Package 'gppm'

August 25, 2025

Version 0.3.0

Title Gaussian Process Panel Modeling

Description Provides an implementation of Gaussian process panel modeling (GPPM). GPPM is described in Karch, Brandmaier & Voelkle (2020; <DOI:10.3389/fpsyg.2020.00351>) and Karch (2016; <DOI:10.18452/17641>). Essentially, GPPM is Gaussian process based modeling of longitudinal panel data. 'gppm' also supports regular Gaussian process regression (with a focus on flexible model specification), and multi-task learning.

Depends R (>= 3.5)

Imports ggplot2 (>= 2.2.1), ggthemes (>= 3.5.0), MASS (>= 7.3-49), methods, mvtnorm (>= 1.0-8), Rcpp (>= 0.12.0), RcppParallel (>= 5.0.1), rstan (>= 2.18.1), rstantools (>= 2.4.0), stats,

Suggests testthat (>= 2.0.0), knitr (>= 1.20), rmarkdown (>= 1.10), roxygen2 (>= 6.0.1), testit (>= 0.12), covr

LinkingTo BH (>= 1.66.0), Rcpp (>= 0.12.0), RcppEigen (>= 0.3.3.3.0), RcppParallel (>= 5.0.1), rstan (>= 2.18.1), StanHeaders (>= 2.18.0)

License GPL-3 | file LICENSE

LazyData true

URL https://github.com/karchjd/gppm

BugReports https://github.com/karchjd/gppm/issues

RoxygenNote 7.3.2

Encoding UTF-8

Biarch true

SystemRequirements GNU make

NeedsCompilation no

Author Julian D. Karch [aut, cre, cph]

Maintainer Julian D. Karch <j.d.karch@fsw.leidenuniv.nl>

Repository CRAN

Date/Publication 2025-08-25 08:40:08 UTC

Contents

`accuracy` *Accuracy Estimates for Predictions*

Description

Estimate the accuracy based on predictions.

Usage

```
accuracy(predRes)
```

Arguments

predRes	object of class GPPMPred as obtained by predict.GPPM

Value

accuracy estimates in the form of the mean squared error (MSE), the negative log-predictive probability (nLPP), and the sum squared error (SSE)

Examples

```
data("demoLGCM")
# remove all measurements from person 1 and the first form person 2
predIdx <- c(which(demoLGCM$ID == 1), which(demoLGCM$ID == 2)[1])
fitDemoLGCM <- demoLGCM[setdiff(1:nrow(demoLGCM), predIdx), ]

lgcm <- gppm(
  "muI+muS*t", "varI+covIS*(t+t#)+varS*t*t#+(t==t#)*sigma",
  fitDemoLGCM, "ID", "y"
)
lgcm <- fit(lgcm)
predRes <- predict(lgcm, demoLGCM[predIdx, ])
accEsts <- accuracy(predRes)
accEsts$MSE # mean squared error
accEsts$nLPP # negative log-predictive probability
accEsts$MAE # mean absolute error
```

---

coef.GPPM *Point Estimates*

Description

Extracts point estimates for all parameters from a fitted GPPM.

Usage

```
## S3 method for class 'GPPM'
coef(object, ...)
```

Arguments

object	object of class GPPM. Must be fitted, that is, a result from fit.GPPM.
...	additional arguments (currently not used).

Value

Point estimates for all parameters as a named numeric vector.

See Also

Other functions to extract from a GPPM: SE(), confint.GPPM(), covFun(), fitted.GPPM(), getData(), getIntern(), logLik.GPPM(), maxNObs(), meanFun(), nObs(), nPars(), nPers(), nPreds(), parEsts(), pars(), preds(), vcov.GPPM()

Examples

```
data("demoLGCM")
lgcm <- gppm(
  "muI+muS*t", "varI+covIS*(t+t#)+varS*t*t#+(t==t#)*sigma",
  demoLGCM, "ID", "y"
)
lgcmFit <- fit(lgcm)
paraEsts <- coef(lgcmFit)
```

---

confint.GPPM *Confidence Intervals*

Description

Computes confidence intervals for one or more parameters in a fitted GPPM.

