Package 'FARM.eval'

November 4, 2026

Type Package

Title FARM Algorithm Evaluation

Version 1.0.0

Description Providing data sets and functions used for evaluating the FARM (Forward Alignment Relevance Measure) algorithm. Datasets include energy consumption series from a university research cube in combination with two and four exogenous series consisting 768 and 62840 samples, respectively. To run particular functions from terminal: Rscript $(R -e 'cat(system.file(``scripts'',``FARM.eval.R'',package = ``FARM.eval''))') --fexecute get.global.rel.meas --parworkers 1 --parobjsize 500.

License GPL-3

Encoding UTF-8

Imports FARM, dplyr, purrr, lubridate, readr, stats, magrittr, dtw, DTWBI, reticulate, graphics, shape, utils, future.apply, nanoparquet, plotrix, tsmp, optparse, tidyr

Depends R (>= 4.1.0)

Config/roxygen2/version 8.1.0

NeedsCompilation no

Author Ramón Christen [aut, cre] (ORCID: <https://orcid.org/0000-0002-3963-0785>)

Maintainer Ramón Christen <hiroshima@bluewin.ch>

Repository CRAN

Date/Publication 2026-11-04 09:30:02 UTC

Contents

: FARM.eval.R 2
: utilies.R 3
align.eval.dewarp.bench 4
align.eval.error.stat 5
alot 5
amss 6

amss.wpath . . . 7
calc.global.rel.meas . . . 7
dewarp.approaches . . . 8
dewarp.idx.univar . . . 8
dist.h . . . 9
dist.matrix.wpath . . . 10
edr . . . 10
edr.dist . . . 11
feat.ext . . . 12
ffalign.exp . . . 12
ffalign.exp.calc . . . 13
fit.list.lengths . . . 14
get.global.rel.meas . . . 14
harm.res . . . 15
match.slice.df . . . 15
msm . . . 16
msm.cost . . . 16
msm.dist . . . 17
multi.warp.series . . . 17
n.vect.phi . . . 18
rm.min . . . 19
robust.minmax . . . 19
scale_quant_range . . . 20
test.dewarp.idx . . . 21
text_extr . . . 21
vect.phi . . . 22
vectors.draw . . . 22
warp.idx . . . 23

Index **25**

: `FARM.eval.R` *Functions for Evaluating the FARM Algorithm*

Description

: Collection of functions that evaluate the alignment capabilities and relevance measure accuracy of the FARM algorithm (provided in R-package: FARM).

load evaluation data sets

loads, prprocesses and slices all data required for experiments. Does nothing, if variable 'ds.eval' already exist.

IMPORTANT: This package only provides the data sources recorded by the authors itself. Thus, to run this function load.data() correctly, you have to save the other data sets from the official sources in .csv files within the folder /extdata beside the existing data files. In fact, you have to add missing data files (.csv and .parquet files) while maintaining the following folder structure: ./inst/ |-extdata/ |-ResC1W.no.f.pwr.cons.csv |-ResC1W.no.f.co2.csv |-ResC1W.no.f.ecar.csv

|-ResC1Y.hslu.rc.data.large.clean.csv |-Finance.PETR4.SA_raw_data.csv |-ETTh1.csv |-ETTh2.csv |-ETTm1.csv |-ETTm2.csv |-Weather.csv |-epex-de.parquet

The additional data sets can be downloaded from: ETTx: https://github.com/zhouhaoyi/ETDataset Finance: https://finance.yahoo.com/quote/PETR4.SA/history/?period1=1514764800&period2=1672444800 Weather: https://www.ncei.noaa.gov/data/local-climatological-data/ EPEX: https://zenodo.org/records/4624805

Usage

```
load.data()
```

Details

##

Value

list: ds.eval comprising all data required for the experiments

Note

: date: 2026-09-22

Author(s)

: rch

`: utilies.R` ######################

Description

: the script provides various helpful functions for time series experiments

load data or apply method

loads (RDS) or applies and saves (RDS) data from/to a file. The function ensures that the method is applied if results are not available in provided file.

