Package 'glmmSeq'

October 13, 2022

Title General Linear Mixed Models for Gene-Level Differential Expression

Version 0.5.5

Description Using mixed effects models to analyse longitudinal gene expression can highlight differences between sample groups over time. The most widely used differential gene expression tools are unable to fit linear mixed effect models, and are less optimal for analysing longitudinal data. This package provides negative binomial and Gaussian mixed effects models to fit gene expression and other biological data across repeated samples. This is particularly useful for investigating changes in RNA-Sequencing gene expression between groups of individuals over time, as described in: Rivellese, F., Surace, A. E., Goldmann, K., Sciacca, E., Cubuk, C., Giorli, G., ... Lewis, M. J., & Pitzalis, C. (2022) Nature medicine <doi:10.1038/s41591-022-01789-0>.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

biocViews

RoxygenNote 7.2.1

Language en-gb

URL `https://myles-lewis.github.io/glmmSeq/`,
`https://github.com/myles-lewis/glmmSeq`

BugReports `https://github.com/myles-lewis/glmmSeq/issues`

Suggests knitr, rmarkdown, kableExtra, DESeq2, edgeR, emmeans

VignetteBuilder knitr

Depends R (>= 3.6.0)

Imports MASS, car, stats, ggplot2, ggpubr, glmmTMB, graphics, lme4, lmerTest, methods, plotly, qvalue, pbapply, pbmcapply

NeedsCompilation no

Author Myles Lewis [aut, cre] (<`https://orcid.org/0000-0001-9365-5345`>),
Katriona Goldmann [aut] (<`https://orcid.org/0000-0002-9073-6323`>),
Elisabetta Sciacca [aut] (<`https://orcid.org/0000-0001-7525-1558`>),

Cankut Cubuk [ctb] (<https://orcid.org/0000-0003-4646-0849>),
Anna Surace [ctb] (<https://orcid.org/0000-0001-9589-3005>)

Maintainer Myles Lewis <myles.lewis@qmul.ac.uk>

Repository CRAN

Date/Publication 2022-10-08 13:50:02 UTC

R topics documented:

fcPlot *Plotly or ggplot fold change plots*

Description

Plotly or ggplot fold change plots

Usage

```
fcPlot(
  object,
  x1var,
  x2var,
  x1Values = NULL,
  x2Values = NULL,
  pCutoff = 0.01,
  labels = c(),
  useAdjusted = FALSE,
  plotCutoff = 1,
  graphics = "ggplot",
  fontSize = 12,
  labelFontSize = 4,
  colours = c("grey", "goldenrod1", "red", "blue"),
  verbose = FALSE,
  ...
)
```

Arguments

`object` A glmmSeq object created by `glmmSeq::glmmSeq()`.

`x1var` The name of the first (inner) x parameter

`x2var` The name of the second (outer) x parameter

`x1Values` Timepoints or categories in `x1var` used to calculate fold change. If `NULL` the first two levels in `x1var` are used.

`x2Values` Categories in `x2var` to be compared on x and y axis.

`pCutoff` The significance cut-off for colour-coding (default = 0.01)

`labels` Row names or indices to label on plot

`useAdjusted` whether to use adjusted p-values (must have q-values in `object`). Default = FALSE

`plotCutoff` Which probes to include on plot by significance cut-off (default = 1, for all markers)

`graphics` Graphics system to use: "ggplot" or "plotly"

`fontSize` Font size

`labelFontSize` Font size for labels

`colours` Vector of colours to use for significance groups

`verbose` Whether to print statistics

`...` Other parameters to pass to plotly or ggplot

Value

Returns a plot for fold change between x1Values in one x2Value subset on x axis and fold change in the other x2Value on the y axis.

Examples

```
data(PEAC_minimal_load)

disp <- apply(tpm, 1, function(x) {
(var(x, na.rm = TRUE)-mean(x, na.rm = TRUE))/(mean(x, na.rm = TRUE)**2)
})

glmmFit <- glmmSeq(~ Timepoint * EULAR_6m + (1 | PATID),
                   countdata = tpm[1:5, ],
                   metadata = metadata,
                   dispersion = disp,
                   verbose = FALSE)

fcPlot(object = glmmFit,
         x1var = "Timepoint",
         x2var = "EULAR_6m",
         x2Values = c("Good", "Non-response"),
         pCutoff = 0.05,
         useAdjusted = FALSE,
         plotCutoff = 1,
         graphics = "plotly")
```

`ggmodelPlot` *Mixed model effects plot using ggplot2*

Description

Plot to show differences between groups and over time using ggplot2.

