

Package ‘qpcR’

November 6, 2009

Type Package

LazyLoad yes

LazyData yes

Title Modelling and analysis of real-time PCR data

Version 1.2-3

Date 2009-09-11

Author Andrej-Nikolai Spiess <a.spiess@uke.uni-hamburg.de>, Christian Ritz
<ritz@kvl.dk>

Maintainer Andrej-Nikolai Spiess <a.spiess@uke.uni-hamburg.de>

Description Model fitting, optimal model selection and calculation of various features that are essential in the analysis of quantitative real-time polymerase chain reaction (qPCR).

License GPL (>= 2)

Depends R (>= 2.0.0), MASS, rgenoud, gplots, gtools, minpack.lm

Repository CRAN

Date/Publication 2009-11-06 10:18:03

R topics documented:

| | |
|--------------------------|----|
| AICc | 3 |
| akaike.weights | 4 |
| batchstat | 5 |
| batschetal | 6 |
| BIC | 7 |
| calib | 8 |
| calib2 | 11 |
| curvemean | 13 |
| Cy0 | 15 |

| | |
|----------------|----|
| eff | 16 |
| efficiency | 17 |
| evidence | 19 |
| expcomp | 21 |
| expfit | 22 |
| fitprob | 23 |
| guescini1 | 25 |
| guescini2 | 26 |
| LR | 27 |
| maxRatio | 28 |
| meanlist | 29 |
| midpoint | 30 |
| modlist | 31 |
| mselect | 32 |
| outlier | 33 |
| pcrbatch | 35 |
| pcrboot | 37 |
| pcrfit | 39 |
| pcrGOF | 41 |
| pcrimport | 42 |
| pcropt1 | 43 |
| pcropt2 | 45 |
| pcrsim | 46 |
| plot.pcrfit | 49 |
| predict.pcrfit | 50 |
| PRESS | 52 |
| propagate | 53 |
| qpcR_functions | 59 |
| ratiocalc | 61 |
| replist | 66 |
| reps | 67 |
| reps2 | 67 |
| reps3 | 68 |
| resplot | 69 |
| resVar | 70 |
| RMSE | 70 |
| Rsq | 71 |
| Rsq.ad | 72 |
| Rsq.cor | 73 |
| RSS | 74 |
| rutledge | 74 |
| S27 | 75 |
| sliwin | 76 |
| update.pcrfit | 77 |

| | |
|------|---|
| AICc | <i>Akaike's second-order corrected Information Criterion for small sample sizes</i> |
|------|---|

Description

Calculates the second-order corrected Akaike Information Criterion for objects of class `drc`, `lm`, `glm`, `nls` or any other models from which `coefficients` and `residuals` can be extracted. This is a modified version of the original AIC which compensates for bias with small n . As qPCR data usually has $n/\text{par} < 40$ (see original reference), AICc was implemented to correct for this.

Usage

```
AICc(object)
```

Arguments

`object` a fitted model.

Details

Extends the AIC such that

$$AICc = AIC + \frac{2k(k+1)}{n-k-1}$$

with k = number of parameters + 1, and n = number of observations. For large n , AICc converges to AIC.

Value

The second-order corrected AIC value.

Author(s)

Andrej-Nikolai Spiess

References

Sakamoto Y, Ishiguro M, and Kitagawa G (1986). *Akaike Information Criterion Statistics*. D. Reidel Publishing Company.

Hurvich CM & Tsai CL (1989). Regression and Time Series Model Selection in Small Samples. *Biometrika* **76**, 297-307.

See Also

[AIC](#), [logLik](#).

Examples

```
m <- pcrfit(reps, 1, 2, 15)
AICc(m)
```

 akaike.weights

Calculation of Akaike weights/relative likelihoods/delta-AICs

Description

Calculates Akaike weights from a vector of AIC values.

Usage

```
akaike.weights(x)
```

Arguments

`x` a vector containing the AIC values.

Details

Although Akaike's Information Criterion is recognized as a major measure for selecting models, it has one major drawback: The AIC values lack intuitivity despite higher values meaning less goodness-of-fit. For this purpose, Akaike weights come to hand for calculating the weights in a regime of several models. Additional measures can be derived, such as delta-AIC's and relative likelihoods that demonstrate the probability of one model being in favor over the other. This is done by using the following formulae:

delta AICs:

$$\Delta_i(AIC) = AIC_i - \min(AIC)$$

relative likelihood:

$$L \propto \exp \left\{ -\frac{1}{2} \Delta_i(AIC) \right\}$$

Akaike weights:

$$w_i(AIC) = \frac{\exp \left\{ -\frac{1}{2} \Delta_i(AIC) \right\}}{\sum_{k=1}^K \exp \left\{ -\frac{1}{2} \Delta_k(AIC) \right\}}$$

Value

A list containing the following items:

| | |
|-----------------------|---------------------------|
| <code>deltaAIC</code> | the delta-AIC values. |
| <code>rel.LL</code> | the relative likelihoods. |
| <code>weights</code> | the Akaike weights. |

Author(s)

Andrej-Nikolai Spiess

References

Classical literature:

Sakamoto Y, Ishiguro M, and Kitagawa G (1986). Akaike Information Criterion Statistics. D. Reidel Publishing Company.

Burnham KP, Anderson DR. Model selection and inference: a practical information-theoretic approach (2002). Springer Verlag, New York, USA

A good summary:

Wagenmakers EJ, Farrell Simon. AIC model selection using Akaike weights (2004). *Psychonomic Bull Review*, **11**: 192-196.

See Also

[AIC](#), [logLik](#).

Examples

```
## apply a list of different sigmoidal models to data
## and analyze GOF statistics with Akaike weights
## on 6 different sigmoidal models
modList <- list(15, 14, 13, b5, b4, b3)
aics <- sapply(modList, function(x) AIC(pcrfit(reps, 1, 2, x)))
akaike.weights(aics)$weights
```

batchstat

Concatenating or calculating statistics on a 'pcrbatch'

Description

This function will either concatenate data from several `pcrbatches` or calculate some user-defined statistic on the runs within a `pcrbatch`. If the latter is chosen, a grouping vector must be supplied for defining the runs to be subjected to statistical analysis.

Usage

```
batchstat(..., group = NULL, do = c("cbind", "stat"), statfun = mean)
```

Arguments

| | |
|----------------------|--|
| <code>...</code> | one or more <code>pcrbatches</code> . See 'Examples'. |
| <code>group</code> | in case of <code>do = "stat"</code> , a vector defining the groups for statistical analysis. |
| <code>do</code> | concatenate or analyse? |
| <code>statfun</code> | the statistical function to be used if <code>do = "stat"</code> . |

Details

statfun can be any internal R function, i.e. sd, median etc.

Value

Either a concatenated dataframe (do = "cbind"), or a list containing a dataframe(s) with the statistical output for each factor level defined in group, if do = "stat".

Author(s)

Andrej-Nikolai Spiess

Examples

```
## create 3 'pcrbatch'es
## and concatenate
dat1 <- pcrbatch(reps, 2:5, 14)
dat2 <- pcrbatch(reps, 6:9, 14)
dat3 <- pcrbatch(reps, 10:13, 14)
batchstat(dat1, dat2, dat3)

## one 'pcrbatch' and doing
## mean on replicates
## defined by 'group'
dat4 <- pcrbatch(reps, 2:9, 14)
GROUP <- c(1, 1, 1, 1, 2, 2, 2, 2)
batchstat(dat4, do = "stat", group = GROUP, statfun = mean)

## get the standard deviation
batchstat(dat4, do = "stat", group = GROUP, statfun = sd)

## do stats on many 'pcrbatch'es
## All batches must have same length!
batchstat(dat1, dat2, dat3, do = "stat",
          group = c(1, 1, 2, 2))
```

batschetal

*qPCR dilution experiments with replicates (Lightcycler 1.0) from
Batsch et al*

Description

High quality 4-fold dilution experiments with 5 dilution steps and 3 replicates each. See 'Details' for the different setups.

Usage

```
batsch1  
batsch2  
batsch3  
batsch4  
batsch5
```

Format

Data frames with the PCR cycles and 15 qPCR runs with 3 replicates of five 4-fold dilutions. The replicates are defined by FX.Y (X = dilution number, Y = replicate number).

Details

The real-time PCR was conducted with a Lightcycler 1.0 instrument using the following setups:

```
batsch1: Primers for rat SLC6A14, Taqman probes  
batsch2: Primers for human SLC22A13, Taqman probes  
batsch3: Primers for pig EMT, Taqman probes  
batsch4: Primers for chicken ETT, SybrGreen  
batsch5: Primers for human GAPDH, SybrGreen
```

Source

Additional File 5 to the paper.

References

Simultaneous fitting of real-time PCR data with efficiency of amplification modeled as Gaussian function of target fluorescence.
Batsch A et al., *BMC Bioinformatics*, 2008, **9**: 95.

Examples

```
data(batsch1)  
ml <- modlist(batsch1, model = 14)  
plot(ml)
```

Description

Calculates the Bayesian Information Criterion for objects of class `drc`, `lm`, `glm`, `nls` or any other models from which `logLik`, `coef` and `residuals` can be extracted.

Usage

```
BIC(object)
```

Arguments

```
object      a fitted model.
```

Details

$$BIC = -2 * \logLik(object) + npar * \log(nobs)$$

with npar = number of parameters, nobs = number of observations.

Value

The BIC value.

Author(s)

Andrej-Nikolai Spiess

References

Schwartz GE (1978).
Estimating the dimension of a model.
Annals of Statistics, **6**: 461-464.

See Also

[AIC](#), [logLik](#).

Examples

```
m <- pcrfit(reps, 1, 2, 15)
BIC(m)
```

Description

This function calculates the PCR efficiency from a classical qPCR dilution experiment. The threshold cycles are plotted against the logarithmized concentration (or dilution) values, a linear regression line is fit and the efficiency calculated by $E = 10^{\frac{-1}{\text{slope}}}$. A graph is displayed with the raw values plotted with the threshold cycle and the linear regression curve. The threshold cycles are calculated either by some arbitrary fluorescence value (i.e. as given by the qPCR software) or calculated from the second derivative maximum of the first (highest concentration) curve. If values to be predicted are given, they are calculated from the curve and also displayed within. An iterative search of the optimal threshold border can be conducted, thereby going through all slopes and intercepts and selecting the combination that minimizes the AIC of the acquired linear regression curves. See 'Details' for more information on this (maybe controversial) procedure.

Usage

```
calib(refcurve, predcurve = NULL, thresh = "cpD2", term = NULL,
      dil = NULL, plot = TRUE, plot.map = TRUE,
      conf = 0.95, opt = c("none", "inter", "slope"),
      opt.step = c(50, 50), quan = 0.5, slope = NULL, count = 1)
```

Arguments

| | |
|-----------|---|
| refcurve | a 'modlist' containing the curves for calibration. |
| predcurve | an (optional) 'modlist' containing the curves for prediction. |
| thresh | the fluorescence value from which the threshold cycles are defined. Either "cpD2" or a numeric value. |
| term | an (optional) numeric value for the terminating intercept. See 'Details'. |
| dil | a vector with the concentration (or dilution) values corresponding to the calibration curves. |
| plot | logical. Should the optimization process be displayed? If FALSE, only values are returned. |
| plot.map | logical. Should a final heatmap display from the goodness-of-fit of all iterations be displayed? |
| conf | the confidence interval. Defaults to 95%, can be omitted with NULL. |
| opt | type of optimization. See 'Details'. |
| opt.step | a two-element vector. Number of iterations for the intercept as first item, number of slope iterations as second. |
| quan | the top quantile of iterations to be shown in the heatmap. |
| slope | a slope to be defined for the threshold line. Mostly used internally by the iteration process. |
| count | internal counter for recursive purposes. Not to be altered. |

Details

The iterative function conducts a search through all combinations of slope and intercept. For each iteration, either R-square or AIC of the resulting calibration curves are collected, and finally the combination is selected that minimized the AIC. The function goes through all combinations as to avoid local maxima that are likely to happen in this approach. The different settings for `opt` are:

"none" only second derivative maximum or single threshold value.

"inter" iterate along the y-axis intercept.

"slope" iterate along y-axis intercept and slope values.

The paradigm is such that the iterations will start at the second derivative of the first (lowest dilution; highest copy number) curve and terminate at the outlier cycle of the last (highest dilution; lowest copy number) curve. Alternatively a y-value can be given with `term` for the termination threshold. The number of iterations can be defined by `opt.step` but the default values usually give reasonable results. Not to forget, an iterative search only throughout all intercepts can be chosen, as well as a classical approach using the second derivative maximum of the first curve or a defined threshold value from the qPCR software, to be defined in `thresh`. See 'Examples'.

Value

A list with the following components:

| | |
|----------------------------|---|
| <code>refcyc</code> | the calculated threshold cycles for the calibration curves. |
| <code>refcyc.conf</code> | the confidence intervals for <code>refcyc</code> . |
| <code>predcyc</code> | the calculated threshold cycles for the prediction curves. |
| <code>predconc</code> | the predicted concentrations. |
| <code>predconc.conf</code> | the confidence intervals for <code>predconc</code> . |
| <code>eff</code> | the efficiency as calculated from the calibration curve. |
| <code>aic</code> | the AIC value of the linear fit. |
| <code>aicc</code> | the corrected AIC value of the linear fit. |
| <code>rsq</code> | The R-square of the linear fit. |
| <code>rsq.ad</code> | The adjusted R-square of the linear fit. |
| <code>aicMat</code> | a matrix with the calibration curve AIC of each iteration. |

ATTENTION: If iterations were used, the values reflect the analysis of the best fit!

Author(s)

Andrej-Nikolai Spiess

Examples

```
## Define calibration curves,
## dilutions (or copy numbers)
## and curves to be predicted.
## Do background subtraction using
## average of first 8 cycles
```

```

CAL <- modlist(reps, fluo = c(2, 6, 10, 14, 18, 22), backsub = 1:8)
COPIES <- c(100000, 10000, 1000, 100, 10, 1)
PRED <- modlist(reps, fluo = c(3, 7, 11), backsub = 1:8)

## conduct normal quantification using
## the second derivative maximum of
## first curve
res <- calib(refcurve = CAL, predcurve = PRED, thresh = "cpD2", dil = COPIES)

## using a defined treshold value
res <- calib(refcurve = CAL, predcurve = PRED, thresh = 0.5, dil = COPIES)

## iterating the intercept with 50 steps
res <- calib(refcurve = CAL, predcurve = PRED, dil = COPIES, opt = "inter",
             opt.step = c(50, 0))

## iterating the intercept/slope with 20 steps
## Not run:
res <- calib(refcurve = CAL, predcurve = PRED, dil = COPIES, opt = "slope",
             opt.step = c(20, 20))

## End(Not run)

## using replicates for reference curve
ml <- modlist(reps, model = 14)
DIL <- rep(10^(6:0), each = 4)
res <- calib(refcurve = ml, dil = DIL)

```

calib2

Calculation of qPCR efficiency by dilution curve analysis and bootstrapping of dilution curve replicates

Description

This function calculates the PCR efficiency from a classical qPCR dilution experiment. The threshold cycles are plotted against the logarithmized concentration (or dilution) values, a linear regression line is fit and the efficiency calculated by $E = 10^{\frac{-1}{slope}}$. A graph is displayed with the raw values plotted with the threshold cycle and the linear regression curve. The threshold cycles are calculated either by some arbitrary fluorescence value (i.e. as given by the qPCR software) or calculated from the second derivative maximum of the dilution curves. If values to be predicted are given, they are calculated from the curve and also displayed within. `calib2` uses a bootstrap approach if replicates for the dilutions are supplied. See 'Details'.

