KSPM: A Package For Kernel Semi-Parametric Models

by Catherine Schramm, Sébastien Jacquemont, Karim Oualkacha, Aurélie Labbe and Celia M.T. Greenwood

Abstract Kernel semi-parametric models and their equivalence with linear mixed models provide analysts with the flexibility of machine learning methods and a foundation for inference and tests of hypothesis. These models are not impacted by the number of predictor variables, since the kernel trick transforms them to a kernel matrix whose size only depends on the number of subjects. Hence, methods based on this model are appealing and numerous, however only a few R programs are available and none includes a complete set of features. Here, we present the **KSPM** package to fit the kernel semi-parametric model and its extensions in a unified framework. **KSPM** allows multiple kernels and unlimited interactions in the same model. It also includes predictions, statistical tests, variable selection procedure and graphical tools for diagnostics and interpretation of variable effects. Currently **KSPM** is implemented for continuous dependent variables but could be extended to binary or survival outcomes.

1 Introduction

In the last decades, the popularity and accessibility of machine learning has increased as a result of both the availability of big data and technical progress in computer science. The flexibility of machine learning methods enables avoidance of assumptions about functional relationships, such as strong linear or additive hypotheses, that are often involved in classical statistical models. However, this flexibility and adaptability also limits the capacity to interpret results or make inference. When understanding and inference are required, simpler statistical models are often preferred, as they are easy to understand and implement. Methods from the machine learning field might better model complex relationships, and yet would be more attractive if the results could be made interpretable. With this goal in mind, Liu et al. (2007) clearly demonstrated the under-appreciated equivalence between a machine learning tool and a classical statistic model through the link between kernel semi-parametric models – developed first in the machine learning field – and more classical linear mixed models. This equivalence allows analysts to take advantage of knowledge advances in both the machine learning domain and the traditional statistical inference domain, including hypotheses testing, when using kernel semi-parametric models (Table 1).

Features of kernel models	
from traditional statistical models	from machine learning models
• Inference, confidence/prediction intervals • Tests, p values • Information criteria (variable selection) • Interpretation via estimation of functional form of variable effects on outcome	• No need to explicitly define outcome-predictors relationship • May deal with a large amount of data (big data and fat data) • Potential for reinforcement learning

Table 1: Features of kernel models combine features from traditional statistical models and features from machine learning models.

The kernel semi-parametric model assumes that the outcome is related to the set of variables through an unknown smooth function, which is simply approximated by computing the similarity matrix between subjects according to the set of variables and a chosen kernel function (i.e., the kernel trick). Matrix size depends only on the number of subjects, making kernel models particularly suited to datasets where the number of features is very large ($p >> n$). The equivalence between kernel semi-parametric models and linear mixed models motivates a score test, which is simple to implement, for the simultaneous effect of all variables on the outcome.

Methods based on the kernel semi-parametric model and its extensions are appealing and numerous, but only a few programs are available, and none includes a complete set of the features that correspond to recent developments. Table 2 gives some examples of existing R packages according to their features of interest.

	Adjustment	*User's own kernel*	*Single kernel test*	*Test of interaction*	*Predictions*	*Interpretation plot*	*Diagnostic plots*	*Variable selection*
coxme	✓	✓						
SKAT		✓	✓					
SPA3G				✓				
KRLS					✓	✓		
e1071					✓			
KSPM	✓	✓	✓	✓	✓	✓	✓	✓

Table 2: Features incorporated in **KSPM** (Schramm, 2020), as well as in several other R packages for kernel nonparametric or semiparametric models: **coxme** (Therneau, 2018), **SKAT** (Lee et al., 2017), **KRLS** (Hainmueller and Hazlett, 2017), **e1071** (Meyer et al., 2018) and **SPA3G** (Li and Cui, 2012a). **Adjustment** refers to models including a kernel for adjusting the model on correlation structure similarly to a random factor. **User's own kernel** refers to a kernel function explicitly defined by the user of the package, in contrast to traditional kernel functions that are already implemented in the package. **Single kernel test** refers to the test of the joint effect of a set of variables on the outcome. **Test of interaction** refers to the interaction between two sets of variables and its effect on the outcome. **Predictions** refers to the possibility of displaying predictions with confidence and prediction intervals. **Interpretation plot** refers to graphical tools for interpretation of individual effects of each variable in the kernel on the outcome. **Diagnostic plots** refers to graphical tools based on residuals and leverage measures to check the validity conditions of a model and identify outlier samples. **Variable selection** refers to the implementation of a variable selection procedure.

Several packages in Table 2 were developed in the genetics field where interest often lies in testing the contribution of a group of variables (variants) simultaneously, notably through the sequence kernel association test (SKAT) for single nucleotide polymorphisms (Wu et al., 2011; Chen et al., 2016). Extensions from single to multiple kernel model were motivated by (i) interaction tests (Li and Cui, 2012b; Wang et al., 2017; Ge et al., 2015; Marceau et al., 2015) and (ii) estimating the conditional effect of one set of variables after adjusting for another set of variables, or after adjusting for population structure (Oualkacha et al., 2013). The latter goal may be achieved easily using the `lmekin()` function from the **coxme** package under mixed model theory where kinship matrix corresponds to the kernel matrix measuring similarity between subjects, and defines the covariance matrix of the random effect term.

In the machine learning field, packages like **e1071** have been developed to perform accurate predictions and they focus on a single kernel function. Traditionally, model interpretation has remained an outstanding challenge of the machine learning field. However it has been recently demonstrated that the kernel semi-parametric models can be used to interpret effects of variables on the outcome through a graphical tool based on derivatives (Hainmueller and Hazlett, 2013) and this is implemented in the **KRLS** package. When the kernel function and the corresponding approximated smooth function are differentiable with respect to the variable of interest, pointwise derivatives capture the effect of this variable on the outcome.

Since researchers may be interested in all of these features, we have consolidated them in the R package **KSPM**. The package and a vignette including detailed examples are available from the comprehensive R archive network (CRAN) at https://CRAN.R-project.org/package=KSPM. Our package, currently designed for continuous outcomes and normal errors, fits kernel semi-parametric models and their extensions in a unified framework incorporating all the previously described model fitting features and tests. It allows multiple kernels and unlimited interactions in the same model. Although most popular kernel functions are available in the package, the user also has the option of designing and using his/her own kernel functions. Furthermore, whenever interest lies in prediction or making inference from the model, diagnostic assessments of the model may be performed through graphical tools to detect data points with large residuals or high leverage that may greatly influence the results. Finally, we have also included a variable selection procedure based on Akaike's and Bayesian information criteria (AIC and BIC). These last two options are not included in other software packages.

The **KSPM** package is a new tool for semi-parametric regression. It is not competing with other R packages for semi- or non-parametric regression models since either our methods or our objectives are different. Indeed, the estimation method involved in **KSPM** is based on regularized least squares in the kernel Hilbert space and should not be confused with local kernel smoothing based on Nadaraya-Watson estimator (**np** package, Racine and Hayfield (2020)). Similarly, **KSPM** is also different from other smoothing methods connecting regression segments through knots among which we may cite splines (**mgcv** package, Wood (2020)).

However, since the kernel semi-parametric model is equivalent to a linear mixed effect model, our package could be compared to **lme4** (Bates et al., 2019) or **nlme** (Pinheiro et al., 2019) packages for linear mixed effect models. Although it is possible to obtain similar inference from any variables involved in a linear part of each model, and to obtain similar predictions from both overall models, loss function maximization are different. Moreover, **KSPM** has the advantage of being easier-to-use as the user does not need to compute kernel matrix nor matrix of interactions and provides interpretations for variable effects that cannot be obtained with traditional packages.

2 Kernel semi-parametric models

Single kernel semi-parametric model

Let $Y = (Y_1, ..., Y_n)^\top$ be a $n \times 1$ vector of continuous outcomes where Y_i is the value for subject i ($i = 1, ..., n$), and X and Z are $n \times p$ and $n \times q$ matrices of predictors, such that Y depends on X and $Z = (\mathbf{z_{.1}}, ..., \mathbf{z_{.q}})$ as follows:

$$Y = X\beta + h(Z) + e \tag{1}$$

where β is a $p \times 1$ vector of regression coefficients for the linear parametric part, $h(.)$ is an unknown smooth function and e is an $n \times 1$ vector of independent and homoscedastic errors such as $e_i \sim \mathcal{N}(0, \sigma^2)$. Throughout this article, $X\beta$ will be referred to as the linear part of the model and we assume that X contains an intercept. $h(Z)$ will be referred to as the kernel part. The function $h(.)$ need not be explicitly specified but it is assumed to lie in $\mathcal{H}_k$, the function space generated by a kernel function $k(.,.)$ which is, by definition, symmetric and positive definite. We note that K, the $n \times n$ Gram matrix of $k(.,.)$ such that $K_{ij} = k(\mathbf{z_{i.}}, \mathbf{z_{j.}})$, represents the similarity between subjects i and j according to Z. Of note, in the estimation process, $h(.)$ will be expressed as a linear combination of $k(.,.)$ (see Equation (7)). See "Fitting the kernel semi-parametric model with `kspm`" for straightforward coding.

The multiple kernel semi-parametric model

Now suppose there are L matrices $Z_1, ..., Z_L$ of dimension $n \times q_1$, ..., $n \times q_L$ respectively. Keeping a similar notation, Y can be assumed to depend on X and Z_1, ..., Z_L as follows:

$$Y = X\beta + \sum_{\ell=1}^{L} h_\ell(Z_\ell) + e \tag{2}$$

where $\forall \ell \in \{1, ..., L\}$, h_ℓ is an unknown smooth function from $\mathcal{H}_{k_\ell}$, the function space generated by a kernel function $k_\ell(.,.)$ whose Gram matrix is noted K_ℓ. We assume $\forall \ell \neq m$, $h_\ell(Z_\ell) \neq h_m(Z_m)$ either because $Z_\ell \neq Z_m$ or $k_\ell(.,.) \neq k_m(.,.)$ or both. Note that when $L = 1$, the model corresponds to Equation (1).

Suppose there is an interaction between two sets of variables represented by Z_1 and Z_2, $n \times q_1$ and $n \times q_2$ matrices, and Y depends on X, Z_1 and Z_2 as follows:

$$Y = X\beta + h_1(Z_1) + h_2(Z_2) + h_{12}(Z_1, Z_2) + e \tag{3}$$

where $h_1(.)$, $h_2(.)$ and $h_{12}(.,.)$ are unknown smooth functions from $\mathcal{H}_{k_1}$, $\mathcal{H}_{k_2}$ and $\mathcal{H}_{k_{12}}$ respectively. The $h_{12}(.,.)$ function represents the interaction term if its associated kernel function $k_{12}(.,.)$ is defined as the product of kernel functions $k_1(.,.)$ and $k_2(.,.)$ as follows:

$$k_{12}\left((Z_1, Z_2)_i, (Z_1, Z_2)_j\right) = k_1(\mathbf{z_{1i.}}, \mathbf{z_{1j.}}) \times k_2(\mathbf{z_{2i.}}, \mathbf{z_{2j.}}) \tag{4}$$

such that $K_{12} = K_1 \odot K_2$ where $\odot$ is the Hadamard product (Ge et al., 2015). Model (3) is a particular case of the multiple kernel semi-parametric model (2) with $L = 3$, $Z_3 = (Z_1, Z_2)$, $q_3 = q_1 + q_2$ and h_3 = h_{12}. Of notes, in **KSPM**, different kernel choices can be made for $k_1(.,.)$ and $k_2(.,.)$. Obviously, this 2-way interaction model could be generalized to higher order interaction terms in a similar manner.

