Package 'lmms'

September 11, 2014

Type Package

Title Linear mixed effect model splines for modelling and analysis of time course data

Version 1.1

Date 2014-05-06

Depends R (>= 3.0.0)

Imports stats, methods, nlme, lmeSplines, parallel, reshape, gdata,gplots, ggplot2, graphics

Author Jasmin Straube, Kim-Anh Le Cao and Emma Huang with contributions of Dominique Gorse

Maintainer Jasmin Straube <j.straube@qfab.org>

Description The package implements linear mixed effect model splines for modelling and differential expression for highly dimensional data sets: investNoise for quality control and filterNoise for removing non-informative trajectories; lmmSpline to model time course expression profiles and lmmsDE performs differential expression analysis to identify differential expression between groups, time and/or time x group interaction.

License GPL (>= 2)

NeedsCompilation no

Repository CRAN

Date/Publication 2014-09-11 08:52:15

R topics documented:

lmms-package 2
deriv.lmmspline 3
filterNoise 4
investNoise 5
kidneySimTimeGroup 7
lmms-class 8

lmmsDE 8
lmmsde-class 11
lmmSpline 11
lmmspline-class 14
noise-class 15
plot.lmmsde 15
plot.lmmspline 16
plot.noise 17
predict.lmmspline 18
summary.lmmsde 19
summary.lmmspline 19
summary.noise 20

Index **22**

`lmms-package` *Data-driven mixed effect model splines fit and differential expression analysis*

Description

The package provides quality control and filtering methods as well as linear mixed effect model splines techniques for modelling and differential expression analysis to model and mine highly dimensional data sets: investNoise to identify noisy profiles and filterNoise to remove them; lmmSpline to model heterogeneous time course expression profiles; lmmsDE to perform differential expression analysis of time course data to identify differential expression over time, between groups or time and group interaction.

Details

Package:	lmms
Type:	Package
Version:	1.14
Date:	2014-05-08
License:	GPL-2
LazyLoad:	yes

Functions for quality control and filtering: `investNoise`, `filterNoise`,`summary.noise`,`plot.noise`
Functions for data modeling: `lmmSpline`, `lmmsDE`,`deriv.lmmspline`,`predict.lmmspline`
Functions for summarization: `summary.lmmspline`, `summary.lmmsde`
Functions for plots: `plot.lmmspline`, `plot.lmmsde`

Author(s)

Jasmin Straube with contributions from Kim-Anh Le Cao, Emma Huang and Dominique Gorse

Maintainer: Jasmin Straube <j.straube@qfab.org>

`deriv.lmmspline` *Derivative information for* `lmmspline` *objects*

Description

Calculates the derivative information for `lmmspline` objects with a `"p-spline"` or `"cubic p-spline"` basis.

Usage

```
## S3 method for class 'lmmspline'
deriv(expr, numCores, ...)
```

Arguments

`expr`	An object of class `lmmspline`.
`numCores`	alternative `numeric` value indicating the number of CPU cores to be used for parallelization. Default value is automatically estimated.
`...`	Additional arguments which are passed to `deriv`.

Value

deriv returns an object of class `lmmspline` containing the following components:

`predSpline`	`data.frame` containing the predicted derivative values based on the linear model object or the linear mixed effect model object.
`modelsUsed`	`numeric` vector indicating the model used to fit the data. 0 = linear model, 1 = linear mixed effect model spline (LMMS) with defined basis ("cubic" by default), 2 = LMMS taking subject-specific random intercept, 3 = LMMS with subject specific intercept and slope.
`model`	`list` of models used to model time profiles.
`derivative`	`logical` value indicating if the predicted values are the derivative information.

Examples

```
## Not run:
data(kidneySimTimeGroup)
# run lmmSpline on the samples from group 1 only
G1 <- which(kidneySimTimeGroup$group=="G1")
testLMMSplineTG<- lmmSpline(data=kidneySimTimeGroup$data[G1,],
                  time=kidneySimTimeGroup$time[G1],
                  sampleID=kidneySimTimeGroup$sampleID[G1],
                  basis="p-spline")
testLMMSplineTGDeri <- deriv(testLMMSplineTG)
summary(testLMMSplineTGDeri)
## End(Not run)
```

