Package ‘xwf’

February 20, 2020

Version 0.2-3
Date 2020-02-19
Title Extrema-Weighted Feature Extraction
Author Willem van den Boom [aut, cre]
Maintainer Willem van den Boom <willem@wvdboom.nl>
Description Extrema-weighted feature extraction for varying length functional data. Functional data analysis method that performs dimensionality reduction based on predefined features and allows for quantile weighting. Method implemented as presented in van den Boom et al. (2018) <doi:10.1093/bioinformatics/bty120>.
License MIT + file LICENSE
Imports mgcv
RoxygenNote 6.0.1
NeedsCompilation no
Repository CRAN
Date/Publication 2020-02-20 07:00:02 UTC

R topics documented:

  default_psi ......................................................... 2
  xwf ................................................................. 2
  xwfGAM ............................................................ 3
  xwfGridsearch ..................................................... 4
  XWFpValues ....................................................... 6

Index 9
default_psi  Default psi list

Description
List with the same local feature functions psi as in the original paper

Usage
default_psi()

Value
List with 4 different local features psi

Examples
default_psi()

xwf  Compute XWFs

Description
Compute extrema-weighted features based on functions, predefined local features, and weighting functions

Usage
xwf(xx, t, n.i, psi, w = function(t, i) ifelse(left, min(1, (1 - F(xx[[i]](t)))/(1 - b)), min(1, F(xx[[i]](t))/b)), b = 0.5, F = NULL, t.min = NULL, t.max = NULL, t.range = NULL, rel.shift = 0.001, left = TRUE)

Arguments
xx  List of function for which to compute the XWFs
t  Matrix containing the times at which the functions xx were measured: Element (i,j) contains the time of the j-th measurement of the i-th function.
n.i  Vector containing the number of measurements for each function. The first n.i[i] elements of the i-th row of t should not be NA.
psi  Predefined local feature which is a function of a function (first argument) and a measurement time (second argument)
w  Weighting function. The default is the one used in the original paper.
**Description**

Evaluate the generalized additive model for a set of computed extrema-weighted features

**Usage**

```r
xwfGAM(wL, wR, y, z = NULL)
```
Arguments

L Matrix with left extrema-weighted features
wR Matrix with right extrema-weighted features
y Binary vector with outcomes
z Optional matrix z with extra, linear predictors

Examples

```
xwf::xwfGAM(wL = rep(1:45, 10), wR = rep(1:90, 5), y = c(rep(0:1, 225)))
```

Description

Adaptive grid search to optimize the weighting functions in the extrema-weighted features.

Usage

```
xwfGridsearch(y, xx, t, n.i, psi.list = default_psi(), F = NULL, z = NULL, iter = 3, w = function(t, i, b, left) ifelse(left, min(1, (1 - F(xx[[i]](t)))/(1 - b)), min(1, F(xx[[i]](t))/b)), rel.shift = 0.001, progressbar = TRUE)
```

Arguments

y Vector with binary outcomes data
xx List of functions for which to compute the XWFs
t Matrix containing the times at which the functions xx were measured: Element (i,j) contains the time of the j-th measurement of the i-th function.
n.i Vector containing the number of measurements for each function. The first n.i[i] elements of the i-th row of t should not be NA.
psi.list List of predefined local features which are functions of a function (first argument) and a measurement time (second argument)
F CDF of the values of the functions xx. Ignored if weighting function w is not the default.
z Optional matrix with covariates to be included as linear predictors in the generalized additive model
iter Number of levels in the adaptive grid search. The resolution in b obtained is $2^{-1-\text{iter}}$.
w Weighting function. The default is the one used in the original paper. See the default for what the roles of its 3 arguments are.
rel.shift  Optional relative reduction of the integration range to avoid instabilities at the end of the integration ranges. Set to 0 if no such correction is desired.

progressbar  Boolean specifying whether a progress bar indicating what level of the adaptive grid has been completed should be displayed.

Value  
List containing the final XWFs (wL and wR), the parameters for the optimal weighting functions (b.left and b.right), and the gmcv::gamObject corresponding to the final optimal generalized additive model fit.

