

# Package ‘KRMM’

January 20, 2025

**Type** Package

**Title** Kernel Ridge Mixed Model

**Version** 1.0

**Author** Laval Jacquin [aut, cre]

**Maintainer** Laval Jacquin <jacquin.julien@gmail.com>

**Description** Solves kernel ridge regression, within the the mixed model framework, for the linear, polynomial, Gaussian, Laplacian and ANOVA kernels. The model components (i.e. fixed and random effects) and variance parameters are estimated using the expectation-maximization (EM) algorithm. All the estimated components and parameters, e.g. BLUP of dual variables and BLUP of random predictor effects for the linear kernel (also known as RR-BLUP), are available. The kernel ridge mixed model (KRMM) is described in Jacquin L, Cao T-V and Ahmadi N (2016) A Unified and Comprehensive View of Parametric and Kernel Methods for Genomic Prediction with Application to Rice. *Front. Genet.* 7:145. <doi:10.3389/fgene.2016.00145>.

**Depends** R (>= 3.3.0)

**Imports** stats,MASS,kernlab,cvTools,robustbase

**License** GPL-2 | GPL-3

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 5.0.1

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2017-06-03 17:46:04 UTC

## Contents

KRMM-package	2
EM_REML_MM	4
Kernel_Ridge_MM	5
Predict_kernel_Ridge_MM	8
Tune_kernel_Ridge_MM	10

<b>Index</b>	<b>14</b>
--------------	-----------

**Description**

Solves kernel ridge regression, within the the mixed model framework, for the linear, polynomial, Gaussian, Laplacian and ANOVA kernels. The model components (i.e. fixed and random effects) and variance parameters are estimated using the expectation-maximization (EM) algorithm. All the estimated components and parameters, e.g. BLUP of dual variables and BLUP of random predictor effects for the linear kernel (also known as RR-BLUP), are available. The kernel ridge mixed model (KRMM) is described in Jacquin L, Cao T-V and Ahmadi N (2016) A Unified and Comprehensible View of Parametric and Kernel Methods for Genomic Prediction with Application to Rice. *Front. Genet.* 7:145.

**Details**

This package solves kernel ridge regression for various kernels within the following mixed model framework:  $Y = X * \text{Beta} + Z * U + E$ , where X and Z correspond to the design matrices of predictors with fixed and random effects respectively. The functions provided with this package are `Kernel_Ridge_MM`, `Tune_kernel_Ridge_MM`, `Predict_kernel_Ridge_MM` and `EM_REML_MM`.

**Author(s)**

Laval Jacquin Maintainer: Laval Jacquin <jacquin.julien@gmail.com>

**References**

- Jacquin et al. (2016). A unified and comprehensible view of parametric and kernel methods for genomic prediction with application to rice (in peer review).
- Robinson, G. K. (1991). That blup is a good thing: the estimation of random effects. *Statistical science*, 534 15-32
- Foulley, J.-L. (2002). Algorithme em: théorie et application au modèle mixte. *Journal de la Société française de Statistique* 143, 57-109