Usage

```
load.apply.save(file, method, ...)
```

Arguments

`file`	relative path and name of .rds file to load.
`method`	method to apply on data.
`...`	parameters passed to method. positional or keyword arguments allowed.

Value

result from provided method.

Note

: date: 2023-08-29

Author(s)

: rch

`align.eval.dewarp.bench` *dewarp work bench for alignment evaluation*

Description

dewarps query time series using algorithms defined in test bench and calculates alignment results with respect to the original query time series

Usage

```
align.eval.dewarp.bench(
  ts.ref,
  ts.qry,
  qry.gt,
  warp.gt,
  approaches = NULL,
  measures = NULL
)
```

Arguments

`ts.ref`	vector: reference time series. Must be of same length as ts.qry.
`ts.qry`	vector: query time series. Must be of same length as ts.ref.
`qry.gt`	vector: ground truth of query time series.
`warp.gt`	vector: series of warp width for every single data point of ground truth series
`approaches`	list of applied approaches defined in: dewarp.approaches. default: NULL (all methods)
`measures`	list of calculated evaluation measures defined in: align.eval.error.stat. default: NULL (all measures)

Value

experiment results

align.eval.error.stat *extract error statistics of alignment evaluation*

Description

returns different statistical alignment performance values for given list of alignment results

Usage

```
align.eval.error.stat(align.res, ts.gt, ts.qry, warp.gt, measures = NULL)
```

Arguments

align.res	list of alignment results
ts.gt	ground truth time series referred to for quality measures
ts.qry	query time series used for experiment
warp.gt	series of warp width for every single data point of ground truth series
measures	list of calculated evaluation measures. ['sum.d', 'mae', 'rmae', 'mape', 'corr', 'mean.p', 'slm.f', 'sum.xd', 'gl.rel', 'wp.sim', 'wp.sim.r']. default: NULL (ALL)

Value

list sum.d: sum of distance of absolute values mae: mean absolute error (sum.d / length(qts)) cor: correlation of query and reference ts mean.p: ... see farm mean.p slm.f: ... see farm slm.f sum.xd: sum of distance of 1st derivation gl.rel: global relevance value (i.e. measure value) wp.sim: sum of warp difference between de-warped result and ground truth

alot *alot dynamic distance*

Description

calculates alot distance for two given series

Usage

```
alot(x, v2 = NULL, w = 2, s = 3)
```

Arguments

x	time series: reference only if v2 query
v2	time series: query
w	window width: default = 2, as used by Ogul when algorithm was published
s	sigma: default = 3, center of set {2,3,4} also used by Ogul in alg. publication

Value

dist: alot distance, dist.m: distance matrix, path: warping path

Note

this variation is speed optimized by calculating the distance in a preceding separate step

Examples

```
alot(x = runif(100), v2 = runif(100))
alot(x = data.frame(v1 = runif(100), v2 = runif(100)))
```

---

amss *amss dynamic distance*

Description

AMSS dynamic distance calculation according to the documentation in the official publication

Usage

```
amss(x, v2 = NULL)
```

Arguments

x	input vector 1 OR data.frame with input vectors 1 and 2
v2	input vector 2 if v2 missing in x

Value

dist: AMSS distance, dist.m: distance matrix, path: warping path

Examples

```
amss(x = runif(100), v2 = runif(100))
amss(x = data.frame(v1 = runif(100), v2 = runif(100)))
```

amss.wpath *amss warping path*

Description

AMSS warping path calculation according the documentation in the official publication

Usage

```
amss.wpath(dist.m)
```

Arguments

dist.m	distance matrix between compared feature vectors

Value

warping path

calc.global.rel.meas *get.global.rel.meas handler*

Description

Runs the method get.global.rel.meas and saves all results in the file "res/eval.res.global.rel.meas.Rdata" if the file does not exist already.

