R Package SpiceFP: a Sparse and Structured Procedure to Identify Combined Effects of Functional Predictors

Girault Gnanguenon Guesse
MISTEA INRAE
Université Montpellier

Patrice Loisel
MISTEA INRAE
Université Montpellier

Bénédicte Fontez
MISTEA Institut Agro
Université Montpellier

Thierry Simonneau
LEPSE INRAE
Université Montpellier

Nadine Hilgert
MISTEA INRAE
Université Montpellier

Isabelle Sanchez
MISTEA INRAE
Université Montpellier

Abstract

The R package named **SpiceFP** implements a scalar-on-function approach whose main objective is to model the joint influence of two functional predictors while providing interpretable coefficients. The originality is to partition the range of values of each functional predictor into intervals and then to create a contingency table (defining a partition). Several candidate partitions are defined this way, depending on the choice of the bin size. Each candidate partition forms a candidate design matrix which is used in a linear multiple regression model. A contiguity constraint was added to manage possible colinearity. Identification is performed through a Generalized Fused Lasso. The selection of the best candidate and of its relative regression coefficients is achieved by minimizing an information criteria. After a brief description of the approach, we present the different functionalities available in the package and then illustrate it on an example involving real observations.

Keywords: Joint influence, functional predictors, interpretable coefficients, Generalized Fused Lasso, R.

1. Introduction

The development of information technologies, observed over the last decades, has not occurred without some changes in the various data collected. The flow of information, sometimes obtained continuously, generates new types of data (curves, images, surfaces, etc.). In order to extract knowledge from these data or to use them in the prediction of a variable of interest, it is necessary to propose new statistical solutions or to adapt existing ones. In this context, the regression with functional data is commonly used (Ramsay and Silverman (2005)). In most production systems (industry, agronomy, health etc.), the variable of interest is observed at the final point (as the yield, at the end of the process). This final value is conditioned and impacted by past values of covariables we are able now to observe continuously (during the production process). This context is usually modeled with 'scalar-on-function' approaches, Reiss, Goldsmith, Shang, and Ogden (2017); Ma, Xiao, Liu, and Lindquist (2019). In a

different context, the development of available spatialised data (from medical imagery, GPS, image from satellite or drone etc.) have led to specific approaches for 'scalar-on-image' or 'spatial functional' regressions like (Goldsmith, Bobb, Crainiceanu, Caffo, and Reich (2011), Guillas and Lai (2010). Both contexts offer new tools for multiple functional regression, we can cite as an example FAME (Functional Adaptive Model Estimation) from James and Silverman (2005). Most of the linear or generalized linear modeling also included a penalized estimation like Marx and Eilers (2005) or Gertheiss, Maity, and Staicu (2013) for examples. Some R packages were developed for multivariate functional models.

According to Usset, Staicu, and Maity (2016), 'a common assumption made by all the above mentioned models is that the effects of the functional predictors are additive, thus any interaction between the functional covariates is not taken into account'. Ignoring interaction effect in linear models may lead to biased estimation and incorrect conclusions. Therefore Usset *et al.* (2016) proposed an interaction model for functional regression. Their proposition is a generalization of what is usually done in multiple linear regression with a product (convolution) between functional predictors to model the interaction term. Another way to account for interaction is to define combinations of ranges of functional predictors values and identify the combinations that have an impact on the variable of interest. This way offers an alternative to the scalar-on-function regression that is more interpretable. The notion of interpretability for functional linear regression has grown up with James, Wang, and Zhu (2009) (R code available). A Bayesian extension was proposed by Grollemund, Abraham, Baragatti, and Pudlo (2019) with the R package **bliss** available on CRAN. The 'scalar-on-function' multiple linear or generalized linear regressions are implemented in R packages (**FDA**, **funcreg** or **refund**), no identification of interaction terms is provided. The package **SpiceFP** offers to fill this gap. The full description of the method is available in Gnanguenon Guesse, Loisel, Fontez, Simonneau, and Hilgert (202X).

The originality of the Sparse and Structured Procedure to Identify Combined Effects of Functional Predictors (SpiceFP) is to partition the range of values of each functional predictor into intervals (with bins of equal size). The idea is to transform each functional variable into a categorical variable and then to create a contingency table (which is in fact a partition of the predictors' observation domain). It is assumed that only the time spent in a combined class of intervals affects the variable of interest. This assumption is supported by the idea of a certain kind of independence and stationarity during the production process: same causes, same effects, not conditioned by the past. Several candidate partitions are defined this way, depending on the choice of the bin size. Each candidate partition forms a candidate design matrix which is used in a linear multiple regression model. A contiguity constraint was added between adjacent class intervals to manage possible colinearity. Identification is performed through a Generalized Fused Lasso using each candidate matrix as input variables. The selection of the best candidate and of its relative regression coefficients is achieved by minimizing an information criteria.

In a nutshell, the package **SpiceFP** offers a new functional approach whose main objective is the interpretability of the result. Under the hypothesis of combined influence of univariate functional predictors on a scalar response, the SpiceFP approach provides a surface (support) describing areas of influence and non-influence.

This paper is organised as follows. Section 2 briefly presents the methodology of this approach. The reader may refer to Gnanguenon Guesse *et al.* (202X) for an in-depth presentation of the

method. Some examples of use are presented in section 4.

2. Presentation of the approach

The SpiceFP algorithm is based on a transformation of functional variables into categorical variables by defining joint modalities from which we derived a collection of multiple regression models, where the regressors are the frequencies associated to the joint modalities. Regressors are candidate matrices under contiguity constraints. Generalized Fused Lasso regressions are performed in order to identify the best model. The **SpiceFP** package provides a set of easy-to-use functions to implement the algorithm on a dataset. To combine our exploration objectives with technical constraints (such as a potentially small amount of data, for example), we also propose an iterative extension of the SpiceFP algorithm, which explores a larger space of solutions, possibly at the cost of a slight overestimation.

The 4 main steps of the procedure are described in this section:

1. The functional variables $\mathcal{A}$ and $\mathcal{B}$ are transformed into categorical variables which requires the construction of a contingency table (counting t) at the scale of each individual i, $i = 1, ..., n$, see figure 1. The information contained in the n contingency tables is used to construct a matrix of predictors with n rows.

2. Generalized Lasso Regression is used to estimate the coefficients of the joint modalities. The generalized version allows a constraint on the model estimation to enforce continuity between adjacent joint modalities.

3. Model selection is done with respect to Akaike or Bayesian information criteria

4. The iterative step, optional, relies on the residuals: once an iteration is completed, the residuals are used as the response variable in a new iteration.

The overall approach is outlined in the remainder of this section.

2.1. Transformation of functional predictors into a set of candidate matrices

For n statistical individuals, consider the samples of two explanatory functional variables $\mathcal{A}$ and $\mathcal{B}$ expressed as $(\mathcal{A}_i)_{i=1,...,n}$ and $(\mathcal{B}_i)_{i=1,...,n}$, where i stands for an individual and $(y_i)_{i=1,...,n}$ the sample of the corresponding scalar response variable y.

Both $\mathcal{A}$ and $\mathcal{B}$ are observed on the same set T of fixed observation variables. We note $\mathcal{A}_i(t)$, respectively $\mathcal{B}_i(t)$, the observations of $\mathcal{A}$, respectively $\mathcal{B}$, at $t \in T$ (t can be time, wavelength or other observation variable) for an individual i.

Contingency table per individual

Using $n_{\mathcal{A}} + 1$ (*resp.* $n_{\mathcal{B}} + 1$) breaks, the explanatory variables $\mathcal{A}_i$ (*resp.* $\mathcal{B}_i$) are partitioned in $n_{\mathcal{A}}$ (*resp.* $n_{\mathcal{B}}$) class intervals, the same for all i, according to a linear scale (equidistant breaks). These breaks, denoted $L_{\mathcal{A}}(v),\ v = 1, \ldots, n_{\mathcal{A}} + 1$ (*resp.* $L_{\mathcal{B}}(w),\ w = 1, \ldots, n_{\mathcal{B}} + 1$) are computed as follows:

$$L_{\mathcal{A}}(v) = \underline{\mathcal{A}} + \frac{v-1}{n_{\mathcal{A}}} \left(\bar{\mathcal{A}} - \underline{\mathcal{A}}\right),\ v = 1, \ldots, n_{\mathcal{A}} + 1, \tag{1}$$