Usage

```
## S3 method for class 'GPPM'
confint(object, parm, level = 0.95, ...)
```

Arguments

object	object of class GPPM. Must be fitted, that is, a result from fit.GPPM.
parm	vector of strings. The parameters for which confidence intervals are desired. If missing, confidence intervals for all parameters are returned.
level	scalar from 0 to 1. The confidence level required.
...	additional arguments (currently not used).

Value

A matrix (or vector) with columns giving lower and upper confidence limits for each parameter. These will be labeled as (1-level)/2 and 1 - (1-level)/2 in \

See Also

Other functions to extract from a GPPM: SE(), coef.GPPM(), covFun(), fitted.GPPM(), getData(), getIntern(), logLik.GPPM(), maxNObs(), meanFun(), nObs(), nPars(), nPers(), nPreds(), parEsts(), pars(), preds(), vcov.GPPM()

Examples

```
data("demoLGCM")
lgcm <- gppm(
  "muI+muS*t", "varI+covIS*(t+t#)+varS*t*t#+(t==t#)*sigma",
  demoLGCM, "ID", "y"
)
lgcmFit <- fit(lgcm)
confInts <- confint(lgcmFit)
```

---

covFun *Covariance Function*

Description

Extracts the covariance function from a GPPM.

Usage

```
covFun(gpModel)
```

Arguments

gpModel	object of class GPPM.

Value

The covariance function as a character string.

See Also

Other functions to extract from a GPPM: SE(), coef.GPPM(), confint.GPPM(), fitted.GPPM(), getData(), getIntern(), logLik.GPPM(), maxNObs(), meanFun(), nObs(), nPars(), nPers(), nPreds(), parEsts(), pars(), preds(), vcov.GPPM()

Examples

```
data("demoLGCM")
lgcm <- gppm(
  "muI+muS*t", "varI+covIS*(t+t#)+varS*t*t#+(t==t#)*sigma",
  demoLGCM, "ID", "y"
)
myCov <- covFun(lgcm)
```

createLeavePersonsOutFolds *Create Leave-persons-out Folds*

Description

This function is used to create a leave-persons-out cross-validation fold vector to be used by crossvalidate.

Usage

```
createLeavePersonsOutFolds(gpModel, k = 10)
```

Arguments

gpModel	object of class GPPM.
k	integer scalar. Number of folds to create.

Details

The folds are created such that the data of each person is fully in one fold.

Value

A fold vector, which is a vector of length nrow(getData(gpModel)) of integers from 1 to k. If foldVector[i]=j, then data point i is assigned to fold j.

See Also

crossvalidate for how to use the created fold vector to perform cross-validation.

Examples

```
data("demoLGCM")
lgcm <- gppm(
  "muI+muS*t", "varI+covIS*(t+t#)+varS*t*t#+(t==t#)*sigma",
  demoLGCM, "ID", "y"
)
theFolds <- createLeavePersonsOutFolds(lgcm)
```

`crossvalidate` *Cross-validation.*

Description

Performs cross-validation of a Gaussian process panel model.

Usage

```
crossvalidate(gpModel, foldVector)
```

Arguments

`gpModel`	object of class GPPM.
`foldVector`	integer vector. Describes the foldstructure to use. For example, created by `createLeavePersonsOutFolds`.

Details

The fold vector, must be a vector of length nrow(getData(gpModel)) of integers from 1 to k. If `foldVector[i]=j`, then data point i is assigned to fold j.

Value

Cross-validation estimates of the mean squared error (MSE) and the negative log-predictive probability (nLPP)

Examples

```
data("demoLGCM")
lgcm <- gppm(
  "muI+muS*t", "varI+covIS*(t+t#)+varS*t*t#+(t==t#)*sigma",
  demoLGCM, "ID", "y"
)
theFolds <- createLeavePersonsOutFolds(lgcm, k = 2) # for speed, in practive rather use default k=10
crosRes <- crossvalidate(lgcm, theFolds)
crosRes$MSE # mean squared error
crosRes$nLPP # negative log-predictive probability
```

demoLGCM *Simulated Data From a Latent Growth Curve Model.*

Description

Simulated Data From a Latent Growth Curve Model.