Link with linear mixed models

As shown by Liu et al. (2007), model (2) is equivalent, without additional conditions, to the following linear mixed model:

$$Y = X\beta + \sum_{\ell=1}^{L} h_\ell + e \tag{5}$$

where $\forall \ell \in \{1, ..., L\}$, $h_\ell \sim \mathcal{N}(0, \tau_\ell K_\ell)$ with K_ℓ is a matrix of similarity between subjects as defined in model (2). Throughout the paper, we denote the variance parameters as $\theta = (\tau_1, ..., \tau_L, \sigma^2)$ and $\Sigma_\theta = \sum\limits_{\ell=1}^{L} \tau_\ell K_\ell + \sigma^2 I$, the variance-covariance matrix of Y, where I is the $n \times n$ identity matrix.

Estimation of parameters

The parameter estimates can be obtained either by maximizing the penalized log-likelihood associated with the kernel semi-parametric model (2), or the log-likelihood derived from the corresponding linear mixed model (5) (Liu et al., 2007). Even though the latter option may be computationally less time-consuming, we implemented the **KSPM** package using a penalized log-likelihood associated with the kernel semi-parametric model, because this method leads to an estimation of h useful for prediction, and also leads to suitable approaches for interpretation through kernel derivatives. Estimation of parameters is obtained by maximizing the penalized log-likelihood function:

$$l(\beta, h) = -\frac{1}{2} \sum_{i=1}^{n} \left(Y_i - X_i^\top \beta - \sum_{\ell=1}^{L} h_\ell(Z_{\ell i}) \right)^2 - \frac{1}{2} \sum_{\ell=1}^{L} \lambda_\ell \parallel h_\ell \parallel^2_{\mathcal{H}_{k_\ell}} \tag{6}$$

where λ_1, ..., λ_L are tuning parameters associated with each smooth function $h_1(.)$, ..., $h_L(.)$ and $\parallel . \parallel^2_{\mathcal{H}_{k_\ell}}$ defines the inner product on $\mathcal{H}_{k_\ell}$ space. According to the Representer theorem (Kimeldorf and Wahba, 1971), the functions h_1, ..., h_L satisfying Equation (6) can be expressed as:

$$\forall \ell \in \{1, ..., L\}, \quad h_\ell(.) = \sum_{i=1}^{n} \alpha_{\ell i} k(., Z_{\ell i}) \tag{7}$$

where $\forall \ell \in \{1, ..., L\}$, $\alpha_\ell = (\alpha_{\ell 1}, ..., \alpha_{\ell n})^\top$ is a $n \times 1$ vector of unknown parameters. Estimating the kernel semi-parametric model parameters consists in estimating $\alpha_1, ..., \alpha_L$ and β. Then, estimators of $h_1(.)$, ..., $h_L(.)$ are deduced from $\hat{\alpha}_1, ..., \hat{\alpha}_L$.

In **KSPM**, we estimate penalization parameters by minimizing the mean sum of squares of the leave-one-out errors (LOOE). An advantage of LOOE compared to other cross-validation methods is that we do not need to recompute new model(s), because its value may be derived directly from the primary model parameters:

$$\forall i \in \{1, ..., n\}, \qquad \text{LOOE}_i(\lambda_1, ..., \lambda_L) = \frac{Y_i - \hat{Y}_i}{1 - H_{ii}} \tag{8}$$

where H_{ii} is the i^{th} diagonal element of the Hat matrix H such as $\hat{Y} = HY$.

Penalization parameters and tuning parameters, if applicable, are estimated simultaneously during the optimization algorithm, by minimizing the LOOE. If only one parameter needs to be estimated, the convexity of the function to be minimized makes the problem easier and the convergence faster. In that case, **KSPM** uses the standard `optimize()` function from the basic R package, based on the golden section search algorithm (Brent, 2013). When several hyperparameters need to be estimated, the resulting function to be minimized may not be convex. Hence, more complex optimization algorithms should be envisaged, and **KSPM** uses the `DEoptim()` function from the **DEoptim** package (Ardia et al., 2020) based on the differential evolution algorithm (Mullen et al., 2009). Given the random nature of the algorithm, it would be safe to apply the algorithm several times to ensure convergence toward the global minimum.

Tests of hypotheses in KSPM

For either a single kernel or a multiple kernel semi-parametric model, a standard test of interest is $H_0 : h_\ell(.) = 0$, i.e., there is no effect, singly or jointly, of a set of variables on the outcome. Following Liu et al. (2007), this test is equivalent to $H_0 : \tau_\ell = 0$, a test of the variance component in the linear mixed model (5) using the REML-based score test. The corresponding test statistic Q_ℓ follows a mixture of independent chi-square distributions with one degree of freedom (Zhang and Lin, 2003). The **KSPM** package uses exact distribution of Q_ℓ in single kernel model but in multiple kernel model, we use Davies' method to approximate this distribution (Davies, 1980). Based on similar methodology, **KSPM** also provides the global test for multiple kernel semi-parametric models $H_0 : h_1(.) = ... = h_L(.) = 0$ i.e., $H_0 : \tau_1 = ... = \tau_L = 0$.

Interpretation of variable effects

A kernel represents similarity between subjects through combining a set of variables in a possibly complex way, that may include their possible interactions. Hence the general effect of a kernel on an outcome is hard to see and difficult to interpret. Nevertheless, the marginal effect of each variable in the kernel may be illustrated using a partial derivative function (Hainmueller and Hazlett, 2013). Indeed, the derivative measures how the change in a variable of interest impacts the outcome in an intuitively understandable way. When a variable's effect is linear, the interpretation is straightforward since the derivative corresponds to standard slope (β) coefficients. In kernel semi-parametric models, we are simultaneously modeling a set of variables. Therefore, when exploring the effect of any one variable of interest, the other variables in the kernel must be taken into account. Thus, plotting pointwise derivatives of the prediction for each subject against the value of the variable of interest may help in interpreting the effect of this variable on the outcome. Although a variable-level summary statistic may be obtained by averaging the pointwise derivatives across subjects (Hainmueller and Hazlett, 2013), we did not implement this option in **KSPM** because the average can mask relevant variability. For example, when positive and negative derivatives occur for the same variable, the average may be zero.

The choice of kernel functions

In any kernel semi-parametric model, the smooth unknown function $h_\ell(.)$ is approximated using basis functions from $\mathcal{H}_{k_\ell}$. Since the inner product of the basis functions corresponds to the kernel function $k_\ell(.,.)$, the choice of the kernel determines the function space used to approximate $h_\ell(.)$. The **KSPM** package includes the most popular kernel functions, described below. The linear kernel function $k(Z_i, Z_j) = Z_i^\top Z_j$ assumes that variables have a linear effect on outcome. It generates a linear function space so that the kernel semi-parametric model leads to a penalized multiple linear model using an L^2 norm (equivalent to a ridge regression). The polynomial kernel function $k(Z_i, Z_j) = (\rho\, Z_i^\top Z_j + \gamma)^d$ assumes that d^{th}-order products of the variables have a linear effect on the outcome, and is equivalent to a d^{th}-order interaction model. The Gaussian kernel function $k(Z_i, Z_j) = exp(- \parallel Z_i - Z_j \parallel^2 /\rho)$ generates the infinite function space of radial basis functions. The sigmoid kernel function $k(Z_i, Z_j) = tanh(\rho\, Z_i^\top Z_j + \gamma)$ and the inverse quadratic kernel $k(Z_i, Z_j) = (\parallel Z_i - Z_j \parallel^2 + \gamma)^{-1/2}$ are also often cited in the literature. Finally, we propose also the equality kernel $k(Z_i, Z_j) = 1$ if $Z_i = Z_j$ and 0 otherwise. Of note, users can define their own kernel function in **KSPM** by providing the corresponding kernel matrix. Some kernel functions, such as the linear or polynomial kernels, make assumptions about the shape of the effect of the variables on the continuous outcome, whereas other kernels like the gaussian may, in theory, handle all types of effects, regardless of their complexity. Indeed, in contrast to the linear and polynomial kernels, the gaussian kernel function captures a kernel space of infinite size leading to higher flexibility for approximation of $h(.)$. If true effects are linear, the linear kernel and gaussian kernel should converge toward similar results. However, in practice, if sample size is low or noise is large, the gaussian kernel will tend to retain a sinusoidal shape to the fit even when the truth is linear.

Kernel functions often involve tuning parameters; above, the parameters ρ and γ were used to indicate these kernel specific parameters. In practice, these tuning parameters are usually chosen by the user. However, if user does not provide a value for these parameters, the **KSPM** package estimates them at the same time as the penalization parameter(s), by minimizing the mean sum of squares of LOOE. The choice of tuning parameters may strongly impact the results and modify the smoothness of the $\hat{h}(.)$ function leading to overfitting or underfitting. For example, with a ρ value that is too large, the Gaussian kernel tends to lose its non-linear properties, whereas with ρ too small, it is very sensitive to noise.

In general, the choice of tuning parameters may strongly impact the results With that in mind, sensitivity analyses may include a comparison of results obtained with different values for these parameters. Also, comparing models based on information criteria such as AIC and BIC may help to choose the kernel function and its tuning parameter(s).

3 Package presentation

The **KSPM** package provides an R interface to perform kernel semi-parametric models and is available from the comprehensive R archive network (CRAN) at https://CRAN.R-project.org/package=KSPM. The package is called through the main function `kspm()` taking data and model hyperparameters as inputs, fitting the model, and returning a model fit object of class `"kspm"`.

Fitting the kernel semi parametric model with `kspm`

The main function of the package, `kspm()`, can fit the single or multiple kernel semi-parametric models described in the earlier, as detailed below:

```
kspm(response,linear,kernel,data,...)
```

The argument `response` indicates a continuous outcome variable. It can be specified as a string corresponding to the column name associated with the response in the dataset provided in the `data` argument, as a vector of length n, or as a $n \times 1$ matrix containing the continuous values of the outcomes. Of note, `kspm` does not deal with multivariate outcomes, and if an $n \times r$ (> 1) matrix is provided, only the first column is used. Argument `linear` specifies the linear part of the model and could be either a formula, a vector of length n if only one variable is included in the linear part, or an $n \times p$ design matrix containing the p variables included in the linear part. By default, an intercept is added. To remove the intercept term, the user should use the formula specification and add the term `-1`, as usual. `kernel` specifies the kernel part of the model. Its argument should be a formula of `Kernel` object(s), described below. For a multiple kernel semi-parametric model, `Kernel` objects are separated by the usual signs `"+"`, `"*"` and `":"` to specify addition and interaction between kernels.