Usage

```
ggmodelPlot(
  object,
  geneName = NULL,
  x1var = NULL,
  x2var = NULL,
  x2shift = NULL,
  xlab = NULL,
  ylab = geneName,
  plab = NULL,
  title = geneName,
  logTransform = is(object, "GlmmSeq"),
  shapes = 19,
  colours = "grey60",
  lineColours = "grey60",
  markerSize = 1,
  fontSize = 12,
  alpha = 0.7,
  x2Offset = 5,
  addPoints = TRUE,
  addModel = TRUE,
  modelSize = 4,
  modelColours = "blue",
  modelLineSize = 1,
  modelLineColours = modelColours,
  addBox = FALSE,
  ...
)
```

Arguments

`object` A glmmSeq/lmmSeq object created by `glmmSeq::glmmSeq()` or `glmmSeq::lmmSeq()`

`geneName` The gene/row name to be plotted

`x1var` The name of the first (inner) x parameter, typically 'time'. This is anticipated to have different values when matched by ID.

`x2var` The name of an optional second (outer) x parameter, which should be a factor.

`x2shift` Amount to shift along x axis for each level of `x2var`. By default the function will arrange each level of `x2var` side by side.

`xlab` Title for the x axis

`ylab` Title for the y axis

`plab` Optional character vector of labels for p-values. These must align with column names in `object@stats$pvals`.

`title` Plot title. If NULL gene name is used

`logTransform` Whether to perform a log10 transform on the y axis

`shapes` The marker shapes (default=19)

`colours` The marker colours as vector

`lineColours` The line colours (default='grey60') as vector

`markerSize` Size of markers (default=1)

`fontSize` Plot font size

`alpha` Line and marker opacity (default=0.7)

`x2Offset` Vertical adjustment to secondary x-axis labels (default=5)

`addPoints` Whether to add underlying data points (default=TRUE)

`addModel` Whether to add the fit model with markers (default=TRUE)

`modelSize` Size of model points (default=4)

`modelColours` Colour of model fit markers (default="blue") as vector

`modelLineSize` Size of model points (default=1) as vector

`modelLineColours` Colour of model fit lines

`addBox` Logical whether to add boxplots for mean and IQR

`...` Other parameters to pass to `ggplot2::theme()`.

Value

Returns a paired plot for matched samples.

Examples

```
data(PEAC_minimal_load)

disp <- apply(tpm, 1, function(x){
  (var(x, na.rm=TRUE)-mean(x, na.rm=TRUE))/(mean(x, na.rm=TRUE)**2)
})

MS4A1glmm <- glmmSeq(~ Timepoint * EULAR_6m + (1 | PATID),
                     countdata = tpm['MS4A1', , drop = FALSE],
                     metadata = metadata,
                     dispersion = disp,
                     verbose = FALSE)

ggmodelPlot(object = MS4A1glmm,
           geneName = 'MS4A1',
           x1var = 'Timepoint',
           x2var = 'EULAR_6m',
           colours = c('skyblue', 'goldenrod1', 'mediumvioletred'))
```

glmmQvals *Glmm Sequencing qvalues*

Description

Add qvalue columns to the glmmSeq dataframe

Usage

```
glmmQvals(object, cutoff = 0.05, verbose = TRUE)
```

Arguments

object	A glmmSeq/lmmSeq object created by `glmmSeq::glmmSeq()`.
cutoff	Prints a table showing the number of probes considered significant by the pvalue cut-off (default=0.05)
verbose	Logical whether to print the number of significant probes (default=TRUE)

Value

Returns a GlmmSeq object with results for gene-wise general linear mixed models with adjusted p-values using the qvalue function