`filterNoise` *Filter non-informative trajectories*

Description

Function to remove non-informative trajectories

Usage

```
filterNoise(data, noise, RTCutoff, RICutoff, propMissingCutoff, fcCutoff)


  ## S4 method for signature
## 'matrixOrframe,
##   noise,
##   missingOrnumeric,
##   missingOrnumeric,
##   missingOrnumeric,
##   missingOrnumeric'
filterNoise(data,
  noise, RTCutoff, RICutoff, propMissingCutoff, fcCutoff)
```

Arguments

`data`	`data.frame` or `matrix` containing the samples as rows and features as columns.
`noise`	an object of class `noise` containing time and individual to molecule sd ratios number of missing values and maximum fold changes.
`RTCutoff`	`numeric` the R_T cutoff to remove non-informative trajectories.
`RICutoff`	`numeric` the R_I to remove non-informative trajectories.
`propMissingCutoff`	`numeric` maximum proportion of missing values in trajectories.
`fcCutoff`	`numeric` the minimum fold change observed between the mean of any two time points.

Details

filterNoise removes noisy or non-informative profiles based on selected theresholds R_I, R_T (Straube *et al.* 2014), maximum foldchanges and/or missing values.

Value

filterNoise returns an object of class `list` containing the following components:

`data`	`numeric` filtered data.
`removedIndices`	`numeric` removed indices

References

Straube J., Gorse D., Huang B.E., Le Cao K.-A.(2014). *A linear mixed model spline framework for analyzing time course 'omics' data* Submitted

See Also

investNoise

Examples

```
## Not run:
data(kidneySimTimeGroup)
G1 <- kidneySimTimeGroup$group=="G1"
noiseTest <-investNoise(data=kidneySimTimeGroup$data[G1,],time=kidneySimTimeGroup$time[G1],
            sampleID=kidneySimTimeGroup$sampleID[G1])
data <-filterNoise(data=kidneySimTimeGroup$data[G1,],noise=noiseTest,RTCutoff=0.9,
              RICutoff=0.3,propMissingCutoff=0.5)$data


#Alternatively model-based clustering can be used for filtering
library(mclust)
clusterFilter <- Mclust(cbind(noiseTest@RT,noiseTest@RI),G=2)
plot(clusterFilter,what = "classification")
meanRTCluster <-tapply(noiseTest@RT,clusterFilter$classification,mean)
bestCluster <- names(meanRTCluster[which.min(meanRTCluster)])
filterdata <- kidneySimTimeGroup$data[G1,clusterFilter$classification==bestCluster]


## End(Not run)
```

`investNoise` *Quality control for time course profiles*

Description

Function to calculate filter ratios of trajectories.

Usage

```
investNoise(data, time, sampleID, log, numCores)


  ## S4 method for signature
## 'matrixOrframe,
##   numeric,
##   factorOrcharacterOrnumeric,
##   missingOrlogical,
##   missingOrnumeric'
investNoise(data,
  time, sampleID, log, numCores)
```

Arguments

data	data.frame or matrix containing the samples as rows and features as columns
time	numeric vector containing the sample time point information.
sampleID	character, numeric or factor vector containing information about the unique identity of each sample
log	logical indicating log transformation of the data. Default value is TRUE
numCores	alternative numeric value indicating the number of CPU cores to be used for parallelization. Default value is automatically estimated.

Details

investNoise calculates filter ratios R_T and R_I based on the time, individual and overall standard deviation as proposed by Straube *et al.* 2014.

Value

investNoise returns an object of class noise containing the following components:

name	character the colnames or the index.
RT	numeric the time to molecule sd ratio of each trajectory.
RI	numeric the individual to molecule sd ratio of each trajectory.
propMissing	numeric Proportion of missing values for each trajectory.
fc	numeric the maximum absolute fold change (either for log transformed data max(time)-min(time) or not log transformed data max(time)/min(time)) observed between the mean of any two time points.

References

Straube J., Gorse D., Huang B.E., Le Cao K.-A.(2014). *A linear mixed model spline framework for analyzing time course 'omics' data* Submitted

See Also

summary.noise, plot.noise, filterNoise

Examples

```
## Not run:
data(kidneySimTimeGroup)
G1 <- kidneySimTimeGroup$group=="G1"
noiseTest <-investNoise(data=kidneySimTimeGroup$data[G1,],time=kidneySimTimeGroup$time[G1],
            sampleID=kidneySimTimeGroup$sampleID[G1])
summary(noiseTest)
plot(noiseTest,colorBy="propMissing")
## End(Not run)
```

kidneySimTimeGroup *Kidney Simulation Data*

Description

This data set contains the simulated expression of 140 proteins in 40 samples from either group 1 or group 2 measured on the time points 0, 0.5, 1, 2, 3, 4.