**Examples**

```
## Not run:

library(KRMM)

### SIMULATE DATA
set.seed(123)
p=200
N=100

beta=rnorm(p, mean=0, sd=1.0)
X=matrix(runif(p*N, min=0, max=1), ncol=p, byrow=TRUE) #X: covariates (i.e. predictors)
```

```

f=X%*%beta          #f: data generating process (i.e. DGP)
E=rnorm(N, mean=0, sd=0.5)

Y=f+E              #Y: observed response data

hist(f)
hist(beta)
Nb_train=floor((2/3)*N)

###=====###
### CREATE TRAINING AND TARGET SETS FOR RESPONSE AND PREDICTOR VARIABLES ###
###=====###

Index_train=sample(1:N, size=Nb_train, replace=FALSE)

### Covariates (i.e. predictors) for training and target sets

Predictors_train=X[Index_train, ]
Response_train=Y[Index_train]

Predictors_target=X[-Index_train, ]
True_value_target=f[-Index_train] #True value (generated by DGP) we want to predict

###=====###
### PREDICTION WITH KERNEL RIDGE REGRESSION SOLVED WITHIN THE MIXED MODEL FRAMEWORK ###
###=====###

#Linear kernel

Linear_KRR_model_train = Kernel_Ridge_MM(Y_train=Response_train,
Matrix_covariates_train=Predictors_train, method="RR-BLUP")

f_hat_target_Linear_KRR = Predict_kernel_Ridge_MM( Linear_KRR_model_train,
Matrix_covariates_target=Predictors_target )

#Gaussian kernel

Gaussian_KRR_model_train = Kernel_Ridge_MM( Y_train=Response_train,
Matrix_covariates_train=Predictors_train, method="RKHS", rate_decay_kernel=5.0)

f_hat_target_Gaussian_KRR = Predict_kernel_Ridge_MM( Gaussian_KRR_model_train,
Matrix_covariates_target=Predictors_target )

#Graphics for RR-BLUP

dev.new(width=30, height=20)
par(mfrow=c(3,1))
plot(f_hat_target_Linear_KRR, True_value_target)
plot(Linear_KRR_model_train$Gamma_hat, xlab="Feature (i.e. covariate) number",
ylab="Feature effect (i.e. Gamma_hat)", main="BLUP of covariate effects based on training data")
hist(Linear_KRR_model_train$Gamma_hat, main="Distribution of BLUP of
covariate effects based on training data" )

```

```
# Compare prediction based on linear (i.e. RR-BLUP) and Gaussian kernel

dev.new(width=30, height=20)
par(mfrow=c(1,2))
plot(f_hat_target_Linear_KRR, True_value_target)
plot(f_hat_target_Gaussian_KRR, True_value_target)

mean((f_hat_target_Linear_KRR - True_value_target)^2)
mean((f_hat_target_Gaussian_KRR - True_value_target)^2)

## End(Not run)
```

---

EM_REML_MM	<i>Expectation-Maximization (EM) algorithm for the restricted maximum likelihood (REML) associated to the mixed model</i>
------------	---

---

### Description

EM\_REML\_MM estimates the components and variance parameters of the following mixed model;  $Y = X * \text{Beta} + Z * U + E$ , using the EM-REML algorithm.

### Usage

```
EM_REML_MM( Mat_K_inv, Y, X, Z, init_sigma2K,
            init_sigma2E, convergence_precision,
            nb_iter, display )
```

### Arguments

Mat_K_inv	numeric matrix; the inverse of the kernel matrix
Y	numeric vector; response vector
X	numeric matrix; design matrix of predictors with fixed effects
Z	numeric matrix; design matrix of predictors with random effects
init_sigma2K, init_sigma2E	numeric scalars; initial guess values, associated to the mixed model variance parameters, for the EM-REML algorithm
convergence_precision, nb_iter	convergence precision (i.e. tolerance) associated to the mixed model variance parameters, for the EM-REML algorithm, and number of maximum iterations allowed if convergence is not reached
display	boolean (TRUE or FALSE character string); should estimated components be displayed at each iteration

**Value**

Beta\_hat            Estimated fixed effect(s)  
 Sigma2K\_hat, Sigma2E\_hat  
                          Estimated variance components

**Author(s)**

Laval Jacquin <jacquin.julien@gmail.com>

**References**

Foulley, J.-L. (2002). Algorithme em: théorie et application au modèle mixte. Journal de la Société française de Statistique 143, 57-109

---

Kernel\_Ridge\_MM            *Kernel ridge regression in the mixed model framework*

---

**Description**

Kernel\_Ridge\_MM solves kernel ridge regression for various kernels within the following mixed model framework:  $Y = X \cdot \text{Beta} + Z \cdot U + E$ , where  $X$  and  $Z$  correspond to the design matrices of predictors with fixed and random effects respectively.

