Package 'funreg'

August 24, 2016

Type Package

Title Functional Regression for Irregularly Timed Data

Version 1.2

Date 2016-08-22

Depends R (>= 2.15.0)

Imports MASS, mgcv, mvtnorm, splines

Description Performs functional regression, and some related approaches, for intensive longitudinal data (see the book by Walls & Schafer, 2006, Models for Intensive Longitudinal Data, Oxford) when such data is not necessarily observed on an equally spaced grid of times. The approach generally follows the ideas of Goldsmith, Bobb, Crainiceanu, Caffo, and Reich (2011)<DOI:10.1198/jcgs.2010.10007> and the approach taken in their sample code, but with some modifications to make it more feasible to use with long rather than wide, non-rectangular longitudinal datasets with unequal and potentially random measurement times. It also allows easy plotting of the correlation between the smoothed covariate and the outcome as a function of time, which can add additional insights on how to interpret a functional regression. Additionally, it also provides several permutation tests for the significance of the functional predictor. The heuristic interpretation of ``time" is used to describe the index of the functional predictor, but the same methods can equally be used for another unidimensional continuous index, such as space along a north-south axis. The development of this package was part of a research project supported by Award R03 CA171809-01 from the National Cancer Institute and Award P50 DA010075 from the National Institute on Drug Abuse. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institute on Drug Abuse, the National Cancer Institute, or the National Institutes of Health.

License GPL (>= 2)

LazyData true

Author John Dziak [aut, cre],
Mariya Shiyko [aut]

Maintainer John Dziak <jjd264@psu.edu>

NeedsCompilation no

Repository CRAN

Date/Publication 2016-08-24 18:38:32

R topics documented:

`coef.funreg` *coef method for funreg object*

Description

Returns coefficient information on a `funreg` object.

Usage

```
## S3 method for class 'funreg'
coef(object, digits = 4, silent = FALSE, ...)
```

Arguments

object	An object of class funreg
digits	The number of digits past the decimal place to use when printing numbers
silent	If TRUE, indicates that the summary should be returned as a list object but not printed to the screen.
...	Other arguments that may be passed from another method.

Value

At least for now, this is identical to the summary.funreg function.

---

fitted.funeigen *fitted method for funeigen object*

Description

Returns fitted values for a funeigen object.

Usage

```
## S3 method for class 'funeigen'
fitted(object, type = "functions", ...)
```

Arguments

object	A funeigen object.
type	A character string, one of the following: functions, eigenfunctions, loadings, eigenvalues, mean, centered, covariance, noise.variance, midpoints. These are the constructs for which fitted values can be returned.
...	Other optional arguments which may be passed from other methods but ignored by this one.

Details

A funeigen object represents a principal component analysis of irregular longitudinal data, following the method used by Goldsmith et al. (2011).

Value

A matrix or vector containing the appropriate fitted values. What is returned depends on the type parameter. functions gives the fitted values of the smooth latent x(t) functions at a grid of time points. eigenfunctions gives the estimated eigenfunctions at each time point. loadings gives the loading of each subject on each estimated eigenfunction. mean gives the mean value for the smooth latent x(t) functions. centered gives the centered x(t) functions (the estimated function subtracting the mean function) . covariance gives the estimated covariance matrix of x(s) and x(t) on a grid of time points s and t. noise.variance gives the estimated measurement error variance on the x(t)

functions. midpoints gives the time points for the grid, on which functions, mean, centered, and covariance are defined; they are viewed as midpoints of bins of observation times (see Goldsmith et al., 2011).

References

Goldsmith, J., Bobb, J., Crainiceanu, C. M., Caffo, B., and Reich, D. (2011). Penalized functional regression. Journal of Computational and Graphical Statistics, 20(4), 830-851. DOI: 10.1198/jcgs.2010.10007.

---

fitted.funreg *fitted method for funreg object*

Description

Returns fitted values for a funreg object.

