Package 'plmDE'

July 2, 2014

Type Package

Title Additive partially linear models for differential gene expression analysis

Version 1.0

Date 2012-05-01

Author Jonas Mueller

Maintainer Jonas <jonasmueller303@hotmail.com>

Depends R (>= 2.14.2), MASS, splines, limma, R.oo

Suggests Biobase, edgeR, DESeq

Description A set of tools for identifying genes whose differential expression is associated with measurements of other covariates on a continuous scale. These methods rely on generalized additive partially linear models which can be fitted efficiently using a B-spline basis approximation. Still under development: methods for interfacing with objects extending the eSet class and a function to pass linear models in edgeR and DEseq format.

License GPL (>= 2)

Repository CRAN

Date/Publication 2012-05-09 04:06:58

NeedsCompilation no

R topics documented:

`plmDE-package` *Generalized Additive Partially Linear Models for Gene Expression Data*

Description

This package is intended for the analysis of gene expression data which is accompanied by some quantitative measurements (such as weight or tumor size) for each sample. It provides a very flexible framework for testing numerous differential-expression-related hypotheses regarding such data. To properly formulate such hypotheses, one must have a solid grasp of the models on which they are founded, and I therefore provide an introduction to this methodology, which should facilitate successful use of the package.

In a disease for which severity level (or any specific trait of interest) can be numerically expressed by some measure S, it is reasonable to suppose the measured expression level of a gene Y in a profiling experiment (where I_D indicates the presence of the disease) can be described by the following generalized additive partially linear model:

$$g(E[Y|I_D, S]) = \beta_0 + \beta_1 I_D + I_D f(S)$$

where g is some specified link function and f is a function (with an intercept = 0) which describes the effects of interaction between the disease and its severity level on the expression of the gene. Because of the complex nature of the interactions between genes and their environment, few assumptions are placed on f. Given a expression profiling dataset of this sort, if we identify differentially expressed genes as those for which $\beta_1 = f(S) = 0$, we presumably obtain a set of genes whose differential expression is more likely to be attributed to their effect on S in the course of the disease than the set of genes identified as differentially expressed through only testing $\beta_1 = 0$ in a simpler model where f is set to 0.

Generalizing this scenario, suppose we now have groups $D_1, \ldots D_G$ and baseline group N into which each sample can be classified, as well as numerous quantitative covariates $S_1, \ldots S_C$ which are measured from each sample. Then, Y_j, the expression level of gene j in a sample X can be modeled as:

$$g(E[Y_j \mid data\ on\ X]) = \beta_{N,j} + \sum_{i=1}^{G} \beta_{i,j} I_{D_i}(X) + \sum_{i=1}^{C} I_{D_i} f_{i,j}(S_{i,X})$$

From such a model, we can test a number of hypotheses. An example of one that might be of interest would be: For each gene j, simultaneously test whether $I_{D_r} f_{2,j} = I_{D_s} f_{2,j}$ and $\beta_{r,j} = \beta_{s,j}$. The

genes whose expression levels are rejected by this test would be candidate members of the set whose expression is involved in changes in S_2 between groups D_r and D_s.

To test such a model, we first express $f_{i,j}$ in terms of a linear combination of predefined basis functions. Then, we can fit a reduced model to the data in which we select one set of coefficients for these basis functions that best fits the expression level data of both groups at gene j, and we fit a full model which adds on top of the reduced model another subset of basis coefficients to better fit the expression levels from the second group. Since both the full and reduced model have been transformed into generalized linear models through the basis approximation, the significance of the additional coefficients in the full model over the reduced can easily be tested (using for example Chi-square or F tests).

This package contains methods to perform such tests, using B-splines as the basis functions, and methods for viewing the fit of the estimated functions on the expression data. All it requires from the user is the specification of the full and reduced models representing the test to be conducted on a dataset of gene expression measurements.

