Package ‘MCMC.qpcr’

October 12, 2022

Type Package

Title Bayesian Analysis of qRT-PCR Data

Version 1.2.4

Date 2020-03-27

Author Mikhail V. Matz

Maintainer Mikhail V. Matz <matz@utexas.edu>

Description Quantitative RT-PCR data are analyzed using generalized linear mixed models based on lognormal-Poisson error distribution, fitted using MCMC. Control genes are not required but can be incorporated as Bayesian priors or, when template abundances correlate with conditions, as trackers of global effects (common to all genes). The package also implements a lognormal model for higher-abundance data and a ``classic'' model involving multi-gene normalization on a by-sample basis. Several plotting functions are included to extract and visualize results. The detailed tutorial is available here: <https://matzlab.weebly.com/uploads/7/6/2/2/76229469/mcmc.qpcr.tutorial.v1.2.4.pdf>.

License GPL-3

Depends MCMCglmm,ggplot2,coda

NeedsCompilation no

Repository CRAN

Date/Publication 2020-03-29 04:10:02 UTC

R topics documented:

`MCMC.qpcr-package` *Bayesian analysis of qRT-PCR data*

Description

This package implements generalized linear mixed model analysis of qRT-PCR data so that the increase of variance towards higher Ct values is properly dealt with, and the lack of amplification is informative (function mcmc.qpcr). Sample-loading effects, gene-specific variances, and responses of all genes to each factor combination are all jointly estimated within a single model. The control genes can be specified as priors, with adjustable degree of expected stability. The analysis also works well without any control gene specifications.

For higher-abundance datasets, a lognormal model is implemented that does not require Cq to counts conversion (function mcmc.qpcr.lognormal).

For higher-abundance datasets datasets in which the quality and/or quantity of RNA samples varies systematically (rather than randomly) across conditions, the analysis based on multigene normalization is implemented (function mcmc.qpcr.classic).

The package includes several functions for plotting the results and calculating statistical significance (HPDplot, HPDplotBygene, HPDplotBygeneBygroup).

The detailed step-by-step tutorial is here: http://www.bio.utexas.edu/research/matz_lab/matzlab/Methods_files/mcmc.qpcr.tu

Details

Package:	MCMC.qpcr
Type:	Package
Version:	1.2.3
Date:	2016-11-07
License:	GPL-3

Author(s)

Mikhail V. Matz, University of Texas at Austin <matz@utexas.edu>

References

Matz MV, Wright RM, Scott JG (2013) No Control Genes Required: Bayesian Analysis of qRT-PCR Data. PLoS ONE 8(8): e71448. doi:10.1371/journal.pone.0071448

Examples

```
data(beckham.data)
data(beckham.eff)

# analysing the first 5 genes
# (to try it with all 10 genes, change the line below to gcol=4:13)
gcol=4:8
ccol=1:3 # columns containing experimental conditions

# recalculating into molecule counts, reformatting
qs=cq2counts(data=beckham.data,genecols=gcol,
condcols=ccol,effic=beckham.eff,Cq1=37)

# creating a single factor, 'treatment.time', out of 'tr' and 'time'
qs$treatment.time=as.factor(paste(qs$tr,qs$time,sep="."))

# fitting a naive model
naive=mcmc.qpcr(
fixed="treatment.time",
data=qs,
nitt=3000,burnin=2000 # remove this line in actual analysis!
)

#summary plot of inferred abundances
#s1=HPDsummary(model=naive,data=qs)

#summary plot of fold-changes relative to the global control
s0=HPDsummary(model=naive,data=qs,relative=TRUE)
```

```
#correcting p-values for multiple comparisons
s0.adj=padj.hpdsummary(s0,controls=c("gapdh"))

# pairwise differences and their significances for each gene:
s0.adj$geneWise
```

---

amp.eff *amplification efficiencies and experimental Cq1 (optional column)*

Description

Fold-amplification within each PCR cycle for each qRT-PCR assay. Determined by qPCR of serial dilutions of the template (see Pfaffl 2001, Nucleic Acids Res 29:00)

Usage

```
data(amp.eff)
```

Format

A data frame with 17 observations on the following 3 variables.

`gene` a factor with levels `actin` `adk` `c3` `chrom` `clect` `eif3h` `g3pdh` `gsp2` `hsp16` `hsp60` `hsp90` `nd5` `r18s` `rpl11` `spon2` `tgoln` `ubl3`

`efficiency` a numeric vector

`Cq1` a numeric vector

Examples

```
data(amp.eff)
```

---

beckham.data *Cellular heat stress response data.*

Description

A typical qRT-PCR dataset where a series of treatments are compared to the global control.

Usage

```
data(beckham.data)
```

Format

A data frame with 45 observations on the following 13 variables.

`sample` cDNA sample (biological replicate): a factor with levels `a` `b` `c` `d` `e` `f` `g` `h` `i` `j` `k` `l` `m` `n` `o`

`tr` treatment: a factor with levels `ctl`:control `hs`:heat stress `pshs`:pre-stress followed by heat stress

`time` hours of heat stress exposure: a numeric vector

`gapdh` a numeric vector

`hsp110` a numeric vector

`hspb` a numeric vector

`egr` a numeric vector

`gadd` a numeric vector

`dnajb1` a numeric vector

`dnajb4` a numeric vector

`atf` a numeric vector

`dnaja4` a numeric vector

`fos` a numeric vector

Details

Includes global control (ctl), heat stressed cells (hs), and cells pre-stressed by mild heating prior to heat stress (pshs). Two time points: one hour and three hours.

Source

data for Figure 6 in Beckham et al. Microarray analysis of cellular thermotolerance. Lasers Surg Med. 2010 Dec;42(10):752-65. doi: 10.1002/lsm.20983

Examples

```
data(beckham.data)
```

`beckham.eff` *amplification efficiencies for beckham.data*

Description

amplification efficiencies for beckham.data

Usage

```
data(beckham.eff)
```

Format

A data frame with 11 observations on the following 2 variables.

`gene` a factor with levels `atf` `dnaja4` `dnajb1` `dnajb4` `egr` `fos` `gadd` `gapdh` `hsp110` `hspb` `r18s`

`eficiency` a numeric vector

Source

Beckham et al. Microarray analysis of cellular thermotolerance. Lasers Surg Med. 2010 Dec;42(10):752-65. doi: 10.1002/lsm.20983

Examples

```
data(beckham.eff)
```

`coral.stress` *RT-qPCR of stress response in coral Porites astreoides*

Description

Timepoint one is one day of heat-light stress; timepoint two is recovery from it the next day. Controls are the fragments never exposed to stress. Individual denotes a coral colony that yielded several fragments for different treatments.

Usage

```
data(coral.stress)
```

Format

A data frame with 64 observations on the following 19 variables.

