

Package ‘adept’

June 18, 2019

Type Package

Title Adaptive Empirical Pattern Transformation

Version 1.1.2

Description Designed for optimal use in performing fast, accurate walking strides segmentation from high-density data collected from a wearable accelerometer worn during continuous walking activity.

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

URL <https://github.com/martakarass/adept>

BugReports <https://github.com/martakarass/adept/issues>

Depends R (>= 2.10)

Suggests knitr, rmarkdown, testthat, ggplot2, lubridate, reshape2, gridExtra, spelling, cluster, adeptdata

VignetteBuilder knitr

Imports tibble, dplyr, future, magrittr, dvmisc

Language en-US

NeedsCompilation no

Author Marta Karas [aut, cre] (<<https://orcid.org/0000-0001-5889-3970>>),
Jacek Urbanek [aut] (<<https://orcid.org/0000-0002-1890-8899>>),
Ciprian Crainiceanu [aut],
John Muschelli [ctb] (<<https://orcid.org/0000-0001-6469-1750>>),
Adi Gherman [ctb]

Maintainer Marta Karas <marta.karass@gmail.com>

Repository CRAN

Date/Publication 2019-06-18 06:50:03 UTC

R topics documented:

scaleTemplate	2
segmentPattern	3
similarityMatrix	7
windowSmooth	8

Index	10
--------------	-----------

scaleTemplate	<i>Templates Scaling</i>
---------------	--------------------------

Description

Compute a list of scaled templates via linear interpolation.

Usage

```
scaleTemplate(template, template.vl)
```

Arguments

template	A list of numeric vectors. Each vector represents a distinct template.
template.vl	A numeric vector. A grid of vector lengths that each element of template is to be linearly interpolated into.

Value

A list of lists of numeric vectors. Each element of the returned list is a list of templates scaled according to a particular vector length. The number of elements in the returned list equals the length of template.vl.

Examples

```
## Construct a list of two templates
template <- list(sin(seq(0, 2 * pi, length.out = 100)),
                 cos(seq(0, 2 * pi, length.out = 100)))
## A grid of vector lengths to which each of templates is scaled into
template.vl <- c(50, 100, 200)
## Compute list of rescaled templates
out <- scaleTemplate(template, template.vl)

## Plot 1st template after rescaling to three values of vector length
par(mfrow = c(2, 1), cex = 0.7)
plot(out[[3]][[1]], type = "l",
      main = "Pattern: sin([0, 2 * pi]) rescaled according to different scales",
      ylab = "Pattern", xlab = "Index")
lines(out[[2]][[1]], col = "red")
lines(out[[1]][[1]], col = "blue")
```

```
## Plot 2nd template after rescaling to three values of vector length
plot(out[[3]][[2]], type = "l",
      main = "Pattern: cos([0, 2 * pi]) rescaled according to different scales",
      ylab = "Pattern", xlab = "Index")
lines(out[[2]][[2]], col = "red")
lines(out[[1]][[2]], col = "blue")
```

segmentPattern

Pattern Segmentation From a Time-series via ADEPT

Description

Segment pattern from a time-series x via Adaptive Empirical Pattern Transformation (ADEPT).

Usage

```
segmentPattern(x, x.fs, template, pattern.dur.seq,
               similarity.measure = "cov", similarity.measure.thresh = 0,
               x.adept.ma.W = NULL, finetune = NULL, finetune.maxima.ma.W = NULL,
               finetune.maxima.nbh.W = NULL, run.parallel = FALSE,
               run.parallel.cores = NULL, x.cut = TRUE, x.cut.v1 = 6000,
               compute.template.idx = FALSE)
```

Arguments

<code>x</code>	A numeric vector. A time-series to segment pattern from.
<code>x.fs</code>	A numeric scalar. Frequency at which a time-series x is collected, expressed in a number of observations per second.
<code>template</code>	A list of numeric vectors, or a numeric vector. Each vector represents a distinct pattern template used in segmentation.
<code>pattern.dur.seq</code>	A numeric vector. A grid of potential pattern durations used in segmentation. Expressed in seconds. See: Details.
<code>similarity.measure</code>	A character scalar. Statistic used to compute similarity between a time-series x and pattern templates. Currently supported values: <ul style="list-style-type: none"> "cov" - covariance, "cor" - correlation, Default is "cov".
<code>similarity.measure.thresh</code>	A numeric scalar. Threshold of minimal similarity value between a time-series x and a template below which the algorithm does not identify a pattern occurrence from x . Default is 0.