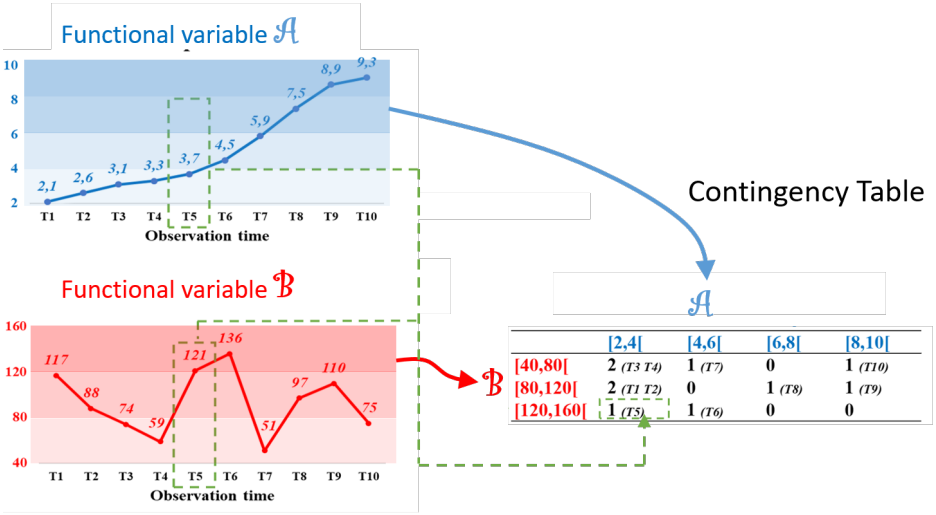

Figure 1: Contingency table per individual

with $\underline{\mathcal{A}} \in \mathbb{R}$ and $\bar{\mathcal{A}} \in \mathbb{R}$ the minimum and maximum of the observed values of $(\mathcal{A}_i)_i$. The breaks $L_{\mathcal{B}}(w)$ can equivalently be obtained by using $\underline{\mathcal{B}} \in \mathbb{R}$ and $\bar{\mathcal{B}} \in \mathbb{R}$ the minimum and maximum of the observed values of $(\mathcal{B})_i$. The class intervals obtained for partitioning the $(\mathcal{A})_i$ are $I_{\mathcal{A}}(v) = [L_{\mathcal{A}}(v), L_{\mathcal{A}}(v+1)[,\ v = 1, \ldots, n_{\mathcal{A}}$ and those for partitioning the $(\mathcal{B})_i$ are $I_{\mathcal{B}}(w) = [L_{\mathcal{B}}(w), L_{\mathcal{B}}(w+1)[,\ w = 1, \ldots, n_{\mathcal{B}}$. The numbers of class intervals $n_{\mathcal{A}}$ and $n_{\mathcal{B}}$ have to be set to compute these breaks. Let $u = (n_{\mathcal{A}}, n_{\mathcal{B}})$ denote the partition vector. For all couples $(\mathcal{A}_i, \mathcal{B}_i)$, we obtain the frequency bivariate histogram as a contingency table C_i^u, of dimension $n_{\mathcal{A}} \times n_{\mathcal{B}}$, whose components $C^u_{i,(v,w)}$ are obtained through:

$$C^u_{i,(v,w)} = \sum_{t \in T} \mathbb{1}_{\mathcal{A}_i(t) \in I^u_{\mathcal{A}}(v),\ \mathcal{B}_i(t) \in I^u_{\mathcal{B}}(w)} = Card\ \{t \in T | \mathcal{A}_i(t) \in I^u_{\mathcal{A}}(v), \mathcal{B}_i(t) \in I^u_{\mathcal{B}}(w)\}, \quad (2)$$

for all $v = 1, \ldots, n_{\mathcal{A}}$, $w = 1, \ldots, n_{\mathcal{B}}$ and each $u = (n_{\mathcal{A}}, n_{\mathcal{B}})$, with $\sum_{v=1}^{n_{\mathcal{A}}} \sum_{w=1}^{n_{\mathcal{B}}} C^u_{i,(v,w)} = Card(T)$. From a theoretical point of view, when t is a time step, $C^u_{i,(v,w)}$ represents a discrete approximation of the density of the time spent by the individual i with variable $\mathcal{A}_i$ in $L_{\mathcal{A}}(v)$ and variable $\mathcal{B}_i$ in $L_{\mathcal{B}}(w)$. In practice, it is the number of times that the observations of $\mathcal{A}_i$ and $\mathcal{B}_i$ are at the same time in $I^u_{\mathcal{A}}(v) \times I^u_{\mathcal{B}}(w)$. These combinations will be referred to as joint modalities hereafter.

Predictor and penalty matrices per partition vector

For each fixed partition vector u, a matrix of predictors X^u is obtained by vectorization (stacking column by column) and transposition of the contingency tables:

$$X^u_i = {}^t Vect(C^u_i),\ X^u_i \in \mathbb{R}^{n_{\mathcal{A}} n_{\mathcal{B}}}. \quad (3)$$

Each row X^u_i of X^u contains all the information relative to one individual and each column those relative to one joint modality. X^u has n rows and $n_{\mathcal{A}} \times n_{\mathcal{B}}$ columns. For each partition vector u, X^u is called a candidate matrix. To this matrix, we add the information

about the contiguity constraints between the joint modalities by creating a graph $G^u(V^u, E^u)$ where V^u represents the columns (new variables) of the candidate matrix X^u and E^u the set of edges connecting two close joint modalities. The definition proposed by default in the **SpiceFP** package is that two joint modalities are said to be close if the classes following the variable $\mathcal{A}$ (indexed by v) or (exclusive) the classes following the variable $\mathcal{B}$ (indexed by w) are consecutive. The package offers the user the possibility to define the proximity between the joint modalities in a different way.

2.2. Models and estimation

The model under SpiceFP is defined, for each partition u and each individual i, by:

$$y_i = X_i^u \beta^u + \varepsilon_i, \tag{4}$$

where X_i^u is given in Equation (3), $\beta^u_{(v,w)}$ is the coefficient to be estimated on the 2D interval $(I^u_{\mathcal{A}}(v) \times I^u_{\mathcal{B}}(w))$ and ε_i is an i.i.d. Gaussian error. Investigating the suitability of a partition vector u requires the construction of a matrix of predictors X^u associated to a graph G^u, the estimation of the coefficients in (4) and finally the evaluation of the goodness of fit of the model related to u.

Set of candidate matrices to be evaluated

There will be as many candidate matrices as partition vectors to investigate. The set of all partition vectors is defined from the set of possible values of each parameter involved in u ($n_{\mathcal{A}}$ and $n_{\mathcal{B}}$ in the default case, as discussed in the present section). For example, if we set 15 possible values for $n_{\mathcal{A}}$ and 20 values for $n_{\mathcal{B}}$, we will have 300 partition vectors to investigate or 300 candidate matrices.

Fitting of all candidate models

The estimation issue here is similar to a well-known topic in statistics called the scalar-on-image regression Kang, Reich, and Staicu (2018); Li, Zhang, Wang, Gonzalez, Maresh, and Coan (2015). The objective is to control the smoothness of the non-zero estimated coefficients. In the present case, the "image" (2D image) is a bivariate representation (indexed by v and w) of both functional variables. To estimate coefficients of scalar-on-image regression, you can choose among Bayesian approaches Goldsmith, Huang, and Crainiceanu (2014), total variation penalizing approaches Wang, Zhu, and for the Alzheimer's Disease Neuroimaging Initiative (2017) or approaches that take L_1 regularization or neighborhood into account, as Li, Sun, Deng, Zhang, Wang, and Jin (2020). We selected the Generalized Fused Lasso (GFL) Xin, Kawahara, Wang, and Gao (2014) which promotes smoothness and sparsity over neighboring variables by using constraints on the parameter differences. In SpiceFP, the coefficients of this Generalized Fused Lasso are estimated using the framework of the generalized Lasso model, introduced by Tibshirani and Taylor (2011) as an encapsulation of statistical models using the L_1 norm to impose additional constraints. This estimator is as follows, for a fixed partition vector u:

$$\widehat{\beta}^{u,\gamma}(\lambda) = \underset{\beta \in \mathbb{R}^{n_{\mathcal{A}}\, n_{\mathcal{B}}}}{\operatorname{argmin}} \frac{1}{2}||y - X^u\beta||_2^2 + \lambda||D^{u,\gamma}\beta||_1, \tag{5}$$

where :

- $y = {}^t(y_1, y_2, \ldots, y_n) \in \mathbb{R}^n$ is the response vector.
- $X^u \in \mathbb{R}^{n \times n_{\mathcal{A}} n_{\mathcal{B}}}$ is the matrix of predictors defined in Equation (3).
- $\lambda \in \mathbb{R}$ is a penalty parameter that controls the smoothness of the coefficients.
- $\gamma \in \mathbb{R}^+$ is a ratio that controls the sparsity among the coefficients. If $\gamma = 0$, there is no sparsity i.e pure fusion of the coefficients. The higher its value, the more parsimonious the model is.
- $\widehat{\beta}^{u,\gamma}(\lambda) \in \mathbb{R}^{n_{\mathcal{A}} n_{\mathcal{B}}}$ is the vector of estimated coefficients for fixed values of u, γ and λ.
- $D^{u,\gamma} = \begin{pmatrix} D^{u,f1} \\ D^{u,f2} \\ D^{u,\gamma,p} \end{pmatrix} \in \mathbb{R}^{(3n_{\mathcal{A}} n_{\mathcal{B}} - n_{\mathcal{A}} - n_{\mathcal{B}}) \times n_{\mathcal{A}} n_{\mathcal{B}}}$ is a specified penalty matrix for fixed values of u and γ with: $((v,w)(v',w')$ defines one of the edge in E^u for the graph $G^u)$

$$
\begin{aligned}
D^{u,f1}_{(v,w)(v',w')} &= \begin{cases} 1 & \text{if} \quad (v',w') = (v+1,w) \\ -1 & \text{if} \quad (v',w') = (v,w) \text{ and } v < n_{\mathcal{A}} \\ 0 & \text{if} \quad \text{not} \end{cases} \\
D^{u,f2}_{(v,w)(v',w')} &= \begin{cases} 1 & \text{if} \quad (v',w') = (v,w+1) \\ -1 & \text{if} \quad (v',w') = (v,w) \text{ and } w < n_{\mathcal{B}} \\ 0 & \text{if} \quad \text{not} \end{cases} \\
D^{u,\gamma,p} &= \gamma . \mathbb{I}_{n_{\mathcal{A}} . n_{\mathcal{B}}}
\end{aligned}
\tag{6}
$$

where $\gamma \geq 0$ and $\mathbb{I}_{n_{\mathcal{A}} . n_{\mathcal{B}}}$ is the identity matrix. $D^{u,f1}$ and $D^{u,f2}$ are relative to the smoothness of the coefficients following both functional variables and $D^{u,\gamma,p}$ is relative to sparsity among them.