Usage

```
calc.global.rel.meas()
```

Value

list with data frames for each data set providing the different relevance measures for each exogenous variable

dewarp.approaches *apply de-warping with selected dynamic methods*

Description

applies the bunch of de-warping algorithms (or parts of it) on given query and reference time series. For a reasonable comparison, all distance measures that represent the last sum value of the distance matrix, the value was divided by the length of query time series.

Usage

```
dewarp.approaches(ts.ref, ts.qry, approaches = NULL, edr.e = EDR.e)
```

Arguments

ts.ref	reference time series
ts.qry	query time series
approaches	list of applied approaches: ['pure', farm', 'farm.mtr', 'farm.mtr.nfuzz', 'amss', 'dtw', 'ddtw', 'alot', 'edr', 'msm', 'shapedtw']. default: all
edr.e	float: the EDR epsilon parameter

Value

list of de-warping results for all applied approaches

Note

de-warp series with different approaches all results must provide the list elements: - rts.decomp: decomposed reference time series - qts.decomp: decomposed query time series - path: data.frame providing warping path (x, y) - rel.global: global relevance measure - wp.univar: univariate warping path: warping compensation for every single data point

dewarp.idx.univar *de-warping univariate series*

Description

returns de-warp indices for amplitude algorithms (dtw) CAUTION: the returned de-warp size equals inverse of original warp size, ideally.

Usage

```
dewarp.idx.univar(y.warp.idx, x.warp.idx, abs)
```

Arguments

y.warp.idx	[int]: reference indexes (dtw.index2, farm.ref)
x.warp.idx	[int]: query indexes (dtw.index1, farm.qry)
abs	bool: flag if warping indexes are related to absolute values (TRUE) or derivatives (FALSE) like for FARM, AMSS

Value

ideally the inverse of the applied warping

Examples

```
dewarp.idx.univar(y.warp.idx = c(1,2,3,3,4,5,5), x.warp.idx = c(1,2,2,2,3,4,5), abs = TRUE)
```

---

dist.h *alot texture distance*

Description

alot distance calculation between extracted textures

Usage

```
dist.h(x, y)
```

Arguments

x	texture 1
y	texture 2

Value

distance between textures

dist.matrix.wpath *warping path extraction*

Description

extracts the best fitting mapping path from a distance matrix by applying the given conditional function

Usage

```
dist.matrix.wpath(dist.m, cond.f = which.min)
```

Arguments

dist.m	distance matrix
cond.f	conditional function for backtracking the mapping path - default: min

Value

warping path with x and y indexes

Examples

```
dist.matrix.wpath(matrix(runif(9), nrow=3, ncol=3))
```

edr *edr dynamic distance*

Description

the function calculates the edr distance of two trajectories. Thereby, it applies the threshold e on the abs(delta x) and abs(delta y) and returns a binary value: 0 if delta < e else 1 For time series, the condition is always true for the delta x. For this reason, the function proves the delta y only.

Usage

```
edr(x, v2 = NULL, e = 1)
```

Arguments

x	time series: reference if v2 not NULL
v2	time series: query
e	epsilon: threshold parameter

Value

dist: edr distance, dist.m: distance matrix, path: warping path

Examples

```
edr(x = runif(100), v2 = runif(100))
edr(x = data.frame(v1 = runif(100), v2 = runif(100)))
```

`edr.dist` ######################

Description

: implementation of the edr algorithm according the publication chen et al. https://doi.org/10.1145/1066157.1066213

edr feature distance

calculation of the edr distance

Usage

```
edr.dist(x, v2 = NULL, e = 1)
```

Arguments

x	data.frame(x, y) comprising comparing vectors. if it contains single vector only, it must provide dx and dy of first vector.
v2	components (dx, dy) of second vector for comparison if x represents single first vector.
e	epsilon threshold for vector comparison

Value

edr distance

Note

: date: 2023-10-19

Author(s)

: rch

feat.ext ######################

Description

: implementation of the amss algorithm according the publication Nakamura et al. https://doi.org/10.1007/s10044-011-0262-6

amss feature extraction

AMSS feature extraction of a vector

Usage

```
feat.ext(x)
```

Arguments

x	vector for feature extraction

Value

vector with features of x

Note

: date: 2023-10-19

Author(s)