Usage

```
demoLGCM
```

Format

A data frame with 1998 rows and 3 variables:

ID Subject ID

t Time index

y Generic measurement

fit *Generic Method For Fitting a model*

Description

Generic method for fitting a model.

Usage

```
fit(gpModel, ...)
```

Arguments

gpModel	a model.
...	additional arguments.

Value

A fitted model

See Also

fit.GPPM

`fit.GPPM` *Fit a Gaussian process panel model*

Description

This function is used to fit a Gaussian process panel model, which has been specified fit using `gppm`.

Usage

```
## S3 method for class 'GPPM'
fit(
  gpModel,
  init = "random",
  useOptimizer = TRUE,
  verbose = FALSE,
  hessian = TRUE,
  ...
)
```

Arguments

`gpModel`	object of class GPPM. The Gaussian process panel model to be fitted.
`init`	string or named numeric vector. Used to specify the starting values for the parameters. Can either be the string 'random' (default) or a numeric vector startVal of starting values. Which value belongs to which parameter is determined by the names attribute of startVal. See also the example.
`useOptimizer`	boolean. Should the optimizer be used or not? For false the (possibly random) starting values are returned as the maximum likelihood estimates.
`verbose`	boolean. Print diagnostic output?
`hessian`	boolean. Compute the hessian at the maximum likelihood estimate?
`...`	additional arguments (currently not used).

Value

A fitted Gaussian process panel model, which is an object of class 'GPPM'.

See Also

Functions to extract from a fitted GPPM:

Examples

```
# regular usage
data("demoLGCM")
lgcm <- gppm(
  "muI+muS*t", "varI+covIS*(t+t#)+varS*t*t#+(t==t#)*sigma",
```

```
  demoLGCM, "ID", "y"
)
lgcmFit <- fit(lgcm)

# starting values as ML results
startVals <- c(10, 1, 10, 3, 10, 1)
names(startVals) <- pars(lgcm)
lgcmFakeFit <- fit(lgcm, init = startVals, useOptimizer = FALSE)
stopifnot(identical(startVals, coef(lgcmFakeFit)))
```

`fitted.GPPM` *Person-specific mean vectors and covariance matrices*

Description

A fitted GPPM implies a mean vector and a covariance matrix for each person. These are returned by this function.

Usage

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

Arguments

`object`	object of class GPPM. Must be fitted, that is, a result from `fit.GPPM`.
`...`	additional arguments (currently not used).

Value

Returns a list structure with mean and covariances matrices. See example.

See Also

Other functions to extract from a GPPM: `SE()`, `coef.GPPM()`, `confint.GPPM()`, `covFun()`, `getData()`, `getIntern()`, `logLik.GPPM()`, `maxNObs()`, `meanFun()`, `nObs()`, `nPars()`, `nPers()`, `nPreds()`, `parEsts()`, `pars()`, `preds()`, `vcov.GPPM()`

Examples

```
data("demoLGCM")
lgcm <- gppm(
  "muI+muS*t", "varI+covIS*(t+t#)+varS*t*t#+(t==t#)*sigma",
  demoLGCM, "ID", "y"
)
lgcmFit <- fit(lgcm)
meansCovs <- fitted(lgcmFit)
```

```
person1Mean <- meansCovs$mean[[1]]
person1Cov <- meansCovs$cov[[1]]
person1ID <- meansCovs$ID[[1]]
```

---

getData *Data Set*

Description

Extracts the data set from a GPPM.

Usage

```
getData(gpModel)
```

Arguments

gpModel	object of class GPPM.

Value

The data set associated with the GPPM.

See Also

Other functions to extract from a GPPM: SE(), coef.GPPM(), confint.GPPM(), covFun(), fitted.GPPM(), getIntern(), logLik.GPPM(), maxNObs(), meanFun(), nObs(), nPars(), nPers(), nPreds(), parEsts(), pars(), preds(), vcov.GPPM()

Examples

```
data("demoLGCM")
lgcm <- gppm(
  "muI+muS*t", "varI+covIS*(t+t#)+varS*t*t#+(t==t#)*sigma",
  demoLGCM, "ID", "y"
)
myData <- getData(lgcm)
```

getIntern *Generic Extraction Function*

Description

Extracts internals from a GPPM.