Usage

```
## S3 method for class 'funreg'
fitted(object, type = "response", which.coef = 1, ...)
```

Arguments

object	A funreg object
type	Either response or correlation. If response, fitted values for the scalar response variable are returned. If correlation, estimated individual-level correlation coefficients of the smoothed value of the functional covariate at various time points with the scalar response variable are returned.
which.coef	Only required if type is correlation and there is more than one functional covariate. This specifies which functional covariate is of interest.
...	Other optional arguments which may be passed from other methods but ignored by this one.

Value

Returns the fitted values for the responses if type is response, or the fitted values for the correlations of the which.coefth functional covariate with the response, if type is correlation.

funeigen *Perform eigenfunction decomposition on functional covariate*

Description

A function to do the eigenfunction decomposition as part of a penalized functional regression as in Goldsmith et al. (2011)

Usage

```
funeigen(id, time, x, num.bins = 35, preferred.num.eigenfunctions = 30)
```

Arguments

id	A vector of subject ID's.
time	A vector of measurement times.
x	A single functional predictor represented as a vector or a one-column matrix.
num.bins	The number of knots used in the spline basis for the beta function. The default is based on the Goldsmith et al. (2011) sample code.
preferred.num.eigenfunctions	The number of eigenfunctions to use in approximating the covariance function of x (see Goldsmith et al., 2011)

Note

The algorithm for this function follows that of "sparse_simulation.R", which was written on Nov. 13, 2009, by Jeff Goldsmith; Goldsmith noted that he used some code from Chongzhi Di for the part about handling sparsity. "sparse_simulation.R" was part of the supplementary material for Goldsmith, Bobb, Crainiceanu, Caffo, and Reich (2011). The sample code can be found at http://www.jeffgoldsmith.com/Downloads/PFR_Code.zip. The num.bins parameter corresponds to N.fit in Goldsmith et al, sparse_simulation.R and preferred.num.eigenfunctions corresponds to Kz in Goldsmith et al.

References

Goldsmith, J., Bobb, J., Crainiceanu, C. M., Caffo, B., and Reich, D. (2011). Penalized functional regression. Journal of Computational and Graphical Statistics, 20(4), 830-851. DOI: 10.1198/jcgs.2010.10007.

See Also

fitted.funeigen, link{plot.funeigen}

funreg *Perform penalized functional regression*

Description

Performs a penalized functional regression as in Goldsmith et al. (2012) on irregularly measured data such as that found in ecological momentary assessment (see Walls & Schafer, 2006; Shiffman, Stone, & Hufford, 2008).

Usage

```
funreg(id, response, time, x, basis.method = 1, deg = 2, deg.penalty = 2,
  family = gaussian, other.covariates = NULL, num.bins = 35,
  preferred.num.eigenfunctions = 30, preferred.num.knots.for.beta = 35,
  se.method = 1, smoothing.method = 1, times.for.fit.grid = NULL)
```

Arguments

id	An integer or string uniquely identifying the subject to which each observation belongs
response	The response, as a vector, one for each subject
time	The time of observation, as a vector, one for each observation (i.e., each assessment on each person)
x	The functional predictor(s), as a matrix, one row for each observation (i.e., for each assessment on each person)
basis.method	An integer, either 1 or 2, describing how the beta function should be internally modeled. A value of 1 indicates that a truncated power spline basis should be used, and a value of 2 indicates that a B-spline basis should be used.
deg	An integer, either 1, 2, or 3, describing how complicated the behavior of the beta function between knots may be. 1, 2, or 3 represent linear, quadratic or cubic function between knots.
deg.penalty	Only relevant for B-splines. The difference order used to weight the smoothing penalty (see Eilers and Marx, 1996)
family	The response distribution. For example, this is family=gaussian for normal linear models, family=binomial for logistic regression models, or family=poisson for count models. See the gam documentation in the mgcv package, or use help(family) for details on family objects.
other.covariates	Subject-level (time-invariant) covariates, if any, as a matrix, one column per covariate. The default, NULL, means that no subject-level covariates will be included in the model.
num.bins	The number of knots used in the spline basis for the beta function. The default is based on the Goldsmith et al. (2011) sample code.