2.3. Selection among the candidate models

Let's n_u be the number of candidate matrices, Γ the set of γ values, and n_λ the number of λ values. We are thus interested in the best models obtained from the $n_u \times Card(\Gamma) \times n_\lambda$ estimated models through a model selection procedure. Table 1 presents three criteria that are provided in the package. Once the criteria are computed, the best candidate matrix $X^{\widehat{u}}$ according to a specified criterion is the one involved in the best model identified by this criterion.

These criteria require the computation of degrees of freedom. A theorem is proposed by Tibshirani and Taylor (2012) to compute the degrees of freedom in Lasso problems. The computation of the degree of freedom for the Generalized Fused Lasso is presented in Gnanguenon Guesse *et al.* (202X). Its value is equivalent to the number of sets of indexes of non zero $\widehat{\beta}^{u,\gamma}_{v,w}$ that are linked together via the $D^{u,\gamma}$ matrix (6) and that all share the same real value. We will

Table 1: **Criteria used to select a model.**
Those criteria needs the computation of the Residual Sum of Squares $RSS = \|y - X^u\widehat{\beta}^{u,\gamma}(\lambda)\|_2^2$, where X^u is a candidate matrix, $\widehat{\beta}^{u,\gamma}(\lambda)$ its estimated coefficients, df the degree of freedom and where σ^2 is assumed to be the known variance of y.

Criteria	Formula
Akaike Information Criterion Akaike (1973)	$AIC = n\ \log(2\pi\sigma^2) + \frac{RSS}{\sigma^2} + 2\text{df}$
Bayesian Information Criterion Schwarz (1978)	$BIC = n\ \log(2\pi\sigma^2) + \frac{RSS}{\sigma^2} + \log(n)\text{df}$
Mallows's C_p Mallows (1973)	$C_p = RSS + 2\sigma^2\text{df}$

refer to these sets as connected components. In other words, a connected component is a set of joint modalities linked by $D^{u,\gamma}$ and having a common influence on the response variable.

The criteria presented in Table 1 also require the knowledge of σ^2, the variance of y. Since the variance must remain the same for all the models to be compared, we decided to estimate it by the empirical variance of y: $\widehat{\sigma}^2 = \frac{1}{n-1}||y - \overline{y}||_2^2$. It's a biased estimator of σ^2, but this bias remains fixed for all models compared Hirose, Tateishi, and Konishi (2013a). Such an estimator may lead to an overestimation of the variance, which penalizes the introduction of new coefficients in the model. This bias can be partly offset by an iterative approach. Therefore, we set up an iterative extension where the residuals are used as a new response variable and so on, until stopping conditions are verified.

2.4. An optional iterative extension

To capture all potential non-zero coefficients, one should take a fine partition with values of $n_{\mathcal{A}}$ and $n_{\mathcal{B}}$ large enough. This could imply a very low or even zero number of points in the joint class intervals (making the method ineffective) and prohibitively long computation times Mairal and Yu (2012). As a trade off between thinness and work-ability we chose to develop an iterative approach to explore a large space of solutions (that allows addition of different thinness of partition).

After identifying $X^{\widehat{u}}$ at one iteration, the residuals of the best model according to the same criterion may be computed and used as response variable at a next iteration. The iterative process is stopped when the vector of estimated coefficients is the null vector, or when the maximum number K of iterations is reached. The final model is the sum of all the models estimated at each iteration.

2.5. Sum up

The figure 2 illustrates the overall approach.

The input data are the functional explanatory variables $\mathcal{A}_i$ and $\mathcal{B}_i$ discretized on a grid T and a response variable y_i with $i = 1, \ldots, n$. Other elements are also required: Γ, a set of positive reals representing γ ratios of regularization parameters, $\mathcal{U}_{\mathcal{A}}$ and $\mathcal{U}_{\mathcal{B}}$ the sets of numbers of class intervals n_A and n_B, n_λ the selected number of pairs (among N_λ) $(\lambda, \widehat{\beta}^{u,\gamma}(\lambda))$, the information criterion to be used, and K the maximum number of iterations to explore. The

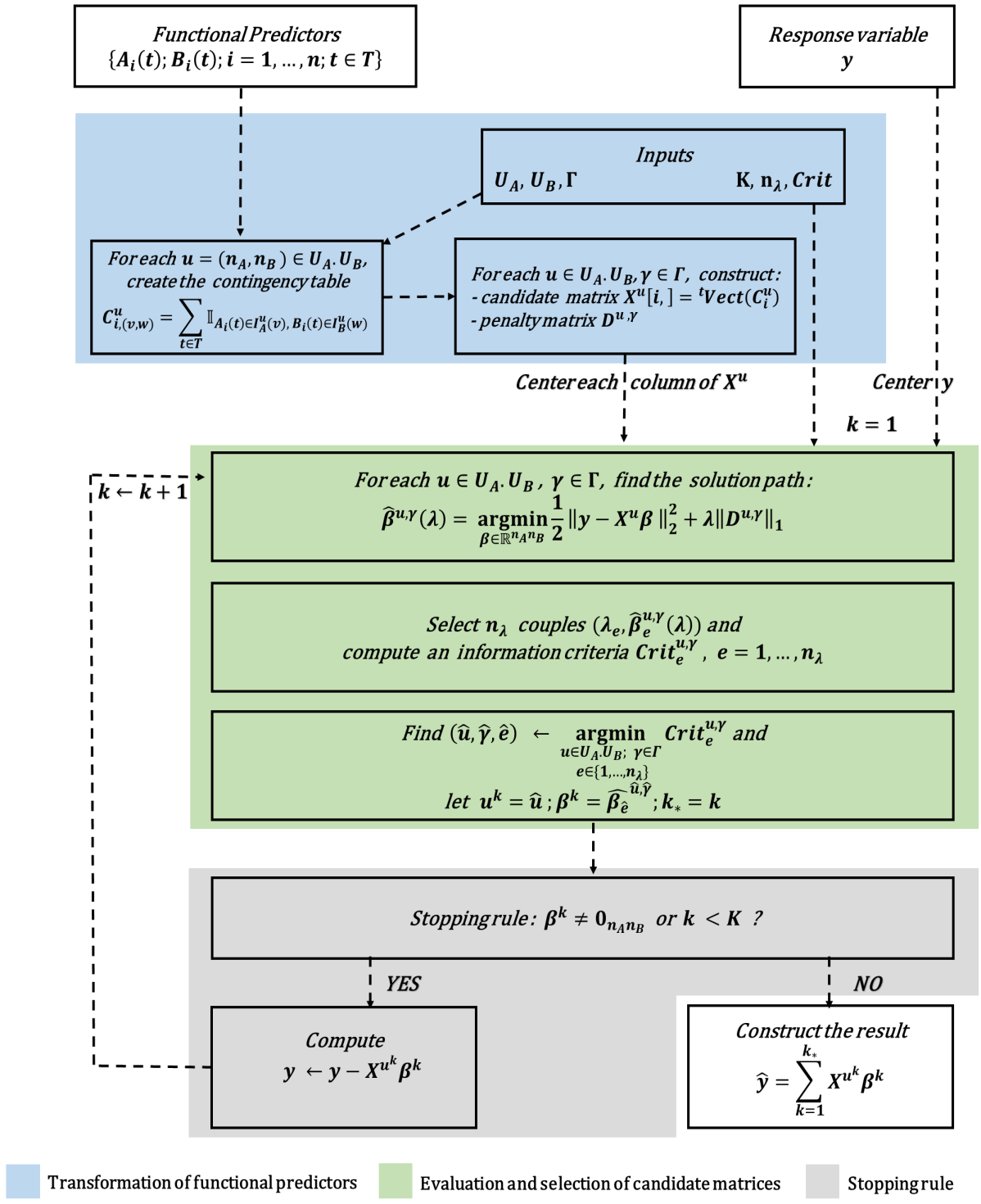

Figure 2: Summary diagram of the SpiceFP approach.

n_λ values of λ are chosen equally spaced on the log scale (see Genlasso package Arnold and Tibshirani (2019)).

