Package ‘fdaMixed’

September 13, 2023

Type Package

Title Functional Data Analysis in a Mixed Model Framework

Version 0.6.1

Date 2023-09-13

Description Likelihood based analysis of 1-dimension functional data in a mixed-effects model framework. Matrix computation are approximated by semi-explicit operator equivalents with linear computational complexity. Markussen (2013) <doi:10.3150/11-BEJ389>.

License GPL (>= 3)

LazyLoad yes

Imports Formula, Rcpp

LinkingTo Rcpp, RcppArmadillo

Repository CRAN

NeedsCompilation yes

Author Bo Markussen [aut, cre]

Maintainer Bo Markussen <bomar@math.ku.dk>

Date/Publication 2023-09-13 20:30:02 UTC

R topics documented:

`fdaMixed-package` *Functional Data Analysis in a Mixed Model Framework*

Description

Likelihood based analysis of 1-dimension functional data in a mixed-effets model framework. The methodology is designed for equidistantly sampled high frequency data, where the needed matrix computation may be approximated by semi-explicit operator equivalents with linear computational complexity. Extensions exist for non-equidistantly sampled data, but these have not been implemented.

Author(s)

Bo Markussen <bomar@math.ku.dk>

References

Bo Markussen (2013), "Functional data analysis in an operator based mixed model framework", Bernoulli, vol. 19, pp. 1-17.

Conrad Sanderson (2010), "Armadillo: An open source C++ linear algebra library for fast prototyping and computationally intensive experiments", NICTA technical report.

Dirk Eddelbuettel, "Rcpp: Seamless R and C++ Integration with Rcpp", UseR!, Springer, 2013.

See Also

Implementation done using the package `RcppArmadillo`. For penalized likelihood analysis of functional data see the packages `fda` and `fda.usc`.

Examples

```
x <- seq(0,2*pi,length.out=200)
y.true <- sin(x)+x
y.obs <- y.true + rnorm(200)
est0 <- fdaLm(y.obs~0,Fright="open",right=2*pi)
est1 <- fdaLm(y.obs~0+x,Fright="open",right=2*pi)
plot(x,y.obs,main="Estimating the sum of a line and a curve")
lines(x,y.true,lty=2)
lines(x,est0$xBLUP[,1,1],col=2)
lines(x,est1$betaHat*x+est1$xBLUP[,1,1],col=3)
legend("topleft",c("True curve","Smooth","Line + smooth"),col=1:3,lty=c(2,1,1))
```

dataTrans *Scale invariant Box-Cox transformation*

Description

Performs forward and backward Box-Cox power transformation including the invariance scaling based on the geometric mean.

Usage

```
dataTrans(y, mu, direction = "backward", geoMean = NULL)
```

Arguments

y	The numeric variable object to be transformed.
mu	The power parameter, where zero corresponds to the logarithmic transformation.
direction	A character variable. If the lower case of the first letter equals "b" (default), then the backward transformation is performed. If the lower case of the first letter equals "f", then the forward transformation is performed.
geoMean	If a numeric is stated, then this is taken as the geometric mean of the untransformed observations. If NULL (default), then the geometric mean is computed from the observation y. The latter is only available for the forward transformation.

Value

The transformed variable.

Note

This function is intended to be used in conjunction with fdaLm to achieve estimates on the orginal scale. Thus, the geometric mean of the original observations should be kept in order to have the correct backtransformation.

Author(s)

Bo Markussen <bomar@math.ku.dk>

Examples

```
# -----------------------------------------------------
# Make 3 samples with the following characteristics:
#   1) length N=500
#   2) sinusoid form + linear fixed effect + noise
#   3) exponential transformed
# -----------------------------------------------------
```

```
N <- 500
sample.time <- seq(0,2*pi,length.out=N)
z <- c("a","b","c")
x0 <- c(0,10,20)
x1 <- rep(x0,each=N)
y <- c(sin(sample.time),sin(sample.time),sin(sample.time))+x1+rnorm(3*N)

# Make exponential-Box-Cox-backtransformation
# Scaling with geometric mean requires that we solve the Whiteker function
geoMean <- mean(y)
geoMean <- uniroot(function(x){x*log(x)-geoMean},c(exp(-1),(1+geoMean)^2))$root
y <- dataTrans(y,0,"b",geoMean)

# ---------------------------------------------------
# Do fda's with global and marginal fixed effects
# Also seek to find Box-Cox transformation with mu=0
# ---------------------------------------------------

est0 <- fdaLm(y|z~x0,boxcox=1)
est1 <- fdaLm(y|z~x1,boxcox=1)

# ----------------------------------------------------
# Display results
# ----------------------------------------------------

# Panel 1
plot(sample.time,dataTrans(est0$betaHat[,"(Intercept)"]+est0$betaHat[,"x0"],
                           est0$boxcoxHat,"b",geoMean)/
                 dataTrans(est0$betaHat[,"(Intercept)"],est0$boxcoxHat,"b",geoMean),
     main="Effect of x (true=1.2)",xlab="time",
     ylab="response ratio")
abline(h=dataTrans(est1$betaHat["(Intercept)"]+est1$betaHat["x1"],
                   est1$boxcoxHat,"b",geoMean)/
         dataTrans(est1$betaHat["(Intercept)"],est1$boxcoxHat,"b",geoMean),col=2)
legend("topleft",c("marginal","global"),pch=c(1,NA),lty=c(NA,1),col=1:2)

# Panel 2
plot(sample.time,dataTrans(est0$betaHat[,"(Intercept)"]+est0$betaHat[,"x0"],
                           est0$boxcoxHat,"b",geoMean)-
                 dataTrans(est0$betaHat[,"(Intercept)"],est0$boxcoxHat,"b",geoMean),
     main="Effect of x (true=1)",xlab="time",
     ylab="response difference")
abline(h=dataTrans(est1$betaHat["(Intercept)"]+est1$betaHat["x1"],
                   est1$boxcoxHat,"b",geoMean)-
         dataTrans(est1$betaHat["(Intercept)"],est1$boxcoxHat,"b",geoMean),col=2)
legend("bottomleft",c("marginal","global"),pch=c(1,NA),lty=c(NA,1),col=1:2)

# Panel 3
plot(sample.time,est0$xBLUP[,1,1],type="l",
     main="Marginal ANOVA",xlab="time",ylab="x BLUP")

# Panel 4
plot(sample.time,est1$xBLUP[,1,1],type="l",
```

```
main="Global ANOVA",xlab="time",ylab="x BLUP")
```