Usage

```

calib2(refcurve, predcurve = NULL, thresh = "cpD2", dil = NULL,
       group = NULL, plot = TRUE, conf = 0.95, B = 200)

```

Arguments

| | |
|-----------|---|
| refcurve | a 'modlist' containing the curves for calibration. |
| predcurve | an (optional) 'modlist' containing the curves for prediction. |
| thresh | the fluorescence value from which the threshold cycles are defined. Either "cpD2" or a numeric value. |
| dil | a vector with the concentration (or dilution) values corresponding to the calibration curves. |
| group | a factor defining the group membership for the replicates. See 'Examples'. |
| plot | logical. Should the fitting (bootstrapping) be displayed? If FALSE, only values are returned. |
| conf | the confidence interval. Defaults to 95%, can be omitted with NULL. |
| B | the number of bootstraps. |

Details

calib2 calculates confidence intervals for efficiency, AICc, adjusted R-square and the prediction curve concentrations. If single replicates per dilution are supplied by the user, confidence intervals for the prediction curves are calculated based on asymptotic normality. If multiple replicates are supplied, the regression curves are calculated by randomly sampling one of the replicates from each dilution group. The confidence intervals are then calculated from the bootstrapped results.

Value

A list with the following components:

| | |
|-----------|---|
| eff | the efficiency. |
| AICc | the second-order corrected AIC. |
| Rsq.ad | the adjusted R-square. |
| predconc | the (log) concentration of the predicted curves. |
| conf.boot | a list containing the confidence intervals for the efficiency, the AICc, Rsq.ad and the predicted concentrations. |

A plot is also supplied for efficiency, AICc, Rsq.ad and predicted concentrations including confidence intervals in red.

Author(s)

Andrej-Nikolai Spiess

Examples

```
## Define calibration curves,
## dilutions (or copy numbers)
## and curves to be predicted.
## Do background subtraction using
## average of first 8 cycles
CAL <- modlist(reps, fluo = c(2, 6, 10, 14, 18, 22), backsub = 1:8)
```

```

COPIES <- c(100000, 10000, 1000, 100, 10, 1)
PRED <- modlist(reps, fluo = c(3, 7, 11), backsub = 1:8)

## conduct normal quantification using
## the second derivative maximum of
## first curve
res <- calib2(refcurve = CAL, predcurve = PRED, thresh = "cpD2", dil = COPIES)

## using a defined treshold value
res <- calib2(refcurve = CAL, predcurve = PRED, thresh = 0.5, dil = COPIES)

## using six dilutions with
## four replicates/dilution
## Not run:
CAL2 <- modlist(reps, fluo = 2:25, backsub = 1:8)
res <- calib2(refcurve = CAL2, predcurve = PRED, thresh = "cpD2",
              dil = COPIES, group = gl(6,4))

## End(Not run)

```

curvemean

Building a model which averages a batch of qPCR curves

Description

Starting with a batch of qPCR curves from class `modlist`, the function builds a sigmoidal model which averages the cycle numbers or expression values of the curves at every y-value (raw fluorescence). Thus, in contrast to existing qPCR averaging methods, not only the efficiencies or threshold cycles (as can be obtained from multiple reference genes) are averaged, but the complete structure of the batch at every fluorescence value. Beware: This is curve averaging, NOT model averaging!

Usage

```

curvemean(ml, type = c("fluo", "expval"), mean = c("amean", "gmean", "hmean"),
          which = 1, plot = TRUE)

```

Arguments

| | |
|--------------------|--|
| <code>ml</code> | a qPCR batch object of class 'modlist'. |
| <code>type</code> | what to average. Either the cycles deduced from the fluorescence or from the expression values. See 'Details'. |
| <code>mean</code> | the averaging function, see 'Details'. Default is the arithmetic mean. |
| <code>which</code> | the position of the curve in the list which is used for defining the y-values. Defaults to the first curve. |
| <code>plot</code> | should results be plotted? |

Details

Starting from the raw fluorescence values or expression values of a defined curve in the list, the cycle number of all other curves at this value are calculated and averaged with the method as under mean. After that, a new sigmoidal model is fit to the obtained data, using the same sigmoidal model as in `ml`. Note: often works better (more data points can be averaged) if `norm = TRUE` when building the model list with `modlist`. Three different averaging functions can be used:

Arithmetic mean:

$$\bar{x} = \frac{\sum_{i=1}^n x_i}{n}$$

Geometric mean:

$$\bar{x} = \left(\prod_{i=1}^n x_i \right)^{\frac{1}{n}}$$

Harmonic mean:

$$\bar{x} = \frac{n}{\sum_{i=1}^n \frac{1}{x_i}}$$

If `type = "expval"`, the cycles are not averaged from the calculated fluorescence values but from the expression value calculated for each model i at each cycle j by

$$CYC_{new} = \overline{expval} = \overline{EFF_{i,j}^{CYC_{i,j}}}$$

Value

A new model of class 'nls' and 'pcrfit' from the averaged curve.

Author(s)

Andrej-Nikolai Spiess

References

Accurate normalization of real-time quantitative RT-PCR data by geometric averaging of multiple internal control genes. Vandesompele et al., *Genome Biology*, 2002, **3**: research0034.1-0034.11.

Examples

```
## create arithmetic mean curve of four
## serial dilutions
ml <- modlist(reps, fluo = c(2, 6, 10, 14))
curvemean(ml)

## effect of normalizing data to [0, 1]:
## more averaged datapoints
ml <- modlist(reps, fluo = c(2, 6, 10, 14), norm = TRUE)
curvemean(ml)

## averaging the expression value
## (eff^cp)
curvemean(ml, type = "expval")
```

Description

An alternative to the classical crossing point/threshold cycle estimation as described in Guescini et al. A tangent is fit to the first derivative maximum (point of inflection) of the modeled curve and the intersect with the x-axis is calculated.

Usage

```
Cy0(object, plot = FALSE, add = FALSE, ...)
```

Arguments

| | |
|--------|--|
| object | a fitted object of class 'pcrfit'. |
| plot | if TRUE, displays a plot of Cy0. |
| add | if TRUE, a plot is added to any other existing plot, i.e. as from <code>plot.pcrfit</code> . |
| ... | other parameters to be passed to <code>plot.pcrfit</code> or <code>points</code> . |

Details

The function calculates the first derivative maximum (cpD1) of the curve and the slope and fluorescence at that point. Cy0 is then calculated by $Cy0 = cpD1 - (Fluo/slope)$.

Value

The Cy0 value.

Author(s)

Andrej-Nikolai Spiess

References

A new real-time PCR method to overcome significant quantitative inaccuracy due to slight amplification inhibition.
Guescini M et al, *BMC Bioinformatics*, 2008, **9**: 326.

Examples

```
## single curve with plot
m <- pcrfit(reps, 1, 2, 15)
Cy0(m, plot = TRUE)

## add to 'efficiency' plot
efficiency(m)
Cy0(m, add = TRUE)
```

```
## compare s.d. of replicates between
## Cy0 and cpD2 method. cpD2 wins!
ml <- modlist(reps, model = 14)
cy0 <- sapply(ml, function(x) Cy0(x))
cpd2 <- sapply(ml, function(x) efficiency(x, plot = FALSE)$cpD2)
tapply(cy0, gl(7, 4), function(x) sd(x))
tapply(cpd2, gl(7, 4), function(x) sd(x))
```

 eff

The amplification efficiency curve of a fitted object

Description

Calculates the efficiency curve from the fitted object by $E = \frac{F(n)}{F(n-1)}$, with E = efficiency, F = raw fluorescence, n = Cycle number.

Usage

```
eff(object, sequence = NULL, plot = FALSE)
```

Arguments

| | |
|----------|---|
| object | an object of class 'pcrfit'. |
| sequence | a 3-element vector (from; to; by) defining the sequence for the efficiency curve, defaults to [min(Cycles), max(Cycles)] with 100 points per cycle. |
| plot | should the efficiency be plotted? |

Value

A list with the following components:

| | |
|----------|---|
| eff.x | the cycle points. |
| eff.y | the efficiency values at eff.x. |
| effmax.x | the cycle number with the highest efficiency. |
| effmax.y | the maximum efficiency. |

Author(s)

Andrej-Nikolai Spiess

Examples

```
m <- pcrfit(reps, 1, 2, 15)

## with default 100 points per cycle
eff(m, plot = TRUE)

## not all data and only 10 points per cycle
eff(m, sequence = c(5, 35, 0.1), plot = TRUE)
```

| | |
|------------|---|
| efficiency | <i>Calculation of qPCR efficiency and several other important qPCR parameters</i> |
|------------|---|

Description

This function calculates the PCR efficiency of a model of class 'pcrfit' and several other important values for qPCR quantification like the first and second derivatives and the corresponding maxima thereof (i.e. threshold cycles). These values can be subsequently used for the calculation of PCR kinetics, fold induction etc. All values are included in a graphical output of the fit. Additionally, several measures of goodness-of-fit are calculated, i.e. the Akaike Information Criterion (AIC), the residual variance and the R-square value.

Usage

```
efficiency(object, plot = TRUE, type = "cpD2", thresh = NULL,
           shift = 0, amount = NULL)
```

Arguments

| | |
|--------|--|
| object | an object of class 'pcrfit'. |
| plot | logical. If TRUE, a graph is displayed. If FALSE, values are printed out. |
| type | the method of efficiency estimation. See 'Details'. |
| thresh | an (optional) numeric value for a fluorescence threshold border. Overrides type. |
| shift | a user defined shift in cycles from the values defined by type. See 'Examples'. |
| amount | the template amount or molecule number for quantitative calibration. |

Details

The efficiency is always calculated from the efficiency curve (in blue), which is calculated according to $E = \frac{F(n)}{F(n-1)}$ from the fitted curve, but taken from different points at the curve, as to be defined in type:

- "cpD2" taken from the maximum of the second derivative curve,
- "cpD1" taken from the maximum of the first derivative curve,
- "maxE" taken from the maximum of the efficiency curve,
- "expR" taken from the exponential region by $expR = cpD2 - (cpD1 - cpD2)$,
- "CQ" taken from the 20% value of the fluorescence at "cpD2" as developed by Corbett Research (comparative quantification),
- "Cy0" the intersection of a tangent on the first derivative maximum with the abscissa as calculated according to Guescini et al. or
- a numeric value taken from the threshold cycle output of the PCR software, i.e. 15.24 as defined in type or
- a numeric value taken from the fluorescence threshold output of the PCR software as defined in thresh.

The initial fluorescence $F(0)$ for relative or absolute quantification is either calculated by setting $x = 0$ in the sigmoidal model of `object` giving `init1` or by calculating an exponential model down (`init2`) with $F(0) = \frac{F(n)}{E(n)^{Cyc}}$, with $F(n)$ = raw fluorescence at the cycle number defined by `type`, $E(n)$ = PCR efficiency at the cycle number defined by `type` and Cyc = the cycle number defined by `type`. If a template amount is defined, a conversion factor $cf = \frac{amount}{F(0)}$ is given. The different measures for goodness-of-fit give an overview for the validity of the efficiency estimation. First and second derivatives are calculated from the fitted function and the maxima of the derivatives curve and the efficiency curve are obtained.

Value

A list with the following components:

| | |
|---------------------|--|
| <code>eff</code> | the PCR efficiency. |
| <code>resVar</code> | the residual variance. |
| <code>AICc</code> | the bias-corrected Akaike Information Criterion. |
| <code>AIC</code> | the Akaike Information Criterion. |
| <code>Rsq</code> | the R-square value. |
| <code>Rsq.ad</code> | the adjusted R-square value. |
| <code>cpD1</code> | the first derivative maximum (point of inflection in 'l4' or 'b4' models, can be used for defining the threshold cycle). |
| <code>cpD2</code> | the second derivative maximum (turning point of <code>cpD1</code> , more often used for defining the threshold cycle). |
| <code>cpE</code> | the PCR cycle with the highest efficiency. |
| <code>cpR</code> | the PCR cycle within the exponential region calculated as under 'Details'. |
| <code>cpT</code> | the PCR cycle corresponding to the fluorescence threshold as defined in <code>thresh</code> . |
| <code>Cy0</code> | the PCR threshold cycle 'Cy0' according to Guescini et al. See 'Details'. |
| <code>cpCQ</code> | the PCR cycle corresponding to the 20% fluorescence value at 'cpD2'. |
| <code>fluo</code> | the raw fluorescence value at the point defined by <code>type</code> or <code>thresh</code> . |
| <code>init1</code> | the initial template fluorescence from the sigmoidal model, calculated as under 'Details'. |
| <code>init2</code> | the initial template fluorescence from an exponential model, calculated as under 'Details'. |
| <code>cf</code> | the conversion factor between raw fluorescence and template amount, if the latter is defined. |

Note

Three parameter models ('b3' or 'l3') do not work very well in calculating the PCR efficiency. It is advisable not to take too many cycles of the plateau phase prior to fitting the model as this has a strong effect on the validity of the efficiency estimates.

Author(s)

Andrej-Nikolai Spiess

References

- Weihong Liu and David A. Saint (2002) Validation of a quantitative method for real time PCR kinetics, *BBRC*, **294**, 347 - 353. A new real-time PCR method to overcome significant quantitative inaccuracy due to slight amplification inhibition.
- Guescini M et al, *BMC Bioinformatics*, 2008, **9**: 326.

Examples

```
## Fitting initial model
m1 <- pcrfit(reps, 1, 2, 14)
efficiency(m1)

## selection of best model
## using one cycle 'downstream'
## of second derivative max
m2 <- mselect(m1)
efficiency(m2, type = "cpD2", shift = -1)

## using "maxE" method, with calculation of PCR efficiency
## 2 cycles 'upstream' from the cycle of max efficiency
efficiency(m2, type = "maxE", shift = 2)

## using the exponential region
efficiency(m2, type = "expR")

## using threshold cycle (i.e. 15.32)
## from PCR software
efficiency(m2, type = 15.32)

## using Cy0 method from
## Guescini et al. (2008),
## add Cy0 tangent
efficiency(m2, type = "Cy0")
Cy0(m2, add = TRUE)

## using a defined fluorescence
## threshold value from PCR software
efficiency(m2, thresh = 1)

## using the first 30 cycles and a template amount
## (optical calibration)
m3 <- pcrfit(reps[1:30, ], 1, 2, 15)
efficiency(m3, amount = 1E3)
```

Description

The evidence ratio

$$\frac{1}{\exp(-0.5 * (AIC2 - AIC1))}$$

is calculated for either two fitted models or two numerical values. Models can be compared that are not nested and where the f-test on residual-sum-of-squares is not applicable.

Usage

```
evidence(x, y, type = c("AICc", "AIC"))
```

Arguments

| | |
|------|--|
| x | a fitted object or numerical value. |
| y | a fitted object or numerical value. |
| type | Bias-corrected (AICc , default) or original (AIC) versions of the Akaike Information Criterion. |

Details

Small differences in AIC values can mean substantial more 'likelihood' of one model over the other. For example, a model with AIC = -130 is nearly 150 times more likely than a model with AIC = -120.