<code>x.adept.ma.W</code>	A numeric scalar. A length of a window used in moving average smoothing of a time-series <code>x</code> for similarity matrix computation. Expressed in seconds. Default is NULL (no smoothing applied).
<code>finetune</code>	A character scalar. A type of fine-tuning procedure employed in segmentation. Defaults to NULL (no fine-tuning procedure employed). Currently supported values: <ul style="list-style-type: none"> • "maxima" - tunes preliminarily identified beginning and end of a pattern so as they correspond to local maxima of time-series <code>x</code> (or smoothed version of <code>x</code>) found within neighbourhoods of preliminary locations.
<code>finetune.maxima.ma.W</code>	A numeric scalar. A length of a window used in moving average smoothing of a time-series <code>x</code> in "maxima" fine-tuning procedure. Expressed in seconds. Default is NULL (no smoothing applied).
<code>finetune.maxima.nbh.W</code>	A numeric scalar. A length of the two neighborhoods centered at preliminarily identified beginning and end of a pattern within which we search for local maxima of <code>x</code> (or smoothed version of <code>x</code>) in "maxima" fine-tuning procedure. Expressed in seconds. Default is NULL. Note: if the length provided corresponds to an even number of <code>x</code> vector indices, it will be rounded down so as the corresponding number of vector indices is its closest odd number.
<code>run.parallel</code>	A logical scalar. Whether or not to use parallel execution in the algorithm. The future package is used to execute code asynchronously. Default is FALSE.
<code>run.parallel.cores</code>	An integer scalar. The number of cores to use for parallel execution. Default is NULL. If not specified, the number of cores is set to a number of cores available minus 1.
<code>x.cut</code>	A logical scalar. Whether or not to use time optimization procedure in which a time-series <code>x</code> is cut into parts and segmentation is performed for each part of <code>x</code> separately. Recommended for a time-series <code>x</code> of vector length above 30,000. Default is TRUE.
<code>x.cut.vl</code>	An integer scalar. Defines a vector length of parts that <code>x</code> vector is cut into during the execution time optimization procedure. Default is 6000 (recommended).
<code>compute.template.idx</code>	A logical scalar. Whether or not to compute and return information about which of the provided pattern templates yielded a similarity matrix value that corresponds to an identified pattern occurrence. Setting to TRUE may increase computation time. Default is FALSE.

Details

Function implements Adaptive Empirical Pattern Transformation (ADEPT) method for pattern segmentation from a time-series `x`. ADEPT is optimized to perform fast, accurate walking strides segmentation from high-density data collected with a wearable accelerometer during walking.

ADEPT identifies patterns in a time-series `x` via maximization of chosen similarity statistic (correlation, covariance, etc.) between a time-series `x` and a pattern template(s). It accounts for variability in both (1) pattern duration and (2) pattern shape.

Value

A data.frame with segmentation results. Each row describes one identified pattern occurrence:

- tau_i - index of x where pattern starts,
- T_i - pattern duration, expressed in x vector length,
- sim_i - similarity between a pattern and x; note: if "maxima" fine-tune and/or x smoothing is employed, the similarity value between the final segmented pattern and a template may differ from the value in this table,
- template_i - if compute.template.idx equals TRUE: index of a template best matched to x; if compute.template.idx equals FALSE: NA.