SpiceFP constructs for each couple (u, γ), a matrix of explanatory variables X^u and a penalty matrix $D^{u,\gamma}$. This first step is performed only once and the set of candidate explanatory matrices remains unchanged. For each candidate matrix X^u a solution path $(\lambda, \widehat{\beta}^{u,\gamma}(\lambda))$ is obtained from the second step of the algorithm. Once the criterion $Crit_e^{u,\gamma}$ associated with each model is computed, the optimal triplet $(\widehat{u}, \widehat{\gamma}, \widehat{e})$ is the argument that minimizes $Crit_e^{u,\gamma}$, where e is the index of the solution path-coefficients $(\lambda_e, \hat{\beta}_e^{u,\gamma})$. $X^{\widehat{u}}$, $D^{\widehat{u},\widehat{\gamma}}$ and $\widehat{\beta}_{\widehat{e}}^{\widehat{u},\widehat{\gamma}}$ respectively represent the optimal matrices of the explanatory, penalty and coefficient variables.

At each iteration k, we denote $u^k = \widehat{u}$ and $\beta^k = \widehat{\beta}_{\widehat{e}}^{\widehat{u},\widehat{\gamma}}$. SpiceFP then checks if the selected coefficients according to the criterion $Crit_e^{u,\gamma}$ correspond to a zero vector or if the maximum number of iterations K is reached. The algorithm is stopped when at least one of these conditions is verified. When none of these conditions are verified, the residuals of the optimal model are computed and used as a response variable at the next iteration of the algorithm.

The final prediction is the sum of all the predictions obtained at each iteration: $\widehat{y} = \sum_{k=1}^{k_*} X^{u^k} \beta^k$, where k_* is the final number of iterations.

We remark that each vector β^k at each iteration may have different length $dim(u^k)$.

3. Implementation in R

The SpiceFP implementation in R follows the S3 methods. The main function `spicefp` allows to execute all the approach and returns estimated coefficients which are visualized as a 2D image. An extension including a third explanatory variable (combination of 3 class intervals) is also provided. The package allows the user to define his own partition of variables. The `meancoef` function enables to extend the approach while using coefficients estimated in the framework of **SpiceFP**. The user can access various functions required to achieve some of the approach's steps such as `candidates`, `evaluate.candidates` or `coef_spicefp` functions. Data are also provided in the package to test the functions.

3.1. Transformation of functional predictors: the "candidates" function

There is two main ways to construct candidate matrices. One is based on the framework defined by the **SpiceFP**'s functions and the other allows the user to construct the matrices as he wishes, before submitting them to the `spicefp` function via the `candmatrices` argument. These two options are detailed hereafter.

Construction of candidate matrices in the package framework

The **SpiceFP** package constructs candidate matrices through the function `candidates`. Its philosophy is to return matrices of new predictors with in columns combinations of two (three) class intervals related to two (three) functional predictors. To achieve this goal, the function requires as inputs each functional predictor, an R function that will be used to partition this functional predictor and arguments of this function. The other requirements inform about the number of cores to use (parallelization with **doParallel** package (Microsoft-Corporation and Weston 2019)) and define whether or not the columns of candidate matrices (joint modalities) should be centered or scaled. Each functional predictor is presented as one numerical matrix

with in columns observations of one statistical individual. Due to the fact that the functional predictors should be observed with the same set T of time steps, the numerical matrices contain $Card(T)$ rows and each row is relative to the observation variable $t \in T$. Statistical individuals keep the same order through the matrices to be provided for `fp1`, `fp2` or `fp3` arguments.

In order to partition the observations of the functional predictors, the functions (`fun1`, `fun2` or `fun3`) must comply with certain constraints. The only output of these functions must be a numerical vector of breaks that will be used to generate the class intervals. They must also consider at least two separate arguments. The first argument is a numerical vector (the vector to be partitioned). The second is a list of partitioning parameters to be optimized by the approach. The parameters to be optimized are the components of the partition vector u. The parameters that do not need to be optimized can be defined as additional arguments and set by default. To illustrate, let's look at two partition functions. The first one, `linbreaks`, presented below, allows to partition a numerical vector according to a linear scale. Its use in the construction of the `example1` vector allows the identification of 13 breaks that will be used to formulate 12 consecutive class intervals.

```
R> linbreaks <- function(x,n){
+     round(seq(trunc(min(x)),
+                     ceiling(max(x)),
+                     length.out = unlist(n)+1),
+                 1)
+ }
R> set.seed(284)
R> example1 <- linbreaks(rpois(1000,100), list(12))
R> example1

 [1]  67.0  72.2  77.5  82.8  88.0  93.2  98.5 103.8 109.0 114.2 119.5 124.8
[13] 130.0
```

For the second example, we focus on the `logbreaks` function in the **SpiceFP** package which allows to get breaks according to a logarithmic scale. The reader can refer to the help page of this function for more details. To compute the breaks, `logbreaks` requires two essential parameters (`alpha` and `J`) provided by the argument `parlist` as well as other additional arguments. One way to use `logbreaks` in the **SpiceFP** approach is to create a new function `Logbreaks` setting the additional arguments by default as presented below. This framework therefore offers the user a high degree of flexibility in the way functional predictors are partitioned.

```
R> Logbreaks <- function(x, Parameterlist){
+   logbreaks(x, Parameterlist,  round_breaks = 1, plot_breaks = FALSE,
+             effect.threshold.begin = NA,   effect.threshold.end = NA)
+ }
R> set.seed(284)
R> example2 <- Logbreaks(rpois(1000,100), list(0.05,12))
R> example2
```

```
 [1]  67.0  69.5  72.4  75.5  79.1  83.2  87.7  92.9  98.6 105.2 112.5 120.7
[13] 130.0
```

Once functional predictors and partition functions are defined, another important argument of `candidate` function is `parlists`. It is a list. Its length equals the number of functional predictors involved in the model (i.e. 2 or 3). Each element of this list is related to one functional predictor and its partition function. The length of each of these elements is exactly the number of candidate matrices to create. More precisely, each element in `parlists` represents a list with all the second arguments related to one partition function that will help to create various candidate matrices. In order to illustrate this, let us consider the creation of the argument `parlists` required to transform two functional predictors `var1` and `var2` into a set of candidate matrices. Suppose that one parameter is needed to partition `var1` namely `var1.nclass` (4 values are possible: 10, 12, 14, 16) and two parameters for `var2` : `var2.nclass` (3 possible values: 20, 22, 24) and `var2.alpha` (3 possible values: 0.01, 0.02, 0.03). To construct $4 \times 3 \times 3 = 36$ candidate matrices covering the different combinations of parameters, `parlists` can be constructed using the following commands :

```
R> var1.nclass <- c(10, 12, 14, 16)
R> var2.nclass <- c(20, 22, 24)
R> var2.alpha <- c(0.01, 0.02, 0.03)
R> p2 <- expand.grid(var1.nclass, var2.alpha, var2.nclass)
R> parlist.var1 <- split(p2[,1], seq(nrow(p2)))
R> parlist.var1 <- lapply(parlist.var1, function(x){list(x[[1]])})
R> parlist.var2 <- split(p2[,2:3], seq(nrow(p2)))
R> parlist.var2 <- lapply(parlist.var2, function(x){list(x[[1]],x[[2]])})
R> parlists <- list(parlist.var1,  parlist.var2)
R> length(parlists[[1]])

[1] 36
```

Then, the `candidates` function uses the partition function related to each functional predictor to compute the breaks that should be used to construct its class intervals. These breaks together with the observations of the functional variables ($\mathcal{A}_i$, $\mathcal{B}_i$, etc.), are the required inputs of the functions `hist_2d` or `hist_3d`, used to construct, for each individual i, histograms or contingency tables C_i^u. Each of these n contingency tables will be transformed into X_i^u, as presented in the equation (3). All these matrices of explanatory variables X^u are saved in a list with the same length as each element of `parlists`. Attention was also paid to associate with each of these matrices, a numerical vector giving information on the index of the associated vector u and the number of class intervals of each functional predictor in the order `fp1`, `fp2` (and `fp3`, if three functional predictors are used). These numbers of class intervals will be used to generate the penalty matrix $D^{u,\gamma}$.

Construction of candidate matrices independently of the package framework

There is one constraint to respect in the construction of the candidate matrices that will be provided directly as inputs to the function `spicefp`: the organization of these candidate matrices as well as their presentation must be similar to that provided by the function

`candidates`. TIn order to remove this constraint, we present here in details the `candidates` function output. This output is a list of eleven named elements respectively and organized as follows: to help complying with this constraint, we present here in details the `candidates` function output. This output is a list of eleven elements, named and organized as follows:

- `spicefp.dimension` : scalar number equal to 2 or 3, which represents the number of class intervals that compose a joint modality (in column of the X^u matrix).