Details

Package:	plmDE
Type:	Package
Version:	1.0
Date:	2012-05-01
License:	GPL Version 2 or newer

Author(s)

Jonas Mueller

Maintainer: <jonasmueller303@hotmail.com>

References

Wang, L., Xiang, L., Liang, H., and Carroll, R. Estimation and variable selection for generalized additive partial linear models. *Annals of Statistics* **39**, 1827-51 (2011).

Examples

```
## create an object of type \code{plmDE} containing disease with
## "control" and "disease" and measures of weight and severity:
ExpressionData = as.data.frame(matrix(abs(rnorm(10000, 1, 1.5)), ncol = 100))
names(ExpressionData) = sapply(1:100, function(x) paste("Sample", x))
Genes = sapply(1:100, function(x) paste("Gene", x))
DataInfo = data.frame(sample = names(ExpressionData), group = c(rep("Control", 50),
rep("Diseased", 50)), weight = abs(rnorm(100, 50, 20)), severity = c(rep(0, 50),
abs(rnorm(50, 100, 20))))
plmDEobject = plmDEmodel(Genes, ExpressionData, DataInfo)
```

```
## test whether severity and the indicator variable
## for disease are simultaneously significant:
test = fitGAPLM(plmDEobject, continuousCovariates.fullModel
= c("weight", "severity"), compareToReducedModel = TRUE,
indicators.reducedModel = NULL, continuousCovariates.reducedModel = "weight")

## find genes with most evidence for differential expression under the model:
mostDE(test)

## plot the model's fit on the expression data of the 5th gene:
plot(test, "weight", 5, plmDEobject)
```

`fitBspline` *Compute B-Spline Basis for Data*

Description

Provides a heuristic framework for fitting a set of B-spline basis functions to a set of data points under the constraint of a limited number of degrees of freedom.

Usage

```
fitBspline(dataValues, continuousCovariates, indicators, group, covariate)
```

Arguments

`dataValues`	A vector of data points to which the spline is to be fit.
`continuousCovariates`	Vector that contains all the continuous variables whose coefficients must be estimated in this model.
`indicators`	This vector contains all the indicator variables whose coefficients will also be estimated in the model. The number of parameters in `indicators` and `continuousCovariates` together with the length of `dataValues` determine the available number of degrees of freedom for the fitting of the spline.
`group`	String denoting the subgroup from which the measurements of these `dataValues` are coming.
`covariate`	String denoting the variable that is being measured by the `dataValues`.

Details

This method is intended for users who do not wish to specify B-spline properties. It adopts the heuristic of modified equipotent arrangement put forth by Yanagihara and Ohtaki (1991) in selecting knot-placement of the splines.

Value

Uses the return format of `bs` in the `splines` package, which is a matrix corresponding to the values of each fitted B-spline basis function evaluated at each point in `dataValues`.

Author(s)

Jonas Mueller

References

Yanagihara, H. and Ohtaki, M. Knot-Placement to Avoid Over Fitting in *B*-Spline Scedastic Smoothing. *Communications in Statistics*, **32**, 771-85 (1991).

See Also

splines for a basic B-spline package on which fitBspline heavily relies.

Examples

```
data = runif(n = 30, min = 50, max = 100)
variables = c("height", "weight", "insulin")
indicators = c("sex", "diseased")
subgroup = "diabetes"
covariate = "height"
BsplineFit = fitBspline(data, variables, indicators, subgroup, covariate)
```

---

fitGAPLM — *Fit a Generalized Additive Partially Linear Model on Gene Expression Data*

Description

Given an plmDE object containing preprocessed/normalized measures of the expression of a set of genes under different conditions as well as related values of quantitatively-measured covariates of interest, fitGAPLM tests each gene for differential expression under a model specified by the user. The test is conducted based on the significance of a full Model fit to the expression data when compared with the fit of a reduced model (F statistic). The variables of interest should be present in the full model and absent in the reduced. This method is very flexible and can fit count data (eg. expression measures from high-throughput sequencing) as well as microarray data. Using fitGAPLM, the user can choose to model the gene expression measures by any mixture of additive functions of the numerical variables with linear terms of the factorial information available. Each of these functions is approximated through a B-spline fit with the intercept of the spline constrained at zero for identifiability. Although fitGAPLM seems to take in a daunting amount of input, many of the inputs already set to sensible defaults, and models of the complexity represented in this class must be well thought out and each parameter requires careful consideration.