Examples

```
data(PEAC_minimal_load)
disp <- apply(tpm, 1, function(x) {
(var(x, na.rm=TRUE)-mean(x, na.rm = TRUE))/(mean(x, na.rm = TRUE)**2)
})
```

```
MS4A1glmm <- glmmSeq(~ Timepoint * EULAR_6m + (1 | PATID),
                     countdata = tpm[1:5, ],
                     metadata = metadata,
                     dispersion = disp[1:5],
                     verbose=FALSE)
MS4A1glmm <- glmmQvals(MS4A1glmm)
```

glmmRefit *Refit mixed effects model*

Description

Based on a 'GlmmSeq' or 'lmmSeq' class result object, this function attempts to refit an identical model for a specific gene based on the data and fitting parameters stored in the results object and refitting using either `lme4::glmer()` for `GlmmSeq` objects or `lmer()` for `lmmSeq` objects. The fitted model can then be passed on to other packages such as `emmeans` to look at estimated marginal means for the model.

Usage

```
glmmRefit(object, gene, ...)

## S3 method for class 'lmmSeq'
glmmRefit(object, gene, formula = object@formula, ...)

## S3 method for class 'GlmmSeq'
glmmRefit(
  object,
  gene,
  formula = object@formula,
  control = object@info$control,
  family = NULL,
  ...
)
```

Arguments

`object`	A fitted results object of class `GlmmSeq` or `lmmSeq`
`gene`	A character value specifying a single gene to extract a fitted model for
`...`	Optional arguments passed to either `lme4::glmer()` or `lme4::lmer()`
`formula`	Optional formula to use when refitting model
`control`	Optional control parameters, see `lme4::lmerControl()` or `lme4::glmerControl()`
`family`	Optional GLM family when refitting GLMM using `lme4::glmer()` or `glmmTMB()`

Value

Fitted model of class `lmerMod` in the case of LMM, or `glmerMod` or `glmmTMB` for a GLMM dependent on the original method.

Examples

```
data(PEAC_minimal_load)
disp <- apply(tpm, 1, function(x) {
  (var(x, na.rm = TRUE)-mean(x, na.rm = TRUE))/(mean(x, na.rm = TRUE)**2)
})
glmmtest <- glmmSeq(~ Timepoint * EULAR_6m + (1 | PATID),
                    countdata = tpm[1:2, ],
                    metadata = metadata,
                    dispersion = disp,
                    verbose = FALSE)

# show summary for single gene
summary(glmmtest, "MS4A1")

# refit a single model using lme4::glmer()
fit <- glmmRefit(glmmtest, "MS4A1")

# refit model with reduced formula
fit2 <- glmmRefit(glmmtest, "MS4A1",
                 formula = count ~ Timepoint + EULAR_6m + (1 | PATID))

# LRT
anova(fit, fit2)
```

`glmmSeq` *GLMM with negative binomial distribution for sequencing count data*

Description

Fits many generalised linear mixed effects models (GLMM) with negative binomial distribution for analysis of overdispersed count data with random effects. Designed for longitudinal analysis of RNA-Sequencing count data.

Usage

```
glmmSeq(
  modelFormula,
  countdata,
  metadata,
  id = NULL,
  dispersion = NA,
  sizeFactors = NULL,
  reduced = NULL,
  modelData = NULL,
  designMatrix = NULL,
  method = c("lme4", "glmmTMB"),
  control = NULL,
  family = nbinom2,
  cores = 1,
  removeSingles = FALSE,
  zeroCount = 0.125,
  verbose = TRUE,
  returnList = FALSE,
  progress = FALSE,
  ...
)
```

Arguments

`modelFormula` the model formula. This must be of the form `"~ ..."` where the structure is assumed to be `"counts ~ ..."`. The formula must include a random effects term. For more information on formula structure for random effects see `lme4::glmer()`

`countdata` the sequencing count data matrix with genes in rows and samples in columns

`metadata` a dataframe of sample information with variables in columns and samples in rows

`id` Optional. Used to specify the column in metadata which contains the sample IDs to be used in repeated samples for random effects. If not specified, the function defaults to using the variable after the "|" in the random effects term in the formula.

`dispersion` a numeric vector of gene dispersion. Not required for `glmmTMB` models, as these determine and fit dispersion for each gene.

