.

See Also

[twdtwRaster-class](#), [twdtwApply](#), [plotArea](#), [plotChanges](#), and [plotDistance](#).

Examples

```
## Not run:
# Run TWDTW analysis for raster time series
patt = MOD13Q1.MT.yearly.patterns
evi = brick(system.file("lucc_MT/data/evi.tif", package="dtwSat"))
ndvi = brick(system.file("lucc_MT/data/ndvi.tif", package="dtwSat"))
red = brick(system.file("lucc_MT/data/red.tif", package="dtwSat"))
blue = brick(system.file("lucc_MT/data/blue.tif", package="dtwSat"))
nir = brick(system.file("lucc_MT/data/nir.tif", package="dtwSat"))
mir = brick(system.file("lucc_MT/data/mir.tif", package="dtwSat"))
doy = brick(system.file("lucc_MT/data/doy.tif", package="dtwSat"))
timeline = scan(system.file("lucc_MT/data/timeline", package="dtwSat"), what="date")
rts = twdtwRaster(evi, ndvi, red, blue, nir, mir, timeline = timeline, doy = doy)

time_interval = seq(from=as.Date("2007-09-01"), to=as.Date("2013-09-01"),
                    by="12 month")
log_fun = weight.fun=logisticWeight(-0.1,50)

r_twdtw = twdtwApply(x=rts, y=patt, weight.fun=log_fun, breaks=time_interval,
                    filepath=~ /test_twdtw", overwrite=TRUE, format="GTiff", mc.cores=3)

plotDistance(r_twdtw)

## End(Not run)
```

plotMaps

Plotting maps

Description

Method for plotting time series of maps.

Usage

```
plotMaps(  
  x,  
  time.levels = NULL,  
  time.labels = NULL,  
  class.levels = NULL,  
  class.labels = NULL,  
  class.colors = NULL  
)
```

Arguments

x	An object of class twdtwRaster .
time.levels	A character or numeric vector with the layers to plot. For plot type "change" the minimum length is two.
time.labels	A character or numeric vector with the labels of the layers. It must have the same length as time.levels. Default is NULL.
class.levels	A character or numeric vector with the levels of the raster values. Default is NULL.
class.labels	A character or numeric vector with the labels of the raster values. It must have the same length as class.levels. Default is NULL.
class.colors	A set of aesthetic values. It must have the same length as class.levels. Default is NULL. See scale_fill_manual for details.

Value

A [ggplot](#) object.

Author(s)

Victor Maus, <vwmaus1@gmail.com>

References

Maus V, Camara G, Appel M, Pebesma E (2019). "dtwSat: Time-Weighted Dynamic Time Warping for Satellite Image Time Series Analysis in R." *Journal of Statistical Software*, **88**(5), 1–31. [doi:10.18637/jss.v088.i05](https://doi.org/10.18637/jss.v088.i05).

Maus V, Camara G, Cartaxo R, Sanchez A, Ramos FM, de Queiroz GR (2016). "A Time-Weighted Dynamic Time Warping Method for Land-Use and Land-Cover Mapping." *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, **9**(8), 3729–3739. [doi:10.1109/JSTARS.2016.2517118](https://doi.org/10.1109/JSTARS.2016.2517118).

See Also

[twdtwRaster-class](#), [twdtwApply](#), [plotArea](#), [plotChanges](#), and [plotDistance](#).

Examples

```
## Not run:
# Run TWDWTW analysis for raster time series
patt = MOD13Q1.MT.yearly.patterns
evi = brick(system.file("lucc_MT/data/evi.tif", package="dtwSat"))
ndvi = brick(system.file("lucc_MT/data/ndvi.tif", package="dtwSat"))
red = brick(system.file("lucc_MT/data/red.tif", package="dtwSat"))
blue = brick(system.file("lucc_MT/data/blue.tif", package="dtwSat"))
nir = brick(system.file("lucc_MT/data/nir.tif", package="dtwSat"))
mir = brick(system.file("lucc_MT/data/mir.tif", package="dtwSat"))
doy = brick(system.file("lucc_MT/data/doy.tif", package="dtwSat"))
timeline = scan(system.file("lucc_MT/data/timeline", package="dtwSat"), what="date")
rts = twdtwRaster(evi, ndvi, red, blue, nir, mir, timeline = timeline, doy = doy)

time_interval = seq(from=as.Date("2007-09-01"), to=as.Date("2013-09-01"),
                    by="12 month")
log_fun = weight.fun=logisticWeight(-0.1,50)

r_twdtw = twdtwApply(x=rts, y=patt, weight.fun=log_fun, breaks=time_interval,
                    filepath=~"/test_twdtw", overwrite=TRUE, format="GTiff", mc.cores=3)

r_lucc = twdtwClassify(r_twdtw, format="GTiff", overwrite=TRUE)

plotMaps(r_lucc)

## End(Not run)
```

plotMapSamples

Plotting maps

Description

Method for plotting maps and samples.

Usage

```
plotMapSamples(x, samples = "all", ...)
```

Arguments

x	An object of class <code>twdtwAssessment</code> .
samples	A character defining the samples to plot "correct", "incorrect", "all". Default is "all".
...	Other arguments to pass to <code>twdtwRaster</code>

Value

A `ggplot` object.

Author(s)

Victor Maus, <vwmaus1@gmail.com>

References

Maus V, Camara G, Appel M, Pebesma E (2019). “dtwSat: Time-Weighted Dynamic Time Warping for Satellite Image Time Series Analysis in R.” *Journal of Statistical Software*, **88**(5), 1–31. doi:10.18637/jss.v088.i05.

Maus V, Camara G, Cartaxo R, Sanchez A, Ramos FM, de Queiroz GR (2016). “A Time-Weighted Dynamic Time Warping Method for Land-Use and Land-Cover Mapping.” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, **9**(8), 3729–3739. doi:10.1109/JSTARS.2016.2517118.

See Also

[twdtwAssessment](#), [plotAccuracy](#), and [plotAdjustedArea](#).

Examples

```
## Not run:

# Example of TWDTW analysis using raster files
library(dtwSat)
library(caret)

# Load raster data
evi <- brick(system.file("lucc_MT/data/evi.tif", package = "dtwSat"))
ndvi <- brick(system.file("lucc_MT/data/ndvi.tif", package = "dtwSat"))
red <- brick(system.file("lucc_MT/data/red.tif", package = "dtwSat"))
blue <- brick(system.file("lucc_MT/data/blue.tif", package = "dtwSat"))
nir <- brick(system.file("lucc_MT/data/nir.tif", package = "dtwSat"))
mir <- brick(system.file("lucc_MT/data/mir.tif", package = "dtwSat"))
doy <- brick(system.file("lucc_MT/data/doy.tif", package = "dtwSat"))
timeline <-
  scan(system.file("lucc_MT/data/timeline", package = "dtwSat"), what="date")

# Create raster time series
rts <- twdtwRaster(evi, ndvi, red, blue, nir, mir, timeline = timeline, doy = doy)

# Load field samples and projection
field_samples <-
  read.csv(system.file("lucc_MT/data/samples.csv", package = "dtwSat"))
proj_str <-
  scan(system.file("lucc_MT/data/samples_projection", package = "dtwSat"),
        what = "character")

# Split samples for training (10%) and validation (90%) using stratified sampling
set.seed(1)
I <- unlist(createDataPartition(field_samples$label, p = 0.1))
training_samples <- field_samples[I, ]
validation_samples <- field_samples[-I, ]
```



```
# Get time series form raster
training_ts <- getTimeSeries(rts, y = training_samples, proj4string = proj_str)
validation_ts <- getTimeSeries(rts, y = validation_samples, proj4string = proj_str)

# Create temporal patterns
temporal_patterns <- createPatterns(training_ts, freq = 8, formula = y ~ s(x))

# Set TWDTW weight function
log_fun <- logisticWeight(-0.1, 50)

# Run TWDTW analysis
system.time(
  r_twdtw <-
    twdtwApply(x = rts, y = temporal_patterns, weight.fun = log_fun, progress = 'text')
)

# Plot TWDTW distances for the first year
plot(r_twdtw, type = "distance", time.levels = 1)

# Classify raster based on the TWDTW analysis
r_lucc <- twdtwClassify(r_twdtw, progress = 'text')

# Plot TWDTW classification results
plot(r_lucc, type = "map")

# Assess classification
twdtw_assess <-
  twdtwAssess(object = r_lucc, y = validation_samples,
              proj4string = proj_str, conf.int = .95)

# Plot map accuracy
plot(twdtw_assess, type = "accuracy")

# Plot area uncertainty
plot(twdtw_assess, type = "area")

# Plot misclassified samples
plot(twdtw_assess, type = "map", samples = "incorrect")

# Get latex table with error matrix
twdtwXtable(twdtw_assess, table.type = "matrix")

# Get latex table with error accuracy
twdtwXtable(twdtw_assess, table.type = "accuracy")

# Get latex table with area uncertainty
twdtwXtable(twdtw_assess, table.type = "area")

## End(Not run)
```

plotMatches	<i>Plotting matching points</i>
-------------	---------------------------------

Description

Method for plotting the matching points from TWDTW analysis.

Usage

```
plotMatches(  
  x,  
  timeseries.labels = 1,  
  patterns.labels = NULL,  
  k = 1,  
  attr = 1,  
  shift = 0.5,  
  show.dist = FALSE  
)
```

Arguments

x	An object of class twdtwMatches .
timeseries.labels	The label or index of the time series. Default is 1.
patterns.labels	A vector with labels of the patterns. If not declared the function will plot one alignment for each pattern.
k	A positive integer. The index of the last alignment to include in the plot. If not declared the function will plot the best match for each pattern.
attr	An integer or character vector indicating the attribute for plotting. Default is 1.
shift	A number that shifts the pattern position in the x direction. Default is 0.5.
show.dist	Show the distance for each alignment. Default is FALSE.

Value

A [ggplot](#) object.

Author(s)

Victor Maus, <vwmaus1@gmail.com>

References

Maus V, Camara G, Appel M, Pebesma E (2019). “dtwSat: Time-Weighted Dynamic Time Warping for Satellite Image Time Series Analysis in R.” *Journal of Statistical Software*, **88**(5), 1–31. doi:10.18637/jss.v088.i05.

Maus V, Camara G, Cartaxo R, Sanchez A, Ramos FM, de Queiroz GR (2016). “A Time-Weighted Dynamic Time Warping Method for Land-Use and Land-Cover Mapping.” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, **9**(8), 3729–3739. doi:10.1109/JSTARS.2016.2517118.

See Also

[twdtwMatches-class](#), [twdtwApply](#), [plotPaths](#), [plotCostMatrix](#), [plotAlignments](#), and [plotClassification](#).

Examples

```
log_fun = logisticWeight(-0.1, 100)
ts = twdtwTimeSeries(MOD13Q1.ts.list)
patt = twdtwTimeSeries(MOD13Q1.patterns.list)
mat1 = twdtwApply(x=ts, y=patt, weight.fun=log_fun, keep=TRUE, legacy=TRUE)

plotMatches(mat1)

plotMatches(mat1, patterns.labels="Soybean", k=4)

plotMatches(mat1, patterns.labels=c("Soybean", "Maize"), k=4)

plotMatches(mat1, patterns.labels=c("Soybean", "Cotton"), k=c(3,1))
```

plotPaths

Plotting paths

Description

Method for plotting low cost paths in the TWDTW cost matrix.

Usage

```
plotPaths(x, timeseries.labels = NULL, patterns.labels = NULL, k = NULL)
```

Arguments

`x` An object of class [twdtwMatches](#).

`timeseries.labels` The label or index of the time series. Default is 1.

`patterns.labels` A vector with labels of the patterns. If not declared the function will plot one alignment for each pattern.

k A positive integer. The index of the last alignment to include in the plot. If not declared the function will plot all low cost paths.

Value

A `ggplot` object.

Author(s)

Victor Maus, <vwmaus1@gmail.com>

References

Maus V, Camara G, Appel M, Pebesma E (2019). “dtwSat: Time-Weighted Dynamic Time Warping for Satellite Image Time Series Analysis in R.” *Journal of Statistical Software*, **88**(5), 1–31. doi:10.18637/jss.v088.i05.

Maus V, Camara G, Cartaxo R, Sanchez A, Ramos FM, de Queiroz GR (2016). “A Time-Weighted Dynamic Time Warping Method for Land-Use and Land-Cover Mapping.” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, **9**(8), 3729–3739. doi:10.1109/JSTARS.2016.2517118.

See Also

`twdtwMatches-class`, `twdtwApply`, `plotAlignments`, `plotCostMatrix`, `plotMatches`, and `plotClassification`.

Examples

```
log_fun = logisticWeight(-0.1, 100)
ts = twdtwTimeSeries(MOD13Q1.ts.list)
patt = twdtwTimeSeries(MOD13Q1.patterns.list)
mat1 = twdtwApply(x=ts, y=patt, weight.fun=log_fun, keep=TRUE, legacy=TRUE)

plotPaths(mat1)

plotPaths(mat1, patterns.labels="Soybean", k=1:2)

plotPaths(mat1, patterns.labels=c("Maize", "Cotton"), k=2)
```

plotPatterns

Plotting temporal patterns

Description

Method for plotting the temporal patterns.