`sample` a numeric vector

`individual` a factor with levels `s1` `s12` `s13` `s15` `s2` `s4` `s5` `s8`

`condition` a factor with levels `control` `heat`

`timepoint` a factor with levels `one` `two`

`hsp16` a numeric vector

`actin` a numeric vector

`adk` a numeric vector

`c3` a numeric vector

`chrom` a numeric vector

`clect` a numeric vector

`eif3h` a numeric vector

`g3pdh` a numeric vector

`gsp2`	a numeric vector
`hsp60`	a numeric vector
`hsp90`	a numeric vector
`nd5`	a numeric vector
`rpl11`	a numeric vector
`spon2`	a numeric vector
`ubl3`	a numeric vector

Examples

```
data(coral.stress)
```

`cq2counts` *Prepares qRT-PCR data for mcmc.qpcr analysis*

Description

Converts Cq values into molecule counts, and stacks the dataset

Usage

```
cq2counts(data, genecols, condcols, effic, Cq1 = NULL)
```

Arguments

`data`	Raw qRT-PCR dataset, one Cq column per gene, plus columns containing factors. The Cq columns, in addition to the proper Cq values, may contain NA (missing data) and -1, which means no amplification observed (i.e., zero target molecules at the start of qPCR reaction). Column headers are either gene names or factor names. Any number of fixed factors is allowed; any number of random factors that are gene-specific scalars (such as effect of genotype, or block) Must have a column called "sample", denoting individual cDNA preps. Technical replicates should not be averaged, they should be represented as independent rows with the same sample ID.
`genecols`	columns that contain Cq data
`condcols`	columns corresponding to factors, including "sample" factor
`effic`	The PCR efficiency data for each of the analyzed genes. This is data frame with two columns: gene name (must exactly match the headers of gene columns in Cq data table!) and efficiency (fold- amplification per PCR cycle, determined from qPCR of serial dilutions; see PrimEff() function)
`Cq1`	The Cq of a single molecule. If left unspecified, it will be calculated from the efficiency (E) using approximate formula Cq1=51.6-7.56*E, derived empirically for Roche's LightCycler 480. Cq1 does not seem to have much effect on relative quantification results unless it is wildly off (by 2-3 cycles). For an unknown qPCR instrument a single Cq1=37 could be assumed for all genes.

Note: If all experimental Cq values are less than 30, Cq1 variation (within a reasonable range, 35-39) will not have any effect on the results whatsoever, so just go for Cq1=37.

Value

Returns a dataframe with a single response variable column ("count"), gene column ("gene") and several columns containing factors.

Note: The purpose of Cq to counts conversion is to enable generalized linear model analysis, which would take care of the heteroscedasticity and occasional 'empty' amplification trials for low-abundant targets. Although this works well, the absolute values of molecule counts returned by cq2counts are still approximate, so they should not be relied upon for true single-molecule analysis.

Author(s)

Mikhail V. Matz, UT Austin <matz@utexas.edu>

References

Matz MV, Wright RM, Scott JG (2013) No Control Genes Required: Bayesian Analysis of qRT-PCR Data. PLoS ONE 8(8): e71448. doi:10.1371/journal.pone.0071448

Examples

```
data(coral.stress)
data(amp.eff)
genecolumns=c(5:19) # specifying where the Ct data are in the data table
conditions=c(1:4) # specifying data table columns containing factors
# calculating molecule counts and reformatting:
dd=cq2counts(data=coral.stress,genecols=genecolumns,
condcols=conditions,effic=amp.eff,Cq1=37)
```

`cq2genorm` *Reformats raw Ct data for geNorm analysis (non-parametric selection of stable control genes) as implemented in selectHKgenes function (package SLqPCR)*

Description

Does similar procedures as cq2log, but in the end converts the data into relative expression values, leaves it unstacked, and removes factor columns.

Usage

```
cq2genorm(data, genes, effic, noamp=38)
```

Arguments

`data`	raw qRT-PCR data; see help for cq2counts for details on formatting
`genes`	Vector of names of the potential control genes. Make sure you select only the potential control genes for this analysis, otherwise geNorm might prefer actual responsive genes if they co-vary and have very low variance.
`effic`	PCR efficiency data for each of the analyzed genes; see help for cq2counts for details on formatting.
`noamp`	what to do about Ct values '-1', denoting no amplification. By default, these will be assigned an arbitrarily low expression value corresponding to slightly less than one molecule. Specify 'noamp=NA' if you want to skip samples containing any of these.

Details

See cq2counts help page for details.

Value

A dataset to be fed into the function selectHKgenes (package SLqPCR).

Author(s)

Mikhail V. Matz, UT Austin

References

Matz MV, Wright RM, Scott JG (2013) No Control Genes Required: Bayesian Analysis of qRT-PCR Data. PLoS ONE 8(8): e71448. doi:10.1371/journal.pone.0071448

`cq2log` *Prepares qRT-PCR data for mcmc.qpcr analysis using lognormal and "classic" (normalization-based) models*

Description

Log-transforms and efficiency-corrects Cq values, converting them to natural logs fo relative abundances, and stacks the dataset

Usage

```
cq2log(data, genecols, condcols, effic, noamp = 38, stacked=TRUE)
```

Arguments

`data`	Raw qRT-PCR dataset, one Cq column per gene, plus columns containing factors. The Cq columns, in addition to the proper Cq values, may contain NA (missing data) and -1, which means no amplification observed (i.e., zero target molecules at the start of qPCR reaction). Column headers are either gene names or factor names. Any number of fixed factors is allowed; any number of random factors that are gene-specific scalars (such as effect of genotype, or block) Must have a column called "sample", denoting individual cDNA preps. Technical replicates should not be averaged, they should be represented as independent rows with the same sample ID.
`genecols`	columns containing Cq data
`condcols`	columns corresponding to factors, including "sample" factor
`effic`	The PCR efficiency data for each of the analyzed genes. This is data frame with two columns: gene name (must exactly match the headers of gene columns in Cq data table!) and efficiency (fold- amplification per PCR cycle, determined from qPCR of serial dilutions; see PrimEff() function)
`noamp`	Value to replace instances of no amplification with. These instances would be coded by -1 in the data table. Specify 'noamp=NA' if you want to disregard them, but by default they will be converted into an arbitrarily low value, 38
`stacked`	Logical: whether to return stacked data for mcmc.qpcr modeling, or the originally-formatted table.

Details

The models that process cq2log output are expected to work well for datasets that don't have too many Cq values above 30 and don't have instances of no amplification. For examples, see cq2counts function.

Value

Returns a dataframe with a single response variable column ("count", even though it is actually a log-transformed relative abundance value), gene column ("gene") and several columns containing factors.

Author(s)

Mikhail V. Matz, UT Austin

References

Matz MV, Wright RM, Scott JG (2013) No Control Genes Required: Bayesian Analysis of qRT-PCR Data. PLoS ONE 8(8): e71448. doi:10.1371/journal.pone.0071448

`diagnostic.mcmc` *Plots three diagnostic plots to check the validity of the assumptions of linear model analysis.*

Description

Predicted vs observed plot tests for linearity, Scale-location plot tests for homoscedasticity, and Normal QQ plot tests for normality of the residuals.

Usage

```
diagnostic.mcmc(model, ...)
```

Arguments

`model`	MCMCglmm object (a model fitted by mcmc.qpcr or mcmc.qpcr.gauss), obtained with additional options, 'pl=T, pr=T'
`...`	Various plot() options to modify color, shape and size of the plotteed points.

Value

A plot with three panels.