Usage

```
data(kidneySimTimeGroup)
```

Format

A list containing the following components:

`data` data matrix with 192 rows and 140 columns. Each row represents an experimental sample, and each column a single protein.

`time` a numeric vector containing the time points on which each sample is measured

`sampleID` a character vector containing the biological replicate information of each sample

`group` a character vector indicating the group of each sample

Details

This simulated data set consists of 40 samples and 140 proteins and was based on an the existing study from Freue *et al.* (2010). Samples were measured on maximum 6 time points: 0, 0.5, 1, 2, 3, 4. Some samples have missing time points. 50 molecules were randomly selected to introduce a fold change of log(2).

Source

The Kidney Simulation Data is based on the the paper of Freue *et al.* (2010).

References

Freue, G. V. C. et al. (2010). Proteomic signatures in plasma during early acute renal allograft rejection. *Molecular & cellular proteomics*, **9**, 1954-67.

lmms-class — lmms *class a S4 superclass to extend* lmmspline *and* lmmsde *class.*

Description

lmms class is a superclass for classes lmmspline and lmmsde. These classes inherit common slots.

Slots

basis An object of class character describing the basis used for modelling.

knots An object of class numeric, describing the boundaries of the splines.

errorMolecules Vector of class character, describing the molecules that could not be modelled.

lmmsDE — *Differential expression analysis using linear mixed effect model splines.*

Description

Function to fit a linear mixed effect model splines to perform differential expression analysis. The lmmsDE function fits LMM models with either a cubic, p-spline or cubic p-spline basis and compares the models to the null models. The type of basis to use is specified with the basis argument.

Usage

```
lmmsDE(data, time, sampleID, group, type,
experiment, basis, knots, numCores)


  ## S4 method for signature
## 'matrixOrframe,
##   numeric,
##   factorOrcharacterOrnumeric,
##   factorOrcharacterOrnumeric,
##   missingOrcharacter,
##   missingOrcharacter,
##   missingOrcharacter,
##   missingOrnumeric,
##   missingOrnumeric'
lmmsDE(data,
  time, sampleID, group, type, experiment, basis, knots, numCores)
```

Arguments

`data`	`data.frame` or `matrix` containing the samples as rows and features as columns
`time`	`numeric` vector containing the sample time point information.
`sampleID`	`character`, `numeric` or `factor` vector containing information about the unique identity of each sample
`group`	`character`, `numeric` or `factor` vector containing information about the group (or class) of each sample
`type`	`character` indicating what type of analysis is to be performed. Options are `"time"` to identify differential expression over time, `"group"` to identify profiles with different baseline levels (intercepts), and `"time*group"` an interaction between these two . Use `"all"` to calculate all three types.
`experiment`	`character` describing the experiment performed. `"timecourse"` for replicated experiments with less variation in individual expression values (e.g. model organism, cell culture), `"longitudinal1"` for different intercepts and `"longitudinal2"` for different intercepts and slopes.
`basis`	`character` string. What type of basis to use, matching one of `"cubic"` smoothing spline as defined by Verbyla *et al.* 1999, `"p-spline"` Durban *et al.* 2005 or a `"cubic p-spline"`.
`knots`	can take an integer value corresponding to the number of knots for the chosen basis or by default calculated as in Ruppert 2002. Not in use for the 'cubic' smoothing spline basis.
`numCores`	alternative `numeric` value indicating the number of CPU cores to be used for parallelization. Default value is automatically estimated.

