Package: fdesigns (via r-universe)

October 29, 2024

Type Package

Title Optimal Experimental Designs for Functional/Dynamic Models

Version 1.0

Date 2024-08-10

Maintainer Damianos Michaelides <dm3g15@soton.ac.uk>

Description Optimal experimental designs for functional linear and functional generalised linear models, for scalar responses and profile/dynamic factors. The designs are optimised using the coordinate exchange algorithm. The methods are discussed by Michaelides (2023) <https://eprints.soton.ac.uk/474982/1/Thesis_DamianosMichaelides_Final_pdfa_1_.pdf>.

License GPL-2

Imports Rcpp, Matrix, parallel, mvQuad, mvtnorm, stats, graphics

LinkingTo Rcpp, RcppArmadillo

NeedsCompilation yes

Author Damianos Michaelides [aut, cre], Antony Overstall [aut], Dave Woods [aut]

Repository CRAN

Date/Publication 2024-08-29 19:10:06 UTC

Contents

`fdesigns-package` *Optimal Experimental Designs for Functional/Dynamic Models*

Description

Optimal experimental designs for functional linear and functional generalised linear models, for scalar responses and profile/dynamic factors. The designs are optimised using the coordinate exchange algorithm. The methods are discussed by Michaelides (2023) <https://eprints.soton.ac.uk/474982/1/Thesis_Damianos

Details

The DESCRIPTION file:

Package:	fdesigns
Type:	Package
Title:	Optimal Experimental Designs for Functional/Dynamic Models
Version:	1.0
Date:	2024-08-10
Authors@R:	c(person(given = "Damianos", family = "Michaelides", role = c("aut", "cre"), email = "dm3g15@soton.ac.uk")
Maintainer:	Damianos Michaelides <dm3g15@soton.ac.uk>
Description:	Optimal experimental designs for functional linear and functional generalised linear models, for scalar respons
License:	GPL-2
Imports:	Rcpp, Matrix, parallel, mvQuad, mvtnorm, stats, graphics
LinkingTo:	Rcpp, RcppArmadillo
Author:	Damianos Michaelides [aut, cre], Antony Overstall [aut], Dave Woods [aut]

The most important functions are pflm and pfglm which can be used to identify optimal designs for functional linear and generalised linear models, respectively, using the coordinate exchange algorithm.

Author(s)

Damianos Michaelides [aut, cre], Antony Overstall [aut], Dave Woods [aut]

Maintainer: Damianos Michaelides <dm3g15@soton.ac.uk>

References

Michaelides, D. (2023). Design of experiments for models involving profile factors (Doctoral dissertation, University of Southampton).