Author(s)

Mikhail V. Matz, UT Austin <matz@utexas.edu>

References

Matz MV, Wright RM, Scott JG (2013) No Control Genes Required: Bayesian Analysis of qRT-PCR Data. PLoS ONE 8(8): e71448. doi:10.1371/journal.pone.0071448

Examples

```
# loading Cq data and amplification efficiencies
data(coral.stress)
data(amp.eff)
# extracting a subset of data
cs.short=subset(coral.stress, timepoint=="one")

genecolumns=c(5,6,16,17) # specifying columns corresponding to genes of interest
conditions=c(1:4) # specifying columns containing factors

# calculating molecule counts and reformatting:
dd=cq2counts(data=cs.short,genecols=genecolumns,
condcols=conditions,effic=amp.eff,Cq1=37)

# fitting the model
```

```
mm=mcmc.qpcr(
fixed="condition",
data=dd,
controls=c("nd5","rpl11"),
pr=TRUE,pl=TRUE, # these flags are necessary for diagnostics
nitt=4000 # remove this line when analyzing real data!
)
diagnostic.mcmc(mm)
```

`dilutions` *Data to determine amplification efficiency*

Description

Cq data for a series of four-fold dilutions for two targets; the input for PrimEff()

Usage

```
data(dilutions)
```

Format

A data frame with 76 observations on the following 3 variables.

`dna` a numeric vector

`cq` a numeric vector

`gene` a factor with levels `chrom` `eif3h`

Examples

```
data(dilutions)
PrimEff(dilutions)
```

`getNormalizedData` *Extracts qPCR model predictions*

Description

Generates a table of model-derived log2-transformed transcript abundances without global sample effects (i.e., corresponding to efficiency-corrected and normalized qPCR data)

Usage

```
getNormalizedData(model, data, controls=NULL)
```

Arguments

`model`	qPCR model: the output of mcmc.qpcr or mcmc.qpcr.lognormal function fitted with two additional options: random="sample", pr=TRUE . These options do not change the inferences of main effects but make it possible to retain among-sample variation of expression for each gene while still subtracting the global sample effects (i.e., perform "normalization")
`data`	The dataset that was analysed to generate the model (output of cq2counts or cq2log functions)
`controls`	List of control genes; required if the mcmc.qpcr model was fit with the option normalize=TRUE

Value

The function returns a list of two data frames. The first one, normData, is the model-predicted log2-transformed transcript abundances table. It has one column per gene and one row per sample. The second data frame, conditions, is a table of experimental conditions corresponding to the normData table.

Author(s)

Mikhail V. Matz, University of Texas at Austin <matz@utexas.edu>

References

Matz MV, Wright RM, Scott JG (2013) No Control Genes Required: Bayesian Analysis of qRT-PCR Data. PLoS ONE 8(8): e71448. doi:10.1371/journal.pone.0071448

Examples

```
library(MCMC.qpcr)

# loading Cq data and amplification efficiencies
data(coral.stress)
data(amp.eff)

genecolumns=c(5,6,16,17) # specifying columns corresponding to genes of interest
conditions=c(1:4) # specifying columns containing factors

# calculating molecule counts and reformatting:
dd=cq2counts(data=coral.stress,genecols=genecolumns,
condcols=conditions,effic=amp.eff,Cq1=37)

# fitting the model (must include random="sample", pr=TRUE options)
mm=mcmc.qpcr(
fixed="condition",
data=dd,
controls=c("nd5","rpl11"),
nitt=4000,
pr=TRUE,
random="sample"
)

# extracting model predictions
pp=getNormalizedData(mm,dd)

# here is the normalized data:
pp$normData

# and here are the corresponding conditions:
pp$conditions

# putting them together for plotting:
ppcombo=cbind(stack(pp$normData),rep(pp$conditions))
names(ppcombo)[1:2]=c("expression","gene")

# plotting boxplots of normalized data:
ggplot(ppcombo,aes(condition,expression,colour=timepoint))+
geom_boxplot()+
facet_wrap(~gene,scales="free")+
theme_bw()
```

HPDplot *Plotting fixed effects for all genes for a single combination of factors*

Description

Calculates and plots posterior means with 95% credible intervals for specified fixed effects (or their combination) for all genes

Usage

```
HPDplot(model, factors, factors2 = NULL, ylimits = NULL,
hpdtype = "w", inverse = FALSE, jitter = 0, plot = TRUE, grid = TRUE,
zero = TRUE, ...)
```

Arguments

`model`	The output of mcmc.qpcr function.
`factors`	A vector of names of fixed effects of interest; see details.
`factors2`	A second vector of fixed effect names to be subtracted from the first; see details.
`ylimits`	Y-limits for the plot such as c(-3,6); autoscale by default.
`hpdtype`	Specify hpdtype="l" to plot the upper and lower 95% credible limits as a continuous dashed line across all genes. This is useful to compare credible intervals among several models on the same plot. By default (hpdtype="w") the limits are plotted as whiskers around each point.
`inverse`	Plot the inverse of the result.

`jitter`	For hpdtype="w", shifts the plotted values and whiskers by the specified distance along the x axis (reasonable jitter values are 0.15 or -0.15, for example). This helps plot several results (different models or factor combinations) on the same plot (use HPDpoints to add to existing plot)
`plot`	if plot = FALSE the function returns a table of calculated posterior modes, means, upper and lower 95% credible limits (all on log(2) scale), and two types of p-values: derived from Bayesian z-scores, and derived directly from MCMC sample. All such outputs for a given experiment should be concatenated with rbind and processed by padj.qpcr() function to adjust the p-values for multiple comparisons (disregarding the entries corresponding to control genes)
`grid`	Whether to draw vertical grid lines to separate genes.
`zero`	Whether to draw a horizontal line at 0.
`...`	Various plot() options; such as col (color of lines and symbols), pch (type of symbol), main (plot title) etc.

Details

Use summary(MCMCglmm object) first to see what fixed effect names are actually used in the output. For example, if summary shows:

gene1:conditionheat
gene2:conditionheat
....
gene1:timepointtwo
gene2:timepointtwo
....
gene1:conditionheat:timepointtwo
gene2:conditionheat:timepointtwo

, it is possible to specify factors="conditionheat" to plot only the effects of the heat.

If a vector of several fixed effect names is given, for example: factors=c("timepointtwo","treatmentheat:timepointtwo") the function will plot the posterior mean and credible interval for the sum of these effects.

If a second vector is also given, for example,
factors=c("f1","f2"), factors2=c("f3","f4")
the function will plot the difference between the sums of these two groups of factors. This is useful for pairwise analysis of differences in complicated designs.

Value

A plot or a table (plot = F).

Use the function HPDpoints() if you need to add graphs to already existing plot.

Author(s)

Mikhail V. Matz, UT Austin <matz@utexas.edu>

References

Matz MV, Wright RM, Scott JG (2013) No Control Genes Required: Bayesian Analysis of qRT-PCR Data. PLoS ONE 8(8): e71448. doi:10.1371/journal.pone.0071448

Examples

```
# loading Cq data and amplification efficiencies
data(coral.stress)
data(amp.eff)
# extracting a subset of data
cs.short=subset(coral.stress, timepoint=="one")

genecolumns=c(5,6,16,17) # specifying columns corresponding to genes of interest
conditions=c(1:4) # specifying columns containing factors

# calculating molecule counts and reformatting:
dd=cq2counts(data=cs.short,genecols=genecolumns,
condcols=conditions,effic=amp.eff,Cq1=37)

# fitting the model
mm=mcmc.qpcr(
fixed="condition",
data=dd,
controls=c("nd5","rpl11"),
nitt=3000,burnin=2000 # remove this line when analyzing real data!
)

# plotting log2(fold change) in response to heat stress for all genes
HPDplot(model=mm,factors="conditionheat",main="response to heat stress")
```

`HPDplotBygene` *Plots qPCR analysis results for individual genes.*

Description

For a particular gene, plots model-predicted values (and credible intervals) for a series of specified fixed effect combinations ("conditions").