Usage

```
getIntern(gpModel, quantity)
```

Arguments

gpModel	object of class GPPM.
quantity	character string. Name of the quantity to extract. Possible values are • "parsedmFormula" for the parsed mean formula • "parsedcFormula" for the parsed covariance formula • "stanData" for the data set in the form needed for rstan • "stanModel" for the created rstan model • "stanOut" for the created stan output

Value

The requested quantity

See Also

Other functions to extract from a GPPM: SE(), coef.GPPM(), confint.GPPM(), covFun(), fitted.GPPM(), getData(), logLik.GPPM(), maxNObs(), meanFun(), nObs(), nPars(), nPers(), nPreds(), parEsts(), pars(), preds(), vcov.GPPM()

Examples

```
data("demoLGCM")
lgcm <- gppm(
  "muI+muS*t", "varI+covIS*(t+t#)+varS*t*t#+(t==t#)*sigma",
  demoLGCM, "ID", "y"
)
lgcmFit <- fit(lgcm)
getIntern(lgcmFit, "parsedmFormula")
```

gppm *Define a Gaussian process panel model*

Description

This function is used to specify a Gaussian process panel model (GPPM), which can then be fit using `fit.GPPM`.

Usage

```
gppm(mFormula, cFormula, myData, ID, DV, control = gppmControl())
```

Arguments

`mFormula`	character string. Contains the specification of the mean function. See details for more information.
`cFormula`	character string. Contains the specification of the covariance function. See details for more information.
`myData`	data frame. Contains the data to which the model is fitted. Must be in the long-format.
`ID`	character string. Contains the column label in myData which describes the subject ID.
`DV`	character string. Contains the column label in myData which contains the to be modeled variable.
`control`	object of class GPPMControl. Used for storing technical settings. Default should only be changed by advanced users. Generated via `gppmControl`.

Details

mFormula and cFormula contain the specification of the mean and the covariance function respectively. These formulas are defined using character strings. Within these strings there are four basic elements:

- Parameters
- Functions and operators
- References to observed variables in the data frame myData
- Mathematical constants

The gppm function automatically recognizes which part of the string refers to which elements. To be able to do this certain relatively common rules need to be followed:

Parameters: Parameters may not have the same name as any of the columns in myData to avoid confusing them with a reference to an observed variable. Furthermore, to avoid confusing them with functions, operators, or constants, parameter labels must always begin with a lower case letter and only contain letters and digits.

Functions and operators: All functions and operators that are supported by stan can be used; see https://mc-stan.org/docs/ for a full list. In general, all basic operators and functions are supported.

References: A reference must be the same as one of the elements of the output of `names(myData)`. For references, the same rules apply as for parameters. That is, the column names of myData may only contain letters and digits and must start with a letter.

Constants: Again, all constants that are supported by stan can be used and in general the constants are available by their usual name.

Value

A (unfitted) Gaussian process panel model, which is an object of class 'GPPM'

See Also

`fit.GPPM` for how to fit a GPPM

Examples

```
# Defintion of a latent growth curve model

data("demoLGCM")
lgcm <- gppm(
  "muI+muS*t", "varI+covIS*(t+t#)+varS*t*t#+(t==t#)*sigma",
  demoLGCM, "ID", "y"
)
```

`gppmControl` *Define settings for a Gaussian process panel model*

Description

This function is used to specify the settings of a Gaussian process panel model generated by `gppm`.

Usage

```
gppmControl(stanModel = TRUE)
```

Arguments

`stanModel`	boolean. Should the corresponding stan model be created? Should only be set to FALSE for testing purposes. Not creating the stan model makes model fitting impossible but saves a lot of time.

Value

Settings for a Gaussian process panel model in an object of class 'GPPMControl'

See Also

gppm

---

logLik.GPPM *Log-Likelihood*

Description

Compute the log-likelihood for a GPPM at the maximum likelihood parameter values.

Usage

```
## S3 method for class 'GPPM'
logLik(object, ...)
```

Arguments

object	object of class GPPM. Must be fitted, that is, a result from fit.GPPM.
...	additional arguments (currently not used).