The `Kernel` object regroups all information about a kernel including the choice of kernel function and its parameters. It is specified using the `Kernel` function as follows:

```
Kernel(x,kernel.function,scale,rho,gamma,d)
```

Argument `x` represents either the variables included in the kernel part of the model, or a kernel Gram matrix. In the latter case, the user should specify `kernel.function = "gram.matrix"` and all other arguments are not used. When `x` represents the variables included in the kernel part, it may be specified as a formula, a vector, or a matrix, and `kernel.function` indicates which kernel function should be used (e.g., `"gaussian"`, `"polynomial"`, ...). `scale` is a boolean indicating if variables should be scaled before computing the gram matrix. The need for other arguments depends on the choice of kernel function: `rho` and `gamma` are tuning parameters and `d` is the highest order in a polynomial kernel function; these parameters correspond to the ρ, γ and d introduced earlier. It is worth noting that in a multiple kernel model, **KSPM** allows kernel objects to follow different formats.

Different options were considered for the interface with respect to specification of the linear and kernel parts of the model. We decided to use an interface with separate formulae for the two parts. This structure makes it straightforward to manage variables coming from different sources or data structures within the package. For example, genetic data or high dimensional genomic data are often provided in a matrix format, whereas other variables and phenotypes are saved in vectors or data frames with meaningful variable names. This diversity of data source and format cannot be handled by a unique formula. If data elements are assembled from different sources, they should include the same individuals or observations ($i = 1, .., n$), with identical ordering; if not, `kspm` will return an error. It is worth noting that the **KSPM** package does tolerate observations containing missing values, although these observations will be removed prior to model fitting.

The function `kspm` returns an object of class `"kspm"` with `summary()` and `plot()` commands available, the first giving estimates and p-values and the second displaying diagnostic plots.

Methods applicable to objects of the class `"kspm"`

An object of class `"kspm"` results from a kernel semi-parametric model fit. It contains obviously estimated coefficients, fitted values and residuals, but also information about kernels such as $n \times n$ kernel matrices, hyperparameters, penalization parameters.

The `"kspm"` object can be summarized or viewed using commands and methods very similar to those implemented in `"lm"` or `"glm"`.

```
> methods(class = "kspm")
 [1] case.names  coef          confint    cooks.distance  deviance
 [6] extractAIC  fitted        logLik     nobs            plot
[11] predict     print         residuals  rstandard       sigma
[16] summary     variable.names
see '?methods' for accessing help and source code
```

Predictions

A `predict()` command has been implemented for the class "kspm".

```
predict.kspm(object,newdata.linear,newdata.kernel,interval,level)
```

where `object` refers to a "kspm" object. If a new dataset is not specified, `predict.kspm` will return the fitted values from the original data. If `predict.kspm` is applied to a new dataset, all variables used in the original model should be provided in the `newdata.linear` and `newdata.kernel` arguments. `newdata.linear` should be a data frame, a vector or design matrix of variables used in the linear part. `newdata.kernel` is a list containing data frames, vectors and/or design matrices for each kernel. Formats depend on the ones previously used in model specification as shown in Table 3. For simplicity, users may follow the list returned by the `info.kspm()` function.

kernel specifications	new data specification
formula of q variables	a `data.frame` with columns names corresponding to variables included in the formula. Number of rows is $n^{\star}$, number of columns is q
vector	a vector of length $n^{\star}$
design matrix $n \times q$	a matrix $n^{\star} \times q$
kernel matrix $n \times n$	a matrix $n^{\star} \times n$ where cell i, j represents the similarity between new subject i and j^{th} subject included in the model.

Table 3: How new data should be specified in `predict.kspm`, depending on original model specifications for $n^{\star}$ new subjects. The first column refers to how the kernel is specified in the current model. The second column refers to how the new data should be specified in the `predict.kspm` function.

In `predict.kspm`, `interval` can be either "none", "confidence", or "prediction" according to whether the user wants a confidence or prediction interval. The level of confidence/prediction interval is specified using the `level` argument. By default, `level = 0.95` is used.

Variable selection procedures for the single kernel semi-parametric model

A variable selection procedure algorithm has been implemented for the class "kspm". It is similar to the `step()` command existing for other regression packages.

```
stepKSPM(object,linear.lower,linear.upper,kernel.lower,kernel.lower,k,direction)
```

As before, `object` refers to a "kspm" object. However, in contrast to the generality allowed for fitting a single kernel semi-parametric model, here the `Kernel` object should not be specified with the Gram matrix option. The arguments of `linear.lower`, `linear.upper`, `kernel.lower` and `kernel.lower` are formulae corresponding to the desired boundaries for the smallest and largest numbers of variables to be included. As is standard in many R packages, all variables in the `linear.lower` and `kernel.lower` formulae cannot be removed from the model and variables that are not in the `linear.upper` and `kernel.upper` formulae cannot be added to the model. The procedure to select variables is based on AIC or BIC depending on the value of `k`. If `k` is set to 2, AIC is used, if `k` is set to $ln(n)$, BIC is used instead. The `direction` argument may be "forward" for a forward selection, "backward" for a backward selection and "both" for a stepwise selection. Our package was implemented so that variable selection for the linear part and the kernel part of the model may be done simultaneously. However, it is also possible to perform the variable selection procedure on each part separately by giving the appropriate formula to `linear.lower`, `linear.upper`, `kernel.lower` and `kernel.lower`.

Of note, as for the standard `stepAIC()` function, this procedure can only be used on complete observations. Thus missing data should be removed by the user prior to calling `stepKSPM()` so that the number of observations is identical at each step.

Graphical tools

The `plot()` method has been implemented for `"kspm"` and `"derivatives.kspm"` objects. The former gives usual diagnostic plots including residual distribution, leverage, Cook's distance,... The latter gives interpretation plot from pointwise derivatives.

4 Example 1: the Movie ratings data

This first example illustrates the functions provided in **KSPM** included the model fit, the diagnotic plot, the interpetation plot based on pointwise derivatives, the test of interaction as well as the variable selection procedure.

The conventional and social media movies (CSM) dataset is available on the UCI machine learning repository (https://archive.ics.uci.edu/ml/index.php) as well as in **KSPM** and is fully described in Ahmed et al. (2015). It contains data on 232 movies from 2014 and 2015; the movies are described in term of conventional features (sequel, budget in USD, gross income in USD, number of screens in USA) as well as social media features (aggregate actor followers on Twitter, number of views, likes, dislikes, comments of movie trailer on Youtube, sentiment score) in the aim of predicting ratings. In all subsequent analyses, we used only the 187 entries without missing data.

```
> data("csm")
> head(csm)

  Year Ratings     Gross  Budget Screens Sequel Sentiment   Views Likes
1 2014     6.3      9130 4.0e+06      45      1         0 3280543  4632
2 2014     7.1 192000000 5.0e+07    3306      2         2  583289  3465
3 2014     6.2  30700000 2.8e+07    2872      1         0  304861   328
4 2014     6.3 106000000 1.1e+08    3470      2         0  452917  2429
5 2014     4.7  17300000 3.5e+06    2310      2         0 3145573 12163
7 2014     6.1  42600000 4.0e+07    3158      1         0 3013011  9595
  Dislikes Comments Aggregate.Followers
1      425      636             1120000
2       61      186            12350000
3       34       47              483000
4      132      590              568000
5      610     1082             1923800
7      419     1020             8153000
```

Predict ratings using conventional features

In our first model, we assume a gaussian kernel function to fit the joint effect of the conventional features on ratings. Here we do not provide any value for the ρ parameter. It will be estimated at the same time as the penalization parameter by minimizing the LOOE. We also do not provide a `linear` argument, meaning that only an intercept will be included in the linear part of the model.

```
> csm.fit1 <- kspm(response = "Ratings", kernel = ~Kernel(~Gross + Budget +
+    Screens + Sequel, kernel.function = "gaussian"), data = csm)
> summary(csm.fit1)

Call:
kspm(response = "Ratings", kernel = ~Kernel(~Gross + Budget +
    Screens + Sequel, kernel.function = "gaussian"),
    data = csm)

Sample size:
n = 187

Residuals:
    Min      Q1  Median      Q3     Max
-3.0066 -0.4815  0.0109  0.5534  2.1228

Coefficients (linear part):
            Estimate Std. Error  t value     Pr(>|t|)
(Intercept) 6.297723   1.058707 5.948505 1.427565e-08
```

```
Score test for non-parametric kernel:
         lambda      tau      p-value
Ker1 0.04804093 16.13793 5.625602e-06

Residual standard error: 0.88 on 175.82 effective degrees of freedom
Multiple R-squared: 0.2643, Adjusted R-squared: 0.2217
```

The summary output gives information about the sample size used in the model, the residual distribution, the coefficient for the linear part (similar to other regression R packages), and the penalization parameter and score test associated with the kernel part. The kernel results indicate that the conventional features are strongly associated with ratings. In such a complex model, the number of free parameters – i.e., the standard definition of the degrees of freedom of a model – is undefined, and we use instead the effective degrees of freedom of the model, which is not necessarily a whole number. However, our definition for effective degrees of freedom is similar to the one used in linear regression models and depends on the trace of the hat matrix. The ρ parameter may be extracted using the following code:

```
> csm.fit1$kernel.info$Ker1$rho

 par1
61.22
```

This value alone may not provide much information about linearity of the kernel function. However interpretation is feasible when comparing gaussian kernel functions with different ρ parameter values, or when comparing the gaussian and linear kernel functions.

The plot() command may be applied to the model to display diagnostic plots. Figure 1 has been generated using the following code:

```
> par(mfrow = c(2,2), mar = c(5, 5, 5, 2))
> plot(csm.fit1, which = c(1, 3, 5), cex.lab = 1.5, cex = 1.3, id.n = 7)
> hist(csm$Ratings, cex.lab = 1.5, main = "Histogram of Ratings", xlab =
+    "Ratings")
```

Outlier points are annotated and we can see movie 134 (*Jurassic World*) has high leverage. Also of note, the lower tail of the residuals distribution is not as expected for a Normal distribution. The histogram of ratings shows that the deviation could be due to the left skewed distribution of the response variable.

The derivative function may help to interpret the effect of each variable individually on the outcome. Indeed, the sign of the derivatives captures variational changes of the effect of the variable on the outcome. To illustrate this feature, Figure 2 displays the derivatives for Gross income, Budget and Screens. It has been generated with the following code:

```
> par(mfrow = c(1,2), mar = c(5,5,5,2))
> plot(derivatives(csm.fit1), subset = "Gross", cex.lab = 1.5, cex = 1.3,
+    main = "Pointwise derivatives according to Gross Income")
> plot(derivatives(csm.fit1), subset = "Screens", col = csm$Sequel,
+    cex.lab = 1.5, cex = 1.3, pch = 16, main = "Pointwise derivatives
+    according to Number of Screens \n and Sequel")
> legend("topleft", fill = palette()[1:7], legend = 1:7, title = "Sequel",
+    horiz = TRUE)
```

By default, the X-axis label gives the name of the variable and, in brackets, the kernel in which it is included. When only one kernel is included in the kernel, it is named Ker1. Because genre and sequel variables, even if they are numeric, refer to categorical variables, it is possible to easily highlight some patterns of interaction between these variables and the others. Derivatives according to Gross income are mostly positive meaning that higher Gross income is associated with higher ratings. However the slope of this effect decreases as the gross income grows. It is difficult to interpret the derivatives for Gross income higher than 2.5e+08 since this category includes only a few movies. Based on the display in the right panel, whether a movie has sequels seems to interact with the effect of the number of screens on the ratings. Indeed when the movie is the first or second release (Sequel = 1 or 2), the derivatives are always negative meaning that as the number of screens on which the movie is released increases, the ratings tend to decrease. However, this relationship seems to be stronger for the first release than the second. No clear pattern can be observed for subsequent sequels. It is difficult to know if this reveals an absence of effect or whether there are simply too few movies past sequel 2 in these data.

