

# Package ‘fire’

July 9, 2025

**Type** Package

**Title** Functional I-prior Regression using RKHS norm

**Version** 0.1.0

**Description** Provide methods to perform functional I-prior regression model with RKHS norm.  
The estimation of hyperparameters is done via EM algorithm.

**License** GPL (>= 3.0)

**Language** en-US

**Depends** R (>= 3.5.0)

**Encoding** UTF-8

**LazyData** true

**Imports** MASS,  
parallel,  
cli,  
knitr,  
crayon,  
gridExtra,  
ggplot2

**Suggests** spelling,  
testthat

**RoxygenNote** 7.3.2

**URL** <https://github.com/ZiqingHo/fire>

**BugReports** <https://github.com/ZiqingHo/fire/issues>

## Contents

fire . . . . .	2
fitted.fire_matrix . . . . .	5
Housing . . . . .	6
iprior_get_gram . . . . .	7
kernels_fire . . . . .	8
Manure . . . . .	9
plot.fire_fitted . . . . .	10
predict.fire_matrix . . . . .	11
print.fire . . . . .	12
sim_dat . . . . .	13

Stock . . . . .	14
summary.fire_matrix . . . . .	15
utils_tensor . . . . .	16

<b>Index</b>	<b>18</b>
--------------	-----------

---

fire	<i>Fit a FIRE model</i>
------	-------------------------

---

## Description

A function to perform functional regression using I-priors and Reproducing Kernel Hilbert Space (RKHS). The FIRE model parameters are estimated by using EM algorithm.

## Usage

```
fire(X, Y, ...)

## Default S3 method:
fire(
  X,
  Y,
  ...,
  dat_T,
  scale = TRUE,
  kernels,
  kernels_params,
  kernel_iprior = "cfbm",
  iprior_param = NULL,
  maxiter = 200,
  stop.eps = 1e-05,
  center = FALSE,
  par_init = NULL,
  os_type = "Apple",
  asymptote = TRUE,
  sample_id = 1
)

## S3 method for class 'data.frame'
fire(X, Y, ...)

## S3 method for class 'array'
fire(X, Y, ...)

## S3 method for class 'list'
fire(X, Y, ...)

## S3 method for class 'matrix'
fire(
  X,
  Y,
  ...,
```

```

    dat_T,
    scale = TRUE,
    kernels = list(cfbm),
    kernels_params = list(0.5),
    kernel_iprior = "cfbm",
    iprior_param = NULL,
    maxiter = 200,
    stop.eps = 1e-05,
    center = FALSE,
    par_init = NULL,
    os_type = "Apple",
    asymptote = TRUE
)

## S3 method for class 'tensor'
fire(
  X,
  Y,
  ...,
  dat_T,
  scale = TRUE,
  kernels,
  kernels_params,
  kernel_iprior = "cfbm",
  iprior_param = NULL,
  maxiter = 200,
  stop.eps = 1e-05,
  center = FALSE,
  par_init = NULL,
  constant = TRUE,
  os_type = "Apple",
  asymptote = TRUE,
  sample_id = 1
)

```

## Arguments

X	A numeric inputs in matrix, data.frame, array or list form
Y	A numeric response vector
...	Additional arguments passed to methods
dat_T	List of index sets for each mode
scale	Logical indicating whether to center the response by subtracting mean(Y)
kernels	List of kernel functions for each mode, may refer <a href="#">kernels_fire</a> for details
kernels_params	List of parameters for each kernel
kernel_iprior	Type of I-prior kernel
iprior_param	Parameter for I-prior kernel: <ul style="list-style-type: none"> <li>• "cfbm" - Hurst</li> <li>• "rbf" - lengthscale</li> <li>• "linear" - offset</li> <li>• "poly" - degree and offset</li> </ul>

maxiter	Maximum number of EM iterations
stop.eps	Convergence tolerance
center	Logical indicating whether to center the kernel matrix
par_init	Optional list of initial parameter values (lambda, noise)
os_type	Operating system type for compatibility ("Apple" or "Windows")
asymptote	Logical to use asymptotic initial values
sample_id	The sample mode identifier, either the 1st or last mode
constant	Logical indicating whether to include the constant kernel term

### Details

The fire function is able to take matrix, data.frame, array and list inputs, the syntax is as per the default S3 method.