`preferred.num.eigenfunctions`
: The number of eigenfunctions to use in approximating the covariance function of x (see Goldsmith et al., 2011)

`preferred.num.knots.for.beta`
: number of knots to use in the spline estimation. The default, is based on the Goldsmith et al (2011) sample code.

`se.method`
: An integer, either 1 or 2, describing how the standard errors should be calculated. A value of 1 means that the uncertainty related to selecting the smoothing parameter is ignored. Option 2 means that a Bayesian approach is used to try to take this uncertainty into account (see the documentation for Wood's `mgcv` package).

`smoothing.method`
: An integer, either 1 or 2, describing how the weight of the smoothing penalty should be determined. Option 1 means that the smoothing weight should be estimated using an approach similar to restricted maximum likelihood, and Option 2 means an approach similar to generalized cross-validation. Option 1 is strongly recommended (based both on our experience and on remarks in the documentation for the gam function in the mgcv package).

`times.for.fit.grid`
: Points at which to calculate the estimated beta function. The default, NULL, means that the code will choose these times automatically.

Value

An object of type `funreg`. This object can be examined using `summary`, `print`, or `fitted`.

Note

This function mostly follows code by Jeff Goldsmith and co-workers: the sample code from Goldsmith et al (2011), and the "pfr" function in the "refund" R package. However, this code is adapted here to allow idiosyncratic measurement times and unequal numbers of observations per subject to be handled easily, and also allows the use of a different estimation method. Also follows some sample code for penalized B-splines from Eilers and Marx (1996) in implementing B-splines. As the pfr function in refund also does, the function calls the gam function in the mgcv package (Wood 2011) to do much of the internal calculations.

In the example below, to fit a more complicated model, replace `x=SampleFunregData$x1` with `x=cbind(SampleFunregData$x1, SampleFunregData$x2),other.covariates=cbind(SampleFunregData$s1, Samp`
This model will take longer to run, perhaps 10 or 20 seconds. Then try `plot(complex.model)`.

References

Crainiceanu, C., Reiss, P., Goldsmith, J., Huang, L., Huo, L., Scheipl, F. (2012). refund: Regression with Functional Data (version 0.1-6). R package Available online at cran.r-project.org.

Eilers, P. H. C., and Marx, B. D. (1996). Flexible smoothing with B-splines and penalties. Statistical Science, 11, 89-121. DOI:10.1.1.47.4521.

Goldsmith, J., Bobb, J., Crainiceanu, C. M., Caffo, B., and Reich, D. (2011). Penalized functional regression. Journal of Computational and Graphical Statistics, 20(4), 830-851. DOI: 10.1198/jcgs.2010.10007.

The sample code can be found at www.jeffgoldsmith.com/Downloads/PFR_Code.zip; in writing parts of this function I especially followed "PFR_Example.R", written on Jan. 15 2010, by Jeff Goldsmith.

Ruppert, D., Wand, M., and Carroll, R. (2003). Semiparametric regression. Cambridge, UK: Cambridge University Press.

Shiffman, S., Stone, A. A., and Hufford, M. R. (2008). Ecological momentary assessment. Annual Review of Clinical Psychology, 4, 1-32. DOI:10.1146/annurev.clinpsy.3.022806.091415.

Walls, T. A., & Schafer, J. L. (2006) Models for intensive longitudinal data. New York: Oxford.

Wood, S.N. (2006) Generalized Additive Models: An Introduction with R. Chapman and Hall/CRC.