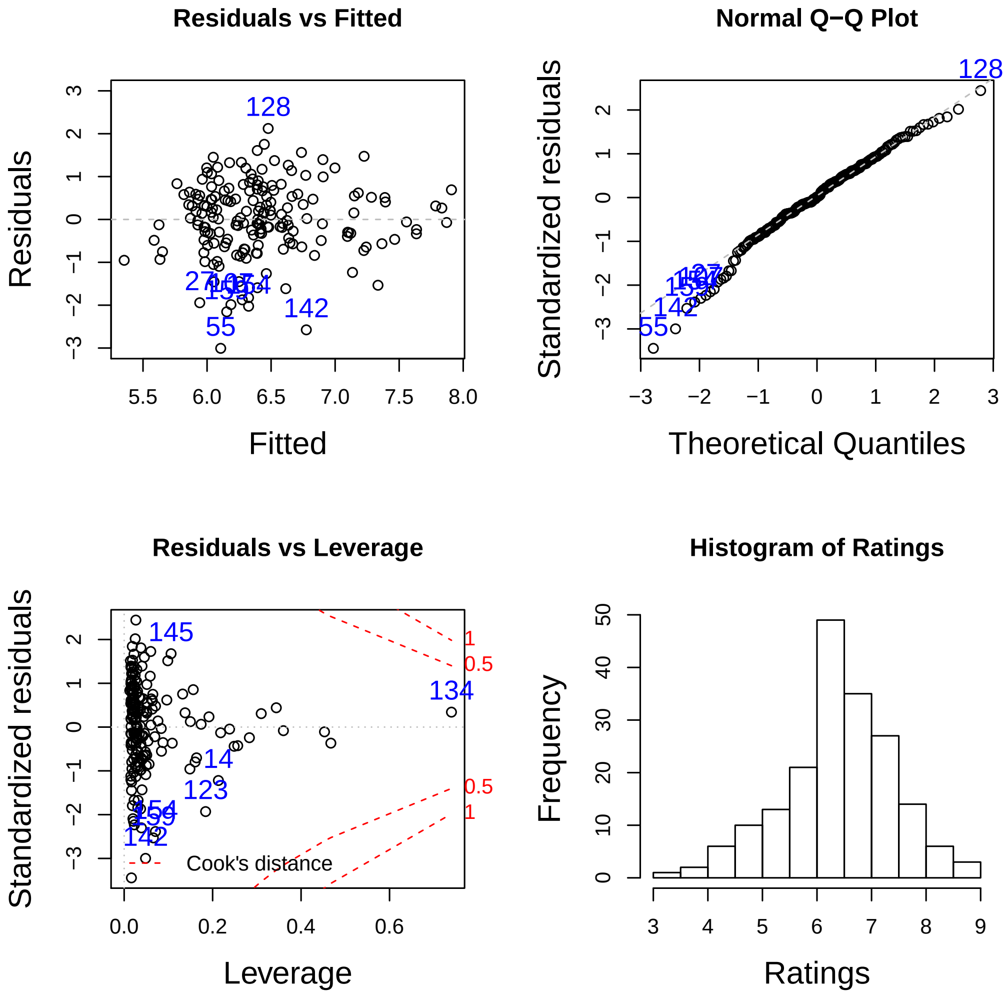

Figure 1: Diagnostic plots of CSM data. Plots at the top left, top right and bottom left were obtained with `plot.kspm`. They represent respectively residuals against fitted values, the normal Q-Q plot of residuals and residuals against leverage with the Cook's distance information. Plot at the bottom right represents the distribution of ratings in the database.

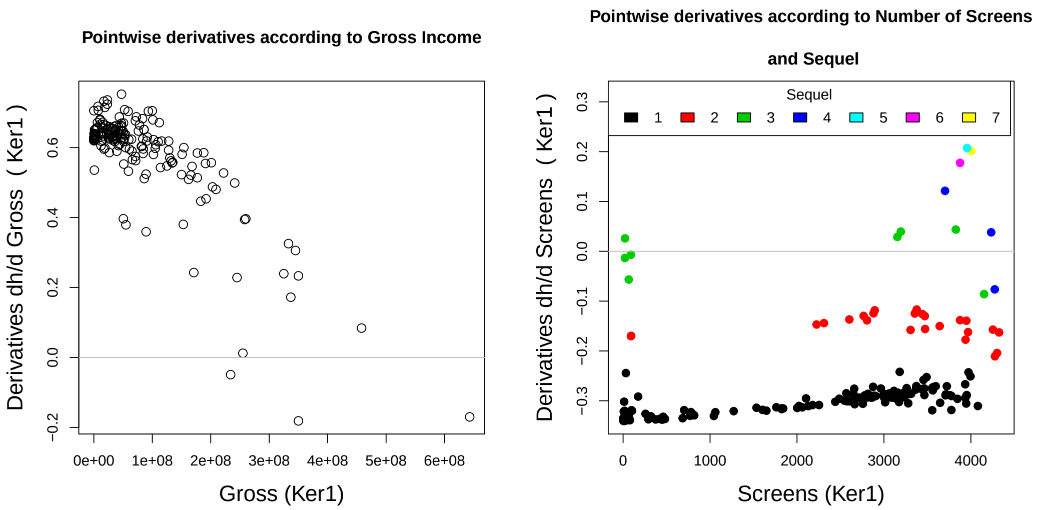

Figure 2: Derivative plots on CSM data obtained with `plot.derivatives`. Each point corresponds to an observation. Plot on the left represents the pointwise derivatives according to the Gross income variable. Plot on the right represents the pointwise derivatives according to the Screens variable and are colored according Sequel variable showing a probable interaction between Screens and Sequel.

To help in the choice of the kernel function, we may compare several models by using information criteria. As an example, we fit a second model assuming a polynomial kernel function with fixed tuning parameters ($\rho = 1, \gamma = 1$ and $d = 2$). This model can be compared to the previous one using information criteria such as the AIC or BIC. By default the `extractAIC()` command gives the AIC.

```
> csm.fit2 <- kspm(response = "Ratings", kernel = ~Kernel(~Gross + Budget +
+    Screens + Sequel, kernel.function = "polynomial", rho = 1, gamma = 1,
+    d = 2), data = csm, level = 0)
> extractAIC(csm.fit1)

[1] 941.4521

> extractAIC(csm.fit2)

[1] 944.4618
```

Here, we concluded that gaussian kernel function fits our data better than the polynomial kernel function, given the tuning parameters we considered.

Adding social media features to the model: a model with kernel interaction

Now, we assume a model with two kernel parts, one for conventional features and one for social media features, as well as their interaction. We propose to use the gaussian kernel function for each set of features, although different kernels could be used. The hyperparameters we chose are those obtained for each kernel separately.

```
> csm.fit3 <- kspm(response = "Ratings", linear = NULL, kernel = ~Kernel(~
+    Gross + Budget + Screens + Sequel, kernel.function = "gaussian",
+    rho = 61.22) * Kernel(~ Sentiment + Views + Likes + Dislikes + Comments +
+    Aggregate.Followers, kernel.function = "gaussian", rho = 1.562652),
+    data = csm)
```

While the model is running, R returns a summary of the kernel part(s) and interaction(s) included in the model.

```
------------------------------------------------------------------
The model includes the following kernels:
Ker1
Ker2
Ker1:Ker2
```

```
-----------------------------------------------------------------
Details:
Ker1: ~Kernel(~Genre + Gross + Budget + Screens + Sequel,
kernel.function = "gaussian", rho = 55.5897)
Ker2: ~Kernel(~Sentiment + Views + Likes + Dislikes + Comments +
Aggregate.Followers, kernel.function = "gaussian", rho = 1.562652)
-----------------------------------------------------------------
```

As defined by model (4), the `summary()` command will return the p value of tests $H_0 : h_1(.) = 0$, $H_0 : h_2(.) = 0$ and $H_0 : h_{12}(.) = 0$. By default all tests are performed. However, if our interest lies only in the test of interaction, the `kernel.test` option may be used to choose the test of interest and reduce the computation time. If interest lies in the global test $H_0 : h_1(.) = h_2(.) = h_{12}(.) = 0$, the `global.test` option should be set at `TRUE`.

```
> summary(csm.fit3, kernel.test = "Ker1:Ker2", global.test = TRUE)

Call:
kspm(response = "Ratings", linear = NULL, kernel = ~Kernel(~Gross +
    Budget + Screens + Sequel, kernel.function = "gaussian",
    rho = 61.22) * Kernel(~Sentiment + Views + Likes + Dislikes +
    Comments + Aggregate.Followers, kernel.function = "gaussian",
    rho = 1.562652), data = csm)

Sample size:
n = 187

Residuals:
    Min      Q1  Median      Q3     Max
-1.4185 -0.3396  0.0112  0.3291  1.3597

Coefficients (linear part):
            Estimate Std. Error  t value     Pr(>|t|)
(Intercept) 4.548524   1.256813 3.619095 0.0004324838

Score test for non-parametric kernel:
          lambda        tau   p-value
Ker1:Ker2    187 0.00208359 0.7376828

Global test: p-value = 6e-04

Residual standard error: 0.62 on 121.17 effective degrees of freedom
Multiple R-squared: 0.7452, Adjusted R-squared: 0.6089
```

Adding social media features to the model improved the predictions as indicated by the adjusted R^2. However the smooth function associated with the kernel interaction does not significantly differ from the null, leading to the conclusion that there is no interaction effect between conventional and social media features on the ratings.

Suppose now, we want to predict the ratings that will be attributed to the three artificial movies described in tables 4 and 5 below, according to the model `csm.fit3`.