Details

lmmsDE extends the LMMS modelling framework to permit tests between groups, across time, and for interactions between the two. Suppose we have R different groups of individuals, with h_i denoting the group for each individual i. Further we define h_{ir} to be the indicator for the r^{th} group, that is, $h_{ir} = 1$ if $h_i = r$ and 0 otherwise. The mean curve for each group f_{h_i} in the full LMMSDE `experiment="timecourse"` is given by:

$$f_{h_i}(t_{ij}) = \beta_0 + \beta_1 t_{ij} + \sum_{k=1}^{K} u_k (t_{ij} - \kappa_k)_+$$

$$+ \sum_{r=2}^{R} h_{ir}(\alpha_{0r} + \alpha_{1r} t_{ij}) + \sum_{r=2}^{R} h_{ir} \sum_{k=1}^{K} v_{rk}(t_{ij} - \kappa_k)_+.$$

Here $\boldsymbol{\alpha_0} = \alpha_{0r}$ are the differences between the intercepts between each group and the first group; $\boldsymbol{\alpha_1} = \alpha_{1r}$ are the differences in slope between each group and the first group; and v_{rk} are the differences in spline coefficients between each group and the first group. We can use this model to test the effects of time, group and interactions between the two. For a single group (`type="time"`), all $h_{ir} = 0$, and time effects will be detected if the null hypothesis $\beta_1 = 0$ is rejected. To detect differences between groups (`type="group"`), we set $\boldsymbol{\alpha_1} = 0$, and test the null hypotheses $\boldsymbol{\alpha_0} = \mathbf{0}$. Finally, including all parameters allows us to test for time * group interactions (`type="time*group"`), by permitting

different slopes in different groups and different intercepts. For `experiment="longitudinal1"` we include subject-specific random effects and assume them to be parallel to the mean curve. Finally, for `experiment="longitudinal2"` we assume subject-specific random effects to be straight lines and assuming independence between the random intercept and slope, so the covariance matrix for the random effects Σ is diagonal. In each case we compare the model fit of the expanded model with the respective null model using the function `anova`.

Value

lmmsDE returns an object of class `lmmsde` containing the following components:

`DE`	`data.frame` returning p-values and adjusted p-values using Benjamini-Hochberg correction for multiple testing of the differential expression testing over time, group or their interaction.
`modelTime`	a `list` of class `lme`, containing the models for every feature modelling the time effect.
`modelGroup`	a `list` of class `lme`, containing the models for every feature modelling group effect.
`modelTimeGroup`	a `list` of class `lme`, containing the models for every feature modelling time and group interaction effect.
`type`	an object of class `character`, describing the test performed either time, group, time*group or all.
`experiment`	an object of class `character` describing the model used to perform differential expression analysis.

References

Durban, M., Harezlak, J., Wand, M. P., & Carroll, R. J. (2005). *Simple fitting of subject-specific curves for longitudinal data.* Statistics in medicine, 24(8), 1153-67.

Ruppert D. (2002). *Selecting the number of knots for penalized splines.* Journal of Computational and Graphical Statistics 11, 735-757

Verbyla, A. P. Cullis, B. R., & Kenward, M. G. (1999). *The analysis of designed experiments and longitudinal data by using smoothing splines.* Appl.Statist.(1999), 18(3), 269-311.

See Also

`summary.lmmsde`, `plot.lmmsde`

Examples

```
## Not run:
data(kidneySimTimeGroup)
lmmsDEtest <-lmmsDE(data=kidneySimTimeGroup$data,time=kidneySimTimeGroup$time,
              sampleID=kidneySimTimeGroup$sampleID,group=kidneySimTimeGroup$group)
summary(lmmsDEtest)
## End(Not run)
```

lmmsde-class — lmmsde *class a S4 class that extends* lmms *class.*

Description

lmmsde class inherits from class lmms and extends it with six further slots: DE, model.time, model.group, model.time.group, type and experiment. The class lmmsde is returned when applying lmmsDE method.

Slots

DE A data.frame returning p-values and adjusted p-values using Benjamini-Hochberg correction for multiple testing of the differential expression testing over time, group or their interaction.

modelTime A list of classlme, containing the models for every molecule modelling the time effect.

modelGroup A list of class lme, containing the models for every molecule modelling group effect.

modelTimeGroup A list of class lme, containing the models for every molecule modelling time and group interaction effect.

type An object of class character, describing the test performed.

experiment An object of class character describing the model used to perform differential expression analysis.

lmmSpline — *Data-driven linear mixed effect model spline modelling*

Description

Function that models a linear or limear mixed model depending on the best fit. Alternatively, the function can return THE derivation information of the fitted models for the fixed (original) times points and a chosen basis.