Usage

```
fitGAPLM(dataObject, generalizedLM = FALSE, family = poisson(link = log),
 NegativeBinomialUnknownDispersion = FALSE, test = "LRT", weights = NULL,
offset = NULL, pValueAdjustment = "fdr", significanceLevel = 0.05,
indicators.fullModel = as.character(unique(dataObject$sampleInfo[,2])[-1]),
```

```
continuousCovariates.fullModel = NULL,
groups.fullModel = as.character(unique(dataObject$sampleInfo[,2])[-1]),
groupFunction.fullModel = rep("AdditiveSpline", length(groups.fullModel)),
fitSplineFromData.fullModel = TRUE,
splineDegrees.fullModel = rep(3, length(groups.fullModel)),
splineKnots.fullModel = rep(0, length(groups.reducedModel)),
compareToReducedModel = FALSE,
indicators.reducedModel = as.character(unique(dataObject$sampleInfo[,2])[-1]),
continuousCovariates.reducedModel = NULL,
groups.reducedModel = as.character(unique(dataObject$sampleInfo[,2])[-1]),
 groupFunction.reducedModel = rep("AdditiveSpline",
length(groups.reducedModel)), fitSplineFromData.reducedModel = TRUE,
 splineDegrees.reducedModel = rep(3, length(groups.reducedModel)),
splineKnots.reducedModel = rep(0, length(groups.reducedModel)),
splineKnotSpread = "quantile")
```

Arguments

dataObject	Object of type plmDE containing the gene expression and sample information.
generalizedLM	If TRUE, a link function is introduced to generalize the linear model. Use for gene-level count data.
family	One of the distribution families that may be used in the function glm. For gene-level count data, the negative binomial (see negative.binomial) is recommended to account for over dispersion.
NegativeBinomialUnknownDispersion	In the case of a negative binomial fit, has the dispersion of the data been estimated or does it remain unknown? If TRUE, then glm.nb from the MASS package is called, which includes routines for fitting the GLM and estimating the dispersion parameter.
test	The test that should be used in the case that a GLM is requested to estimate the significance of the model. See stat.anova for details.
weights	an optional vector of prior weights to be used in the fitting of the (generalized) linear model. Should be NULL or a numeric factor.
offset	an optional *a priori* known component to be included in the fitting of the (generalized) linear model. One or more offset terms may be included in the model.
pValueAdjustment	Choice of multiple testing correction method to be passed to p.adjust
significanceLevel	The significance level at which genes should be identified as differentially expressed.
indicators.fullModel	The indicator terms which should go into the full model. These must match the groups in the second column of the sample information in dataObject. Under the default setting, the indicators will consist of all groups except for the first one (used as the baseline for comparison).

continuousCovariates.fullModel
: The quantitative covariates that should go into the full model. These must match the column names of the sample information in dataObject.

groups.fullModel
: The subgroups of our sample for which we wish to estimate a function relating their measurement of continuousCovariates to their expression levels in dataObject.

groupFunction.fullModel
: A vector of the same length as groups.fullModel which contains consists of strings matching: "AdditiveSpline", "AdditiveLinear", "CommonSpline", or "CommonLinear". If AdditiveSpline is chosen, then a B-spline basis is fitted to the continuousCovariate values of the corresponding group in groups.fullModel to estimate a function that represents the effect of this group's continuousCovariate values on their measured expression levels. This function implicitly assumes an indicator term so it evaluates to 0 for the measurements of continuousCovariate from other groups, and its overall effects are assumed to be additive with respect to the other parameters being estimated. If "AdditiveLinear" is selected, then this function is taken to be the identity function (no spline basis fit) times a parameter to be fit by the model. To estimate one function to account for the same effect across multiple groups, they must all be listed in groups.fullModel and their corresponding index in goupFunction must be set to "CommonSpline". Likewise to assume a linear effect across multiple groups, they must also be listed in groups.fullModel and the corresponding indices of groupFunction must read "CommonLinear",

fitSplineFromData.fullModel
: Should the B-spline functions in the full model be automatically fitted based on the heuristic in fitBspline?