	Gross	Budget	Screens	Sequel
Movie 1	5.0e+07	1.8e+08	3600	2
Movie 2	50000	5.2e+05	210	1
Movie 3	10000	1.3e+03	5050	1

Table 4: The conventional features of three artificial movies. Rows represent the new movies and columns represent the features.

```
> newdata.Ker1 <- data.frame(Gross = c(5.0e+07, 50000, 10000),
+    Budget = c(1.8e+08, 5.2e+05, 1.3e+03), Screens = c(3600, 210, 5050),
+    Sequel = c(2, 1, 1))
> newdata.Ker2 <- data.frame(Sentiment = c(1, 2, 10), Views = c(293021,
+    7206, 5692061), Likes = c(3698, 2047, 5025), Dislikes = c(768, 49,
```

	Sentiment	Views	Likes	Dislikes	Comments	Aggregate.Followers
Movie 1	1	293021	3698	768	336	4530000
Movie 2	2	7206	2047	49	70	350000
Movie 3	10	5692061	5025	305	150	960000

Table 5: The social media features of three artificial movies. Rows represent the new movies and columns represent the features.

```
+    305), Comments = c(336, 70, 150), Aggregate.Followers = c(4530000,
+    350000, 960000))
> predict(csm.fit3, newdata.kernel = list(Ker1 = newdata.Ker1, Ker2 =
+    newdata.Ker2), interval = "prediction")

       fit      lwr      upr
1 4.682560 3.147755 6.217365
2 6.401853 5.100309 7.703396
3 6.128641 4.395417 7.861864
```

The output of the `predict()` function gives the predicted values (`fit`) and the lower (`lwr`) and upper (`upr`) bounds of prediction intervals.

We may obtain the predictions for the original data directly from the model or from the `predict()` function. With the latter, confidence intervals may be additionally obtained.

```
> pred <- csm.fit3$fitted.value
> pred <- predict(csm.fit3, interval = "confidence")
> plot(csm$Ratings, pred$fit, xlim = c(2, 10), ylim = c(2, 10),
+    xlab = "Observed ratings", ylab = "Predicted ratings", cex.lab = 1.3)
> abline(a = 0, b = 1, col = "red", lty = 2)
```

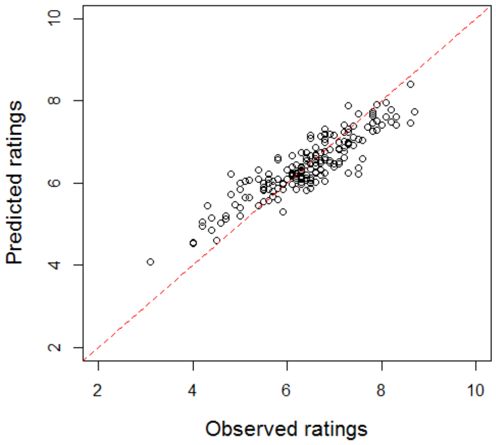

Figure 3: Predicted versus observed ratings in CSM dataset. Red dotted line represents a perfect concordance between predictions and observations.

Figure (3) shows that for smaller values, the model overestimates the outcome. This is again probably due to the left skewness of the outcome distribution.

An example of the variable selection procedure

Suppose we fit a single kernel semi-parametric model with a gaussian kernel to adjust the social media features in the CSM data. The kernel part contains the set of social media features. We want to select the relevant variables to be included in the kernel. We therefore can perform a stepwise variable selection procedure based on AIC, while letting ρ vary at each iteration. To do so, we first fit the full model including all features.

```
> csm.fit4 <- kspm(response = "Ratings", kernel = ~Kernel(~ Sentiment + Views
+    + Likes + Dislikes + Comments + Aggregate.Followers, kernel.function =
+    "gaussian"), data = csm)
```

Then, we apply the stepKSPM() command on the full model as follows.

```
> stepKSPM(csm.fit4, kernel.lower = ~1, kernel.upper = ~ Sentiment + Views
+    + Likes + Dislikes + Comments + Aggregate.Followers, direction = "both",
+    k = 2, kernel.param = "change", data = csm)
```

At each iteration, R returns the current model, and the list of variables that may be added or removed.

```
Start: AIC = 913.2
Linear part: ~ 1
Kernel part: ~ Sentiment + Views + Likes + Dislikes + Comments +
               Aggregate.Followers

                        Part      AIC
- Sentiment           kernel 910.8282
<none>                       913.1769
- Views               kernel 913.9753
- Likes               kernel 917.6532
- Comments            kernel 921.2163
- Aggregate.Followers kernel 925.4491
- Dislikes            kernel 969.4304


Step: AIC = 910.8
Linear part: ~ 1
Kernel part: ~ Views + Likes + Dislikes + Comments + Aggregate.Followers

                        Part      AIC
- Views               kernel 905.2908
<none>                       910.8282
+ Sentiment           kernel 913.1769
- Aggregate.Followers kernel 915.5481
- Likes               kernel 916.8125
- Comments            kernel 921.9627
- Dislikes            kernel 970.3804


Step: AIC = 905.3
Linear part: ~ 1
Kernel part: ~ Likes + Dislikes + Comments + Aggregate.Followers

                        Part      AIC
<none>                       905.2908
+ Views               kernel 910.8282
+ Sentiment           kernel 913.9753
- Aggregate.Followers kernel 916.0224
- Comments            kernel 917.8758
- Likes               kernel 925.0230
- Dislikes            kernel 968.2502
```

The final model includes the variables Likes, Dislikes, Comments and Aggregate.Followers.

5 Example 2: Consumption of energy data

Our second example illustrates how **KSPM** may be efficient with complex data as time series and show how the choice of tuning parameters impacts the results.

The energy data is a set of data on energy consumption each hour on Ouessant island (France) from September the 13th, 2015 to October the 4th, 2015, that were made publicly available by *Electricité de*

France at https://iles-ponant-edf-sei.opendatasoft.com. The data set also contains corresponding meteorologic data such as temperature (Celsius degrees), pressure (Pa) and humidity rate (g/m^3). These measures are collected by *Meteo France* every 3 hours and are publicly available on www.infoclimat.fr. We obtained hourly values by linear interpolation. In total the data set contains 504 measurements.

```
> data("energy")
> head(energy)

     power       date        T        P       HR hour hour.num
1 526.1667 2015-09-13 12.43333 1007.933 81.66667  01h        1
2 495.0000 2015-09-13 12.36667 1007.167 82.33333  02h        2
3 446.1667 2015-09-13 12.30000 1006.400 83.00000  03h        3
4 365.8333 2015-09-13 12.30000 1005.833 82.66667  04h        4
5 341.0000 2015-09-13 12.30000 1005.267 82.33333  05h        5
6 352.3333 2015-09-13 12.30000 1004.700 82.00000  06h        6
```

These data demonstrate strong periodicity depending on time of day (Figure 4). Of note, if data had been collected for a period longer than one year, a second periodicity would be visible corresponding to seasons.

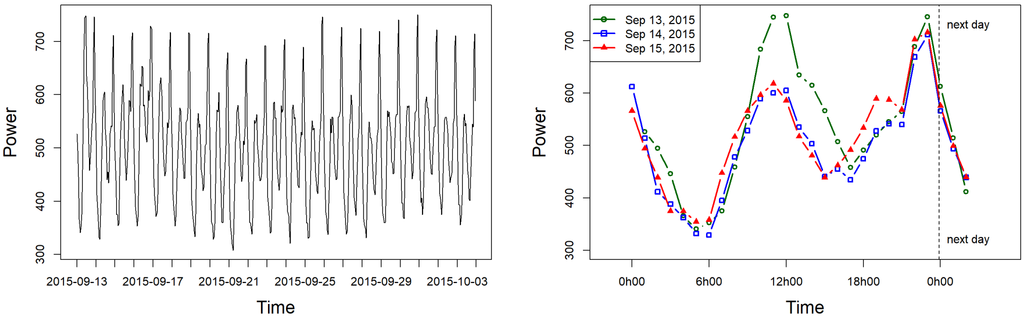

Figure 4: Pattern of energy consumption over the entire data set from Ouessant island (Left) and on the three first days (Right). The power is observed each hour and show a one day periodicity.

We will consider the 408 first measurements (17 days) as the training set. The others 4 days will be used as a test set, where we want to predict the energy consumption.

```
> energy_train_ <- energy[1:408, ]
> energy_test_ <- energy[409:504, ]
```

Modeling

We fit a single kernel semi-parametric model to the training data. We assume that energy depends linearly on temperature (T), and therefore this variable is included in the linear part of the model. The other meteorologic data, as well as hours in 24-hour numeric format, are included in the kernel part of the model. We used a gaussian kernel and we left the ρ parameter free to be estimated by the model.

```
> energy.fit1 <- kspm(response = "power", linear = ~T, kernel = ~Kernel(~
+    hour.num + P + HR, kernel.function = "gaussian") , data = energy_train_)
> energy.fit1$kernel.info$Ker1$rho

     par1
0.7028723
```

Impact of the ρ parameter on derivatives and predictions

We recomputed the model using other values for the ρ parameter to explore sensitivity of the results to this key kernel parameter. Values were chosen tenfold larger and smaller than the estimated value of 0.70 in energy.fit1.

```
> energy.fit2 <- kspm(response = "power", linear = ~T, kernel = ~Kernel(~
+    hour.num + P + HR, kernel.function = "gaussian", rho = 7) , data =
+    energy_train_)
> energy.fit3 <- kspm(response = "power", linear = ~T, kernel = ~Kernel(~
+    hour.num +  P + HR, kernel.function = "gaussian", rho = 0.07) , data =
+    energy_train_)
```

Figure 5 displays the predictions obtained on both the training and test data sets, as well as the derivatives, as a function of the hours variable, for the three models `energy.fit1`, `energy.fit2` and `energy.fit3` that differ only on the value of the tuning parameter ρ.

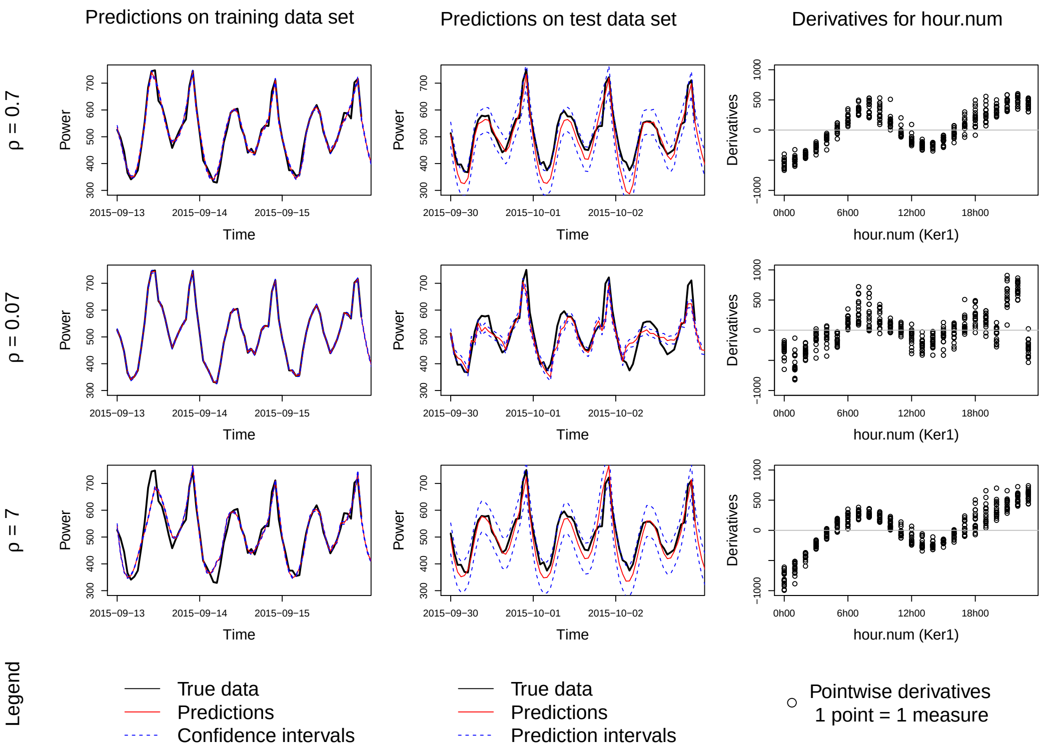

Figure 5: Predictions and derivatives obtained for different values of ρ in the energy dataset. Predictions on training data set are displayed with confidence intervals, whereas predictions on test data set are displayed with prediction intervals. The first row corresponds to the model `energy.fit1` with estimated $\rho = 0.7$. The second row corresponds to a tenfold smaller $\rho = 0.07$ (overfitting) and third row corresponds to a tenfold larger $\rho = 7$ (underfitting). The last column shows how the choice of ρ impacts the derivatives.