Examples

```
## Example 1:
## This example involves finding an A-optimal design for a functional linear model of 4 runs
## depending on one profile factor. The settings of the profile factor are represented by a
## B-spline basis of degree zero and a single knot at (0.5). The single functional parameter
```

```
## is represented with a linear power series basis. Five random starts are chosen.

example1 <- pflm(formula = ~ x1, nsd = 5, mc.cores = 1, npf = 1,
  tbounds = c(0, 1), nruns = 4, startd = NULL,  dx = c(0),
  knotsx = list(c(0.5)), pars = c("power"), db = c(1),
  knotsb = list(c()), criterion = "A", lambda = 0,
  dlbound = -1, dubound = 1, tol = 0.0001, progress = FALSE)

print(example1) ##  prints the output of example1.
##
## The number of profile factors is: 1
##
## The number of runs is: 4
##
## The objective criterion is: A-optimality
##
## The objective value is: 8.75
##
## The number of iterations is: 6
##
## The computing elapsed time is: 00:00:00


## Example 2:
## This example involves finding an A-optimal design for a functional logistic
## model of 12 runs depending on one profile factor. The settings of the profile
## factor are represented by a B-spline basis of degree zero and a three interior knots
## at (0.25, 0.50, 0.75). The single functional parameter is represented with a linear
## power series basis. The method of approximation is Monte Carlo with the prior
## specified by the function prmc. Three random starts are chosen.

set.seed(100) ## Set seed to achieve reproducibility.

prmc <- function(B,Q) {
  matrix(rnorm(B*Q, mean=0, sd=sqrt(2)), nrow=B, ncol=Q)
}
## A function which specifies the prior. This function returns a
## B by Q matrix of randomly generated values from the prior
## distribution for the model parameters.


example2 <- pfglm(formula = ~ 1 + x1, nsd = 3, mc.cores = 1, npf = 1,
             tbounds = c(0,1), nruns = 12, startd = NULL,
             dx = c(0), knotsx = list(c(0.25,0.50,0.75)),
             pars = c("power"), db = c(1), knotsb = list(c()),
             lambda = 0, criterion = "A", family = binomial,
             method=c("MC"), level = 6, B = 10000, prior = prmc,
             dlbound = -1, dubound = 1, tol = 0.0001, progress = TRUE)

print(example2) ##  prints the output of example1.
##
## The number of profile factors is: 1
##
```

```
## The number of runs is: 12
##
## The objective criterion is: A-optimality
##
## The objective value is: 20.23283
##
## The number of iterations is: 5
##
## The method of approximation is: MC
##
## The family distribution and the link function are: binomial and logit
##
## The computing elapsed time is: 00:00:12
```

`fglmobjects` *Print and Summary of* `fglm` *Objects*

Description

Print and Summary of objects of class `"fglm"`.

Usage

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

Arguments

`x`	An object of class `"fglm"`.
`object`	An object of class `"fglm"`.
`...`	Additional arguments to be passed to other methods.

Value

The functions return the number of profile factors in the functional generalised linear model, the number of runs, the criterion, the objective value of the final design, the number of iterations of the coordinate exchange algorithm to get to the final design, the method of approximation of the expectation of the objective function, the family distribution and link function, and the computational elapsed time in hours:minutes:seconds.

Note

For examples see pfglm.

Author(s)

Damianos Michaelides <<dm3g15@soton.ac.uk>>, Antony Overstall, Dave Woods

flmobjects *Print and Summary of* flm *Objects*

Description

Print and Summary of objects of class "flm".

Usage

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

Arguments

x	An object of class "flm".
object	An object of class "flm".
...	Additional arguments to be passed to other methods.

Value

The functions return the number of profile factors in the functional linear model, the number of runs, the criterion, the objective value of the final design, the number of iterations of the coordinate exchange algorithm to get to the final design, and the computational elapsed time in hours:minutes:seconds.

Note

For examples see pflm.

Author(s)

Damianos Michaelides <<dm3g15@soton.ac.uk>>, Antony Overstall, Dave Woods

P *Compute Profile Factor Polynomials*

Description

A function to be used in the formula argument in pflm and pfglm. It computes polynomials of the basis functions of profile factors to be passed in functional models and find optimal designs.

Usage

```
P(x, deg)
```

Arguments

x	A coefficient matrix from the basis expansion of a profile factor. The name passed needs to match the name of the profile factor in startd in pflm and pfglm.
deg	The degree of the polynomial effect for the profile factor.

Details

The function P is intented to be used in the formula argument in the function pflm and pfglm.

In the traditional linear models, polynomial effects of factors are handled using the function I. However, profile factors are expanded using basis functions and the coefficients are a matrix instead of a vector.

In other words, the function P is an extension to the function I, but for functional models with profile factors.

Value

The function returns an attributes list. The list contains the polynomial coefficient matrix, the argument x, and the argument deg.