`sizeFactors` size factors (default = NULL). If provided the `glmer` offset is set to log(sizeFactors). For more information see“ `lme4::glmer()`

`reduced` Optional reduced model formula. If this is chosen, a likelihood ratio test is used to calculate p-values instead of the default Wald type 2 Chi-squared test.

`modelData` Optional dataframe. Default is generated by call to `expand.grid` using levels of variables in the formula. Used to calculate model predictions (estimated means & 95% CI) for plotting via modelPlot. It can therefore be used to add/remove points in modelPlot.

`designMatrix` custom design matrix, used only for prediction

`method` Specifies which package to use for fitting GLMM models. Either "lme4" or "glmmTMB" depending on whether to use lme4::glmer or glmmTMB::glmmTMB to fit GLMM models.

`control` the `glmer` optimizer control. If `method = "lme4"` default is `glmerControl(optimizer = "bobyqa")`). If `method = "glmmTMB"` default is `glmmTMBControl()`

`family` Only used with `glmmTMB` models. Default is `nbinom2`. See glmmTMB::nbinom2

`cores` number of cores to use. Default = 1.

`removeSingles` whether to remove individuals without repeated measures (default = FALSE)

`zeroCount` numerical value to offset zeroes for the purpose of log (default = 0.125)

`verbose` Logical whether to display messaging (default = TRUE)

returnList	Logical whether to return results as a list or glmmSeq object (default = FALSE). Useful for debugging.
progress	Logical whether to display a progress bar
...	Other parameters to pass to lme4::glmer()

Details

This function is a wrapper for lme4::glmer(). By default, p-values for each model term are computed using Wald type 2 Chi-squared test as per car::Anova(). The underlying code for this has been optimised for speed. However, if a reduced model formula is specified by setting reduced, then a likelihood ratio test is performed instead using stats::anova. This will double computation time since two GLMM have to be fitted.

Parallelisation is provided using parallel::mclapply on Unix/Mac or parallel::parLapply on PC.

Setting method = "glmmTMB" enables an alternative method of fitting GLMM using the glmmTMB package. This gives access to a variety of alternative GLM family functions. Note, glmmTMB negative binomial models are substantially slower to fit than glmer models with known dispersion due to the extra time taken by glmmTMB to optimise the dispersion parameter.

The id argument is usually optional. By default the id column in the metadata is determined as the term after the bar in the random effects term of the model. Note that id is not passed to glmer or glmmTMB. It is only really used to remove singletons from the countdata matrix and metadata dataframe. The id is also stored in the output from glmmSeq and used by plotting function modelPlot(). However, due to its flexible nature, in theory glmmSeq should allow for more than one random effect term, although this has not been tested formally. In this case, it is probably prudent to specify a value for id.

Value

Returns an S4 class GlmmSeq object with results for gene-wise general linear mixed models. A list of results is returned if returnList is TRUE which is useful for debugging. If all genes return errors from glmer, then an error message is shown and a character vector containing error messages for all genes is returned.

See Also

lme4::glmer lme4::glmerControl glmmTMB::glmmTMB glmmTMB::nbinom2 glmmTMB::glmmTMBControl car::Anova

Examples

```
data(PEAC_minimal_load)
disp <- apply(tpm, 1, function(x) {
(var(x, na.rm = TRUE)-mean(x, na.rm = TRUE))/(mean(x, na.rm = TRUE)**2)
})
MS4A1glmm <- glmmSeq(~ Timepoint * EULAR_6m + (1 | PATID),
                     countdata = tpm[1:2, ],
                     metadata = metadata,
                     dispersion = disp,
                     verbose = FALSE)
names(attributes(MS4A1glmm))
```

`GlmmSeq-class` *An S4 class to define the glmmSeq output*

Description

An S4 class to define the glmmSeq output

Slots

- `info` List including the matched call, dispersions, offset, designMatrix
- `formula` The model formula
- `stats` Statistics from fitted models
- `predict` Predicted values
- `reduced` Optional reduced formula for LRT
- `countdata` The input expression data with count data in rows
- `metadata` The input metadata
- `modelData` Model data for predictions
- `optInfo` Information on whether the model was singular or converged
- `errors` Any errors
- `vars` List of variables stored from the original call, including the `id` variable (by default automatically identified from the random effect term in the model) and `removeSingles` argument

`lmmSeq` *Linear mixed models for data matrix*

Description

Fits many linear mixed effects models for analysis of gaussian data with random effects, with parallelisation and optimisation for speed. It is suitable for longitudinal analysis of high dimensional data. Wald type 2 Chi-squared test is used to calculate p-values.