The first row of Figure 5 corresponds to the model with the value of ρ estimated by the model. Predictions fit well for both the training and the test sets. The derivative graph shows that between 6h00 and 12h00 and between 18h00 and 23h00 the derivatives are positive. This result is coherent with the increase of energy consumption during these times of the day (Figure 4-Right). Inversely between 0h00 and 6h00 and between 12h00 and 18h00, the energy consumption decreases as showed by negative values of derivatives. One might have expected that derivative values should be close between 0h00 and 23h00 but the peak consumption observed at 23h00 everyday (Figure 4) is probably associated with a sudden change in derivative values that is difficult to smooth accurately.

This example shows how a gaussian kernel function may handle complex functional data. However the choice of the ρ parameter may influence the results. A higher ρ leads to a smoother approximation of $h(.)$ and worsens the predictions. Indeed, a higher ρ may perfectly fit the training data set, but this corresponds to a case of model overfitting, which results in biased predictions in the test set. In contrast, a lower ρ may lead to underfitting of training and test data sets. The choice of the ρ parameter also impacts the derivatives; indeed a higher ρ induces more noise among the derivatives.

6 Example 3: Gene-gene interaction

This example shows how the interaction test may be applied to gene-gene interaction using standard linear kernel and SNPs involved in genes.

We simulated a data of 300 subjects and 9 SNPs, 6 belonging to gene A and 3 belonging to gene B, using the `glSim()` function into the **adegenet** R package (Jombart and Ahmed, 2011). SNPs are coded 0, 1 or 2 according to the number of alleles of each type carried by the subject. A continuous outcome was simulated as a linear combination of the interaction terms between the 6 SNPs of gene A and the three SNPs of gene B, where the weight coefficients where randomly and uniformly chosen in the interval $[-2; 0]$. We added an intercept of 100 and a normally distributed error of mean 0 and standard deviation 15. Code details are available in Supplement S1.

Let `y` be the vector of continuous outcomes, `geneA` be the 300 × 6 matrix of SNPs belonging to gene A and `geneB` be the 300 × 3 matrix of SNPs belonging to gene B.

We test the interaction between the genes using the code below. Of note, in this example, the data are specified using vector and design matrices instead of using formulae and data frame as in previous examples.

```
> gene.fit <- kspm(response = y, kernel = ~ Kernel(geneA, kernel = "linear")
+    * Kernel(geneB, kernel = "linear"))
> summary(gene.fit, kernel.test = "Ker1:Ker2")

Call:
kspm(response = y, kernel = ~Kernel(geneA, kernel = "linear") *
    Kernel(geneB, kernel = "linear"))

Sample size:
n = 300

Residuals:
     Min       Q1   Median       Q3      Max
-34.1111  -8.7300  -1.1659   9.7299  38.6887

Coefficients (linear part):
            Estimate Std. Error  t value      Pr(>|t|)
(Intercept) 88.80183  0.8178659 108.5775 2.531469e-231

Score test for non-parametric kernel:
            lambda       tau     p-value
Ker1:Ker2 199.4587 0.9862838 0.006646781

Residual standard error: 14.03 on 280.74 effective degrees of freedom
Multiple R-squared: 0.2387, Adjusted R-squared: 0.1892
```

The result suggests that genes A and B impact the continuous outcome y through their interaction.

7 Summary

This new **KSPM** package provides a flexible implementation of the kernel semi-parametric model and its extensions in a unified framework, using nomenclatures similar to other regression R packages. Thanks to the kernel trick, such a model is useful when interest lies in prediction and our `predict()` command makes this easy. Nevertheless, inference is also possible through the confidence intervals and tests provided by the package, such as through the `summary()` command. Moreover, we have provided many options for model diagnostics (residuals, leverage, ...), model interpretation (derivatives, ...) and model comparisons (AIC, stepwise, ...).

Model estimation for multiple kernel model is based on an iterative estimation of regression parameters we develop and prove in Supplement S2. Penalization and tuning parameter estimation involves `optimize()` and `DEoptim()` functions. The both integrate C code ensuring a faster convergence of the algorithms. However, the overall **KSPM** algorithm includes inversion of $n \times n$ matrices, thereby resulting in slower optimization as n increases. It would improve **KSPM**'s performance to combine matrix inversion code and optimization code in a single efficient set of C code and we are considering this for a future implementation. It is worth noting, however, that computation time is not severely impacted by the number of predictors even if $n << q$.

In the **KRLS** package, Hainmueller and Hazlett (2013) proposed a marginal test for the individual effect of each variable included in the kernel. We decided not to implement this test in our package since the marginal effect may mask interesting but symmetric effects of a variable on the continuous outcome. However, we have implemented a test of hypothesis based on variance components to test the joint effect of a set of variables on the continuous outcome. The distribution of this test statistic follows a mixture of χ^2 distributions that we approximated using Davies' method (Davies, 1980), available through the **CompQuadForm** package (de Micheaux, 2017; Duchesne and de Micheaux, 2010). The distribution can also be approximated by a scaled chi-squared distribution using Satterthwaite's method where parameters are computed by moment matching. Schifano et al. (2012) compared the two approximation methods on the type-I error rate in a single kernel semi-parametric model and showed that the rate is inflated with Satterthwaite's method when the α-level is low, and therefore we recommend the Davies method. To assess global testing of multiple kernels, we implemented a sum of the L single-kernel test statistics as an overall test statistic. In general, the computation of the p value of this overall test involves the joint distribution of $(Q_1, \dots Q_L)$. To derive analytical values for the overall p value, one can use techniques similar to those used in Sun et al. (2019), which relied on a copula-based model to approximate this joint distribution. This last option is not yet implemented in **KSPM**.

For all our tests, the model has to be re-estimated under the null hypothesis for the kernel of interest. If the resulting null model still contains one or more kernel part(s), we made the choice to recompute penalization parameter(s), since this choice ensures an optimized model under the null hypothesis. However, we have also decided to leave the kernel tuning parameters fixed when re-estimating under the null. Indeed, a change in tuning parameters induces a change in the choice of kernel functions - and thus in model assumptions - and hence, keeping the tuning parameters fixed ensures comparability of model assumptions under the null and the alternative hypotheses.

In our **KSPM** package, we have also included an algorithm for selection of variables based on these information criteria. Backward, forward or stepwise approaches can be chosen for single kernel semi-parametric models. Although these concepts may be easily extended to the case of multiple kernel models, such analyses require large computing times. Parallelization of such a process may greatly increase the appeal of this procedure. For now, we recommend investigating variables within a single kernel before starting to fit multiple kernel models.

In summary, the **KSPM** package is a flexible comprehensible option for studying the effects of groups of variables on a continuous outcome. These tools may be extended to the case of binary or survival outcomes in future work.

8 Acknowledgements

CS was supported by the Institute for Data Valorization (IVADO) fellowship. CG was supported by the Ludmer Centre for Neuroinformatics and Mental Health, the Canadian Institutes for Health Research, as well as the CIHR PJT 148620 grant. SJ is a recipient of a Canada Research Chair in neurodevelopmental disorders, and a chair from the Jeanne et Jean Louis Levesque Foundation. This research was enabled in part by support provided by Calcul Quebec (http://www.calculquebec.ca) and Compute Canada (www.computecanada.ca). This work is supported by a grant from the Brain Canada Multi-Investigator initiative and CIHR grant 159734 (SJ, CMTG). This work was also supported by an NIH award U01 MH119690 granted (SJ) and U01 MH119739.

Bibliography

M. Ahmed, M. Jahangir, H. Afzal, A. Majeed, and I. Siddiqi. Using crowd-source based features from social media and conventional features to predict the movies popularity. In *IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity)*, pages 273–278. IEEE, 2015. URL https://doi.org/10.1109/SmartCity.2015.83. [p89]

D. Ardia, K. Mullen, B. Peterson, and J. Ulrich. *DEoptim: Global Optimization by Differential Evolution*, 2020. URL https://CRAN.R-project.org/package=DEoptim. R package version 2.2-5. [p85]

D. Bates, M. Maechler, B. Bolker, and S. Walker. *lme4: Linear Mixed-Effects Models using 'Eigen' and S4*, 2019. URL https://CRAN.R-project.org/package=lme4. R package version 1.1-21. [p84]

R. P. Brent. *Algorithms for Minimization without Derivatives*. Prentice-Hall, Englewood Clifts, New Jersey, 2013. [p85]

J. Chen, W. Chen, N. Zhao, M. C. Wu, and D. J. Schaid. Small sample kernel association tests for human genetic and microbiome association studies. *Genetic epidemiology*, 40(1):5–19, 2016. URL https://doi.org/10.1002/gepi.21934. [p83]

R. B. Davies. Distribution of a linear combination of chi-square random variables: Algorithm as 155. *Applied Statistics*, 29(3):323–33, 1980. [p86, 99]

P. L. de Micheaux. *CompQuadForm: Distribution Function of Quadratic Forms in Normal Variables*, 2017. URL https://CRAN.R-project.org/package=CompQuadForm. R package version 1.4.3. [p99]

P. Duchesne and P. L. de Micheaux. Computing the distribution of quadratic forms: Further comparisons between the Liu-Tang-Zhang approximation and exact methods. *Computational Statistics and Data Analysis*, 54:858–862, 2010. URL https://doi.org/10.1016/j.csda.2009.11.025. [p99]

T. Ge, T. E. Nichols, D. Ghosh, E. C. Mormino, J. W. Smoller, M. R. Sabuncu, and the Alzheimer's Disease Neuroimaging Initiative. A kernel machine method for detecting effects of interaction between multidimensional variable sets: An imaging genetics application. *Neuroimage*, 109:505–514, 2015. URL https://doi.org/10.1016/j.neuroimage.2015.01.029. [p83, 85]

J. Hainmueller and C. Hazlett. Kernel regularized least squares: Reducing misspecification bias with a flexible and interpretable machine learning approach. *Political Analysis*, 22(2):143–168, 2013. [p83, 86, 98]