**Usage**

```
Kernel_Ridge_MM( Y_train, X_train=as.vector(rep(1,length(Y_train))),
  Z_train=diag(1,length(Y_train)), Matrix_covariates_train, method="RKHS",
  kernel="Gaussian", rate_decay_kernel=0.1, degree_poly=2, scale_poly=1,
  offset_poly=1, degree_anova=3, init_sigma2K=2, init_sigma2E=3,
  convergence_precision=1e-8, nb_iter=1000, display="FALSE" )
```

**Arguments**

Y\_train            numeric vector; response vector for training data  
 X\_train            numeric matrix; design matrix of predictors with fixed effects for training data  
                          (default is a vector of ones)  
 Z\_train            numeric matrix; design matrix of predictors with random effects for training  
                          data (default is identity matrix)  
 Matrix\_covariates\_train  
                          numeric matrix of entries used to build the kernel matrix  
 method            character string; RKHS, GBLUP or RR-BLUP

kernel	character string; Gaussian, Laplacian or ANOVA (kernels for RKHS regression ONLY, the linear kernel is automatically built for GBLUP and RR-BLUP and hence no kernel is supplied for these methods)
rate_decay_kernel	numeric scalar; hyperparameter of the Gaussian, Laplacian or ANOVA kernel (default is 0.1)
degree_poly, scale_poly, offset_poly	numeric scalars; parameters for polynomial kernel (defaults are 2, 1 and 1 respectively)
degree_anova	numeric scalar; parameter for ANOVA kernel (defaults is 3)
init_sigma2K, init_sigma2E	numeric scalars; initial guess values, associated to the mixed model variance parameters, for the EM-REML algorithm (defaults are 2 and 3 respectively)
convergence_precision, nb_iter	numeric scalars; convergence precision (i.e. tolerance) associated to the mixed model variance parameters, for the EM-REML algorithm, and number of maximum iterations allowed if convergence is not reached (defaults are 1e-8 and 1000 respectively)
display	boolean (TRUE or FALSE character string); should estimated components be displayed at each iteration

### Details

The matrix `Matrix_covariates_train` is mandatory to build the kernel matrix for model estimation, and prediction (see `Predict_kernel_Ridge_MM`).

### Value

Beta_hat	Estimated fixed effect(s)
Sigma2K_hat, Sigma2E_hat	Estimated variance components
Vect_alpha	Estimated dual variables
Gamma_hat	RR-BLUP of covariates effects (i.e. available for RR-BLUP method only)

### Author(s)

Laval Jacquin <jacquin.julien@gmail.com>

### References

- Jacquin et al. (2016). A unified and comprehensible view of parametric and kernel methods for genomic prediction with application to rice (in peer review).
- Robinson, G. K. (1991). That blup is a good thing: the estimation of random effects. *Statistical science*, 534 15-32
- Foulley, J.-L. (2002). Algorithme em: théorie et application au modèle mixte. *Journal de la Société française de Statistique* 143, 57-109

**Examples**

```

## Not run:

library(KRMM)

### SIMULATE DATA
set.seed(123)
p=200
N=100

beta=rnorm(p, mean=0, sd=1.0)
X=matrix(runif(p*N, min=0, max=1), ncol=p, byrow=TRUE) #X: covariates (i.e. predictors)

f=X%*%beta #f: data generating process (i.e. DGP)
E=rnorm(N, mean=0, sd=0.5)

Y=f+E #Y: observed response data

hist(f)
hist(beta)
Nb_train=floor((2/3)*N)

#####
### CREATE TRAINING AND TARGET SETS FOR RESPONSE AND PREDICTOR VARIABLES ###
#####

Index_train=sample(1:N, size=Nb_train, replace=FALSE)

### Covariates (i.e. predictors) for training and target sets

Predictors_train=X[Index_train, ]
Response_train=Y[Index_train]

Predictors_target=X[-Index_train, ]
True_value_target=f[-Index_train] #True value (generated by DGP) we want to predict

#####
### PREDICTION WITH KERNEL RIDGE REGRESSION SOLVED WITHIN THE MIXED MODEL FRAMEWORK ###
#####

#Linear kernel

Linear_KRR_model_train = Kernel_Ridge_MM(Y_train=Response_train,
Matrix_covariates_train=Predictors_train, method="RR-BLUP")

f_hat_target_Linear_KRR = Predict_kernel_Ridge_MM( Linear_KRR_model_train,
Matrix_covariates_target=Predictors_target )

#Gaussian kernel

