R package **fdesigns**: optimal designs for functional models

Damianos Michaelides[*], Antony Overstall, Dave Woods

Using the fdesigns package

This vignette provides examples of the usage of the `R` package **fdesigns**. The package implements optimal experimental designs for functional models. The two main functions are `pflm()` and `pfglm()`. Both implement optimal experimental designs for models involving profile factors. The package also includes a support function `P()`, as well as printing and plotting functions for the resulting objects.

- `pflm()` = parallel functional linear models (coordinate exchange algorithm).
- `pfglm()` = parallel functional generalised linear models (coordinate exchange algorithm).
- `P()` = constructs polynomials for profile factors.

Examples

The examples focus on the main functions `pflm()` and `pfglm()`. The support function `P()` is used inside the `formula` argument to specify polynomials. The functions are computationally efficient for multiple profile factors. Generalised models are more computationally expensive, so examples are kept simple.

FLM with one profile factor

In this example, a functional linear model with a single profile factor is considered.

The profile factor is represented by a BS basis of degree 0 with 3 equally spaced interior knots, and time boundaries between 0 and 1. This gives 4 basis functions. The number of runs is 4.

```
tbounds <- c(0, 1)
nruns <- 4
npf <- 1
dx <- c(0)
knotsx <- list(c(0.25, 0.50, 0.75))
nx <- rep(0, npf)
for (j in 1:npf) {
  nx[j] <- dx[j] + length(knotsx[[j]]) + 1
}
```

[*]Corresponding author: `dm3g15@soton.ac.uk`

One hundred starting designs are considered. The argument `startd` is left to its default `NULL`. Thus, the starting designs are automatically generated within the function `pflm()`.

The functional parameter is assumed to be a linear power series, and the goal is to find an A-optimal design. To achieve this, the argument `criterion` is set to `"SE"`, which corresponds to the squared error loss function. The smoothing parameter `lambda` is set to zero, i.e., an improper prior that reduces the criterion to A-optimality. All other arguments are kept at their default values.

```
example1a <- pflm(formula = ~ x1, nsd = 100, mc.cores = 1,
                  npf = npf, tbounds = tbounds,
                  nruns = nruns, dx = dx, knotsx = knotsx,
                  pars = c("power"), db = c(1),
                  knotsb = list(c()), criterion = "SE",
                  lambda = 0)
```

Printing the resulting `"flm"` object using the code:

```
print(example1a)
```

provides the summary of the outcome.

```
The number of profile factors is: 1
The number of runs is: 4
The loss function is: SE
The objective value is: 8.75
The number of iterations is: 5
The computing elapsed time is: 00:00:00
```

The final design is extracted using the code,

```
example1a$design
```

and the outcome is a 4×4 design matrix.

```
     [,1] [,2] [,3] [,4]
[1,]    1    1    1    1
[2,]    1    1   -1   -1
[3,]   -1   -1    1    1
[4,]   -1   -1    1    1
```

The optimal functions of the profile factor are plotted using the code,

```
par(mfrow = c(2,2))
plot(example1a, pf = 1)
```

where 1 corresponds to the profile factor to plot.

Alternatively, suppose that the interest becomes D-optimality (Shannon Information loss function with an improper prior) and the functional parameter is represented by a quadratic power basis, with everything else remain unchanged, then the code is,

```
example1b <- pflm(formula = ~ x1, nsd = 100, mc.cores = 1,
                  npf = npf, tbounds = tbounds,
                  nruns = nruns, dx = dx, knotsx = knotsx,
                  pars = c("power"), db = c(2),
                  knotsb = list(c()), criterion = "SI",
                  lambda = 0)
```

```
print(example1b)
```

and the summary of the outcome is,

```
The number of profile factors is: 1
The number of runs is: 4
The loss function is: SI
The objective value is: 4.618802
The number of iterations is: 3
The computing elapsed time is: 00:00:00
\end{verbatim}
```

FLM with one profile factor and roughness penalty

A single profile factor is still considered as in the previous section. The addition to the previous example is that the complexity of the parameter is penalised through a smoothing parameter $\lambda = 10$. The loss function is weighted squared error and the parameter basis is a quadratic power basis. The optimal design can be found using the code:

```
example2 <- pflm(formula = ~ x1, nsd = 100, mc.cores = 1,
                 npf = 1, tbounds = c(0, 1), nruns = 4,
                 dx = dx, knotsx = knotsx,
                 pars = c("power"), db = c(2),
                 knotsb = list(c()),
                 criterion = "WSE", lambda = 10)

print(example2)
```

```
The number of profile factors is: 1
The number of runs is: 4
The loss function is: WSE
The objective value is: 1.416839
The number of iterations is: 5
The computing elapsed time is: 00:00:00
```

FLM with one profile factor and three scalar factors

In this example, the FLM with a single profile factor and three scalar factors from Section @ref(packex2) is considered.
It is assumed that control of the profile factor is represented by a B-spline basis of degree zero with three equally spaced interior knots and time boundaries again being 0 and 1.

