Package 'SpiceFP'

September 15, 2021

Type Package

Title Sparse Method to Identify Joint Effects of Functional Predictors

Version 0.1.0

Author Girault Gnanguenon Guesse [aut, cre],
Patrice Loisel [aut],
Benedicte Fontez [aut],
Nadine Hilgert [aut],
Thierry Simonneau [ctr],
Isabelle Sanchez [ctr]

Maintainer Girault Gnanguenon Guesse <girault.gnanguenon@gmail.com>

Description A set of functions allowing to implement the 'SpiceFP' approach which is iterative. It involves transformation of functional predictors into several candidate explanatory matrices (based on contingency tables), to which relative edge matrices with contiguity constraints are associated. Generalized Fused Lasso regression are performed in order to identify the best candidate matrix, the best class intervals and related coefficients at each iteration. The approach is stopped when the maximal number of iterations is reached or when retained coefficients are zeros. Supplementary functions allow to get coefficients of any candidate matrix or mean of coefficients of many candidates. <https://hal.archives-ouvertes.fr/hal-03298977>.

License GPL-3

Encoding UTF-8

LazyData true

Depends R (>= 3.6.0)

Imports doParallel, foreach, stringr, tidyr, Matrix, genlasso, purrr, gplots

Suggests rmarkdown, knitr, fields

RoxygenNote 7.1.1

NeedsCompilation no

Repository CRAN

Date/Publication 2021-09-15 08:40:18 UTC

R topics documented:

`SpiceFP-package` *A Sparse and Structured Procedure to Identify Combined Effects of Functional Predictors*

Description

A set of functions allowing to implement the 'SpiceFP' approach which is iterative. It involves transformation of functional predictors into several candidate explanatory matrices (based on contingency tables), to which relative edge matrices with contiguity constraints are associated. Generalized Fused Lasso regression are performed in order to identify the best candidate matrix, the best class intervals and related coefficients at each iteration. The approach is stopped when the maximal number of iterations is reached or when retained coefficients are zeros. Supplementary functions allow to get coefficients of any candidate matrix or mean of coefficients of many candidates.

Details

The main function of the package is the `spicefp` function. It directly performs the three main steps of the SpiceFP approach, by using intermediate functions of the package.
1) At he first step, contingency tables are constructed by defining joint modalities using class intervals or bins. Several candidate partitions are then defined. For each statistical individual i and each candidate partition (denoted u here), the 2 (resp. 3) functional predictors are transformed into frequency bi(resp. tri)-variate histograms (or contingency tables), stored as row vectors. The combination of these row vectors for all individuals enables the construction of a candidate explanatory matrix indexed by u (denoted here X^u). The function `candidates` is designed to build these candidate matrices.
2) At the second step, for each candidate explanatory matrix, an edge matrix is defined to represent the contiguity constraints between modalities of the contingency table.
3) Finally at the last step, the best class intervals and related regression coefficients are defined by:

i) performing a Generalized Fused Lasso using each candidate explanatory matrix. The SpiceFP model is the following

$$y_i = X_i^u \beta^u + \varepsilon_i,$$

where β^u is the coefficient to be estimated on the 2D (resp. 3D) intervals. The estimator of β is obtained as follows:

$$\hat{\beta}^{u,\gamma}(\lambda) = argmin \frac{1}{2} \|y - X^u \beta\|_2^2 + \lambda \|D^{u,\gamma} \beta\|_1,$$

where λ is a penalty parameter that controls the smoothness of the coefficients, and γ is the ratio between the regularization parameters of parsimony and fusion. ii) choosing the best candidate matrix and selecting its variables using an information criterion and checking the shutdown conditions to stop the approach. Indeed, SpiceFP may be used in an iterative way. It therefore allows to identify up to K best candidate matrices and related coefficients.

Author(s)

Maintainer: Girault Gnanguenon Guesse <girault.gnanguenon@gmail.com>

Authors:

- Patrice Loisel <patrice.loisel@inrae.fr>
- Benedicte Fontez <benedicte.fontez@supagro.fr>
- Nadine Hilgert <nadine.hilgert@inrae.fr>

Other contributors:

- Thierry Simonneau <thierry.simonneau@inrae.fr> [contractor]
- Isabelle Sanchez <isabelle.sanchez@inrae.fr> [contractor]

`candidates` *candidates*

Description

The "candidates" function essentially provides the candidate matrices and their characteristics. These candidate matrices can be constructed from 2 or 3 functional predictors.

Usage

```
candidates(
  fp1,
  fp2,
  fp3 = NULL,
  fun1,
  fun2,
  fun3 = NULL,
  parlists,
  ncores = parallel::detectCores() - 1,
  xcentering = TRUE,
  xscaling = FALSE
)
```

Arguments

fp1	numerical matrix with in columns observations of one statistical individual to partition. Each column corresponds to the functional predictor observation for one statistical individual. The order of statistical individuals is the same as in fp2. It is assumed that no data are missing and that all functional predictors are observed on an equidistant (time) scale.
fp2	numerical matrix with the same number of columns and rows as fp1. Columns are also observations. The order of statistical individuals is the same as in fp1.
fp3	NULL by default. numerical matrix with the same number of columns and rows as fp1 and fp2. The order of statistical individuals is the same as in fp1 and fp2.
fun1	a function object with 2 arguments. First argument is fp1 and the second is a list of parameters that will help to partition fp1, such as the number of class intervals, etc. For example, the list of parameters for using the logbreaks function is equivalent to list(alpha, J). All arguments to be varied for the creation of different candidate matrices must be stored in the parameter list. The other arguments must be set by default.
fun2	a function object with 2 arguments. First argument is fp2 and the second is a list of parameters.
fun3	NULL by default. Same as fun1 and fun2, a function with 2 arguments fp3 and a list of parameters.
parlists	list of 2 elements when fp3 and fun3 are equal to NULL or of 3 elements when fp3 and fun3 are provided. All elements of parlists are lists that have the same length. Each list contains all the lists of parameters required to create different candidates. The first element of parlists concerns the list of parameters required for fun1, the second element is relative to fun2 and the third to fun3. See Example 2 below.
ncores	numbers of cores that will be used for parallel computation. By default, it is equal to detectCores()-1.
xcentering	TRUE by default. Defined whether or not the variables in the new candidate matrices should be centered.
xscaling	FALSE by default. Defined whether or not the variables in the candidate matrices should be scaled.