```

```

Gaussian_KRR_model_train = Kernel_Ridge_MM( Y_train=Response_train,
Matrix_covariates_train=Predictors_train, method="RKHS", rate_decay_kernel=5.0)
f_hat_target_Gaussian_KRR = Predict_kernel_Ridge_MM( Gaussian_KRR_model_train,
Matrix_covariates_target=Predictors_target )

#Graphics for RR-BLUP

dev.new(width=30, height=20)
par(mfrow=c(3,1))
plot(f_hat_target_Linear_KRR, True_value_target)
plot(Linear_KRR_model_train$Gamma_hat, xlab="Feature (i.e. covariate) number",
ylab="Feature effect (i.e. Gamma_hat)", main="BLUP of covariate effects based on training data")
hist(Linear_KRR_model_train$Gamma_hat, main="Distribution of BLUP of
covariate effects based on training data" )

# Compare prediction based on linear (i.e. RR-BLUP) and Gaussian kernel

par(mfrow=c(1,2))
plot(f_hat_target_Linear_KRR, True_value_target)
plot(f_hat_target_Gaussian_KRR, True_value_target)

mean((f_hat_target_Linear_KRR - True_value_target)^2)
mean((f_hat_target_Gaussian_KRR - True_value_target)^2)

## End(Not run)

```

---

Predict\_kernel\_Ridge\_MM

*Predict function for Kernel\_Ridge\_MM object*

---

### **Description**

Predict the value(s) for a vector or a design matrix of covariates (i.e. features)

### **Usage**

```

Predict_kernel_Ridge_MM( Model_kernel_Ridge_MM, Matrix_covariates_target,
X_target=as.vector(rep(1,dim(Matrix_covariates_target)[1])),
Z_target=diag(1,dim(Matrix_covariates_target)[1]) )

```



**Arguments**

Model_kernel_Ridge_MM	a Kernel_Ridge_MM object
Matrix_covariates_target	numeric matrix; design matrix of covariates for target data
X_target	numeric matrix; design matrix of predictors with fixed effects for target data (default is a vector of ones)
Z_target	numeric matrix; design matrix of predictors with random effects for target data (default is identity matrix)

**Details**

The matrix Matrix\_covariates\_target is mandatory to build the kernel matrix (with Matrix\_covariates\_train from Model\_kernel\_Ridge\_MM) for prediction.

**Value**

f_hat	Predicted value for target data, i.e. $f\_hat = X\_target * Beta\_hat + Z\_target * U\_target$ where $U\_target = K\_target\_train * alpha\_train$ and alpha_train is the BLUP of alpha for the model, i.e. $alpha\_train = Cov(alpha, Y\_train) * Var(Y\_train)^{-1} * (Y\_train - E[Y\_train])$
-------	---

**Author(s)**

Laval Jacquin <jacquin.julien@gmail.com>

**Examples**

```
## Not run:

library(KRMM)

### SIMULATE DATA
set.seed(123)
p=200
N=100

beta=rnorm(p, mean=0, sd=1.0)
X=matrix(runif(p*N, min=0, max=1), ncol=p, byrow=TRUE) #X: covariates (i.e. predictors)

f=X%%beta #f: data generating process (i.e. DGP)
E=rnorm(N, mean=0, sd=0.5)

Y=f+E #Y: observed response data

hist(f)
hist(beta)
Nb_train=floor((2/3)*N)

###=====###
```

```

### CREATE TRAINING AND TARGET SETS FOR RESPONSE AND PREDICTOR VARIABLES ###
#####

Index_train=sample(1:N, size=Nb_train, replace=FALSE)

### Covariates (i.e. predictors) for training and target sets

Predictors_train=X[Index_train, ]
Response_train=Y[Index_train]

Predictors_target=X[-Index_train, ]
True_value_target=f[-Index_train] #True value (generated by DGP) we want to predict

#####
### PREDICTION WITH KERNEL RIDGE REGRESSION SOLVED WITHIN THE MIXED MODEL FRAMEWORK ###
#####

Gaussian_KRR_model_train = Kernel_Ridge_MM( Y_train=Response_train,
Matrix_covariates_train=Predictors_train, method="RKHS", rate_decay_kernel=5.0)

### Predict new entries for target set and measure prediction error

f_hat_target_Gaussian_KRR = Predict_kernel_Ridge_MM( Gaussian_KRR_model_train,
Matrix_covariates_target=Predictors_target )

plot(f_hat_target_Gaussian_KRR, True_value_target)

## End(Not run)