Usage

```
plotPatterns(x, labels = NULL, attr, year = 2005)
```

Arguments

x	An object of class twdtwTimeSeries , zoo , or list of zoo .
labels	A vector with labels of the time series. If not declared the function will plot all time series.
attr	An integer vector or character vector indicating the attribute for plotting. If not declared the function will plot all attributes.
year	An integer. The base year to shift the dates of the time series to. If NULL then the time series is not shifted. Default is 2005.

Value

A [ggplot](#) object.

Author(s)

Victor Maus, <vwmaus1@gmail.com>

References

Maus V, Camara G, Appel M, Pebesma E (2019). “dtwSat: Time-Weighted Dynamic Time Warping for Satellite Image Time Series Analysis in R.” *Journal of Statistical Software*, **88**(5), 1–31. [doi:10.18637/jss.v088.i05](#).

Maus V, Camara G, Cartaxo R, Sanchez A, Ramos FM, de Queiroz GR (2016). “A Time-Weighted Dynamic Time Warping Method for Land-Use and Land-Cover Mapping.” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, **9**(8), 3729–3739. [doi:10.1109/JSTARS.2016.2517118](#).

See Also

[twdtwTimeSeries-class](#) and [plotTimeSeries](#)

Examples

```
patt = twdtwTimeSeries(MOD13Q1.patterns.list)
plotPatterns(patt)
plotPatterns(patt, attr="evi")
```

plotTimeSeries

Plotting time series

Description

Method for plotting the temporal patterns.

Usage

```
plotTimeSeries(x, labels = NULL, attr)
```

Arguments

x	An object of class twdtwTimeSeries , zoo , or list of class zoo .
labels	A vector with labels of the time series. If missing, all elements in the list will be plotted (up to a maximum of 16).
attr	An integer vector or character vector indicating the attribute for plotting. If not declared the function will plot all attributes.

Value

A [ggplot](#) object.

Author(s)

Victor Maus, <vwmaus1@gmail.com>

References

Maus V, Camara G, Appel M, Pebesma E (2019). “dtwSat: Time-Weighted Dynamic Time Warping for Satellite Image Time Series Analysis in R.” *Journal of Statistical Software*, **88**(5), 1–31. [doi:10.18637/jss.v088.i05](https://doi.org/10.18637/jss.v088.i05).

Maus V, Camara G, Cartaxo R, Sanchez A, Ramos FM, de Queiroz GR (2016). “A Time-Weighted Dynamic Time Warping Method for Land-Use and Land-Cover Mapping.” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, **9**(8), 3729–3739. [doi:10.1109/JSTARS.2016.2517118](https://doi.org/10.1109/JSTARS.2016.2517118).

See Also

[twdtwTimeSeries-class](#) and [plotPatterns](#)

Examples

```
ts = twdtwTimeSeries(MOD13Q1.ts.list)
plotTimeSeries(ts)
plotTimeSeries(ts, attr="evi")
```

resampleTimeSeries *Resample time series*

Description

Resample time series in the object to have the same length.

Usage

```
resampleTimeSeries(object, length = NULL)

## S4 method for signature 'twdtwTimeSeries'
resampleTimeSeries(object, length = NULL)
```

Arguments

object	an object of class <code>twdtwTimeSeries</code> .
length	An integer. The number of samples to resample the time series to. If not declared the length is set to the length of the longest time series.

Value

An object of class `twdtwTimeSeries` whose time series have the same number of samples (points).

Author(s)

Victor Maus, <vwmaus1@gmail.com>

See Also

[twdtwTimeSeries-class](#), and [twdtwApply](#)

Examples

```
# Resampling time series from objects of class twdtwTimeSeries
patt = twdtwTimeSeries(MOD13Q1.patterns.list)
npatt = resampleTimeSeries(patt, length=46)
nrow(patt)
nrow(npatt)
```

shiftDates	<i>Shift dates</i>
------------	--------------------

Description

This function shifts the dates of the time series to a given base year.

Usage

```
shiftDates(object, year = NULL)

## S4 method for signature 'twdtwTimeSeries'
shiftDates(object, year = NULL)

## S4 method for signature 'list'
shiftDates(object, year = NULL)

## S4 method for signature 'zoo'
shiftDates(object, year = NULL)
```

Arguments

object `twdtwTimeSeries` objects, `zoo` objects or a list of `zoo` objects.
year the base year to shift the time series to.

Value

An object of the same class as the input object.

Author(s)

Victor Maus, <vwmaus1@gmail.com>

References

- Maus V, Camara G, Appel M, Pebesma E (2019). “dtwSat: Time-Weighted Dynamic Time Warping for Satellite Image Time Series Analysis in R.” *Journal of Statistical Software*, **88**(5), 1–31. doi:10.18637/jss.v088.i05.
- Maus V, Camara G, Cartaxo R, Sanchez A, Ramos FM, de Queiroz GR (2016). “A Time-Weighted Dynamic Time Warping Method for Land-Use and Land-Cover Mapping.” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, **9**(8), 3729–3739. doi:10.1109/JSTARS.2016.2517118.
- Maus V, Camara G, Appel M, Pebesma E (2019). “dtwSat: Time-Weighted Dynamic Time Warping for Satellite Image Time Series Analysis in R.” *Journal of Statistical Software*, **88**(5), 1–31. doi:10.18637/jss.v088.i05.

Maus V, Camara G, Cartaxo R, Sanchez A, Ramos FM, de Queiroz GR (2016). “A Time-Weighted Dynamic Time Warping Method for Land-Use and Land-Cover Mapping.” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 9(8), 3729–3739. doi:10.1109/JSTARS.2016.2517118.

See Also

[twdtwTimeSeries-class](#)

Examples

```
patt = twdtwTimeSeries(MOD13Q1.patterns.list)
npatt = shiftDates(patt, year=2005)
index(patt)
index(npatt)
```

subset	<i>Subset time series</i>
--------	---------------------------

Description

Get subsets from objects of class twdtw*.

Usage

```
## S4 method for signature 'twdtwTimeSeries'
subset(x, labels = NULL)

## S4 method for signature 'twdtwMatches'
subset(x, timeseries.labels = NULL, patterns.labels = NULL, k = NULL)

## S4 method for signature 'twdtwRaster'
subset(x, e = NULL, layers = NULL)
```

Arguments

x	An objects of class twdtw*.
labels	A character vector with time series labels.
timeseries.labels	a vector with labels of the time series.
patterns.labels	a vector with labels of the patterns.
k	A positive integer. The index of the last alignment to include in the subset.
e	An extent object, or any object from which an Extent object can be extracted. See crop for details.
layers	A vector with the names of the twdtwRaster object to include in the subset.

Value

An object of class `twdtw*`.

Author(s)

Victor Maus, <vwmaus1@gmail.com>

References

Maus V, Camara G, Appel M, Pebesma E (2019). “dtwSat: Time-Weighted Dynamic Time Warping for Satellite Image Time Series Analysis in R.” *Journal of Statistical Software*, **88**(5), 1–31. doi:10.18637/jss.v088.i05.

Maus V, Camara G, Cartaxo R, Sanchez A, Ramos FM, de Queiroz GR (2016). “A Time-Weighted Dynamic Time Warping Method for Land-Use and Land-Cover Mapping.” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, **9**(8), 3729–3739. doi:10.1109/JSTARS.2016.2517118.

See Also

[twdtwRaster-class](#), [twdtwTimeSeries-class](#), and [twdtwMatches-class](#)

Examples

```
# Getting time series from objects of class twdtwTimeSeries
ts = twdtwTimeSeries(MOD13Q1.ts.list)
ts = subset(ts, 2)
ts
# Getting time series from objects of class twdtwTimeSeries
patt = twdtwTimeSeries(MOD13Q1.patterns.list)
mat = twdtwApply(x=ts, y=patt, weight.fun=logisticWeight(-0.1,100),
                 keep=TRUE, legacy=TRUE)
mat = subset(mat, k=4)

## This example creates a twdtwRaster object and extracts time series from it.

# Creating objects of class twdtwRaster with evi and ndvi time series
evi = brick(system.file("lucc_MT/data/evi.tif", package="dtwSat"))
ndvi = brick(system.file("lucc_MT/data/ndvi.tif", package="dtwSat"))
timeline = scan(system.file("lucc_MT/data/timeline", package="dtwSat"), what="date")
rts = twdtwRaster(evi, ndvi, timeline=timeline)

rts_evi = subset(rts, layers="evi")

field_samples = read.csv(system.file("lucc_MT/data/samples.csv", package="dtwSat"))
prj_string = scan(system.file("lucc_MT/data/samples_projection", package="dtwSat"),
                 what = "character")

# Extract time series
ts_evi = getTimeSeries(rts_evi, y = field_samples, proj4string = prj_string)

# Subset all labels = "Forest"
```

```
ts_forest = subset(ts_evi, labels="Forest")
```

twdtwApply

Apply TWDTW analysis

Description

This function performs a multidimensional Time-Weighted DTW analysis and retrieves the matches between the temporal patterns and a set of time series (Maus et al. 2019).

Usage

```
twdtwApply(  
  x,  
  y,  
  resample = TRUE,  
  length = NULL,  
  weight.fun = function(phi, psi) phi,  
  dist.method = "Euclidean",  
  step.matrix = symmetric1,  
  n = NULL,  
  span = NULL,  
  min.length = 0,  
  ...  
)  
  
## S4 method for signature 'twdtwTimeSeries'  
twdtwApply(  
  x,  
  y,  
  resample,  
  length,  
  weight.fun,  
  dist.method,  
  step.matrix,  
  n,  
  span,  
  min.length,  
  legacy = FALSE,  
  keep = FALSE,  
  ...  
)  
  
## S4 method for signature 'twdtwRaster'  
twdtwApply(  
  x,
```

```

y,
resample,
length,
weight.fun,
dist.method,
step.matrix,
n,
span,
min.length,
breaks = NULL,
from = NULL,
to = NULL,
by = NULL,
overlap = 0.5,
filepath = "",
fill = NULL,
legacy = FALSE,
progress = "text",
minrows = 1,
alpha = -0.1,
beta = 50,
...
)

```

Arguments

x	An object of class <code>twdtw*</code> . This is the target time series. Usually, it is a set of unclassified time series.
y	An object of class <code>twdtwTimeSeries</code> . The temporal patterns.
resample	Resample the patterns to have the same length. Default is TRUE. See resampleTimeSeries for details.
length	An integer. Length of patterns used with <code>patterns.length</code> . If not declared the length of the output patterns will be the length of the longest pattern.
weight.fun	A function. Any function that receives two matrices and performs a computation on them, returning a single matrix with the same dimensions. The first matrix is the DTW local cost matrix and the second a matrix of the time differences in days. The function should return a matrix of DTW local cost weighted by the time differences. If not declared the time-weight is zero. In this case the function runs the standard version of the dynamic time warping. See details.
dist.method	A character. Method to derive the local cost matrix. Default is "Euclidean" see dist in package proxy .
step.matrix	See stepPattern in package dtw (Giorgino 2009).
n	An integer. The maximum number of matches to perform. NULL will return all matches.
span	A number. Span between two matches, <i>i.e.</i> the minimum interval between two matches; for details see (Muller 2007). If not declared it removes all overlapping

	matches of the same pattern. To include overlapping matches of the same pattern use <code>span=0</code> .
<code>min.length</code>	A number between 0 and 1. This argument removes overfittings. Minimum length after warping. Percentage of the original pattern length. Default is 0.5, meaning that the matching cannot be shorter than half of the pattern length.
<code>...</code>	Arguments to pass to <code>writeRaster</code> and <code>pbCreate</code>
<code>legacy</code>	logical. If FALSE, runs a faster new TWDTW implementation. Default FALSE
<code>keep</code>	Preserves the cost matrix, inputs, and other internal structures. Default is FALSE. For plot methods use <code>keep=TRUE</code> .
<code>breaks</code>	A vector of class <code>Dates</code> . This replaces the arguments <code>from</code> , <code>to</code> , and <code>by</code> .
<code>from</code>	A character or <code>Dates</code> object in the format "yyyy-mm-dd".
<code>to</code>	A <code>character</code> or <code>Dates</code> object in the format "yyyy-mm-dd".
<code>by</code>	A <code>character</code> with the interval size, e.g. "6 month".
<code>overlap</code>	A number between 0 and 1. The minimum overlapping between one match and the interval of classification. Default is 0.5, i.e. an overlap minimum of 50%.
<code>filepath</code>	A character. The path at which to save the raster with results. If not provided the function saves in the current work directory.
<code>fill</code>	A character to fill the classification gaps. For signature <code>twdtwTimeSeries</code> the default is <code>fill="unclassified"</code> , for signature <code>twdtwRaster</code> the default is <code>fill="unclassified"</code> .
<code>progress</code>	character. 'text' or 'window'.
<code>minrows</code>	Integer. Minimum number of rows in each block
<code>alpha</code>	Numeric. The steepness of TWDTW logistic weight.
<code>beta</code>	Numeric. The midpoint of TWDTW logistic weight.

Details

The linear `linearWeight` and `logisticWeight` weight functions can be passed to `twdtwApply` through the argument `weight.fun`. This will add a time-weight to the dynamic time warping analysis. The time weight creates a global constraint useful for analyzing time series with phenological cycles of vegetation that are usually bound to seasons. In previous studies by (Maus et al. 2016) the logistic weight had better results than the linear for land cover classification. See (Maus et al. 2016; Maus et al. 2019) for details about the method.

