Package: adept (via r-universe)

September 2, 2024

Type Package

Title Adaptive Empirical Pattern Transformation

Version 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 7.1.1

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, covr

VignetteBuilder knitr

Imports dplyr, magrittr, dvmisc, parallel, pracma

Language en-US

Repository https://martakarass.r-universe.dev

RemoteUrl https://github.com/martakarass/adept

RemoteRef HEAD

RemoteSha d7165add91e75b4674e8206ee6053a0c3f7824c9

Contents

scaleTemplate 2
segmentPattern 3
segmentWalking 7
similarityMatrix 10
windowSmooth 11

Index **13**

`scaleTemplate` *Templates Scaling*

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 = 1L,
  x.cut = TRUE,
  x.cut.vl = 6000,
  compute.template.idx = FALSE
)
```

Arguments

x	A numeric vector. A time-series to segment pattern from.
x.fs	A numeric scalar. Frequency at which a time-series x is collected, expressed in a number of observations per second.
template	A list of numeric vectors, or a numeric vector. Each vector represents a distinct pattern template used in segmentation.
pattern.dur.seq	A numeric vector. A grid of potential pattern durations used in segmentation. Expressed in seconds. See: Details.
similarity.measure	A character scalar. Statistic used to compute similarity between a time-series x and pattern templates. Currently supported values: • "cov" - covariance, • "cor" - correlation, Default is "cov".

`similarity.measure.thresh`
: 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`.

`x.adept.ma.W`
: A numeric scalar. A length of a window used in moving average smoothing of a time-series `x` for similarity matrix computation. Expressed in seconds. Default is `NULL` (no smoothing applied).

`finetune`
: A character scalar. A type of fine-tuning procedure employed in segmentation. Defaults to `NULL` (no fine-tuning procedure employed). Currently supported values:

  - `"maxima"` - tunes preliminarily identified beginning and end of a pattern so as they correspond to local maxima of time-series `x` (or smoothed version of `x`) found within neighbourhoods of preliminary locations.

`finetune.maxima.ma.W`
: A numeric scalar. A length of a window used in moving average smoothing of a time-series `x` in `"maxima"` fine-tuning procedure. Expressed in seconds. Default is `NULL` (no smoothing applied).

`finetune.maxima.nbh.W`
: 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 `x` (or smoothed version of `x`) in `"maxima"` fine-tuning procedure. Expressed in seconds. Default is `NULL`. Note: if the length provided corresponds to an even number of `x` vector indices, it will be rounded down so as the corresponding number of vector indices is its closest odd number.

`run.parallel`
: A logical scalar. Whether or not to use parallel execution in the algorithm with `parallel` package. Default is `FALSE`. DOES NOT WORK ON WINDOWS.

`run.parallel.cores`
: An integer scalar. The number of cores to use for parallel execution. Defaults to 1L (no parallel). DOES NOT WORK ON WINDOWS.

`x.cut`
: A logical scalar. Whether or not to use time optimization procedure in which a time-series `x` is cut into parts and segmentation is performed for each part of `x` separately. Recommended for a time-series `x` of vector length above 30,000. Default is `TRUE`.

`x.cut.vl`
: An integer scalar. Defines a vector length of parts that `x` vector is cut into during the execution time optimization procedure. Default is `6000` (recommended).

`compute.template.idx`
: 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.

References

Karas, M., Straczkiewicz, M., Fadel, W., Harezlak, J., Crainiceanu, C.M., Urbanek, J.K. (2019). Adaptive empirical pattern transformation (ADEPT) with application to walking stride segmentation. Biostatistics. https://doi.org/10.1093/biostatistics/kxz033

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 (vl 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
```

`segmentWalking` *Walking Stride Pattern Segmentation from Raw Accelerometry Data via ADEPT*

Description

Segment walking stride pattern from a raw accelerometry data time-series `(x,y,z)` via Adaptive Empirical Pattern Transformation (ADEPT). Default algorithm parameters are optimized for a wrist-worn sensor and were evaluated with data collected in the free-living environment.