### Value

An object of class `fire_matrix` or `fire_tensor`. The `print()` and `summary()` methods show the corresponding model information.

### Methods

This generic function has methods for different input types:

- `fire.matrix` for matrix/data.frame inputs
- `fire.tensor` for array/list inputs

### See Also

[kernels\\_fire](#), [Manure](#), [Housing](#)

### Examples

```
# Matrix input
data(Manure)
mod1 <- fire(X = Manure$absorp[1:5,], Y = Manure$y$DM[1:5],
  dat_T = list(1:700), stop.eps = 2, maxiter = 4)
summary(mod1)

# Array input
data(Housing)
dat_T <- list(T1 = 1:4, T2 = 1:9)
mod2 <- fire(X = Housing$X[1:5,,], Y = Housing$y[1:5,2],
  kernels = list(kronecker_delta, kronecker_delta),
  kernels_params = list(NA, NA),
  dat_T = dat_T, stop.eps = 2, maxiter = 4)
summary(mod2)
```

---

fitted.fire_matrix	<i>Obtain fitted values of FIRE model</i>
--------------------	---

---

## Description

Obtain fitted values of FIRE model

## Usage

```
## S3 method for class 'fire_matrix'
fitted(object, ...)

## S3 method for class 'fire_tensor'
fitted(object, ...)

## S3 method for class 'fire_fitted'
print(x, ...)
```

## Arguments

object	A fire_matrix or fire_tensor object
...	Not used
x	A fire_fitted object to print

## Value

A list of class fire\_fitted containing the fitted values, training RMSE, residuals and intercept.

The returned object has a print method that displays:

- Training RMSE
- Intercept value
- First few fitted values

## See Also

[fire](#)

## Examples

```
data(Manure)
mod <- fire(X = Manure$absorp[1:5,], Y = Manure$y$DM[1:5],
  dat_T = list(1:700),, stop.eps = 2, maxiter = 4)
fitted(mod)
```

---

Housing

*England Regional Housing Prices*

---

### Description

A comprehensive dataset of monthly housing prices across England regions.

### Usage

Housing

### Format

A list containing:

- x A 3D array of actual monthly prices in £ (132 months × 4 house types × 9 regions)
- y London average prices in £

### Details

x:

- *Time Period*: January 2000 - December 2010 (132 months)
- *House Types*:
  - Detached
  - Semi-Detached
  - Terraced
  - Flats
- *Regions*:
  - London
  - East of England
  - South East
  - West Midlands
  - South West
  - East Midlands
  - North West
  - Yorkshire & Humber
  - North East

y:

- From February 2000 to January 2011
- average prices across all property types in London

### Source

HM Land Registry Open Data (2024). UK House Price Index. <https://landregistry.data.gov.uk/#ukhpi>. Downloaded on 2024-12-24 and processed by package author.

## Examples

```
data(Housing)

# Get all detached home prices in London
london_detached <- Housing$X[, "Detached", "London"]
```

---

iprior\_get\_gram

*Compute Gram Matrices for I-prior Models*


---

## Description

Functions to compute various Gram matrices for I-prior models using RKHS norms.

## Usage

```
rbf_rkhs_kron(nmat, lengthscale = 1)

rbf_rkhs_kron_cross(nmat_cross, lengthscale = 1)

cfbm_rkhs_kron(nmat, Hurst = 0.5)

cfbm_rkhs_kron_cross(nmat, nmat_cross, Hurst = 0.5)
```

## Arguments

nmat	RKHS norm matrix from <a href="#">Kronecker_norm_mat</a>
lengthscale	Lengthscale parameter for RBF kernel
nmat_cross	RKHS norm matrix from <a href="#">Kronecker_norm_cross</a>
Hurst	Hurst coefficient for cfbm kernel

## Value

A Gram matrix of appropriate dimensions:

- For `rbf_rkhs_kron` and `cfbm_rkhs_kron`: Square training Gram matrix
- For `rbf_rkhs_kron_cross` and `cfbm_rkhs_kron_cross`: Cross Gram matrix between test and training data