Value

An object of class `twdtw*`.

Author(s)

Victor Maus, <vwmaus1@gmail.com>

References

Giorgino T (2009). “Computing and Visualizing Dynamic Time Warping Alignments in R: The dtw Package.” *Journal of Statistical Software*, **31**(7), 1–24. doi:10.18637/jss.v031.i07.

Maus V, Camara G, Appel M, Pebesma E (2019). “dtwSat: Time-Weighted Dynamic Time Warping for Satellite Image Time Series Analysis in R.” *Journal of Statistical Software*, **88**(5), 1–31. doi:10.18637/jss.v088.i05.

Maus V, Camara G, Cartaxo R, Sanchez A, Ramos FM, de Queiroz GR (2016). “A Time-Weighted Dynamic Time Warping Method for Land-Use and Land-Cover Mapping.” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, **9**(8), 3729–3739. doi:10.1109/JSTARS.2016.2517118.

Muller M (2007). *Information Retrieval for Music and Motion*. Springer-Verlag, London.

See Also

[twdtwMatches-class](#), [twdtwTimeSeries-class](#), [twdtwRaster-class](#), [getTimeSeries](#), and [createPatterns](#)

Examples

```
# Applying TWDTW analysis to objects of class twdtwTimeSeries
log_fun = logisticWeight(-0.1, 100)
ts = twdtwTimeSeries(MOD13Q1.ts.list)
patt = twdtwTimeSeries(MOD13Q1.patterns.list)
mat1 = twdtwApply(x=ts, y=patt, weight.fun=log_fun, keep=TRUE, legacy=TRUE)
mat1

## Not run:
# Parallel processin
require(parallel)
mat_list = mclapply(as.list(ts), mc.cores=2, FUN=twdtwApply, y=patt, weight.fun=log_fun)
mat2 = twdtwMatches(alignment=mat_list)

## End(Not run)
## Not run:

# Example of TWDTW analysis using raster files
library(dtwSat)
library(caret)

# Load raster data
evi <- brick(system.file("lucc_MT/data/evi.tif", package = "dtwSat"))
ndvi <- brick(system.file("lucc_MT/data/ndvi.tif", package = "dtwSat"))
red <- brick(system.file("lucc_MT/data/red.tif", package = "dtwSat"))
blue <- brick(system.file("lucc_MT/data/blue.tif", package = "dtwSat"))
nir <- brick(system.file("lucc_MT/data/nir.tif", package = "dtwSat"))
mir <- brick(system.file("lucc_MT/data/mir.tif", package = "dtwSat"))
doy <- brick(system.file("lucc_MT/data/doy.tif", package = "dtwSat"))
timeline <-
  scan(system.file("lucc_MT/data/timeline", package = "dtwSat"), what="date")
```

```
# Create raster time series
rts <- twdtwRaster(evi, ndvi, red, blue, nir, mir, timeline = timeline, doy = doy)

# Load field samples and projection
field_samples <-
  read.csv(system.file("lucc_MT/data/samples.csv", package = "dtwSat"))
proj_str <-
  scan(system.file("lucc_MT/data/samples_projection", package = "dtwSat"),
        what = "character")

# Split samples for training (10%) and validation (90%) using stratified sampling
set.seed(1)
I <- unlist(createDataPartition(field_samples$label, p = 0.1))
training_samples <- field_samples[I, ]
validation_samples <- field_samples[-I, ]

# Get time series form raster
training_ts <- getTimeSeries(rts, y = training_samples, proj4string = proj_str)
validation_ts <- getTimeSeries(rts, y = validation_samples, proj4string = proj_str)

# Create temporal patterns
temporal_patterns <- createPatterns(training_ts, freq = 8, formula = y ~ s(x))

# Set TWDTW weight function
log_fun <- logisticWeight(-0.1, 50)

# Run TWDTW analysis
system.time(
  r_twdtw <-
    twdtwApply(x = rts, y = temporal_patterns, weight.fun = log_fun, progress = 'text')
)

# Plot TWDTW distances for the first year
plot(r_twdtw, type = "distance", time.levels = 1)

# Classify raster based on the TWDTW analysis
r_lucc <- twdtwClassify(r_twdtw, progress = 'text')

# Plot TWDTW classification results
plot(r_lucc, type = "map")

# Assess classification
twdtw_assess <-
  twdtwAssess(object = r_lucc, y = validation_samples,
              proj4string = proj_str, conf.int = .95)

# Plot map accuracy
plot(twdtw_assess, type = "accuracy")

# Plot area uncertainty
plot(twdtw_assess, type = "area")
```

```

# Plot misclassified samples
plot(twdtw_assess, type = "map", samples = "incorrect")

# Get latex table with error matrix
twdtwXtable(twdtw_assess, table.type = "matrix")

# Get latex table with error accuracy
twdtwXtable(twdtw_assess, table.type = "accuracy")

# Get latex table with area uncertainty
twdtwXtable(twdtw_assess, table.type = "area")

## End(Not run)

```

twdtwAssess

Assess TWDTW classification

Description

Performs an accuracy assessment of the classified maps. The function returns Overall Accuracy, User's Accuracy, Produce's Accuracy, error matrix (confusion matrix), and estimated area according to Olofsson et al. (2013); Olofsson et al. (2014). The function returns the metrics for each time interval and a summary considering all classified intervals.

Usage

```

## S4 method for signature 'twdtwRaster'
twdtwAssess(
  object,
  y,
  labels = NULL,
  id.labels = NULL,
  proj4string = NULL,
  conf.int = 0.95,
  rm.nosample = FALSE,
  start_date = NULL
)

## S4 method for signature 'data.frame'
twdtwAssess(object, area, conf.int = 0.95, rm.nosample = TRUE)

## S4 method for signature 'table'
twdtwAssess(object, area, conf.int = 0.95, rm.nosample = TRUE)

## S4 method for signature 'matrix'
twdtwAssess(object, area, conf.int = 0.95, rm.nosample = TRUE)

```



```
## S4 method for signature 'twdtwMatches'
twdtwAssess(object, area, conf.int = 0.95, rm.nosample = TRUE)
```

Arguments

object	An object of class <code>twdtwRaster</code> resulting from the classification, i.e. <code>twdtwClassify</code> . The argument can also receive an error matrix (confusion matrix) using the classes <code>data.frame</code> or <code>table</code> . In this case the user must provide the area for each class to the argument area.
y	A <code>data.frame</code> whose attributes are: longitude, latitude, the start "from" and the end "to" of the time interval for each sample. This can also be a <code>SpatialPointsDataFrame</code> whose attributes are the start "from" and the end "to" of the time interval. If missing "from" and/or "to", they are set to the time range of the object.
labels	Character vector with time series labels. For signature <code>twdtwRaster</code> this argument can be used to set the labels for each sample in y, or it can be combined with <code>id.labels</code> to select samples with a specific label.
id.labels	A numeric or character with a column name from y to be used as samples labels. Optional.
proj4string	Projection string, see <code>CRS-class</code> . Used if y is a <code>data.frame</code> .
conf.int	Specifies the confidence level (0-1).
rm.nosample	If sum of columns and sum of rows of the error matrix are zero then remove class. Default is TRUE.
start_date	A date. Required if there is only one map to be assessed. Usually this is the first date of the timeline from satellite images.
area	A numeric vector with the area for each class if the argument object is an error matrix (confusion matrix). If object is <code>twdtwMatches</code> area can be either a vector with the area of each classified object, or a single number if the objects are single pixels.

Author(s)

Victor Maus, <vwmaus1@gmail.com>

References

Olofsson P, Foody GM, Herold M, Stehman SV, Woodcock CE, Wulder MA (2014). "Good Practices for Estimating Area and Assessing Accuracy of Land Change." *Remote Sensing of Environment*, **148**, 42–57. ISSN 0034–4257, doi:10.1016/j.rse.2014.02.015.

Olofsson P, Foody GM, Stehman SV, Woodcock CE (2013). "Making Better Use of Accuracy Aata in Land Change Studies: Estimating Accuracy and Area and Quantifying Uncertainty Using Stratified Estimation." *Remote Sensing of Environment*, **129**, 122–131. doi:10.1016/j.rse.2012.10.031.

Maus V, Camara G, Appel M, Pebesma E (2019). "dtwSat: Time-Weighted Dynamic Time Warping for Satellite Image Time Series Analysis in R." *Journal of Statistical Software*, **88**(5), 1–31. doi:10.18637/jss.v088.i05.

Maus V, Camara G, Cartaxo R, Sanchez A, Ramos FM, de Queiroz GR (2016). "A Time-Weighted Dynamic Time Warping Method for Land-Use and Land-Cover Mapping." *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 9(8), 3729–3739. doi:10.1109/JSTARS.2016.2517118.

See Also

[twdtwClassify](#), [twdtwAssessment](#), and [twdtwXtable](#).

Examples

```
## Not run:

# Example of TWDTW analysis using raster files
library(dtwSat)
library(caret)

# Load raster data
evi <- brick(system.file("lucc_MT/data/evi.tif", package = "dtwSat"))
ndvi <- brick(system.file("lucc_MT/data/ndvi.tif", package = "dtwSat"))
red <- brick(system.file("lucc_MT/data/red.tif", package = "dtwSat"))
blue <- brick(system.file("lucc_MT/data/blue.tif", package = "dtwSat"))
nir <- brick(system.file("lucc_MT/data/nir.tif", package = "dtwSat"))
mir <- brick(system.file("lucc_MT/data/mir.tif", package = "dtwSat"))
doy <- brick(system.file("lucc_MT/data/doy.tif", package = "dtwSat"))
timeline <-
  scan(system.file("lucc_MT/data/timeline", package = "dtwSat"), what="date")

# Create raster time series
rts <- twdtwRaster(evi, ndvi, red, blue, nir, mir, timeline = timeline, doy = doy)

# Load field samples and projection
field_samples <-
  read.csv(system.file("lucc_MT/data/samples.csv", package = "dtwSat"))
proj_str <-
  scan(system.file("lucc_MT/data/samples_projection", package = "dtwSat"),
        what = "character")

# Split samples for training (10%) and validation (90%) using stratified sampling
set.seed(1)
I <- unlist(createDataPartition(field_samples$label, p = 0.1))
training_samples <- field_samples[I, ]
validation_samples <- field_samples[-I, ]

# Get time series form raster
training_ts <- getTimeSeries(rts, y = training_samples, proj4string = proj_str)
validation_ts <- getTimeSeries(rts, y = validation_samples, proj4string = proj_str)

# Create temporal patterns
temporal_patterns <- createPatterns(training_ts, freq = 8, formula = y ~ s(x))

# Set TWDTW weight function
```

```
log_fun <- logisticWeight(-0.1, 50)

# Run TWDTW analysis
system.time(
  r_twdtw <-
    twdtwApply(x = rts, y = temporal_patterns, weight.fun = log_fun, progress = 'text')
)

# Plot TWDTW distances for the first year
plot(r_twdtw, type = "distance", time.levels = 1)

# Classify raster based on the TWDTW analysis
r_lucc <- twdtwClassify(r_twdtw, progress = 'text')

# Plot TWDTW classification results
plot(r_lucc, type = "map")

# Assess classification
twdtw_assess <-
  twdtwAssess(object = r_lucc, y = validation_samples,
              proj4string = proj_str, conf.int = .95)

# Plot map accuracy
plot(twdtw_assess, type = "accuracy")

# Plot area uncertainty
plot(twdtw_assess, type = "area")

# Plot misclassified samples
plot(twdtw_assess, type = "map", samples = "incorrect")

# Get latex table with error matrix
twdtwXtable(twdtw_assess, table.type = "matrix")

# Get latex table with error accuracy
twdtwXtable(twdtw_assess, table.type = "accuracy")

# Get latex table with area uncertainty
twdtwXtable(twdtw_assess, table.type = "area")

## End(Not run)

# Total mapped area by class. Data from [1]
area = c(A = 22353, B = 1122543, C = 610228)

# Error matrix, columns (Reference) rows (Map)
x =
  rbind(
    c( 97, 0, 3),
    c( 3, 279, 18),
    c( 2, 1, 97)
```

```

)

table_assess = twdtwAssess(x, area, conf.int = .95)

table_assess

plot(table_assess, type="area", perc=FALSE)

plot(table_assess, type="accuracy")

## Not run:

# Example of TWDTW analysis using raster files
library(dtwSat)
library(caret)

# Load raster data
evi <- brick(system.file("lucc_MT/data/evi.tif", package = "dtwSat"))
ndvi <- brick(system.file("lucc_MT/data/ndvi.tif", package = "dtwSat"))
red <- brick(system.file("lucc_MT/data/red.tif", package = "dtwSat"))
blue <- brick(system.file("lucc_MT/data/blue.tif", package = "dtwSat"))
nir <- brick(system.file("lucc_MT/data/nir.tif", package = "dtwSat"))
mir <- brick(system.file("lucc_MT/data/mir.tif", package = "dtwSat"))
doy <- brick(system.file("lucc_MT/data/doy.tif", package = "dtwSat"))
timeline <-
  scan(system.file("lucc_MT/data/timeline", package = "dtwSat"), what="date")

# Create raster time series
rts <- twdtwRaster(evi, ndvi, red, blue, nir, mir, timeline = timeline, doy = doy)

# Load field samples and projection
field_samples <-
  read.csv(system.file("lucc_MT/data/samples.csv", package = "dtwSat"))
proj_str <-
  scan(system.file("lucc_MT/data/samples_projection", package = "dtwSat"),
        what = "character")

# Split samples for training (10%) and validation (90%) using stratified sampling
set.seed(1)
I <- unlist(createDataPartition(field_samples$label, p = 0.1))
training_samples <- field_samples[I, ]
validation_samples <- field_samples[-I, ]

# Get time series form raster
training_ts <- getTimeSeries(rts, y = training_samples, proj4string = proj_str)
validation_ts <- getTimeSeries(rts, y = validation_samples, proj4string = proj_str)

# Create temporal patterns
temporal_patterns <- createPatterns(training_ts, freq = 8, formula = y ~ s(x))

# Set TWDTW weight function
log_fun <- logisticWeight(-0.1, 50)