: rch

---

ffalign.exp *ffalign experiment function*

Description

ffalign experiment function

Usage

```
ffalign.exp(
  df,
  vtarget = "OT",
  warp.max = c(3, 5, 8, 16),
  ds.name = NULL,
  align.appr = NULL,
  eval.meas = NULL
)
```

Arguments

df	data frame with target variable (default: OT) and all exogenous variables. date and time variable should be removed
vtarget	name of target variable if other than 'OT'
warp.max	list with maximum warping step width. default: c(3,5,8,16)
ds.name	name of data set used for saving results. deault: name of passed data.frame
align.appr	alignment approaches applied on every single experiment. default: all possible approaches
eval.meas	applied evaluation measures. default: sum.d and corr

Value

list with experiment for all exogenous data

`ffalign.exp.calc` *ffalign.exp.calc*

Description

This function defines all executed alignment experiments. It applies the ffalign.exp function on several experimental data sets and stores the results in rds files accordingly

Usage

```
ffalign.exp.calc(eval.meas = NULL, align.appr = NULL)
```

Arguments

eval.meas	vector: applied evaluation measures from set: 'sum.d', 'mae', 'rmae', 'mape', 'corr', 'mean.p', 'slm.f', 'sum.xd', 'gl.rel', 'wp.sim', 'wp.sim.r'
align.appr	vector: applied alignment approaches from set: 'pure', 'farm', 'farm.mtr', 'farm.mtr.nfuzz', 'amss', 'dtw', 'ddtw', 'alot', 'edr', 'msm', 'shapedtw'

Value

list: results of alignment experiment calculations applied on all tested datasets

fit.list.lengths *fit list size*

Description

fills shorter list elements with NA to the max length of all list elements.

Usage

```
fit.list.lengths(x, len = NA)
```

Arguments

x	list with elements of different length.
len	int length to set for each element in list x.

Value

given list for x where all entries are of the same length.

Examples

```
fit.list.lengths(list(a = c(1,2,3,4,5), b = c(1,2,3,4,5,6,7)))
```

get.global.rel.meas *calculate global relevance measure*

Description

Calculates a set of global relevance measures for each pair of reference and exogenous time series across all evaluation data sets.

Usage

```
get.global.rel.meas(ds.rel.meas = NULL, edr.e = EDR.e)
```

Arguments

ds.rel.meas	list: data sets with reference and exogenous series used for calculating the relevance measure.
edr.e	float: the EDR epsilon parameter

Value

list with data frames for each data set providing the different relevance measures for each exogenous variable

harm.res *harmonising results in dewarp.approaches*

Description

harmonizes result structure such that all result objects are of the same structure

Usage

```
harm.res(x, dec, path, rel.gl, abs)
```

Arguments

x	list: de-warping results
dec	list with decomposed series (ref.ts, qry.ts)
path	data.frame: resulting warping path (x, y)
rel.gl	number: global relevance measure
abs	boolean: absolute measures used in algorithm

match.slice.df *data.frame match-slicing*

Description

selects all data containing given match-string

Usage

```
match.slice.df(x, match, by = 1, format = c)
```

Arguments

x	sourcing data frame
match	vector with matching data for selecting recordings
by	look-up column - default: 1
format	format for type cast of look-up column - default c (vector)

Value

sliced data frame

Examples

```
match.slice.df(x = data.frame(x = c(1,2,3,2,5), y = 5:9), 2)
```

msm *msm dynamic distance*

Description

returns the msm distance for the given time series

Usage

```
msm(x, v2 = NULL, c = 1)
```

Arguments

x	time series: reference only if query given by v2
v2	time series: query
c	cost parameter used for msm.cost calculation