Author(s)

Damianos Michaelides <<dm3g15@soton.ac.uk>>, Antony Overstall, Dave Woods

Examples

```
## Example 1:

d <- list(x1=matrix(runif(48), nrow=12))
within(d, example1 <- P(x = x1, deg = 2))


## Example 2: Use the function in a formula
## This example involves finding a D-optimal design for a functional linear model of n=20 runs
## depending on two profile factors. In addition to the main effects, the model includes the
## interaction of the profile factors and the quadratic effect of the second profile factor.
## The quadratic effect is identified in the formula argument using the \link[fdesigns]{P} function.

example2 <- pflm(formula = ~ x1 + x2 + x1:x2 + P(x2, 2), nsd = 1, mc.cores = 1,
  npf = 2, tbounds = c(0, 1), nruns = 20, startd=NULL, dx = c(2, 2),
  knotsx = list(c(0.33, 0.66), c(0.25, 0.50, 0.75)),
  pars = c("power", "power", "bspline", "bspline"), db = c(1, 1, 2, 2),
  knotsb = list(c(), c(), c(0.25, 0.50, 0.75), c(0.25, 0.50, 0.75)),
  criterion = "D", lambda = 1, tol = 0.0001, dlbound = -1, dubound = 1,
  progress = FALSE)
```

pfglm *Optimal designs for functional generalised linear models using the coordinate exchange algorithm*

Description

Optimal designs for functional generalised linear models for which the functional factors are represented as B-spline basis functions and the functional parameters are represented as power series basis functions or as B-spline basis functions.

Usage

```
pfglm(formula, nsd = 1, mc.cores = 1, npf, tbounds, nruns, startd = NULL,
       dx, knotsx, pars, db, knotsb = NULL, lambda = 0, criterion = c("A","D"),
       family, method = c("quadrature", "MC"), level = NULL, B = NULL, prior,
       dlbound = -1, dubound = 1, tol = 0.0001, progress = FALSE)
```

Arguments

formula	Object of type formula, to create the model equation. Elements need to match the list names for startd. Main effects are called using the names of the profile factors in startd, interactions are called using the names of the profile factors in startd seperated with :, and polynomial effects are called using the function P. Scalar factors are called using the same way and degree and knots through the arguments dx and knotsx are used to specify the scalar factors. A scalar factor is equivalent to a profile factor with degree 0 and no interior knots.
nsd	The number of starting designs. The default entry is 1.
mc.cores	The number of cores to use. The option is initialized from environment variable MC_CORES if set. Must be at least one, and for parallel computing at least two cores are required. The default entry is 1.
npf	The total number of (functional) factors in the model.
tbounds	A time vector of length 2, representing the boundaries of time, i.e., 0 and T.
nruns	The number of runs of the experiment.
startd	Representing the starting design but if NULL then random designs are automatically generated. It should be a list of length nsd, and each component should be a list of length npf.
dx	A vector of length npf, representing the degree of B-spline basis functions for the functions of the functional factors. A scalar factor must have a zero degree entry.
knotsx	A list of length npf, with every object in the list representing the knot vectors of each functional factor. A Scalar factor must have no interior knots, i.e., an empty knot vector.
pars	A vector of length equal to the total terms in formula, representing the basis choice for the (functional) parameters. Entries should be "power" or "bspline". A scalar parameter is represented through a "power" basis.

db — A vector of length equal to the total terms in formula, representing the degree of the basis for the (functional) parameters. For power series basis the degree is: 1 for linear, 2 for quadratic, etc. A scalar parameter must have degree 0.

knotsb — A list of length equal to the total terms in formula, representing the knot vector of each (functional) parameter. For parameters represented by a power series basis, the knot vector should be empty or NULL.

lambda — Smoothing parameter to penalise the complexity of the functions of the profile factors. The default value is 0, i.e., no penalty.

criterion — The choice of objective function. Currently there are two available choices: 1. A-optimality (criterion = "A"); 2. D-optimality (criterion = "D").

family — Specifies the error distribution and the link function of the functional generalised linear model. It can be the name of a family in the form of a character string or an R family function; see family for details. Currently, the methodology is implemented only for the binomial family with the logit link, i.e., family = binomial(link = "logit"), and the Poisson family with the log link, i.e., family = poisson(link = "log").