Usage

```
segmentWalking(
  xyz,
  xyz.fs,
```

```
  template,
  sim_MIN = 0.85,
  dur_MIN = 0.8,
  dur_MAX = 1.4,
  ptp_r_MIN = 0.2,
  ptp_r_MAX = 2,
  vmc_r_MIN = 0.05,
  vmc_r_MAX = 0.5,
  mean_abs_diff_med_p_MAX = 0.5,
  mean_abs_diff_med_t_MAX = 0.2,
  mean_abs_diff_dur_MAX = 0.2,
  compute.template.idx = FALSE,
  run.parallel = FALSE,
  run.parallel.cores = 1
)
```

Arguments

`xyz`	A numeric matrix (or data frame) of n × 3 dimension. Three-dimensional raw accelerometry data time-series; acceleration measurements `(x,y,z)` collected along three orthogonal axes by the sensor's accelerometer.
`xyz.fs`	A numeric scalar. Frequency at which a time-series `(x,y,z)` is collected, expressed in a number of observations per second.
`template`	A list of numeric vectors, or a numeric vector. Distinct pattern template(s) of walking stride.
`sim_MIN`	numeric scalar. Minimum value of correlation between pattern template(s) and (r_t)_t vector magnitude of accelerometry data. Default used is 0.85.
`dur_MIN`	A numeric scalar. Minimum value of a stride duration allowed to be identified. Expressed in seconds. Default used is 0.8.
`dur_MAX`	A numeric scalar. Maximum value of a stride duration allowed to be identified. Expressed in seconds. Default used is 1.4.
`ptp_r_MIN`	A numeric scalar. Minimum value of "peak to peak" difference in (r_t)_t vector magnitude data of a stride. Default used is 0.2.
`ptp_r_MAX`	A numeric scalar. Maximum value of "peak to peak" difference in (r_t)_t vector magnitude data of a stride. Default used is 2.0
`vmc_r_MIN`	A numeric scalar. Minimum value of VMC in (r_t)_t vector magnitude data of a stride. Default used is 0.05.
`vmc_r_MAX`	A numeric scalar. Maximum value of VMC in (r_t)_t vector magnitude data of a stride. Default used is 0.5.
`mean_abs_diff_med_p_MAX`	A numeric scalar. Maximum value of MAD* of Azimuth (az_)_t median for 3 subsequent valid strides. Here, MAD* stands for mean of 2 absolute differences between 3 subsequent values. Default used is 0.5.
`mean_abs_diff_med_t_MAX`	A numeric scalar. Maximum value of MAD* of Elevation (el_)_t median for 3 subsequent valid strides. Here, MAD* stands for mean of 2 absolute differences between 3 subsequent values. Default used is 0.2.

mean_abs_diff_dur_MAX
: A numeric scalar. Maximum value of MAD* of duration time for 3 subsequent valid strides. Here, MAD* stands for mean of 2 absolute differences between 3 subsequent values. Default used is 0.2.

compute.template.idx
: 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.

run.parallel
: A logical scalar. Whether or not to use parallel execution in the algorithm with parallel package. Default is FALSE. DOES NOT WORK ON WINDOWS.

run.parallel.cores
: An integer scalar. The number of cores to use for parallel execution. Defaults to 1L (no parallel). DOES NOT WORK ON WINDOWS.

Value

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

- tau_i - row index of xyz where pattern starts,
- T_i - pattern duration, expressed in xyz vector length,
- sim_i - similarity between a pattern and best-fit template; see segmentPattern for details,
- template_i - if compute.template.idx equals TRUE: index of a template best matched to x; if compute.template.idx equals FALSE: NA,
- is_walking_i – 1 if a pattern is identified as walking stride; 0 otherwise.

Examples

```
library(adeptdata)
library(dplyr)
library(adept)
xyz <-
  adeptdata::acc_walking_IU %>%
  filter(loc_id == "left_wrist", subj_id == "id86237981") %>%
  arrange(time_s) %>%
  select(v1 = x, v2 = y, v3 = z) %>%
  as.matrix()
# define raw accelerometry data sample frequency
xyz.fs <- 100

# define template list based on predefined templates
template_mat <- adeptdata::stride_template$left_wrist[[3]]
template <- list(
  template_mat[1, ],
  template_mat[2, ],
  template_mat[3, ]
)

# run walking segmentation
```

```
# (parallel supported, except for Windows; see run.parallel, run.parallel.cores args)
segmentWalking(xyz, xyz.fs, template)
```

`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)
```

`windowSmooth` *Fast Computation of Moving Window Average*

Description

Compute moving window average of a time-series `x`.

Usage

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

Arguments

`x`	A numeric vector. A time-series for which a moving window average is computed.
`W`	A numeric scalar. A length of a moving window, expressed in time (seconds).
`x.fs`	Frequency of a time-series `x`, 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.vl = round(W * x.fs)`, a length of a moving window expressed in a length of `x` vector object. Note: `W.vl` must be equal or greater than `3`.

- If `W.vl < 3` then an error is thrown.
- If `W.vl` is an even number then `(W.vl-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
segmentWalking, 7
similarityMatrix, 10

windowSmooth, 11