Value

Returns an object of class logLik. Attributes are: "df" (**d**egrees of **f**reedom; number of estimated parameters in the model) and nobs (number of persons in the model)

See Also

Other functions to extract from a GPPM: SE(), coef.GPPM(), confint.GPPM(), covFun(), fitted.GPPM(), getData(), getIntern(), maxNObs(), meanFun(), nObs(), nPars(), nPers(), nPreds(), parEsts(), pars(), preds(), vcov.GPPM()

Examples

```
data("demoLGCM")
lgcm <- gppm(
  "muI+muS*t", "varI+covIS*(t+t#)+varS*t*t#+(t==t#)*sigma",
  demoLGCM, "ID", "y"
)
lgcmFit <- fit(lgcm)
ll <- logLik(lgcmFit)
```

maxNObs *Maximum Number of Observations per Person*

Description

Extracts the maximum number of observations per person from a GPPM.

Usage

```
maxNObs(gpModel)
```

Arguments

gpModel	object of class GPPM.

Value

Maximum number of observations as a numeric.

See Also

Other functions to extract from a GPPM: SE(), coef.GPPM(), confint.GPPM(), covFun(), fitted.GPPM(), getData(), getIntern(), logLik.GPPM(), meanFun(), nObs(), nPars(), nPers(), nPreds(), parEsts(), pars(), preds(), vcov.GPPM()

Examples

```
data("demoLGCM")
lgcm <- gppm(
  "muI+muS*t", "varI+covIS*(t+t#)+varS*t*t#+(t==t#)*sigma",
  demoLGCM, "ID", "y"
)
maxNumberObs <- maxNObs(lgcm)
```

meanFun *Mean Function*

Description

Extracts the mean function from a GPPM.

Usage

```
meanFun(gpModel)
```

Arguments

gpModel	object of class GPPM.

Value

The mean function as a character string.

See Also

Other functions to extract from a GPPM: SE(), coef.GPPM(), confint.GPPM(), covFun(), fitted.GPPM(), getData(), getIntern(), logLik.GPPM(), maxNObs(), nObs(), nPars(), nPers(), nPreds(), parEsts(), pars(), preds(), vcov.GPPM()

Examples

```
data("demoLGCM")
lgcm <- gppm(
  "muI+muS*t", "varI+covIS*(t+t#)+varS*t*t#+(t==t#)*sigma",
  demoLGCM, "ID", "y"
)
myMean <- meanFun(lgcm)
```

---

nObs *Number of Observations*

Description

Extracts the number of observations for each person from a GPPM.

Usage

```
nObs(gpModel)
```

Arguments

gpModel	object of class GPPM.

Value

Number of observations for each person as a numeric vector. The corresponding IDs are in the IDs attribute.

See Also

Other functions to extract from a GPPM: SE(), coef.GPPM(), confint.GPPM(), covFun(), fitted.GPPM(), getData(), getIntern(), logLik.GPPM(), maxNObs(), meanFun(), nPars(), nPers(), nPreds(), parEsts(), pars(), preds(), vcov.GPPM()

Examples

```
data("demoLGCM")
lgcm <- gppm(
  "muI+muS*t", "varI+covIS*(t+t#)+varS*t*t#+(t==t#)*sigma",
  demoLGCM, "ID", "y"
)
numberObs <- nObs(lgcm)
```

---

nPars *Number of Parameters*

Description

Extracts the number of parameters from a GPPM.

Usage

```
nPars(gpModel)
```

Arguments

gpModel	object of class GPPM.

Value

Number of parameters as a numeric.

See Also

Other functions to extract from a GPPM: SE(), coef.GPPM(), confint.GPPM(), covFun(), fitted.GPPM(), getData(), getIntern(), logLik.GPPM(), maxNObs(), meanFun(), nObs(), nPers(), nPreds(), parEsts(), pars(), preds(), vcov.GPPM()

Examples

```
data("demoLGCM")
lgcm <- gppm(
  "muI+muS*t", "varI+covIS*(t+t#)+varS*t*t#+(t==t#)*sigma",
  demoLGCM, "ID", "y"
)
numberParas <- nPars(lgcm)
```

nPers *Number of persons*

Description

Extracts the number of persons from a GPPM.