Wood, S.N. (2011) Fast stable restricted maximum likelihood and marginal likelihood estimation of semiparametric generalized linear models. Journal of the Royal Statistical Society (B) 73(1):3-36. DOI:10.1111/j.1467-9868.2010.00749.x

See Also

fitted.funreg, link{plot.funreg}, print.funreg, link{summary.funreg}

Examples

```
simple.model <- funreg(id=SampleFunregData$id,
                       response=SampleFunregData$y,
                       time=SampleFunregData$time,
                       x=SampleFunregData$x1,
                       family=binomial);
print(simple.model);
par(mfrow=c(2,2));
plot(x=simple.model$model.for.x[[1]]$bin.midpoints,
     y=simple.model$model.for.x[[1]]$mu.x.by.bin,
     xlab="Time t",ylab="X(t)",main="Smoothed mean x values");
# The smoothed average value of the predictor function x(t) at different times t.
# The ``[[1]]'' after model.for.x is there because model.for.x is a list with one entry.
# This is because more than one functional covariate is allowed.
plot(simple.model,type="correlations");
# The marginal correlation of x(t) with y at different times t.
# It appears that earlier time points are more strongly related to y.
plot(simple.model,type="coefficients");
# The functional regression coefficient of y on x(t).
# It also appears that earlier time points are more strongly related to y.
plot(simple.model$subject.info$response,
     simple.model$subject.info$fitted,
     main="Predictive Performance",
     xlab="True Y",
     ylab="Fitted Y");
```

`funreg.permutation` *Do a permutation test for functional regression*

Description

Performs a permutation F test (Ramsay, Hooker, and Graves, 2009, p. 145) for the significance of a functional covariate, and a permutation likelihood ratio test. The permutation test function currently doesn't allow models with multiple functional covariates, but subject-level covariates are allowed.

Usage

```
funreg.permutation(object, num.permute = 500, seed = NULL)
```

Arguments

`object`	An object of class funreg
`num.permute`	The number of permutations to use. Ramsay, Hooker and Graves (2009) recommended "several hundred" (p. 145), but for a quicker initial look it might suffice to use 100.
`seed`	An optional random number seed.

Value

Returns a list with several components. First, `pvalue.F` is the p-value for the F test. Second, `conf.int.for.pvalue.F` is the confidence interval for estimating the p-value that would be obtained from the dataset as `num.permute` approached infinity. The idea of a confidence interval for a p-value is explained further by Sen (2013), with a STATA example. See "Permutation Tests" by Saunak Sen (2013) at http://www.epibiostat.ucsf.edu/biostat/sen/statgen/permutation.html. Third, `orig.F` is the F statistic calculated on the original dataset. Last, `permuted.F` is the vector of F statistics calculated on each of the random permuted datasets. Also included are `pvalue.LR`, `conf.int.for.pvalue.LR`, `orig.LR`, `permuted.LR` for the permutation test with a likelihood ratio statistic. A more conservative alternative formula for the p-value is used in `pvalue.F.better` and `pvalue.LR.better`. It is not obvious whether to define the p-value as the proportion of permuted datasets with statistics less than or equal to the original, or simply less than the original. This should usually not matter, as a tie is not likely. We made the arbitrary decision to use the former here because it was presented in this way in the Wikipedia article for permutation tests. The conservative alternative formula is the number of less extreme permuted datasets plus one, over the total number of datasets plus one. Adding one to the numerator and denominator is suggested by some authors, partly in order to prevent a nonsensical zero p-value (Onghena & May, 1995; Phipson, Belinda & Smyth, 2010).

References

Onghena, P., & May, R. B. (1995). Pitfalls in computing and interpreting randomization test p values: A commentary on Chen and Dunlap. Behavior Research Methods, Instruments, & Computers, 27(3), 408-411. DOI: 10.3758/BF03200438.

Phipson, Belinda and Smyth, Gordon K. (2010) Permutation P-values Should Never Be Zero: Calculating Exact P-values When Permutations Are Randomly Drawn. Statistical Applications in Genetics and Molecular Biology: Vol. 9: Iss. 1, Article 39. DOI: 10.2202/1544-6115.1585.