splineDegrees.fullModel
: If fitSplineFromData.fullModel has not been selected, then the user may specify, in a vector format, the degree of each B-spline basis that is fitted to the groups.

splineKnots.fullModel
: If fitSplineFromData.fullModel has not been selected, then the user may also specify, in a vector, the number of knots to include in each corresponding basis.

compareToReducedModel
: If TRUE, then the user must specify a model that the full model should be tested against. Otherwise, the all terms (besides intercept) of the full model are simultaneously tested for significance.

indicators.reducedModel
: See corresponding parameter for full model.

continuousCovariates.reducedModel
: See corresponding parameter for full model.

groups.reducedModel
: See corresponding parameter for full model.

groupFunction.reducedModel
: See corresponding parameter for full model.

`fitSplineFromData.reducedModel`
: See corresponding parameter for full model.

`splineDegrees.reducedModel`
: See corresponding parameter for full model.

`splineKnots.reducedModel`
: See corresponding parameter for full model.

`splineKnotSpread`
: Determines whether B-spline knots are uniformly spread over the range of the data or over the quartiles of the data (takes values: "uniform" or "quantile"), but does not affect the `fitBSpline` method.

Value

Returns an object of type `DEresults` containing various information about the analysis.

`allgenes`
: Data frame consisting of information on all the genes, their p-values and adjusted p-values, and whether or not this test identifies them as differentially expressed.

`DEgenes`
: Data frame consisting of the genes which were expressed at significantly differing levels according to this model.

`PredictorFormula.fullModel`
: contains the formula followed by the predictors in the B-spline-approximated linear model, but leaves the dependent variable term out.

`PredictorFormula.reducedModel`
: contains the formula followed by the predictors in the reduced model (leaving out the dependent variable term).

`modelForm.fullModel`
: contains the indicators and covariates incorporated into the full model.

`modelForm.reducedModel`
: contains the indicators and covariates incorporated into the reduced model.

`GLMinfo`
: tracks the glm parameters used in the fitting of this model (for plotting purposes).

Note

Because `fitGAPLM` is implemented in R rather than a compiled language, it tends to run slowly for larger expression assays (takes ~20 min to run an analysis on 65 samples of ~50,000 probes from the HG-U133 array). If the GAPLM is intended to be fit to Microarray data, the `limmaPLM` function should be used instead. However, `fitGAPLM` must be used if un-moderated F-test statistics or plots of the fitted functions are desired.

Author(s)

Jonas Mueller

References

Wang, L., Liu, X., Liang, H., and Carroll, R. J. Generalized Additive Partial Linear Models- Polynomial Spline Smoothing Estimation and Variable Selection Procedures. The Annals of Statistics **39**:4, 1827-1851 (2011)

See Also

limmaPLM for analysis of microarray data. fitBspline for default spline fitting heuristic.

Examples

```
## create an object of type \code{plmDE} containing disease with
## "control" and "disease" and measures of weight and severity:
ExpressionData = as.data.frame(matrix(abs(rnorm(10000, 1, 1.5)), ncol = 100))
names(ExpressionData) = sapply(1:100, function(x) paste("Sample", x))
Genes = sapply(1:100, function(x) paste("Gene", x))
DataInfo = data.frame(sample = names(ExpressionData), group = c(rep("Control", 50),
 rep("Diseased", 50)), weight = abs(rnorm(100, 50, 20)), severity = c(rep(0, 50),
 abs(rnorm(50, 100, 20))))
plmDEobject = plmDEmodel(Genes, ExpressionData, DataInfo)

## test whether severity and the indicator variable
## for disease are simultaneously significant:
test = fitGAPLM(plmDEobject, continuousCovariates.fullModel =
c("weight", "severity"), compareToReducedModel = TRUE,
indicators.reducedModel = NULL, continuousCovariates.reducedModel = "weight")
```

---

head.DEresults *Preview of Data in* DEresults *object*

Description

Modification of the head method for objects of type DEresults.