Value

A value of the first model x being more likely than the second model y. If large, first model is better. If small, second model is better.

Author(s)

Andrej-Nikolai Spiess

Examples

```
## compare two four-parameter and five-parameter
## log-logistic models
m1 <- pcrfit(reps, 1, 2, 14)
m2 <- pcrfit(reps, 1, 2, 15)
evidence(m2, m1)

## ratio of two AIC's
evidence(-120, -123)
```

| | |
|---------|---|
| expcomp | <i>Comparison of all available sigmoidal models by RMSE within the exponential region</i> |
|---------|---|

Description

The exponential region of the qPCR data is identified by the studentized outlier method, as in [expfit](#). The root-mean-squared-error of all available sigmoidal models within this region is then calculated. The result of the fits are plotted and models returned in order of ascending RMSE.

Usage

```
expcomp(object, ...)
```

Arguments

| | |
|--------|--|
| object | an object of class 'pcrfit'. |
| ... | other parameters to be passed to <code>expfit</code> . |

Details

The following sigmoidal models are fitted: b3, b4, b5, l3, l4, l5, w3, w4, baro5

Value

A dataframe with names of the models, in ascending order of RMSE.

Author(s)

Andrej-Nikolai Spiess

Examples

```
m <- pcrfit(reps, 1, 2, 14)
expcomp(m)
```

 expfit

Calculation of PCR efficiency by fitting an exponential model

Description

An exponential model is fit to a window of defined size on the qPCR raw data. The window is identified either by the 'studentized outlier' method as described in Tichopad et al. (2003), the 'midpoint' method (Peirson et al., 2003) or by subtracting the difference of cpD1 and cpD2 from cpD2 ('ERBCP', unpublished).

Usage

```
expfit(object, method = c("outlier", "midpoint", "ERBCP"), pval = 0.05,
       n.outl = 3, n.ground = 1:5, corfact = 1,
       fix = c("top", "bottom", "middle"), nfit = 5, plot = TRUE, ...)
```

Arguments

| | |
|----------|---|
| object | an object of class 'pcrfit'. |
| method | one of the three possible methods to be used for defining the position of the fitting window. |
| pval | for method "outlier", the p-value for the outlier test. |
| n.outl | for method "outlier", the number of successive outlier cycles. |
| n.ground | for method "midpoint", the number of cycles in the noisy ground phase to calculate the standard deviation from. |
| corfact | for method "ERBCP", the correction factor for finding the exponential region. See 'Details'. |
| fix | for methods "midpoint" and "ERBCP", the orientation of the fitting window based on the identified point. See 'Details'. |
| nfit | the size of the fitting window. |
| plot | logical. If TRUE, a graphical display of the curve and the fitted region is shown. |
| ... | other parameters to be passed to the plotting function. |

Details

The exponential growth function $f(x) = b * exp(k * x) + e$ is fit to the data. Calls `outlier` for the calculation of the studentized residuals and 'outlier' cycle, and `midpoint` for calculation of the exponential phase 'midpoint'. For method 'ERBCP' (Exponential Region By Crossing Points), the exponential region is calculated by $expR = cpD2 - corfact * (cpD1 - cpD2)$. The efficiency is calculated a) from the exponential fit with $Eff = exp(k)$ and b) for each cycle within the exponential region from the raw fluorescence values by $Eff = \frac{F(n)}{F(n-1)}$. The initial template fluorescence (F0) is derived from parameter b.

Value

A list with the following components:

| | |
|------------|--|
| point | the point within the exponential region as identified by one of the three methods. |
| cycles | the cycles of the identified region as defined by <code>method</code> , <code>fix</code> and <code>nfit</code> . |
| eff | the efficiency calculated from the exponential fit. |
| eff.cycles | the efficiencies of all points within the identified region. |
| AIC | the Akaike Information Criterion of the fit. |
| resVar | the residual variance of the fit. |
| RMSE | the root-mean-squared-error of the fit. |
| init | the initial template fluorescence. |
| mod | the exponential model of class 'nls'. |

Author(s)

Andrej-Nikolai Spiess

References

Standardized determination of real-time PCR efficiency from a single reaction set-up. Tichopad et al., *Nucleic Acids Research*, 2003, **e122**.

Experimental validation of novel and conventional approaches to quantitative real-time PCR data analysis. Peirson et al., *Nucleic Acids Research*, 2003, **e73**.

Examples

```
## using 'outlier' method
m1 <- pcrfit(reps, 1, 2, 15)
expfit(m1)

## 'midpoint' method and 7 cycle window
expfit(m1, method = "midpoint", nfit = 7)

## 'ERBCP' method with window centered around
## fixpoint
expfit(m1, method = "ERBCP", fix = "middle")
```

fitprob

The (chi-square) fit probability

Description

Calculates the fit probability for objects of class `drc`, `lm`, `glm`, `nls` or any other models from which `residuals` and `coef` can be extracted.

Usage

```
fitprob(object)
```

Arguments

object a fitted model.

Details

Although the residual variance is a measure for the quality of a fit, it provides no information of how good a fit performs in relation to other fits in the (hypothetical) infinite population of assays performed under the same conditions. Under the assumption that the responses are approximately normally distributed and that the regression's expectation surface is approximately linear in the neighbourhood of the best fit, it can be shown that the residual sum-of-squares (RSS) obeys a χ^2 distribution with $df = \text{number of curve points} - \text{number of parameters}$. The p-value $\chi^2(RSS, df)$ can be viewed as the fraction of an infinite number of assays that, under the same conditions, would be expected to have a better curve fit, i.e. a smaller RSS.

Value

The p-value.

Author(s)

Andrej-Nikolai Spiess

References

Draper NR, Smith H.
Applied Regression Analysis, 3rd Ed.
Wiley, New York, 1998.

Examples

```
## 'good model'  
m1 <- pcrfit(reps, 1, 2, 14)  
fitprob(m1)  
  
## 'bad model'  
m2 <- pcrfit(reps, 1, 2, w4)  
fitprob(m1)
```

`guescini1`*A qPCR dilution experiment with replicates (Lightcycler 480) from Guescini et al*

Description

A high quality 10-fold dilution experiment with 7 dilution steps and 12 replicates each.

Usage

```
data(guescini1)
```

Format

A data frame with the PCR cycles and 84 qPCR runs with 12 replicates of seven 10-fold dilutions. The replicates are defined by FX.Y (X = dilution number, Y = replicate number).

Details

The real-time PCR was conducted with primers for the NADH dehydrogenase 1 in a Lightcycler 480 (Roche). The data is background subtracted.

Source

Supplemental data 1 to the paper.

References

A new real-time PCR method to overcome significant quantitative inaccuracy due to slight amplification inhibition.
Guescini M et al, *BMC Bioinformatics*, 2008, **9**: 326.

Examples

```
## Not run:  
data(guescini1)  
ml <- modlist(guescini1, model = 14)  
plot(ml)  
  
## End(Not run)
```

`guescini2`*A qPCR experiment with replicates and simulated inhibition (Lightcycler 480) from Guescini et al*

Description

A high quality experiment in which a decreasing amount of PCR mix mimics PCR inhibition.

Usage

```
data(guescini2)
```

Format

A data frame with the PCR cycles and 60 qPCR runs from 12 replicates with 5 decreasing steps of PCR mix. The replicates are defined by FX.Y (X = PCR mix dilution number, Y = replicate number). For example:

```
F1.X 100% Mix
F2.X 90% Mix
F3.X 80% Mix
F4.X 70% Mix
F5.X 60% Mix
```

Details

The real-time PCR was conducted with primers for the NADH dehydrogenase 1 in a Lightcycler 480 (Roche). The data is background subtracted.

Source

Supplemental data 1 to the paper.

References

A new real-time PCR method to overcome significant quantitative inaccuracy due to slight amplification inhibition.
Guescini M et al, *BMC Bioinformatics*, 2008, **9**: 326.

Examples

```
## effect of decreasing mix
## on PCR efficiency
data(guescini2)
ml <- modlist(guescini2, model = 14)
effs <- sapply(ml, function(x) efficiency(x, plot = FALSE)$eff)
```

```
## mean for the replicates
tapply(effs, gl(5, 12), function(x) mean(x, na.rm = TRUE))
```

LR

*Calculation of likelihood ratios for nested models***Description**

Calculates the likelihood ratio and p-value from a chi-square distribution for two nested models.

Usage

```
LR(objX, objY)
```

Arguments

`objX` Either a value of class `logLik` or a model for which `logLik` can be applied.
`objY` Either a value of class `logLik` or a model for which `logLik` can be applied.

Details

The likelihood ratio statistic is

$$LR = \frac{f(X, \hat{\phi}, \hat{\psi})}{f(X, \phi, \hat{\psi}_0)}$$

The usual test statistic is

$$\Lambda = 2 * (l(\hat{\phi}, \hat{\psi}) - l(\phi, \hat{\psi}_0))$$

Following the large sample theory, if H_0 is true, then

$$\Lambda \sim \chi_p^2$$

Value

A list containing the following items:

`ratio` the likelihood ratio statistic.
`df` the change in parameters.
`p.value` the p-value from a chi-square distribution. See Details.

Author(s)

Andrej-Nikolai Spiess

See Also

[AIC](#), [logLik](#).

Examples

```
## compare 15 and 14 model
m1 <- pcrfit(reps, 1, 2, 15)
m2 <- pcrfit(reps, 1, 2, 14)
LR(m1, m2)
```

maxRatio

The maxRatio method as in Shain et al

Description

The maximum ratio (MR) is determined along the interpolated curve of $F(x)/F(x-1)$ and the corresponding cycle number at MR is taken. A respective cycle number (FCN) is then calculated for MR.

Usage

```
maxRatio(m1, plot = TRUE, ...)
```

Arguments

| | |
|------|--|
| m1 | an object of class 'modlist'. |
| plot | Should diagnostic plots be displayed? |
| ... | other parameters to be passed to <code>plot</code> . |

Details

In the original paper the authors smooth the datapoints and then apply a cubic spline on the ratio curve, in order to attain a resolution of 0.01 cycles. The function here calculates the ratio along a curve of the sigmoidal fit, which results in essentially the same.

Value

A list with the following components:

| | |
|-------|---|
| mr | the maximum ratio. |
| fcn | the cycle number at mr. |
| fcna | a corrected fcn, as described in Shain et al. |
| names | the names of the runs as taken from the original dataframe. |

Author(s)

Andrej-Nikolai Spiess

References

A new method for robust quantitative and qualitative analysis of real-time PCR.
Shain & Clemens, *Nucleic Acids Research*, 2008, **36**, e91.

Examples

```
ml <- modlist(reps, model = 15)
maxRatio(ml)
```

| | |
|----------|--|
| meanlist | <i>Amalgamation of single data models to a model containing the averaged model</i> |
|----------|--|

Description

Starting from a 'modlist' containing qPCR models from single data, `meanlist` amalgamates the models according to the grouping structure as defined in `group`. The result is a 'modlist' with models obtained from averaging the replicates by `pcrfit`.

Usage

```
meanlist(object, group, type = c("mean", "median"))
```

Arguments

| | |
|---------------------|--|
| <code>object</code> | an object of class 'modlist'. |
| <code>group</code> | a vector defining the replicates for each group. |
| <code>type</code> | how to average the data. |

Details

As being defined by `group`, the average data of the curves is subjected to `pcrfit` and a new modlist with the averaged models is created. Similar to `replist` but does not contain the replicates within the 'nls' model but the averaged model with only ONE curve.

Value

An object of class 'modlist' containing the averaged models of class 'nls'/'pcrfit'.

Author(s)

Andrej-Nikolai Spiess

See Also

`modlist`, `replist`.

Examples

```
ml <- modlist(reps, model = 14)
res <- meanlist(ml, group = gl(7, 4))
plot(res)
efficiency(res[[1]])
```

| | |
|----------|--|
| midpoint | <i>Calculation of the exponential region midpoint according to Peirson et al</i> |
|----------|--|

Description

Calculates the exponential region midpoint using the algorithm described under 'References'.

Usage

```
midpoint(object, noise.cyc = 1:5)
```

Arguments

| | |
|-----------|---|
| object | a fitted object. |
| noise.cyc | the cycles defining the background noise. |

Details

The 'midpoint' region is calculated by

$$F_{noise} \times \sqrt{\frac{F_{max}}{F_{noise}}}$$

with F_{noise} = the standard deviation of the background cycles and F_{max} = the maximal fluorescence.

Value

A list with the following components:

| | |
|--------|---|
| f.mp | the 'midpoint' fluorescence. |
| cyc.mp | the 'midpoint' cycle, as predicted from f.mp. |

Author(s)

Andrej-Nikolai Spiess

References

Experimental validation of novel and conventional approaches to quantitative real-time PCR data analysis. Peirson et al., *Nucleic Acids Research*, 2003, **e73**.

Examples

```
m1 <- pcrfit(reps, 1, 2, 15)
mp <- midpoint(m1)
plot(m1)
abline(h = mp$f.mp, col = 2)
abline(v = mp$cyc, col = 2)
```

modlist *Create nonlinear models from a dataframe and coerce them into a list*

Description

Essential function to create a list of nonlinear models from the columns of a qPCR dataframe. Very handy if following functions should be applied to different qPCR models, i.e. by [sapply](#).

Usage

```
modlist(x, cyc = 1, fluo = NULL, model = 14, opt = FALSE, norm = FALSE,
        backsub = NULL, opt.method = "LM", nls.method = "port",
        sig.level = 0.05, crit = "ftest", ...)
```

Arguments

| | |
|------------|---|
| x | a dataframe containing the qPCR data. |
| cyc | the column containing the cycle data. Defaults to first column. |
| fluo | the column(s) (runs) to be analyzed. If NULL, all runs will be considered. |
| model | the model to be used. |
| opt | logical. Should model selection be applied? |
| norm | logical. Should the raw data be normalized within [0; 1] before model fitting? |
| backsub | background subtraction. If NULL, not applied. Otherwise, a numeric sequence such as 1:10. See 'Details' in pcrbatch . |
| opt.method | see pcrfit . |
| nls.method | see pcrfit . |
| sig.level | see mselect . |
| crit | see mselect . |
| ... | other parameters to be passed to pcrfit or mselect . |

Value

A list with each item containing the model from each column. A 'names' item containing the column name is attached to each model.

Author(s)

Andrej-Nikolai Spiess

Examples

```
## calculate efficiencies for each run in
## the 'reps' data
## subtract background using the first 8 cycles
ml <- modlist(reps, model = 15, backsub = 1:8)
sapply(ml, function(x) efficiency(x, plot = FALSE)$eff)

## 'crossing points' for the first 3 runs (normalized)
## and using best model from Akaike weights
ml <- modlist(reps, 1, 2:4, model = 15, opt = TRUE, norm = TRUE, crit = "weights" )
sapply(ml, function(x) efficiency(x, plot = FALSE)$cpD2)
```

mselect

Selection of the best model by nested F-tests/likelihood ratios/Akaike weights

Description

Model selection by comparison of different models using

- 1) the maximum log likelihood value,
- 2) Akaike's Information Criterion (AIC),
- 3) bias-corrected Akaike's Information Criterion (AICc),
- 4) the estimated residual variance,
- 5) the p-value from a nested F-test on the residual variance,
- 6) the p-value from the likelihood ratio (chi-square),
- 7) the Akaike weights based on AIC and
- 8) the Akaike weights based on AICc.