Ramsay, J. O., Hooker, G., & Graves, S. (2009). Functional data analysis with R and MATLAB. NY: Springer.

Sen, S. (2014) Permutation Tests. Available at http://www.biostat.ucsf.edu/sen/statgen14/permutation-tests.html

---

`generate.data.for.demonstration`

Generate data for some demonstration examples

Description

Simulates a dataset with two functional covariates, four subject-level scalar covariates, and a binary outcome.

Usage

```
generate.data.for.demonstration(nsub = 400, b0.true = -5, b1.true = 0,
  b2.true = +1, b3.true = -1, b4.true = +1, nobs = 500,
  observe.rate = 0.1)
```

Arguments

`nsub`	The number of subjects in the simulated dataset.
`b0.true`	The true value of the intercept.
`b1.true`	The true value of the first covariate.
`b2.true`	The true value of the second covariate.
`b3.true`	The true value of the third covariate.
`b4.true`	The true value of the fourth covariate.
`nobs`	The total number of possible observation times.
`observe.rate`	The average proportion of those possible times at which any given subject is observed.

Value

Returns a `data.frame` representing `nobs` measurements for each subject. The rows of this `data.frame` tell the values of two time-varying covariates on a dense grid of `nobs` observation times. It also contains an `id` variable, four subject-level covariates (`s1`, ..., `s4`) and one subject-level response (`y`), which are replicated for each observation. For each observation, there is also its observation time `time`, there are both the smooth latent value of the covariates (`true.x1` and `true.x2`) and versions observed with error (`x1` and `x2`), and there are also the local values of the functional regression coefficients (`true.betafn1` and `true.betafn2`). Lastly, each row has a random value for `include.in.subsample`, telling whether it should be considered as an observed data point (versus an unobserved moment in the simulated subject's life). `include.in.subsample` is simply generated as a Bernoulli random variable with success probability `observe.rate`.

Note

nobs is the number of simulated data rows per simulated subject. It should be selected to be large because x covariates are conceptually supposed to be smooth functions of time. However, in the simulated data analyses we actually only use a small random subset of the generated time points, because this is more realistic for many behavioral and medical science datasets. Thus, the number of possible observation times per subject is nobs, and the mean number of actual observation times per subject is nobs times observe.rate. This smaller 'observed' dataset can be obtained by deleting from the dataset those observations having include.in.subsample==FALSE.

---

make.funreg.basis *Make basis for functional regression (for internal use by other package functions)*

Description

This is a function for internal use (i.e., a user will not need to call it directly for usual data analysis tasks). Recall that functional coefficients are estimated as a linear combination of basis functions, thus changing a nonparametric into a parametric estimation problem. This function constructs the matrix of basis function values for doing a functional regression.

Usage

```
make.funreg.basis(basis.type, deg, num.knots, times)
```

Arguments

basis.type	is a character string, either TruncatedPower or BSpline. This tells whether the basis functions should be calculated as B-splines (see Eilers and Marx, 1996) or as truncated power splines (see Ruppert, Wand, and Carroll, 2003).
deg	is the degree of the basis functions (roughly, their amount of complexity) and should generally be 1, 2, or 3.
num.knots	is the number of knots in the basis; the higher this is, the more flexible the estimated function will be. If it is too low, the estimated function may be too simple (i.e.,biased towards being too smooth). If it is too high, the function may be hard to interpret.
times	is the vector of measurement times (more technically, real-valued index values for the functional covariate) at which the basis functions should be evaluated.

Value

Returns a list with two components. The first, interior.knot.locations, tells the selected locations on the time axis for each interior knot. The second, basis.for.betafn, is a matrix with one row for each time value in the input vector times and one column for each basis function. It represents the values of the basis functions themselves.

References

Eilers, P. H. C., and Marx, B. D. (1996). Flexible smoothing with B-splines and penalties (with comments and rejoinder). Statistical Science, 11, 89-121.