Usage

```
HPDplotBygene(model, gene, conditions, pval = "mcmc", newplot = T,
ylimits = NULL, inverse = F, jitter = 0, plot = T, yscale = "log2",
interval = "ci", grid = F, zero = F, ...)
```

Arguments

`model`	model object produced by MCMC.qpcr()
`gene`	name of the gene to plot
`conditions`	A list naming the conditions to plot and defining them as combination of fixed effects (See example below). '0' denotes gene-specific intercept.
`pval`	Type of p-value to report: 'mcmc' - MCMC-based (default), 'z' - based on Bayesian z-score. Use 'z' to approximate p-values lower than 2/[MCMC sample size] (with default MCMC.qpcr settings, this comes to 0.002)
`newplot`	When TRUE, a new plot should be created. When FALSE, or a graph will be added to an existing plot.
`ylimits`	Y-limits for the plot such as c(-3,6); autoscale by default.
`inverse`	When TRUE, the inverse of the data will be plotted.
`jitter`	Shifts the plotted values and whiskers by the specified distance along the x axis (reasonable jitter values are 0.15 or -0.15, for example). This helps plot several graphs on the same plot without overlapping.
`plot`	When FALSE, no plot will be generated; the function will just list mean pairwise differences and p-values.
`yscale`	Scale on which to represent the data. In all mcmc.qpcr models the model scale is natural logarithm, which I prefer to translate into log2 or log10 (if the differences are orders of magnitude) for better human readability. The default is 'log2'; other options are 'log10' and 'native' (no rescaling of the model data). There is also a beta-option 'proportion', which is not useful for qPCR. It was added to cannibalize HPDplotBygene function for plotting results of the model operating with arcsin-square root transfromed proportions. With yscale="proportions", the plot will be on the original proportion scale but the tukey-like differences will still be reported on the asin(sqrt()) transformed scale.
`interval`	'ci' (default) will plot 95% credible limits of the posterior distribution, 'sd' will plot the mean plus/minus one standard deviation of the posterior.
`grid`	When TRUE, a vertical grid separating conditions will be added
`zero`	When TRUE, a y=0 line will be added.
`...`	Various plot() options.

Value

Generates a point-whiskers plot, lists pairwise mean differenes between all conditions, calculates and lists pairwise p-values (not corrected for multiple testing).

Author(s)

Mikhal V. Matz, UT Austin, matz@utexas.edu

References

Matz MV, Wright RM, Scott JG (2013) No Control Genes Required: Bayesian Analysis of qRT-PCR Data. PLoS ONE 8(8): e71448. doi:10.1371/journal.pone.0071448

Examples

```
# loading Cq data and amplification efficiencies
data(coral.stress)
data(amp.eff)
# extracting a subset of data
cs.short=subset(coral.stress, timepoint=="one")

genecolumns=c(5,6,16,17) # specifying columns corresponding to genes of interest
conditions=c(1:4) # specifying columns containing factors

# calculating molecule counts and reformatting:
dd=cq2counts(data=cs.short,genecols=genecolumns,
condcols=conditions,effic=amp.eff,Cq1=37)

# fitting the model
mm=mcmc.qpcr(
fixed="condition",
data=dd,
controls=c("nd5","rpl11"),
nitt=3000,burnin=2000 # remove this line when analyzing real data!
)

# plotting abundances of individual genes across all conditions
# step 1: defining conditions
cds=list(
  control=list(factors=0), # gene-specific intercept
  stress=list(factors=c(0,"conditionheat")) # multiple effects will be summed up
  )

# step 2: plotting gene after gene on the same panel
HPDplotBygene(model=mm,gene="actin",conditions=cds,col="cyan3",
pch=17,jitter=-0.1,ylim=c(-3.5,15),pval="z")
HPDplotBygene(model=mm,gene="hsp16",conditions=cds,
newplot=FALSE,col="coral",pch=19,jitter=0.1,pval="z")

# step 3: adding legend
legend(0.5,10,"actin",lty=1,col="cyan3",pch=17,bty="n")
legend(0.5,7,"hsp16",lty=1,col="coral",pch=19,bty="n")
```

`HPDplotBygeneBygroup` *Plots qPCR analysis results for individual genes*

Description

For a specified gene, makes overlayed plots such as produced by HPDplotBygene()

Usage

```
HPDplotBygeneBygroup(model, gene, group1, group2, group3 = NULL,
interval = "ci", colors = c("coral", "cyan3", "grey50"),
symbols = c(19, 17, 15), jitter = 0.16, yscale = "log2", ...)
```

Arguments

`model`	model object produced by mcmc.qpcr()
`gene`	name of the gene to plot
`group1`	Combination of factors defining the first group (see HPDplotBygene() for details).
`group2`	Combination of factors defining the second group.
`group3`	(optional) Combination of factors defining the third group.
`interval`	'ci' (default) will plot 95% credible limits of the posterior distribution, 'sd' will plot the mean plus/minus one standard deviation of the posterior.
`colors`	Colors to use for different groups (see ?par -> col).
`symbols`	Symbols to use for different groups (see ?par -> pch).
`jitter`	Jitter distance between groups.
`yscale`	Scale on which to represent the data. In all mcmc.qpcr models the model scale is natural logarithm, which I prefer to translate into log2 or log10 (if the differences are orders of magnitude) for better human readability. The default is 'log2'; other options are 'log10' and 'native' (no rescaling of the model data). There is also a beta-option 'proportion', which is not useful for qPCR. It was added to cannibalize HPDplotBygene function for plotting results of the model operating with arcsin-square root transfromed proportions. With yscale="proportions", the plot will be on the original proportion scale but the tukey-like differences will still be reported on the asin(sqrt()) transformed scale.
`...`	additional parameters for HPDplotBygene() function, such as pval (see HPDplotBygene() help)

Value

Generates a point-whiskers plot, lists pairwise mean differenes between all conditions, calculates and lists pairwise p-values (not corrected for multiple testing).

Author(s)

Mikhal V. Matz, UT Austin <matz@utexas.edu>

References

Matz MV, Wright RM, Scott JG (2013) No Control Genes Required: Bayesian Analysis of qRT-PCR Data. PLoS ONE 8(8): e71448. doi:10.1371/journal.pone.0071448

HPDpoints *HPDplot, HPDpoints*

Description

Calculates and plots posterior means with 95% credible intervals for specified fixed effects (or their combination) for all genes. HPDpoints only adds graphs to an existing plot.

Usage

```
HPDpoints(model, factors, factors2 = NULL, ylimits = NULL,
hpdtype = "w", inverse = F, jitter = 0, ...)
```

Arguments

`model`	The output of mcmc.qpcr function.
`factors`	A vector of names of fixed effects of interest; see details.
`factors2`	A second vector of fixed effect names to be subtracted from the first; see details.
`ylimits`	Y-limits for the plot such as c(-3,6); autoscale by default.
`hpdtype`	Specify hpdtype="l" to plot the upper and lower 95% credible limits as a continuous dashed line across all genes. By default (hpdtype="w") the limits are plotted as whiskers around each point.
`inverse`	Plot the inverse of the result.
`jitter`	For hpdtype="w", shifts the plotted values and whiskers by the specified distance along the x axis (reasonable jitter values are 0.15 or -0.15, for example).
`...`	Various plot() options; such as col (color of lines and symbols), pch (type of symbol), main (plot title) etc.

Details

See details in HPDplot()

Value

A graph added to a plot.