Details

The function begins by partitioning each of the functional predictors using the function and associated parameter lists. Once the class intervals are obtained for each predictor, a contingency table is created for each statistical individual. This table counts the components of the observation variable (time for time series). The contingency table is then transformed into a row vector that corresponds to a row of the candidate matrix created. The number of candidate matrices is equal to the length of each element contained in parlists. For a fixed index, the functional predictors (fp1, fp2, fp3), the functions (fun1, fun2, fun3) and the lists of parameters associated to the index in each element of parlists allow to create a single candidate matrix. In addition to constructing the candidate matrices, the function associates with each matrix a vector containing the index and the numbers of class intervals used per predictor.

Value

The function returns a list with:

spicefp.dimension the dimension of the approach. Equal to 2 if fp3=NULL and 3 if not

candidates a list that has the same length as the elements of parlists. Each element of this list contains a candidate matrix and a vector with index and the numbers of class intervals used per predictor

fp1, fp2, fp3, fun1, fun2, fun3, parlists, xcentering, xscaling same as inputs

Examples

```
##linbreaks: a function allowing to obtain equidistant breaks
linbreaks<-function(x,n){
    sort(round(seq(trunc(min(x)),
         ceiling(max(x)+0.001),
         length.out =unlist(n)+1),
         1)
    )
}

p<-expand.grid(c(12,15),c(15,20))
pl<-list(split(p[,1], seq(nrow(p))),
         split(p[,2], seq(nrow(p))))

# Setting ncores=2 for this example check purpose
test<-candidates(fp1=matrix(rnorm(1000,52,15),ncol=10),
                 fp2=matrix(rpois(1000,50),ncol=10),
                 fun1=linbreaks,
                 fun2=linbreaks,
                 parlists=pl,
                 xcentering = FALSE,
                 xscaling = FALSE,
                 ncores=2)
str(test)
names(test)

# Example 2 from the spiceFP data
tpr.nclass=seq(10,16,2)
irdc.nclass=seq(20,24,2)
irdc.alpha=c(0.01,0.02,0.03)
p2<-expand.grid(tpr.nclass, irdc.alpha, irdc.nclass)
parlist.tpr<-split(p2[,1], seq(nrow(p2)))
parlist.irdc<-split(p2[,2:3], seq(nrow(p2)))
parlist.irdc<-lapply(
  parlist.irdc,function(x){
    list(x[[1]],x[[2]])}
)
m.irdc <- as.matrix(Irradiance[,-c(1)])
m.tpr <- as.matrix(Temperature[,-c(1)])
test2<-candidates(fp1=m.irdc,
                  fp2=m.tpr,
```

```
                    fun1=logbreaks,
                    fun2=linbreaks,
                    parlists=list(parlist.irdc,
                                  parlist.tpr),
                    xcentering = TRUE,
                    xscaling = FALSE,
                    ncores=2)
length(test2$candidates)
class(test2$candidates)
#View(test2$candidates[[1]][[1]])
dim(test2$candidates[[1]][[1]])
test2$candidates[[1]][[2]]

# Closing the connections for the example check purpose
closeAllConnections()
```

---

coef_spicefp *coef_spicefp*

Description

This function allows to obtain the coefficients of a model (involving a candidate matrix and 2 regularization parameters). There are two possible options to use this function: 1/ by minimizing an information criterion and selecting a number of model (option by default), or 2/ directly by providing the parameters of the model(s) that the user wishes to reconstruct.

Usage

```
coef_spicefp(
  spicefp.result,
  iter_,
  criterion = "AIC_",
  nmodels = 1,
  model.parameters = NULL,
  dim.finemesh = NULL,
  ncores = parallel::detectCores() - 1,
  write.external.file = TRUE
)
```

Arguments

spicefp.result	List. Outputs of the spicefp function.
iter_	integer. number of the iteration of interest.
criterion	character. One of "AIC_", "BIC_", "Cp_". Can be NULL, "AIC_" by default. If specified, nmodels must also be provided.
nmodels	integer. Equal to 1 by default. Represents the number of best models, according to the information criterion used. Should be NULL if criterion = NULL.

`model.parameters`
: data.frame. NULL by default. One or more rows contained in the file where the model statistics were stored. Be careful to use the file related to the selected iteration. Names used in model.parameters shoud be the same in the file.

`dim.finemesh`
: numeric vector of length 2 or 3. This vector informs about the dimension of the fine-mesh arrays (or matrices).

`ncores`
: numbers of cores that will be used for parallel computation. By default, it is equal to detectCores()-1.

`write.external.file`
: logical. indicates whether the result table related to each iteration has been written as a file (txt) in your working directory. This argument must be equal to the argument with the same name in the spicefp function.

Details

By providing criterion and nmodels, the function returns the coefficients of the nmodels best models chosen by the selected information criterion. When model.parameters is instead provided, it returns the coefficients of the models described on each row of the data.frame.

Value

Returns a list of 2 elements:

Model.parameters data.frame where each row contains statistics related to the models of interest. Same as input if model.parameters is provided.

coef.list List of length nmodels or the number of rows in Model.parameters. Each element of this list contains the model results as provided by the genlasso package, its coefficients without and with NA, a fine-mesh array with the coefficients, and the estimation of $X\beta$. Coefficients with NA are coefficient vector where the coefficient value of never-observed joint modalities is NA.