Ruppert, D., Wand, M. P., and Carroll, R. J. (2003) Semiparametric regression. Cambridge: Cambridge.

`marginal.cor` *Calculate marginal correlations with response*

Description

Calculates marginal correlations between a functional covariate and a scalar response.

Usage

```
marginal.cor(object, id = NULL, response = NULL, alpha = 0.05)
```

Arguments

`object`	An object of type `funeigen` or `funreg`. One or the other of these is needed in order to provide a smoothed reconstructed curves for the functional covariate for each subject.
`id`	The vector of subject id's. These tell which responses in `response` correspond to which curves in `object`.
`response`	The vector of responses
`alpha`	The alpha level for confidence intervals (one minus the two-sided coverage)

Value

Returns a list with one component for each functional covariate. Each such component contains the between-subjects correlations between the fitted smoothed latent values of the functional covariate, and the response variable. We call this a marginal correlation because it simply ignores the other functional covariates (rather than trying to adjust or control for them). Both the functional regression coefficient and the marginal correlation can be useful, although they have different substantive interpretations.

`marginal.cor.funeigen` *Calculate marginal correlations with response, from a funeigen object*

Description

A function for internal use. Its main job is to be called by `MarginalCor`, and do the technical work for calculating estimated marginal correlations. It uses R. A. Fisher's classic r-to-z transform to create confidence intervals for the correlations. This process is explained in easy-to-follow detail by David Shen and Zaizai Lu in a technical report.

Usage

```
marginal.cor.funeigen(object, id, response, alpha = 0.05)
```

Arguments

`object`	An object of type `funeigen`.
`id`	The vector of subject id's. These tell which responses in `response` correspond to which curves in `object`.
`response`	The vector of responses
`alpha`	The alpha level for confidence intervals (one minus the two-sided coverage)

Value

Returns a `data.frame` with four columns. The first, `time`, is the time index of the rows. That is, it is a grid of points t along the time axis and these points correspond to the rows. The next three are the lower bound, best estimate, and upper bound, of the correlation between the smoothed value of the covariate x(t) and the response y at each of the time points t. We refer to the correlation function estimated here as marginal because it ignores any other functional covariates (rather than trying to adjust or control for them).

Note

The confidence intervals are simply based on Fisher's r-to-z transform and do not take into account the uncertainty in estimating the smoothed value of x(t).

References

Shen, D., and Lu, Z. (2006). Computation of correlation coefficient and its confidence interval in SAS (r). SUGI 31 (March 26-29, 2006), paper 170-31. Available online at http://www2.sas.com/proceedings/sugi31/170-31.pdf.

num.functional.covs.in.model *Count the functional covariates in a model (for internal use by other package functions)*

Description

A very simple function, mainly for internal use by package code, to count the number of functional covariates in an object of class `funreg`.

Usage

```
num.functional.covs.in.model(object)
```

Arguments

`object`	An object of class `funreg`, representing a fitted penalized functional regression with one or more functional covariates.

Value

The number of functional covariates as an integer.

plot.funeigen *plot method for funeigen object*

Description

Creates a visual representation of some of the information in an object of class `funeigen` (i.e., in an eigenfunction decomposition of a functional variable). Several kinds of plots are available.

Usage

```
## S3 method for class 'funeigen'
plot(x, type = "correlation", how.many = NULL,
  xlab = "", ylab = "", ...)
```

Arguments

`x`	An object of class `funeigen`
`type`	A character string telling the kind of information to include in the plot. It may be `functions`, `eigenfunctions`, `eigenvalues`, `mean`, `covariance`, or `correlation`.
`how.many`	How many fitted curves to show (in a plot of fitted curves; the default is all of them), or how many estimated eigenfunctions to show (in a plot of eigenfunctions; the default is all of them)

xlab	Label for the x axis of the plot.
ylab	Label for the y axis of the plot.
...	Other optional arguments to be passed on to the plot function.