```

```
# Run TWDTW analysis
system.time(
  r_twdtw <-
    twdtwApply(x = rts, y = temporal_patterns, weight.fun = log_fun, progress = 'text')
)

# Plot TWDTW distances for the first year
plot(r_twdtw, type = "distance", time.levels = 1)

# Classify raster based on the TWDTW analysis
r_lucc <- twdtwClassify(r_twdtw, progress = 'text')

# Plot TWDTW classification results
plot(r_lucc, type = "map")

# Assess classification
twdtw_assess <-
  twdtwAssess(object = r_lucc, y = validation_samples,
              proj4string = proj_str, conf.int = .95)

# Plot map accuracy
plot(twdtw_assess, type = "accuracy")

# Plot area uncertainty
plot(twdtw_assess, type = "area")

# Plot misclassified samples
plot(twdtw_assess, type = "map", samples = "incorrect")

# Get latex table with error matrix
twdtwXtable(twdtw_assess, table.type = "matrix")

# Get latex table with error accuracy
twdtwXtable(twdtw_assess, table.type = "accuracy")

# Get latex table with area uncertainty
twdtwXtable(twdtw_assess, table.type = "area")

## End(Not run)
```

twdtwAssessment-class *class "twdtwAssessment"*

Description

This class stores the map assessment metrics.

Usage

```
## S4 method for signature 'twdtwAssessment'  
show(object)
```

Arguments

object an object of class twdtwAssessment.

Details

If the twdtwRaster is unprojected (longitude/latitude) the estimated area is the sum of the approximate surface area in km² of each cell (pixel). If the twdtwRaster is projected the estimated area is calculated using the the pixel resolution in the map unit.

Slots

accuracySummary: Overall Accuracy, User's Accuracy, Producer's Accuracy, Error Matrix (confusion matrix), and Estimated Area, considering all time periods.

accuracyByPeriod: Overall Accuracy, User's Accuracy, Producer's Accuracy, Error Matrix (confusion matrix), and Estimated Area, for each time periods independently from each other.

data: A [SpatialPointsDataFrame](#) with sample ID, period, date from, date to, reference labels, predicted labels, and TWDTW distance.

map: A [twdtwRaster](#) with the raster maps.

Author(s)

Victor Maus, <vwmaus1@gmail.com>

References

Maus V, Camara G, Appel M, Pebesma E (2019). "dtwSat: Time-Weighted Dynamic Time Warping for Satellite Image Time Series Analysis in R." *Journal of Statistical Software*, **88**(5), 1–31. [doi:10.18637/jss.v088.i05](https://doi.org/10.18637/jss.v088.i05).

Maus V, Camara G, Cartaxo R, Sanchez A, Ramos FM, de Queiroz GR (2016). "A Time-Weighted Dynamic Time Warping Method for Land-Use and Land-Cover Mapping." *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, **9**(8), 3729–3739. [doi:10.1109/JSTARS.2016.2517118](https://doi.org/10.1109/JSTARS.2016.2517118).

See Also

[twdtwClassify](#), [twdtwAssess](#), and [twdtwXtable](#).

twdtwClassify	<i>Classify time series</i>
---------------	-----------------------------

Description

This function classifies the intervals of a time series based on the TWDTW results.

Usage

```
twdtwClassify(x, ...)  
  
## S4 method for signature 'data.frame'  
twdtwClassify(  
  x,  
  y,  
  step.matrix = symmetric1,  
  breaks = NULL,  
  from = NULL,  
  to = NULL,  
  by = NULL,  
  overlap = 0.5,  
  fill = length(y),  
  alpha = -0.1,  
  beta = 50,  
  time.window = FALSE,  
  keep = FALSE,  
  ...  
)  
  
## S4 method for signature 'list'  
twdtwClassify(  
  x,  
  y,  
  step.matrix = symmetric1,  
  breaks = NULL,  
  from = NULL,  
  to = NULL,  
  by = NULL,  
  overlap = 0.5,  
  fill = length(y),  
  alpha = -0.1,  
  beta = 50,  
  time.window = FALSE,  
  keep = FALSE,  
  ...  
)
```

```

## S4 method for signature 'twdtwTimeSeries'
twdtwClassify(
  x,
  patterns.labels = NULL,
  from = NULL,
  to = NULL,
  by = NULL,
  breaks = NULL,
  overlap = 0.5,
  thresholds = Inf,
  fill = "unclassified",
  ...
)

## S4 method for signature 'twdtwMatches'
twdtwClassify(
  x,
  patterns.labels = NULL,
  from = NULL,
  to = NULL,
  by = NULL,
  breaks = NULL,
  overlap = 0.5,
  thresholds = Inf,
  fill = "unclassified"
)

## S4 method for signature 'twdtwRaster'
twdtwClassify(
  x,
  patterns.labels = NULL,
  thresholds = Inf,
  fill = 255,
  filepath = "",
  ...
)

```

Arguments

x	An object of class <code>twdtw*</code> . This is the target time series. Usually, it is a set of unclassified time series.
...	Arguments to pass to specific methods for each <code>twdtw*</code> class and other arguments to pass to <code>writeRaster</code> and <code>pbCreate</code> . If x of <code>twdtwTimeSeries-class</code> additional arguments passed to <code>twdtwApply</code> .
y	a list of <code>data.frame</code> objects similar to x. The temporal patterns used to classify the time series in x.
step.matrix	See <code>stepPattern</code> in package <code>dtw</code> (Giorgino 2009).
breaks	A vector of class <code>Dates</code> . This replaces the arguments <code>from</code> , <code>to</code> , and <code>by</code> .

from	A character or Dates object in the format "yyyy-mm-dd".
to	A character or Dates object in the format "yyyy-mm-dd".
by	A character with the interval size, <i>e.g.</i> "6 month".
overlap	A number between 0 and 1. The minimum overlapping between one match and the interval of classification. Default is 0.5, <i>i.e.</i> an overlap minimum of 50%.
fill	A character to fill the classification gaps. For signature <code>twdtwTimeSeries</code> the default is <code>fill="unclassified"</code> , for signature <code>twdtwRaster</code> the default is <code>fill="unclassified"</code> .
alpha	Numeric. The steepness of TWDTW logistic weight.
beta	Numeric. The midpoint of TWDTW logistic weight.
time.window	logical. TRUE will constrain the TWDTW computation to the value of the parameter beta defined in the logistic weight function. Default is FALSE.
keep	Preserves the cost matrix, inputs, and other internal structures. Default is FALSE. For plot methods use <code>keep=TRUE</code> .
patterns.labels	a vector with labels of the patterns.
thresholds	A numeric vector the same length as <code>patterns.labels</code> . The TWDTW dissimilarity thresholds, <i>i.e.</i> the maximum TWDTW cost for consideration in the classification. Default is <code>Inf</code> for all <code>patterns.labels</code> .
filepath	A character. The path at which to save the raster with results. If not provided the function saves in the same directory as the input time series raster.

Value

An object of class `twdtw*`.

Author(s)

Victor Maus, <vwmaus1@gmail.com>

References

Maus V, Camara G, Appel M, Pebesma E (2019). "dtwSat: Time-Weighted Dynamic Time Warping for Satellite Image Time Series Analysis in R." *Journal of Statistical Software*, **88**(5), 1–31. [doi:10.18637/jss.v088.i05](https://doi.org/10.18637/jss.v088.i05).

Maus V, Camara G, Cartaxo R, Sanchez A, Ramos FM, de Queiroz GR (2016). "A Time-Weighted Dynamic Time Warping Method for Land-Use and Land-Cover Mapping." *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, **9**(8), 3729–3739. [doi:10.1109/JSTARS.2016.2517118](https://doi.org/10.1109/JSTARS.2016.2517118).

See Also

[twdtwApply](#), [twdtwMatches-class](#), [twdtwTimeSeries-class](#), and [twdtwRaster-class](#),

Examples

```

## Not run:

# Example of TWDTW analysis using raster files
library(dtwSat)
library(caret)

# Load raster data
evi <- brick(system.file("lucc_MT/data/evi.tif", package = "dtwSat"))
ndvi <- brick(system.file("lucc_MT/data/ndvi.tif", package = "dtwSat"))
red <- brick(system.file("lucc_MT/data/red.tif", package = "dtwSat"))
blue <- brick(system.file("lucc_MT/data/blue.tif", package = "dtwSat"))
nir <- brick(system.file("lucc_MT/data/nir.tif", package = "dtwSat"))
mir <- brick(system.file("lucc_MT/data/mir.tif", package = "dtwSat"))
doy <- brick(system.file("lucc_MT/data/doy.tif", package = "dtwSat"))
timeline <-
  scan(system.file("lucc_MT/data/timeline", package = "dtwSat"), what="date")

# Create raster time series
rts <- twdtwRaster(evi, ndvi, red, blue, nir, mir, timeline = timeline, doyr = doyr)

# Load field samples and projection
field_samples <-
  read.csv(system.file("lucc_MT/data/samples.csv", package = "dtwSat"))
proj_str <-
  scan(system.file("lucc_MT/data/samples_projection", package = "dtwSat"),
        what = "character")

# Split samples for training (10%) and validation (90%) using stratified sampling
set.seed(1)
I <- unlist(createDataPartition(field_samples$label, p = 0.1))
training_samples <- field_samples[I, ]
validation_samples <- field_samples[-I, ]

# Get time series from raster
training_ts <- getTimeSeries(rts, y = training_samples, proj4string = proj_str)
validation_ts <- getTimeSeries(rts, y = validation_samples, proj4string = proj_str)

# Create temporal patterns
temporal_patterns <- createPatterns(training_ts, freq = 8, formula = y ~ s(x))

# Set TWDTW weight function
log_fun <- logisticWeight(-0.1, 50)

# Run TWDTW analysis
system.time(
  r_twdtw <-
    twdtwApply(x = rts, y = temporal_patterns, weight.fun = log_fun, progress = 'text')
)

# Plot TWDTW distances for the first year
plot(r_twdtw, type = "distance", time.levels = 1)

```

```
# Classify raster based on the TWDTW analysis
r_lucc <- twdtwClassify(r_twdtw, progress = 'text')

# Plot TWDTW classification results
plot(r_lucc, type = "map")

# Assess classification
twdtw_assess <-
  twdtwAssess(object = r_lucc, y = validation_samples,
              proj4string = proj_str, conf.int = .95)

# Plot map accuracy
plot(twdtw_assess, type = "accuracy")

# Plot area uncertainty
plot(twdtw_assess, type = "area")

# Plot misclassified samples
plot(twdtw_assess, type = "map", samples = "incorrect")

# Get latex table with error matrix
twdtwXtable(twdtw_assess, table.type = "matrix")

# Get latex table with error accuracy
twdtwXtable(twdtw_assess, table.type = "accuracy")

# Get latex table with area uncertainty
twdtwXtable(twdtw_assess, table.type = "area")

## End(Not run)

## Not run:

# Example of TWDTW analysis using raster files
library(dtwSat)
library(caret)

# Load raster data
evi <- brick(system.file("lucc_MT/data/evi.tif", package = "dtwSat"))
ndvi <- brick(system.file("lucc_MT/data/ndvi.tif", package = "dtwSat"))
red <- brick(system.file("lucc_MT/data/red.tif", package = "dtwSat"))
blue <- brick(system.file("lucc_MT/data/blue.tif", package = "dtwSat"))
nir <- brick(system.file("lucc_MT/data/nir.tif", package = "dtwSat"))
mir <- brick(system.file("lucc_MT/data/mir.tif", package = "dtwSat"))
doy <- brick(system.file("lucc_MT/data/doy.tif", package = "dtwSat"))
timeline <-
  scan(system.file("lucc_MT/data/timeline", package = "dtwSat"), what="date")

# Create raster time series
rts <- twdtwRaster(evi, ndvi, red, blue, nir, mir, timeline = timeline, doy = doy)
```

```

# Load field samples and projection
field_samples <-
  read.csv(system.file("lucc_MT/data/samples.csv", package = "dtwSat"))
proj_str <-
  scan(system.file("lucc_MT/data/samples_projection", package = "dtwSat"),
        what = "character")

# Split samples for training (10%) and validation (90%) using stratified sampling
set.seed(1)
I <- unlist(createDataPartition(field_samples$label, p = 0.1))
training_samples <- field_samples[I, ]
validation_samples <- field_samples[-I, ]

# Get time series form raster
training_ts <- getTimeSeries(rts, y = training_samples, proj4string = proj_str)
validation_ts <- getTimeSeries(rts, y = validation_samples, proj4string = proj_str)

# Create temporal patterns
temporal_patterns <- createPatterns(training_ts, freq = 8, formula = y ~ s(x))

# Set TWDTW weight function
log_fun <- logisticWeight(-0.1, 50)

# Run TWDTW analysis
system.time(
  r_twdtw <-
    twdtwApply(x = rts, y = temporal_patterns, weight.fun = log_fun, progress = 'text')
)

# Plot TWDTW distances for the first year
plot(r_twdtw, type = "distance", time.levels = 1)

# Classify raster based on the TWDTW analysis
r_lucc <- twdtwClassify(r_twdtw, progress = 'text')

# Plot TWDTW classification results
plot(r_lucc, type = "map")

# Assess classification
twdtw_assess <-
  twdtwAssess(object = r_lucc, y = validation_samples,
              proj4string = proj_str, conf.int = .95)

# Plot map accuracy
plot(twdtw_assess, type = "accuracy")

# Plot area uncertainty
plot(twdtw_assess, type = "area")

# Plot misclassified samples
plot(twdtw_assess, type = "map", samples = "incorrect")

# Get latex table with error matrix