Examples

```
##linbreaks: a function allowing to obtain equidistant breaks
linbreaks<-function(x,n){
       sort(round(seq(trunc(min(x)),
                ceiling(max(x)+0.001),
                length.out =unlist(n)+1),
            1)
           )
}
# In this example, we will evaluate 2 candidates with 14 temperature
# classes and 15 irradiance classes. The irradiance breaks are obtained
# according to a log scale (logbreaks function) with different alpha
# parameters for each candidate (0.005, 0.01).
## Data and inputs
tpr.nclass=14
irdc.nclass=15
```

```
irdc.alpha=c(0.005, 0.01)
p2<-expand.grid(tpr.nclass, irdc.alpha, irdc.nclass)
parlist.tpr<-split(p2[,1], seq(nrow(p2)))
parlist.irdc<-split(p2[,2:3], seq(nrow(p2)))
parlist.irdc<-lapply(
     parlist.irdc,function(x){
     list(x[[1]],x[[2]])}
)
m.irdc <- as.matrix(Irradiance[,-c(1)])
m.tpr <- as.matrix(Temperature[,-c(1)])

# For the constructed models, only two regularization parameter ratios
# penratios=c(1/25,5) is used. In a real case, we will have to evaluate
# more candidates and regularization parameters ratio.
ex_sp<-spicefp(y=FerariIndex_Difference$fi_dif,
              fp1=m.irdc,
              fp2=m.tpr,
              fun1=logbreaks,
              fun2=linbreaks,
              parlists=list(parlist.irdc,
                            parlist.tpr),
              penratios=c(1/25,5),
              appropriate.df=NULL,
              nknots = 100,
              ncores =2,
              write.external.file = FALSE)

# coef_spicefp
## coefficients based on the parameters of the model
## focus on model selected by Mallows's Cp at iteration 1

start_time_spc <- Sys.time()
results.eval.iter1<-ex_sp$Evaluations[[1]]$Evaluation.results$evaluation.result
c.mdl <- coef_spicefp(ex_sp, iter_=1,
                      criterion =NULL,
                      nmodels=NULL,
  model.parameters=results.eval.iter1[which.min(results.eval.iter1$Cp_),],
                      ncores = 1,
                      write.external.file =FALSE)

g1<-c.mdl$coef.list$'231'$Candidate.coef.NA.finemeshed
g1.x<-as.numeric(rownames(g1))
g1.y<-as.numeric(colnames(g1))
duration_spc <- Sys.time() - start_time_spc

library(fields)
plot(c(10,2000),c(15,45),type= "n", axes = FALSE,
     xlab = "Irradiance (mmol/m2/s - Logarithmic scale)",
     ylab = "Temperature (deg C)",log = "x")
rect(min(g1.x),min(g1.y),max(g1.x),max(g1.y), col="black", border=NA)
image.plot(g1.x,g1.y,g1, horizontal = FALSE,
           col=designer.colors(64, c("blue","white")),
           add = TRUE)
```

```
axis(1) ; axis(2)

## Let's visualize the same model from other arguments of coef_spicefp
c.crit <- coef_spicefp(ex_sp, iter_=1,
                       criterion ="Cp_",nmodels=1,
                       ncores = 1,
                       write.external.file =FALSE)
g2<-c.crit$coef.list$'231'$Candidate.coef.NA.finemeshed
g2.x<-as.numeric(rownames(g2))
g2.y<-as.numeric(colnames(g2))
plot(c(10,2000),c(15,45),type= "n", axes = FALSE,
     xlab = "Irradiance (mmol/m2/s - Logarithmic scale)",
     ylab = "Temperature (deg C)",log = "x")
rect(min(g2.x),min(g2.y),max(g2.x),max(g2.y), col="black", border=NA)
image.plot(g2.x,g2.y,g2, horizontal = FALSE,
           col=designer.colors(64, c("blue","white")),
           add = TRUE)
axis(1) ; axis(2)
closeAllConnections()
```

`evaluate.candidates` *evaluate.candidates*

Description

This function performs for each candidate matrix, a Generalized Fused Lasso (sparse fused lasso 2d or 3d) and computes various statistics and information criteria related to the constructed model.

Usage

```
evaluate.candidates(
  candmatrices,
  y,
  penratios,
  nknots,
  appropriate.df = NULL,
  ncores = parallel::detectCores() - 1,
  penfun = NULL,
  file_name = "parametertable",
  write.external.file = TRUE
)
```

Arguments

`candmatrices`	List. Output of the "candidates" function. The spicefp dimension is the first element. The second contains many lists of one candidate matrix and related vector with index and numbers of class intervals used per predictor. The other

elements of the lists are the inputs of "candidates" function. If the user does not need the "candidates" function for the creation of candmatrices, it is possible to build a list provided that it respects the same structure as well as the names of the outputs of the "candidates" function. In this case only the first two elements of the list are essential: spicefp.dimension and candidates. The remaining elements can be NULL.

`y` numerical vector. Contains the dependent variable. This vector will be used as response variable in the construction of models involving each candidate matrix.

`penratios` numeric vector with values greater than or equal to 0. It represents the ratio between the regularization parameters of parsimony and fusion. When penratios=0, it corresponds to the pure fusion. The higher its value, the more parsimonious the model is.

`nknots` integer. For one value in penratios vector, it represents the number of models that will be constructed for each candidate matrix. It is the argument "nlam" of `coef.genlasso` function. This argument can also be NULL. In this case, the argument appropriate.df must be provided.

`appropriate.df` (appropriate degree of freedom) NULL by default. Numerical vector with values greater than or equal to 1. The degree of freedom of generalized fused problem is equal the number of connected components. A connected component gives information on a group of non-zero coefficients sharing the same value and connected by a contiguity matrix. More simply, it can be interpreted as a group of coefficients that have a unique influence. When the user has a prior idea of the number of zones of influence that the desired solution could contain, it is advisable to provide appropriate.df, a vector of appropriate degrees of freedom. In this case, nknots must be NULL.

`ncores` numbers of cores that will be used for parallel computation. By default, it is equal to detectCores()-1.

`penfun` function with 2 arguments (dim1, dim2) when dealing with 2 dimensional spiceFP or 3 arguments (dim1, dim2, dim3) when dealing with 3 dimensional spiceFP. The argument order in the penalty function is associated with the order of numbers of class intervals used per predictor in the second element of candmatrices argument. NULL by default. When penfun=NULL, getD2dSparse of genlasso or getD3dSparse is used according to the dimension of spiceFP.

`file_name` character. It is the name of the file in which the evaluation summary of all the candidate matrices is stored. This file is saved in your working directory.

`write.external.file`
logical. Indicates whether the result table should be written as a file (txt) in your working directory. It is recommended to use write.external.file=TRUE when evaluating a large number of candidate matrices (more than 100) in order to keep memory available.