- `candidates` : list in which each element is a list containing a matrix X^u and a vector Z^u, both relative to the same candidate. The matrix, the first element of the list, always contains n rows. The number of columns varies according to the candidates. The columns are named by the combination of class intervals separated by the symbol "underscore (_)". As example, for the two functional variables $\mathcal{A}$ and $\mathcal{B}$, we get $[L_{\mathcal{A}}(v), L_{\mathcal{A}}(v+1)[\_[L_{\mathcal{B}}(w), L_{\mathcal{B}}(w+1)[,\ v = 1, \ldots, n_{\mathcal{A}},\ w = 1, \ldots, n_{\mathcal{B}}$. The order of the class intervals in editing the combination is important. Here, $\mathcal{A}$ is considered the first and $\mathcal{B}$ the second. This order appears in the construction of the vector Z^u ($Z^u = c(\text{match}(u, U), n_{\mathcal{A}}, n_{\mathcal{B}})$). Its first element is an identification key that allows to associate the X^u matrix with the parameters used to create it. The rest of the elements Z^u are the numbers of class intervals created by functional predictor arranged in the same order.

- `fp1, fp2, fp3, fun1, fun2, fun3, parlists, xcentering, xscaling` : these terms respectively represent the functional predictors, the associated partition functions, the partition vectors and the logical parameters that indicate whether the candidate columns should be centered or scaled. They correspond to the elements used or not in the construction of the candidates. The user can assign them the `NULL` value but he must provide a value for each term.

3.2. Evaluation of candidate models by generalized fused lasso: the "evaluate.candidates" function

After constructing the candidate matrices, the second step of the approach is implemented by the `evaluate.candidates` function. This function mainly returns a matrix of information criteria values, with in rows the models and in columns the associated parameters. This function consists in estimating the parameters $(\lambda, \widehat{\beta}^{u,\gamma}(\lambda))$ of equation (5) and computing, for each model, different information criteria Hirose, Tateishi, and Konishi (2013b). To achieve this, we start by defining a set Γ containing all the γ values. For each candidate X^u, we construct $Card(\Gamma)$ penalty matrices $D^{u,\gamma}$. Each $D^{u,\gamma}$ is generated using the argument `penfun` of `evaluate.candidates`. `penfun` is a function taking as inputs all elements of the Z^u vector except the first one. By default, it takes the `NULL` value. In this case, either the `getD2dSparse` function of the **genlasso** package Arnold and Tibshirani (2020) (when `spicefp.dimension` = 2) or the `getD3dSparse` function of the **SpiceFP** package (when `spicefp.dimension` = 3) are used. `getD2dSparse` allows to define a rook's case contiguity between the joint modalities. `getD3dSparse` is an extension of `getD2dSparse` to a third dimension. More precisely, it is the part of $D^{u,\gamma}$ (see equation (6)) penalizing the fusion of the coefficients that is constructed by `penfun`. It can be noticed that γ is not part of its arguments. The creation of $D^{u,\gamma,p}$, the part

Table 2: Statistics used to summarize the information associated with a model involving a response variable y, a candidate matrix X^u, estimated coefficients $\beta^{u,\gamma}(\lambda)$, degree of freedom df and σ^2 the variance of y.

Statistics	Formula
Residual Sum of Squares	$RSS = \|y - X^u\beta^{u,\gamma}(\lambda)\|_2^2$
Generalized Cross Validation Craven and Wahba (1979)	$GCV = \frac{1}{n}\frac{RSS}{(1-\mathrm{df}/n)^2}$
Slope of regression $\widehat{y} \sim y$; $(\widehat{y} = X^u\beta^{u,\gamma}(\lambda))$	$Slope = ({}^t yy)^{-1}\ {}^t y\widehat{y}$
Variance ratio of $\widehat{y}$ and y	$R_{var} = \sigma^2_{\widehat{y}}/\sigma^2$

that penalizes parsimony, is automatically done in the next step by the `fusedlasso` function (**genlasso**).

In statistics, they are many fast algorithms for solving linear regression with l_1 penalty at a single value of the tuning parameter λ. But when the solution is requested for many values of the tuning parameter, the least-angle regression algorithm (LARS) Efron, Hastie, Johnstone, and Tibshirani (2004) provides a computational advantage in the sense that it help to solve linear regression with l_1 penalty for all $\lambda \in [0, \infty[$. The algorithm in **genlasso** is derived from the LARS path algorithm in order to solve problems that use the l_1 norm to enforce structural constraints instead of pure sparsity on the coefficients in a linear regression Tibshirani and Taylor (2011). Since **SpiceFP** has to estimate a tuning parameter as well as coefficients for different candidate matrices, this computational advantage is welcome. The **genlasso** package is designed to compute the solution path of the generalized lasso problem, which minimizes a criterion similar to the one presented in equation (5). This solution path is computed by solving the equivalent Lagrange dual problem and its implementation is presented in Arnold and Tibshirani (2016). For a candidate X^u and a penalty matrix $D^{u,\gamma}$, `fusedlasso` returns N_λ couples $(\lambda, \widehat{\beta}^{u,\gamma}(\lambda))$. The values of lambda in these couples are those at which the solution path changes slope. By default, $N_\lambda = 2000$ when using **genlasso** Arnold and Tibshirani (2020). But we will only work with $n_\lambda < N_\lambda$ pairs chosen according to a logarithmic scale in the path solution (argument `nknots`) or from an *a priori* knowledge of the number of expected areas of influence (argument `appropriate.df`).

3.3. Post-evaluation treatment and result construction

The **SpiceFP** package returns areas of influence of non null coefficients.

`spicefp` is the main function of the **SpiceFP** package. It allows to implement the SpiceFP approach from already constructed candidate matrices to be provided as inputs (argument `candmatrices`) or candidate matrices to be constructed via functional predictors (`fp1, fp2, fp3`) and associated categorization elements (`fun1, fun2, fun3, parlists`). In this latter case, the function `candidates` is used to construct these matrices. At each iteration, the candidates are evaluated and the best model is selected via the AIC or the BIC. As a result, vectors of coefficients of different lengths (i.e. from different meshes) can be selected at the various iterations. This situation does not pose any difficulty in estimating the predicted

values $\widehat{y}$. It is obtained by summing the products $X^u\beta^{u,\gamma}(\lambda)$ of the models retained at each iteration. In order to visualize the identified areas of influence, it is essential to be able to sum the coefficients retained at each iteration. The functions `finemeshed2d` and `finemeshed3d` respectively allow to transform a vector of coefficients (named with combinations of classes) into a 2 and 3 dimensional table with an extremely fine mesh. Once the same fine mesh is used to transform all the coefficient vectors, it is easy to sum up them and to visualize the areas of influence. This process allows to approximate the area of coefficients (continuous) from discrete estimates. All these procedures are already implemented in the function `spicefp`.

The function `coef_spicefp`, provides from the outputs of `spicefp`, the possibility to reconstruct any model evaluated during the approach, regardless of the iteration. The candidate matrices as well as the result of their evaluation, the coefficients retained at each iteration, and the sum of the fine meshes associated with these coefficients are the main outputs of `spicefp`.

Remark: It is possible to obtain other results from the `spicefp` outputs. Instead of successive iterations, we can focus only on the first iteration and make an average of the best models obtained at this iteration. The number of best models to be averaged has to be defined by the user. In the simulations presented in (Gnanguenon Guesse *et al.* 202X), this average of coefficients performs slightly less than the approach itself, but has the advantage of reconstructing the contours of the areas of influence much more faithfully. This model average can be implemented by the function `meancoef` of the package **SpiceFP**.

4. Example

In this section, we are interested in exploring the joint influence of two functional variables (`Temperature` and `Irradiance`) on a response variable (`FerariIndex_Difference`) giving information on the quality of the grape berry. The data are available and described in the package **SpiceFP**. The hypothesis of a joint influence is made because both temperature and irradiance are affected by solar radiation. But before exploring this joint influence, let us first look at each of the functional predictors in order to better partition them. The distribution of the Temperature values (see figure 3) suggests that a linear scale is suitable for its partitioning. If this scale was used for Irradiance, the majority of observations would be distributed in very few classes.

We then propose to use a logarithmic scale to partition the Irradiance values. This scale has the advantage of expanding the low values and compressing the high ones. The function `logbreaks` of **SpiceFP** allows to obtain such results. It is necessary to provide the argument of this function: the parameter α. For a fixed number of breaks, this parameter controls the proportion of breaks chosen from the low values. By visualizing some Irradiance distributions made from a logarithmic scale (figure 4), the user can get an idea of the values to scan to build the different candidate matrices.

Once all this information is available, we decide to build candidate matrices with the number of Temperature classes (linear scale) varying between 10 and 18 and the number of Irradiance classes (logarithmic scale) between 15 and 25. The values of the parameter `alpha`, necessary for the logarithmic scale will vary between e^{-5} and e^{-1}. At this step, the user can visualize, with the function `hist_2d` of **SpiceFP**, a distribution of the observations in two dimensions (Figure 5). This provides insight into areas of relative importance, if any.