Usage

```
lmmSeq(
  modelFormula,
  maindata,
  metadata,
  id = NULL,
  offset = NULL,
  test.stat = c("Wald", "F", "LRT"),
  reduced = NULL,
  modelData = NULL,
  designMatrix = NULL,
  control = lmerControl(),
  cores = 1,
  removeSingles = FALSE,
  verbose = TRUE,
  returnList = FALSE,
  progress = FALSE,
  ...
)
```

Arguments

`modelFormula` the model formula. This must be of the form `"~ ..."` where the structure is assumed to be `"gene ~ ..."`. The formula must include a random effects term. See formula structure for random effects in `lme4::lmer()`

`maindata` data matrix with genes in rows and samples in columns

`metadata` a dataframe of sample information with variables in columns and samples in rows

`id` Optional. Used to specify the column in metadata which contains the sample IDs to be used in repeated samples for random effects. If not specified, the function defaults to using the variable after the "|" in the random effects term in the formula.

`offset` Vector containing model offsets (default = NULL). If provided the `lmer()` offset is set to `offset`. See `lme4::lmer()`

`test.stat` Character value specifying test statistic. Current options are "Wald" for type 2 Wald Chi square test using code derived and modified from car::Anova to improve speed for matrix tests. Or "F" for conditional F tests using Saiterthwaite's method of approximated Df. This uses lmerTest::lmer and is somewhat slower.

`reduced` Optional reduced model formula. If this is chosen, a likelihood ratio test is used to calculate p-values instead of the default Wald type 2 Chi-squared test.

`modelData` Optional dataframe. Default is generated by call to `expand.grid` using levels of variables in the formula. Used to calculate model predictions (estimated means & 95% CI) for plotting via modelPlot. It can therefore be used to add/remove points in modelPlot.

`designMatrix` Optional custom design matrix generated by call to `model.matrix` using `modelData` and `FEformula`. Used to calculate model predictions for plotting.

`control` the `lmer` optimizer control (default = `lmerControl()`). See `lme4::lmerControl()`.

`cores` number of cores to use for parallelisation. Default = 1.

`removeSingles` whether to remove individuals with no repeated measures (default = FALSE)

`verbose` Logical whether to display messaging (default = TRUE)

`returnList` Logical whether to return results as a list or lmmSeq object (default = FALSE). Helpful for debugging.

`progress` Logical whether to display a progress bar

`...` Other parameters passed to `lmerTest::lmer()`. Only available if `test.stat = "F"`.

Details

By default, p-values for each model term are computed using Wald type 2 Chi-squared test as per `car::Anova()`. The underlying code for this has been optimised for speed. However, if a reduced model formula is specified by setting `reduced`, then a likelihood ratio test (LRT) is performed instead using `anova`. This will double computation time since two LMM have to be fitted for each gene. For LRT, models being compared are optimised by maximum likelihood and not REML (`REML=FALSE`).

Two key methods are used to speed up computation above and beyond simple parallelisation. The first is to speed up `lme4::lmer()` by calling `lme4::lFormula()` once at the start and then updating the `lFormula` output with new data. The 2nd speed up is through optimised code for repeated type 2 Wald Chi-squared tests (original code was derived from car::Anova). For example, elements such as the hypothesis matrices are generated only once to reduce unnecessarily repetitive computation, and the generation of p-values from Chi-squared values is vectorised and performed at the end. F-tests using the `lmerTest` package have not been optimised and are therefore slower.

Parallelisation is performed using parallel::mclapply on unix/mac and parallel::parLapply on windows. Progress bars use pbmcapply::pbmclapply on unix/mac and pbapply::pblapply on windows.