The best model is chosen by 5), 6), or 8) and returned as a new model.

Usage

```
mselect(object, fctList = NULL, sig.level = 0.05, verbose = TRUE,
        crit = c("ftest", "ratio", "weights"), do.all = FALSE)
```

Arguments

| | |
|-----------|--|
| object | an object of class 'pcrfit'. |
| fctList | a list of functions to be analyzed, i.e. for a non-nested regime. Should also contain the original model. |
| sig.level | the significance level for the nested F-test. |
| verbose | logical. If TRUE, the result matrix is displayed in the console. |
| crit | the criterium for model selection. Either 'ftest'/'ratio' for nested models or 'weights' for nested and non-nested models. |
| do.all | if TRUE, all available sigmoidal models are tested and the best one is selected based on AICc weights. |

Details

Criteria 5) and 6) cannot be used for comparison unless the models are nested. Criterion 7), Akaike weights, can be used for nested and non-nested regimes. For criterion 1) the larger the better. For criteria 2), 3) and 4): the smaller the better. The best model is chosen either from the nested F-test, likelihood ratio or corrected Akaike weights and returned as a new model. When using 'ftest'/'ratio' the corresponding nested functions are analyzed automatically, i.e. b3/b4/b5; 13/14/15. If supplying nested models, please do this with ascending number of parameters.

Value

A model of the best fit selected by the nested F-tests, likelihood ratios or Akaike weights. The new model has an additional list item 'retMat' with a result matrix of the criterion tests.

Author(s)

Andrej-Nikolai Spiess

See Also

[LR](#), [akaike.weights](#)

Examples

```
## choose best model based on F-tests
## on the corresponding nested models
m1 <- pcrfit(reps, 1, 2, 13)
m2 <- mselect(m1)
summary(m2) ## Converted to 15 model !

## use Akaike weights on non-nested models
## compare to original model
m2 <- mselect(m1, fctList = list(13, 15, b3), crit = "weights")
summary(m2) ## Also converted to 15 model !

## try all sigmoidal models
m3 <- pcrfit(reps, 1, 20, 14)
mselect(m3, do.all = TRUE) ## baro5 wins by far!
```

outlier

Calculation of qPCR outlier cycles

Description

Calculates the first significant outlier cycle using the studentized residuals method.

Usage

```
outlier(object, pval = 0.05, nsig = 3)
```

Arguments

| | |
|---------------------|--|
| <code>object</code> | an object of class 'pcrfit'. |
| <code>pval</code> | the p-value for the outlier test. |
| <code>nsig</code> | the number of successive outlier tests. See 'Details'. |

Details

Outliers are calculated essentially as described in the reference below. The steps are:

- 1) Fitting a linear model to some background cycles 1:x.
- 2) Calculation of the studentized residuals.
- 3) Test if the last residual is an outlier in terms of t-distribution.
- 4) Test if the next `nsig - 1` cycles are also outlier cycles.
- 5) If so, take cycle from 3), otherwise $x = x + 1$ and start at 1).

Value

A list with the following components:

| | |
|---------------------|---|
| <code>outl</code> | the outlier cycle. |
| <code>f.outl</code> | the fluorescence at <code>outl</code> . |

Author(s)

Andrej-Nikolai Spiess

References

Standardized determination of real-time PCR efficiency from a single reaction set-up. Tichopad et al., *Nucleic Acids Research*, 2003, **e122**.

Examples

```
m <- pcrfit(reps, 1, 2, 15)
out <- outlier(m)
plot(m)
abline(v = out$outl, col = 2)
abline(h = out$f.outl, col = 2)
```

| | |
|----------|--|
| pcrbatch | <i>Batch calculation of qPCR efficiency and several other important qPCR parameters with different methods</i> |
|----------|--|

Description

This function batch calculates the results obtained from [efficiency](#), [sliwin](#), and [expfit](#) on a dataframe containing many qPCR runs. The input can also be a list obtained from [modlist](#), which simplifies things in many cases. The output is a dataframe with the estimated parameters and model description. Very easy to use on datasheets containing many qPCR runs, i.e. as can be imported from Excel. The data is automatically copied to the clipboard.

Usage

```
pcrbatch(x, cols = NULL, model = 14, group = NULL, type = "cpD2",
         opt = FALSE, smooth = c("none", "tukey", "lowess"), norm = FALSE,
         fact = 1, ave = c("mean", "median"), backsub = NULL,
         retPar = FALSE, crit, ...)
```

Arguments

| | |
|---------|--|
| x | a dataframe containing the qPCR raw data from the different runs or a list obtained from modlist . |
| cols | the columns (runs) to be analyzed. If NULL, all runs will be considered. |
| group | a vector containing the grouping for possible replicates. |
| model | the model to be used. |
| type | the point on the amplification curve from which the efficiency is estimated. See efficiency . |
| opt | logical. Should model optimization take place? If TRUE, model selection is applied. |
| smooth | the smoothing algorithm for the data. Either Tukey's running median or non-parametric lowess smoothing. |
| norm | logical. Normalization of the raw data within [0, 1]. See references. |
| fact | a constant multiplication factor for the raw qPCR data. |
| ave | averaging method for replicates. Defaults to "mean", another option is "median". |
| backsub | background subtraction. If NULL, not applied. Otherwise, a numeric sequence such as 1:10. See 'Details'. |
| retPar | logical. Should the parameters from the fit be included in the output? |
| crit | the criterium for model selection. See mselect . |
| ... | other parameters to be passed to downstream methods. |

Details

The qPCR raw data should be arranged with the cycle numbers in the first column with the name "Cycles". All subsequent columns must be plain raw data with sensible column descriptions. If data of class `modlist` is given, all previous data manipulations (i.e. smoothing or background subtraction) are eliminated as the raw data is taken. Thus, if `opt` was applied with `modlist`, this has to be repeated by setting `opt = TRUE`. If replicates are defined, the output will contain a numbering of groups (i.e. "group1" for the first replicate group). The model selection process is optional, but we advocate using this for obtaining better parameter estimates. Normalization has been described to improve certain qPCR analyses, but this has still to be independently evaluated. Background subtraction is done by averaging the `backsub` cycles of the run and subtracting this from all data points.

Value

A dataframe with the results in columns containing the calculated values with descriptions and the method used as the name prefix.

Note

When subsequent use of `ratiocalc` is desired, use `pcrbatch` on the single run level with `group = NULL`, otherwise error propagation will fail.

Author(s)

Andrej-Nikolai Spiess

References

A standard curve based method for relative real time PCR data processing. Larionov et al., *BMC Bioinformatics*, **6**: 62.

Examples

```
## complete dataset
## Not run:
temp <- pcrbatch(reps)

## End(Not run)

## first 4 runs and return parameters of fit
## do background subtraction using the first 8 cycles
res1 <- pcrbatch(reps, 2:4, retPar = TRUE, backsub = 1:8)

## first 8 runs, with 4 replicates each, 15 model
res2 <- pcrbatch(reps, 2:9, model = 15, c(1,1,1,1,2,2,2,2))

## using model selection (likelihood ratio) on the first 4 runs,
## run 1+2 are replicates
res3 <- pcrbatch(reps, 2:5, group = c(1,1,2,3), opt = TRUE, crit = "ratio")
```

```
## converting a 'modlist' to 'pcrbatch'
ml <- modlist(reps, 1, 2:5, b3)
res4 <- pcrbatch(ml)
```

pcrboot

Bootstrapping and jackknifing qPCR (and other) data

Description

Confidence intervals for the estimated parameters and goodness-of-fit measures are calculated for a nonlinear qPCR data fit by either

- a) bootstrapping the residuals of the fit or
- b) jackknifing and refitting the data.

If data of class `pcrfit` is supplied, confidence intervals are also calculated for all parameters obtained from the `efficiency` analysis. Works for all models with a `model$data` item and a `fitted` and `residuals` function. See 'Details'.

Usage

```
pcrboot(object, type = c("boot", "jack"), B = 100, njack = 1,
        plot = TRUE, do.eff = TRUE, conf = 0.95, verbose = TRUE, ...)
```

Arguments

| | |
|----------------------|---|
| <code>object</code> | an object of class 'pcrfit' or other. |
| <code>type</code> | either bootstrapping or jackknifing. |
| <code>B</code> | numeric. The number of iterations. |
| <code>njack</code> | numeric. In case of <code>type = "jack"</code> , how many datapoints to exclude. Defaults to leave-one-out. |
| <code>plot</code> | should the fitting and final results be displayed as a plot? |
| <code>do.eff</code> | logical. If TRUE, <code>efficiency</code> analysis will be performed. |
| <code>conf</code> | the confidence level. |
| <code>verbose</code> | logical. If TRUE, the iterations will be printed on the console. |
| <code>...</code> | other parameters to be passed on to the plotting functions. |

Details

Non-parametric bootstrapping is applied using the centered residuals.

- 1) Obtain the residuals from the fit:

$$\hat{\epsilon}_t = y_t - f(x_t, \hat{\theta})$$

- 2) Draw bootstrap pseudodata:

$$y_t^* = f(x_t, \hat{\theta}) + \epsilon_t^*$$

where ϵ_t^* are i.i.d. from distribution \hat{F} , where the residuals from the original fit are centered at zero.

3) Fit $\hat{\theta}^*$ by nonlinear least-squares.

4) Repeat B times, yielding bootstrap replications

$$\hat{\theta}^{*1}, \hat{\theta}^{*2}, \dots, \hat{\theta}^{*B}$$

One can then characterize the EDF and calculate confidence intervals for each parameter:

$$\theta \in [EDF^{-1}(\alpha/2), EDF^{-1}(1 - \alpha/2)]$$

The jackknife alternative is to perform the bootstrap on the data-predictor vector, i.e. eliminating a certain number of datapoints.

If the residuals are correlated or have non-constant variance the latter is recommended. This may be the case in qPCR data, as the variance in the low fluorescence region (ground phase) is usually much higher than in the rest of the curve.

Value

A list containing the following items:

ITER a list containing each of the results from the iterations.

CONF a list containing the confidence intervals for each item in ITER.

Each item contains subitems for the coefficients (`coef`), root-mean-squared error (`rse`), residual sum-of-squares (`rss`), goodness-of-fit measures (`gof`) and the efficiency analysis (`eff`). If `plot = TRUE`, all data is plotted as boxplots including confidence intervals.

Author(s)

Andrej-Nikolai Spiess

References

- Bates DM and Watts DG (1988).
Nonlinear regression analysis and its applications.
Wiley, Chichester, UK.
- Seber GAF and Wild CJ (1989).
Nonlinear regression.
Wiley, New York.
- Roy T (1994).
Bootstrap accuracy for non-linear regression models.
J Chemometrics, **8**: 37-44.

Examples

```
## simple bootstrapping with
## too less iterations...
par(ask = FALSE)
m1 <- pcrfit(reps, 1, 2, 14)
res <- pcrboot(m1, B = 20)
```

```
## jackknifing with leaving
## 5 datapoints out
m2 <- pcrfit(reps, 1, 2, 14)
res <- pcrboot(m2, type = "jack", njack = 5, B = 20)
```

pcrfit

Function for qPCR model fitting

Description

This is the main workhorse function of the qpcR package that fits one of the available models to qPCR data using nonlinear least-squares fitting from `nls`, with sensible starting parameters obtained from either `nls.lm`, `optim` or `genoud`.

Usage

```
pcrfit(data, cyc = 1, fluo, model = 14, do.optim = TRUE,
       opt.method = "LM", nls.method = "port",
       start = NULL, robust = FALSE, control = nls.control(),
       weights, ...)
```

Arguments

| | |
|-------------------------|---|
| <code>data</code> | the name of the dataframe containing the qPCR runs. |
| <code>cyc</code> | the column containing the cycle data. Defaults to 1. |
| <code>fluo</code> | the column containing the raw fluorescence data of the run. |
| <code>model</code> | the model to be used for the analysis. Defaults to 14. |
| <code>do.optim</code> | if <code>FALSE</code> , refinement of starting values by <code>nls.lm</code> , <code>optim</code> or <code>nls.lm</code> will be skipped. |
| <code>opt.method</code> | one of the available refinement methods. See 'Details'. |
| <code>nls.method</code> | one of the available methods in <code>nls</code> . Default is "port", which works quite well. |
| <code>start</code> | a vector of starting values that can be supplied externally. |
| <code>robust</code> | logical. If <code>TRUE</code> , robust nonlinear regression is used. See 'Details'. |
| <code>control</code> | an optional list of control settings for <code>nls</code> or <code>qpcR:::rnls</code> . |
| <code>weights</code> | a vector of weights for nonlinear fitting. Must be same lengths as the data. |
| <code>...</code> | other parameters to be passed to <code>optim</code> , <code>genoud</code> or <code>nls</code> . |

Details

The fitting procedure works as follows (hopefully ensuring maximum safeness against convergence errors):

- 1) Approximate starting values are acquired from `model$ssfct`.
- 2) Starting values are refined by any of the methods available in `optim`, the `genoud` method from the 'rgenoud' package or the Levenberg-Marquardt algorithm (`nls.lm`). The `opt.methods` can be combined to tweak the robustness, whereby the starting parameters are passed to each succeeding method, i.e. `rep("Nelder", 5)` will do 5 successive Nelder-Mead optimisations or `c("GA", "Nelder")` will pass the starting values from "GA" to "Nelder". If problems arise, "GA" has shown to be very robust (but slow!) in the refinement of starting values. Levenberg-Marquardt ("LM") is very fast and relatively reliable in many scenarios and is thus the default.
- 3) One of the possible methods from `nls` is then applied with the starting values obtained from 2).

This function is to be used at the single run level. Otherwise use `pcrbatch` or `modlist`.

The output from the `optim` methods is checked by ensuring all eigenvalues from the hessian are positive, otherwise a notice will occur.

If `robust = TRUE`, robust nonlinear fitting will be used. To do this, the internal function `qpcR:::rnls` is called which is a modification of the `nlrob` function of the 'robustbase' package. Modifications were done such that all available generic functions for objects of class 'nls' can be used on the output of `qpcR:::rnls`, such as `predict`, `confint` etc.

Value

A model of class 'nls' and 'pcrfit' with the following items attached:

| | |
|-------------------------|--|
| <code>DATA</code> | the initial data used for fitting. |
| <code>MODEL</code> | the model used for fitting. |
| <code>call2</code> | the call to <code>pcrfit</code> . |
| <code>parMat</code> | the trace of the starting values for each applied method. Can be used to track problems. |
| <code>opt.method</code> | the parameter <code>opt.method</code> . |

Author(s)

Andrej-Nikolai Spiess

References

Bioassay analysis using R.
Ritz C & Streibig JC.
J Stat Soft (2005), **12**: 1-22.

A Method for the Solution of Certain Problems in Least Squares.
K. Levenberg.
Quart Appl Math (1944), **2**: 164-168.

An Algorithm for Least-Squares Estimation of Nonlinear Parameters.
 D. Marquardt.
SIAM J Appl Math (19639), **11**: 431-441.