Usage

```
lmmSpline(data, time, sampleID, timePredict, deri, basis, knots, numCores)


  ## S4 method for signature
## 'matrixOrFrame,
##   numeric,
##   factorOrcharacterOrnumeric,
##   missingOrnumeric,
##   missingOrlogical,
##   missingOrcharacter,
##   missingOrnumeric,
```

```
##   missingOrnumeric'
lmmSpline(data,
  time, sampleID, timePredict, deri, basis, knots, numCores)
```

Arguments

`data`	`data.frame` or `matrix` containing the samples as rows and features as columns
`time`	`numeric` vector containing the sample time point information.
`sampleID`	`ANY` vector containing information about the source of the sample
`timePredict`	`numeric` vector containing the time points to be predicted. By default set to the original time points observed in the experiment.
`deri`	`logical` value. If `TRUE` returns the predicted derivative information on the observed time points.By default set to `FALSE`.
`basis`	`character` string. What type of basis to use, matching one of `"cubic"`, `"p-spline"` or `"cubic p-spline"`. The `"cubic"` basis (`default`) is the cubic smoothing spline as defined by Verbyla *et al.* 1999, the `"p-spline"` is the truncated p-spline basis as defined by Durban *et al.* 2005.
`knots`	Alternatively an `integer`, the number of knots used for the `"p-spline"` or `"cubic p-spline"` basis calculation. Otherwise calculated as proposed by Ruppert 2002. Not used for the "cubic" smoothing spline basis.
`numCores`	Alternative `numeric` value indicating the number of CPU cores to be used. Default value is automatically estimated.

Details

The first model (`modelsUsed=0`) assumes the response is a straight line not affected by individual variation.

Let $y_{ij}(t_{ij})$ be the expression of a feature for individual (or biological replicate) i at time t_{ij}, where $i = 1, 2, ..., n$, $j = 1, 2, ..., m_i$, n is the sample size and m_i is the number of observations for individual i for the given feature. We fit a simple linear regression of expression $y_{ij}(t_{ij})$ on time t_{ij}. The intercept β_0 and slope β_1 are estimated via ordinary least squares: $y_{ij}(t_{ij}) = \beta_0 + \beta_1 t_{ij} + \epsilon_{ij}$, where $\epsilon_{ij} \tilde{} N(0, \sigma_\epsilon^2)$. The second model (`modelsUsed=1`) is nonlinear where the straight line in regression replaced with a curve modelled using here for example a spline truncated line basis (`basis="p-spline"`) as proposed Durban *et al.* 2005:

$$y_{ij}(t_{ij}) = f(t_{ij}) + \epsilon_{ij},$$

where $\epsilon_{ij} \tilde{} N(0, \sigma_\epsilon^2)$.

The penalized spline is represented by f, which depends on a set of knot positions $\kappa_1, ..., \kappa_K$ in the range of t_{ij}, some unknown coefficients u_k, an intercept β_0 and a slope β_1. The first term in the above equation can therefore be expanded as:

$$f(t_{ij}) = \beta_0 + \beta_1 t_{ij} + \sum_{k=1}^{K} u_k (t_{ij} - \kappa_k)_+,$$

with $(t_{ij} - \kappa_k)_+ = t_{ij} - \kappa_k$, if $t_{ij} - \kappa_k > 0, 0$ otherwise.

The choice of the number of knots K and their positions influences the flexibility of the curve. If the argument `knots=missing`, we use a method proposed by Ruppert 2002 to estimate the number of knots given the measured number of time points T, so that the knots $\kappa_1 \dots \kappa_K$ are placed at quantiles of the time interval of interest:

$$K = max(5, min(floor(\frac{T}{4}), 40)).$$

In order to account for individual variation, our third model (`modelsUsed=2`) adds a subject-specific random effect U_i to the mean response $f(t_{ij})$. Assuming $f(t_{ij})$ to be a fixed (yet unknown) population curve, U_i is treated as a random realization of an underlying Gaussian process with zero-mean and variance σ_U^2 and is independent from the random error ϵ_{ij}:

$$y_{ij}(t_{ij}) = f(t_{ij}) + U_i + \epsilon_{ij}$$

with $U_i \tilde{} N(0, \sigma_U^2)$ and $\epsilon_{ij} \tilde{} N(0, \sigma_\epsilon^2)$. In the equation above, the individual curves are expected to be parallel to the mean curve as we assume the individual expression curves to be constant over time. A simple extension to this model is to assume individual deviations are straight lines. The fourth model (`modelsUsed=3`) therefore fits individual-specific random intercepts a_{i0} and slopes a_{i1}:

$$y_{ij}(t_{ij}) = f(t_{ij}) + a_{i0} + a_{i1} t_{ij} + \epsilon_{ij}$$

with $\epsilon_{ij} \tilde{} N(0, \sigma_\epsilon^2)$ and $(a_{i0}, a_{i1})^T \sim N(0, \Sigma)$. We assume independence between the random intercept and slope.