Value

dist: msm distance, dist.m: distance matrix, path: warping path

Examples

```
msm(x = runif(100), v2 = runif(100))
msm(x = data.frame(v1 = runif(100), v2 = runif(100)))
```

msm.cost *######################*

Description

: implementation of the msm algorithm according the publication Stefan et al. https://doi.org/10.1109/TKDE.2012.88

msm feature cost

calculates the msm cost for given series elements

Usage

```
msm.cost(x, xp, y, c)
```

Arguments

x	first element
xp	element before first element
y	second element
c	constant value > should be evaluated experimentally with the set of {0.01, 0.1, 1, 10, 100}

Value

msm cost

Note

: date: 2023-10-24

Author(s)

: rch

`msm.dist` *msm feature distance*

Description

calculates the msm distance between two elements

Usage

```
msm.dist(x, y)
```

Arguments

`x`	element 1
`y`	element 2

Value

msm distance

`multi.warp.series` *time-warp multiple series*

Description

applies warp.type warping on time series for multiple warp length. It returns a list of series that are warped (negative time shift, i.e. warped ts present features ahead in time to the given ts) by 'warp.len'.

Usage

```
multi.warp.series(ts, warp.len, warp.type)
```

Arguments

ts	vector: time series to warp
warp.len	vector with given warp length
warp.type	['const', 'sweep']: const: constant shift of given ts

Value

wrp.ts: list of warped time series

wrp.size: list of series with number of warping steps for each point in time

Examples

```
multi.warp.series(ts = 1:150, warp.len = c(5,10,20), warp.type = 'sweep')
```

---

n.vect.phi *n-dim vector angle*

Description

angle (phi) between two nD vectors

Usage

```
n.vect.phi(x, v2 = NULL, cos = FALSE)
```

Arguments

x	data.frame of vectors [v1, v2] (then v2 == NULL), else single vector
v2	vector: second vector c(...). Optional if x is single vector i.e. c(x1,y1,z1,...)
cos	bool: if cosine(phi) returned, default: FALSE

Value

phi or cosine(phi) of signed difference between vector arguments

Examples

```
n.vect.phi(data.frame(v1 = c(1,5,2), v2 = c(6,3,5)))
```

rm.min *remove minimum values*

Description

removes offset to the minimum. It is applied to positive and negative values.

Usage

```
rm.min(ts)
```

Arguments

ts	time series to apply

Value

offset removed time series with x is element of natural numbers.

Examples

```
rm.min(c(1,2,3,4,5))
```

robust.minmax *min-max normalisation*

Description

a robust min-max normalizing function that ignores outliers

Usage

```
robust.minmax(
  x,
  l.lim = 0,
  u.lim = 1,
  lower = 0.01,
  upper = 0.99,
  mean.set = NA
)
```

Arguments

x	vector: input series to apply min-max
l.lim	float: lower scale limit
u.lim	float: upper scale limit
lower	float: lower percentile
upper	float: upper percentile
mean.set	float: subtracted mean offset

Value

min-max normalized series

Examples

```
robust.minmax(x = runif(20))
```

---

scale_quant_range *scale by quantiles*

Description

normalize data to range [0,1] based on a reduced set excluding outliers. The factor for normalization is applied on full series including outliers

Usage

```
scale_quant_range(x, range = 0.95)
```

Arguments

x	vector: data set (time series) to normalize
range	default: 0.95: percentage range of considered data for excluding outliers

Value

normalized data including outliers where given band (excluding outliers) is in the range [0,1].

Examples

```
scale_quant_range(x = runif(20))
```

test.dewarp.idx *test de-warping function*

Description

used for testing the function dewarp.idx.univar

Usage

```
test.dewarp.idx()
```

text_extr *#################*

Description

Implementation of the ALoT algorithm

alot texture extraction

alot texture extraction for passed vector and window size

Usage

```
text_extr(x, w)
```

Arguments

x	vector to extract textures from
w	window size for texture extraction

Value

extracted textures for input vector x

Note

: date: 2023-10-17

Author(s)

: rch

vect.phi *vector angle*

Description

angle (phi) between two 2D vectors

Usage

```
vect.phi(x, v2 = NULL, cos = FALSE)
```

Arguments

x	data.frame of vectors [$x, $y] (then v2 == NULL) or single vector with x-y component c(x,y)
v2	vector with x-y component c(x,y). Optional if x is single vector i.e. c(x,y)
cos	bool: if cosine(phi) returned, default: FALSE