Examples

```
## simple l4 fit of F1.1 of the 'reps' dataset
pcrfit(reps, 1, 2, 14)

## same with five-parameter model
## use "GA" method for optim
pcrfit(reps, 1, 2, 15, opt.method = "GA")

## using BFGS and Nelder from 'optim'
pcrfit(reps, 1, 2, 15, opt.method = c("BFGS", "Nelder"))

## skip 'optim' method and supply
## own starting values
pcrfit(reps, 1, 2, 14, do.optim = FALSE, start = c(-5, -0.05, 11, 16))

## make a robust model
pcrfit(reps, 1, 2, 14, robust = TRUE)

## use weights
pcrfit(reps, 1, 2, 14, weights = 49:1)
```

pcrGOF

Summarize measures for the goodness-of-fit

Description

Calculates all implemented measures for the goodness-of-fit and returns them as a list. Works for objects of class `pcrfit`, `drc`, `lm`, `glm`, `nls` and many others...

Usage

```
pcrGOF(object, PRESS = FALSE)
```

Arguments

| | |
|---------------------|---|
| <code>object</code> | a fitted object. |
| <code>PRESS</code> | logical. If TRUE, the more calculation intensive P-square is also returned. |

Value

A list with the following components:

| | |
|----------------------|-------------------------------|
| <code>Rsqr</code> | the R-squared value. |
| <code>Rsqr.ad</code> | the adjusted R-squared value. |

| | |
|----------|--|
| AIC | the Akaike Information Criterion. |
| AICc | the bias-corrected Akaike Information Criterion. |
| BIC | the Bayesian Information Criterion. |
| resVar | the residual variance. |
| RMSE | the root-mean-squared-error. |
| prob | the chi-square fit probability. |
| P.square | the PRESS P-square, if <code>PRESS = TRUE</code> . |

Author(s)

Andrej-Nikolai Spiess

Examples

```
m <- pcrfit(reps, 1, 2, 15)
pcrGOF(m, PRESS = TRUE)
```

pcrimport

Simple qPCR data import function (i.e. from text files or clipboard).

Description

Simple wrapper function to easily import qPCR data from the clipboard (default) or tab-delimited text files.

Usage

```
pcrimport(file = "clipboard", sep = "\t", header = TRUE, quote = "",
          dec = ".", colClasses = "numeric", ...)
```

Arguments

| | |
|------------|--|
| file | the name of the file which the data are to be read from (full path). |
| sep | the field separator character. |
| header | a logical value indicating whether the file contains the names of the variables as its first line. |
| quote | the set of quoting characters. |
| dec | the character used in the file to denote decimal points. |
| colClasses | character. A vector of classes to be assumed for the columns. |
| ... | further arguments to be passed on to <code>read.table</code> . |

Details

For a more detailed description of the arguments see `read.table`.

Value

A data frame containing a representation of the data in the file.

Author(s)

Andrej-Nikolai Spiess

Examples

```
## paste some Excel data into the clipboard ###
## Not run:
temp <- pcrimport()

## End(Not run)
## from a tab-delimited text file ###
## Not run:
temp <- pcrimport("c:\temp\foo.txt")

## End(Not run)
```

pcropt1

Combinatorial elimination of plateau and ground phase cycles

Description

The estimation of PCR efficiency and calculation of initial fluorescence (F0) is analyzed by refitting the (optimized) model on subsets of the data, thereby using all possible combinations of datapoints. The estimated parameters are then collated in a dataframe. This is intended to be the prerequisite for finding the optimal datapoints that minimize the fit or exhibit the best correlation to a calibration curve.

Usage

```
pcropt1(object, fact = 3, opt = FALSE, plot = TRUE, ...)
```

Arguments

| | |
|--------|--|
| object | an object of class 'pcrfit'. |
| fact | numeric. The multiplier for the scan border. See 'Details'. |
| opt | logical. If true, model selection is applied for each combination of cycles. Beware: Slow!! |
| plot | logical. If TRUE, boxplots of the resulting values are displayed. |
| ... | other parameters to be passed on to efficiency , i.e. where to take the PCR efficiency from. |

Details

It has been shown by Rutledge (2004) that the estimation of PCR efficiency gives more realistic values when the number of plateau cycles are decreased. This paradigm is the basis for this function, but we also consider the cycles in the ground phase and all combinations between ground/plateau cycles. All datapoints between the lower border $cpD1 - fact * (cpD1 - cpD2)$ and upper border $cpD1 + fact * (cpD1 - cpD2)$ are cycled through.

Value

A dataframe with the border values, AIC, AICc, residual variance, efficiency and the estimated F(0) from the exponential and sigmoidal model.

Author(s)

Andrej-Nikolai Spiess

References

Sigmoidal curve fitting redefines quantitative real-time PCR with the prospective of developing automated high-throughput applications. Rutledge RG, *Nucleic Acids Research*, 2004, **e178**.

See Also

The function [efficiency](#) that is called by this function.

Examples

```
## Using one model throughout
## Not run:
m <- pcrfit(reps, 1, 2, 14)
pcropt1(m)

## End(Not run)

## Selecting the best model for each combination
## by Akaike weights
## Not run:
m <- pcrfit(reps, 1, 2, 14)
pcropt1(m, opt = TRUE, crit = "weights")

## End(Not run)
```

pcropt2 *Elimination of qPCR cycles with low (high) impact on fitted parameters*

Description

The qPCR curve containing n cycles is refitted $n-1$ times, each time leaving out one cycle. The difference of the new coefficients of the fit in comparison to the original coefficients is calculated and those cycles are eliminated that have a weak (strong) influence on change of coefficients. A new model is returned with the selected cycles left out.

Usage

```
pcropt2(object, plot = TRUE, which.par = "all", quan = 0.1,
        delete = c("low", "high"), ...)
```

Arguments

| | |
|-----------|--|
| object | an object of class 'pcrfit'. |
| plot | logical. If TRUE, the refitting and the final result are plotted. |
| which.par | The coefficient(s) to be analysed. Either "all" for all coefficients, or the coefficient name, i.e. "b". |
| quan | the quantile for selecting the cycles exhibiting weak (strong) influence on the coefficient estimation. |
| delete | which cycles to delete. Those with low influence on the coefficients or those with a high one. |
| ... | other parameters to be passed on to the plotting functions. |

Details

For each deletion of cycle $i = 1, \dots, n$, the qPCR data is refitted yielding new parameter estimates

$$\hat{\theta}^{*1}, \dots, \hat{\theta}^{*i}$$

The difference to the original coefficients $\hat{\theta}$ is calculated by

$$crit = \frac{|\hat{\theta} - \hat{\theta}^{*i}|}{s.e.(\hat{\theta})}$$

with s.e. = standard error. The user then chooses the cycles with $F^{-1}(p) = inf\{crit \in \mathbf{R} : F(crit) \geq p\}$ with p = the selected quantile.

Value

A new model of class 'pcrfit' and 'nls' with the corresponding cycles removed.

Author(s)

Andrej-Nikolai Spiess

References

Bates DM and Watts DG (1988).
Nonlinear regression analysis and its applications.
Wiley, Chichester, UK.

See Also

The function `pcropt1` that removes cycles sequentially from both sides of the curve.

Examples

```
m <- pcrfit(reps, 1, 2, 14)
## which cycles have low influence
## on parameter 'c' (the lower
## asymptote)?
pcropt2(m, which.par = "c", quan = 0.3, delete = "low")

## and on 'b' and 'e'?
m <- pcrfit(reps, 1, 2, 14)
pcropt2(m, which.par = c("b", "e"), quan = 0.3, delete = "low")

## very high influence on 'd'
## (upper asymptote)?
m <- pcrfit(reps, 1, 2, 14)
m2 <- pcropt2(m, which.par = c("d"), quan = 0.1, delete = "high")

## plot new model
plot(m2)
```

pcrsim

Simulation of sigmoidal qPCR data with goodness-of-fit analysis for different models

Description

Simulated sigmoidal qPCR curves are generated from an initial model to which some user-defined noise is added. One or more models can then be fit to this random data and goodness-of-fit (GOF) measures are calculated for each of the models. This is essentially a Monte-Carlo approach testing for the best model in dependence to some noise structure in sigmoidal models.

Usage

```
pcrsim(cyc = 1:30, model = 14, par = NULL, nsim = 10,
       error = 0.02, errfun = function(y) 1, plot = TRUE,
       fitmodel = NULL, select = FALSE,
       statfun = function(y) mean(y, na.rm = TRUE), ...)
```

Arguments

| | |
|-----------------------|---|
| <code>cyc</code> | the number of cycles to be simulated. |
| <code>model</code> | the initial model used for generating the simulated curves. |
| <code>par</code> | a numeric vector (required) with the parameter values for <code>model</code> . |
| <code>nsim</code> | the number of simulated curves. |
| <code>error</code> | the gaussian error used for the simulation. See 'Details'. |
| <code>errfun</code> | an optional function for the error distribution. See 'Details'. |
| <code>plot</code> | should the simulated and fitted curves be displayed? |
| <code>fitmodel</code> | a model or model list to test against the initial model. |
| <code>select</code> | if TRUE, a matrix is returned with the best model in respect to each of the GOF measures. |
| <code>statfun</code> | a function to be finally applied to all collected GOF measures, default is the average. |
| <code>...</code> | other parameters to be passed on to <code>plot</code> or <code>pcrfit</code> . |

Details

The value defined under `error` is just the standard deviation added plainly to each `yi` value from the initial model, thus generating a dataset with random homoscedastic noise. With aid of `errfun`, the distribution of the error along the `yi` values can be altered and therefore be used to generate heteroscedastic variance along the curve, so that the standard deviation is a function of the magnitude.

Example:

```
errfun = function(y) 1
same variance for all yi, as is.
```

```
errfun = function(y) y
variance as a function of the y-magnitude.
```

```
errfun = function(y) 1/y
variance as an inverse function of the y-magnitude.
```

For the effect, see 'Examples'.

Value

A list containing the following items:

| | |
|----------|--|
| cyc | same as in 'arguments'. |
| fluoMat | a matrix with the simulated qPCR data in columns. |
| coefList | a list with the coefficients from the fits for each model, as subitems. |
| gofList | a list with the GOF measures for each model, as subitems. |
| statList | a list with the GOF measures summarized by statfun for each model, as subitems. |
| modelMat | if select = TRUE, a matrix with the best model for each GOF measure and each simulation. |

Author(s)

Andrej-Nikolai Spiess

Examples

```
## generate initial model
m <- pcrfit(reps, 1, 2, 14)

## simulate homoscedastic error
## and test initial model, w3 and 15
## model on data
res <- pcrsim(cyc = 1:30, model = 14, par = coef(m),
              error = 0.2, nsim = 20, fitmodel = list(14, w3, 15))

## use heteroscedastic noise typical for
## qPCR: more noise at lower fluorescence
## Not run:
res2 <- pcrsim(cyc = 1:30, model = 14, par = coef(m),
               error = 0.01, errfun = function(y) 1/y,
               nsim = 20, fitmodel = list(14, w3, 15))

## End(Not run)

## get 95% confidence interval for
## the models GOF in question (14, w3, 15)
## Not run:
res <- pcrsim(cyc = 1:30, model = 14, par = coef(m),
              error = 0.2, nsim = 20, fitmodel = list(14, w3, 15),
              statfun = function(y) quantile(y, c(0.025, 0.975)))
res$statList

## End(Not run)

## show best model for each simulation
## based on different GOF measures
## Not run:
res <- pcrsim(cyc = 1:30, model = 14, par = coef(m),
```

```

                                error = 0.2, nsim = 20, fitmodel = list(14, w3, 15),
                                select = TRUE)
res$modelMat

## End(Not run)

```

plot.pcrfit

Plotting qPCR data with fitted curves/confidence bands/error bars

Description

A plotting function for data of class 'pcrfit' (single curves), 'modlist' (batch curves) or 'replst' (replicate curves) displaying the data points, the fitted curve and (if desired) confidence/prediction bands/error bars on replicates.

Usage

```

## S3 method for class 'pcrfit':
plot(x, fitted = TRUE, subset = NULL,
     confband = c("none", "confidence", "prediction"),
     errbar = c("none", "sd", "se", "conf"), add = FALSE,
     colvec = NULL, level = 0.95, ...)

```

Arguments

| | |
|----------|--|
| x | an object of class 'pcrfit', 'modlist' or 'replst'. |
| fitted | should the fitted line be displayed? |
| subset | a 2-element vector analog to xlim of the data that should be plotted. |
| confband | should confidence/prediction bands be displayed? See confint . |
| errbar | the type of error bar on the plot if replicates exist. See 'Examples'. |
| add | should the curve be added to an existing plot? |
| colvec | an optional color vector for the individual curves. |
| level | the confidence level used for confband or errbar. |
| ... | other parameters to be passed to plot or predict . |

Value

A qPCR plot. If object was of class 'replst', colour coding of the curves is done automatically.

Author(s)

Andrej-Nikolai Spiess

Examples

```
## single plot
m1 <- pcrfit(reps, 1, 2, 15)
plot(m1)

## add another plot in blue
## with 99% prediction interval
m2 <- pcrfit(reps, 1, 12, 15)
plot(m2, add = TRUE, colvec = 4, confband = "confidence", level = 0.99)

## plot a 'modlist' batch with coloring of replicates
m1 <- modlist(reps, 1, 2:13, model = 14)
plot(m1, colvec = gl(3,4))

## only the fitted curves
## and a subset of data
plot(m1, colvec = gl(3,4), type = "n", subset = c(10, 30))

## plot a 'replist'
m1 <- modlist(reps, 1, 2:9, model = 14)
r1 <- replist(m1, group = gl(2, 4))
plot(r1)

## standard deviation instead of
## replicate points
plot(r1, type = "none", errbar = "sd")
```

predict.pcrfit

Value prediction from a fitted qPCR model

Description

After fitting the appropriate model, either the raw fluorescence values can be predicted from the cycle number or *vice versa*.

Usage

```
## S3 method for class 'pcrfit':
predict(object, newdata, which = c("y", "x"),
        interval = c("none", "confidence", "prediction"),
        level = 0.95, ...)
```

Arguments

| | |
|---------|--|
| object | an object of class 'pcrfit'. |
| newdata | a dataframe containing the values to estimate from, using the same variable naming as in the fitted model. |

| | |
|----------|--|
| which | either "y" (default) for prediction of the raw fluorescence or "x" for prediction of the cycle number. |
| interval | if not "none", confidence or prediction intervals are calculated. |
| level | the confidence level. |
| ... | some methods for this generic require additional arguments. None are used in this method. |

Details

y-values (fluorescence) are estimated from `object$MODEL$expr`, x-values (cycles) are estimated from `object$MODEL$inv`. Confidence levels are calculated from the gradient of these and the variance-covariance matrix of `object` by `grad(f) %*% vcov(object) %*% grad(f)` and are based on asymptotic normality (t-distribution).

Value

A dataframe containing the estimated values and (if chosen) standard error/upper confidence limit/lower confidence limit. The gradient is attached to the dataframe and can be accessed with `attr(..., "gradient")`.

Note

The estimation of x (cycles) from fluorescence data if `which = "x"` is problematic in the asymptotic regions of the sigmoidal curves (often gives NaN, due to logarithmation of negative values) and works fairly well in the ascending part.