Details

This function mainly returns statistics on the models built based on the candidate matrices. For each candidate matrix, length(penratios) x nknots or length(penratios) x length(appropriate.df) models are constructed in order to estimate the regularization parameters and to perform a variable selection. The computed statistics provide information on the quality of the models. For obvious reasons

of memory management, the coefficients related to each of these models are not stored. The statistics are stored in a file named via the argument file_name and can be consulted to get an idea of the state of progress of the program. The genlasso package is used for the implementation of the Generalized Fused Lasso.

Value

The output is a list with :

evaluation.result Same as file_name. The file contains a matrix with in columns : the candidate index (Candidate_id), the value of penratios used for this model (Pen_ratio), the parameter that penalizes the difference in related coefficients (PenPar_fusion), the degree of freedom of the model (Df_), the residual sum of squares (RSS_), the Akaike information criterion (AIC_), the Bayesian information criterion (BIC_), the Mallows' Cp (Cp_), the Generalized Cross Validation (GCV_), the slope of the regression lm($y \sim X\beta$) (Slope_), the ratio $var(y - X\beta)/var(y)$ (Var_ratio).

response.variable, penalty.ratios, nknots, appropriate.df, penalty.function Exactly the inputs y, penratios, nknots, appropriate.df, penfun

Examples

```
# Constructing 2 candidates for spiceFP data (temperature and Irradiance)
linbreaks<-function(x,n){
     sort(round(seq(trunc(min(x)),
                ceiling(max(x)+0.001),
                length.out =unlist(n)+1),
           1)
         )
}
# In this example, we will evaluate 2 candidates (each having 10
# temperature classes and respectively 10 and 20 irradiance classes).
# Only one value is used for alpha (logbreaks argument)
tpr.nclass=10
irdc.nclass=c(10,20)
irdc.alpha=0.005
p2<-expand.grid(tpr.nclass, irdc.alpha, irdc.nclass)
parlist.tpr<-split(p2[,1], seq(nrow(p2)))
parlist.irdc<-split(p2[,2:3], seq(nrow(p2)))
parlist.irdc<-lapply(
   parlist.irdc,function(x){
   list(x[[1]],x[[2]])}
)
m.irdc <- as.matrix(Irradiance[,-c(1)])
m.tpr <- as.matrix(Temperature[,-c(1)])
test2<-candidates(fp1=m.irdc,
                  fp2=m.tpr,
                  fun1=logbreaks,
                  fun2=linbreaks,
                  parlists=list(parlist.irdc,
                                parlist.tpr),
                  xcentering = TRUE,
```

```
                xscaling = FALSE,
                ncores=2)
# Evaluating candidates
# For the constructed models, only one regularization parameter ratio
# penratios=c(1) is used. In a real case, we will have to evaluate
# more candidates and regularization parameters ratio.
start_time_ev <- Sys.time()
evcand<-evaluate.candidates(candmatrices = test2,
                            y=FerariIndex_Difference$fi_dif,
                            penratios=c(1),
                            appropriate.df=NULL,
                            nknots = 100,
                            ncores=2,
                            write.external.file = FALSE)
duration_ev <- Sys.time() - start_time_ev
tab_res<-evcand$evaluation.result
dim(tab_res)
tab_res[which.min(tab_res$AIC_),]

closeAllConnections()
```

`FerariIndex_Difference`

FerariIndex_Difference of vine dataset

Description

Data were collected during an experiment conducted on a vineyard of the INRAE/Institut Agro campus at Montpellier in 2014 (Syrah vines). The objective of the experiment was to study the influence of the micro-climate (temperature and irradiance) at the grape level on the anthocyanin contents of the berries indicated by the Ferari index. This dataset contains Ferari index differences between August 01, 2014 at 09:00 am and July 24th, 2014 at 09:00 am. The individuals are in rows. The individuals' names (Indiv1,...,Indiv32) are used to name the rows. The same individuals are also present in the irradiance and temperaure datasets.

Usage

```
FerariIndex_Difference
```

Format

A data frame with 32 observations and 1 variable.

fi_dif numeric. Ferari index differences between July 24th, 2014 at 09:00 am and August 01, 2014 at 09:00 am.

Source

These data were acquired during the Innovine project, funded by the Seventh Framework Programme of the European Community (FP7/2007-2013), under Grant Agreement No. FP7-311775.

`finemeshed2d` *finemeshed2d*

Description

Function that helps to transform a vector into a matrix (with a fine mesh). In the implementation of the spiceFP approach, it allows to transform matrices of coefficients having different dimensions into matrices of the same dimension in order to perform arithmetic operations. In practice, the matrix to be transformed is associated with a contingency table, which implies numerical variables for which classes have been created.

Usage

```
finemeshed2d(
  x,
  n.breaks1 = 1000,
  n.breaks2 = 1000,
  round.breaks1 = 9,
  round.breaks2 = 9
)
```

Arguments

`x`	vector or one column matrix to scale. This vector comes from the vectorization of the matrices to be transformed. x is named using the concatenation of the names of the rows and the names of the columns of the matrix to be transformed, as shown in the example below.
`n.breaks1`	integer. Number of breaks needed for the first variable. The variable for which classes are in first position when constructing x's names is the first variable.
`n.breaks2`	integer. Number of breaks needed for the second variable. The variable for which classes are in second position when constructing x's names is the second variable.
`round.breaks1`	integer. Number of decimals for breaks of the first variable.
`round.breaks2`	integer. Number of decimals for breaks of the second variable.

Details

This function is designed to return a fine meshed matrix and breaks associated. In order to obtain a fine mesh, a high number of breaks must be fixed.

Value

Returns:

finemeshed.matrix Matrix of dimension n.breaks2 x n.breaks1. The row and column names of finemeshed.matrix are the breaks created from each variable and the associated n.breaks. Each value of finemeshed.matrix is equal to the value of x indexed by the classes containing the row and column names of finemeshed.matrix

finemeshed.values1 First variable breaks

finemeshed.values2 Second variable breaks

Examples

```
set.seed(45)
count_table<- hist_2d(x = rnorm(1000),
                      y = rnorm( 1000,5,0.1),
                      breaks_x = seq(-4, 4, by =1),
                      breaks_y = seq(2, 8, by =1))$Hist.Values

df.x<-as.data.frame.table(count_table)
x<-df.x$Freq
names(x)<-paste0(df.x$Var1,"_",df.x$Var2)

res.fm2d <- finemeshed2d(x,100,100)
dim(res.fm2d$finemeshed.matrix)
```

`finemeshed3d` *finemeshed3d*

Description

Function that helps to transform a vector into a 3 dimensional array (with a fine mesh). In the implementation of the spiceFP approach, it allows to transform matrices of coefficients having different dimensions into matrices of the same dimension in order to perform arithmetic operations. In practice, the 3d array to be transformed is associated with a contingency table, which implies numerical variables for which classes have been created.