## Examples

```
# Using internal functions to compute pairwise RKHS norm
G <- list(matrix(c(2,1,1,2),2), matrix(c(3,1,1,3),2))
X <- list(matrix(rnorm(4),2), matrix(rnorm(4),2))
nmat <- fire:::Kronecker_norm_mat(X, G, alpha=c(0.5,0.5))
ncross <- fire:::Kronecker_norm_cross(X, X[1], G, alpha=c(0.5,0.5))

# Get Gram matrices
K_rbf <- rbf_rkhs_kron(nmat)
K_cfbm <- cfbm_rkhs_kron(nmat)
K_rbf_cross <- rbf_rkhs_kron_cross(ncross)
K_cfbm_cross <- cfbm_rkhs_kron_cross(nmat, ncross)
```

---

kernels_fire	<i>Kernel Functions for FIRE</i>
--------------	----------------------------------

---

**Description**

A collection of kernel functions including:

- fbm: Fractional Brownian motion kernel
- cfbm: Centered fractional Brownian motion kernel
- cfbm\_sd: Standardized cfbm kernel
- kronecker\_delta: Kronecker delta (identity) kernel
- rbf: Radial basis function (squared exponential) kernel
- polynomial: Polynomial kernel
- mercer: Mercer kernel with cosine basis

**Usage**

```
fbm(X, Hurst = 0.5)
```

```
cfbm(X, Hurst = 0.5)
```

```
cfbm_sd(X, Hurst = 0.5)
```

```
kronecker_delta(X, center = F)
```

```
rbf(X, lengthscale = 1, center = F)
```

```
polynomial(X, d = 1, offset = 0, center = F)
```

```
mercier(X, delta = 1, max_terms = 1000, center = F)
```

**Arguments**

X	Input data (vector or matrix)
Hurst	Hurst parameter for fbm/cfbm ( $0 < \text{Hurst} \leq 1$ )
center	Logical indicating whether to center the kernel matrix
lengthscale	Bandwidth parameter for RBF kernel
d	Degree of polynomial
offset	Constant offset in polynomial kernel
delta	Smoothness parameter for Mercer kernel
max_terms	Maximum number of terms in Mercer series expansion

**Value**

A symmetric positive definite Gram matrix of size  $n \times n$  where  $n$  is the number of observations in  $X$ . The matrix has attributes including:

- "kernel" - the type of kernel used
- "parameters" - the kernel parameters used



### Examples

```
set.seed(1)
X <- matrix(rnorm(4), ncol=2)

# Different kernels
fbm(X)
cfbm(X)
cfbm_sd(X)
kronecker_delta(X)
rbf(X)
polynomial(X, d = 2, offset = 1)
mercer(X)
```

---

Manure	<i>French Manure Data</i>
--------	---------------------------

---

### Description

Combined near-infrared spectra and chemical composition measurements of cattle and poultry manure samples.

### Usage

Manure

### Format

A list containing two components:

**absorp** A matrix of NIR absorbance spectra (332 samples  $\times$  700 wavelengths). Wavelength range: 1100-2500 nm (2 nm resolution)

**y** A matrix of chemical measurements (332 samples  $\times$  3 variables)

- DM: Dry matter content (% of wet weight)
- NH4: Total ammonium nitrogen (% of wet weight)
- N: Total nitrogen (% of wet weight)

### References

Gogé, F., Thuriès, L., Fouad, Y., Damay, N., Davrieux, F., Moussard, G., ... & Morvan, T. (2021). Dataset of chemical and near-infrared spectroscopy measurements of fresh and dried poultry and cattle manure. *Data in Brief*, 34, 106647. doi:10.1016/j.dib.2020.106647

### Examples

```
data(Manure)
```

---

plot.fire_fitted	<i>Plot method for FIRE models</i>
------------------	------------------------------------

---

## Description

The type of plot produced depends on the class of the model object `fire_fitted` or `fire_prediction`

## Usage

```
## S3 method for class 'fire_fitted'
plot(x, ...)

## S3 method for class 'fire_prediction'
plot(x, ...)
```

## Arguments

<code>x</code>	A <code>fire_fitted</code> or <code>fire_prediction</code> object
<code>...</code>	Not used

## Value

Returns a ggplot object or a gridExtra-arranged plot object. The exact return depends on the input:

- For `fire_fitted` object: Returns a single ggplot object showing residuals vs fitted values
- For `fire_prediction` object:
  - When test data is available: Returns a gridExtra-arranged plot containing both residuals vs predicted and actual vs predicted plots
  - When no test data is available: Returns a single ggplot object showing the distribution of predicted values with density overlay

## Examples

```
data(Manure)
idx <- 1:5
mod <- fire(X = Manure$absorp[idx,], Y = Manure$y$DM[idx],
  dat_T = list(1:700), stop.eps = 2, maxiter = 4)

Yfitted = fitted(mod)
plot(Yfitted)

Ypred = predict(mod, newdata = Manure$absorp[idx+10,],
  Ynew = Manure$y$DM[idx+10])
plot(Ypred)
```

---

predict.fire\_matrix      *Prediction methods for FIRE models*

---

## Description

Obtain predicted values from `fire_matrix` or `fire_tensor` object, optionally with test set evaluation.

## Usage

```
## S3 method for class 'fire_matrix'
predict(object, newdata, Ynew = NULL, ...)

## S3 method for class 'fire_tensor'
predict(object, newdata, Ynew = NULL, ...)

## S3 method for class 'fire_prediction'
print(x, ...)
```

## Arguments

<code>object</code>	A <code>fire_matrix</code> or <code>fire_tensor</code> object
<code>newdata</code>	New data for prediction
<code>Ynew</code>	Optional numeric vector of true response values for the newdata. Must be the same length as the number of rows in newdata.
<code>...</code>	Not used
<code>x</code>	A <code>fire_prediction</code> object to print

## Value

A list of class `fire_prediction` containing the predicted values, and including test RMSE, residuals if `Ynew` is provided.

The returned object has a `print` method that displays:

- Number of predictions
- Test RMSE (if available)
- First few predicted values

## See Also

[fire](#)

## Examples

```
data(Manure)
idx <- 1:5
mod <- fire(X = Manure$absorp[idx,], Y = Manure$y$DM[idx],
  dat_T = list(1:700), stop.eps = 2, maxiter = 4)
predict(mod, newdata = Manure$absorp[idx+10,],
  Ynew = Manure$y$DM[idx+10])
```

---

print.fire	<i>Print method for FIRE models</i>
------------	-------------------------------------

---

### Description

Compact display of fire\_matrix or fire\_tensor objects.

### Usage

```
## S3 method for class 'fire_matrix'
print(x, ...)
```

```
## S3 method for class 'fire_tensor'
print(x, ...)
```

### Arguments

x	A fire_matrix or fire_tensor object
...	Not used

### Value

The input object (invisibly) for piping. Prints to console:

- Basic model information
- Convergence status
- Dimensions
- Estimated hyperparameters
- Marginal log-likelihood

### See Also

[fire](#)

### Examples

```
data(Manure)
mod <- fire(X = Manure$absorp[1:10,], Y = Manure$y$DM[1:10],
  dat_T = list(1:700), stop.eps = 2, maxiter = 4)
print(mod)
```

sim\_dat

*Simulate Multi-dimensional Functional Data with I-prior***Description**

Generates simulated data for multi-dimensional functional regression models using I-prior methodology.

**Usage**

```
sim_dat(
  kernels,
  kernels_param,
  alpha,
  dat_T,
  tau = 1,
  intercept_y = 0,
  intercept_x = 0,
  kernel_iprior = "cfbm",
  iprior_param = NULL,
  sigma_v = 1,
  sigma = 1,
  sigma_w = NULL,
  constant = TRUE,
  center = FALSE,
  N,
  Ntrain,
  os_type = "Apple",
  cores = NULL
)
```

**Arguments**

kernels	List of kernel functions for each mode, may refer to <a href="#">kernels_fire</a>
kernels_param	List of parameters for each kernel function
alpha	Vector of scale parameters for kernels
dat_T	List of index sets for each mode
tau	Scale parameter for the I-prior kernel
intercept_y	Intercept term for response
intercept_x	Intercept term for covariates
kernel_iprior	Type of I-prior kernel ('cfbm', 'rbf', 'linear' or 'poly')
iprior_param	Parameter for I-prior kernel (Hurst for cfbm, lengthscale for rbf)
sigma_v	Standard deviation for random effects
sigma	Noise standard deviation
sigma_w	Standard deviation for weights
constant	Logical indicating whether to include constant kernel term
center	Whether to center the kernel matrices