Usage

```
nPers(gpModel)
```

Arguments

gpModel	object of class GPPM.

Value

Number of persons as a numeric.

See Also

Other functions to extract from a GPPM: SE(), coef.GPPM(), confint.GPPM(), covFun(), fitted.GPPM(), getData(), getIntern(), logLik.GPPM(), maxNObs(), meanFun(), nObs(), nPars(), nPreds(), parEsts(), pars(), preds(), vcov.GPPM()

Examples

```
data("demoLGCM")
lgcm <- gppm(
  "muI+muS*t", "varI+covIS*(t+t#)+varS*t*t#+(t==t#)*sigma",
  demoLGCM, "ID", "y"
)
numberPersons <- nPers(lgcm)
```

nPreds *Number of Predictors*

Description

Extracts the number of predictors from a GPPM.

Usage

```
nPreds(gpModel)
```

Arguments

gpModel	object of class GPPM.

Value

Number of predictors as numeric.

See Also

Other functions to extract from a GPPM: SE(), coef.GPPM(), confint.GPPM(), covFun(), fitted.GPPM(), getData(), getIntern(), logLik.GPPM(), maxNObs(), meanFun(), nObs(), nPars(), nPers(), parEsts(), pars(), preds(), vcov.GPPM()

Examples

```
data("demoLGCM")
lgcm <- gppm(
  "muI+muS*t", "varI+covIS*(t+t#)+varS*t*t#+(t==t#)*sigma",
  demoLGCM, "ID", "y"
)
numberPreds <- nPreds(lgcm)
```

---

parEsts *Essential Parameter Estimation Results*

Description

Extracts the essential parameter estimation results for a GPPM.

Usage

```
parEsts(object, level = 0.95)
```

Arguments

object	object of class GPPM. Must be fitted, that is, a result from fit.GPPM.
level	scalar from 0 to 1. The confidence level required.

Value

A data.frame containing the estimated parameters, standard errors, and the lower and upper bounds of the confidence intervals.

See Also

Other functions to extract from a GPPM: SE(), coef.GPPM(), confint.GPPM(), covFun(), fitted.GPPM(), getData(), getIntern(), logLik.GPPM(), maxNObs(), meanFun(), nObs(), nPars(), nPers(), nPreds(), pars(), preds(), vcov.GPPM()

Examples

```
data("demoLGCM")
lgcm <- gppm(
  "muI+muS*t", "varI+covIS*(t+t#)+varS*t*t#+(t==t#)*sigma",
  demoLGCM, "ID", "y"
)
lgcmFit <- fit(lgcm)
paramEssentials <- parEsts(lgcmFit)
```

---

pars *Parameter Names*

Description

Extracts the parameter names from a GPPM.

Usage

```
pars(gpModel)
```

Arguments

gpModel	object of class GPPM.

Value

The names of the parameters

See Also

Other functions to extract from a GPPM: SE(), coef.GPPM(), confint.GPPM(), covFun(), fitted.GPPM(), getData(), getIntern(), logLik.GPPM(), maxNObs(), meanFun(), nObs(), nPars(), nPers(), nPreds(), parEsts(), preds(), vcov.GPPM()

Examples

```
data("demoLGCM")
lgcm <- gppm(
  "muI+muS*t", "varI+covIS*(t+t#)+varS*t*t#+(t==t#)*sigma",
  demoLGCM, "ID", "y"
)
parameters <- pars(lgcm)
```

`plot.GPPMPred` *Plotting predictions*

Description

Plots person-specific predictions

Usage

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

Arguments

`x`	object of class `GPPMPred` as obtained by `predict.GPPM`
`plotId`	character string or integer. ID of the person for which the predictions should be plotted
`...`	additional arguments (currently not used).