Value

lmmSpline returns an object of class `lmmspline` containing the following components:

`predSpline`	`data.frame` containing predicted values based on linear model object or linear mixed effect model object.
`modelsUsed`	`numeric` vector indicating the model used to fit the data. 0 = linear model, 1=linear mixed effect model spline (LMMS) with defined basis ('cubic' by default) 2 = LMMS taking subject-specific random intercept, 3 = LMMS with subject specific intercept and slope.
`model`	`list` of models used to model time profiles.
`derivative`	`logical` value indicating if the predicted values are the derivative information.

References

Durban, M., Harezlak, J., Wand, M. P., & Carroll, R. J. (2005). *Simple fitting of subject-specific curves for longitudinal data.* Statistics in medicine, 24(8), 1153-67.

Ruppert D. (2002). *Selecting the number of knots for penalized splines.* Journal of Computational and Graphical Statistics 11, 735-757

Verbyla, A. P. Cullis, B. R., & Kenward, M. G. (1999). *The analysis of designed experiments and longitudinal data by using smoothing splines.* Appl.Statist.(1999), 18(3), 269-311.

See Also

summary.lmmspline, plot.lmmspline, predict.lmmspline, deriv.lmmspline

Examples

```
## Not run:
data(kidneySimTimeGroup)
# running for samples in group 1
G1 <- which(kidneySimTimeGroup$group=="G1")
testLMMSpline<- lmmSpline(data=kidneySimTimeGroup$data[G1,],time=kidneySimTimeGroup$time[G1],
                 sampleID=kidneySimTimeGroup$sampleID[G1])
summary(testLMMSpline)
DerivTestLMMSplineTG<- lmmSpline(data=as.data.frame(kidneySimTimeGroup$data[G1,]),
                  time=kidneySimTimeGroup$time[G1],sampleID=kidneySimTimeGroup$sampleID[G1],
                       deri=TRUE,basis="p-spline")
summary(DerivTestLMMSplineTG)
## End(Not run)
```

`lmmspline-class` `lmmspline` *class a S4 class that extends* `lmms` *class.*

Description

`lmmspline` class inherits from class `lmms` and extends it with three further slots: `predSpline`, `modelsUsed`, `models`. The class is returned when applying `lmmSpline` method.

Slots

`predSpline` A `data.frame` returning the fitted values for the time points of interest.

`modelsUsed` A vector of class `numeric`, describing the polynomial boundaries of the splines. If not defined or if basis='cubic' knots are automatically estimated using Ruppert 2002 or are the design points when using 'cubic'.

`models` A `list` of class `lm` or `lme`, containing the models used to model the particular feature of interest.

`data` A `data.frame` containing the data used to model the particular molecule of interest.

`derivative` A `logical` value indicating if the derivative was calculated.

noise-class noise *S4 class*

Description

The class noise is returned when applying investNoise method.

Slots

name character vector. The name of the trajectory.

RT A numeric vector, containing the time to molecule standard deviation ratios for every trajectory.

RI A numeric vector, containing the individual to molecule standard deviation ratios for every trajectory.

propMissing A numeric vector, containing the proportion of missing values for every trajectory.

foldChange A numeric vector, containing the maximum fold change of the mean between any two time points.

plot.lmmsde *Plot of* lmmsde *objects*

Description

Plot of the raw data the mean and the fitted lmmsde profile.

Usage

```
## S3 method for class 'lmmsde'
plot(x, y, type, smooth, ...)
```

Arguments

x	An object of class lmmsde.
y	numeric or character value. Either the row index or the row name determining which feature should be plotted.
type	a character indicating what model to plot. Default 'all', options: 'time', 'group','group*time'.
smooth	an optional logical value.By default set to FALSE. If TRUE smooth representation of the fitted values.
...	Additional arguments which are passed to plot.

Value

plot showing raw data, mean profile and fitted profile.