N	Total sample size
Ntrain	Training sample size
os_type	Operating system type ('Apple' or 'Windows')
cores	Number of cores used in parallel computation

Value

- A list containing:
- X: Simulated covariate data
  - y: Simulated response vector

See Also

[kernels\\_fire](#)

Examples

```
# 2D example
set.seed(1)
dat_T <- list(1:3, 1:2)
sim_dat(kernels = list(cfbm, rbf), kernels_param = list(0.5, 1),
alpha = c(0.5, 0.5), dat_T = dat_T, N = 10, Ntrain = 5, os_type = 'Apple', cores = 1)

# 3D example
set.seed(1)
dat_T <- list(1:3, 1:2, 1:4)
sim_dat(kernels = list(cfbm, cfbm, cfbm), kernels_param = list(0.5, 0.5,0.5),
alpha = c(0.5, 0.5, 0.5), dat_T = dat_T, N = 10, Ntrain = 5,
kernel_iprior = 'rbf', os_type = 'Apple', cores = 1)
```

---

Stock	<i>S&amp;P 500 Component Stocks Dataset (Q1 2023)</i>
-------	---

---

Description

A comprehensive dataset containing daily market data for S&P 500 component stocks and corresponding index returns for Q1 2023.

Usage

Stock

Format

- A list with two components:
- features** A 4-dimensional array [63 days × 8 features × 5 lags × 371 stocks]
  - y** A vector [63 days] of S&P 500 log returns at the next time point

## Details

### Component features:

- Date: Daily from 2023-01-03 to 2023-04-03 (63 trading days)
- Features: 8 financial metrics:
  - lr\_AdjClose: Log returns of adjusted closing price
  - lr\_Close: Log returns of closing price
  - lr\_High: Log returns of daily high price
  - lr\_Low: Log returns of daily low price
  - lr\_Open: Log returns of opening price
  - log\_Volume: Logarithm of trading volume
  - PB: Price-to-book ratio
  - TE: TEV/EBITDA
- Lags: 5 time points (T0 = current day, T1-T4 = previous days)
- Tickers: 371 S&P 500 component stocks (e.g., "AAPL", "MSFT", "AMZN")

### Component y:

- Log-returns of S&P 500 index at next trading day, calculated from adjusted closing prices

## Source

- Yahoo Finance for price/volume data and log returns
- Capital IQ for fundamental ratios

## Examples

```
data(Stock)
```

---

summary.fire_matrix	<i>Summary method for FRe models</i>
---------------------	--------------------------------------

---

## Description

Provides a comprehensive summary of fire\_matrix or fire\_tensor object, including estimated parameters, convergence information, model fit statistics, and kernel specifications.

## Usage

```
## S3 method for class 'fire_matrix'
summary(object, ...)

## S3 method for class 'summary.fire_matrix'
print(x, ...)

## S3 method for class 'fire_tensor'
summary(object, ...)

## S3 method for class 'summary.fire_tensor'
print(x, ...)
```

**Arguments**

object	A fire_matrix or fire_tensor object
...	Not used
x	A summary.fire_matrix or summary.fire_tensor object

**Value**

For summary.fire\_matrix and summary.fire\_tensor: Returns an object of class summary.fire\_matrix or summary.fire\_tensor containing:

- Estimated parameters
- Convergence information
- Model fit statistics
- Kernel specifications
- Computation timing

The print methods for these objects display nicely formatted output to the console.