Usage

```
finemeshed3d(
  x,
  n.breaks1 = 10,
  n.breaks2 = 1000,
  n.breaks3 = 500,
  round.breaks1 = 9,
  round.breaks2 = 9,
  round.breaks3 = 9
)
```

Arguments

x	vector or one column matrix to scale. This vector comes from the vectorization of the 3d array to be transformed. x is named using the concatenation of the names of the dimension of the array to be transformed, as shown in the example below.
n.breaks1	integer. Number of breaks needed for the first variable The variable for which classes are in first position when constructing x's names is the first variable.
n.breaks2	integer. Number of breaks needed for the second variable. The variable for which classes are in second position when constructing x's names is the second variable.
n.breaks3	integer. Number of breaks needed for the third variable. The variable for which classes are in third position when constructing x's names is the third variable.
round.breaks1	integer. Number of decimals for breaks of the first variable.
round.breaks2	integer. Number of decimals for breaks of the second variable.
round.breaks3	integer. Number of decimals for breaks of the third variable.

Details

This function is designed to return a 3d fine meshed array and breaks associated. In order to obtain a fine mesh, a high number of breaks must be fixed.

Value

Returns:

finemeshed.array Array of dimension n.breaks1 x n.breaks2 x n.breaks3. The dimension names of finemeshed.array are the breaks created from each variable and the associated n.breaks. Each value of finemeshed.array is equal to the value of x indexed by the classes containing the row and column names of finemeshed.array

finemeshed.values1 First variable breaks

finemeshed.values2 Second variable breaks

finemeshed.values3 Third variable breaks

Examples

```
set.seed(4)
count_table<-hist_3d(x = rnorm(1000),
                     y = rnorm( 1000,5,0.1),
                     z = rnorm( 1000,2,1),
                     breaks_x = seq(-4, 4, by =1),
                     breaks_y = seq(2, 8, by =1),
                     breaks_z = seq(-3, 6, by =1))$Hist.Values

df.x<-as.data.frame.table(count_table)
x<-df.x$Freq
names(x)<-paste0(df.x$Var1,"_",df.x$Var2,"_",df.x$Var3)
```

```
res.fm3d<- finemeshed3d(x,10,50,100)
dim(res.fm3d$finemeshed.array)
```

---

getD3dSparse *getD3dSparse*

Description

getD3dSparse is a function that helps to construct generalized lasso penalty matrix D when using the fusedlasso function over a 3 dimensional grid

Usage

```
getD3dSparse(dim1, dim2, dim3)
```

Arguments

dim1	positive integer. Based on a 3 dimensional grid, dim1 represents the number of units represented on the first dimension
dim2	positive integer which represents the number of units represented on the second dimension
dim3	positive integer which represents the number of units represented on the third dimension

Details

The function returns a sparse penalty matrix providing information on the connections between the variables during the implementation of a generalizad fused lasso.

Value

a matrix with dim1 x dim2 x dim3 columns. Each row represents an edge (a link between 2 variables) and is constructed with the couple (-1, 1), relative to these 2 variables and 0 for all others. In the context of a generalized fused lasso, this matrix penalizes only the differences in coefficients (fusion). To obtain parsimony in addition to the fusion, a diagonal matrix with the same number of columns must be bound to the penalty matrix constructed by getD3dSparse. This matrix will contain diagonally the ratio: parsimony penalty parameter on fusion penalty parameter. When using fusedlasso function, this operation is performed when you provide the argument gamma.

Examples

```
library(genlasso)
library(Matrix)
D<-getD3dSparse(2,3,2)
plot(getGraph(D))
```

hist_2d *hist_2d*

Description

This function results from a modification of the hist2d function of the gplots package in order to build the 2D histogram with breaks directly provided as inputs of the new function.

Usage

```
hist_2d(
  x,
  y,
  breaks_x,
  breaks_y,
  same.scale = FALSE,
  na.rm = TRUE,
  FUN = base::length
)
```

Arguments

x	either a numerical vector to be partitioned or a matrix of 2 numerical columns to be partitioned.
y	a numerical vector to be partitioned. Not required if x is a matrix.
breaks_x	a numerical vector. Contains the breaks related to x for the histogram
breaks_y	a numerical vector. Contains the breaks related to y for the histogram
same.scale	logical. Default to FALSE. If TRUE, breaks_x will be used for x and y
na.rm	logical. Default to TRUE. Indicates whether missing values should be removed
FUN	function used to summarize bin contents.

Details

The default function used for the argument FUN is the function length. When another function is used, it is applied on x, or on the first column of x if this is a two-column matrix. The lower limit of each class interval is included in the class and the upper limit is not.

Value

Using a given set of breaks per each variable, the function returns :

Hist.Values a matrix with in rows class intervals of x and in columns class intervals of y. Contingency table is returned if FUN=length

breaks_x, breaks_y same as the inputs of the function

Midpoints.x, Midpoints.y the midpoints for each bin per variable

nobs.x , nobs.y number of observations of x and y

n.bins vector of 2 elements containing the number of bins for x and y

Examples

```
set.seed(45)
hist_2d(x = rnorm(1000),
        y = rnorm( 1000,5,0.1),
        breaks_x = seq(-4, 4, by =1),
        breaks_y = seq(2, 8, by =1))
```

`hist_3d` *hist_3d*

Description

This function can be used in order to construct a 3D histogram based on 3 variables and relative breaks directly provided as inputs.

Usage

```
hist_3d(
  x,
  y,
  z,
  breaks_x,
  breaks_y,
  breaks_z,
  same.scale = FALSE,
  na.rm = TRUE,
  FUN = length
)
```

Arguments

x	either a numerical vector to be partitioned or a matrix with 3 numerical columns to be partitioned.
y	a numerical vector to be partitioned. Not required if x is a matrix.
z	a numerical vector to be partitioned. Not required if x is a matrix
breaks_x	a numerical vector. Contains the breaks related to x for the histogram
breaks_y	a numerical vector. Contains the breaks related to y for the histogram
breaks_z	a numerical vector. Contains the breaks related to z for the histogram
same.scale	logical. Default to FALSE. If TRUE, breaks_x will be used for x, y and z
na.rm	logical. Default to TRUE. Indicates whether missing values should be removed
FUN	function used to summarize bin contents.