Examples

```
## Example 1: Simulate a time-series `x`. Assume that
## - `x` is collected at a frequency of 100 Hz,
## - there is one shape of pattern present within `x`,
## - each pattern lasts 1 second,
## - there is no noise in the collected data.
true.pattern <- cos(seq(0, 2 * pi, length.out = 100))
x <- c(true.pattern[1], replicate(10, true.pattern[-1]))
## Segment pattern from x.
out <- segmentPattern(
  x = x,
  x.fs = 100,
  template = true.pattern,
  pattern.dur.seq = c(0.9, 0.95, 1.03, 1.1),
  similarity.measure = "cor",
  compute.template.idx = TRUE)
out
## Segment pattern from x. Now assume a grid of potential pattern duratios
## contains true pattern duration
out <- segmentPattern(
  x = x,
  x.fs = 100,
  template = true.pattern,
  pattern.dur.seq = c(0.9, 0.95, 1, 1.03, 1.1),
  similarity.measure = "cor",
  compute.template.idx = TRUE)
out

## Example 2: Simulate a time-series `x`. Assume that
## - `x` is collected at a frequency of 100 Hz,
## - there are two shapes of pattern present within `x`,
## - patterns have various duration,
## - there is no noise in the collected data.
true.pattern.1 <- cos(seq(0, 2 * pi, length.out = 200))
true.pattern.2 <- true.pattern.1
true.pattern.2[70:130] <- 2 * true.pattern.2[min(70:130)] + abs(true.pattern.2[70:130])
x <- numeric()
for (v1 in seq(70, 130, by = 10)){
  true.pattern.1.s <- approx(
```

```

    seq(0, 1, length.out = 200),
    true.pattern.1, xout = seq(0, 1, length.out = vl))$y
true.pattern.2.s <- approx(
  seq(0, 1, length.out = 200),
  true.pattern.2, xout = seq(0, 1, length.out = vl))$y
x <- c(x, true.pattern.1.s[-1], true.pattern.2.s[-1])
if (vl == 70) x <- c(true.pattern.1.s[1], x)
}
## Segment pattern from x. Use a `template` object consisting of both
## true patterns used in `x` simulation.
out <- segmentPattern(
  x = x,
  x.fs = 100,
  template = list(true.pattern.1, true.pattern.2),
  pattern.dur.seq = 60:130 * 0.01,
  similarity.measure = "cor",
  compute.template.idx = TRUE)
out

## Example 3: Simulate a time-series `x`. Assume that
## - `x` is collected at a frequency of 100 Hz,
## - there are two shapes of a pattern present within `x`,
## - patterns have various duration,
## - there is noise in the collected data.
set.seed(1)
x <- x + rnorm(length(x), sd = 0.5)
## Segment pattern from x.
out <- segmentPattern(
  x = x,
  x.fs = 100,
  template = list(true.pattern.1, true.pattern.2),
  pattern.dur.seq = 60:130 * 0.01,
  similarity.measure = "cor",
  compute.template.idx = TRUE)
out
## Segment pattern from x. Use `x.adept.ma.W` to define a length of a smoothing
## window to smooth `x` for similarity matrix computation.
out <- segmentPattern(
  x = x,
  x.fs = 100,
  template = list(true.pattern.1, true.pattern.2),
  pattern.dur.seq = 60:130 * 0.01,
  similarity.measure = "cor",
  x.adept.ma.W = 0.1,
  compute.template.idx = TRUE)
out
## Segment pattern from x. Use `x.adept.ma.W` to define a length of a smoothing
## window to smooth `x` for similarity matrix computation. Employ a fine-tuning
## procedure for stride identification.
out <- segmentPattern(
  x = x,
  x.fs = 100,
  template = list(true.pattern.1, true.pattern.2),

```

```

pattern.dur.seq = 60:130 * 0.01,
similarity.measure = "cor",
x.adept.ma.W = 0.1,
finetune = "maxima",
finetune.maxima.nbh.W = 0.3,
compute.template.idx = TRUE)
out
## Segment pattern from x. Employ a fine-tuning procedure for stride
## identification. Smooth `x` for both similarity matrix computation
## (set `x.adept.ma.W = 0.1`) and for fine-tune peak detection procedure
## (set `finetune.maxima.nbh.W = 0.3`).
out <- segmentPattern(
  x = x,
  x.fs = 100,
  template = list(true.pattern.1, true.pattern.2),
  pattern.dur.seq = 60:130 * 0.01,
  similarity.measure = "cor",
  x.adept.ma.W = 0.1,
  finetune = "maxima",
  finetune.maxima.nbh.W = 0.3,
  compute.template.idx = TRUE)
out

```

similarityMatrix

ADEPT Similarity Matrix Computation

Description

Compute ADEPT similarity matrix between a time-series `x` and a collection of scaled templates.