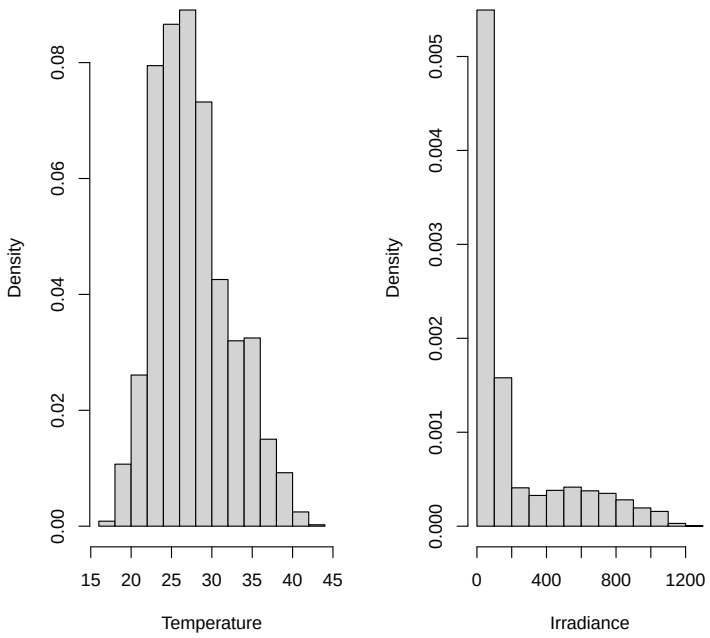

Figure 3: Distribution of Temperature and Irradiance values according to a linear scale

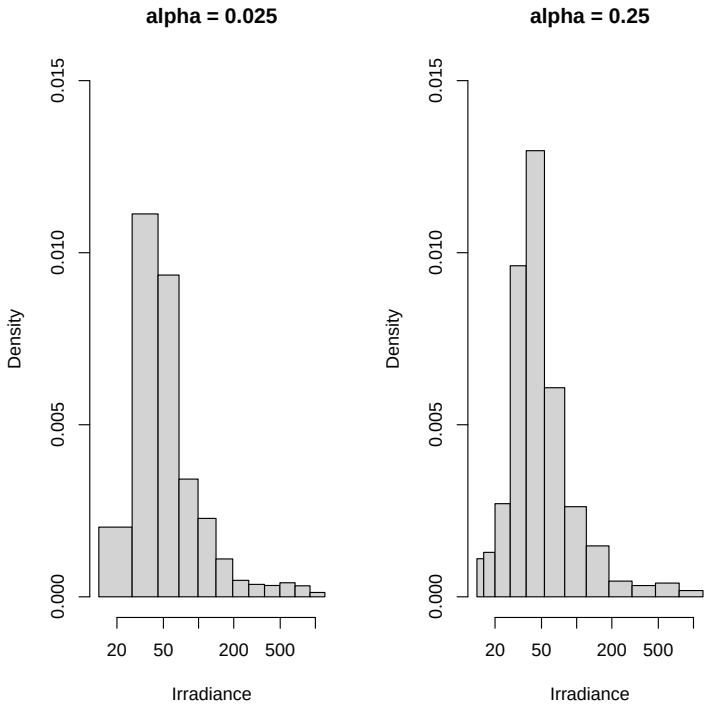

Figure 4: Distribution of Irradiance values according to a logarithmic scale

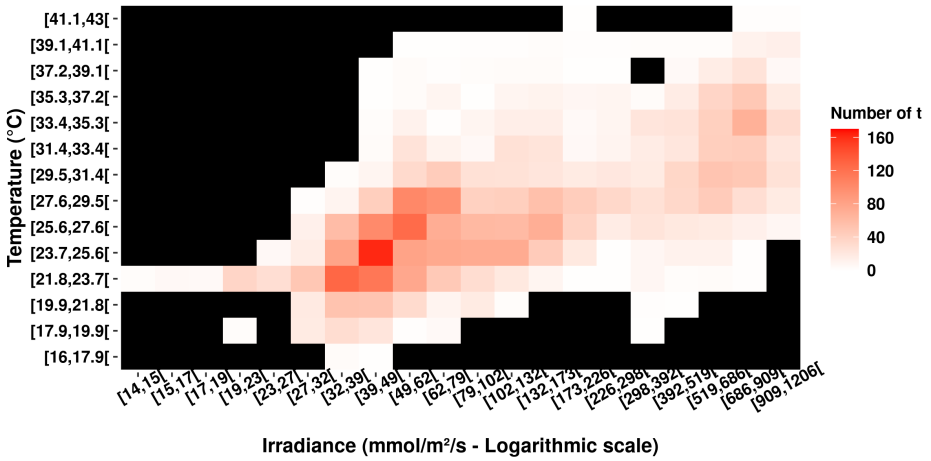

Figure 5: Candidate matrix with 14 Temperature classes on linear scale and 12 Irradiance classes on log scale. ■: Joint modalities that have never been observed.

In this example, the construction of 120 candidate matrices allowed us to roughly browse the value ranges of the three parameters constituting the partition vector u. The user should keep in mind that the execution time of the approach is correlated to the number of candidate matrices to be evaluated, as well as to the complexity of the links between the joint modalities of the candidate matrices. With respect to the γ ratio between parsimony and fusion regulation parameters, 6 ratios were used. The approach can be performed using the following commands :

```
R> ## Data and inputs
R> tpr.nclass=seq(10,18,2)
R> irdc.nclass=seq(15,25,3)
R> irdc.alpha=round(exp(seq(-5,-1,length.out=6 )),4)
R> p2<-expand.grid(tpr.nclass, irdc.alpha, irdc.nclass)
R> parlist.tpr<-split(p2[,1], seq(nrow(p2)))
R> parlist.irdc<-split(p2[,2:3], seq(nrow(p2)))
R> parlist.irdc<-lapply(
+    parlist.irdc,function(x){
+    list(x[[1]],x[[2]])}
+ )
R> start_time_sp <- Sys.time()
R> ex_sp<-spicefp(y = FerariIndex_Difference$fi_dif,
+                 fp1 = m.irdc, fun1 = logbreaks,
+                 fp2 = m.tpr,  fun2 = linbreaks,
+                 parlists = list(parlist.irdc, parlist.tpr),
+                 penratios = c(1/100, 1/10, 1, 10),
+                 K = 2, criterion = "AIC_",
+                 nknots = 100, ncores = 4,
+                 dim.finemesh = c(1000, 1000),
+                 write.external.file = TRUE)
```

```
R> duration_sp <- Sys.time() - start_time_sp
```

About one hour and fifty minutes were needed to perform the calculations (2 iterations) using 4 cores (Intel Core i7-7600U CPU, 2.80GHz × 4). The argument `write.external.file = TRUE` allowed the library to generate as an external file (available in the working directory) the summary table of the model evaluation at each iteration. The final result is shown in figure 6.

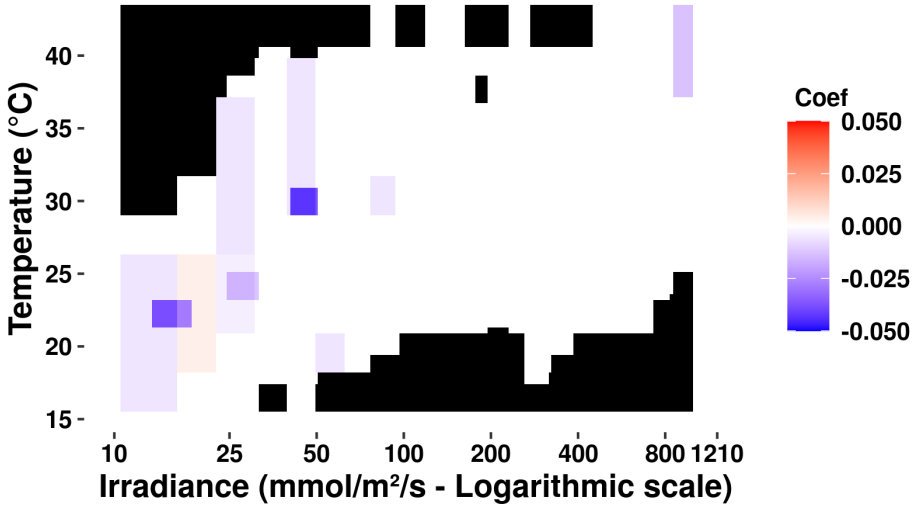

Figure 6: Visualization of the **SpiceFP** result (2 iterations, AIC criterion). ■: Joint modalities that have never been observed.