Author(s)

Mikhail V. Matz, UT Austin <matz@utexas.edu>

References

Matz MV, Wright RM, Scott JG (2013) No Control Genes Required: Bayesian Analysis of qRT-PCR Data. PLoS ONE 8(8): e71448. doi:10.1371/journal.pone.0071448

Examples

```
# loading Cq data and amplification efficiencies
data(coral.stress)
data(amp.eff)
# extracting a subset of data
cs.short=subset(coral.stress, timepoint=="one")

genecolumns=c(5,6,16,17) # specifying columns corresponding to genes of interest
conditions=c(1:4) # specifying columns containing factors

# calculating molecule counts and reformatting:
dd=cq2counts(data=cs.short,genecols=genecolumns,
condcols=conditions,effic=amp.eff,Cq1=37)

# fitting the model
mm=mcmc.qpcr(
fixed="condition",
data=dd,
controls=c("nd5","rpl11"),
nitt=4000 # remove this line when analyzing real data!
)

# plotting log2(fold change) in response to heat stress for all genes
HPDplot(model=mm,factors="conditionheat",main="response to heat stress")
```

`HPDsummary` *Summarizes and plots results of mcmc.qpcr function series.*

Description

Calculates abundances of each gene across factor combinations; calculates pairwise differences between all factor combinations and their significances for each gene; plots results as bar or line graphs with credible intervals (ggplot2) NOTE: only works for experiments involving a single multi-level fixed factor or two fully crossed multi-level fixed factors.

Usage

```
HPDsummary(model, data, xgroup=NULL,genes = NA, relative = FALSE,
log.base = 2, summ.plot = TRUE, ptype="z", ...)
```

Arguments

`model`	Model generated by mcmc.qpcr(),mcmc.qpcr.lognormal() or mcmc.qpcr.classic()
`data`	Dataset used to build the model (returned by cq2counts() or cq2log())
`xgroup`	The factor to form the x-axis of the plot. By default the first factor in the model will be used.

`genes`	A vector of gene names to summarize and plot. If left unspecified, all genes will be summarized.
`relative`	Whether to plot absolute transcript abundances (relative = FALSE) or fold-changes relative to the sample that is considered to be "global control" (relative = TRUE). The "global control" is the combination of factors that served as a reference during model fitting, either because it is alphanumerically first (that happens by default) or because it has been explicitly designated as such using relevel() function (see tutorial).
`log.base`	Base of the logarithm to use.
`summ.plot`	By default, the function generates a summary plot, which is a line-points-95% credible intervals plot of log(absolute abundances) with 'relative=FALSE' and a more typical bar graph of log(fold change relative to the control), again with 95% credible intervals, with 'relative=TRUE'. Specify 'summ.plot=FALSE' if you don't want the summary plot.
`ptype`	Which type of p-values to use. By default p-values based on the Bayesian z-score are used. Specify 'ptype="mcmc"' to output more conventional p-values based on MCMC sampling (these will be limited on the lower end by the size of MCMC sample).
`...`	Additional options for summaryPlot() function. Among those, 'x.order' can be a vector specifying the order of factor levels on the x-axis.

Value

A list of three items:

`summary`	Summary table containing calculated abundances, their SD and 95% credible limits
`geneWise`	A series of matrices listing pairwise differences between factor combinations (upper triangle) and corresponding p-values (lower triangle)
`ggPlot`	the ggplot2 object for plotting. See http://docs.ggplot2.org/0.9.2.1/theme.html for ways to modify it, such as add text, rotate labels, change fonts, etc.

Author(s)

Mikhail V. Matz, University of Texas at Austin <matz@utexas.edu>

References

Matz MV, Wright RM, Scott JG (2013) No Control Genes Required: Bayesian Analysis of qRT-PCR Data. PLoS ONE 8(8): e71448. doi:10.1371/journal.pone.0071448

See Also

See function summaryPlot() for plotting the summary table in other ways.

Examples

```
data(beckham.data)
data(beckham.eff)

# analysing the first 5 genes
# (to try it with all 10 genes, change the line below to gcol=4:13)
gcol=4:8
ccol=1:3 # columns containing experimental conditions

# recalculating into molecule counts, reformatting
qs=cq2counts(data=beckham.data,genecols=gcol,
condcols=ccol,effic=beckham.eff,Cq1=37)

# creating a single factor, 'treatment.time', out of 'tr' and 'time'
qs$treatment.time=as.factor(paste(qs$tr,qs$time,sep="."))

# fitting a naive model
naive=mcmc.qpcr(
fixed="treatment.time",
data=qs,
nitt=3000,burnin=2000 # remove this line in actual analysis!
)

#summary plot of inferred abundances
# s1=HPDsummary(model=naive,data=qs)

#summary plot of fold-changes relative to the global control
s0=HPDsummary(model=naive,data=qs,relative=TRUE)

# pairwise differences and their significances for each gene:
s0$geneWise
```

`mcmc.converge.check` *MCMC diagnostic plots*

Description

A wrapper function for plot.MCMCglmm to plot diagnostic convergence plots for selected fixed effects

Usage

```
mcmc.converge.check(model, factors, ...)
```

Arguments

`model`	output of mcmc.qpcr (or any MCMCglmm class object)

`factors`	A vector of names of fixed effects of interest; see details in HPDplot help page.
`...`	other options to pass to plot.MCMCglmm

Value

A series of plots for each gene-specific fixed effect.

The MCMC trace plot is on the left, to see if there is convergence (lack of systematic trend) and no autocorrelation (no low-frequency waves). If lack of convergence is suspected, try increasing number of iterations and burnin by specifying, for example, nitt=50000, burnin=5000, as additional options for mcmc.qpcr. If autocorrelation is present, increase thinning interval by specifying thin=20 in mcmc.qpcr (you might wish to increase the number of iterations, nitt, to keep the size of MCMC sample the same)

The right plot is posterior density distribution.

Author(s)

Mikhail V. Matz, UT Austin

References

Matz MV, Wright RM, Scott JG (2013) No Control Genes Required: Bayesian Analysis of qRT-PCR Data. PLoS ONE 8(8): e71448. doi:10.1371/journal.pone.0071448

`mcmc.pval` *calculates p-value based on Bayesian z-score or MCMC sampling*

Description

Accessory function to HPD... function series

Usage

```
mcmc.pval(dat, testlim = 0, sided = 2, ptype="z")
```

Arguments

`dat`	a table of MCMC samples
`testlim`	positive value to test whether the posterior crosses it (for variance components)
`sided`	sidedness of the test
`ptype`	'z' for z-score based p-values, 'mcmc' for standard sampling based p-values

Value

A vector of calculated Bayesian p-values. For standard 'mcmc' p=values, the minimal possible value is 1/(MCMC sample size). z-score based p-values are useful to approximate very low p-values with limited MCMC sample sizes.

Author(s)

Mikhail Matz, UT Austin <matz@utexas.edu>

---

mcmc.qpcr *Analyzes qRT-PCR data using generalized linear mixed model*

Description

Wrapper function for MCMCglmm by Jarrod Hadfield, designed for qRT-PCR data analysis.