Usage

```
similarityMatrix(x, template.scaled, similarity.measure)
```

Arguments

`x` A numeric vector. A time-series `x`.

`template.scaled`

A list of lists of numeric vectors, as returned by `scaleTemplate`. Each element of `template.scaled` is a list of templates interpolated to a particular vector length. Number of elements in the `template.scaled` corresponds to the number of unique template length values used in segmentation.

`similarity.measure`

A character scalar. Statistic used in similarity matrix computation; one of the following:

- "cov" - for covariance,
- "cor" - for correlation.

Value

A numeric matrix. Contains values of similarity between a time-series x and scaled templates.

- Number of rows equals `template.scaled` length, number of columns equals x length.
- A particular matrix row consists of similarity statistic between x and a template rescaled to a particular vector length. Precisely, each row's element is a maximum out of similarity values computed for each distinct template used in segmentation.

See Also

`scaleTemplate {adept}`

Examples

```
## Simulate data
par(mfrow = c(1,1))
x0 <- sin(seq(0, 2 * pi * 100, length.out = 10000))
x <- x0 + rnorm(1000, sd = 0.1)
template <- list(x0[1:500])
template.vl <- seq(300, 700, by = 50)

## Rescale pattern
template.scaled <- scaleTemplate(template, template.vl)

## Compute ADEPT similarity matrix
out <- similarityMatrix(x, template.scaled, "cov")

## Visualize
par(mfrow = c(1,1))
image(t(out),
      main = "ADEPT similarity matrix\nfor time-series x and scaled versions of pattern templates",
      xlab = "Time index",
      ylab = "Pattern vector length",
      xaxt = "n", yaxt = "n")
xaxis <- c(1, seq(1000, length(x0), by = 1000))
yaxis <- template.vl
axis(1, at = xaxis/max(xaxis), labels = xaxis)
axis(2, at = (yaxis - min(yaxis))/(max(yaxis) - min(yaxis)), labels = yaxis)
```

Description

Compute moving window average of a time-series x .

Usage

```
windowSmooth(x, W, x.fs = 1)
```


Arguments

<code>x</code>	A numeric vector. A time-series for which a moving window average is computed.
<code>W</code>	A numeric scalar. A length of a moving window, expressed in time (seconds).
<code>x.fs</code>	Frequency of a time-series <code>x</code> , expressed in a number of observations per second. Defaults to 1.

Details

Time-series frequency `x.fs` and a length of a moving window (expressed in time) `W` together determine `W.v1 = round(W * x.fs)`, a length of a moving window expressed in a length of `x` vector object. Note: `W.v1` must be equal or greater than 3.

- If `W.v1 < 3` then an error is thrown.
- If `W.v1` is an even number then `(W.v1-1)` value is silently used instead as a length of a moving window expressed in `x` vector length.

Value

A numeric vector. Moving window average of a time-series `x`. Note: head and tail of the output vector where the moving window is undefined are filled with NA.

Examples

```
## Time-series defined as a function f(x) = x
N <- 100
W <- 20
x <- 1:N
x.smoothed <- windowSmooth(x, W)
plot(x, type = "l")
points(x.smoothed, col = "red")

## Time-series defined as a function f(x) = sin(x) + noise
N <- 1000
W <- 100
x <- sin(seq(0, 4 * pi, length.out = N)) + rnorm(N, sd = 0.1)
x.smoothed <- windowSmooth(x, W)
plot(x, type = "l")
points(x.smoothed, col = "red")
```

Index

scaleTemplate, [2](#)
segmentPattern, [3](#)
similarityMatrix, [7](#)

windowSmooth, [8](#)