method — A character argument specifying the method of approximation of the expectation of the objective function with respect to a prior distribution of the parameters. Currently there are two available choices: 1. Deterministic quadrature approximation (method = "quadrature"); 2. Stochastic Monte Carlo approximation (method = "MC").

level — An optional argument that specifies the accuracy level in the quadrature approximation. It is the number of points in each dimension. If NULL and method = "quadrature", then it defaults to 5. A high value of level may increase the computation time; especially for complicated models. If the model is complicated, i.e., several profile factors or interactions and polynomials, prefer to use method = "MC".

B — An optional argument that specifies the size of the Monte Carlo samples. If NULL and method = "MC", then it defaults to 10000. For method = "quadrature", B is computer automatically according to the dimensionality of the functional model and the level argument.

prior — An argument to specify the prior distribution. For method = "MC", it should be a function of two arguments B and Q. Both arguments are integers. The value of B corresponds to the argument B, and the value of Q represents the total number of basis functions of the functional parameters. The function must generate a matrix of dimensions B by Q, that contains a random sample from the prior distribution of the parameters. For method = "quadrature", normal and uniform prior distribution for the parameters are allowed. For a normal prior distribution, the argument prior needs to be a list of length 2, with the entries named "mu" for the prior mean and "sigma2" for the prior variance-covariance matrix. The prior mean can be a scalar value that means all parameters have the same prior mean, or a vector of prior means with length equal to the number of parameters in the functional model. The prior variance-covariance can be a scalar value that means all parameters have a common variance, or a vector of prior variances with length equal to the number of parameters in the functional model, or a square matrix with the number of rows and columns equal to he

number of parameters in the functional model. For a uniform prior distribution, the argument prior needs to be a list of a single entry named "unifbound" for the lower and upper bounds of the prior distribution. The bounds can be a vector of length 2 that means all parameters have the same bounds, or a matrix with the number of rows equal to 2 and the number of columns equal to the number of parameters in the functional model.

`dlbound` The design's lower bound. The default lower bound is -1.

`dubound` The design's upper bound. The default upper bound is 1.

`tol` The tolerance value in the optimisation algorithm. Default value is 0.0001.

`progress` If TRUE, it returns the progress of iterations from the optimisation process. The default entry is FALSE.

Value

The function returns an object of class "pfglm" which is a list with the following components:

`objective.value`
The objective value of the final design found from pfglm.

`design` The final design found from pfglm. The final design is a list of length equal to the number of profile factors, exactly as the starting design startd.

`n.iterations` The total number of iterations needed to identify the final design.

`time` The computational elapsed time in finding the final design.

`startd` If starting designs were passed as an argument in pfglm, then this is the argument startd. If no starting designs were passed to pfglm, this is the starting design generated randomly by pfglm.

`tbounds` The argument tbounds.

`npf` The argument npf.

`criterion` The argument criterion.

`nruns` The argument nruns.

`formula` The argument formula.

`family` A vector of length equal to 2, containing the family and the link function.

`method` The argument method.

`B` The argument B.

`prior` The argument prior.

`dx` The argument dx.

`knotsx` The argument knotsx.

`lambda` The argument lambda.

`dbounds` A vector of length 2, containing the arguments dlbound and dubound.

`bestrep` A scalar value indicating the repetition that led to the final design.

`allobjvals` A vector of length equal to nsd, representing the objective value from all of the repetitions.

alldesigns	A list of length equal to nsd of all the final designs. Each component of the list is a list of length equal to npf representing the final design in each repetition of the coordinate exchange algorithm.
allstartd	If starting designs were passed as an argument in pfglm, then this is the argument. If no starting designs were passed to pfglm, this is the starting designs generated randomly by pfglm.