**Examples**

```
data(Manure)
mod <- fire(X = Manure$absorp[1:10,], Y = Manure$y$DM[1:10],
  dat_T = list(1:700), stop.eps = 2, maxiter = 4)
summary(mod)
```

---

utils\_tensor

*Tensor Utility Functions*


---

**Description**

Collection of helper functions for tensor operations

**Usage**

```
tensor_sample(X, sample_id = 1)

vectorize_tensor(X, Index)

reshape_tensor(Mat, dim, N, Index)

unfolding(X, n, mode = TRUE)

dat_unfolding(X, sample_id = 1, mode = TRUE)
```



**Arguments**

X	Input tensor
sample_id	Sampling mode (1st or last mode)
Index	Matrix of indices
Mat	A matrix with N rows, each containing values to be assigned to the corresponding positions in the output tensor
dim	Dimensions of output tensor
N	Sample size
n	Unfolding mode
mode	Logical to print status message

**Value**

For tensor\_sample: List of tensor samples

For 'vectorize\_tensor': Vectorized tensor elements

For 'reshape\_tensor': A tensor with specified dimensions and the 1st mode corresponds to sampling mode

For 'unfolding': Unfolded matrix

For 'dat\_unfolding': List of mode-n unfolded matrices

**Examples**

```
# tensor_sample()
X <- array(1:24, dim = c(3,2,4))
tensor_sample(X, sample_id = 1)
# vectorize_tensor()
X <- array(1:8, dim = c(2,2,2))
idx <- expand.grid(T1 = 1:2, T2 = 1:2, T3 = 1:2)
vectorize_tensor(X, idx)
idx <- cbind(rep(1:2, each = 4), rep(c(1,1,2,2), 2), rep(1:2, 4))
vectorize_tensor(X, idx)
# reshape_tensor()
N <- 2
Mat <- matrix(1:12, nrow = N)
dim <- c(2,3)
idx <- expand.grid(1:2, 1:3)
reshape_tensor(Mat, dim, N, idx)
# unfolding()
X <- array(1:8, dim = c(2,2,2))
unfolding(X, 1)
# dat_unfolding()
X <- array(1:24, dim = c(3,2,4))
dat_unfolding(X, sample_id = 1)
```

# Index

## \* datasets

Housing, [6](#)

Manure, [9](#)

Stock, [14](#)

cfbm(kernels\_fire), [8](#)

cfbm\_rkhs\_kron(iprior\_get\_gram), [7](#)

cfbm\_rkhs\_kron\_cross(iprior\_get\_gram),  
[7](#)

cfbm\_sd(kernels\_fire), [8](#)

dat\_unfolding(utils\_tensor), [16](#)

fbm(kernels\_fire), [8](#)

fire, [2](#), [5](#), [11](#), [12](#)

fire.matrix, [4](#)

fire.tensor, [4](#)

fitted.fire\_matrix, [5](#)

fitted.fire\_tensor  
(fitted.fire\_matrix), [5](#)

Housing, [4](#), [6](#)

iprior\_get\_gram, [7](#)

kernels\_fire, [3](#), [4](#), [8](#), [13](#), [14](#)

kronecker\_delta(kernels\_fire), [8](#)

Kronecker\_norm\_cross, [7](#)

Kronecker\_norm\_mat, [7](#)

Manure, [4](#), [9](#)

mercier(kernels\_fire), [8](#)

plot.fire\_fitted, [10](#)

plot.fire\_prediction  
(plot.fire\_fitted), [10](#)

polynomial(kernels\_fire), [8](#)

predict.fire\_matrix, [11](#)

predict.fire\_tensor  
(predict.fire\_matrix), [11](#)

print.fire, [12](#)

print.fire\_fitted(fitted.fire\_matrix),  
[5](#)

print.fire\_matrix(print.fire), [12](#)

print.fire\_prediction

(predict.fire\_matrix), [11](#)

print.fire\_tensor(print.fire), [12](#)

print.summary.fire\_matrix  
(summary.fire\_matrix), [15](#)

print.summary.fire\_tensor  
(summary.fire\_matrix), [15](#)

rbf(kernels\_fire), [8](#)

rbf\_rkhs\_kron(iprior\_get\_gram), [7](#)

rbf\_rkhs\_kron\_cross(iprior\_get\_gram), [7](#)

reshape\_tensor(utils\_tensor), [16](#)

sim\_dat, [13](#)

Stock, [14](#)

summary.fire\_matrix, [15](#)

summary.fire\_tensor  
(summary.fire\_matrix), [15](#)

tensor\_sample(utils\_tensor), [16](#)

unfolding(utils\_tensor), [16](#)

utils\_tensor, [16](#)

vectorize\_tensor(utils\_tensor), [16](#)