Value

A plot visualizing the predictive distribution. The bold line describes the mean and the shaded area the 95\

Examples

```
data("demoLGCM")
# remove all measurements from person 1 and the first form person 2
predIdx <- c(which(demoLGCM$ID == 1), which(demoLGCM$ID == 2)[1])
fitDemoLGCM <- demoLGCM[setdiff(1:nrow(demoLGCM), predIdx), ]

lgcm <- gppm(
  "muI+muS*t", "varI+covIS*(t+t#)+varS*t*t#+(t==t#)*sigma",
  fitDemoLGCM, "ID", "y"
)
lgcm <- fit(lgcm)
predRes <- predict(lgcm, demoLGCM[predIdx, ])
plot(predRes, 1)
```

plot.LongData *Plot a Long Data Frame*

Description

This function is used to plot data from class 'LongData' as it is returned by `getData` `simulate.GPPM`.

Usage

```
## S3 method for class 'LongData'
plot(x, plotIds, by, ID, DV, ...)
```

Arguments

x	longitudinal data frame of class 'LongData'.
plotIds	vector of IDs for which the data should be printed. Can be left empty. Then 5 IDs are picked randomly.
by	label of the variable on the x-axis. Can be left empty.
ID	label of the ID column. Can be left empty.
DV	label of the variable on the y-axis. Can be left empty.
...	additional parameters (currently not used).

Value

a fitted Gaussian process panel model, which is an object of class 'GPPM'

Examples

```
data("demoLGCM")
plot(demoLGCM, plotIds = c(1, 2, 3))
plot(demoLGCM) # five random ids
```

predict.GPPM *GPPM predictions*

Description

Obtain person-specific predictions.

Usage

```
## S3 method for class 'GPPM'
predict(object, newData, ...)
```

Arguments

object	object of class GPPM. Must be fitted, that is, a result from fit.GPPM.
newData	a data frame with the same column names as the data frame used for generating gpModel with gppm. May only contain new data, that is, data that was not used for fitting.
...	additional arguments (currently not used).

Value

Predictions of the dependent variable for all rows in newData. Conditional predictions for all persons in newData that are also present in the data used for fitting gpModel; unconditional predictions for others persons. See examples for format.

Examples

```
data("demoLGCM")
# remove all measurements from person 1 and the first form person 2
predIdx <- c(which(demoLGCM$ID == 1), which(demoLGCM$ID == 2)[1])
fitDemoLGCM <- demoLGCM[setdiff(1:nrow(demoLGCM), predIdx), ]

lgcm <- gppm(
  "muI+muS*t", "varI+covIS*(t+t#)+varS*t*t#+(t==t#)*sigma",
  fitDemoLGCM, "ID", "y"
)
lgcm <- fit(lgcm)
predRes <- predict(lgcm, demoLGCM[predIdx, ])
```

---

preds *Predictors Names*

Description

Extracts the predictor names from a GPPM.

Usage

```
preds(gpModel)
```

Arguments

gpModel	object of class GPPM.

Value

The names of the predictors.

See Also

Other functions to extract from a GPPM: SE(), coef.GPPM(), confint.GPPM(), covFun(), fitted.GPPM(), getData(), getIntern(), logLik.GPPM(), maxNObs(), meanFun(), nObs(), nPars(), nPers(), nPreds(), parEsts(), pars(), vcov.GPPM()

Examples

```
data("demoLGCM")
lgcm <- gppm(
  "muI+muS*t", "varI+covIS*(t+t#)+varS*t*t#+(t==t#)*sigma",
  demoLGCM, "ID", "y"
)
myPreds <- preds(lgcm)
```

SE *Standard Errors*

Description

Returns the standard errors of the parameters of a fitted GPPM.

Usage

```
SE(object)
```

Arguments

object	object of class GPPM. Must be fitted, that is, a result from fit.GPPM.

Value

Standard errors for all parameters as a named numeric vector.

See Also

Other functions to extract from a GPPM: coef.GPPM(), confint.GPPM(), covFun(), fitted.GPPM(), getData(), getIntern(), logLik.GPPM(), maxNObs(), meanFun(), nObs(), nPars(), nPers(), nPreds(), parEsts(), pars(), preds(), vcov.GPPM()

Examples

```
lgcm <- gppm(
  "muI+muS*t", "varI+covIS*(t+t#)+varS*t*t#+(t==t#)*sigma",
  demoLGCM, "ID", "y"
)
lgcmFit <- fit(lgcm)
stdErrors <- SE(lgcmFit)
```

`simulate.GPPM` *Simulate from a Gaussian process panel model*

Description

This function is used to simulate from a Gaussian process panel model, which has been specified using `gppm`.