Usage

```
mcmc.qpcr(fixed=NULL, globalFixed=NULL, random = NULL, globalRandom=NULL, data,
controls = NULL, normalize=FALSE, include = NULL, m.fix = 1.2, v.fix = NULL,
geneSpecRes=TRUE, Covar=FALSE, vprior="uninf", ...)
```

Arguments

`fixed`	desired combination of fixed effects, as a text string. Do not use "*" symbol, list it fully, such as: 'factor1+factor2+factor1:factor2'.
`globalFixed`	Vector of fixed covariates (categorical or continuous) that are expected to affect all genes in the sample in the same way. These would be typically related to quality and/or quantity of RNA, such as RIN value.
`random`	A vector of names for gene-specific scalar random effects, such as 'c("effect1","effect2")'.
`globalRandom`	Random covariates (categorical only) affecting all genes, similar to globalFixed.
`data`	output of the cq2counts() function
`controls`	Vector of control gene names. These will be pushed to the back of the gene list during model fitting, in the reverse order. Controls are NOT NECESSARY for the analysis.
`normalize`	If controls are specified, requiring 'normalize=TRUE' will perform "soft normalization": the geometric mean of control genes will be used to infer global effects (common to all genes) across conditions; these will be subtracted when summarizing the data with HPDsummary(). Use this option when template abundances are correlated with conditions. Note that this is different from normalizing data within each sample, as per Vandesompele et al 2002; this would be "hard normalization" (use mcmc.qpcr.classic() if you want it).
`include`	How many of the control genes ('controls') to actually incorporate as priors during model fitting. If left unspecified, all the 'controls' will be used. If 'include=0', the model will be fitted without using any of the control genes as priors. If 'include' equals some number less than the number of 'controls', only the first 'include' of them will be used as priors. In all these cases, the 'controls' will appear in the same order in the output, in the end of the gene list rather than according to their alphabetical position among other genes. This is useful when visually comparing the results of models fitted with different number of control genes, using HPDplot and HPDpoints functions.

`m.fix`	Allowed average fold-change of the control genes in response to any fixed factor combination.
`v.fix`	Allowed residual fold-change deviation of the control genes. Applies to the residual variation term.
`geneSpecRes`	Whether the model should include gene-specific residuals. Keep it TRUE unless the model fails to converge.
`Covar`	Whether the random effects should be fitted with covariances. This option might require setting vprior="iw" or vprior="iw01" (see below)
`vprior`	The default prior is an uninformative inverse Wishart with assumed variance (V) at 1 and the degree of belief parameter (nu) at 0. With 'prior="iw"' and 'prior="iw01"' nu is equal [number of genes]-0.998, resulting in a weakly informative prior that is commonly used in this type of inference. vprior="iw" will assume large prior variance (1), vprior="iw01" will assume small prior variance (0.1).
`...`	other options for MCMCglmm function, such as nitt (number of iterations), thin (tinning interval), and burnin (number of initial iterations to disregard). For a more precise inference specify 'nitt=45000, thin=20, burnin=5000'. See MCMCglmm documentation for more details.

Details

This function constructs priors and runs an MCMC chain to fit a Poisson-lognormal generalized linear mixed model to the counts data.

The fixed effects for the model by default assume a gene-specific intercept, and gene-specific effect for each of the listed fixed factors.

The user-specified random effects are all assumed to be gene-specific with no covariances.

There is also a universal random factor: the scalar random effect of sample, which accounts for the unequal template loading.

Residual variances are assumed to be gene-specific with no covariances, with weakly informative inverse Wishart prior (variance=1, nu=[number of genes]-0.998).

The priors for fixed effects are diffuse gaussians with a mean at 0 and very large variances (1e+8), except for control genes, for which the prior variances are defined by the m.fix parameter. For the gene-specific random effects and residual variation, non-informative priors are used to achieve results equivalent to ML estimation. For control genes, when v.fix parameter is specified, it will be used to restrict residial variance.

Both m.fix and v.fix parameters should be specified as allowed average fold-change, so the lowest they can go is 1 (no variation).

All control genes share the same m.fix and v.fix parameters.

Value

An MCMCglmm object. HPDsummary() function within this package summaizes these data, calculates all gene-wise credible intervals and p-values, and plots the results either as line-point-whiskers graph or a bar-whiskers graph using ggplot2 functions.

HPDsummary() only works for experiments with a single multilevel factor or two fully crossed multilevel factors. Use finctions HPDplot(), HPDplotBygene() and HPDplotBygeneBygroup() to summarize and plot more complicated designs.

For more useful operations on MCMCglmm objects, such as posterior.mode(), HPDinterval(), and plot(), see documentation for MCMCglmm package.

Author(s)

Mikhail V. Matz, University of Texas at Austin <matz@utexas.edu>

References

Matz MV, Wright RM, Scott JG (2013) No Control Genes Required: Bayesian Analysis of qRT-PCR Data. PLoS ONE 8(8): e71448. doi:10.1371/journal.pone.0071448

Examples

```
data(beckham.data)
data(beckham.eff)

# analysing the first 5 genes
# (to try it with all 10 genes, change the line below to gcol=4:13)
gcol=4:8
ccol=1:3 # columns containing experimental conditions

# recalculating into molecule counts, reformatting
qs=cq2counts(data=beckham.data,genecols=gcol,
condcols=ccol,effic=beckham.eff,Cq1=37)

# creating a single factor, 'treatment.time', out of 'tr' and 'time'
qs$treatment.time=as.factor(paste(qs$tr,qs$time,sep="."))

# fitting a naive model
naive=mcmc.qpcr(
fixed="treatment.time",
data=qs,
nitt=3000,burnin=2000 # remove this line in actual analysis!
)

#summary plot of inferred abundances
#s1=HPDsummary(model=naive,data=qs)

#summary plot of fold-changes relative to the global control
s0=HPDsummary(model=naive,data=qs,relative=TRUE)

# pairwise differences and their significances for each gene:
s0$geneWise
```

mcmc.qpcr.classic — *Analyzes qRT-PCR data using "classic" model, based on multigene normalization.*

Description

Normalizes the data using specified control genes, fits a single model to estimate changes at all genes. Use for datasets with not too many Cq values above 30.

Usage

```
mcmc.qpcr.classic(fixed = NULL, globalFixed = NULL, random = NULL, globalRandom = NULL,
data, controls, genebysample = TRUE, geneSpecRes=FALSE, center = TRUE, ...)
```

Arguments

`fixed`	desired combination of fixed effects, as a text string. Do not use "*" symbol, list it fully, such as: 'factor1+factor2+factor1:factor2'.
`globalFixed`	Vector of fixed covariates (categorical or continuous) that are expected to affect all genes in the sample in the same way. These would be typically related to quality and/or quantity of RNA, such as RIN value.
`random`	A vector of names for gene-specific scalar random effects, such as 'c("effect1","effect2")'.
`globalRandom`	Random covariates (categorical only) affecting all genes, similar to globalFixed.
`data`	output of the cq2log() function
`controls`	Vector of control gene names. These will be pushed to the back of the gene list during model fitting, in the reverse order.
`genebysample`	Whether random gene by sample interactions should be modeled as an additional random effect. If the model fails to converge, specify 'genebysample=F'.
`geneSpecRes`	Whether the model should include gene-specific residuals. This was the default in MCMC.qpcr v.1.0; now it is switched off since it does not have any visible effect on the results (as long as genebysample=TRUE), and only makes the model converge slower or fail to converge.
`center`	Whether to center the normalized log-transformed Cq values arond the mean for each gene. Centering does not affect the inference; it only makes the plots of the results more comprehensible (in my opinion).
`...`	other options for MCMCglmm function, such as nitt (number of iterations), thin (tinning interval), and burnin (number of initial iterations to disregard). For a more precise inference (but longer runs) specify 'nitt=45000, thin=20, burnin=5000'. See MCMCglmm documentation for more details.