```

```

twdtwXtable(twdtw_assess, table.type = "matrix")

# Get latex table with error accuracy
twdtwXtable(twdtw_assess, table.type = "accuracy")

# Get latex table with area uncertainty
twdtwXtable(twdtw_assess, table.type = "area")

## End(Not run)

## Not run:

# Example of TWDTW analysis using raster files
library(dtwSat)
library(caret)

# Load raster data
evi <- brick(system.file("lucc_MT/data/evi.tif", package = "dtwSat"))
ndvi <- brick(system.file("lucc_MT/data/ndvi.tif", package = "dtwSat"))
red <- brick(system.file("lucc_MT/data/red.tif", package = "dtwSat"))
blue <- brick(system.file("lucc_MT/data/blue.tif", package = "dtwSat"))
nir <- brick(system.file("lucc_MT/data/nir.tif", package = "dtwSat"))
mir <- brick(system.file("lucc_MT/data/mir.tif", package = "dtwSat"))
doy <- brick(system.file("lucc_MT/data/doy.tif", package = "dtwSat"))
timeline <-
  scan(system.file("lucc_MT/data/timeline", package = "dtwSat"), what="date")

# Create raster time series
rts <- twdtwRaster(evi, ndvi, red, blue, nir, mir, timeline = timeline, doy = doy)

# Load field samples and projection
field_samples <-
  read.csv(system.file("lucc_MT/data/samples.csv", package = "dtwSat"))
proj_str <-
  scan(system.file("lucc_MT/data/samples_projection", package = "dtwSat"),
        what = "character")

# Split samples for training (10%) and validation (90%) using stratified sampling
set.seed(1)
I <- unlist(createDataPartition(field_samples$label, p = 0.1))
training_samples <- field_samples[I, ]
validation_samples <- field_samples[-I, ]

# Get time series form raster
training_ts <- getTimeSeries(rts, y = training_samples, proj4string = proj_str)
validation_ts <- getTimeSeries(rts, y = validation_samples, proj4string = proj_str)

# Create temporal patterns
temporal_patterns <- createPatterns(training_ts, freq = 8, formula = y ~ s(x))

# Set TWDTW weight function
log_fun <- logisticWeight(-0.1, 50)

```

```
# Run TWDTW analysis
system.time(
  r_twdtw <-
    twdtwApply(x = rts, y = temporal_patterns, weight.fun = log_fun, progress = 'text')
)

# Plot TWDTW distances for the first year
plot(r_twdtw, type = "distance", time.levels = 1)

# Classify raster based on the TWDTW analysis
r_lucc <- twdtwClassify(r_twdtw, progress = 'text')

# Plot TWDTW classification results
plot(r_lucc, type = "map")

# Assess classification
twdtw_assess <-
  twdtwAssess(object = r_lucc, y = validation_samples,
              proj4string = proj_str, conf.int = .95)

# Plot map accuracy
plot(twdtw_assess, type = "accuracy")

# Plot area uncertainty
plot(twdtw_assess, type = "area")

# Plot misclassified samples
plot(twdtw_assess, type = "map", samples = "incorrect")

# Get latex table with error matrix
twdtwXtable(twdtw_assess, table.type = "matrix")

# Get latex table with error accuracy
twdtwXtable(twdtw_assess, table.type = "accuracy")

# Get latex table with area uncertainty
twdtwXtable(twdtw_assess, table.type = "area")

## End(Not run)

## Not run:

# Example of TWDTW analysis using raster files
library(dtwSat)
library(caret)

# Load raster data
evi <- brick(system.file("lucc_MT/data/evi.tif", package = "dtwSat"))
ndvi <- brick(system.file("lucc_MT/data/ndvi.tif", package = "dtwSat"))
red <- brick(system.file("lucc_MT/data/red.tif", package = "dtwSat"))
blue <- brick(system.file("lucc_MT/data/blue.tif", package = "dtwSat"))
```

```

nir <- brick(system.file("lucc_MT/data/nir.tif", package = "dtwSat"))
mir <- brick(system.file("lucc_MT/data/mir.tif", package = "dtwSat"))
doy <- brick(system.file("lucc_MT/data/doy.tif", package = "dtwSat"))
timeline <-
  scan(system.file("lucc_MT/data/timeline", package = "dtwSat"), what="date")

# Create raster time series
rts <- twdtwRaster(evi, ndvi, red, blue, nir, mir, timeline = timeline, doy = doy)

# Load field samples and projection
field_samples <-
  read.csv(system.file("lucc_MT/data/samples.csv", package = "dtwSat"))
proj_str <-
  scan(system.file("lucc_MT/data/samples_projection", package = "dtwSat"),
        what = "character")

# Split samples for training (10%) and validation (90%) using stratified sampling
set.seed(1)
I <- unlist(createDataPartition(field_samples$label, p = 0.1))
training_samples <- field_samples[I, ]
validation_samples <- field_samples[-I, ]

# Get time series form raster
training_ts <- getTimeSeries(rts, y = training_samples, proj4string = proj_str)
validation_ts <- getTimeSeries(rts, y = validation_samples, proj4string = proj_str)

# Create temporal patterns
temporal_patterns <- createPatterns(training_ts, freq = 8, formula = y ~ s(x))

# Set TWDTW weight function
log_fun <- logisticWeight(-0.1, 50)

# Run TWDTW analysis
system.time(
  r_twdtw <-
    twdtwApply(x = rts, y = temporal_patterns, weight.fun = log_fun, progress = 'text')
)

# Plot TWDTW distances for the first year
plot(r_twdtw, type = "distance", time.levels = 1)

# Classify raster based on the TWDTW analysis
r_lucc <- twdtwClassify(r_twdtw, progress = 'text')

# Plot TWDTW classification results
plot(r_lucc, type = "map")

# Assess classification
twdtw_assess <-
  twdtwAssess(object = r_lucc, y = validation_samples,
              proj4string = proj_str, conf.int = .95)

# Plot map accuracy

```

```

plot(twdtw_assess, type = "accuracy")

# Plot area uncertainty
plot(twdtw_assess, type = "area")

# Plot misclassified samples
plot(twdtw_assess, type = "map", samples = "incorrect")

# Get latex table with error matrix
twdtwXtable(twdtw_assess, table.type = "matrix")

# Get latex table with error accuracy
twdtwXtable(twdtw_assess, table.type = "accuracy")

# Get latex table with area uncertainty
twdtwXtable(twdtw_assess, table.type = "area")

## End(Not run)

## Not run:

# Example of TWDTW analysis using raster files
library(dtwSat)
library(caret)

# Load raster data
evi <- brick(system.file("lucc_MT/data/evi.tif", package = "dtwSat"))
ndvi <- brick(system.file("lucc_MT/data/ndvi.tif", package = "dtwSat"))
red <- brick(system.file("lucc_MT/data/red.tif", package = "dtwSat"))
blue <- brick(system.file("lucc_MT/data/blue.tif", package = "dtwSat"))
nir <- brick(system.file("lucc_MT/data/nir.tif", package = "dtwSat"))
mir <- brick(system.file("lucc_MT/data/mir.tif", package = "dtwSat"))
doy <- brick(system.file("lucc_MT/data/doy.tif", package = "dtwSat"))
timeline <-
  scan(system.file("lucc_MT/data/timeline", package = "dtwSat"), what="date")

# Create raster time series
rts <- twdtwRaster(evi, ndvi, red, blue, nir, mir, timeline = timeline, doy = doy)

# Load field samples and projection
field_samples <-
  read.csv(system.file("lucc_MT/data/samples.csv", package = "dtwSat"))
proj_str <-
  scan(system.file("lucc_MT/data/samples_projection", package = "dtwSat"),
        what = "character")

# Split samples for training (10%) and validation (90%) using stratified sampling
set.seed(1)
I <- unlist(createDataPartition(field_samples$label, p = 0.1))
training_samples <- field_samples[I, ]
validation_samples <- field_samples[-I, ]

```



```
# Get time series form raster
training_ts <- getTimeSeries(rts, y = training_samples, proj4string = proj_str)
validation_ts <- getTimeSeries(rts, y = validation_samples, proj4string = proj_str)

# Create temporal patterns
temporal_patterns <- createPatterns(training_ts, freq = 8, formula = y ~ s(x))

# Set TWDTW weight function
log_fun <- logisticWeight(-0.1, 50)

# Run TWDTW analysis
system.time(
  r_twdtw <-
    twdtwApply(x = rts, y = temporal_patterns, weight.fun = log_fun, progress = 'text')
)

# Plot TWDTW distances for the first year
plot(r_twdtw, type = "distance", time.levels = 1)

# Classify raster based on the TWDTW analysis
r_lucc <- twdtwClassify(r_twdtw, progress = 'text')

# Plot TWDTW classification results
plot(r_lucc, type = "map")

# Assess classification
twdtw_assess <-
  twdtwAssess(object = r_lucc, y = validation_samples,
              proj4string = proj_str, conf.int = .95)

# Plot map accuracy
plot(twdtw_assess, type = "accuracy")

# Plot area uncertainty
plot(twdtw_assess, type = "area")

# Plot misclassified samples
plot(twdtw_assess, type = "map", samples = "incorrect")

# Get latex table with error matrix
twdtwXtable(twdtw_assess, table.type = "matrix")

# Get latex table with error accuracy
twdtwXtable(twdtw_assess, table.type = "accuracy")

# Get latex table with area uncertainty
twdtwXtable(twdtw_assess, table.type = "area")

## End(Not run)
```

twdtwCrossValidate *Cross Validate temporal patterns*

Description

Splits the set of time series into training and validation and computes accuracy metrics. The function uses stratified sampling and a simple random sampling for each stratum. For each data partition this function performs a TWDTW analysis and returns the Overall Accuracy, User's Accuracy, Produce's Accuracy, error matrix (confusion matrix), and a `data.frame` with the classification (Predicted), the reference classes (Reference), and the results of the TWDTW analysis.

Usage

```
## S4 method for signature 'twdtwTimeSeries'  
twdtwCrossValidate(object, times, p, ...)
```

Arguments

object	An object of class <code>twdtwTimeSeries</code> .
times	Number of partitions to create.
p	The percentage of data that goes to training. See <code>createDataPartition</code> for details.
...	Other arguments to be passed to <code>createPatterns</code> and to <code>twdtwApply</code> .

Author(s)

Victor Maus, <vwmaus1@gmail.com>

References

Maus V, Camara G, Appel M, Pebesma E (2019). "dtwSat: Time-Weighted Dynamic Time Warping for Satellite Image Time Series Analysis in R." *Journal of Statistical Software*, **88**(5), 1–31. [doi:10.18637/jss.v088.i05](https://doi.org/10.18637/jss.v088.i05).

Maus V, Camara G, Cartaxo R, Sanchez A, Ramos FM, de Queiroz GR (2016). "A Time-Weighted Dynamic Time Warping Method for Land-Use and Land-Cover Mapping." *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, **9**(8), 3729–3739. [doi:10.1109/JSTARS.2016.2517118](https://doi.org/10.1109/JSTARS.2016.2517118).

Examples

```
## Not run:  
# Data folder  
data_folder = system.file("lucc_MT/data", package = "dtwSat")  
  
# Read dates  
dates = scan(paste(data_folder,"timeline", sep = "/"), what = "dates")
```

```

# Read raster time series
evi = brick(paste(data_folder,"evi.tif", sep = "/"))
raster_timeseries = twdtwRaster(evi, timeline = dates)

# Read field samples
field_samples = read.csv(paste(data_folder,"samples.csv", sep = "/"))
table(field_samples[["label"]])

# Read field samples projection
proj_str = scan(paste(data_folder,"samples_projection", sep = "/"),
               what = "character")

# Get sample time series from raster time series
field_samples_ts = getTimeSeries(raster_timeseries,
                                y = field_samples, proj4string = proj_str)
field_samples_ts

# Run cross validation
set.seed(1)
# Define TWDTW weight function
log_fun = logisticWeight(alpha=-0.1, beta=50)
cross_validation = twdtwCrossValidate(field_samples_ts, times=3, p=0.1,
                                     freq = 8, formula = y ~ s(x, bs="cc"), weight.fun = log_fun)
cross_validation

summary(cross_validation)

plot(cross_validation)

twdtwXtable(cross_validation)

twdtwXtable(cross_validation, show.overall=FALSE)

## End(Not run)

```

```

twdtwCrossValidation-class
      class "twdtwCrossValidation"

```

Description

This class stores the results of the cross-validation.