Usage

```
## S3 method for class 'GPPM'
simulate(
  object,
  nsim = 1,
  seed = NULL,
  parameterValues = NULL,
  verbose = FALSE,
  ...
)
```

Arguments

`object`	object of class GPPM. The Gaussian process panel model from which to simulate.
`nsim`	integer. Number of data sets to generate.
`seed`	numeric. Random seed to be used.
`parameterValues`	numeric vector. Used to specify the values for the parameters. Which value belongs to which parameter is determined by the names attribute of parameterValues. See also the example.
`verbose`	boolean. Print diagnostic output?
`...`	additional parameters (currently not used).

Value

A simulated data set, which is an object of class 'LongData'. If `nsim>1` a list of `nsim` simulated data sets.

Examples

```
data("demoLGCM")
lgcm <- gppm(
  "muI+muS*t", "varI+covIS*(t+t#)+varS*t*t#+(t==t#)*sigma",
  demoLGCM, "ID", "y"
)

parameterValues <- c(10, -1, 0, 10, 0, 0.1)
```

```
names(parameterValues) <- c("muI", "muS", "varI", "varS", "covIS", "sigma")
simData <- simulate(lgcm, parameterValues = parameterValues)
```

---

summary.GPPM *Summarizing GPPM*

Description

This function is used to summarize a GPPM. summary method for class 'GPPM'.

Usage

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

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

Arguments

object	object of class GPPM.
...	additional parameters (currently not used).
x	output of fit.GPPM

Value

An object of class "summary.GPPM", which is a list with 4 entries:

- modelSpecification an object of class 'ModelSpecification' describing the model as a list with the following entries
  - meanFormula formula for the mean function; output of meanFun
  - covFormula formula for the covariance function; output of covFun
  - nPars number of parameters; output of nPars
  - params parameter names; output of pars
  - nPreds number of predictors; output of nPreds
  - preds predictors names; output of preds
- parameterEstimates a data frame containing a summary of the parameter estimates; output of parEsts
- modelfit An object of class "ModelFit" describing the modelfit using a list with the following entries
  - AIC AIC of the model; output of AIC
  - BIC BIC of the model; output of BIC
  - logLik log-likelihood of the model; output of logLik

- `dataStats` An object of class "DataStats" describing the data set using a list with the following entries
  - `nPer` number of persons; output of `nPers`
  - `maxTime` maximum number of observations per person; output of `maxNObs`
  - `nTime` number of observations for each person; output of `nObs`

Methods (by generic)

- `print(summary.GPPM)`: Printing a summary.GPPM object

---

trueParas *Parameters used for generating* demoLGCM.

Description

Parameters used for generating `demoLGCM`.

Usage

```
trueParas
```

Format

A parameter vector.

---

vcov.GPPM *Variance-Covariance Matrix*

Description

Returns the variance-covariance matrix of the parameters of a fitted GPPM.

Usage

```
## S3 method for class 'GPPM'
vcov(object, ...)
```

Arguments

`object`	object of class GPPM. Must be fitted, that is, a result from `fit.GPPM`.
`...`	additional arguments (currently not used).

Value

A matrix of the estimated covariances between the parameter estimates. This has row and column names corresponding to the parameter names.

See Also

Other functions to extract from a GPPM: SE(), coef.GPPM(), confint.GPPM(), covFun(), fitted.GPPM(), getData(), getIntern(), logLik.GPPM(), maxNObs(), meanFun(), nObs(), nPars(), nPers(), nPreds(), parEsts(), pars(), preds()

Examples

```
data("demoLGCM")
lgcm <- gppm(
  "muI+muS*t", "varI+covIS*(t+t#)+varS*t*t#+(t==t#)*sigma",
  demoLGCM, "ID", "y"
)
lgcmFit <- fit(lgcm)
covMat <- vcov(lgcmFit)
```

Index