Usage

```
## S3 method for class 'DEresults'
head(x, ...)
```

Arguments

x	An item of type DEresults
...	Additional parameters to be passed to head

Author(s)

Jonas Mueller

See Also

head

head.plmDE *Preview of* plmDE *object*

Description

Modification of head for objects of type plmDE.

Usage

```
## S3 method for class 'plmDE'
head(x, ...)
```

Arguments

x	The object to preview.
...	Additional parameters to be passed to head

Author(s)

Jonas Mueller

See Also

head

hist.DEresults *Create a Histogram of P-values*

Description

Plots the distribution of p-values resulting from testing multiple genes for differential expression.

Usage

```
## S3 method for class 'DEresults'
hist(x, ...)
```

Arguments

x	An object of type DEresults containing the test results for each gene.
...	Additional parameters to be passed to hist

Author(s)

Jonas Mueller

References

Pounds, S. B. Estimation and control of multiple testing error rates for microarray studies. Brief Bioinformatics **7**(1), 25-36 (2006).

`info` *Information about an Object*

Description

Prints out a short summary of the data contained with an Object.

Usage

```
info(Object, ...)
```

Arguments

`Object`	print information about `Object`
`...`	additional instructions

Author(s)

Jonas Mueller

See Also

`info.plmDE`

`info.plmDE` *Obtain Information about* `plmDE` *object*

Description

Returns a short overview of the data contained `plmDE` objects.

Usage

```
## S3 method for class 'plmDE'
info(Object, ...)
```

Arguments

`Object`	An object of type `plmDE`.
`...`	Parameters from `info` left unused.

Author(s)

Jonas Mueller

See Also

info

Examples

```
## create an object of type \code{plmDE} containing disease
## with "control" and "disease" and measurements of weight and severity:
ExpressionData = as.data.frame(matrix(abs(rnorm(10000, 1, 1.5)), ncol = 100))
names(ExpressionData) = sapply(1:100, function(x) paste("Sample", x))
Genes = sapply(1:100, function(x) paste("Gene", x))
DataInfo = data.frame(sample = names(ExpressionData), group = c(rep("Control", 50), rep("Diseased", 50)), weight
plmDEobject = plmDEmodel(Genes, ExpressionData, DataInfo)

## get information:
info(plmDEobject)
```

---

limmaPLM *Adapt Additive Partially Linear Models for Testing via limma*

Description

Uses the methods of fitGAPLM to generate linear models of the class MArrayLM so that the moderated t and F methods of limma may be used to test for differential gene expression. See fitGAPLM for more a more in-depth description of the inputs.

Usage

```
limmaPLM(dataObject, intercept = TRUE,
indicators = as.character(unique(dataObject$sampleInfo[,2])[-1]),
continuousCovariates = NULL,
groups = as.character(unique(dataObject$sampleInfo[,2])[-1]),
groupFunctions = rep("AdditiveSpline", length(groups)),
fitSplineFromData = TRUE, splineDegrees = rep(3, length(groups)),
 splineKnots = rep(0, length(groups)), splineKnotSpread = "quantile", ...)
```

Arguments

dataObject	An object of type plmDE which we wish to test for differential gene expression.
intercept	Should an intercept term be included in the model?
indicators	Same as indicators.fullModel in fitGAPLM. Note that choice of intercept should affect choice of indicators.
continuousCovariates	Same as continuousCovariates.fullModel in fitGAPLM.

groups	Same as groups.fullModel in fitGAPLM.
groupFunctions	Same as groupFunctions.fullModel in fitGAPLM.
fitSplineFromData	Same as fitSplineFromData in fitGAPLM.
splineDegrees	Same as splineDegrees.fullModel in fitGAPLM.
splineKnots	Same as splineKnots.fullModel in fitGAPLM.
splineKnotSpread	Same as splineKnotSpread in fitGAPLM.
...	parameters to be passed to lmFit in limma.

