Package ‘PCRedux’

July 21, 2025

Type Package

Title Quantitative Polymerase Chain Reaction (qPCR) Data Mining and Machine Learning Toolkit as Described in Burdukiewicz (2022) <doi:10.21105/Joss.04407>

Version 1.2-0

Date 2025-06-11

Description Extracts features from amplification curve data of quantitative Polymerase Chain Reactions (qPCR) according to Pabinger et al. 2014 <doi:10.1016/j.bdq.2014.08.002> for machine learning purposes. Helper functions prepare the amplification curve data for processing as functional data (e.g., Hausdorff distance) or enable the plotting of amplification curve classes (negative, ambiguous, positive). The hookreg() and hookregNL() functions of Burdukiewicz et al. (2018) <doi:10.1016/j.bdq.2018.08.001> can be used to predict amplification curves with an hook effect-like curvature. The pcrfit_single() function can be used to extract features from an amplification curve.

License MIT + file LICENSE

LazyLoad yes

LazyData yes

URL `https://CRAN.R-project.org/package=PCRedux`

BugReports `https://github.com/PCRuniversum/PCRedux/issues`

Depends R (>= 3.5.0)

Imports changepoint, chipPCR, ecp, fda.usc, MBmca, pbapply, pracma, qpcR, robustbase, segmented, shiny, stats, utils, zoo

Suggests DT, future, knitr, listenv, RDML, readxl, rmarkdown, shinycssloaders, spelling, testthat, xtable

NeedsCompilation no

VignetteBuilder knitr

Encoding UTF-8

Language en-US

RoxygenNote 7.1.2

Repository CRAN

Date/Publication 2025-06-13 05:00:02 UTC

Author Stefan Roediger [aut] (ORCID: <https://orcid.org/0000-0002-1441-6512>),
Michal Burdukiewicz [aut] (ORCID:
<https://orcid.org/0000-0001-8926-582X>),
Andrej-Nikolai Spiess [cre, aut] (ORCID:
<https://orcid.org/0000-0002-9630-4724>),
Konstantin A. Blagodatskikh [aut] (ORCID:
<https://orcid.org/0000-0002-8732-0300>),
Dominik Rafacz [ctb] (ORCID: <https://orcid.org/0000-0003-0925-1909>)

Maintainer Andrej-Nikolai Spiess <draspiess@gmail.com>

Contents

`PCRedux-package` *PCRedux - quantitative PCR Data Mining and Machine Learning Toolkit*

Description

PCRedux package is a toolbox for the analysis of sigmoid curve (qPCR) data.

Usage

```
l4
```

Format

An object of class `list` of length 11.

Machine learning

In machine learning and statistics, the classification should be used to identify a new unknown observation. This observation is assigned to a number of categories. One basis is training data sets containing observations with known classes. Using the example of sigmoid amplification curves, this could be an assignment to the class "negative","ambiguous" or "positive". Basically, a number of descriptors (e. g., characteristics of curvature) are required to be able to assign classes. This package contains functions for extracting characteristics. In addition, the package contains data sets of classified amplification curves.

Author(s)

Stefan Roediger, Michal Burdukiewcz, Andrej-Nikolai Spiess, Konstantin A. Blagodatskikh

Examples

```
# Use the mblrr function to analyse amplification curves
library(qpcR)
mblrr(x=boggy[, 1], y=boggy[, 2])
```

`armor` *armor: fetch errors gently*

Description

`armor` is a helper function that catches errors and creates an output that can be used for further processing.

Usage

```
armor(f, n = 1)
```

Arguments

`f`	is the function that needs armor.
`n`	is the number of Zero repeats if a function fails.

Value

gives a `numeric` value (S3 class) as output for errors

Author(s)

Andrej Nikolai Spiess, Stefan Roediger

Examples

```
# Fetch the error from the diffQ function
# In the following the approximate derivative of the amplification curve data
# x <- RAS002[, 1] and y <- RAS002[, 2] is calculated by diffQ().
# This will not give an error.
x <- RAS002[, 1]
y <- RAS002[, 2]
armor_diffQ_passes <- armor(diffQ(cbind(x, y), verbose = TRUE)$xy)
armor_diffQ_passes
#
# In the following the approximate derivative of the sequences x <- 1:40
# and y <- 1:40 is calculated by diffQ(). However, this will fail.
# This will give the "internal" error
# >
# Error in list.res[[i]][[8]] : subscript out of bounds
# that is resolved to 0.
x <- 1:40
y <- 1:40
armor_diffQ_fails <- armor(diffQ(cbind(x, y), verbose = TRUE)$xy)
armor_diffQ_fails
```

`autocorrelation_test` *A function to test for autocorrelation of amplification curve data from a quantitative PCR experiment*

Description

`autocorrelation_test` is a function for an autocorrelation analysis from a quantitative PCR experiment. The result of the function is a correlation coefficient.