J. Hainmueller and C. Hazlett. *KRLS: Kernel-Based Regularized Least Squares*, 2017. URL https://CRAN.R-project.org/package=KRLS. R package version 1.0-0. [p83]

T. Jombart and I. Ahmed. Adegenet 1.3-1: New tools for the analysis of genome-wide SNP data. *Bioinformatics*, 2011. URL https://doi.org/10.1093/bioinformatics/btr521. [p98]

G. Kimeldorf and G. Wahba. Some results on Tchebycheffian spline functions. *Journal of mathematical analysis and applications*, 33(1):82–95, 1971. URL https://doi.org/10.1016/0022-247X(71)90184-3. [p85]

S. Lee, with contributions from Larisa Miropolsky, and M. Wu. *SKAT: SNP-Set (Sequence) Kernel Association Test*, 2017. URL https://CRAN.R-project.org/package=SKAT. R package version 1.3.2.1. [p83]

S. Li and Y. Cui. *SPA3G: R Package for the Method of Li and Cui (2012)*, 2012a. URL https://CRAN.R-project.org/package=SPA3G. R package version 1.0. [p83]

S. Li and Y. Cui. Gene-centric gene-gene interaction: A model-based kernel machine method. *The Annals of Applied Statistics*, 6(3):1134–1161, 2012b. URL https://doi.org/10.1214/12-AOAS545. [p83]

D. Liu, X. Lin, and D. Ghosh. Semiparametric regression of multidimensional genetic pathway data: Least-squares kernel machines and linear mixed models. *Biometrics*, 63(4):1079–1088, 2007. URL https://doi.org/10.1111/j.1541-0420.2007.00799.x. [p82, 85, 86, 105]

R. Marceau, W. Lu, S. Holloway, M. M. Sale, B. B. Worrall, S. R. Williams, F.-C. Hsu, and J.-Y. Tzeng. A fast multiple-kernel method with applications to detect gene-environment interaction. *Genetic epidemiology*, 39(6):456–468, 2015. URL https://doi.org/10.1002/gepi.21909. [p83]

D. Meyer, E. Dimitriadou, K. Hornik, A. Weingessel, and F. Leisch. *e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien*, 2018. URL https://CRAN.R-project.org/package=e1071. R package version 1.7-0. [p83]

K. M. Mullen, D. Ardia, D. L. Gil, D. Windover, and J. Cline. Deoptim: An R package for global optimization by differential evolution. *Journal of Statistical Software*, 40(6):1–26, 2009. URL http://www.jstatsoft.org/v40/i06/. [p85]

K. Oualkacha, Z. Dastani, R. Li, P. E. Cingolani, T. D. Spector, C. J. Hammond, J. B. Richards, A. Ciampi, and C. M. Greenwood. Adjusted sequence kernel association test for rare variants controlling for cryptic and family relatedness. *Genetic epidemiology*, 37(4):366–376, 2013. URL https://doi.org/10.1002/gepi.21725. [p83]

J. Pinheiro, D. Bates, and R-core. *nlme: Linear and Nonlinear Mixed Effects Models*, 2019. URL https://CRAN.R-project.org/package=nlme. R package version 3.1-140. [p84]

J. S. Racine and T. Hayfield. *np: Nonparametric Kernel Smoothing Methods for Mixed Data Types*, 2020. URL https://CRAN.R-project.org/package=np. R package version 0.60-10. [p84]

E. D. Schifano, M. P. Epstein, L. F. Bielak, M. A. Jhun, S. L. Kardia, P. A. Peyser, and X. Lin. SNP set association analysis for familial data. *Genetic epidemiology*, 36(8):797–810, 2012. URL https://doi.org/10.1002/gepi.21676. [p99]

C. Schramm. *KSPM: Kernel Semi-Parametric Models*, 2020. URL https://CRAN.R-project.org/package=KSPM. R package version 0.2.1. [p83]

J. Sun, K. Oualkacha, C. M. Greenwood, and L. Lakhal-Chaieb. Multivariate association test for rare variants controlling for cryptic and family relatedness. *Canadian Journal of Statistics*, 47(1):90–107, 2019. URL https://doi.org/10.1002/cjs.11475. [p99]

T. M. Therneau. *coxme: Mixed Effects Cox Models*, 2018. URL https://CRAN.R-project.org/package=coxme. R package version 2.2-10. [p83]

C. Wang, J. Sun, B. Guillaume, T. Ge, D. P. Hibar, C. M. Greenwood, A. Qiu, and the Alzheimer's Disease Neuroimaging Initiative. A set-based mixed effect model for gene-environment interaction and its application to neuroimaging phenotypes. *Frontiers in neuroscience*, 11:191, 2017. URL https://doi.org/10.3389/fnins.2017.00191. [p83]

S. Wood. *mgcv: Mixed GAM Computation Vehicle with Automatic Smoothness Estimation*, 2020. URL https://CRAN.R-project.org/package=mgcv. R package version 1.8-33. [p84]

M. C. Wu, S. Lee, T. Cai, Y. Li, M. Boehnke, and X. Lin. Rare-variant association testing for sequencing data with the sequence kernel association test. *The American Journal of Human Genetics*, 89(1):82–93, 2011. URL https://doi.org/10.1016/j.ajhg.2011.05.029. [p83]

D. Zhang and X. Lin. Hypothesis testing in semiparametric additive mixed models. *Biostatistics*, 4(1):57–74, 2003. URL https://doi.org/10.1093/biostatistics/4.1.57. [p86]

SUPPLEMENT

9 Supplement S1

We simulated a 300 × 9 matrix representing the number of minor allele (0, 1 or 2) of 300 subjects for 9 SNPs, 6 belonging to gene A and 3 belonging to gene B, using the `glSim()` function into the **adegenet** R package.

```
> library(adegenet)
> set.seed(78)
> SNPdata <- as.matrix(glSim(n.ind = 300, n.snp.nonstruc = 9, n.snp.struc = 0, k = 1,
  LD = FALSE, ploidy = 2))
```

We may deduce two matrices, one with data belonging to gene A and one with data belonging to gene B.

```
> geneA <- SNPdata[,1:6]
> geneB <- SNPdata[,7:9]
```

Then we computed the matrix of interactions between SNPs from gene A and SNPs from geneB.

```
> SNPdataAB <- matrix(NA, nrow = 300, ncol = 6*3)
> k <- 1
> for (i in 1:6) {
    for (j in 1:3) {
      SNPdataAB[,k] <- SNPdataA[,i] * SNPdataB[,j]
      k <- k+1
    }
  }
```

A continuous outcome was simulated as a linear combination of the interaction terms between the 6 SNPs of gene A and the three SNPs of gene B, where the weight coefficients where randomly and uniformly chosen in the interval $[-2;0]$ using the following code.

```
> beta <- as.matrix(runif(6*3,-2,0), nrow = 300)
> y <- 100 + SNPdataAB %*% beta + rnorm(n, 0, 15)
```

Of note, we added an intercept of 100 and a normally distributed error of mean 0 and standard deviation 15.

10 Supplement S2

Estimating the kernel semi-parametric model parameters consists in estimating $\alpha_1, ..., \alpha_L$ and β. Then, estimators of $h_1(.), ..., h_L(.)$ are deduced from $\hat{\alpha}_1, ..., \hat{\alpha}_L$. When there are multiple kernels, estimation is iterative and so we proceed as follows. Partial derivatives of $l(\beta, h)$ according to $\alpha_1, ..., \alpha_L$ and β lead to the following equations:

$$\hat{\beta} = \left\{ X^\top \left(I - \sum_{\ell=1}^{L} K_\ell G_\ell^{-1} M_\ell \right) X \right\}^{-1} X^\top \left(I - \sum_{\ell=1}^{L} K_\ell G_\ell^{-1} M_\ell \right) Y \quad (9)$$

$$\forall \ell \in \{1, ..., L\}, \quad \hat{\alpha}_\ell = G_\ell^{-1} M_\ell \left(Y - X\hat{\beta} \right) \quad (10)$$

where I is the $n \times n$ identity matrix and $\forall \ell \in \{1, ..., L\}$, G_ℓ and M_ℓ are computed using an iterative process as described below:

- Re-order elements $\alpha_{1...L}$, $K_{1...L}$ and $\lambda_{1...L}$ such that the last is the ℓ^{th} element
- Note $\alpha_{(1)}$, ..., $\alpha_{(L)}$, $K_{(1)}$, ..., $K_{(L)}$ and $\lambda_{(1)}$, ..., $\lambda_{(L)}$ the re-ordered elements with $\alpha_{(L)} = \alpha_\ell$, $K_{(L)} = K_\ell$ and $\lambda_{(L)} = \lambda_\ell$
- Define iteratively $\mathcal{M}_{(0)...(L)}$ and $\mathcal{G}_{(1)...(L)}$ by $\mathcal{M}_{(0)} = I$ the $n \times n$ identity matrix and $\forall m \in \{1, ..., L\}$:
  $\mathcal{G}_{(m)} = \lambda_{(m)} I + \mathcal{M}_{(m-1)} K_{(m)}$
  $\mathcal{M}_{(m)} = \left(I - \mathcal{M}_{(m-1)} K_{(m)} \mathcal{G}_{(m)}^{-1} \right) \mathcal{M}_{(m-1)}$

- Compute G_ℓ and M_ℓ as:
  $G_\ell = \mathcal{G}_{(L)}$
  $M_\ell = \mathcal{M}_{(L-1)}$

Whereas the parameter estimation for single kernel model has been largely demonstrated, the equations for computing multiple kernel parameters were not theoretically developed yet. Above, we propose an iterative way to estimate the parameters for all kernel models. Below, we propose the proof for equations (9) and (10).

Proof:

Partial derivatives of penalized likelihood (6) lead to equations below:

$$\begin{cases} X^\top (Y - X\beta - \sum_{\ell=1}^{L} K_\ell \alpha_\ell) = 0 \\ Y - X\beta - \sum_{m=1}^{L} K_m \alpha_m - \lambda_\ell \alpha_\ell = 0 \quad \forall \ell \in \{1, ..., L\} \end{cases} \tag{11}$$

Proposition 1: $\forall k \in \{1, ..., L-1\}\,, \alpha_{(k)} = \mathcal{G}_{(k)}^{-1} \mathcal{M}_{(k-1)} \left(Y - X\beta - \sum_{m=k+1}^{L} K_{(m)} \alpha_{(m)} \right)$

Proposition 2: if proposition 1 is true until k, then

$$\sum_{j=1}^{k} K_{(m)} \alpha_{(m)} = (I - \mathcal{M}_{(k)}) \left(Y - X\beta - \sum_{m=k+1}^{L} K_{(m)} \alpha_{(m)} \right)$$

(Proofs of propositions 1 and 2 are given below.)