Details

This function takes an as input the log-transformed relative abundance values, performs multigene normalization as per Vandesompele et al 2002, and runs an MCMC chain to fit a lognormal linear mixed model to estimate gene expression changes jointly at all genes. It is very powerful, as long as the average stability of control genes can be trusted.

This function requires data prepared by cq2log function and must have the control genes specified; otherwise the arguments and syntax are similar to the mcmc.qpcr function.

Value

An MCMCglmm object. See mcmc.qpcr function for details and examples.

Author(s)

Mikhail V. Matz, University of Texas at Austin

References

Matz MV, Wright RM, Scott JG (2013) No Control Genes Required: Bayesian Analysis of qRT-PCR Data. PLoS ONE 8(8): e71448. doi:10.1371/journal.pone.0071448

`mcmc.qpcr.lognormal` *Fits a lognormal linear mixed model to qRT-PCR data.*

Description

Use in combination with cq2log(), on datasets without too many Cq values above 30.

Usage

```
mcmc.qpcr.lognormal(fixed=NULL, globalFixed=NULL, random = NULL, globalRandom=NULL, data,
controls = NULL, normalize=FALSE, include = NULL, m.fix = 1.2, v.fix = NULL,
geneSpecRes=TRUE, Covar=FALSE, vprior="uninf", ...)
```

Arguments

`fixed`	desired combination of fixed effects, as a text string. Do not use "*" symbol, list it fully, such as: 'factor1+factor2+factor1:factor2'.
`globalFixed`	Vector of fixed covariates (categorical or continuous) that are expected to affect all genes in the sample in the same way. These would be typically related to quality and/or quantity of RNA, such as RIN value.
`random`	A vector of names for gene-specific scalar random effects, such as 'c("effect1","effect2")'.
`globalRandom`	Random covariates (categorical only) affecting all genes, similar to globalFixed.
`data`	output of the cq2counts() function

`controls`	Vector of control gene names. These will be pushed to the back of the gene list during model fitting, in the reverse order. Controls are NOT NECESSARY for the analysis.
`normalize`	If controls are specified, requiring 'normalize=TRUE' will perform "soft normalization": the geometric mean of control genes will be used to infer global effects (common to all genes) across conditions; these will be subtracted when summarizing the data with HPDsummary(). Use this option when template abundances are correlated with conditions. Note that this is different from normalizing data within each sample, as per Vandesompele et al 2002; this would be "hard normalization" (use mcmc.qpcr.classic() if you want it).
`include`	How many of the control genes ('controls') to actually incorporate as priors during model fitting. If left unspecified, all the 'controls' will be used. If 'include=0', the model will be fitted without using any of the control genes as priors. If 'include' equals some number less than the number of 'controls', only the first 'include' of them will be used as priors. In all these cases, the 'controls' will appear in the same order in the output, in the end of the gene list rather than according to their alphabetical position among other genes. This is useful when visually comparing the results of models fitted with different number of control genes, using HPDplot and HPDpoints functions.
`m.fix`	Allowed average fold-change of the control genes in response to any fixed factor combination.
`v.fix`	Allowed residual fold-change deviation of the control genes. Applies to the residual variation term.
`geneSpecRes`	Whether the model should include gene-specific residuals. Keep it TRUE unless the model fails to converge.
`Covar`	Whether the random effects should be fitted with covariances. This option might require setting vprior="iw" or vprior="iw01" (see below)
`vprior`	The default prior is an uninformative inverse Wishart with assumed variance (V) at 1 and the degree of belief parameter (nu) at 0. With 'prior="iw"' and 'prior="iw01"' nu is equal [number of genes]-0.998, resulting in a weakly informative prior that is commonly used in this type of inference. vprior="iw" will assume large prior variance (1), vprior="iw01" will assume small prior variance (0.1).
`...`	other options for MCMCglmm function, such as nitt (number of iterations), thin (tinning interval), and burnin (number of initial iterations to disregard). For a more precise inference specify 'nitt=45000, thin=20, burnin=5000'. See MCMCglmm documentation for more details.

Details

This function constructs priors and runs an MCMC chain to fit a lognormal linear mixed model to the log-transformed relative abundances data. The data for this function must be prepared by cq2log instead of cq2counts function; otherwise, the arguments and syntax are identical to mcmc.qpcr function.

Value

An MCMCglmm object. See mcmc.qpcr function for details and examples.

Author(s)

Mikhail V. Matz, University of Texas at Austin

References

Matz MV, Wright RM, Scott JG (2013) No Control Genes Required: Bayesian Analysis of qRT-PCR Data. PLoS ONE 8(8): e71448. doi:10.1371/journal.pone.0071448

`normalize.qpcr` *Internal function called by mcmc.qpcr.classic*

Description

Performs multigene normalization as per Vandesompele et al 2002, centers the resulting gene expression values around the gene's means

Usage

```
normalize.qpcr(data, controls, center = T)
```

Arguments

`data`	The output of cq2log()
`controls`	Vector of control genes, such as 'controls=c("eif3h","nd5","rpl11")'
`center`	Whether to perform centering around mean or not.

Value

normalized dataset

Author(s)

Mikhail Matz, UT Austin

References

Matz MV, Wright RM, Scott JG (2013) No Control Genes Required: Bayesian Analysis of qRT-PCR Data. PLoS ONE 8(8): e71448. doi:10.1371/journal.pone.0071448

`outlierSamples` *detects outlier samples in qPCR data*

Description

reports samples that have too little starting material relative to others (by default, less by two standard deviations)

Usage

```
outlierSamples(model, data, z.cutoff = -2)
```

Arguments

`model`	qPCR model: the output of mcmc.qpcr or mcmc.qpcr.lognormal function fitted with pr=TRUE option
`data`	The dataset that was analysed to generate the model (output of cq2counts or cq2log functions)
`z.cutoff`	z-score cutoff to report an outlier sample.

Value

A vector containing outlier sample names.

Author(s)

Mikhail V. Matz, University of Texas at Austin <matz@utexas.edu>

References

Matz MV, Wright RM, Scott JG (2013) No Control Genes Required: Bayesian Analysis of qRT-PCR Data. PLoS ONE 8(8): e71448. doi:10.1371/journal.pone.0071448

Examples

```
# loading Cq data and amplification efficiencies
data(coral.stress)
data(amp.eff)
# extracting a subset of data
cs.short=subset(coral.stress, timepoint=="one")

genecolumns=c(5,6,16,17) # specifying columns corresponding to genes of interest
conditions=c(1:4) # specifying columns containing factors

# calculating molecule counts and reformatting:
dd=cq2counts(data=cs.short,genecols=genecolumns,
condcols=conditions,effic=amp.eff,Cq1=37)
```

```
# fitting the model
mm=mcmc.qpcr(
fixed="condition",
data=dd,
controls=c("nd5","rpl11"),
nitt=4000, # remove this line when analyzing real data!
pr=TRUE
)

# detecting outliers
outliers=outlierSamples(mm,dd)

# removing outliers
dd=dd[!(dd$sample %in% outliers),]
```

---

padj.hpdsummary — *Adjusts p-values within an HPDsummary() object for multiple comparisons*

Description

Replaces raw p-values in an object returned by HPDsummary() by adjusted p-values corrected for multiple comparisons. Disregards the entries corresponding to control genes.

Usage

```
padj.hpdsummary(hpdsumm, controls = NULL, method = "BH")
```

Arguments

`hpdsumm`	Output of HPDsummary()
`controls`	A vector of control gene names
`method`	p-value correction method (see function p.adjust), default is Benjamini-Hochberg

Value

HPDsummary object with original p-values replaced by the corrected ones.