Details

The default function used for the argument FUN is the function length. When another function is used, it is applied on x or on the first column of x if this is a three-column matrix. The lower limit of each class interval is included in the class and the upper limit is not.

Value

Using a given set of breaks per each variable, the function returns :

Hist.Values a 3 dimensional array. The 1st (respectively 2nd, 3rd) dimension is related to the class intervals of x (resp. y, z). Contingency table is returned if FUN=length

breaks_x, breaks_y, breaks_z same as the inputs of the function

Midpoints.x, Midpoints.y, Midpoints.z the midpoints for each bin per variable

nobs.x , nobs.y, nobs.z number of observations of x, y and z

n.bins vector of 3 elements containing the number of bins for x, y and z

Examples

```
set.seed(4)
hist_3d(x = rnorm(1000),
        y = rnorm( 1000,5,0.1),
        z = rnorm( 1000,2,1),
        breaks_x = seq(-4, 4, by =1),
        breaks_y = seq(2, 8, by =1),
        breaks_z = seq(-2, 6, by =1))
```

Irradiance — *Photosynthetic Photon Flux Density PPFD (PPFD) measurements of vine dataset*

Description

Data were collected during an experiment conducted on a vineyard of the INRAE/Institut Agro campus at Montpellier in 2014 (Syrah vines). The objective of the experiment was to study the influence of the micro-climate (temperature and irradiance) at the grape level on the anthocyanin contents of the berries indicated by the Ferari index. This dataset is related to irradiance measurements in the morning (sunrise to twelve am) between July 24th, 2014 at 09:00 am and August 01, 2014 at 09:00 am. These observations are made at the same time (every 12 minutes) as the temperature observations. The individuals are in columns while the observation times are in rows. The same individuals are also present in the Temperature and FerariIndex_Difference datasets.

Usage

```
Irradiance
```

Format

A data frame (of one functionnal variable) with 127 rows (observation times) and 33 columns: the 1st one is a character vector which corresponds to date-time in format "yyyy-mm-dd hh:mm:ss", the others are numeric vectors made of the observations of irradiance (PPFD) measured in $10^{-6}mol.m^{-2}.s^{-1}$ on each of the 32 statistical individuals Indiv1,...,Indiv32. Irradiance corresponds to the number of incident photons useful for photosynthesis, received per unit of time on a horizontal surface unit.

Source

These data were acquired during the Innovine project, funded by the Seventh Framework Programme of the European Community (FP7/2007-2013), under Grant Agreement No. FP7-311775.

`logbreaks` *logbreaks*

Description

A function that allows to obtain histogram class limits following a logarithmic scale. It also has a parameter that allows to set the scale at your convenience.

Usage

```
logbreaks(
  x,
  parlist = list(alpha, J),
  round_breaks = 0,
  plot_breaks = FALSE,
  effect.threshold.begin = NA,
  effect.threshold.end = NA
)
```

Arguments

`x`	either a numeric vector to be partitioned or a numeric vector containing the minimum and maximum of the vector to be partitioned.
`parlist`	a list of 2 elements. The first one is alpha, a numeric and positive value. It is a parameter affecting the number of breaks closed to the minimum. The second one is J. It is a nonnegative and nonzero integer and represent the selected number of classes.
`round_breaks`	a nonnegative integer. Equal to 0 by default, it is the number of decimal values of the breaks.
`plot_breaks`	logical. FALSE by default. If TRUE, the breaks are plotted.
`effect.threshold.begin`	NA by default. Numeric value between the minimum and maximum of x. If it isn't NA, the first class is created with xmin and effect.threshold.begin.
`effect.threshold.end`	NA by default. Numeric value between the minimum and maximum of x. If it isn't NA, the last class is created with xmax and effect.threshold.end.

Details

The breaks are obtained as follows:

$$L(w) = \min(x) + \frac{e^{\alpha \frac{w-1}{J}} - 1}{e^{\alpha} - 1}(\max(x) - \min(x)),\ w = 1, \ldots, J+1.$$

Value

The return is a numeric vector of length J+1 with the breaks obtained following a log scale.

Examples

```
logbreaks(c(10,1000), parlist=list(0.2,5))
logbreaks(c(10,1000), parlist=list(0.2,5),plot_breaks=TRUE)
```

`meancoef` *meancoef*

Description

This function can be used to compute the mean of coefficients from different partitions in the context of the spicefp approach.

Usage

```
meancoef(coef.list, weight)
```

Arguments

`coef.list`	list. The second element of the coef_spicefp function outputs. It has the same name as the argument.
`weight`	a numerical vector of weights with the same length as coef.list.

Details

Here, the fine-mesh coefficients are weighted and a weighted mean is deduced. If the user wishes, he can use as weights the slopes associated with the qualities of the models concerned.

Value

Returns a list of :

weighted_mean fine-mesh matrix or array with the weighted mean of the coefficients

y.estimated weighted estimation of $X\beta$

coefficients.array An array with all the fine-mesh coefficients that will be used to compute the weighted mean