Author(s)

Damianos Michaelides <<dm3g15@soton.ac.uk>>, Antony Overstall, Dave Woods

Examples

```
## Example 1:
## This example involves finding an A-optimal design for a functional logistic
## model of 12 runs depending on one profile factor. The settings of the profile
## factor are represented by a B-spline basis of degree zero and a three interior knots
## at (0.25, 0.50, 0.75). The single functional parameter is represented with a linear
## power series basis. The method of approximation is Monte Carlo with the prior
## specified by the function prmc. Three random starts are chosen.

set.seed(100) ## Set seed to achieve reproducibility.

prmc <- function(B,Q) {
  matrix(rnorm(B*Q, mean=0, sd=sqrt(2)), nrow=B, ncol=Q)
}
## A function which specifies the prior. This function returns a
## B by Q matrix of randomly generated values from the prior
## distribution for the model parameters.

example1 <- pfglm(formula = ~ 1 + x1, nsd = 3, mc.cores = 1, npf = 1,
             tbounds = c(0,1), nruns = 12, startd = NULL,
             dx = c(0), knotsx = list(c(0.25,0.50,0.75)),
             pars = c("power"), db = c(1), knotsb = list(c()),
             lambda = 0, criterion = "A", family = binomial,
             method=c("MC"), level = 6, B = 10000, prior = prmc,
             dlbound = -1, dubound = 1, tol = 0.0001, progress = FALSE)

print(example1) ##  prints the output of example1.
##
## The number of profile factors is: 1
##
## The number of runs is: 12
##
## The objective criterion is: A-optimality
##
## The objective value is: 20.23283
##
## The number of iterations is: 5
##
## The method of approximation is: MC
##
```

```
## The family distribution and the link function are: binomial and logit
##
## The computing elapsed time is: 00:00:12
##
## plot(example1)
## then give the number of profile factor to plot


## Example 2:
## This example involves finding a A-optimal design for a functional logistic
## model of 8 runs depending on one profile factor. The settings of the profile
## factor are represented by a B-spline basis of degree zero and a single interior knot
## at 0.50. The single functional parameter is represented with a linear
## power series basis. The method of approximation is Quadrature with Normal prior
## distribution, with all parameters having mean 0 and variance 2.
## Five random starts are chosen.

example2 <- pfglm(formula = ~ 1 + x1, nsd = 5, mc.cores = 1, npf = 1,
                  tbounds = c(0,1), nruns = 8, startd = NULL,
                  dx = c(0), knotsx = list(c(0.25,0.50,0.75)),
                  pars = c("power"), db = c(1), knotsb = list(c()),
                  lambda = 0, criterion = "A", family = binomial,
                  method = c("quadrature"), level = NULL, B = NULL,
                  prior = list(mu = c(0), sigma2 = c(2)),
                  dlbound = -1, dubound = 1, tol = 0.0001,
                  progress = FALSE)

## The number of profile factors is: 1
##
## The number of runs is: 8
##
## The objective criterion is: A-optimality
##
## The objective value is: 31.32342
##
## The number of iterations is: 9
##
## The method of approximation is: quadrature
##
## The family distribution and the link function are: binomial and logit
##
## The computing elapsed time is: 00:00:00
```

---

pflm *Optimal designs for functional linear models using the coordinate exchange algorithm*

Description

Optimal designs for functional linear models for which the functional factors are represented as B-spline basis functions and the functional parameters are represented as power series basis functions

or as B-spline basis functions.