Usage

```

## S4 method for signature 'twdtwCrossValidation'
show(object)

## S4 method for signature 'twdtwCrossValidation'
summary(object, conf.int = 0.95, ...)

```

Arguments

object	an object of class twdtwCrossValidation.
conf.int	specifies the confidence level (0-1) for interval estimation of the population mean. For more details see mean_cl_boot .
...	Other arguments. Not used.

Slots

partitions:	A list with the indices of time series used for training.
accuracy:	A list with the accuracy and other TWDTW information for each data partitions.

Author(s)

Victor Maus, <vwmaus1@gmail.com>

References

Maus V, Camara G, Appel M, Pebesma E (2019). “dtwSat: Time-Weighted Dynamic Time Warping for Satellite Image Time Series Analysis in R.” *Journal of Statistical Software*, **88**(5), 1–31. [doi:10.18637/jss.v088.i05](#).

Maus V, Camara G, Cartaxo R, Sanchez A, Ramos FM, de Queiroz GR (2016). “A Time-Weighted Dynamic Time Warping Method for Land-Use and Land-Cover Mapping.” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, **9**(8), 3729–3739. [doi:10.1109/JSTARS.2016.2517118](#).

See Also

[twdtwMatches-class](#), [createPatterns](#), and [twdtwApply](#).

Examples

```
## Not run:
# Data folder
data_folder = system.file("lucc_MT/data", package = "dtwSat")

# Read dates
dates = scan(paste(data_folder,"timeline", sep = "/"), what = "dates")

# Read raster time series
evi = brick(paste(data_folder,"evi.tif", sep = "/"))
raster_timeseries = twdtwRaster(evi, timeline = dates)

# Read field samples
field_samples = read.csv(paste(data_folder,"samples.csv", sep = "/"))
table(field_samples[["label"]])

# Read field samples projection
proj_str = scan(paste(data_folder,"samples_projection", sep = "/"),
               what = "character")
```

```

# Get sample time series from raster time series
field_samples_ts = getTimeSeries(raster_timeseries,
  y = field_samples, proj4string = proj_str)
field_samples_ts

# Run cross validation
set.seed(1)
# Define TWDTW weight function
log_fun = logisticWeight(alpha=-0.1, beta=50)
cross_validation = twdtwCrossValidate(field_samples_ts, times=3, p=0.1,
  freq = 8, formula = y ~ s(x, bs="cc"), weight.fun = log_fun)

cross_validation

summary(cross_validation)

plot(cross_validation)

## End(Not run)

```

```
twdtwMatches-class    class "twdtwMatches"
```

Description

Class for Time-Weighted Dynamic Time Warping results.

Usage

```

## S4 method for signature 'ANY'
twdtwMatches(timeseries = NULL, patterns = NULL, alignments = NULL)

## S4 method for signature 'twdtwMatches'
index(x)

## S4 method for signature 'twdtwMatches'
length(x)

## S4 method for signature 'twdtwMatches'
as.list(x)

## S4 method for signature 'twdtwRaster'
as.list(x)

## S4 method for signature 'twdtwMatches,ANY,ANY,ANY'
x[i, j, drop = TRUE]

## S4 method for signature 'twdtwMatches,numeric,ANY'

```

```
x[[i, j, drop = TRUE]]

## S4 method for signature 'twdtwMatches'
labels(object)

## S4 method for signature 'twdtwMatches'
show(object)

## S4 method for signature 'ANY'
is.twdtwMatches(x)
```

Arguments

timeseries	a twdtwTimeSeries object.
patterns	a twdtwTimeSeries object.
alignments	an object of class list with the TWDTW results with the same length as <code>timeseries</code> or a list of <code>twdtwMatches</code> .
x	an object of class <code>twdtwMatches</code> .
i	indices of the time series.
j	indices of the pattern.
drop	if TRUE returns a data.frame, if FALSE returns a list. Default is TRUE.
object	an object of class <code>twdtwMatches</code> .
labels	a vector with labels of the time series.

Methods (by generic)

- `twdtwMatches(ANY)`: Create object of class `twdtwMatches`.
- `is.twdtwMatches(ANY)`: Check if the object belongs to the class `twdtwMatches`.

Slots

timeseries: An object of class [twdtwTimeSeries-class](#) with the satellite time series.

pattern: An object of class [twdtwTimeSeries-class](#) with the temporal patterns.

alignments: A [list](#) of TWDTW results with the same length as the `timeseries`. Each element in this list has the following results for each temporal pattern in `patterns`:
from: a vector with the starting dates of each match in the format "YYYY-MM-DD",
to: a vector with the ending dates of each match in the format "YYYY-MM-DD",
distance: a vector with TWDTW dissimilarity measure, and
K: the number of matches of the pattern.

This list might have additional elements: if `keep=TRUE` in the `twdtwApply` call the list is extended to include internal structures used during the TWDTW computation:
costMatrix: cumulative cost matrix,
directionMatrix: directions of steps that would be taken from each element of matrix,
startingMatrix: the starting points of each element of the matrix,
stepPattern: [stepPattern](#) used for the computation, see package [dtw](#),

N: the length of the pattern,
M: the length of the time series timeseries,
timeWeight: time weight matrix,
localMatrix: local cost matrix,
matching: A list whose elements have the matching points for each match between pattern
the time series, such that:
-index1: a vector with matching points of the pattern, and
-index2: a vector with matching points of the time series.

Author(s)

Victor Maus, <vwmaus1@gmail.com>

References

Maus V, Camara G, Appel M, Pebesma E (2019). “dtwSat: Time-Weighted Dynamic Time Warping for Satellite Image Time Series Analysis in R.” *Journal of Statistical Software*, **88**(5), 1–31. doi:10.18637/jss.v088.i05.

Maus V, Camara G, Cartaxo R, Sanchez A, Ramos FM, de Queiroz GR (2016). “A Time-Weighted Dynamic Time Warping Method for Land-Use and Land-Cover Mapping.” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, **9**(8), 3729–3739. doi:10.1109/JSTARS.2016.2517118.

See Also

[twdtwApply](#), [twdtwTimeSeries-class](#), and [twdtwRaster-class](#)

Examples

```
ts = twdtwTimeSeries(timeseries=MOD13Q1.ts.list)
patterns = twdtwTimeSeries(timeseries=MOD13Q1.patterns.list)
matches = twdtwApply(x = ts, y = patterns, keep=TRUE, legacy=TRUE)
class(matches)
length(matches)
matches
# Creating objects of class twdtwMatches
ts = twdtwTimeSeries(MOD13Q1.ts.list)
patt = twdtwTimeSeries(MOD13Q1.patterns.list)
mat = twdtwApply(ts, patt, weight.fun = logisticWeight(-0.1, 100),
                 keep=TRUE, legacy=TRUE)
mat = twdtwMatches(ts, patterns=patt, alignments=mat)
mat
```

twdtwRaster-class *class "twdtwRaster"*

Description

Class for set of satellite time series.

Usage

```
## S4 method for signature 'ANY'
twdtwRaster(
  ...,
  timeline,
  doy = NULL,
  layers = NULL,
  labels = NULL,
  levels = NULL,
  filepath = NULL
)

## S4 method for signature 'twdtwRaster'
dim(x)

## S4 method for signature 'twdtwRaster'
res(x)

## S4 method for signature 'twdtwRaster'
extent(x, y, ...)

## S4 method for signature 'twdtwRaster,ANY'
writeRaster(x, filepath = ".", ...)

## S4 method for signature 'twdtwRaster'
ncol(x)

## S4 method for signature 'twdtwRaster'
nrow(x)

## S4 method for signature 'twdtwRaster'
nlayers(x)

## S4 method for signature 'twdtwRaster'
levels(x)

## S4 method for signature 'twdtwRaster'
layers(x)
```



```

## S4 method for signature 'twdtwRaster'
coverages(x)

## S4 method for signature 'twdtwRaster'
bands(x)

## S4 method for signature 'twdtwRaster'
names(x)

## S4 method for signature 'twdtwRaster'
index(x)

## S4 method for signature 'twdtwRaster'
length(x)

## S4 method for signature 'twdtwRaster,ANY,ANY,ANY'
x[i]

## S4 method for signature 'twdtwRaster,ANY,ANY'
x[[i]]

## S4 method for signature 'twdtwRaster'
labels(object)

## S4 method for signature 'twdtwRaster'
crop(x, y, ...)

## S4 method for signature 'twdtwRaster'
coordinates(obj, ...)

## S4 method for signature 'twdtwRaster'
extent(x, y, ...)

## S4 method for signature 'twdtwRaster'
show(object)

## S4 method for signature 'ANY'
is.twdtwRaster(x)

## S4 method for signature 'twdtwRaster'
projecttwdtwRaster(x, crs, ...)

```

Arguments

...	objects of class RasterBrick-class or RasterStack-class .
timeline	a vector with the dates of the satellite images in the format of "YYYY-MM-DD".
doy	A RasterBrick-class or RasterStack-class with a sequence of days of the year for each pixel. doy must have the same spatial and temporal extents as the

	Raster* objects passed to If doy is not provided then at least one Raster* object must be passed through
layers	a vector with the names of the Raster* objects passed to ". . .". If not provided the layers are set to the names of objects in ". . .".
labels	a vector of class <code>character</code> with labels of the values in the Raster* objects. This is useful for categorical Raster* values of land use classes.
levels	a vector of class <code>numeric</code> with levels of the values in the Raster* objects. This is useful for categorical Raster* values of land use classes.
filepath	A character. The path to save the raster time series. If provided the function saves a raster file for each Raster* object in the list, <i>i.e</i> one file for each time series. This way the function retrieves a list of <code>RasterBrick-class</code> . It is useful when the time series are originally stored in separated files. See details.
x	an object of class <code>twdtwRaster</code> .
y	Extent object, or any object from which an Extent object can be extracted.
i	indices of the time series.
object	an object of class <code>twdtwRaster</code> .
obj	object of class <code>twdtwRaster</code> .
crs	character or object of class 'CRS'. PROJ.4 description of the coordinate reference system. For other arguments and more details see projectRaster .

Details

The performance of the functions `twdtwApply` and `getTimeSeries` is improved if the Raster* objects are connected to files with the whole time series for each attribute.

Methods (by generic)

- `twdtwRaster(ANY)`: Create object of class `twdtwRaster`.
- `is.twdtwRaster(ANY)`: Check if the object belongs to the class `twdtwRaster`.
- `projecttwdtwRaster(twdtwRaster)`: project `twdtwRaster` object.

Slots

`timeseries`: A list of multi-layer Raster* objects with the satellite image time series.

`timeline`: A vector of class `date` with dates of the satellite images in `timeseries`.

`layers`: A vector of class `character` with the names of the Raster* objects.

`labels`: A vector of class `factor` with levels and labels of the values in the Raster* objects. This is useful for categorical Raster* values of land use classes.

Author(s)

Victor Maus, <vwmaus1@gmail.com>

References

Maus V, Camara G, Appel M, Pebesma E (2019). “dtwSat: Time-Weighted Dynamic Time Warping for Satellite Image Time Series Analysis in R.” *Journal of Statistical Software*, **88**(5), 1–31. doi:10.18637/jss.v088.i05.

Maus V, Camara G, Cartaxo R, Sanchez A, Ramos FM, de Queiroz GR (2016). “A Time-Weighted Dynamic Time Warping Method for Land-Use and Land-Cover Mapping.” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, **9**(8), 3729–3739. doi:10.1109/JSTARS.2016.2517118.

See Also

[twdtwApply](#), [getTimeSeries](#), [twdtwMatches-class](#), and [twdtwTimeSeries-class](#)

Examples

```
# Creating a new object of class twdtwTimeSeries
evi = brick(system.file("lucc_MT/data/evi.tif", package="dtwSat"))
timeline = scan(system.file("lucc_MT/data/timeline", package="dtwSat"), what="date")
rts = new("twdtwRaster", timeseries = evi, timeline = timeline)

## Not run:
# Creating objects of class twdtwRaster
evi = brick(system.file("lucc_MT/data/evi.tif", package="dtwSat"))
timeline = scan(system.file("lucc_MT/data/timeline", package="dtwSat"), what="date")
ts_evi = twdtwRaster(evi, timeline=timeline)

ndvi = brick(system.file("lucc_MT/data/ndvi.tif", package="dtwSat"))
blue = brick(system.file("lucc_MT/data/blue.tif", package="dtwSat"))
red = brick(system.file("lucc_MT/data/red.tif", package="dtwSat"))
nir = brick(system.file("lucc_MT/data/nir.tif", package="dtwSat"))
mir = brick(system.file("lucc_MT/data/mir.tif", package="dtwSat"))
doy = brick(system.file("lucc_MT/data/doy.tif", package="dtwSat"))
rts = twdtwRaster(doy, evi, ndvi, blue, red, nir, mir, timeline = timeline)

## End(Not run)
```

twdtwReduceTime

Faster version of TWDTW apply

Description

This function is a faster implementation of [twdtwApply](#) that is in average 4x faster. The time weight function is coded in Fortran. It does not keep any intermediate data. It performs a multidimensional TWDTW analysis (Maus et al. 2019) and retrieves only the best matches between the unclassified time series and the patterns for each defined time interval.