Author(s)

Andrej-Nikolai Spiess

Examples

```
m <- pcrfit(reps, 1, 2, 15)

## which raw fluorescence value at cycle number = 17?
predict(m, newdata = data.frame(Cycles = 17))

## cycle numbers 20:25, with 95% confidence?
predict(m, newdata = data.frame(Cycles = 20:25), interval = "confidence")

## which cycle at Fluo = 4, with 95% prediction?
predict(m, newdata = data.frame(Fluo = 4), which = "x", interval = "prediction")
```

PRESS

*Allen's PRESS (Prediction Sum-Of-Squares) statistic, aka P-square***Description**

Calculates the PRESS statistic, a leave-one-out refitting and prediction method, as described in Allen (1971). Works for any regression model with a `call` slot, an `update` and a `predict` function, hence all models of class `lm`, `glm`, `nls` and `drc` (and maybe more...). The function also returns the PRESS analog to R-square, the P-square.

Usage

```
PRESS(object, verbose = TRUE)
```

Arguments

`object` a fitted model.
`verbose` logical. If `TRUE`, iterations are displayed on the console.

Details

From a fitted model, each of the predictors $x_i, i = 1 \dots n$ is removed and the model is refitted to the $n - 1$ points. The predicted value $\hat{y}_{i,-i}$ is calculated at the excluded point x_i and the PRESS statistic is given by:

$$\sum_{i=1}^n (y_i - \hat{y}_{i,-i})^2$$

The PRESS statistic is a surrogate measure of crossvalidation of small sample sizes and a measure for internal validity. Small values indicate that the model is not overly sensitive to any single data point. The P-square value, the PRESS equivalent to R-square, is given by

$$P^2 = \frac{\sum_{i=1}^n \hat{\epsilon}_{-i}^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$$

with $\hat{\epsilon}_{-i} = y_i - \hat{y}_{-i}$.

Value

A list with the following components:

`stat` The PRESS statistic.
`residuals` a vector containing the PRESS residuals for each x_i .
`P.square` the P-square value. See 'Details'.

Note

There is also a `PRESS` function in library 'MPV' that works solely for `lm` models using the hat matrix.

Author(s)

Andrej-Nikolai Spiess

References

The relationship between variable selection and data augmentation and a method for prediction.

Allen DM.

Technometrics (1974), **16**:25-127.

The Prediction Sum of Squares as a Criterion for Selecting Predictor Variables.

Allen DM.

Technical Report Number 23 (1971), Department of Statistics, University of Kentucky.

Classical and Modern Regression with Applications.

Myers RH.

Second Edition (1990), Duxbury Press (PWS-KENT Publishing Company), 299-304.

Examples

```
## example for PCR analysis
m1 <- pcrfit(reps, 1, 2, 15)
PRESS(m1)

## compare PRESS statistic in models
## with fewer parameters
m2 <- pcrfit(reps, 1, 2, 14)
PRESS(m2)
m3 <- pcrfit(reps, 1, 2, 13)
PRESS(m3)

## example for linear regression
x <- 1:10
y <- rnorm(10, x, 0.1)
mod <- lm(y ~ x)
PRESS(mod)

## example for NLS fitting
DNase1 <- subset(DNase, Run == 1)
fm1DNase1 <- nls(density ~ SSlogis(log(conc), Asym, xmid, scal), DNase1)
res <- PRESS(fm1DNase1)

## PRESS residuals plot
barplot(res$residuals)
```

Description

A general function for the calculation of errors. Can be used for qPCR data, but any data that should be subjected to error analysis will do. The different error types can be calculated for any given expression from either replicates or from statistical summary data (mean & standard deviation). These are:

1) Monte-Carlo simulation:

For each variable in the dataset, simulated data with `nsim` samples is generated from a multivariate normal distribution using the mean and s.d. of each variable. All data is coerced into a new dataset that has the same covariance structure as the initial dataset. Each row of the simulated dataset is evaluated and statistics are calculated from the `nsim` evaluations.

2) Permutation approach:

The data of the original dataset is permuted `nperm` times by binding observations together according to `ties`. The `ties` bind observations that can be independent measurements from the same sample. In qPCR terms, this would be a real-time PCR for two different genes on the same sample. If `ties` is omitted, the observations are shuffled independently. In detail, two datasets are created for each permutation: Dataset1 samples the rows (replicates) of the data according to `ties`. Dataset2 is obtained by sampling the columns (samples), also binding columns as in `ties`. For both datasets, the permutations are evaluated and statistics are collected. The confidence interval is calculated from all evaluations of Dataset1. A p-value is calculated from all permutations that follow `perm.crit`, whereby `init` reflects the permutations of the initial data and `perm` the permutations of the randomly reallocated samples. Thus, the p-value gives a measure against the null hypothesis that the result in the initial group is just by chance. See 'Details' and 'Examples'.

3) Error propagation:

For all variables in the original dataset, mean and standard deviation are calculated. Through gaussian error propagation (first-order Taylor expansion), the propagated error is calculated using a matrix approach (See 'Details') by either omitting or including the dataset covariance structure.

Usage

```
propagate(expr, data, type = c("raw", "stat"), do.sim = TRUE, use.cov = FALSE,
          nsim = 10000, do.perm = FALSE, perm.crit = "perm > init",
          ties = NULL, nperm = 2000, alpha = 0.05, plot = TRUE, xlim = NULL, ...)
```

Arguments

| | |
|----------------------|---|
| <code>expr</code> | an expression, such as <code>expression(x/y)</code> . |
| <code>data</code> | a dataframe or matrix containing either a) the replicates in columns or b) the means in the first row and the standard deviations in the second row. The variable names must be defined in the column headers. |
| <code>type</code> | either <code>raw</code> if replicates are given, or <code>stat</code> if means and standard deviations are supplied. |
| <code>do.sim</code> | logical. Should Monte Carlo error analysis be applied? |
| <code>use.cov</code> | logical or variance-covariance matrix with the same column descriptions and column order as <code>data</code> . If <code>TRUE</code> together with replicates, the covariances are calculated from these and used within the Monte-Carlo simulation and error propagation. If <code>type = "stat"</code> , a square variance-covariance matrix can be |

| | |
|------------------------|--|
| | supplied in the right dimensions (n x n, n = number of variables). If <code>FALSE</code> , the Monte-Carlo simulation and error propagation use only the diagonal (variances). |
| <code>nsim</code> | the number of simulations to be performed, minimum is 5000. |
| <code>do.perm</code> | logical. Should permutation error analysis be applied? |
| <code>perm.crit</code> | a character string defining the null hypothesis for the permutation p-value. See 'Details'. |
| <code>ties</code> | a vector defining the columns that should be tied together for the permutations. See 'Details'. |
| <code>nperm</code> | the number of permutations to be performed. |
| <code>alpha</code> | the confidence level. |
| <code>plot</code> | logical. Should histograms with confidence intervals (in blue) be plotted for all analyses? |
| <code>xlim</code> | the x-axis limits for tweaking the histograms (in case of severe skewness). |
| <code>...</code> | other parameters to be supplied to <code>hist</code> , <code>boxplot</code> or <code>abline</code> . |

Details

The propagated error is calculated by gaussian error propagation. Often omitted, but important in models where the variables are dependent (i.e. linear regression), is the second covariance term.

$$\sigma_Y^2 = \sum_i \left(\frac{\partial f}{\partial X_i} \right)^2 \sigma_i^2 + \sum_{i \neq j} \sum_{j \neq i} \left(\frac{\partial f}{\partial X_i} \right) \left(\frac{\partial f}{\partial X_j} \right) \sigma_{ij}$$

`propagate` calculates the propagated error either with or without covariances by using the matrix representation

$$C_Y = F_X C_X F_X^T$$

with C_Y = the propagated error, F_X = the p x n matrix with the results from the partial derivatives, C_X = the p x p variance-covariance matrix and F_X^T = the n x p transposed matrix of F_X . Depending on the input formula, the error propagation may result in an error that is not normally distributed. The Monte Carlo simulation, starting with normal distributions of the variables, can clarify this. The plots obtained from this function will also clarify this potential caveat. A high tendency from deviation of normality is encountered in formulas where the error of the denominator is relatively high or in exponential models where the exponent has a high error. This is one of the problems that is inherent in real-time PCR analysis, as the classical ratio calculation with efficiencies (i.e. by the delta-ct method) is usually of the exponential type.

The criterium for the permutation p-value (`perm.crit`) has to be defined by the user. For example, let's say we calculate some value 0.2 which is a ratio between two groups. We would hypothesize that by randomly reallocating the values between the groups the mean values are not equal or smaller than in the initial data. We would thus define `perm.crit` as "`perm < init`" meaning that we want to test if the mean of the initial data (`init`) is frequently smaller than by the randomly allocated data (`perm`).

Value

A list with the following components:

| | |
|-------------------------|---|
| <code>mean.Sim</code> | the mean from the Monte Carlo simulated data. |
| <code>sd.Sim</code> | the standard deviation from the Monte Carlo simulated data. |
| <code>med.Sim</code> | the median of the Monte Carlo simulated data. |
| <code>mad.Sim</code> | the median average deviation of the Monte Carlo simulated data. |
| <code>data.Sim</code> | the Monte Carlo simulated data with the evaluations in the last column. |
| <code>conf.Sim</code> | the confidence values for the evaluations from <code>data.Sim</code> . |
| <code>mean.Perm</code> | the mean from all permutations. |
| <code>sd.Perm</code> | the standard deviation from all permutations. |
| <code>med.Perm</code> | the median from all permutations. |
| <code>mad.Perm</code> | the median average deviation from all permutations. |
| <code>data.Perm</code> | the data of the permuted observations and samples with the corresponding evaluations and the decision according to <code>perm.crit</code> . See 'Examples'. |
| <code>conf.Perm</code> | the confidence interval obtained from all permutations. |
| <code>pval.Perm</code> | the permutation p-value AGAINST the hypothesis defined in <code>perm.crit</code> . |
| <code>eval.Prop</code> | the result from evaluating the expression with the mean values. |
| <code>error.Prop</code> | the propagated error. |
| <code>conf.Prop</code> | the confidence values of the propagated error, assuming normality. |
| <code>deriv.Prop</code> | a list containing the partial derivatives expressions for each variable. |
| <code>covMat</code> | the covariance matrix used for the Monte-Carlo simulation and error propagation. |

Note

A more elaborate description of the different error types can be found under www.dr-spiess.de/qpcR/errors.pdf.

Author(s)

Andrej-Nikolai Spiess

References

Error propagation (in general):

Taylor JR (1996). An Introduction to error analysis. University Science Books, New York.

A very good technical paper describing error propagation by matrix calculation can be found under www.nada.kth.se/~kai-a/papers/arrasTR-9801-R3.pdf.

Error propagation (in qPCR):

Nordgard O *et al.* (2006). Error propagation in relative real-time reverse transcription polymerase

chain reaction quantification models: The balance between accuracy and precision. *Analytical Biochemistry*, **356**, 182-193.

Hellemans J *et al.* (2007). qBase relative quantification framework and software for management and analysis of real-time quantitative PCR data. *Genome Biology*, **8**: R19.

Multivariate normal distribution:

Ripley BD (1987). *Stochastic Simulation*. Wiley. Page 98.

Testing for normal distribution:

Thode Jr. HC (2002). *Testing for Normality*. Marcel Dekker, New York.

Royston P (1992). Approximating the Shapiro-Wilk W-test for non-normality. *Statistics and Computing*, **2**, 117-119.

See Also

Function `ratiocalc` for error analysis within qPCR ratio calculation.

Examples

```
## From summary data just calculate
## Monte-Carlo and propagated error.
EXPR <- expression(x/y)
x <- c(5, 0.1)
y <- c(1, 0.01)
DF <- cbind(x, y)
res <- propagate(expr = EXPR, data = DF, type = "stat")

## Do Shapiro-Wilks test on Monte Carlo evaluations
## !maximum 5000 datapoints can be used!
## => p.value indicates normality
shapiro.test(res$data.Sim[, 3][1:5000])
## How about a graphical analysis:
qqnorm(res$data.Sim[, 3])

## Using raw data
## bring all vectors to same
## length with NA's.
## Do permutations (swap x and y values)
EXPR <- expression(x*y)
x <- c(2, 2.1, 2.2, 2, 2.3, 2.1)
y <- c(4, 4, 3.8, 4.1, 3.1, 3)
DF <- cbind(x, y)
res <- propagate(EXPR, DF, type = "raw", do.perm = TRUE)
## Have a look at permutation result
res$data.Perm

## For replicate data, using relative
## quantification ratio from qPCR.
## How good is the estimation of the propagated error?
## Done without using covariance in the
```

```

## calculation and simulation.
## STRONG deviation from normality!
## cp's and efficiencys are tied together
## because they are two observations on the
## same sample!
EXPR <- expression((E1^cp1)/(E2^cp2))
E1 <- c(1.73, 1.75, 1.77)
cp1 <- c(25.77,26.14,26.33)
E2 <- c(1.72,1.68,1.65)
cp2 <- c(33.84,34.04,33.33)
DF <- cbind(E1, cp1, E2, cp2)
res <- propagate(EXPR, DF, type = "raw", do.sim = TRUE,
                 do.perm = FALSE)

par(ask = TRUE)
shapiro.test(res$data.Sim[, 3][1:5000])
qqnorm(res$data.Sim[, 3])

## Same setup as above but also
## using a permutation approach
## for resampling confidence interval
## Cp's and efficiencys are tied together
## because they are two observations on the
## same sample!
## Similar to what REST2008 software does.
res <- propagate(EXPR, DF, type = "raw", do.sim = TRUE,
                 perm.crit = "perm < init", do.perm = TRUE,
                 ties = c(1, 1, 2, 2))
## 95% confidence interval for the ratio
res$conf.Perm

## Using covariances in calculation and simulation
res2 <- propagate(EXPR, DF, type = "raw", do.sim = TRUE,
                  perm.crit = "perm < init", do.perm = TRUE,
                  ties = c(1, 2, 1, 2), use.cov = TRUE)

## Proof that covariance of Monte-Carlo
## simulated dataset is the same as from
## initial data
res2$covMat
cov(res2$data.Sim)

## Error propagation in a linear model
## using the covariance matrix from summary.lm
## Estimate error of y for x = 7
x <- 1:10
set.seed(123)
y <- x + rnorm(10, 0, 1) ##generate random data
mod <- lm(y ~ x)
summ <- summary(mod)
## make matrix of parameter estimates and standard error
DF <- t(coef(summ)[, 1:2])
colnames(DF) <- c("b", "m")

```

```

CM <- vcov(mod) ## take var-cov matrix
colnames(CM) <- c("b", "m")
propagate(expression(m*7 + b), DF, type = "stat", use.cov = CM)

## In a x/y regime, when does the propagated error start to
## deviate from normality if error of denominator increases?
## Watch increasing skewness of histogram!
## Not run:
x <- c(5, 1)
for (i in seq(0.01, 0.5, by = 0.01)) {
  y <- c(1, i)
  DF <- cbind(x, y)
  res <- propagate(expression(x/y), DF, type = "stat",
                    do.sim = TRUE, plot = FALSE)
  hist(res$data.Sim[, 3], nclass = 100, main = paste("sd(y):", i))
}

## End(Not run)

```

Description

A summary of all available models implemented in this package.