Main effects of the scalar factors are considered. The number of runs is 12. The scalar factors are passed as profile factors to the function `pflm()`, with degree zero and no interior knots to specify that they are scalar factors. Thus, the argument `npf` is equal to 4.

```
tbounds <- c(0, 1)
nruns <- 12
npf <- 4
dx <- c(0, 0, 0, 0)
```

```
knotsx <- list(c(0.25, 0.50, 0.75), c(), c(), c())
nx <- rep(0, npf)
for (j in 1:npf) {
  nx[j] <- dx[j] + length(knotsx[[j]]) + 1
}
```

Fifty starting designs are generated and passed to the function manually, with the bounds of the factors set to the defaults -1 and 1. The factors are named x_1, x_2, x_3, x_4 and must match the factors in the formula argument.

```
indd <- list()
startd <- list()
dlbound <- -1
dubound <- 1
nsd <- 50
for (c in 1:nsd) {
  set.seed(c)
  for (i in 1:npf) {
    indd[[i]] <- matrix(runif(nruns * nx[i], dlbound, dubound),
                        nrow = nruns, ncol = nx[i])
    names(indd)[i] <- paste0("x", i, sep="")
  }
  startd[[c]] <- indd
}
```

The functional parameter is assumed to be represented by a linear power series and the scalar parameters are represented through a power series of degree zero. Moreover, the objective criterion is A-optimality, i.e., SE loss function with improper prior.

```
example3a <- pflm(formula = ~ x1 + x2 + x3 + x4, nsd = nsd,
                  mc.cores = 1, npf = npf, tbounds = tbounds,
                  nruns = nruns, startd = startd, dx = dx,
                  knotsx = knotsx,
                  pars = c("power", "power", "power", "power"),
                  db = c(1, 0, 0, 0),
                  knotsb = list(c(), c(), c(), c()),
                  criterion = "A", lambda = 0, dlbound = dlbound,
                  dubound = dubound, tol = 0.0001)
```

Printing the resulting "flm" object using the code:

```
print(example3a)
```

provides the summary of the outcome.

```
The number of profile factors is: 4
The number of runs is: 12
The loss function is: SE
The objective value is: 2.833333
The number of iterations is: 10
The computing elapsed time is: 00:00:01
```

FGLM with one profile factor and quadrature approximation

A functional logistic model depending on a single profile factor is considered. Thus, the `family` choice in R is `"binomial"`.

It is assumed that control of the profile factor is represented by a B-spline basis of degree zero and for this example, the choice of seven equally spaced interior knots is considered, with time boundaries being 0 and 1. Thus, there are eight basis functions for the profile factor. The number of experimental runs is eight.

```r
tbounds <- c(0, 1)
nruns <- 8
npf <- 1
dx <- c(0)
knotsx <- list(c(0.125, 0.250, 0.375, 0.500,
                 0.625, 0.750, 0.875))
nx <- rep(0, npf)
for (j in 1:npf) {
  nx[j] <- dx[j] + length(knotsx[[j]]) + 1
}
```

Fifty starting designs are considered, which are generated automatically within `pfglm()`.
The bounds of the profile factor are set to the defaults.

The functional parameter is assumed to be a linear power series.
The objective is to identify pseudo-Bayesian A-optimal designs.
The prior of the parameters is assumed to be normal, with mean zero and variance one.
To approximate the expectation with respect to the prior, a normal quadrature scheme with abscissas and weights is applied.
All other arguments are kept to their default values.

```r
example4 <- pfglm(formula = x1, nsd = 50, mc.cores = 1,
                  npf = npf, tbounds = tbounds,
                  nruns = nruns, dx = dx, knotsx = knotsx,
                  pars = c("power"), db = c(1),
                  knotsb = list(c()), criterion = "A",
                  family = binomial, method = c("quadrature"),
                  level = NULL, B = NULL,
                  prior = list(mu = c(0), sigma2 = c(1)),
                  dlbound = -1, dubound = 1)
```

Printing the resulting `"fglm"` object using the code:

```r
print(example4)
```

```
The number of profile factors is: 1
The number of runs is: 8
The objective criterion is: A-optimality
The objective value is: 21.64537
The number of iterations is: 5
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
```

FGLM with one profile factor depending on main effect and MC approximation

In this example, the functional Poisson model depending on a single profile factor is considered. Thus, the `family` choice in **R** is `"poisson"`. It is assumed that control of the profile factor is represented by a B-spline basis of degree one, with the choice of four equally spaced interior knots. Thus, there are six basis functions for the profile factor. The time boundaries are 0 and 1, and the number of runs is 8.

The model with main and quadratic effect of the profile factor is tackled. The parameters are assumed to be represented by a B-spline basis of degree one and a single knot at $t = 0.5$.

```
tbounds <- c(0, 1)
nruns <- 8
npf <- 1
dx <- c(1)
knotsx <- list(c(0.20, 0.40, 0.60, 0.80))
nx <- rep(0, npf)
for (j in 1:npf) {
  nx[j] <- dx[j] + length(knotsx[[j]]) + 1
}
```

The use of 8 runs and B-spline basis for the parameters increases the complexity, and hence the computational expense.

For this reason, the Monte Carlo approximation, with a normal prior (mean zero and variance two), is preferred. The `prior` argument, when `method = "MC"`, must be a function. The function used is:

```
set.seed(100)
prmc <- function(B, Q){
  matrix(rnorm(B * Q, mean = 0, sd = sqrt(2)), nrow = B, ncol = Q)
}
```

```
example5 <- pfglm(formula = ~ 1 + x1 + P(x1, 2), nsd = 1, mc.cores = 1,
                  npf = 1, tbounds = tbounds, nruns = nruns,
                  startd = NULL, dx = dx, knotsx = knotsx,
                  pars = c("bspline", "bspline"), db = c(1, 1),
                  knotsb = list(c(0.5), c(0.5)), lambda = 0,
                  criterion = "D", family = poisson, method = c("MC"),
                  level = NULL, B = 10000, prior = prmc, tol = 0.01)
```

with printed output,

```
The number of profile factors is: 1
The number of runs is: 8
The objective criterion is: D-optimality
The objective value is: 12.13859
The number of iterations is: 9
The method of approximation is: MC
The family distribution and the link function are: poisson and log
The computing elapsed time is: 00:00:16
```