Usage

```
twdtwReduceTime(
  x,
  y,
  alpha = -0.1,
  beta = 50,
  time.window = FALSE,
  dist.method = "Euclidean",
  step.matrix = symmetric1,
  from = NULL,
  to = NULL,
  by = NULL,
  breaks = NULL,
  overlap = 0.5,
  fill = length(y) + 1,
  keep = FALSE,
  ...
)
```

Arguments

x	a data.frame with the target time series. Usually, it is an unclassified time series. It must contain two or more columns, one column called date with dates in the format "YYYY-MM-DD". The other columns can have any names (e.g., red, blue, nir, evi, ndvi) as long as they match the column names in the temporal patterns y.
y	a list of data.frame objects similar to x. The temporal patterns used to classify the time series in x.
alpha	Numeric. The steepness of TWDTW logistic weight.
beta	Numeric. The midpoint of TWDTW logistic weight.
time.window	logical. TRUE will constrain the TWDTW computation to the value of the parameter beta defined in the logistic weight function. Default is FALSE.
dist.method	A character. Method to derive the local cost matrix. Default is "Euclidean" see dist in package proxy .
step.matrix	See stepPattern in package dtw (Giorgino 2009).
from	A character or Dates object in the format "yyyy-mm-dd".
to	A character or Dates object in the format "yyyy-mm-dd".
by	A character with the interval size, e.g. "6 month".
breaks	A vector of class Dates . This replaces the arguments from, to, and by.
overlap	A number between 0 and 1. The minimum overlapping between one match and the interval of classification. Default is 0.5, i.e. an overlap minimum of 50%.
fill	An integer to fill the classification gaps.
keep	Preserves the cost matrix, inputs, and other internal structures. Default is FALSE. For plot methods use keep=TRUE.
...	Arguments to pass to writeRaster and pbCreate

Author(s)

Victor Maus, <vwmaus1@gmail.com>

Examples

```
## Not run:

library(dtwSat)
from = "2009-09-01"
to = "2017-09-01"
by = "12 month"

# S4 objects for original implementation
tw_patt = readRDS(system.file("lucc_MT/patterns/patt.rds", package = "dtwSat"))
tw_ts = twdtwTimeSeries(MOD13Q1.ts)

# Table from csv for faster version
mn_patt <- lapply(dir(system.file("lucc_MT/patterns", package = "dtwSat"),
  pattern = ".csv$", full.names = TRUE), read.csv, stringsAsFactors = FALSE)
mn_ts <- read.csv(system.file("reduce_time/ts_MODIS13Q1.csv", package = "dtwSat"),
  stringsAsFactors = FALSE)

# Benchmark
rbenchmark::benchmark(
  legacy_twdtw = twdtwClassify(twdtwApply(x = tw_ts, y = tw_patt, weight.fun = log_fun),
    from = from, to = to, by = by)[[1]],
  fast_twdtw = twdtwReduceTime(x = mn_ts, y = mn_patt, rom = from, to = to, by = by)
)

## End(Not run)
```

twdtwTimeSeries-class *class "twdtwTimeSeries"*

Description

Class for setting irregular time series.

Usage

```
## S4 method for signature 'ANY'
twdtwTimeSeries(..., labels = NULL)

## S4 method for signature 'twdtwTimeSeries'
dim(x)

## S4 method for signature 'twdtwTimeSeries'
index(x)
```

```

## S4 method for signature 'twdtwTimeSeries'
nrow(x)

## S4 method for signature 'twdtwTimeSeries'
ncol(x)

## S4 method for signature 'twdtwTimeSeries'
length(x)

## S4 method for signature 'twdtwTimeSeries'
as.list(x)

## S4 method for signature 'twdtwTimeSeries'
as.data.frame(x)

## S4 method for signature 'twdtwTimeSeries,ANY,ANY,ANY'
x[i]

## S4 method for signature 'twdtwTimeSeries,ANY,ANY'
x[[i]]

## S4 method for signature 'twdtwTimeSeries'
labels(object)

## S4 method for signature 'twdtwTimeSeries'
levels(x)

## S4 method for signature 'twdtwTimeSeries'
show(object)

## S4 method for signature 'ANY'
as.twdtwTimeSeries(x)

## S4 method for signature 'ANY'
is.twdtwTimeSeries(x)

```

Arguments

...	twdtwTimeSeries objects, zoo objects or a list of zoo objects.
labels	a vector with labels of the time series.
x	an object of class twdtwTimeSeries .
i	indices of the time series.
object	an object of class twdtwTimeSeries .

Methods (by generic)

- [twdtwTimeSeries\(ANY\)](#): Create object of class [twdtwTimeSeries](#).

- `as.twdtwTimeSeries(ANY)`: convert list of data.frame to class `twdtwTimeSeries`.
- `is.twdtwTimeSeries(ANY)`: Check if the object belongs to the class `twdtwTimeSeries`.

Slots

`timeseries`: A list of `zoo` objects.

`labels`: A vector of class `factor` with time series labels.

Author(s)

Victor Maus, <vwmaus1@gmail.com>

References

Maus V, Camara G, Appel M, Pebesma E (2019). “dtwSat: Time-Weighted Dynamic Time Warping for Satellite Image Time Series Analysis in R.” *Journal of Statistical Software*, **88**(5), 1–31. doi:10.18637/jss.v088.i05.

Maus V, Camara G, Cartaxo R, Sanchez A, Ramos FM, de Queiroz GR (2016). “A Time-Weighted Dynamic Time Warping Method for Land-Use and Land-Cover Mapping.” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, **9**(8), 3729–3739. doi:10.1109/JSTARS.2016.2517118.

See Also

`twdtwMatches-class`, `twdtwRaster-class`, `getTimeSeries`, and `twdtwApply`

Examples

```
# Creating a new object of class twdtwTimeSeries
ptt = new("twdtwTimeSeries", timeseries = MOD13Q1.patterns.list,
         labels = names(MOD13Q1.patterns.list))
class(ptt)
labels(ptt)
levels(ptt)
length(ptt)
nrow(ptt)
ncol(ptt)
dim(ptt)
# Creating objects of class twdtwTimeSeries from zoo objects
ts = twdtwTimeSeries(MOD13Q1.ts)
ts

# Creating objects of class twdtwTimeSeries from list of zoo objects
patt = twdtwTimeSeries(MOD13Q1.patterns.list)
patt

# Joining objects of class twdtwTimeSeries
tsA = twdtwTimeSeries(MOD13Q1.ts.list[[1]], labels = "A")
tsB = twdtwTimeSeries(B = MOD13Q1.ts.list[[2]])
ts = twdtwTimeSeries(tsA, tsB, C=MOD13Q1.ts)
ts
```

twdtwXtable

LaTeX table from accuracy metrics

Description

Creates LaTeX table from accuracy metrics

Usage

```
## S4 method for signature 'twdtwAssessment'
twdtwXtable(
  object,
  table.type = "accuracy",
  show.prop = TRUE,
  category.name = NULL,
  category.type = NULL,
  rotate.col = FALSE,
  time.labels = NULL,
  caption = NULL,
  digits = 2,
  show.footnote = TRUE,
  ...
)

## S4 method for signature 'twdtwCrossValidation'
twdtwXtable(
  object,
  conf.int = 0.95,
  show.overall = TRUE,
  category.name = NULL,
  category.type = NULL,
  caption = NULL,
  digits = 2,
  show.footnote = TRUE,
  ...
)
```

Arguments

object	an object of class twdtwAssessment.
table.type	Table type, 'accuracy' for User's and Producer's Accuracy, 'errormatrix' for error matrix, and 'area' for area and uncertainty. Default is 'accuracy'.
show.prop	If TRUE shows the estimated proportion of area. Used with table.type='accuracy'. Default is TRUE.

category.name	A character vector defining the class names. If NULL the class names in the object x are used. Default is NULL.
category.type	A character defining the categories type "numeric" or "letter", if NULL the class names are used. Default is NULL.
rotate.col	Rotate class column names in latex table. Default is FALSE.
time.labels	A character or numeric for the time period or NULL to include all classified periods. Default is NULL.
caption	The table caption.
digits	Number of digits to show.
show.footnote	Show confidence interval in the footnote.
...	Other arguments to pass to <code>print.xtable</code> .
conf.int	Specifies the confidence level (0-1).
show.overall	If TRUE shows the overall accuracy of the cross-validation. Default is TRUE.

Author(s)

Victor Maus, <vwmaus1@gmail.com>

References

Maus V, Camara G, Appel M, Pebesma E (2019). "dtwSat: Time-Weighted Dynamic Time Warping for Satellite Image Time Series Analysis in R." *Journal of Statistical Software*, **88**(5), 1–31. [doi:10.18637/jss.v088.i05](https://doi.org/10.18637/jss.v088.i05).

Maus V, Camara G, Cartaxo R, Sanchez A, Ramos FM, de Queiroz GR (2016). "A Time-Weighted Dynamic Time Warping Method for Land-Use and Land-Cover Mapping." *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, **9**(8), 3729–3739. [doi:10.1109/JSTARS.2016.2517118](https://doi.org/10.1109/JSTARS.2016.2517118).

See Also

[twdtwAssess](#) and [twdtwAssessment](#).

Examples

```
## Not run:

# Create raster time series
evi = brick(system.file("lucc_MT/data/evi.tif", package="dtwSat"))
ndvi = brick(system.file("lucc_MT/data/ndvi.tif", package="dtwSat"))
red = brick(system.file("lucc_MT/data/red.tif", package="dtwSat"))
blue = brick(system.file("lucc_MT/data/blue.tif", package="dtwSat"))
nir = brick(system.file("lucc_MT/data/nir.tif", package="dtwSat"))
mir = brick(system.file("lucc_MT/data/mir.tif", package="dtwSat"))
doy = brick(system.file("lucc_MT/data/doy.tif", package="dtwSat"))
timeline = scan(system.file("lucc_MT/data/timeline", package="dtwSat"), what="date")
rts = twdtwRaster(evi, ndvi, red, blue, nir, mir, timeline = timeline, doy = doy)
```

```
# Read field samples
field_samples = read.csv(system.file("lucc_MT/data/samples.csv", package="dtwSat"))
proj_str = scan(system.file("lucc_MT/data/samples_projection",
                           package="dtwSat"), what = "character")

# Split samples for training (10%) and validation (90%) using stratified sampling
library(caret)
set.seed(1)
I = unlist(createDataPartition(field_samples$label, p = 0.1))
training_samples = field_samples[I,]
validation_samples = field_samples[-I,]

# Create temporal patterns
training_ts = getTimeSeries(rts, y = training_samples, proj4string = proj_str)
temporal_patterns = createPatterns(training_ts, freq = 8, formula = y ~ s(x))

# Run TWDTW analysis for raster time series
log_fun = weight.fun=logisticWeight(-0.1,50)
r_twdtw = twdtwApply(x=rts, y=temporal_patterns, weight.fun=log_fun, format="GTiff",
                    overwrite=TRUE)

# Classify raster based on the TWDTW analysis
r_lucc = twdtwClassify(r_twdtw, format="GTiff", overwrite=TRUE)
plot(r_lucc)

# Assess classification
twdtw_assess = twdtwAssess(object = r_lucc, y = validation_samples,
                          proj4string = proj_str, conf.int=.95)
twdtw_assess

# Create latex tables
twdtwXtable(twdtw_assess, table.type="errormatrix", rotate.col=TRUE,
            caption="Error matrix", digits=2, comment=FALSE)
twdtwXtable(twdtw_assess, table.type="accuracy", category.type="letter",
            caption="Accuracy metrics.")
twdtwXtable(twdtw_assess, table.type="area", category.type="letter",
            digits = 0, caption="Area and uncertainty")

## End(Not run)
```

Index

- * **datasets**
 - MOD13Q1.MT.yearly.patterns, [12](#)
 - MOD13Q1.patterns.list, [13](#)
 - MOD13Q1.ts, [14](#)
 - MOD13Q1.ts.labels, [15](#)
 - MOD13Q1.ts.list, [15](#)
- [, twdtwMatches, ANY, ANY, ANY-method (twdtwMatches-class), [69](#)
- [, twdtwRaster, ANY, ANY, ANY-method (twdtwRaster-class), [72](#)
- [, twdtwTimeSeries, ANY, ANY, ANY-method (twdtwTimeSeries-class), [77](#)
- [[, twdtwMatches, numeric, ANY-method (twdtwMatches-class), [69](#)
- [[, twdtwRaster, ANY, ANY-method (twdtwRaster-class), [72](#)
- [[, twdtwTimeSeries, ANY, ANY-method (twdtwTimeSeries-class), [77](#)