Author(s)

Mikhail V. Matz, matz@mail.utexas.edu

References

Matz MV, Wright RM, Scott JG (2013) No Control Genes Required: Bayesian Analysis of qRT-PCR Data. PLoS ONE 8(8): e71448. doi:10.1371/journal.pone.0071448

`padj.qpcr` *Calculates adjusted p-values corrected for multiple comparisons*

Description

Takes the output of HPDplot(...,plot=FALSE), adds two columns of adjusted p-values (z and mcmc) Disregards the entries corresponding to control genes

Usage

```
padj.qpcr(data, controls = NULL, method = "BH")
```

Arguments

`data`	Output of HPDplot(...,plot=FALSE); may be several several such outputs concatenated with rbind
`controls`	A vector of control gene names
`method`	p-value correction method (see function p.adjust), default is Benjamini-Hochberg

Value

The dataframe derived from the input, with added columns "padj.z" and "padj.mcmc". See tutorial for examples.

Author(s)

Mikhail V. Matz, matz@mail.utexas.edu

References

Matz MV, Wright RM, Scott JG (2013) No Control Genes Required: Bayesian Analysis of qRT-PCR Data. PLoS ONE 8(8): e71448. doi:10.1371/journal.pone.0071448

`PrimEff` *Determines qPCR amplification efficiencies from dilution series*

Description

Runs linear regression on Cq versus log2(RNA concentration), plots graph, reports slope (ideally should be -1), and efficiency (with 95 percent credible limits)

Usage

```
PrimEff(data, plot = TRUE)
```

Arguments

`data`	a dataframe containing three columns. First is RNA concentration. This could be absolute as well as relative concentration (1/dilution factor). Second is the Cq value. Third is gene name. Replicate the same name across all the corresponding RNA concentrations. The dataframe may contain data for multiple genes.
`plot`	set plot=FALSE if the plot is not required

Details

Run with at least 8 2-fold dilutions per gene

Value

Plots the regression and under it, the values of slope and efficiency (plus and minus one SD). The dataframe may contain data for multiple genes, which will all be plotted together (so the reasonable limit is something like 25 genes)

Also returns a dataframe with columns: gene, efficiency, plus one SD, minus one SD, and intercept.

Author(s)

Mikhail V. Matz, UT Austin <matz@utexas.edu>

References

Matz MV, Wright RM, Scott JG (2013) No Control Genes Required: Bayesian Analysis of qRT-PCR Data. PLoS ONE 8(8): e71448. doi:10.1371/journal.pone.0071448

Examples

```
data(dilutions)
PrimEff(dilutions)
```

`softNorm` *Accessory function to mcmc.qpcr() to perform soft normalization*

Description

adds a fake NORM gene with counts equal to geometric mean of conrtol genes, removes control genes from the dataset

Usage

```
softNorm(data,controls)
```

Arguments

`data`	The dataset; output of cq2counts()
`controls`	Vector fo control gene names

Value

A modified dataset with NORM gene set as the reference, and no control genes

Author(s)

Mikhail Matz, UT Austin <matz@utexas.edu>

`summaryPlot` *Wrapper function for ggplot2 to make bar and line graphs of mcmc.qpcr() results*

Description

This function is called automatically by HPDsummary() and also can be used separately to plot the results produced by HPDsummary() with more plotting options

Usage

```
summaryPlot(data, xgroup, facet = NA, type = "bar", x.order = NA,
whiskers = "ci", genes = NA, log.base=2)
```

Arguments

`data`	A summary table generated by HPDplot(), it is the first element in the returned list.
`xgroup`	Which factor will be used to form the x axis (for 2-way designs).
`facet`	The factor by which the plot will be split into facets (for 2-way designs).
`type`	Two types are supported: "bar" and "line" ("line" also has points). "bar" is more useful to plot fold-changes returned when HPDsummary() is run with the option 'relative=TRUE'. "line" is better for plotting actual inferred transcript abundances across factor levels; it is particularly good for time courses and other cases when multiple factor levels must be compared to each other. "bar" is good to plot log(fold-changes) relative to global control.
`x.order`	A vector giving the order of factor levels on the x-axis. If unspecified, an alphanumeric order will be used.
`whiskers`	The interval indicated by the whiskers. Default is "ci", the 95% credible interval; another option is "sd" - standard deviation of the posterior.
`genes`	Vector of gene names to plot. By default, all genes in the summary will be plotted.
`log.base`	Base of the logarithm to indicate on y-axis (does not affect plotting).

Details

The function invokes ggplot() functon from the ggplot2 package to plot the results either as a single panel (one-way designs) or a multi-panel (2-way designs, one panel per level of the factor specified by 'facet' argument).

Value

A ggplot object. See http://docs.ggplot2.org/0.9.2.1/theme.html for ways to modify it, such as add text, rotate labels, change fonts, etc.

Author(s)

Mikhail V. Matz, University of Texas at Austin <matz@utexas.edu>

References

Matz MV, Wright RM, Scott JG (2013) No Control Genes Required: Bayesian Analysis of qRT-PCR Data. PLoS ONE 8(8): e71448. doi:10.1371/journal.pone.0071448

`trellisByGene` *For two-way designs, plots mcmc.qpcr model predictions gene by gene*

Description

For each gene, plots model-predicted values and 95% credible intervals.

Usage

```
trellisByGene(modelSummary,xFactor,groupFactor,
nrow=2,lineWidth=0.4,whiskerWidth=0.2,pointSize=2.5,
facetScales="free_y",ylab="log(abundance)",
legendPos="bottom",posDodge=0.3)
```

Arguments

`modelSummary`	two-way design model summary produced by HPDsummary()
`xFactor`	factor to form the x-axis
`groupFactor`	factor to form separate lines on the plot
`nrow`	number of rows in the resulting trellis plot
`lineWidth`	line width, passed as 'lwd' to geom_errorbar function (ggplot2)
`whiskerWidth`	width of the line denoting 95% CI margin, passed as 'width' to geom_errorbar function (ggplot2)
`pointSize`	passed as 'size' to geom_point function of ggplot2
`facetScales`	passed as 'scales' to facet_wrap function of ggplot2
`ylab`	y-axis label
`legendPos`	passed as 'legend.position' to theme function of ggplot2
`posDodge`	position dodge, increase for more jitter

Value

A ggplot2 type object

Author(s)

Mikhal V. Matz, UT Austin, matz@utexas.edu

References

Matz MV, Wright RM, Scott JG (2013) No Control Genes Required: Bayesian Analysis of qRT-PCR Data. PLoS ONE 8(8): e71448. doi:10.1371/journal.pone.0071448

Examples

```
# loading Cq data and amplification efficiencies
data(coral.stress)
data(amp.eff)

genecolumns=c(5,6,16,17) # specifying columns corresponding to genes of interest
conditions=c(1:4) # specifying columns containing factors

# calculating molecule counts and reformatting:
dd=cq2counts(data=coral.stress,genecols=genecolumns,
condcols=conditions,effic=amp.eff,Cq1=37)

# fitting the 2-way model
mm=mcmc.qpcr(
fixed="condition+timepoint+condition:timepoint",
data=dd,
nitt=4000 # remark this line to analyze real data!
)

# summarizing results
ss=HPDsummary(mm,data=dd,summ.plot=FALSE)

# plotting predicted means and 95% CIs gene by gene
trellisByGene(ss,xFactor="condition",groupFactor="timepoint")
```

Index