fdaLm *Linear mixed-effects model for functional data*

Description

Fits variance and smoothing parameters, and possibly also Box-Cox transformation, by maximum restricted likelihood. Estimate fixed parameters, predict random effects, and predict serial correlated effect at point of maximum restricted likelihood. Linear models for fixed and random effects may be global or marginal over sample times.

Usage

```
fdaLm(formula, data, design, boxcox = NULL, G = 1, lambda = 1, nlSearch
= TRUE, K.order = 1, D.order = NULL, Fleft = "tied", Fright = "tied",
left = NULL, right = NULL)
```

Arguments

- `formula` A multiple formula of the type `Y|id ~ fixed|random`. Here `Y` is the response variable, `id` is a factor separating the samples, `fixed` is a linear model for the fixed effect, and `random` is a linear model for the random effect.
- `data` An optional data frame containing the variables. See details below.
- `design` An optional data frame containing the design variables in the specification of the fixed and the random effects. See details below.
- `boxcox` The power parameter in the scale invariant Box-Cox transformation. If `NULL` (default), then no transformation is performed. If a numeric value is provided, then a scale invariant Box-Cox transformation of the response variable is performed. The numeric value is either used as it is (`nlSearch=FALSE`) or as the starting point for a non-linear optimization (`nlSearch=TRUE`.)
- `G` Variance of the random effects. Present implementation only allows for independent random effects, i.e. `G` is scalar. Used depending on `nlSearch` as described above.
- `lambda` Start value for the `lambda` parameter describing the L-operator. Presently the following forms are implemented: If `K.order` is odd, then `lambda` may have length=1 corresponding to `L=-lambda[1]*D^(2*K.order)`, or length=2 corresponding to `L=-lambda[1]*D^(2*K.order)+lambda[2]`. If `K.order` is even, then `lambda` may have length=1 corresponding to `L=-lambda[1]*D^(2*K.order)`, length=2 corresponding to `L=-lambda[1]*D^(2*K.order)+lambda[2]*D^K.order`, or length=3 corresponding to `L=-lambda[1]*D^(2*K.order)+lambda[2]*D^K.order+lambda[3]`. Used depending on `nlSearch` as described above. All coefficients must be non-negative, and the leading coefficient `lambda[1]` must be strictly positive. Coefficients equal to zero are kept fixed at zero in the non-linear optimization.

nlSearch — If TRUE (default), then a non-linear optimization of the parameters boxcox, G, lambda is performed (present implementation uses nlminb). If FALSE, then the initial values of the non-linear parameters are used.

K.order — The order of the K-operator.

D.order — The requested order of derivatives of the prediction of the serial correlated effect xBLUP. If NULL (default), then D.order is set to the maximal recommended order K.order.

Fleft — Specification of the K.order boundary conditions at the left limit of the sampling interval. Value "tied" (default) gives bridge-type conditions. Value "open" shifts up the bridge-type conditions one differential order, hence removing the restriction on the level (corresponding to the open end of a Brownian motion). Otherwise arbitrary linear boundary conditions may be specified as a matrix with dimension (K.order,2*K.order).

Fright — Similarly for the K.order boundary conditions at the right limit of the sampling interval.

left — Left limit of the sampling interval. If NULL (default), then left is set to 0.

right — Right limit of the sampling interval. If NULL (default), then right is set to the number of sampling points. Thus, the default values of left and right give sampling distance equal to 1.

Details

The response variable Y is taken from the data frame data (subsidiary the parent environment). If there is more than one sample, then the responses must be stacked sample-wise on top of each other. The sample identifier id is sought for in both data frames data and design (subsidiary the parent environment). The primarily function of the identifier is to decide the number of samples. But if id is present in both data frames, and if there is more that one sample, then this variable is also used to match the reponse vector to the design variables (i.e. these need not appear in the same order).