- as.data.frame, twdtwTimeSeries-method (twdtwTimeSeries-class), [77](#)
- as.list, twdtwMatches-method (twdtwMatches-class), [69](#)
- as.list, twdtwRaster-method (twdtwMatches-class), [69](#)
- as.list, twdtwTimeSeries-method (twdtwTimeSeries-class), [77](#)
- as.twdtwTimeSeries (twdtwTimeSeries-class), [77](#)
- as.twdtwTimeSeries, ANY-method (twdtwTimeSeries-class), [77](#)

- bands (twdtwRaster-class), [72](#)
- bands, twdtwRaster-method (twdtwRaster-class), [72](#)

- character, [4](#), [20](#), [22](#), [24](#), [26](#), [28](#), [30](#), [34](#), [37](#), [38](#), [45](#), [57](#), [74](#), [76](#)
- coordinates, twdtwRaster-method (twdtwRaster-class), [72](#)

- coverages (twdtwRaster-class), [72](#)
- coverages, twdtwRaster-method (twdtwRaster-class), [72](#)
- createDataPartition, [66](#)
- createPatterns, [3](#), [12–14](#), [46](#), [66](#), [68](#)
- createPatterns, twdtwTimeSeries-method (createPatterns), [3](#)
- createPatterns-twdtwMatches (createPatterns), [3](#)
- crop, [41](#)
- crop, twdtwRaster-method (twdtwRaster-class), [72](#)

- data.frame, [9](#), [15](#), [49](#), [66](#)
- date, [74](#)
- Dates, [4](#), [7](#), [13–15](#), [45](#), [56](#), [57](#), [76](#)
- dim, twdtwRaster-method (twdtwRaster-class), [72](#)
- dim, twdtwTimeSeries-method (twdtwTimeSeries-class), [77](#)
- dist, [44](#), [76](#)
- dtw, [70](#)
- dtwSat, [5](#)

- extent, twdtwRaster-method (twdtwRaster-class), [72](#)

- factor, [74](#), [79](#)
- function, [10](#), [11](#)

- gam, [4](#)
- get, [6](#)
- getAlignments (get), [6](#)
- getAlignments, twdtwMatches-method (get), [6](#)
- getDatesFromDOY, [7](#)
- getInternals (get), [6](#)
- getInternals, twdtwMatches-method (get), [6](#)
- getMatches (get), [6](#)

- getMatches, twdtwMatches-method (get), 6
- getPatterns (getTimeSeries), 8
- getPatterns, twdtwMatches-method (getTimeSeries), 8
- getPatterns-twdtwMatches (getTimeSeries), 8
- getTimeSeries, 5, 8, 46, 74, 75, 79
- getTimeSeries, twdtwMatches-method (getTimeSeries), 8
- getTimeSeries, twdtwRaster-method (getTimeSeries), 8
- getTimeSeries, twdtwTimeSeries-method (getTimeSeries), 8
- getTimeSeries-twdtwMatches (getTimeSeries), 8
- getTimeSeries-twdtwRaster (getTimeSeries), 8
- getTimeSeries-twdtwTimeSeries (getTimeSeries), 8
- ggplot, 17–19, 21, 22, 24, 26–28, 30, 31, 34, 36–38
- index, twdtwMatches-method (twdtwMatches-class), 69
- index, twdtwRaster-method (twdtwRaster-class), 72
- index, twdtwTimeSeries-method (twdtwTimeSeries-class), 77
- integer, 20, 26, 34, 37, 38
- is.twdtwMatches (twdtwMatches-class), 69
- is.twdtwMatches, ANY-method (twdtwMatches-class), 69
- is.twdtwRaster (twdtwRaster-class), 72
- is.twdtwRaster, ANY-method (twdtwRaster-class), 72
- is.twdtwTimeSeries (twdtwTimeSeries-class), 77
- is.twdtwTimeSeries, ANY-method (twdtwTimeSeries-class), 77
- labels, twdtwMatches-method (twdtwMatches-class), 69
- labels, twdtwRaster-method (twdtwRaster-class), 72
- labels, twdtwTimeSeries-method (twdtwTimeSeries-class), 77
- layers (twdtwRaster-class), 72
- layers, twdtwRaster-method (twdtwRaster-class), 72
- length, twdtwMatches-method (twdtwMatches-class), 69
- length, twdtwRaster-method (twdtwRaster-class), 72
- length, twdtwTimeSeries-method (twdtwTimeSeries-class), 77
- levels, twdtwRaster-method (twdtwRaster-class), 72
- levels, twdtwTimeSeries-method (twdtwTimeSeries-class), 77
- linearWeight, 10
- list, 70
- logisticWeight, 11
- mean_cl_boot, 68
- MOD13Q1.MT.yearly.patterns, 12
- MOD13Q1.patterns.list, 13, 14, 16
- MOD13Q1.ts, 14, 14, 15, 16
- MOD13Q1.ts.labels, 15
- MOD13Q1.ts.list, 14, 15
- names, twdtwRaster-method (twdtwRaster-class), 72
- ncol, twdtwRaster-method (twdtwRaster-class), 72
- ncol, twdtwTimeSeries-method (twdtwTimeSeries-class), 77
- nlayers, twdtwRaster-method (twdtwRaster-class), 72
- nrow, twdtwRaster-method (twdtwRaster-class), 72
- nrow, twdtwTimeSeries-method (twdtwTimeSeries-class), 77
- numeric, 22, 24, 28, 30, 74
- pbCreate, 45, 56, 76
- plot, 16
- plot, twdtwAssessment, ANY-method (plot), 16
- plot, twdtwCrossValidation, ANY-method (plot), 16
- plot, twdtwMatches, ANY-method (plot), 16
- plot, twdtwRaster, ANY-method (plot), 16
- plot, twdtwTimeSeries, ANY-method (plot), 16
- plot-twdtwAssessment (plot), 16
- plot-twdtwMatches (plot), 16
- plot-twdtwRaster (plot), 16
- plot-twdtwTimeSeries (plot), 16

- plotAccuracy, [17, 32](#)
- plotAdjustedArea, [19, 32](#)
- plotAlignments, [17, 20, 26, 28, 35, 36](#)
- plotArea, [17, 21, 24, 29, 30](#)
- plotChanges, [17, 22, 23, 29, 30](#)
- plotClassification, [17, 21, 25, 28, 35, 36](#)
- plotCostMatrix, [17, 21, 26, 27, 35, 36](#)
- plotDistance, [22, 24, 28, 29, 30](#)
- plotMaps, [17, 22, 24, 29](#)
- plotMapSamples, [31](#)
- plotMatches, [17, 21, 26, 28, 34, 36](#)
- plotPaths, [17, 21, 26, 28, 35, 35](#)
- plotPatterns, [17, 36, 38](#)
- plotTimeSeries, [17, 37, 37](#)
- print.xtable, [81](#)
- projectRaster, [74](#)
- projecttwdtwRaster (twdtwRaster-class), [72](#)
- projecttwdtwRaster, twdtwRaster-method (twdtwRaster-class), [72](#)

- res, twdtwRaster-method (twdtwRaster-class), [72](#)
- resampleTimeSeries, [39, 44](#)
- resampleTimeSeries, twdtwTimeSeries-method (resampleTimeSeries), [39](#)
- resampleTimeSeries-twdtwMatches (resampleTimeSeries), [39](#)

- scale_fill_manual, [22, 24, 30](#)
- shiftDates, [8, 40](#)
- shiftDates, list-method (shiftDates), [40](#)
- shiftDates, twdtwTimeSeries-method (shiftDates), [40](#)
- shiftDates, zoo-method (shiftDates), [40](#)
- shiftDates-list (shiftDates), [40](#)
- shiftDates-twdtwTimeSeries (shiftDates), [40](#)
- shiftDates-zoo (shiftDates), [40](#)
- show, twdtwAssessment-method (twdtwAssessment-class), [53](#)
- show, twdtwCrossValidation-method (twdtwCrossValidation-class), [67](#)
- show, twdtwMatches-method (twdtwMatches-class), [69](#)
- show, twdtwRaster-method (twdtwRaster-class), [72](#)
- show, twdtwTimeSeries-method (twdtwTimeSeries-class), [77](#)
- smean.cl.normal, [18](#)
- SpatialPointsDataFrame, [9, 49, 54](#)
- stepPattern, [44, 56, 70, 76](#)
- subset, [41](#)
- subset, twdtwMatches-method (subset), [41](#)
- subset, twdtwRaster-method (subset), [41](#)
- subset, twdtwTimeSeries-method (subset), [41](#)
- subset-twdtwMatches (subset), [41](#)
- subset-twdtwRaster (subset), [41](#)
- subset-twdtwTimeSeries (subset), [41](#)
- summary, twdtwCrossValidation-method (twdtwCrossValidation-class), [67](#)

- table, [49](#)
- twdtwApply, [5–7, 10–12, 21, 22, 24, 26, 28–30, 35, 36, 39, 43, 56, 57, 66, 68, 71, 74, 75, 79](#)
- twdtwApply, twdtwRaster-method (twdtwApply), [43](#)
- twdtwApply, twdtwTimeSeries-method (twdtwApply), [43](#)
- twdtwApply-twdtwRaster (twdtwApply), [43](#)
- twdtwApply-twdtwTimeSeries (twdtwApply), [43](#)
- twdtwAssess, [18, 20, 48, 54, 81](#)
- twdtwAssess, data.frame-method (twdtwAssess), [48](#)
- twdtwAssess, matrix-method (twdtwAssess), [48](#)
- twdtwAssess, table-method (twdtwAssess), [48](#)
- twdtwAssess, twdtwMatches-method (twdtwAssess), [48](#)
- twdtwAssess, twdtwRaster-method (twdtwAssess), [48](#)
- twdtwAssess-data.frame (twdtwAssess), [48](#)
- twdtwAssess-matrix (twdtwAssess), [48](#)
- twdtwAssess-table (twdtwAssess), [48](#)
- twdtwAssess-twdtwMatches (twdtwAssess), [48](#)
- twdtwAssess-twdtwRaster (twdtwAssess), [48](#)
- twdtwAssessment, [18–20, 31, 32, 50, 81](#)
- twdtwAssessment (twdtwAssessment-class), [53](#)

- twdtwAssessment-class, [53](#)
- twdtwClassify, [26](#), [49](#), [50](#), [54](#), [55](#)
- twdtwClassify, data.frame-method
(twdtwClassify), [55](#)
- twdtwClassify, list-method
(twdtwClassify), [55](#)
- twdtwClassify, twdtwMatches-method
(twdtwClassify), [55](#)
- twdtwClassify, twdtwRaster-method
(twdtwClassify), [55](#)
- twdtwClassify, twdtwTimeSeries-method
(twdtwClassify), [55](#)
- twdtwClassify-data.frame
(twdtwClassify), [55](#)
- twdtwClassify-list (twdtwClassify), [55](#)
- twdtwClassify-twdtwRaster
(twdtwClassify), [55](#)
- twdtwClassify-twdtwTimeSeries
(twdtwClassify), [55](#)
- twdtwCrossValidate, [66](#)
- twdtwCrossValidate, twdtwTimeSeries-method
(twdtwCrossValidate), [66](#)
- twdtwCrossValidate-twdtwTimeSeries
(twdtwCrossValidate), [66](#)
- twdtwCrossValidation, [18](#), [19](#)
- twdtwCrossValidation
(twdtwCrossValidation-class),
[67](#)
- twdtwCrossValidation-class, [67](#)
- twdtwMatches, [20](#), [26](#), [27](#), [34](#), [35](#), [49](#)
- twdtwMatches (twdtwMatches-class), [69](#)
- twdtwMatches, ANY-method
(twdtwMatches-class), [69](#)
- twdtwMatches-class, [69](#)
- twdtwMatches-create
(twdtwMatches-class), [69](#)
- twdtwRaster, [9](#), [22](#), [24](#), [28](#), [30](#), [31](#), [49](#), [54](#)
- twdtwRaster (twdtwRaster-class), [72](#)
- twdtwRaster, ANY-method
(twdtwRaster-class), [72](#)
- twdtwRaster-class, [72](#)
- twdtwRaster-create (twdtwRaster-class),
[72](#)
- twdtwReduceTime, [75](#)
- twdtwTimeSeries, [4](#), [9](#), [12](#), [37–40](#), [44](#), [66](#), [70](#),
[78](#)
- twdtwTimeSeries
(twdtwTimeSeries-class), [77](#)
- twdtwTimeSeries, ANY-method
(twdtwTimeSeries-class), [77](#)
- twdtwTimeSeries-class, [77](#)
- twdtwTimeSeries-create
(twdtwTimeSeries-class), [77](#)
- twdtwXtable, [50](#), [54](#), [80](#)
- twdtwXtable, twdtwAssessment-method
(twdtwXtable), [80](#)
- twdtwXtable, twdtwCrossValidation-method
(twdtwXtable), [80](#)
- twdtwXtable-twdtwAssessment
(twdtwXtable), [80](#)
- twdtwXtable-twdtwCrossValidation
(twdtwXtable), [80](#)
- writeRaster, [45](#), [56](#), [76](#)
- writeRaster, twdtwRaster, ANY-method
(twdtwRaster-class), [72](#)
- zoo, [13–15](#), [37](#), [38](#), [40](#), [78](#), [79](#)