```

---

Tune\_kernel\_Ridge\_MM *Tune kernel ridge regression in the mixed model framework*

---

### Description

Tune\_kernel\_Ridge\_MM tunes the rate of decay parameter of kernels, by K-folds cross-validation, for kernel ridge regression

### Usage

```

Tune_kernel_Ridge_MM( Y_train, X_train=as.vector(rep(1,length(Y_train))),
Z_train=diag(1,length(Y_train)), Matrix_covariates_train,
method="RKHS", kernel="Gaussian", rate_decay_kernel=0.1,
degree_poly=2, scale_poly=1, offset_poly=1,

```

```

degree_anova=3, init_sigma2K=2, init_sigma2E=3,
convergence_precision=1e-8, nb_iter=1000, display="FALSE",
rate_decay_grid=seq(0.1,1.0,length.out=10),
nb_folds=5, loss="mse")

```

### Arguments

rate_decay_grid	Grid over which the rate of decay is tuned by K-folds cross-validation
nb_folds	Number of folds, i.e. $K=nb\_folds$ (default is 5)
loss	mse (mean square error) or cor (correlation) (default is mse)
Y_train	numeric vector; response vector for training data
X_train	numeric matrix; design matrix of predictors with fixed effects for training data (default is a vector of ones)
Z_train	numeric matrix; design matrix of predictors with random effects for training data (default is identity matrix)
Matrix_covariates_train	numeric matrix of entries used to build the kernel matrix
method	character string; RKHS, GBLUP or RR-BLUP
kernel	character string; Gaussian, Laplacian or ANOVA (kernels for RKHS regression ONLY, the linear kernel is automatically built for GBLUP and RR-BLUP and hence no kernel is supplied for these methods)
rate_decay_kernel	numeric scalar; hyperparameter of the Gaussian, Laplacian or ANOVA kernel (default is 0.1)
degree_poly, scale_poly, offset_poly	numeric scalars; parameters for polynomial kernel (defaults are 2, 1 and 1 respectively)
degree_anova	numeric scalar; parameter for ANOVA kernel (defaults is 3)
init_sigma2K, init_sigma2E	numeric scalars; initial guess values, associated to the mixed model variance parameters, for the EM-REML algorithm (defaults are 2 and 3 respectively)
convergence_precision, nb_iter	numeric scalars; convergence precision (i.e. tolerance) associated to the mixed model variance parameters, for the EM-REML algorithm, and number of maximum iterations allowed if convergence is not reached (defaults are 1e-8 and 1000 respectively)
display	boolean (TRUE or FALSE character string); should estimated components be displayed at each iteration



```
Tuned_Gaussian_KRR_train = Tune_kernel_Ridge_MM( Y_train=Response_train, Matrix_covariates_train
=Predictors_train, method='RKHS', rate_decay_grid=seq(1,10,length.out=10), nb_folds=5, loss='mse' )

Tuned_Gaussian_KRR_model_train = Tuned_Gaussian_KRR_train$tuned_model
Tuned_Gaussian_KRR_train$optimal_h
Tuned_Gaussian_KRR_train$rate_decay_grid
Tuned_Gaussian_KRR_train$expected_loss_grid

dev.new()
plot(Tuned_Gaussian_KRR_train$rate_decay_grid, Tuned_Gaussian_KRR_train$expected_loss_grid,
type="l", main="Tuning the rate of decay (for Gaussian kernel) with K-folds cross-validation")

### Predict with tuned model

f_hat_target_tuned_Gaussian_KRR = Predict_kernel_Ridge_MM( Tuned_Gaussian_KRR_model_train,
Matrix_covariates_target=Predictors_target )

mean((f_hat_target_tuned_Gaussian_KRR-True_value_target)^2)
cor(f_hat_target_tuned_Gaussian_KRR,True_value_target)

## End(Not run)
```

# Index

\* **package**

    KRMM-package, [2](#)

EM\_REML\_MM, [4](#)

Kernel\_Ridge\_MM, [5](#)

    KRMM-package, [2](#)

Predict\_kernel\_Ridge\_MM, [8](#)

Tune\_kernel\_Ridge\_MM, [10](#)