Value

phi or cosine(phi) of signed difference between vector arguments

Examples

```
vect.phi(data.frame(x = c(2,5,3), y = c(1,2,7)))
```

vectors.draw *vector drawing*

Description

draws vectors in a new plot. it is restricted to 'positive' vectors (i.e. abs(x) >= abs(xo) & abs(y) >= abs(yo))

Usage

```
vectors.draw(
  x,
  y,
  xo = 0,
  yo = 0,
  vectors,
  xlim = NA,
  ylim = NA,
  lty = NA,
```

```
  colors = 1,
  vlines = FALSE,
  zabline = TRUE,
  arc.r = NA,
  symbols = c(),
  add.notes = NA
)
```

Arguments

x	vector: with x-end coordinates
y	vector: with y-end coordinates
xo	vector: with x-origin coordinates. Filled up with 0 if not of length x - default: 0
yo	vector: with y-origin coordinates. Filled up with 0 if not of length y - default: 0
vectors	list of vectors [[dx1,dy1], [dx2,dy2], [dx3,dy3]]
xlim	x-limits c(start, end) - default: range(xo,x)
ylim	y-limits c(start, end) - default: range(yo,y)
lty	line type for vectors: default: NA
colors	vector: colors for vectors - default: NA
vlines	vertical dash lines from vector end to base line: default F
zabline	boolean: if set, it draws a horizontal grey30 line on level y=0 - default: T
arc.r	vector: radius for angle arc (only applicable if symbols available). default length of 0.33 * min(x, y) is used if 0 - default: NA
symbols	vector: of expressions to enclose two vectors. iterating through vector list while starting with v1-v2 - default: c()
add.notes	list with additional notes list(c(x, y, str), c(x, y, str)) - default: NA

Value

creates a plot containing all passed vectors.

Examples

```
vectors.draw(x = c(3,2.1), y = c(5,9.3))
```

---

warp.idx *time-warp series*

Description

creates a vector of size x.len with warping indices. By the results, data series can be warped using the resulting indices. Indices will be calculated with warping ahead in time following a sweep like sine function. Data points will be deleted for deletions and interpolated for insertions. In case of constant warping (i.e. phase shift) the series length is reduced by 'max.warp'. The sweep increases frequency from omega=1 to omega=3. The sweep keeps a safe tail of n=warp.max unchanged to prevent NA for high frequencies.

Usage

```
warp.idx(x.len, warp.max = 10, warp.fnc = "sweep", n.period = 3)
```

Arguments

x.len	length of warping: i.e. the size of the series to be warped
warp.max	int: maximum warping width
warp.fnc	keyword: warping function {'sweep', 'const'}
n.period	int: number of full periods for sweep function - default: 3

Value

w.stp: the warping steps for every single element of the warping index

w.idx: the index to access the warped element of the original series

Examples

```
warp.idx(x.len = 150)
```

Index

: FARM.eval.R, 2
: utilies.R, 3

align.eval.dewarp.bench, 4
align.eval.error.stat, 5
alot, 5
amss, 6
amss.wpath, 7

calc.global.rel.meas, 7

dewarp.approaches, 8
dewarp.idx.univar, 8
dist.h, 9
dist.matrix.wpath, 10

edr, 10
edr.dist, 11

feat.ext, 12
ffalign.exp, 12
ffalign.exp.calc, 13
fit.list.lengths, 14

get.global.rel.meas, 14

harm.res, 15

load.apply.save *(*: utilies.R*)*, 3
load.data *(*: FARM.eval.R*)*, 2

match.slice.df, 15
msm, 16
msm.cost, 16
msm.dist, 17
multi.warp.series, 17

n.vect.phi, 18

rm.min, 19
robust.minmax, 19

scale_quant_range, 20

test.dewarp.idx, 21
text_extr, 21

vect.phi, 22
vectors.draw, 22

warp.idx, 23