Usage

```
pflm(formula, nsd = 1, mc.cores = 1, npf, tbounds,
  nruns, startd = NULL, dx, knotsx, pars, db,
  knotsb = NULL, criterion = c("A", "D"), lambda = 0,
  dlbound = -1, dubound = 1, tol = 1e-04, progress = FALSE)
```

Arguments

formula	Object of type formula, to create the model equation. Elements need to match the list names for startd. Main effects are called using the names of the profile factors in startd, interactions are called using the names of the profile factors in startd seperated with :, and polynomial effects are called using the function P. Scalar factors are called using the same way and degree and knots through the arguments dx and knotsx are used to specify the scalar factors. A scalar factor is equivalent to a profile factor with degree 0 and no interior knots.
nsd	The number of starting designs. The default entry is 1.
mc.cores	The number of cores to use. The option is initialized from environment variable MC_CORES if set. Must be at least one, and for parallel computing at least two cores are required. The default entry is 1.
npf	The total number of (profile) factors in the model.
tbounds	A time vector of length 2, representing the boundaries of time, i.e., 0 and T.
nruns	The number of runs of the experiment.
startd	Representing the starting design but if NULL then random designs are automatically generated. It should be a list of length nsd, and each component should be a list of length npf.
dx	A vector of length npf, representing the degree of B-spline basis functions for the functions of the functional factors. A scalar factor must have a zero degree entry.
knotsx	A list of length npf, with every object in the list representing the knot vectors of each functional factor. A Scalar factor must have no interior knots, i.e., an empty knot vector.
pars	A vector of length equal to the total terms in formula, representing the basis choice for the (functional) parameters. Entries should be "power" or "bspline". A scalar parameter is represented through a "power" basis.
db	A vector of length equal to the total terms in formula, representing the degree of the basis for the (functional) parameters. For power series basis the degree is: 1 for linear, 2 for quadratic, etc. A scalar parameter must have degree 0.
knotsb	A list of length equal to the total terms in formula, representing the knot vector of each (functional) parameter. For parameters represented by a power series basis, the knot vector should be empty or NULL.
lambda	Smoothing parameter to penalise the complexity of the functions of the profile factors. The default value is 0, i.e., no penalty.

criterion	The choice of objective function. Currently there are two available choices: 1. A-optimality (criterion = "A"); 2. D-optimality (criterion = "D").
tol	The tolerance value in the optimisation algorithm. Default value is 0.0001.
dlbound	The design's lower bound. The default lower bound is -1.
dubound	The design's upper bound. The default upper bound is 1.
progress	If TRUE, it returns the progress of iterations from the optimisation process. The default entry is FALSE.

Value

The function returns an object of class "pflm" which is a list with the following components:

objval	The objective value of the final design found from pflm.
design	The final design found from pflm. The final design is a list of length equal to the number of profile factors, exactly as the starting design startd.
nits	The total number of iterations needed to identify the final design.
time	The computational elapsed time in finding the final design.
startd	If starting designs were passed as an argument in pflm, then this is the starting design from the argument startd that led to the final design. If no starting designs were passed to pflm, this is the starting design generated randomly by pflm that led to the final design.
tbounds	The argument tbounds.
npf	The argument npf.
criterion	The argument criterion.
nruns	The argument nruns.
formula	The argument formula.
dx	The argument dx.
knotsx	The argument knotsx.
lambda	The argument lambda.
dbounds	A vector of length 2, containing the arguments dlbound and dubound.
bestrep	A scalar value indicating the repetition that led to the final design.
allobjvals	A vector of length equal to nsd, representing the objective value from all of the repetitions.
alldesigns	A list of length equal to nsd of all the final designs. Each component of the list is a list of length equal to npf representing the final design in each repetition of the coordinate exchange algorithm.
allstartd	If starting designs were passed as an argument in pflm, then this is the argument. If no starting designs were passed to pflm, this is the starting designs generated randomly by pflm.