`plot.funreg` *plot method for funreg object*

Description

Plots information from an object of class `funreg`.

Usage

```
## S3 method for class 'funreg'
plot(x, frames = FALSE, type = "coefficients", ...)
```

Arguments

x	An object of class `funreg`, representing a fitted functional regression model.
frames	If there are multiple functional covariates, this tells whether or not the plot for each covariate should all be drawn together as different panels on the same figure.
type	A string telling what kind of plot to produce. One can specify `coefficients`, which means that the functional coefficient for each functional covariate will be plotted as a function of time; or one can specify
...	Other optional arguments to be passed on to the plot function. `correlations`, which means that the correlation of the covariate (at each given time), with the scalar outcome (presumably taken at a single time) will be plotted instead.

`print.funreg` *print method for funreg object*

Description

Prints information from an object of class `funreg`.

Usage

```
## S3 method for class 'funreg'
print(x, digits = 4, show.fits = FALSE, ...)
```

Arguments

x	Object of class funreg.
digits	Number of digits past the decimal place to show when printing quantities.
show.fits	Whether to also print a table of fitted values for the functional coefficients along a grid of times.
...	Any other optional arguments to be passed to the default print function.

---

redo.funreg *Redo a funreg with different data (for internal use by permutation test)*

Description

For internal use by package functions. Not intended to be directly called by the data analyst. This function repeats the analysis for an object of class funreg, with the same settings but with possibly different data. This is convenient in doing resampling techniques like bootstrapping or permutation testing.

Usage

```
redo.funreg(object, id, response, time, other.covariates, x)
```

Arguments

object	A funreg object.
id	The new values for id (see the funreg function documentation)
response	The new values for response
time	The new values for time
other.covariates	The new values for other.covariates (which may be NULL)
x	The new values for the functional covariate.

Value

The funreg object for the new fitted model.

SampleFunregData *Sample dataset for funreg*

Description

A data frame generated using the following code: `set.seed(123); SampleFunregData <- generate.data.for.de`

Format

A data frame for a simulated longitudinal study, in "tall" rather than "wide" format (multiple rows per individual, one for each measurement time) with 8109 rows and 13 columns.

id Integer uniquely identifying the subject to whom this data row pertains.

s1,s2,s3,s4 Four subject-level (time-invariant) covariates.

y A response, coded as 0 or 1, which is to be modeled using a functional regression. It is also subject-level (i.e., either time-invariant or measured only once). However, like `s1` through `s4`, its value is repeated for each row of data for a subject.

time The time variable, arbitrarily chosen to range from a low of 0 to a high of 10, which identifies when this row's observations are taken.

true.x1,true.x2 The unknown smooth expected values of two time-varying variables which can be treated as functional covariates. They vary by subject and time and are therefore different in each row.

true.betafn1,true.betafn2 The unknown true functional regression coefficient function used to generate `y` from the two time-varying predictors. The latter is always zero because x2 is unrelated to y.

x1,x2 The observed values of the two functional regression predictors, as measured for a given time on a given subject.

summary.funreg *summary method for funreg object*

Description

Returns summary information on a `funreg` object.

Usage

```
## S3 method for class 'funreg'
summary(object, digits = 4, silent = FALSE, ...)
```

Arguments

object	An object of class funreg
digits	The number of digits past the decimal place to use when printing numbers
silent	If TRUE, indicates that the summary should be returned as a list object but not printed to the screen.
...	Any other optional arguments that may be passed from other methods (but currently ignored by this one).

Value

Returns a list with four components. First, call.info summarizes the inputs that were sent into the funreg function. Second, intercept.estimate.uncentered gives the estimated functional coefficient for the intercept in the functional regression model. Third, functional.covariates.table provides estimated values for the functional coefficients at each of a grid of time points. Fourth, subject.level.covariates.table provides estimated values for subject-level covariates if any are in the model.

Index