Usage

```

l5
l4
l3
b5
b4
b3
w4
w3
baro5
expGrowth

```

Details

The following nonlinear models are implemented:

l5:

$$f(x) = c + \frac{d - c}{(1 + \exp(b(\log(x) - \log(e))))^f}$$

l4:

$$f(x) = c + \frac{d - c}{1 + \exp(b(\log(x) - \log(e)))}$$

l3:

$$f(x) = \frac{d}{1 + \exp(b(\log(x) - \log(e)))}$$

b5:

$$f(x) = c + \frac{d - c}{(1 + \exp(b(x - e)))^f}$$

b4:

$$f(x) = c + \frac{d - c}{1 + \exp(b(x - e))}$$

b3:

$$f(x) = \frac{d}{1 + \exp(b(x - e))}$$

w4:

$$f(x) = c + (d - c)\exp(-\exp(b(\log(x) - \log(e))))$$

w3:

$$f(x) = d\exp(-\exp(b(\log(x) - \log(e))))$$

expGrowth:

$$f(x) = a * \exp(b * x) + c$$

baro5:

$$f(x) = c + \frac{d - c}{1 + f\exp(b1(\log(x) - \log(e))) + (1 - f)\exp(b2(\log(x) - \log(e)))}$$

with

$$f = \frac{1}{1 + \exp((2b1b2/|b1 + b2|)(\log(x) - \log(e)))}$$

The functions are defined as a list containing the following items:

\$expr the function as an expression for the fitting procedure.
 \$fct the function defined as $f(x, \text{parm})$.
 \$ssfct the self-starter function.
 \$d1 the first derivative function.
 \$d2 the second derivative function.
 \$inv the inverse function.
 \$expr.grad the function as an expression for gradient calculation.
 \$inv.grad the inverse functions as an expression for gradient calculation.
 \$parnames the parameter names.
 \$name the function name.
 \$type the function type as a character string.

Author(s)

Andrej-Nikolai Spiess

Examples

```

m1 <- pcrfit(reps, 1, 2, b3)
m2 <- pcrfit(reps, 1, 2, b5)
m3 <- pcrfit(reps, 1, 2, w4)

## get the second derivative
## curve of m2
d2 <- b5$d2(m2$DATA[, 1], coef(m2))
plot(m2)
lines(d2, col = 2)

```

| | |
|-----------|---|
| ratiocalc | <i>Calculation of ratios/propagated errors/confidence intervals/permutation p-values from qPCR runs with/without reference data</i> |
|-----------|---|

Description

For multiple qPCR data from type 'pcrbatch', this function calculates ratios between two samples, using normalization against a reference gene, if supplied. The input can be single qPCR data or (more likely) data containing replicates. Errors and confidence intervals for the obtained ratios can be calculated by Monte-Carlo simulation, a permutation approach similar to the popular REST software and by (first-order) error propagation. Statistical significance for the ratios is calculated by a permutation approach of randomly reallocated vs. non-reallocated data. See 'Details'.

Usage

```

ratiocalc(data, group = NULL, which.eff = c("sig", "sli", "exp"),
          type.eff = c("individual", "mean.single", "median.single",
                      "mean.pair", "median.pair"),
          which.cp = c("cpD2", "cpD1", "cpE", "cpR", "cpT", "Cy0"),
          ...)

```

Arguments

| | |
|-----------|--|
| data | multiple qPCR data generated by pcrbatch . |
| group | a character vector defining the replicates (if any) as well as target and reference data. See 'Details'. |
| which.eff | efficiency calculated by which method. Defaults to sigmoidal fit. See output of pcrbatch . Alternatively, a fixed numeric value between 1 and 2 that is used for all runs. |
| type.eff | type of efficiency to be supplied for the error analysis. See 'Details'. |
| which.cp | type of crossing point to be used for the analysis. See output of efficiency . |
| ... | other parameters to be passed to propagate . |

Details

The replicates of the 'pcrbatch' data columns are to be defined as a character vector with the following abbreviations:

"gs": gene-of-interest in target sample
 "gc": gene-of-interest in control sample
 "rs": reference gene in target sample
 "rc": reference gene in control sample

There is no distinction between the different runs of the same sample, so that three different runs of a gene of interest in a target sample are defined as c("gs", "gs", "gs"). The error analysis calculates statistics from ALL replicates, so that a further sub-categorization of runs seems superfluous.

Examples:

No replicates: NULL.

2 runs with 2 replicates each, no reference: c("gs", "gs", "gs", "gs", "gc", "gc", "gc", "gc").

1 run with two replicates each and reference data: c("gs", "gs", "gc", "gc", "rs", "rs", "rc", "rc").

`type.eff` defines the pre-processing of the efficiencies before being transferred to `propagate`. The qPCR community sometimes uses single efficiencies, or averaged over replicates etc., so that different settings were implemented. In detail, these are the following:

"individual": The individual efficiencies from each run are used.

"mean.single": Efficiencies are averaged over all replicates.

"median.single": Same as above but median instead of mean.

"mean.pair": Efficiencies are averaged from all replicates of target sample and control.

"median.pair": Same as above but median instead of mean.

The ratios are calculated according to the following formulas:

Without reference PCR:

$$\frac{E.gc^{cp.gc}}{E.gs^{cp.gs}}$$

With reference PCR:

$$\frac{E.gc^{cp.gc}}{E.gs^{cp.gs}} \cdot \frac{E.rs^{cp.rs}}{E.rc^{cp.rc}}$$

The permutation approach permutes crossing point and efficiency replicates within sample and control groups. The sample runs and control runs (and their respective efficiencies) are tied together, which is similar to the popular REST software approach ("pairwise-reallocation test"). Ratios are calculated for each permutation and compared to ratios obtained if samples were randomly reallocated from the sample to the control group. A p-value is calculated from all permutations in which the reallocation gave a higher/lower ratio than the original data. The resulting p-value is thus an indication for the significance AGAINST the null hypothesis that ratios calculated by permutation are just by chance.

Confidence values are returned for all three methods (Monte Carlo, permutation, error propagation) as follows:

Monte-Carlo: From the evaluations of the Monte-Carlo simulated data.
 Permutation: From the evaluations of the within-group permuted data.
 Propagation: From the propagated error, assuming normality.

Value

A list with the following components:
 The complete output from `propagate`, attached with the data that was transferred to `propagate` for the error analysis as item `data`.

Note

The error calculated from qPCR data by `propagate` often seems quite high. This largely depends on the error of the exponent (i.e. threshold cycles) of the exponential function. The error usually decreases when setting `use.cov = TRUE` in the `...` part of the function. It can be debated anyhow, if the variables 'efficiency' and 'threshold cycles' have a covariance structure. As the efficiency is deduced at the second derivative maximum of the sigmoidal curve, variance in the second should show an effect on the first, such that the use of a var-covar matrix might be feasible. It is also commonly encountered that the propagated error is much higher when using reference data, as the number of partial derivative functions increases.

Author(s)

Andrej-Nikolai Spiess

References

Livak KJ *et al.* (2001) Analysis of relative gene expression data using real-time quantitative PCR and the 2(-Delta Delta C(T)) method. *Methods*, **25**: 402-428.
 Tichopad A *et al.* (2003) Standardized determination of real-time PCR efficiency from a single reaction set-up. *Nucleic Acids Res*, **31**: e122.
 Liu W & Saint DA (2002) Validation of a quantitative method for real time PCR kinetics. *Biochem Biophys Res Commun*, **294**: 347-53.
 Pfaffl M *et al.* (2002) Relative expression software tool (REST) for group-wise comparison and statistical analysis of relative expression results in real-time PCR. *Nucl Acids Res*, **30**: e36.

Examples

```
## Only target sample and control,
## no reference, 4 replicates each
## individual efficiencies for error
## calculation
DAT <- pcrbatch(reps, 2:9, 14)
GROUP <- c("gs", "gs", "gs", "gs", "gc", "gc", "gc", "gc")
res <- ratiocalc(DAT, GROUP, which.eff = "sli",
                type.eff = "individual", which.cp = "cpD2")

## Typical for using individual efficiencies,
## this inflates the error. 95% confidence intervals
## include 1 (no differential regulation) and errors are
## also extremely high (over 100%).
```

```

res$conf.Sim
res$conf.Perm
res$error.Prop/res$eval.Prop
res$pval.Perm

## Gets better using averaged efficiencies
## over all replicates
res2 <- ratiocalc(DAT, GROUP, which.eff = "sli",
                  type.eff = "mean.single", which.cp = "cpD2")
res2$conf.Sim
res2$conf.Perm
res2$error.Prop/res2$eval.Prop

## p-value indicates significant
## upregulation in comparison to randomly reallocated
## samples (similar to REST software)
res2$pval.Perm

## Using reference data.
## Toy example is same data as above
## but replicated as reference such
## that the ratio should be 1.
## Not run:
DAT2 <- pcrbatch(reps, c(2:9, 2:9), 14)
GROUP2 <- c("gs", "gs", "gs", "gs",
            "gc", "gc", "gc", "gc",
            "rs", "rs", "rs", "rs",
            "rc", "rc", "rc", "rc")
res3 <- ratiocalc(DAT2, GROUP2, which.eff = "sli",
                  type.eff = "mean.single", which.cp = "cpD2")
res3$conf.Sim
res3$conf.Perm
res3$error.Prop/res3$eval.Prop
res3$pval.Perm

## End(Not run)

## Same as above, but reference data
## is mirrored such that the ratio
## is squared.
DAT3 <- pcrbatch(reps, c(2:9, 9:2), 14)
GROUP3 <- c("gs", "gs", "gs", "gs",
            "gc", "gc", "gc", "gc",
            "rs", "rs", "rs", "rs",
            "rc", "rc", "rc", "rc")
res4 <- ratiocalc(DAT3, GROUP3, which.eff = "sli",
                  type.eff = "mean.single", which.cp = "cpD2")
res4$conf.Sim
res4$conf.Perm
res4$error.Prop/res4$eval.Prop
res4$pval.Perm

## Example without replicates

```

```

## => no Monte-Carlo and permutations
## and no plots.
DAT <- pcrbatch(reps, 2:5, 14)
GROUP <- c("gs", "gc", "rs", "rc")
res5 <- ratiocalc(DAT, GROUP, which.eff = "sli",
                  type.eff = "individual", which.cp = "cpD2")

res5$conf.Sim
res5$conf.Perm
res5$error.Prop/res4$eval.Prop
res5$pval.Perm

## Compare 'propagate' to REST software
## using the data from the REST 2008
## manual (http://rest.gene-quantification.info/),
## Have to create dataframe with values as we do
## not use 'pcrbatch', but external cp's & eff's!
## Ties define random reallocation of crossing points
## keeping controls and samples together.
## See help for 'propagate'.
## Not run:
EXPR <- expression((2.01^(cp.gc - cp.gs)/1.97^(cp.rc - cp.rs)))
cp.rc <- c(26.74, 26.85, 26.83, 26.68, 27.39, 27.03, 26.78, 27.32, NA, NA)
cp.rs <- c(26.77, 26.47, 27.03, 26.92, 26.97, 26.97, 26.07, 26.3, 26.14, 26.81)
cp.gc <- c(27.57, 27.61, 27.82, 27.12, 27.76, 27.74, 26.91, 27.49, NA, NA)
cp.gs <- c(24.54, 24.95, 24.57, 24.63, 24.66, 24.89, 24.71, 24.9, 24.26, 24.44)
DAT <- cbind(cp.rc, cp.rs, cp.gc, cp.gs)
res6 <- propagate(EXPR, DAT, do.sim = TRUE, do.perm = TRUE,
                  perm.crit = "perm > init", ties = c(1, 2, 1, 2))

res6$conf.Sim
res6$conf.Perm
res6$eval.Prop
res6$error.Prop
res6$pval.Perm

## End(Not run)

## Does error propagation in qPCR quantitation make sense?
## In ratio calculations based on (E1^cp1)/(E2^cp2),
## only 2% error in each of the variables result in
## over 50% propagated error!
## Not run:
x <- NULL
y <- NULL
for (i in seq(0, 0.1, by = 0.01)) {
  E1 <- c(1.7, 1.7 * i)
  cp1 <- c(15, 15 * i)
  E2 <- c(1.7, 1.7 * i)
  cp2 <- c(18, 18 * i)
  DF <- cbind(E1, cp1, E2, cp2)
  res <- propagate(expression((E1^cp1)/(E2^cp2)), DF, type = "stat", plot = FALSE)
  x <- c(x, i * 100)
  y <- c(y, (res$error.Prop/res$eval.Prop) * 100)
}

```

```
plot(x, y, xlim = c(0, 10), lwd = 2, xlab = "c.v. [%]", ylab = "c.v. (prop) [%]")
## End(Not run)
```

replist

Amalgamation of single data models to a model containing replicates

Description

Starting from a 'modlist' containing qPCR models from single data, `replist` amalgamates the models according to the grouping structure as defined in `group`. The result is a 'modlist' with models obtained from fitting the replicates by `pcrfit`.

Usage

```
replist(object, group)
```

Arguments

| | |
|---------------------|--|
| <code>object</code> | an object of class 'modlist'. |
| <code>group</code> | a vector defining the replicates for each group. |

Details

As being defined by `group`, the raw data of the curves are averaged and starting values for the averaged values are calculated by `optim`. Finally, a `nls` model is built from the `stacked` raw data values using these starting values.

Value

An object of class 'modlist' containing the replicate models of class 'nls'/'pcrfit'.

Author(s)

Andrej-Nikolai Spiess

See Also

`modlist`, `pcrfit`.

Examples

```
## Not run:
ml <- modlist(reps, model = 14)
rl <- replist(ml, group = gl(7, 4))
plot(rl)
summary(rl[[1]])

## End(Not run)
```

`reps`*qPCR dilution experiment with replicates (Roche Lightcycler)*

Description

A dilution experiment with seven 10-fold dilutions of the cDNA, and four replicates for each dilution.

Usage

```
data(reps)
```

Format

A data frame with the PCR cycles and 28 qPCR runs with four replicates of seven 10-fold dilutions. The replicates are defined by F1.1 - F1.4 (first dilution), F2.1 - F2.4 (second dilution) etc.

Details

The real-time PCR was conducted with primers for the S27a housekeeping gene in a Lightcycler 1.0 instrument (Roche Diagnostics).

Source

Andrej-Nikolai Spiess & Nadine Mueller, Institute for Hormone and Fertility Research, Hamburg, Germany.

Examples

```
data(reps)
m1 <- pcrfit(reps, 1, 2, 15)
plot(m1)
```

`reps2`*Another qPCR dilution experiment with replicates (Roche Lightcycler)*

Description

A dilution experiment of two different cDNAs with five 4-fold dilutions of the cDNA, and three replicates for each dilution.

Usage

```
data(reps2)
```

Format

A data frame with the PCR cycles and 30 qPCR runs with three replicates of five 4-fold dilutions. The replicates are defined by FX.Y.Z (X = cDNA number, Y = dilution number, Z = replicate number).

Details

The real-time PCR was conducted with primers for the S27a housekeeping gene in a Lightcycler 1.0 instrument (Roche Diagnostics).