weight same as inputs

Examples

```
##linbreaks: a function allowing to obtain breaks linearly
linbreaks<-function(x,n){
     sort(round(seq(trunc(min(x)),
                ceiling(max(x)+0.001),
                length.out =unlist(n)+1),
           1)
          )
}
# In this example, we will evaluate 2 candidates with 14 temperature
# classes and 15 irradiance classes. The irradiance breaks are obtained
# according to a log scale (logbreaks function) with different alpha
# parameters for each candidate (0.005, 0.01).
## Data and inputs
tpr.nclass=14
irdc.nclass=15
irdc.alpha=c(0.005, 0.01)
p2<-expand.grid(tpr.nclass, irdc.alpha, irdc.nclass)
parlist.tpr<-split(p2[,1], seq(nrow(p2)))
parlist.irdc<-split(p2[,2:3], seq(nrow(p2)))
parlist.irdc<-lapply(
   parlist.irdc,function(x){
   list(x[[1]],x[[2]])}
)
m.irdc <- as.matrix(Irradiance[,-c(1)])
m.tpr <- as.matrix(Temperature[,-c(1)])

# For the constructed models, only two regularization parameter ratios
# penratios=c(1/25,5) is used. In a real case, more candidates
# and regularization parameter ratios should be evaluated.
ex_sp<-spicefp(y=FerariIndex_Difference$fi_dif,
              fp1=m.irdc,
              fp2=m.tpr,
              fun1=logbreaks,
              fun2=linbreaks,
              parlists=list(parlist.irdc,
                            parlist.tpr),
              penratios=c(1/25,5),
              appropriate.df=NULL,
              nknots = 100,
              ncores =2,
              write.external.file = FALSE)

## Focus on the 2 best models retained by the AIC criterion at iteration 1
c.mdls <- coef_spicefp(ex_sp, iter_=1, criterion ="AIC_",
                      nmodels=2, ncores = 2,
                      dim.finemesh=c(1000,1000),
                      write.external.file = FALSE)

# meancoef
# Compute the mean of the coefficients of these models
```

```
mean.c.mdls<-meancoef(c.mdls$coef.list,
                      weight = c.mdls$Model.parameters$Slope_)
g3<-mean.c.mdls$weighted_mean
g3.x<-as.numeric(rownames(g3))
g3.y<-as.numeric(colnames(g3))


library(fields)
plot(c(10,2000),c(15,45),type= "n", axes = FALSE,
     xlab = "Irradiance (mmol/m2/s - Logarithmic scale)",
     ylab = "Temperature (deg C)",log = "x")
rect(min(g3.x),min(g3.y),max(g3.x),max(g3.y), col="black", border=NA)
image.plot(g3.x,g3.y,g3, horizontal = FALSE,
           col=designer.colors(256, c("blue","white","red")),
           add = TRUE)
axis(1) ; axis(2)

closeAllConnections()
```

---

spicefp *spicefp*

Description

This function is used to implement the spiceFP approach. This approach transforms 2 (by default) or 3 functional predictors into candidate explonatory matrices in order to identify joint classes of influence. It can take functional predictors and partitioning functions as inputs in order to create candidate matrices to be evaluated. The user can choose among the existing partitioning functions (as logbreaks) or provide his own partitioning functions specific to the functional predictors under consideration. The user can also directly provide candidate matrices already constructed as desired.

Usage

```
spicefp(
  y,
  fp1,
  fp2,
  fp3 = NULL,
  fun1,
  fun2,
  fun3 = NULL,
  parlists,
  xcentering = TRUE,
  xscaling = FALSE,
  candmatrices = NULL,
  K = 2,
```

```
  criterion = "AIC_",
  penratios = c(1/10, 1/5, 1/2, 1, 2, 5, 10),
  nknots = 50,
  appropriate.df = NULL,
  penfun = NULL,
  dim.finemesh = c(1000, 1000),
  file_name = paste0("parametertable", 1:2),
  ncores = parallel::detectCores() - 1,
  write.external.file = TRUE
)
```

Arguments

y	a numerical vector. Contains the dependent variable. This vector will be used as response variable in the construction of models involving each candidate matrix.
fp1	a numerical matrix with in columns observations of one statistical individual to partition. Each column corresponds to the functional predictor observation for one statistical individual. The order of the statistical individuals is the same as in fp2. It is assumed that no data are missing and that all functional predictors are observed on an equidistant (time) scale.
fp2	a numerical matrix with the same number of columns and rows as fp1. Columns are also observations. The order of the statistical individuals is the same as in fp1.
fp3	NULL by default. A numerical matrix with the same number of columns and rows as fp1 and fp2. The order of the statistical individuals is the same as in fp1 and fp2.
fun1	a function object with 2 arguments. First argument is fp1 and the second is a list of parameters that will help to partition fp1, such as the number of class intervals, etc. For example using the logbreaks function, the list of parameters is equivalent to list(alpha, J). All the arguments to be varied for the creation of different candidate matrices must be stored in the parameter list. The other arguments must be set by default.
fun2	a function object with 2 arguments. First argument is fp2 and the second is a list of parameters.
fun3	NULL by default. Same as fun1 and fun2, a function with 2 arguments fp3 and a list of parameters.
parlists	a list of 2 elements when fp3 and fun3 are equal to NULL or of 3 elements when fp3 and fun3 are provided. All the elements of parlists are lists that have the same length. Each list contains all the lists of parameters that have to be used to create different candidates. The first element of parlists concerns the first functional predictor fp1, the second element is relative to fp2 and the third to fp3.
xcentering	TRUE by default. Defined whether or not the variables in the new candidate matrices should be centered.
xscaling	FALSE by default. Defined whether or not the variables in the candidate matrices should be scaled.

`candmatrices` NULL by default. List. Output of the "candidates" function. The spiceFP dimension is its first element. The second contains many lists of one candidate matrix and related vector with index and numbers of class intervals used per predictor. The other elements of the lists are the inputs of "candidates" function. If the user does not need the "candidates" function for the creation of candmatrices, it is possible to build a list while making sure that it respects the same structure as well as the names of the outputs of the "candidates" function. In this case, only the first two elements of the list are essential: spicefp.dimension and candidates. The remaining elements can be NULL.

`K` number of iterations of the spiceFP approach. Equal to 2 by default.

`criterion` character. One of "AIC_", "BIC_", "Cp_". The criterion to be used in each iteration in order to identify the best candidate matrix and to estimate the regulation parameters. This criterion is used to perform model selection as well as variable selection.

`penratios` a numeric vector with values greater than or equal to 0. It represents the ratio between the regularization parameters of parsimony and fusion. When penratios=0, it corresponds to the pure fusion. The higher its value, the more parsimonious the model is.

`nknots` integer. For one value in penratios vector, it represents the number of models that will be constructed for each candidate matrix. It is the argument "nlam" of `coef.genlasso` function. This argument can be also NULL. In this case, the argument appropriate.df must be provided.