Author(s)

Damianos Michaelides <<dm3g15@soton.ac.uk>>, Antony Overstall, Dave Woods

Examples

```
## Example 1:
## This example involves finding an A-optimal design for a functional linear model of 4 runs
## depending on one profile factor. The settings of the profile factor are represented by a
## B-spline basis of degree zero and a single knot at (0.5). The single functional parameter
## is represented with a linear power series basis. Five random starts are chosen.

example1 <- pflm(formula = ~ x1, nsd = 5, mc.cores = 1, npf = 1,
  tbounds = c(0, 1), nruns = 4, startd = NULL,  dx = c(0),
  knotsx = list(c(0.5)), pars = c("power"), db = c(1),
  knotsb = list(c()), criterion = "A", lambda = 0,
  dlbound = -1, dubound = 1, tol = 0.0001, progress = FALSE)

print(example1) ##  prints the output of example1.
##
## The number of profile factors is: 1
##
## The number of runs is: 4
##
## The objective criterion is: A-optimality
##
## The objective value is: 8.75
##
## The number of iterations is: 6
##
## The computing elapsed time is: 00:00:00

## plot(example1)
## then give the number of profile factor to plot


## Example 2:
## This example involves finding a D-optimal design for a functional linear model of n=20 runs
## depending on two profile factors. In addition to the main effects, the model includes the
## interaction of the profile factors and the quadratic effect of the second profile factor.
## The settings of the profile factors are represented by B-spline basis of quadratic degrees
## and knots at (0.33, 0.66) and (0.25, 0.50, 0.75). The functional parameters are represented
## with linear power basis and quadratic B-spline basis with knots at (0.25, 0.50, 0.75).
## The complexity of the designs is penalised with the smoothing value equal to 1.

example2 <- pflm(formula = ~ x1 + x2 + x1:x2 + P(x2, 2), nsd = 1, mc.cores = 1,
  npf = 2, tbounds = c(0, 1), nruns = 20, startd = NULL, dx = c(2, 2),
  knotsx = list(c(0.33, 0.66), c(0.25, 0.50, 0.75)),
  pars = c("power", "power", "bspline", "bspline"), db = c(1, 1, 2, 2),
  knotsb = list(c(), c(), c(0.25, 0.50, 0.75), c(0.25, 0.50, 0.75)),
  criterion = "D", lambda = 1, tol = 0.0001, dlbound = -1, dubound = 1,
  progress = FALSE)

print(example2) ##  prints the output of example2.
##
## The number of profile factors is: 2
##
```

```
## The number of runs is: 20
##
## The objective criterion is: D-optimality
##
## The objective value is: 0.05706758
##
## The number of iterations is: 6
##
## The computing elapsed time is: 00:00:17
```

`plot.fglm` *Plot of* `fglm` *Objects*

Description

Plot of `"fglm"` objects. For the choice of a profile factor, the optimal functions are plotted.

Usage

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

Arguments

`x`	An object of class `"fglm"`.
`...`	Additional arguments to be passed to other methods.

Value

The function returns the question: "Which profile factor to plot?". The answer needs to be an integer to specify the profile factor for which to plot the optimal functions in every run. The value needs to be between 1 and the argument npf from the function pfglm.

After that, the function returns n.runs (see pfglm) plots of the optimal functions, of the profile factor indicated. The x-axis represents the time, the y-axis represents the values of the function of the profile factor, and the title indicated the number of run of each plot.

Note

For examples see pflm.

Author(s)

Damianos Michaelides <<dm3g15@soton.ac.uk>>, Antony Overstall, Dave Woods

plot.flm *Plot of* flm *Objects*

Description

Plot of "flm" objects. For the choice of a profile factor, the optimal functions are plotted.

Usage

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

Arguments

x	An object of class "flm".
...	Additional arguments to be passed to other methods.

Value

The function returns the question: "Which profile factor to plot?". The answer needs to be an integer to specify the profile factor for which to plot the optimal functions in every run. The value needs to be between 1 and the argument npf from the function pflm.

After that, the function returns n.runs (see pflm) plots of the optimal functions of the profile factor indicated. The x-axis represents the time, the y-axis represents the values of the function of the profile factor, and the title indicated the number of run of each plot.

Note

For examples see pflm.

Author(s)

Damianos Michaelides <<dm3g15@soton.ac.uk>>, Antony Overstall, Dave Woods

Index