The `id` argument is usually optional. By default the `id` column in the metadata is determined as the term after the bar in the random effects term of the model. Note that `id` is not passed to `lmer`. It is only really used to remove singletons from the `maindata` matrix and `metadata` dataframe. The `id` is also stored in the output from `lmmSeq` and used by plotting function `modelPlot()`. However, due to its flexible nature, in theory `lmmSeq` should allow for more than one random effect term, although this has not been tested formally. In this case, it is probably prudent to specify a value for `id`.

Value

Returns an S4 class `lmmSeq` object with results for gene-wise linear mixed models; or a list of results if `returnList` is `TRUE`, which is useful for debugging. If all genes return errors from `lmer`, then an error message is shown and a character vector containing error messages for all genes is returned.

Examples

```
data(PEAC_minimal_load)
logtpm <- log2(tpm +1)
lmmtest <- lmmSeq(~ Timepoint * EULAR_6m + (1 | PATID),
                  maindata = logtpm[1:2, ],
                  metadata = metadata,
                        verbose = FALSE)
names(attributes(lmmtest))
```

lmmSeq-class *An S4 class to define the lmmSeq output*

Description

An S4 class to define the lmmSeq output

Slots

`info` List including matched call, offset, designMatrix

`formula` The model formula

`stats` Statistics from fitted models

`predict` Predicted values

`reduced` Optional reduced formula for LRT

`maindata` The input expression data with variables in rows

`metadata` The input metadata

`modelData` Model data for predictions

`optInfo` Information on whether the model was singular or converged

`errors` Any errors

`vars` List of variables stored from the original call

maPlot *MA plots*

Description

MA plots

Usage

```
maPlot(
  object,
  x1var,
  x2var,
  x1Values = NULL,
  x2Values = NULL,
  pCutoff = 0.01,
  plotCutoff = 1,
  zeroCountCutoff = 50,
  colours = c("grey", "midnightblue", "mediumvioletred", "goldenrod"),
  labels = c(),
  fontSize = 12,
  labelFontSize = 4,
  useAdjusted = FALSE,
  graphics = "ggplot",
  verbose = FALSE
)
```

Arguments

`object`	A glmmSeq object created by `glmmSeq::glmmSeq()`.
`x1var`	The name of the first (inner) x parameter
`x2var`	The name of the second (outer) x parameter
`x1Values`	Timepoints or categories in `x1var` to be used to calculate fold change. If `NULL` the first two levels in `x1var` are used.
`x2Values`	Categories in `x2var` to be compared on x and y axis.
`pCutoff`	The significance cut-off for colour-coding (default=0.01)
`plotCutoff`	Which probes to include by significance cut-off (default=1 for all markers)
`zeroCountCutoff`	Which probes to include by minimum counts cut-off (default=50)
`colours`	Vector of colours to use for significance groups
`labels`	Row names or indices to label on plot
`fontSize`	Font size
`labelFontSize`	Font size for labels
`useAdjusted`	whether to use adjusted p-values (must have q-values in `object`)
`graphics`	Either "ggplot" or "plotly"
`verbose`	Whether to print statistics

Value

List of three plots. One plot for each `x2Value` and one combined figure

Examples

```
data(PEAC_minimal_load)

disp <- apply(tpm, 1, function(x){
(var(x, na.rm=TRUE)-mean(x, na.rm=TRUE))/(mean(x, na.rm=TRUE)**2)
})

resultTable <- glmmSeq(~ Timepoint * EULAR_6m + (1 | PATID),
                       countdata = tpm[1:5, ],
                       metadata = metadata,
                       dispersion = disp)
```

```
plots <- maPlot(resultTable,
                x1var='Timepoint',
                x2var='EULAR_6m',
                x2Values=c('Good', 'Non-response'),
                graphics="plotly")

plots$combined
```

metadata *Minimal metadata from PEAC*

Description

Minimal metadata for paired longitudinal response analysis.

Usage

```
metadata
```

Format

A data frame

PATID Id for matching patients

Timepoint timepoints

EULAR_6m response data

modelPlot *Mixed model effects plot*

Description

Plot to show differences between groups over time using base graphics.