Value

This method returns an MarrayLM object on which we can call eBayes() and topTable() to test for differentially expressed genes.

Author(s)

Jonas Mueller

References

Smyth, G. K. Linear Models and empirical Bayes methods for assesing differential expression in microarray experiments. Stat Appl Genet Mol Biol. **3**, Article 3 (2004).

See Also

fitGAPLM, plmDE, limma

Examples

```
## create an object of type \code{plmDE} containing disease
## with "control" and "disease" and measurements of weight and severity:
ExpressionData = as.data.frame(matrix(abs(rnorm(10000, 1, 1.5)), ncol = 100))
names(ExpressionData) = sapply(1:100, function(x) paste("Sample", x))
Genes = sapply(1:100, function(x) paste("Gene", x))
DataInfo = data.frame(sample = names(ExpressionData), group = c(rep("Control", 50),
rep("Diseased", 50)), weight = abs(rnorm(100, 50, 20)), severity = c(rep(0, 50),
abs(rnorm(50, 100, 20))))
plmDEobject = plmDEmodel(Genes, ExpressionData, DataInfo)

## create a linear model from which various hypotheses can be tested:
toTest = limmaPLM(plmDEobject, continuousCovariates = c("weight", "severity"),
 fitSplineFromData = TRUE, splineDegrees = rep(3, length(groups)),
 splineKnots = rep(0, length(groups)), splineKnotSpread = "quantile")

## view the coefficients/variables in the model:
toTest$coefficients[1, ]
weightCoefficients = c("DiseasedBasisFunction.weight.1",
"DiseasedBasisFunction.weight.2", "DiseasedBasisFunction.weight.3",
"DiseasedBasisFunction.weight.4", "DiseasedBasisFunction.weight.5",
```

```
"DiseasedBasisFunction.weight.6", "DiseasedBasisFunction.weight.7",
"DiseasedBasisFunction.weight.8", "DiseasedBasisFunction.weight.9")

## test the significance of weight in variation of the expression levels:
toTestCoefficients = contrasts.fit(toTest, coefficients = weightCoefficients)
moderatedTest = eBayes(toTestCoefficients)
topTableF(moderatedTest)
```

`mostDE` *Find Genes with the Most Evidence for Differential Expression*

Description

Returns the most significant genes in a `DEresults` object.

Usage

```
mostDE(results, n = 10)
```

Arguments

`results`	A `DEresults` object containing the results of a test for differential gene expression.
`n`	An integer describing the number of highest-significance genes to list.

Value

Returns a data frame of the top `n` genes ordered by significance along with their adjusted p-values under the test.

Author(s)

Jonas Mueller

See Also

`fitGAPLM`, `topTable`

Examples

```
## Run a test for differential gene expression:
ExpressionData = as.data.frame(matrix(abs(rnorm(10000, 1, 1.5)), ncol = 100))
names(ExpressionData) = sapply(1:100, function(x) paste("Sample", x))
Genes = sapply(1:100, function(x) paste("Gene", x))
DataInfo = data.frame(sample = names(ExpressionData), group = c(rep("Control", 50),
 rep("Diseased", 50)), weight = abs(rnorm(100, 50, 20)), severity = c(rep(0, 50),
 abs(rnorm(50, 100, 20))))
plmDEobject = plmDEmodel(Genes, ExpressionData, DataInfo)
test = fitGAPLM(plmDEobject, continuousCovariates.fullModel = c("weight", "severity"),
```

```
 compareToReducedModel = TRUE, indicators.reducedModel = NULL,
continuousCovariates.reducedModel = "weight")

## find 20 genes with most evidence for differential expression:
mostDE(test)
```

`plmDEmodel` *Constructor for objects of type* `plmDE`

Description

Creates an object which can hold expression data and other measurements of interest.