Source

Heike Cappallo-Obermann, Bone Marrow Transplantation Unit, University Hospital Hamburg-Eppendorf.

Examples

```
data(reps2)
m1 <- pcrfit(reps2, 1, 2, 13)
plot(m1)
```

| | |
|-------|--|
| reps3 | <i>A qPCR dilution experiment with replicates (Stratagene MX-Pro3000P)</i> |
|-------|--|

Description

A high quality dilution experiment with six 4-fold dilutions of the cDNA, and three replicates for each dilution.

Usage

```
data(reps3)
```

Format

A data frame with the PCR cycles and 21 qPCR runs with three replicates of six 4-fold dilutions. The replicates are defined by FX.Y (X = dilution number, Y = replicate number).

Details

The real-time PCR was conducted with primers for the S27a housekeeping gene in a MXPro3000P instrument (Stratagene). Data was ROX-normalized, but without smoothing.

Source

Heike Cappallo-Obermann, Bone Marrow Transplantation Unit, University Hospital Hamburg-Eppendorf.

Examples

```
## on single data
data(reps3)
m1 <- pcrfit(reps3, 1, 2, 15)
plot(m1)
```

resplot

A residuals bar-plot with colour-coded bars

Description

A simple plotting function which displays a barplot of the residuals of any object from which `residuals` can be extracted. The bars are colour-coded with heat colours proportional to the residual value.

Usage

```
resplot(object, ...)
```

Arguments

`object` either residuals (numeric) or any fitted object, i.e. of class `nls`, `lm`, `drc` etc.
`...` any other parameters to be passed to `barplot`.

Value

A plot as described above.

Author(s)

Andrej-Nikolai Spiess

Examples

```
## create 15 model and plot residuals
m1 <- pcrfit(reps, 1, 2, 15)
resplot(m1)

## compare to 14 model (added smaller bars)
m2 <- pcrfit(reps, 1, 2, 14)
resplot(m2, add = TRUE, width = 0.5, space = c(1.4, 0.9))
```

| | |
|--------|--|
| resVar | <i>Residual variance of a fitted model</i> |
|--------|--|

Description

Calculates the residual variance for objects of class `nls`, `lm`, `glm`, `drc` or any other models from which `coef` and `residuals` can be extracted.

Usage

```
resVar(object)
```

Arguments

`object` a fitted model.

Value

The residual variance of the fit.

Author(s)

Andrej-Nikolai Spiess

Examples

```
m <- pcrfit(reps, 1, 2, 15)
resVar(m)
```

| | |
|------|--|
| RMSE | <i>Root-mean-squared-error of a fitted model</i> |
|------|--|

Description

Calculates the root-mean-squared-error (RMSE) for objects of class `nls`, `lm`, `glm`, `drc` or any other models from which `residuals` can be extracted.

Usage

```
RMSE(object, which = NULL)
```

Arguments

`object` a fitted model.
`which` the part of the curve to be used for RMSE calculation. If not defined, the complete curve is used.

Value

The root-mean-squared-error from the fit or a part thereof.

Author(s)

Andrej-Nikolai Spiess

Examples

```
## for a part of the curve
m <- pcrfit(reps, 1, 2, 15)
RMSE(m, 10:15)
```

Rsq

R-square value of a fitted model

Description

Calculates the R-square value for objects of class `nls`, `lm`, `glm`, `drc` or any other models from which `fitted` and `residuals` can be extracted.

Usage

```
Rsq(object)
```

Arguments

`object` a fitted model.

Details

Uses the most general definition of R-square:

$$R^2 \equiv 1 - \frac{RSS}{TSS}$$

where

$$RSS = \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

and

$$TSS = \sum_{i=1}^n (y_i - \bar{y})^2$$

Value

The R-square value of the fit.

Author(s)

Andrej-Nikolai Spiess

Examples

```
m <- pcrfit(reps, 1, 2, 15)
Rsq(m)
```

Rsq.ad

Adjusted R-square value of a fitted model

Description

Calculates the adjusted R-square value for objects of class `nls`, `lm`, `glm`, `drc` or any other models from which `fitted`, `residuals` and `coef` can be extracted.

Usage

```
Rsq.ad(object)
```

Arguments

`object` a fitted model.

Details

Calculates the adjusted R^2 by

$$R_{adj}^2 = 1 - \frac{n-1}{n-p} * (1 - R^2)$$

with n = sample size, p = number of regressors and R^2 = R-square value.

Value

The adjusted R-square value of the fit.

Author(s)

Andrej-Nikolai Spiess

Examples

```
## single model
m <- pcrfit(reps, 1, 2, 15)
Rsq.ad(m)

## compare different models with increasing
## number of parameters
ml <- lapply(list(13, 14, 15), function(x) pcrfit(reps, 1, 2, x))
sapply(ml, function(x) Rsq.ad(x))
```

`Rsq.cor`*R-square derived from correlation of data and fitted values*

Description

Calculates the R-square value for objects of class `nls`, `lm`, `glm`, `drc` or any other models from which `fitted` and `residuals` can be extracted. The value is calculated as a correlation measure. See 'Details'.

Usage

```
Rsq.cor(object)
```

Arguments

`object` a fitted model.

Details

Calculates the R-square by

$$R_{cor}^2 = cor(y_i, \hat{y}_i)^2$$

Value

The R-square value of the fit.

Author(s)

Andrej-Nikolai Spiess

References

Summarizing the predictive power of a generalized linear model.
B. Zheng & A. Agresti
Statist. Med. (2000), **19**: 1771 - 1781.

Examples

```
m <- pcrfit(reps, 1, 2, 15)
Rsq.cor(m)

## compare to 'standard' R-square
Rsq(m)
```

RSS

Residual sum-of-squares of a fitted model

Description

Calculates the residual sum-of-squares for objects of class `nls`, `lm`, `glm`, `drc` or any other models from which `residuals` can be extracted.

Usage

```
RSS(object)
```

Arguments

`object` a fitted model.

Value

The residual sum-of-squares from the fit.

Author(s)

Andrej-Nikolai Spiess

Examples

```
m <- pcrfit(reps, 1, 2, 15)
RSS(m)
```

`rutledge`*qPCR dilution experiment with replicates and different runs from Rutledge et al (Biorad Opticon)*

Description

A dilution experiment of a 102 bp amplicon with six 10-fold dilutions, with four replicates each in five independent runs.

Usage

```
data(rutledge)
```

Format

A data frame with the PCR cycles and the six 10-fold dilutions, with four replicates each and five independent runs, organized as X.Ry.Z, with x = the dilution number, y = the run number and z = the replicate number.

Details

The real-time PCR was conducted for a 102 bp amplicon in an Opticon2 instrument.

Source

Supplemental data 1 to the paper.

References

Sigmoidal curve-fitting redefines quantitative real-time PCR with the prospective of developing automated high-throughput applications. Rutledge RG, *Nucleic Acids Research*, 2004, **32**, e178.

Examples

```
data(rutledge)
m1 <- pcrfit(rutledge, 1, 2, 15)
plot(m1)
```

S27

PCR data from 15 different testicular biopsies (Roche Lightcycler)

Description

The data was obtained from cDNA reverse transcribed from the isolated RNA of 15 different testicular biopsies.

Usage

```
data(S27)
```

Format

A data frame with cycle number and 15 different qPCR runs (F1 - F15).

Details

The real-time PCR was conducted with primers for the S27a housekeeping gene in a Lightcycler 1.0 instrument (Roche Diagnostics).

Source

Andrej-Nikolai Spiess & Caroline Feig, Department of Andrology, University Hospital Hamburg, Germany.

Examples

```
data(S27)
m1 <- pcrfit(S27, 1, 2, 15)
plot(m1)
```

`sliwin`*Calculation of PCR efficiency by the window-of-linearity method*

Description

A linear model is fit to a sliding window of the logarithmized raw fluorescence and the regression coefficient is calculated. At the point of maximum regression (log-linear range), the PCR efficiency is calculated.

Usage

```
sliwin(object, wsize = 5, border = 7, plot = TRUE)
```

Arguments

| | |
|---------------------|---|
| <code>object</code> | an object of class 'pcrfit'. |
| <code>wsize</code> | the size of the sliding window, default is 5. |
| <code>border</code> | the -/+ border from the second derivative maximum cycle in which to do the fitting procedure. |
| <code>plot</code> | if TRUE the result is plotted, if FALSE the result is displayed on the console. |

Details

To avoid fits with a high R-squared in the baseline region, the second derivative maximum is taken as a fixpoint. This value is consequently always near to the exponential region of the data and avoids the problem above. The efficiency is calculated by $E = \exp(\text{slope})$, as the transformed raw data was based on the natural logarithm. The initial template fluorescence (F0) is thus calculated by $F0 = \exp(\text{intercept})$.

Value

A list with the following components:

| | |
|-------------------|---|
| <code>eff</code> | the (maximal) PCR efficiency found within the sliding window. |
| <code>rmax</code> | the maximum R-squared. |
| <code>init</code> | the initial template fluorescence F0. |

Author(s)

Andrej-Nikolai Spiess

References

Assumption-free analysis of quantitative real-time polymerase chain reaction (PCR) data. Ramak-ers C et al., *Neuroscience Letters*, 2003, **339**, 62-66.

Examples

```
m <- pcrfit(reps, 1, 2, 15)
sliwin(m)
```

| | |
|---------------|--|
| update.pcrfit | <i>Updating and refitting a qPCR model</i> |
|---------------|--|

Description

Updates and re-fits a model of class 'pcrfit'.

Usage

```
## S3 method for class 'pcrfit':
update(object, ..., evaluate = TRUE)
```

Arguments

| | |
|----------|---|
| object | a fitted model of class 'pcrfit'. |
| ... | arguments to alter in object. |
| evaluate | logical. If TRUE, model is re-fit; otherwise an unevaluated call is returned. |

Value

An updated model of class 'pcrfit' and 'nls'.

Author(s)

Andrej-Nikolai Spiess

See Also

The function [pcrfit](#) in this package.

Examples

```
m <- pcrfit(reps, 1, 2, 14)

## update model
update(m, model = baro5)

## update qPCR run
update(m, fluo = 20)

## update data
update(m, data = guescini1)

## update 'optim' method
update(m, opt.method = "GA")
```

Index

*Topic **IO**

pcrimport, 41

*Topic **distribution**

propagate, 52

*Topic **file**

pcrimport, 41

*Topic **htest**

propagate, 52

*Topic **models**

AICc, 2

akaike.weights, 3

batchstat, 4

batschetal, 5

BIC, 6

calib, 7

calib2, 10

curvemean, 12

Cy0, 14

eff, 15

efficiency, 16

evidence, 18

expcomp, 20

expfit, 21

fitprob, 22

guescini1, 24

guescini2, 25

LR, 26

maxRatio, 27

meanlist, 28

midpoint, 29

modlist, 30

mselect, 31

outlier, 32

pcrbatch, 34

pcrboot, 36

pcrfit, 38

pcrGOF, 40

pcropt1, 42

pcropt2, 44

pcrsim, 45

plot.pcrfit, 48

predict.pcrfit, 49

PRESS, 51

qpcR_functions, 58

replist, 65

reps, 66

reps2, 66

reps3, 67

resplot, 68

resVar, 69

RMSE, 69

Rsq, 70

Rsq.ad, 71

Rsq.cor, 72

RSS, 73

rutledge, 73

S27, 74

sliwin, 75

update.pcrfit, 76

*Topic **nonlinear**

AICc, 2

akaike.weights, 3

batchstat, 4

batschetal, 5

BIC, 6

calib, 7

calib2, 10

curvemean, 12

Cy0, 14

eff, 15

efficiency, 16

evidence, 18

expcomp, 20

expfit, 21

fitprob, 22

guescini1, 24

guescini2, 25

LR, 26

- maxRatio, 27
 - meanlist, 28
 - midpoint, 29
 - modlist, 30
 - mselect, 31
 - outlier, 32
 - pcrbatch, 34
 - pcrboot, 36
 - pcrfit, 38
 - pcrGOF, 40
 - pcropt1, 42
 - pcropt2, 44
 - pcrsim, 45
 - plot.pcrfit, 48
 - predict.pcrfit, 49
 - PRESS, 51
 - qpcR_functions, 58
 - ratiocalc, 60
 - replist, 65
 - reps, 66
 - reps2, 66
 - reps3, 67
 - resplot, 68
 - resVar, 69
 - RMSE, 69
 - Rsqr, 70
 - Rsqr.ad, 71
 - Rsqr.cor, 72
 - RSS, 73
 - rutledge, 73
 - S27, 74
 - sliwin, 75
 - update.pcrfit, 76
- AIC, 2, 4, 7, 19, 26
- AICc, 2, 19
- akaike.weights, 3, 32
- b3 (*qpcR_functions*), 58
 - b4 (*qpcR_functions*), 58
 - b5 (*qpcR_functions*), 58
 - baro5 (*qpcR_functions*), 58
 - barplot, 68
 - batchstat, 4
 - batsch1 (*batschetal*), 5
 - batsch2 (*batschetal*), 5
 - batsch3 (*batschetal*), 5
 - batsch4 (*batschetal*), 5
 - batsch5 (*batschetal*), 5
 - batschetal, 5
 - BIC, 6
 - calib, 7
 - calib2, 10
 - coef, 6, 22, 69, 71
 - coefficients, 2
 - confint, 48
 - curvemean, 12
 - Cy0, 14
 - eff, 15
 - efficiency, 16, 34, 36, 42, 43, 60
 - evidence, 18
 - expcomp, 20
 - expfit, 20, 21, 34
 - expGrowth (*qpcR_functions*), 58
 - fitprob, 22
 - fitted, 36, 70–72
 - guescini1, 24
 - guescini2, 25
 - l3 (*qpcR_functions*), 58
 - l4 (*qpcR_functions*), 58
 - l5 (*qpcR_functions*), 58
 - logLik, 2, 4, 6, 7, 26
 - LR, 26, 32
 - maxRatio, 27
 - meanlist, 28
 - midpoint, 29
 - modlist, 13, 28, 30, 34, 39, 65
 - mselect, 30, 31, 34
 - nls, 38, 39, 65
 - optim, 38, 65
 - outlier, 21, 32
 - pcrbatch, 30, 34, 39, 60
 - pcrboot, 36
 - pcrfit, 28, 30, 38, 46, 65, 76
 - pcrGOF, 40
 - pcrimport, 41
 - pcropt1, 42, 45
 - pcropt2, 44
 - pcrsim, 45
 - plot, 27, 46, 48

`plot.pcrfit`, 14, 48
`points`, 14
`predict`, 48, 51
`predict.pcrfit`, 49
PRESS, 51
`propagate`, 52, 60–62

`qpcR_functions`, 58

`ratiocalc`, 35, 56, 60
`read.table`, 41
`replist`, 28, 65
`reps`, 66
`reps2`, 66
`reps3`, 67
`residuals`, 2, 6, 22, 36, 68–73
`resplot`, 68
`resVar`, 69
RMSE, 69
Rsq, 70
`Rsq.ad`, 71
`Rsq.cor`, 72
RSS, 73
`rutledge`, 73

S27, 74
`sapply`, 30
`sliwin`, 34, 75
`stack`, 65

`update`, 51
`update.pcrfit`, 76

`w3` (`qpcR_functions`), 58
`w4` (`qpcR_functions`), 58