Examples

```
## Not run:
data(kidneySimTimeGroup)
lmmsDEtestl1 <-lmmsDE(data=kidneySimTimeGroup$data,time=kidneySimTimeGroup$time,
                sampleID=kidneySimTimeGroup$sampleID,
                group=kidneySimTimeGroup$group,
                experiment="longitudinal1",basis="p-spline")
plot(lmmsDEtestl1,y=2,type="all")
plot(lmmsDEtestl1,y=2,type="time")
plot(lmmsDEtestl1,y=2,type="group")
plot(lmmsDEtestl1,y=2,type="group*time",smooth=TRUE)
## End(Not run)
```

---

plot.lmmspline *Plot of* lmmspline *object*

Description

Plots the raw data, the mean and the fitted or derivative information of the lmmspline object.

Usage

```
## S3 method for class 'lmmspline'
plot(x, y, smooth, ...)
```

Arguments

x	An object of class lmmspline.
y	character or numeric value. Determining which feature should be plotted can be either the index or the name of the feature.
smooth	an optional logical value. Default FALSE, if TRUE smooth representation of the fitted values.
...	Additional arguments which are passed to plot.

Value

xyplot showing raw data, mean profile and fitted profile.

Examples

```
## Not run:
data(kidneySimTimeGroup)
# running for samples in group 1
G1 <- which(kidneySimTimeGroup$group=="G1")
testLmmspline <- lmmSpline(data=kidneySimTimeGroup$data[G1,],
                time=kidneySimTimeGroup$time[G1],
                sampleID=kidneySimTimeGroup$sampleID[G1])
plot(testLmmspline, y=2)
```

```
plot(testLmmspline, y=2, smooth=TRUE)
## End(Not run)
```

`plot.noise` *Plot of* `noise` *objects*

Description

Plot of the filter ratios R_T and R_I as proposed by Straube et al 2014.

Usage

```
## S3 method for class 'noise'
plot(x, colorBy = "propMissing", fcCutoff = NA,
  propMissingCutoff = NA, ...)
```

Arguments

`x`	an object of class `noise`.
`colorBy`	`character` the variable to be colored by. Default `'propMissing'`, options: `'propMissing','fc'`.
`fcCutoff`	an optional `numeric` value to remove ratios with low fold changes.
`propMissingCutoff`	an optional `numeric` value to remove ratios with high number of missing values.
`...`	ignored

Value

plot showing filter ratios R_T and R_I as proposed by Straube *et al.* 2014. Filter ratios can either be colored by proportion of missing values or maximum fold change.

References

Straube J., Gorse D., Huang B.E., Le Cao K.-A.(2014). *A linear mixed model spline framework for analyzing time course 'omics' data* Submitted

See Also

`investNoise`, `filterNoise`

Examples

```
## Not run:
data(kidneySimTimeGroup)
G1 <- kidneySimTimeGroup$group=="G1"
noiseTest <-investNoise(data=kidneySimTimeGroup$data[G1,],time=kidneySimTimeGroup$time[G1],
            sampleID=kidneySimTimeGroup$sampleID[G1])
plot(noiseTest,colorBy="fc")

## End(Not run)
```

`predict.lmmspline` *Predicts fitted values of an* `lmmspline` *Object*

Description

Predicts the fitted values of an `lmmspline` object for time points of interest.

Usage

```
## S3 method for class 'lmmspline'
predict(object, timePredict, numCores, ...)
```

Arguments

`object`	an object inheriting from class `lmmspline`.
`timePredict`	an optional `numeric` vector. Vector of time points to predict fitted values. If `missing` uses design points.
`numCores`	alternative `numeric` value indicating the number of CPU cores to be used for parallelization. By default estimated automatically.
`...`	ignored.

Value

`matrix` containing predicted values for the requested time points from argument `timePredict`.

Examples

```
## Not run:
data(kidneySimTimeGroup)
G1 <- which(kidneySimTimeGroup$group=="G1")
testLMMSpline<- lmmSpline(data=kidneySimTimeGroup$data[G1,],
                time=kidneySimTimeGroup$time[G1],
                sampleID=kidneySimTimeGroup$sampleID[G1])
mat.predict <- predict(testLMMSpline, timePredict=c(seq(1,4, by=0.5)))
## End(Not run)
```

summary.lmmsde *Summary of a* lmmsde *Object*

Description

Summarizes the lmmsde object returned by the lmmsDE method. Including the models fitted, parameter used and the number of features declared as differentially expressed.