Usage

```
modelPlot(
  object,
  geneName = NULL,
  x1var = NULL,
  x2var = NULL,
  x2shift = NULL,
  xlab = NA,
  ylab = geneName,
  plab = NULL,
  title = geneName,
  logTransform = is(object, "GlmmSeq"),
  shapes = 21,
  colours = "grey60",
  lineColours = "grey60",
  markerSize = 0.5,
  fontSize = NULL,
  alpha = 0.7,
  addModel = TRUE,
  addPoints = TRUE,
  modelSize = 2,
  modelColours = "royalblue",
  modelLineSize = 1,
  modelLineColours = modelColours,
  errorBarLwd = 2.5,
  errorBarLength = 0.05,
  ...
)
```

Arguments

`object`	A glmmSeq/lmmSeq object created by `glmmSeq::glmmSeq()` or `glmmSeq::lmmSeq()`
`geneName`	The gene/row name to be plotted
`x1var`	The name of the first (inner) x parameter, typically 'time'. This is anticipated to have different values when matched by ID.
`x2var`	The name of an optional second (outer) x parameter, which should be a factor.
`x2shift`	Amount to shift along x axis for each level of `x2var`. By default the function will arrange each level of `x2var` side by side. Lower values of `x2shift` or `x2shift = 0` can be used to overlap plots similar to 'dodge' or stagger them.
`xlab`	Title for the x axis
`ylab`	Title for the y axis
`plab`	Optional character vector of labels for p-values. These must align with column names in `object@stats$pvals`.
`title`	Plot title. If NULL gene name is used
`logTransform`	Whether to perform a log10 transform on the y axis
`shapes`	The marker shapes (default=19)
`colours`	The marker colours (default='red') as vector or named vector
`lineColours`	The line colours (default='grey60') as vector or named vector
`markerSize`	Size of markers (default=2)
`fontSize`	Plot font size
`alpha`	Line and marker opacity (default=0.7)
`addModel`	Whether to add the fit model with markers (default=TRUE)
`addPoints`	Whether to add underlying data points (default=TRUE)
`modelSize`	Size of model points (default=2)

`modelColours` Colour of model fit markers (default="black") as vector or named vector

`modelLineSize` Size of model points (default=1) as vector or named vector

`modelLineColours` Colour of model fit lines.

`errorBarLwd` Line width of error bars

`errorBarLength` Head width of error bars

`...` Other parameters to pass to `graphics::plot()`

Value

Returns a paired plot for matched samples

Examples

```
data(PEAC_minimal_load)

disp <- apply(tpm, 1, function(x){
  (var(x, na.rm=TRUE)-mean(x, na.rm=TRUE))/(mean(x, na.rm=TRUE)**2)
})

MS4A1glmm <- glmmSeq(~ Timepoint * EULAR_6m + (1 | PATID),
                     countdata = tpm[1:2, ],
                     metadata = metadata,
                     dispersion = disp)

modelPlot(object=MS4A1glmm,
          geneName = 'MS4A1',
          x1var = 'Timepoint',
          x2var='EULAR_6m')
```

`summary.lmmSeq` *Summarise a 'glmmSeq'/'lmmSeq' object*

Description

Summarise results from glmmSeq or lmmSeq analysis

Usage

```
## S3 method for class 'lmmSeq'
summary(object, gene = NULL, digits = max(3L, getOption("digits") - 3L), ...)

## S3 method for class 'GlmmSeq'
summary(object, gene = NULL, ...)
```

Arguments

`object` an object of class `"GlmmSeq"` or `"lmmSeq"`

`gene` an optional character value specifying a single gene whose results are summarised

`digits` integer, used for number formatting

`...` arguments to be passed to other methods

Value

If `gene=NULL` a dataframe of results for all genes is returned. Otherwise the output of GLMM or LMM model results for a single gene including coefficients, test statistics, p-values is printed and the dataframe for all genes is returned invisibly.

See Also

`glmmSeq()`, `lmmSeq()`

`tpm` *TPM count data from PEAC*

Description

Transcripts Per Million (TPM) count data for PEAC synovial biopsies.

Usage

```
tpm
```

Format

An object of class `matrix` (inherits from `array`) with 50 rows and 123 columns.

Index