Now, suppose proposition 1 and proposition 2 are true.

$$\begin{aligned} (11) \Rightarrow \quad & Y - X\beta - \sum_{m=1}^{L} K_{(m)} \alpha_{(m)} - \lambda_{(L)} \alpha_{(L)} = 0 \\ \Leftrightarrow \quad & Y - X\beta - \sum_{m=1}^{L-1} K_{(m)} \alpha_{(m)} - K_{(L)} \alpha_{(L)} - \lambda_{(L)} \alpha_{(L)} = 0 \\ \Leftrightarrow \quad & Y - X\beta - (I - \mathcal{M}_{(L-1)}) \left(Y - X\beta - K_{(L)} \alpha_{(L)} \right) - K_{(L)} \alpha_{(L)} - \lambda_{(L)} \alpha_{(L)} = 0 \\ \Leftrightarrow \quad & \left(\lambda_{(L)} I + \mathcal{M}_{(L-1)} K_{(L)} \right) \alpha_{(L)} - \mathcal{M}_{(L-1)} (Y - X\beta) = 0 \end{aligned}$$

Then, $\alpha_\ell = \alpha_{(L)} = \mathcal{G}_{(L)}^{-1} \mathcal{M}_{(L-1)} \, (Y - X\beta)$. Let G_ℓ and M_ℓ the $n \times n$ matrix defined as $G_\ell = \mathcal{G}_{(L)}$ and $M_\ell = \mathcal{M}_{(L-1)}$, thus $\alpha_\ell = G_\ell^{-1} M_\ell \, (Y - X\beta)$. All this process is done for each $\ell \in \{1, ..., L\}$.

Now, we can derive the estimate for β.

$$\begin{aligned} (11) \Rightarrow \quad & X^\top \left(Y - X\beta - \sum_{\ell=1}^{L} K_\ell \alpha_\ell \right) = 0 \\ \Leftrightarrow \quad & X^\top \left(Y - X\beta - \sum_{\ell=1}^{L} K_\ell G_\ell^{-1} M_\ell (Y - X\beta) \right) = 0 \\ \Leftrightarrow \quad & X^\top \left(I - \sum_{\ell=1}^{L} K_\ell G_\ell^{-1} M_\ell \right) Y = X^\top \left(I - \sum_{\ell=1}^{L} K_\ell G_\ell^{-1} M_\ell \right) X\beta \end{aligned}$$

Thus $\beta = \left\{ X^\top \left(I - \sum_{\ell=1}^{L} K_\ell G_\ell^{-1} M_\ell \right) X \right\}^{-1} X^\top \left(I - \sum_{\ell=1}^{L} K_\ell G_\ell^{-1} M_\ell \right) Y.$

Proof of proposition 1:

- Proposition 1 is true for $k = 1$:

$$\begin{aligned}
(11) \Rightarrow \quad & Y - X\beta - \sum_{m=1}^{L} K_{(m)}\alpha_{(m)} - \lambda_{(1)}\alpha_{(1)} = 0 \\
\Leftrightarrow \quad & Y - X\beta - \sum_{m=2}^{L} K_{(m)}\alpha_{(m)} - K_{(1)}\alpha_{(1)} - \lambda_{(1)}\alpha_{(1)} = 0 \\
\Leftrightarrow \quad & \mathcal{M}_{(0)} \left(Y - X\beta - \sum_{m=2}^{L} K_{(m)}\alpha_{(m)} \right) = \left(\lambda_{(1)} I + \mathcal{M}_{(0)} K_{(1)} \right) \alpha_{(1)} \\
\Leftrightarrow \quad & \mathcal{G}_{(1)}^{-1} \mathcal{M}_{(0)} \left(Y - X\beta - \sum_{m=2}^{L} K_{(m)}\alpha_{(m)} \right) = \alpha_{(1)}
\end{aligned}$$

- If proposition 1 is true for k, then proposition 1 is true for $k+1$:

According to proposition 2:

$$\begin{aligned}
\sum_{m=1}^{k} K_{(m)}\alpha_{(m)} &= (I - \mathcal{M}_{(k)}) \left(Y - X\beta - \sum_{m=k+1}^{L} K_{(m)}\alpha_{(m)} \right) \\
&= (I - \mathcal{M}_{(k)}) \left(Y - X\beta - \sum_{m=k+2}^{L} K_{(m)}\alpha_{(m)} \right) - (I - \mathcal{M}_{(k)}) K_{(k+1)}\alpha_{(k+1)}
\end{aligned}$$

$$\begin{aligned}
(11) \Rightarrow \quad & Y - X\beta - \sum_{m=1}^{L} K_{(m)}\alpha_{(m)} - \lambda_{(k+1)}\alpha_{(k+1)} = 0 \\
\Leftrightarrow \quad & Y - X\beta - \sum_{m=1}^{k} K_{(m)}\alpha_{(m)} - K_{(k+1)}\alpha_{(k+1)} - \sum_{m=k+2}^{L} K_{(m)}\alpha_{(m)} - \lambda_{(k+1)}\alpha_{(k+1)} = 0 \\
\Leftrightarrow \quad & -\left((I - \mathcal{M}_{(k)}) \left(Y - X\beta - \sum_{m=k+2}^{L} K_{(m)}\alpha_{(m)} \right) - (I - \mathcal{M}_{(k)}) K_{(k+1)}\alpha_{(k+1)} \right) \\
& \quad + Y - X\beta - K_{(k+1)}\alpha_{(k+1)} - \sum_{m=k+2}^{L} K_{(m)}\alpha_{(m)} - \lambda_{(k+1)}\alpha_{(k+1)} = 0 \\
\Leftrightarrow \quad & \mathcal{M}_{(k)} \left(Y - X\beta - \sum_{m=k+2}^{L} K_{(m)}\alpha_{(m)} \right) - \mathcal{M}_{(k)} K_{(k+1)}\alpha_{(k+1)} - \lambda_{(k+1)}\alpha_{(k+1)} = 0 \\
\Leftrightarrow \quad & \mathcal{M}_{(k)} \left(Y - X\beta - \sum_{m=k+2}^{L} K_{(m)}\alpha_{(m)} \right) - \mathcal{G}_{(k+1)}\alpha_{(k+1)} = 0 \\
\Leftrightarrow \quad & \mathcal{G}_{(k+1)}^{-1} \mathcal{M}_{(k)} \left(Y - X\beta - \sum_{m=k+2}^{L} K_{(m)}\alpha_{(m)} \right) = \alpha_{(k+1)}
\end{aligned}$$

Proof of proposition 2:

- Proposition 2 is true for $k = 1$:

Suppose $\alpha_{(1)} = \mathcal{G}_{(1)}^{-1} \mathcal{M}_{(0)} \left(Y - X\beta - \sum_{m=2}^{L} K_{(m)}\alpha_{(m)} \right).$

$$\begin{aligned}
\sum_{m=1}^{1} K_{(m)}\alpha_{(m)} &= K_{(1)}\alpha_{(1)} \\
&= K_{(1)}\mathcal{G}_{(1)}^{-1}\mathcal{M}_{(0)}\left(Y - X\beta - \sum_{m=2}^{L} K_{(m)}\alpha_{(m)}\right) \\
&= \left(I - \mathcal{M}_{(0)} + \mathcal{M}_{(0)}K_{(1)}G_1^{-1}\mathcal{M}_{(0)}\right)\left(Y - X\beta - \sum_{m=2}^{L} K_{(m)}\alpha_{(m)}\right) \\
&= \left(I - (I - \mathcal{M}_{(0)}K_{(1)}G_1^{-1})\mathcal{M}_{(0)}\right)\left(Y - X\beta - \sum_{m=2}^{L} K_{(m)}\alpha_{(m)}\right) \\
&= \left(I - \mathcal{M}_{(1)}\right)\left(Y - X\beta - \sum_{m=2}^{L} K_{(m)}\alpha_{(m)}\right)
\end{aligned}$$

• If proposition 2 is true for k, then proposition 2 is true for $k+1$:

Suppose proposition 1 is true until $k+1$, then $\forall j \in \{1, ..., k+1\}$,

$$\alpha_{(j)} = \mathcal{G}_{(j)}^{-1}\mathcal{M}_{(j-1)}\left(Y - X\beta - \sum_{m=j+1}^{L} K_{(m)}\alpha_{(m)}\right)$$

$$\begin{aligned}
\sum_{m=1}^{k+1} K_{(m)}\alpha_{(m)} &= \sum_{m=1}^{k} K_{(m)}\alpha_{(m)} + K_{(k+1)}\alpha_{(k+1)} \\
&= (I - \mathcal{M}_{(k)})\left(Y - X\beta - \sum_{m=k+1}^{L} K_{(m)}\alpha_{(m)}\right) + K_{(k+1)}\alpha_{(k+1)} \\
&= (I - \mathcal{M}_{(k)})\left(Y - X\beta - \sum_{m=k+2}^{L} K_{(m)}\alpha_{(m)}\right) + \mathcal{M}_{(k)}K_{(k+1)}\alpha_{(k+1)} \\
&= (I - \mathcal{M}_{(k)})\left(Y - X\beta - \sum_{m=k+2}^{L} K_{(m)}\alpha_{(m)}\right) + \mathcal{M}_{(k)}K_{(k+1)}\mathcal{G}_{(k+1)}^{-1}\mathcal{M}_{(k)}\left(Y - X\beta - \sum_{m=k+2}^{L} K_{(m)}\alpha_{(m)}\right) \\
&= (I - \mathcal{M}_{(k)} + \mathcal{M}_k K_{(k+1)}\mathcal{G}_{(k+1)}^{-1}\mathcal{M}_k)\left(Y - X\beta - \sum_{m=k+2}^{L} K_{(m)}\alpha_{(m)}\right) \\
&= (I - (I + \mathcal{M}_{(k)}K_{(k+1)}\mathcal{G}_{(k+1)}^{-1})\mathcal{M}_{(k)})\left(Y - X\beta - \sum_{m=k+2}^{L} K_{(m)}\alpha_{(m)}\right) \\
&= (I - \mathcal{M}_{(k+1)})\left(Y - X\beta - \sum_{m=k+2}^{L} K_{(m)}\alpha_{(m)}\right)
\end{aligned}$$

□

Of note, for $L = 1$, equations (9) and (10) correspond to those provided in Liu et al. (2007).

Catherine Schramm
Research center, Ste Justine Hospital
Montreal university
Lady Davis Institute for Medical Research, Jewish General Hospital
Montreal
Canada
ORCID: 0000-0002-1185-8809
cath.schramm@gmail.com

Sébastien Jacquemont
Research center, Ste Justine Hospital

Montreal university
Montreal
Canada
ORCID: 0000-0001-6838-8767
sebastien.jacquemont@umontreal.ca

Karim Oualkacha
Department of Mathematics
Université du Quebec à Montreal
Canada
ORCID: 0000-0002-9911-079X
oualkacha.karim@uqam.ca

Aurélie Labbe
Department of decision sciences
HEC Montreal
Canada
aurelie.labbe@hec.ca

Celia M.T. Greenwood
Lady Davis Institute for Medical Research, Jewish General Hospital
Gerald Bronfman Department of Oncology, Department of Epidemiology, Biostatistics and Occupational Health, and Department of Human Genetics, McGill University
Montreal
Canada
ORCID: 0000-0002-2427-5696
celia.greenwood@mcgill.ca