Usage

```
plmDEmodel(genes, ...)
```

Arguments

`genes`	Object containing the genes of interest from which an object of type `plmDE` should be made.
`...`	additional parameters

Author(s)

Jonas Mueller

See Also

`plmDEmodel.default`

`plmDEmodel.default` *Creates a class to hold Expression Data and Other Measurements*

Description

Given gene expression data as well as phenotypic measurements, `plmDEmodel` encodes them within a single object on which `fitGAPLM` and `limmaPLM` can be run.

Usage

```
## Default S3 method:
plmDEmodel(genes, expressionValues, sampleInfo, ...)
```

Arguments

genes	A vector of the genes (or probes/targets) from which each one is to be tested for differential expression.
expressionValues	A data frame or matrix containing the expression data of these genes. Each column should represent one sample and each row one gene.
sampleInfo	Information about the samples in expressionValues. This data frame must be specially formatted: first column must list samples (and must match the column names of expressionValues), second column must list the subgroups (i.e. 'disease', 'control') each sample belongs to, additional columns hold the measurements of some aspect of interest for each sample.
...	Parameters from plmDEmodel that are unused in this method.

Value

Returns an object of type plmDE on which fitGAPLM may be run to test for differential expression under a specified model.

Author(s)

Jonas Mueller

Examples

```
## create an object of type \code{plmDE} containing disease
## with "control" and "disease" and measurements of weight and severity:
ExpressionData = as.data.frame(matrix(abs(rnorm(10000, 1, 1.5)), ncol = 100))
names(ExpressionData) = sapply(1:100, function(x) paste("Sample", x))
Genes = sapply(1:100, function(x) paste("Gene", x))
DataInfo = data.frame(sample = names(ExpressionData), group = c(rep("Control", 50),
 rep("Diseased", 50)), weight = abs(rnorm(100, 50, 20)), severity = c(rep(0, 50),
 abs(rnorm(50, 100, 20))))
plmDEobject = plmDEmodel(Genes, ExpressionData, DataInfo)
```

---

plot.DEresults *Plot Fit of* DEresults *Model*

Description

Given a model contained in a DEresults object, plot.DEresults plots the fit of the model on the expression data for a specified gene/probe.

Usage

```
## S3 method for class 'DEresults'
plot(x, covariate, geneNumber = 1, plmDEobject,
loess = TRUE, legend = TRUE, legend.coor = "topright", ...)
```

Arguments

x	An object of type DEresults containing the model whose fitted values we wish to plot.
covariate	The covariate we wish to plot against the expression level data.
geneNumber	The index of the gene whose expression data should be plotted on the y-axis.
plmDEobject	An object of type plmDE containing all the data on expression and measurements of the covariates.
loess	Should a loess fit on the covariate and actual expression level data be plotted?
legend	Should a legend be plotted?
legend.coor	the coordinates of the legend. See legend for details.
...	parameters to be passed to plot

Author(s)

Jonas Mueller

See Also

fitGAPLM, plmDE

Examples

```
## create an object of type \code{plmDE} containing disease with "control"
## and "disease" groups with measures of weight and severity. Then fit model:
ExpressionData = as.data.frame(matrix(abs(rnorm(10000, 1, 1.5)), ncol = 100))
names(ExpressionData) = sapply(1:100, function(x) paste("Sample", x))
Genes = sapply(1:100, function(x) paste("Gene", x))
DataInfo = data.frame(sample = names(ExpressionData), group = c(rep("Control", 50),
rep("Diseased", 50)), weight = abs(rnorm(100, 50, 20)), severity = c(rep(0, 50),
abs(rnorm(50, 100, 20))))
plmDEobject = plmDEmodel(Genes, ExpressionData, DataInfo)
model = fitGAPLM(plmDEobject, continuousCovariates.fullModel = c("weight", "severity"),
 compareToReducedModel = TRUE, indicators.reducedModel = NULL,
continuousCovariates.reducedModel = "weight")
plot(model, "weight", 6, plmDEobject)
```

Index