The design variables fixed and random for the fixed and the random effects are taken from the data frame design (subsidiary the parent environment), subsidiary from the data frame data (subsidiary the parent environment).

If the number of observations in the design variables equal the total number of response observations, then a global ANOVA is performed. If the number of observations in the design variables equal the number of sample points, then a marginal ANOVA is performed.

Value

A list with components

logLik — Minus twice the log restricted likelihood taken at the estimates.

ANOVA — Specifies whether fixed and random effects were estimated globally (global) or marginally (marginal).

nlSearch — Specifies whether non-linear optimization was performed (TRUE / FALSE).

counts — Number of computations of the negative log likelihood.

boxcoxHat	Maximum restricted likelihood estimate for the power parameter in the scale invariant Box-Cox transformation. Equal to `not done` if the Box-Cox transformation is not used.
Ghat	Maximum restricted likelihood estimate for the variance matrix of the random effects.
lambdaHat	Maximum restricted likelihood estimate for the lambda parameter describing the L-operator.
sigma2hat	Maximum restricted likelihood estimate for the noise variance.
betaHat	For global ANOVA a vector with estimate for the fixed effect. For marginal ANOVA a matrix with estimate for the fixed effects.
uBLUP	For global ANOVA a vector with prediction of the random effect. For marginal ANOVA a matrix with predictions of the random effects.
xBLUP	Array with predictions of serial correlated effects. The dimension is (sample length,sample numbers,1+`D.order`).
condRes	Matrix of conditional residuals. The dimension is (sample length,sample numbers).
betaVar	Variance matrix of fixed effect estimate.

Note

If the real value of the left most eigenvalues are non-positive, and if the real value of the right most eigenvalues are non-negative, then the underlying algorithm is numerical stable. This will always be the situation for the present restriction of the L-operator.

If `lambda` has length=1, then it may also be interpreted as the smoothing parameter in the penalized likelihood framework.

If `D.order` is chosen larger than `K.order`, this number of derivaties are also computed during the non-linear optimization. This might slow down the computation speed a little bit.

Author(s)

Bo Markussen <bomar@math.ku.dk>

See Also

See also `findRoots` and `dataTrans`.

Examples

```
# ---------------------
# Using a fixed effect
# ---------------------
x <- seq(0,2*pi,length.out=200)
y.true <- sin(x)+x
y.obs <- y.true + rnorm(200)
est0 <- fdaLm(y.obs~0,Fright="open",right=2*pi)
est1 <- fdaLm(y.obs~0+x,Fright="open",right=2*pi)
plot(x,y.obs,main="Estimating the sum of a line and a curve")
```

```
lines(x,y.true,lty=2)
lines(x,est0$xBLUP[,1,1],col=2)
lines(x,est1$betaHat*x+est1$xBLUP[,1,1],col=3)
legend("topleft",c("True curve","Smooth","Line + smooth"),col=1:3,lty=c(2,1,1))

# --------------------------
# Including a random effect
# --------------------------
# Build data frame
test.frame <- data.frame(y=rnorm(50),sample=factor(rep(1:5,each=10)),
                         x=rep(0:9,times=5),
                         f=factor(rnorm(50) < 0,labels=c("a","b")),
                         j=factor(rnorm(50) < 0,labels=c("A","B")))
test.frame$y <- test.frame$y + 2 +
    3*(test.frame$f=="a")*test.frame$x + 5*(test.frame$f=="b")*test.frame$x +
(-10)*(test.frame$j=="A") + 10*(test.frame$j=="B")
# This is the model 'y|sample ~ f:x|j' with intercept=2, slopes (3,5),
# and random effects (-10,10)
est <- fdaLm(y|sample ~ f:x|0+j,data=test.frame)
print(est)
```

`findRoots` *Complex roots of quadratic polynomial*

Description

Find complex roots of polynomials in x that are quadratic polynomials in x^k

Usage

```
findRoots(coefs, k = 1)
```

Arguments

`coefs`	Coefficients `(c_0,c_k,c_2k)` of quadratic polynomial in x^k. Also accepts matrix input (J,3).
`k`	Order of x^k

Details

It is assumed that `c_2k` is non-zero, and that at least one of `c_0` and `c_k` are non-zero (otherwise, we have a double root, which is not treated by `fdaLm` in the present implementation). An error is issued if these assumptions are violated.

Value

A list with components

`left`	The k roots with left most real components
`right`	The k roots with right most real components

Note

This function is intended for internal usage in `fdaLm` to find eigenvalues. If a robust and stable method of finding all the complex roots is a polynomial were available, then this could be used in `fdaLm` instead enhancing the scope of this function.

Author(s)

Bo Markussen <bomar@math.ku.dk>

References

Solved using Section 5.6 in Press et al, "Numerical Recipies in C", second edition.

Examples

```
findRoots(c(-1,0,1),1)
findRoots(c(1,-1,1),2)
```

Index