From the figure 6, one can observe the joint modalities for which no observation is made (value :NA, color black). This makes it possible to differentiate between unobserved and observed but non-influential areas. As an interpretation, we can retain that for low irradiance values ($< 100\ mmol.m^{-2}.s^{-1}$), there is a gradient according to temperature which is not suitable for an increase of the Ferari index. All the information on the models retained in the two iterations are accessible via the commands `ex_sp$Evaluations[[1]]` and `ex_sp$Evaluations[[2]]`. Their characteristics are as follows:

```
R> # Itération 1
R> t(ex_sp$Evaluations[[1]]$thecandidate.parameters)

     Candidate_id Pen_ratio PenPar_fusion Df_ RSS_     AIC_     BIC_
[1,] 108          10        0.1648765     3   13.53096 214.5332 218.9305
     AICc_    Cp_      GCV_      Slope_    Var_ratio
[1,] 56.41074 13.86642 0.5148522 0.5650574 0.327378

R> # Itération 2
R> t(ex_sp$Evaluations[[2]]$thecandidate.parameters)

     Candidate_id Pen_ratio PenPar_fusion Df_ RSS_     AIC_     BIC_
[1,] 101          0.1       0.3617341     4   13.35619 668.4776 674.3406
```

```
     AICc_    Cp_      GCV_      Slope_    Var_ratio
[1,] 53.37039 13.50262 0.5451506 0.2108834 0.6919936
```

An estimate of the goodness of fit of the final estimate can be obtained: *i)* statistically by computing the correlation between the response variable y and the predictions $\widehat{y}$,

```
R> xbeta <- c(ex_sp$Evaluations[[1]]$XBeta + ex_sp$Evaluations[[2]]$XBeta)
R> cor(FerariIndex_Difference$fi_dif , xbeta, method = "pearson")

[1] 0.8884187
```

ii) or graphically (figure 7) by representing y and $\widehat{y}$ in the same scatterplot.

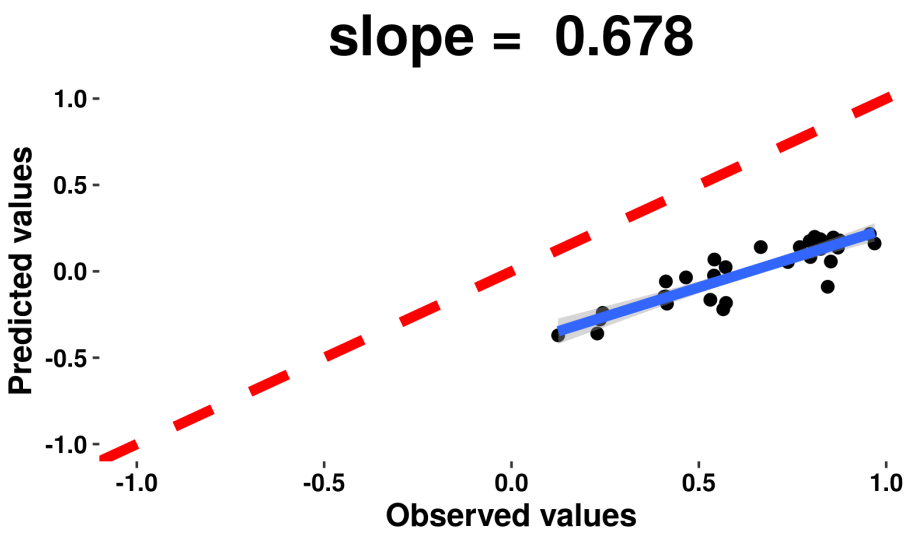

Figure 7: Quality of the SpiceFP estimation

Let's take a closer look at the top 10 models among those selected by the AIC in order to make an average.

```
R> # Models at iteration 1
R> models_iter1 <- read.table(ex_sp$Evaluations[[1]]$Evaluation.results$evaluation.result,
header = TRUE)
R> dim(models_iter1)

[1] 48000    12

R> OrderbyAIC_iter1 <- models_iter1[order(models_iter1$AIC_) ,]
R> head(OrderbyAIC_iter1, n = 10)

      Candidate_id Pen_ratio PenPar_fusion Df_     RSS_     AIC_     BIC_
43122          108     10.00     0.1648765   3 13.53096 214.5332 218.9305
43121          108     10.00     0.1846918   3 13.54702 214.8206 219.2178
```

```
43120          108     10.00     0.2068887   3 13.56718 215.1811 219.5783
43119          108     10.00     0.2317533   3 13.59248 215.6336 220.0308
43009          108      1.00     0.7389190   3 13.59921 215.7539 220.1511
43118          108     10.00     0.2596061   3 13.62422 216.2013 220.5985
40725          103     10.00     0.1380610   4 13.51566 216.2597 222.1226
15526           38     10.00     0.1519366   3 13.63613 216.4143 220.8115
8851            24      0.01     0.9533787   3 13.64396 216.5543 220.9515
15525           38     10.00     0.1697173   3 13.64786 216.6240 221.0212
         AICc_      Cp_      GCV_    Slope_ Var_ratio
43122 56.41074 13.86642 0.5148522 0.5650574 0.3273780
43121 56.44872 13.88249 0.5154635 0.5477581 0.3366471
43120 56.49630 13.90265 0.5162305 0.5283798 0.3482779
43119 56.55591 13.92795 0.5171930 0.5066726 0.3628724
43009 56.57175 13.93467 0.5174491 0.4948788 0.3667548
43118 56.63055 13.95969 0.5184008 0.4823565 0.3811858
40725 53.75021 13.96295 0.5516597 0.5671179 0.3185532
15526 56.65851 13.97160 0.5188539 0.5229218 0.3880567
8851  56.67687 13.97942 0.5191518 0.5076475 0.3925728
15525 56.68601 13.98332 0.5193001 0.5093079 0.3948225
```

The reconstruction of the models is done through the function `coef_spicefp` and the visualization of their mean is presented in figure 8.

```
R> # Estimation of coefficients
R> AICtopten_iter1 <- coef_spicefp( spicefp.result = ex_sp,
+                                   iter_ = 1,
+                                   criterion = "AIC_",
+                                   nmodels = 10,
+                                   model.parameters = NULL,
+                                   dim.finemesh = c(1000, 1000),
+                                   ncores = 4,
+                                   write.external.file = TRUE )
R> # Compute the mean of the coefficients
R> mean_AICtopten_iter1 <- meancoef(coef.list = AICtopten_iter1$coef.list ,
+                                   weight = rep(1,10))
```

The goodness of fit of the model associated with the mean of these coefficients can also be obtained by computing the correlation between y and $\widehat{y}$.

```
R> cor(FerariIndex_Difference$fi_dif , mean_AICtopten_iter1$y.estimated,
+     method = "pearson")

[1] 0.8312705
```

5. Conclusion

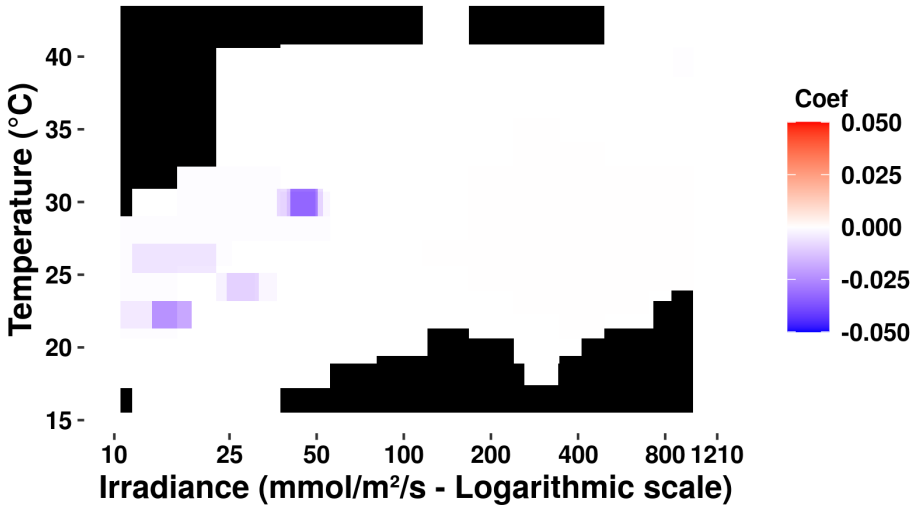

Figure 8: Visualization of coefficient mean of the 10 best models selected by the AIC at iteration 1. ■: Joint modalities that have never been observed.

The **SpiceFP** library offers various functions allowing the implementation of the exploratory approach of the same name. Although exploratory, this approach requires the use of inferential statistics tools. The functional model that supports this approach takes functional observations as predictors and a scalar variable as a response. This falls within the framework of "scalar-on-function" functional models. It implies a joint influence of the predictors and is designed for exploration in this direction. Thus, it allows the identification of areas of influence (derived from combinations of classes of predictors).

Acknowledgments

This work was supported by the French National Research Agency under the Investments for the Future Program, referred as ANR-16-CONV-0004. The data used 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.

References

Akaike H (1973). *Information Theory and an Extension of the Maximum Likelihood Principle*, pp. 199–213. Springer New York, New York, NY.

Arnold TB, Tibshirani RJ (2016). "Efficient Implementations of the Generalized Lasso Dual Path Algorithm." *Journal of Computational and Graphical Statistics*, **25**(1), 1–27. doi:10.1080/10618600.2015.1008638. https://doi.org/10.1080/10618600.2015.1008638, URL https://doi.org/10.1080/10618600.2015.1008638.

Arnold TB, Tibshirani RJ (2019). *genlasso: Path algorithm for generalized lasso problems*. Package version 1.4 for R version 3.6.1.