Examples  
# Data simulation similar to Section 3.2 of the paper  
# Sample size  
n <- 100

# Length of trajectories  
n.i <- rep(5, n)  
max.n.i <- max(n.i)

# Times  
t <- matrix(NA_integer_, nrow = n, ncol = max.n.i)  
for(i in 1:n) t[i, 1:n.i[i]] <- 1:n.i[i]

# Sample periods  
phi <- runif(n = n, min = 1, max = 10)

# Sample offsets  
m <- 10*runif(n = n)

# Blood pressure measurements  
x <- t  
for(i in 1:n) x[i, 1:n.i[i]] <- sin(phi[i] * 2*pi/max.n.i * t[i, 1:n.i[i]]) + m[i]

# Matrix with covariates z  
q <- 2 # Number of covariates  
z <- matrix(rnorm(n = n*q), nrow = n, ncol = q)

# Generate outcomes  
temp <- phi*min(m, 7)  
temp <- 40*temp  
prob <- 1/(1+exp( 2*( median(temp)-temp ) ))  
y <- rbinom(n = n, size = 1, prob = prob)

xx <- list()  
for(i in 1:n) xx[[i]] <- approxfun(x = t[i,1:n.i[i]], y = x[i,1:n.i[i]], rule = 2)

# Estimate f  
weights <- matrix(1/n.i, ncol = max.n.i, nrow = n)[is.na(t)]
XWFpValues

p-value computation for XWFs

Description

Randomization method to compute p-values for an optimized extrema-weighted features generalized additive model fit.

Usage

XWFpValues(GAMobject, xx, t, n.i, psi.list = NULL, F, z = NULL,
            w = function(t, i, b, left) ifelse(left, min(1, (1 - F(xx[[i]](t)))/(1 - b)), min(1, F(xx[[i]](t))/b)), n.boot = 100, progressbar = TRUE)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GAMobject</td>
<td>The GAMobject returned by <code>xwfGridsearch</code></td>
</tr>
<tr>
<td>xx</td>
<td>List of function for which to compute the XWFs</td>
</tr>
<tr>
<td>t</td>
<td>Matrix containing the times at which the functions xx were measured: Element (i,j) contains the time of the j-th measurement of the i-th function.</td>
</tr>
<tr>
<td>n.i</td>
<td>Vector containing the number of measurements for each function. The first n.i[i] elements of the i-th row of t should not be NA.</td>
</tr>
<tr>
<td>psi.list</td>
<td>List of predefined local features which are functions of a function (first argument) and a measurement time (second argument)</td>
</tr>
</tbody>
</table>
CDF of the values of the functions xx. Ignored if weighting function w is not the default.

Optional matrix with covariates to be included as linear predictors in the generalized additive model

Weighting function. The default is the one used in the original paper. See the default for what the roles of its 3 arguments are.

Number for randomizations used to obtain the p-values. The resolution of the p-values is 1/n.boot

Boolean specifying whether a progress bar indicating which randomizations have been completed should be displayed.

Named vector with p-values

# Data simulation similar to Section 3.2 of the paper

# Sample size
n <- 100

# Length of trajectories
n.i <- rep(5, n)
max.n.i <- max(n.i)

# Times
t <- matrix(NA_integer_, nrow = n, ncol = max.n.i)
for(i in 1:n) t[i, 1:n.i[i]] <- 1:n.i[i]

# Sample periods
phi <- runif(n = n, min = 1, max = 10)

# Sample offsets
m <- 10*runif(n = n)

# Blood pressure measurements
x <- t
for(i in 1:n) x[i, 1:n.i[i]] <- sin(phi[i] * 2*pi/max.n.i * t[i, 1:n.i[i]]) + m[i]

# Matrix with covariates z
q <- 2 # Number of covariates
z <- matrix(rnorm(n = n*q), nrow = n, ncol = q)

# Generate outcomes
temp <- phi*min(m, 7)
temp <- 40*temp
prob <- 1/(1+exp(2*(median(temp-temp))
for(i in 1:n) y[i] <- rbinom(n = n, size = 1, prob = prob)
xx <- list()
for(i in 1:n) xx[[i]] <- approxfun(x = t[i,1:n.i[i]], y = x[i,1:n.i[i]], rule = 2)

# Estimate f
weights <- matrix(1/n.i, ncol = max.n.i, nrow = n)[!is.na(t)]
f <- density(  
x = t(sapply(X = 1:n, FUN = function(i) c(xx[[i]](t[i,1:n.i[i]]), rep(NA, max.n.i-n.i[i])))),
weights = weights/sum(weights),
na.rm = T)

# Define CDF of f, F
CDF <- c(0)
for(i in 2:length(f$x)) CDF[i] <- CDF[i-1]+(f$x[i]-f$x[i-1])*(f$y[i]+f$y[i-1])/2
F <- approxfun(x = f$x, y = CDF/max(CDF), yleft = 0, yright = 1)

psi <- list(
  function(x, t) abs(x(t)-x(t-1))
)

XWFresult <- xwfGridsearch(y = y, xx = xx, t = t, n.i = n.i, psi.list = psi, F = F, z = z)

XWFpValues(
  GAMobject = XWFresult$GAMobject,
  xx = xx,
  t = t,
  n.i = n.i,
  psi.list = psi,
  F = F,
  z = z,
  n.boot = 3)

XWFpValues

Index

default_psi, 2

xwf, 2
xwfGAM, 3
xwfGridsearch, 4, 6
XWFpValues, 6