

Main parameters in using AFFSEN function

Use the AFFSEN function to find significant predictors and estimate a smooth version of them in a high dimensional case. We have option to control sparsity and smoothness separately with using two penalty parameters λ_H and λ_K . We aim to estimate a smooth version of β s minimize the following target function.

$$L_{\lambda}(\beta) = \frac{1}{2N} \|\mathbf{Y} - \mathbf{X}\beta\|_{\mathbb{H}}^2 + \frac{\lambda_K}{2} \sum_{i=1}^I \|L(\beta_i)\|_{\mathbb{K}}^2 + \lambda_2 \sum_{i=1}^I \tilde{w}_i \|\beta_i\|_{\mathbb{H}}$$

The following AFFSEN function helps us to estimate the β and find the significant predictors.

```
AFFSEN<- function(X,Y, T_domain = seq(0, 1, length = 50),
  # X (a N*I numerical matrix)    N = #observations    I = #all_predictors
  # Y = Y_ful=XB+E (a N*m matrix of pointwise evaluation!)    m = #time_points
  type_kernel="exponential",param_kernel=8,
  thres=0.02, number_non_zeros=20,
  ratio_lambda_H=0.01, number_lambda_H=100, num_lambda_H_NONad=50,
  lambda_H, lambda_K,
  early_CV=0, early_CV_thres=0.001, max_ite_nadp=10,
  max_ite_adp=30, max_ite_final=50, target_inc=1,
  proportion_training_set=0.75, verbose=FALSE, fold_ad=10)
```

In the following table you can find the functionality of each parameters:

	functionality	Details
X	Numerical design matrix	It should be a N*I matrix N= #observations ; I = #predictors
Y	Matrix of pointwise evaluation for observations on T_domain	It should be a N*m matrix.
T_domain	Time domain for evaluation of Y and generating kernel	Default : T_domain = seq(0,1,m=50)
type_kernel	Type of kernel	‘exponential’, ‘gaussian’, ‘sobolev’

param_kernel	Kernel parameter	In all types, the time domain is seq(0,1,50)
thres	Stopping criteria: beta increment threshold	$ \beta^T - \beta^{T-1} < thres$
number_non_zeros	Stopping Criteria: Kill switch; number of nonzero predictors	
ratio_lambda_H	$\frac{\lambda_{Hmax}}{\lambda_{Hmin}}$	
number_lambda_H	Generate number of log-equally spaced λ_H in $[\lambda_{Hmin}, \lambda_{Hmax}]$	
num_lambda_H_NONad	Number of λ_H in non-adaptive step	
lambda_H	You have option to insert a vector of λ_H	If you want to make the log-equally spaced by above parameters, just set lambda_H=numeric()
lambda_K	Vector of λ_K	
early_CV	0 or 1 : applying the “early_CV_thres” stopping criteria or not.	
early_CV_thres	Stopping Criteria: Breaking point in CV plot.	$\frac{ CV(h-1, k) - CV(h, k) }{CV(h-1, k)} < early_CV_thres$
max_ite_nadp	Stopping Criteria: Maximum iteration of coordinate descent alg. in non-adaptive step	
max_ite_adp	Stopping Criteria: Maximum iteration of coordinate descent alg. in adaptive step	
max_ite_final	Stopping Criteria: Maximum iteration of coordinate descent alg. for the optimum λ_H and λ_K	
target_inc	Stopping Criteria: 0 or 1 : if target function is increased, stop	
proportion_training_set	Proportion of training set for estimation in non-adaptive step.	
fold_ad	Number of fold for using CV in adaptive steps to find optimum λ_H and λ_K and then estimation.	