`appropriate.df` (appropriate degree of freedom) NULL by default. When used, nknots must be NULL. It is the argument "df" of `coef.genlasso` function. When the user has a prior idea of the number of zones of influence that the solution could contain, it is advisable to provide appropriate.df, a vector of appropriate degrees of freedom. appropriate.df is a numerical vector with values greater than or equal to 1. The degree of freedom of generalized fused Lasso models is equal to the number of connected components. A connected component gives information on a group of non-zero coefficients sharing the same value and connected by a contiguity matrix. More simply, it can be interpreted as a group of coefficients that have a unique influence.

`penfun` function with 2 arguments (dim1, dim2) when dealing with 2 dimensional spiceFP, or with 3 arguments (dim1, dim2, dim3) when dealing with 3 dimensional spiceFP. The argument order in the penalty function is associated with the order of numbers of class intervals used per predictor in the second element of candmatrices argument. NULL by default. When penfun=NULL, getD2dSparse of genlasso or getD3dSparse is used according to the dimension of spiceFP.

`dim.finemesh` numeric vector of length 2 or 3. This vector informs about the dimension of the fine-mesh arrays (or matrices) that will be used for the visualization of the sum of the coefficients selected at different iterations.

`file_name` character vector. Of length K, it contains the list of names that will be used to name the files containing informations on the candidate matrix models

`ncores` numbers of cores that will be used for parallel computation. By default, it is equal to detectCores()-1.

`write.external.file`
: logical. indicates whether the result table related to each iteration should be written as a file (txt) in your working directory. It is recommended to use write.external.file=TRUE when evaluating a large number of candidate matrices (more than 100) in order to keep memory available.

Details

Three main steps are involved to implement spiceFP: transformation of functional predictors, creation of a graph of contiguity constraints and identification of the best class intervals and related regression coefficients.

Value

Returns a list with:

Candidate.Matrices a list with candidate matrices and their characteristics. same as candmatrices if it has been provided.

Evaluations List of length less than or equal to K. Each element of the list contains information about an iteration. Contains the results related to the evaluation of the candidate matrices. These include the name of the file where the model information is stored, the best candidate matrix and related coefficients, the partition vector that indexes it, the $X\beta$ estimation, the residuals, etc.

coef.NA List of length less than or equal to K. For each iteration, it contains the coefficient vector where the coefficient value of never-observed joint modalities is NA

coef.NA.finemeshed List of length less than or equal to K. For each iteration, the coefficient vector is transformed into fine-mesh array or matrix allowing arithmetic operations to be performed between coefficients coming from different partitions

spicefp.coef fine-mesh array or matrix. Sum of the coefficients selected at all iterations

Examples

```
##linbreaks: a function allowing to obtain breaks linearly
linbreaks<-function(x,n){
    sort(round(seq(trunc(min(x)),
                ceiling(max(x)+0.001),
                length.out =unlist(n)+1),
            1)
        )
}

# In this example, we will evaluate 2 candidates with 14 temperature
# classes and 15 irradiance classes. The irradiance breaks are obtained
# according to a log scale (logbreaks function) with different alpha
# parameters for each candidate (0.005, 0.01).
## Data and inputs
tpr.nclass=14
irdc.nclass=15
irdc.alpha=c(0.005, 0.01)
```

```
p2<-expand.grid(tpr.nclass, irdc.alpha, irdc.nclass)
parlist.tpr<-split(p2[,1], seq(nrow(p2)))
parlist.irdc<-split(p2[,2:3], seq(nrow(p2)))
parlist.irdc<-lapply(
   parlist.irdc,function(x){
   list(x[[1]],x[[2]])}
)
m.irdc <- as.matrix(Irradiance[,-c(1)])
m.tpr <- as.matrix(Temperature[,-c(1)])

# For the constructed models, only two regularization parameter ratios
# penratios=c(1/25,5) are used. In a real case, we will have to evaluate
# more candidates and regularization parameters ratio.
start_time_sp <- Sys.time()
ex_sp<-spicefp(y=FerariIndex_Difference$fi_dif,
               fp1=m.irdc,
               fp2=m.tpr,
               fun1=logbreaks,
               fun2=linbreaks,
               parlists=list(parlist.irdc,
                             parlist.tpr),
               penratios=c(1/25,5),
               appropriate.df=NULL,
               nknots = 100,
               ncores =2,
               write.external.file=FALSE)

duration_sp <- Sys.time() - start_time_sp
# View(ex_sp$Evaluations[[1]]$Evaluation.results$evaluation.result)
# View(ex_sp$Evaluations[[2]]$Evaluation.results$evaluation.result)
# Visualization of the coefficients
g<-ex_sp$spicefp.coef
g.x<-as.numeric(rownames(g))
g.y<-as.numeric(colnames(g))

library(fields)
plot(c(10,2000),c(15,45),type= "n", axes = FALSE,
     xlab = "Irradiance (mmol/m²/s - Logarithmic scale)",
     ylab = "Temperature (°C)",log = "x")
rect(min(g.x),min(g.y),max(g.x),max(g.y), col="black", border=NA)
image.plot(g.x,g.y,g, horizontal = FALSE,
           col=designer.colors(256, c("blue","white","red")),
           add = TRUE)
axis(1) ; axis(2)

closeAllConnections()
```

Temperature *Temperature measurements of vine dataset*

Description

Data were collected during an experiment conducted on a vineyard of the INRAE/Institut Agro campus at Montpellier in 2014 (Syrah vines). The objective of the experiment was to study the influence of the micro-climate (temperature and irradiance) at the grape level on the anthocyanin contents of the berries indicated by the Ferari index. This dataset is related to temperature measurements in the morning (sunrise to twelve am) between July 24th, 2014 at 09:00 am and August 01, 2014 at 09:00 am. These observations are made at the same time (every 12 minutes) as the irradiance observations. The individuals are in columns while the observation times are in rows. The same individuals are also present in the Irradiance and FerariIndex_Difference datasets.

Usage

```
Temperature
```

Format

A data frame (of one functionnal variable) with 127 rows (observation times) and 33 columns: the 1st one is a character vector which corresponds to date-time in format "yyyy-mm-dd hh:mm:ss", the others are numeric vectors made of the observations of temperature measured in degree celsius on each of the 32 statistical individuals Indiv1,...,Indiv32.

Source

These data were acquired during the Innovine project, funded by the Seventh Framework Programme of the European Community (FP7/2007-2013), under Grant Agreement No. FP7-311775.

Index