Usage

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

Arguments

object	An object of class lmmsde .
...	Additional arguments which are passed to summary.

Value

summary of the lmmsde object.

Examples

```
## Not run:
data(kidneySimTimeGroup)
lmmsDEtest <-lmmsDE(data=kidneySimTimeGroup$data,time=kidneySimTimeGroup$time,
             sampleID=kidneySimTimeGroup$sampleID,group=kidneySimTimeGroup$group)
summary(lmmsDEtest)
## End(Not run)
```

summary.lmmspline *Summary of a* lmmspline *Object*

Description

Summarizes the lmmspline object returned by the lmmSpline method. Including the models fitted and parameter used.

Usage

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

Arguments

object	An object of class lmmspline.
...	Additional arguments which are passed to summary.

Value

Summary of the lmmspline object.

Examples

```
## Not run:
data(kidneySimTimeGroup)
# running for samples in group 1
G1 <- which(kidneySimTimeGroup$group=="G1")
testLMMSplineTG<- lmmSpline(data=kidneySimTimeGroup$data[G1,],
                  time=kidneySimTimeGroup$time[G1],
                  sampleID=kidneySimTimeGroup$sampleID[G1])
summary(testLMMSplineTG)
## End(Not run)
```

---

summary.noise *Summary of a* noise *Object*

Description

Summarizes the noise object returned by the investNoise method.

Usage

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

Arguments

object	An object of class noise.
...	ignored

Value

Summary of the noise object.

Examples

```
## Not run:
data(kidneySimTimeGroup)
# running for samples in group 1
G1 <- which(kidneySimTimeGroup$group=="G1")
noiseTest<- investNoise(data=kidneySimTimeGroup$data[G1,],
                  time=kidneySimTimeGroup$time[G1],
                  sampleID=kidneySimTimeGroup$sampleID[G1])
summary(noiseTest)
## End(Not run)
```

Index

∗Topic **datasets**
kidneySimTimeGroup, 7
∗Topic **package**
lmms-package, 2

anova, *10*

deriv.lmmspline, *2*, 3, *14*

factorOrcharacterOrnumeric,missingOrcharacter,missingOrcharacter,missingOrcharacter,
*(*lmmsDE*)*, 8
filterNoise, *2*, 4, *6*, *17*
filterNoise,matrixOrframe,noise,missingOrnumeric,missingOrnumeric,missingOrnumeric,missingOrnumeric-met
*(*filterNoise*)*, 4

investNoise, *2*, *5*, 5, *15*, *17*, *20*
investNoise,matrixOrframe,numeric,factorOrcharacterOrnumeric,missingOrlogical,missingOrnumeric-method
*(*investNoise*)*, 5
investNoise,matrixOrframe,numeric,factorOrcharacterOrnumeric,missingOrnumeric-method
*(*investNoise*)*, 5

kidneySimTimeGroup, 7

lme, *10*, *11*
lmms *(*lmms-package*)*, 2
lmms-class, 8
lmms-package, 2
lmmsDE, *2*, *8*, 8, *11*, *19*
lmmsDE,matrixOrframe,numeric,factorOrcharacterOrnumeric,
*(*lmmsDE*)*, 8
lmmsDE,matrixOrframe,numeric,factorOrcharacterOrnumeric,factorOrcharacterOrnumeric,missingOrcharacter,m
*(*lmmsDE*)*, 8
lmmsde-class, 11
lmmSpline, *2*, 11, *14*, *19*
lmmSpline,matrixOrFrame,numeric,factorOrcharacterOrnumeric,
*(*lmmSpline*)*, 11
lmmSpline,matrixOrFrame,numeric,factorOrcharacterOrnumeric,missingOrnumeric,missingOrlogical,missingOrc
*(*lmmSpline*)*, 11
lmmspline-class, 14

missingOrlogical,missingOrcharacter,missingOrnumeric,missingOrnumeric-method
*(*lmmSpline*)*, 11
missingOrnumeric,missingOrnumeric-method
*(*lmmsDE*)*, 8

noise-class, 15

plot.lmmsde, *2*, *10*, 15
plot.lmmspline, *2*, *14*, 16
plot.noise, *2*, *6*, 17
predict.lmmspline, *2*, *14*, 18

summary.lmmsde, *2*, *10*, 19
summary.lmmspline, *2*, *14*, 19
summary.noise, *2*, *6*, *20*