Arnold TB, Tibshirani RJ (2020). *genlasso: Path Algorithm for Generalized Lasso Problems*. R package version 1.5, URL https://CRAN.R-project.org/package=genlasso.

Craven P, Wahba G (1979). "Smoothing noisy data with spline functions: Estimating the correct degree of smoothing by the method of generalized cross-validation." *Numer. Math.*, **31**, 377–403.

Efron B, Hastie T, Johnstone I, Tibshirani R (2004). "Least Angle Regression" (with discussions)." *The Annals of Statistics*, **32**, 407–499.

Gertheiss J, Maity A, Staicu AM (2013). "Variable selection in generalized functional linear models." *Stat*, **2**(1), 86–101. doi:https://doi.org/10.1002/sta4.20. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/sta4.20.

Gnanguenon Guesse G, Loisel P, Fontez B, Simonneau T, Hilgert N (202X). "Identification of combined effects of functional variables using contingency tables with ordered categories - Application to agri-environmental issues." *to appear*, **XX**.

Goldsmith J, Bobb J, Crainiceanu CM, Caffo B, Reich D (2011). "Penalized Functional Regression." *Journal of Computational and Graphical Statistics*, **20**(4), 830–851. doi:10.1198/jcgs.2010.10007. PMID: 22368438, https://doi.org/10.1198/jcgs.2010.10007, URL https://doi.org/10.1198/jcgs.2010.10007.

Goldsmith J, Huang L, Crainiceanu CM (2014). "Smooth Scalar-on-Image Regression via Spatial Bayesian Variable Selection." *Journal of Computational and Graphical Statistics*, **23**(1), 46–64. doi:10.1080/10618600.2012.743437. PMID: 24729670, https://doi.org/10.1080/10618600.2012.743437, URL https://doi.org/10.1080/10618600.2012.743437.

Grollemund PM, Abraham C, Baragatti M, Pudlo P (2019). "Bayesian Functional Linear Regression with Sparse Step Functions." *Bayesian Analysis*, **14**(1), 111 – 135. doi:10.1214/18-BA1095. URL https://doi.org/10.1214/18-BA1095.

Guillas S, Lai MJ (2010). "Bivariate splines for spatial functional regression models." *Journal of Nonparametric Statistics*, **22**(4), 477–497. doi:10.1080/10485250903323180. https://doi.org/10.1080/10485250903323180, URL https://doi.org/10.1080/10485250903323180.

Hirose K, Tateishi S, Konishi S (2013a). "Tuning parameter selection in sparse regression modeling." *Computational Statistics & Data Analysis*, **59**, 28 – 40. ISSN 0167-9473. doi:https://doi.org/10.1016/j.csda.2012.10.005. URL http://www.sciencedirect.com/science/article/pii/S0167947312003556.

Hirose K, Tateishi S, Konishi S (2013b). "Tuning parameter selection in sparse regression modeling." *Computational Statistics & Data Analysis*, **59**(C), 28–40. doi:10.1016/j.csda.2012.10.00. URL https://ideas.repec.org/a/eee/csdana/v59y2013icp28-40.html.

James GM, Silverman BW (2005). "Functional Adaptive Model Estimation." *Journal of the American Statistical Association*, **100**(470), 565–576. doi:10.1198/016214504000001556. URL https://doi.org/10.1198/016214504000001556.

James GM, Wang J, Zhu J (2009). "Functional Linear Regression That's Interpretable." *The Annals of Statistics*, **37**(5a), 2083–2108.

Kang J, Reich BJ, Staicu AM (2018). "Scalar-on-image regression via the soft-thresholded Gaussian process." *Biometrika*, **105**(1), 165–184. ISSN 0006-3444. doi:10.1093/biomet/asx075. https://academic.oup.com/biomet/article-pdf/105/1/165/24331693/asx075.pdf, URL https://doi.org/10.1093/biomet/asx075.

Li F, Zhang T, Wang Q, Gonzalez MZ, Maresh EL, Coan JA (2015). "Spatial Bayesian variable selection and grouping for high-dimensional scalar-on-image regression." *Ann. Appl. Stat.*, **9**(2), 687–713. doi:10.1214/15-AOAS818. URL https://doi.org/10.1214/15-AOAS818.

Li Y, Sun H, Deng X, Zhang C, Wang HPB, Jin R (2020). "Manufacturing quality prediction using smooth spatial variable selection estimator with applications in aerosol jet® printed electronics manufacturing." *IISE Transactions*, **52**(3), 321–333. doi:10.1080/24725854.2019.1593556. https://doi.org/10.1080/24725854.2019.1593556, URL https://doi.org/10.1080/24725854.2019.1593556.

Ma W, Xiao L, Liu B, Lindquist MA (2019). "A functional mixed model for scalar on function regression with application to a functional MRI study." *Biostatistics*. ISSN 1465-4644. doi:10.1093/biostatistics/kxz046. Kxz046, https://academic.oup.com/biostatistics/advance-article-pdf/doi/10.1093/biostatistics/kxz046/30249641/kxz046.pdf, URL https://doi.org/10.1093/biostatistics/kxz046.

Mairal J, Yu B (2012). "Complexity Analysis of the Lasso Regularization Path." In J Langford, J Pineau (eds.), *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, ICML '12, pp. 353–360. Omnipress, New York, NY, USA. ISBN 978-1-4503-1285-1.

Mallows CL (1973). "Some Comments on CP." *Technometrics*, **15**(4), 661–675. ISSN 00401706. URL http://www.jstor.org/stable/1267380.

Marx BD, Eilers PHC (2005). "Multidimensional Penalized Signal Regression." *Technometrics*, **47**(1), 13–22. ISSN 00401706. URL http://www.jstor.org/stable/25470930.

Microsoft-Corporation, Weston S (2019). *doParallel: Foreach Parallel Adaptor for the 'parallel' Package*. R package version 1.0.15, URL https://CRAN.R-project.org/package=doParallel.

Ramsay J, Silverman B (2005). *Functional Data Analysis*. Springer Series in Statistics. Springer New York. ISBN 9780387227511. URL https://books.google.fr/books?id=REzuyz_V6OQC.

Reiss PT, Goldsmith J, Shang HL, Ogden RT (2017). "Methods for Scalar-on-Function Regression." *International Statistical Review*, **85**(2), 228–249. doi:10.1111/insr.12163. https://onlinelibrary.wiley.com/doi/pdf/10.1111/insr.12163, URL https://onlinelibrary.wiley.com/doi/abs/10.1111/insr.12163.

Schwarz G (1978). "Estimating the Dimension of a Model." *Ann. Statist.*, **6**(2), 461–464. doi:10.1214/aos/1176344136. URL https://doi.org/10.1214/aos/1176344136.

Tibshirani RJ, Taylor J (2011). "The solution path of the generalized Lasso." *The Annals of Statistics*, **39**(3), 1335–1371. ISSN 00905364. URL http://www.jstor.org/stable/23033600.

Tibshirani RJ, Taylor J (2012). "Degrees of freedom in lasso problems." *Ann. Statist.*, **40**(2), 1198–1232. doi:10.1214/12-AOS1003. URL https://doi.org/10.1214/12-AOS1003.

Usset J, Staicu AM, Maity A (2016). "Interaction models for functional regression." *Computational Statistics and Data Analysis*, **94**, 317–329. ISSN 0167-9473. doi:https://doi.org/10.1016/j.csda.2015.08.020. URL https://www.sciencedirect.com/science/article/pii/S016794731500208X.

Wang X, Zhu H, for the Alzheimer's Disease Neuroimaging Initiative (2017). "Generalized Scalar-on-Image Regression Models via Total Variation." *Journal of the American Statistical Association*, **112:519**, 1156–1168. doi:DOI:10.1080/01621459.2016.1194846.

Xin B, Kawahara Y, Wang Y, Gao W (2014). "Efficient generalized fused lasso and its application to the diagnosis of Alzheimer's disease." *Proceedings of the National Conference on Artificial Intelligence*, **3**, 2163–2169.

Affiliation:

Girault Gnanguenon Guesse
Université Montpellier
MISTEA, Université Montpellier, INRAE, Institut Agro, Montpellier, France
E-mail: girault.gnanguenon@gmail.com

Patrice Loisel
Université Montpellier
MISTEA, Université Montpellier, INRAE, Institut Agro, Montpellier, France
E-mail: patrice.loisel@inrae.fr

Bénedicte Fontez
Université Montpellier
MISTEA, Université Montpellier, INRAE, Institut Agro, Montpellier, France
E-mail: benedicte.fontez@supagro.fr

Thierry Simonneau
Université Montpellier
LEPSE, Université Montpellier, INRAE, Institut Agro, Montpellier, France
E-mail: thierry.simonneau@inrae.fr

Nadine Hilgert
Université Montpellier
MISTEA, Université Montpellier, INRAE, Institut Agro, Montpellier, France
E-mail: nadine.hilgert@inrae.fr

Isabelle Sanchez
Université Montpellier
MISTEA, Université Montpellier, INRAE, Institut Agro, Montpellier, France
E-mail: isabelle.sanchez@inrae.fr