Usage

```
autocorrelation_test(y, n = 8, sig.level = 0.01)
```

Arguments

`y`	is the cycle dependent fluorescence amplitude (y-axis).
`n`	is the number of lagged cycles (default 12).
`sig.level`	is the significance level for the correlation test., Default: 0.01

Value

gives a `numeric` value (S3 class) as output for an autocorrelation

Author(s)

Stefan Roediger, Michal Burdukiewcz

Examples

```
library(qpcR)
default.par <- par(no.readonly = TRUE)
# Test for autocorrelation in amplification curve data
# Test for autocorrelation in the testdat data set
res_ac <- sapply(2:ncol(testdat), function(i) {
                    autocorrelation_test(testdat[, i])
                }
         )

# Plot curve data as overview
# Define the colors for the amplification curves
colors <- rainbow(ncol(testdat)-1, alpha=0.3)
# Names of samplesfile:///home/tux/R_malade
samples <- colnames(testdat)[-1]
layout(matrix(c(1,2,1,3), 2, 2, byrow = TRUE))
matplot(testdat[, 1], testdat[, -1], xlab="Cycle", ylab="RFU",
        main="testdat data set", type="l", lty=1, col=colors, lwd=2)
legend("topleft", samples, pch=19, col=colors, ncol=2, bty="n")

# Curves rated by a human after analysis of the overview. 1 = positive,
# 0 = negative
human_rating <- c(1,1,0,0,1,1,0,0,
                  1,1,0,0,1,1,0,0,
                  1,1,0,0,1,1,0,0)

# Convert the n.s. (not significant) to 0 and others to 1.
# Combine the results of the aromatic autocorrelation_test as variable "ac",
# the human rated values as variable "hr" in a new data frame (res_ac_hr).
res_ac_hr <- as.matrix(data.frame(ac=ifelse(res_ac=="n.s.", 0, 1),
                                  hr=human_rating))
res_performeR <- performeR(res_ac_hr[, "ac"], res_ac_hr[, "hr"])

# Add ratings by human and autocorrelation_test to the plot
par(las=2)
plot(1:nrow(res_ac_hr), res_ac_hr[, "hr"], xlab="Sample", ylab="Decisions",
     xaxt="n", yaxt="n", pch=19)
axis(2, at=c(0,1), labels=c("negative", "positive"), las=2)
axis(1, at=1:nrow(res_ac_hr), labels=colnames(testdat)[-1], las=2)
points(1:nrow(res_ac_hr), res_ac_hr[, "ac"], pch=1, cex=2, col="red")
legend("topleft", c("Human", "autocorrelation_test"), pch=c(19,1),
       bty="n", col=c("black","red"))

barplot(as.matrix(res_performeR[, c(1:10,12)]), yaxt="n",
        ylab="", main="Performance of autocorrelation_test")
axis(2, at=c(0,1), labels=c("0", "1"), las=2)
par(default.par)
```

---

decision_modus *A function to get a decision (modus) from a vector of classes*

Description

decision_modus is a function that can be used to find the most frequent (modus) decision. The classes can be defined by the user (e.g., a", "n", "y" -> "ambiguous", "negative", "positive"). This function is useful if large collections of varying decision (e.g., "a", "a", "a", "n", "n") need to be condensed to a single decision (3 x "a", 2 x "n" -> "a").

Usage

```
decision_modus(data, variables = c("a", "n", "y"), max_freq = TRUE)
```

Arguments

data	is a table containing the classes.
variables	is the class to look for.
max_freq	is a logical parameter (default == TRUE) delivers either the most occurring class or a summary.

Value

gives a factor (S3 class, type of integer) as output for a decision

Author(s)

Stefan Roediger, Michal Burdukiewcz

Examples

```
# First example
# Enter a string of arbitrary of "a","a","y","n"
# Result:
# [1] a
# Levels: a b n y

decision_modus(c("a","a","y","n","b"))

# Second example
# Analyze data from the decision_res_testdat.csv data file
filename <- system.file("decision_res_testdat.csv", package = "PCRedux")
my_data <- read.csv(filename)
head(my_data)

dec <- unlist(lapply(1L:nrow(my_data), function(i) {
       decision_modus(my_data[i, 2:4])
}))

names(dec) <- my_data[, 1]
dec
```

earlyreg	*A function to calculate the slope and intercept of an amplification curve data from a quantitative PCR experiment.*

Description

earlyreg is a function to calculate the slope and intercept of an amplification curve data from a quantitative PCR experiment. The number of cycles to be analyzed is defined by the user (default 6 cycles). The output contains the Maximal Information Coefficient (MIC), which can be interpreted as a correlation measure with a range of [0,1]. A value of 0 mean statistically independent data and 1 approaches in "probability for noiseless functional relationships" (see original study by Reshef, D. N. et al. Detecting novel associations in large data sets. Science, 334, 1518-1524 (2011)).

Usage

```
earlyreg(x, y, range = 5, normalize = FALSE)
```

Arguments

x	is the cycle numbers (x-axis).
y	is the cycle dependent fluorescence amplitude (y-axis).
range	is the number of cycles to be used for the regression.
normalize	is a logical parameter which indicates if the amplification curve data should be normalized to the 99 percent percentile of the amplification curve.

Value

gives a numeric vector (S3 class, type of double) as output for local regression

Author(s)

Stefan Roediger, Michal Burdukiewcz

Examples

```
# Calculate slope and intercept on noise (negative) amplification curve data
# for the cycles 2 to 7 for the C316.amp data set
library(chipPCR)
data(C316.amp)

# Plot the data
plot(C316.amp[, 2], y=C316.amp[, 3], xlab="Cycle", ylab="RFU",
     main="C316.amp data set", lty=1, type="l")
res <- earlyreg(x=C316.amp[, 2], y=C316.amp[, 3], range=5)
res
```

`encu` *A function to calculate numerous features from amplification curve data from a quantitative PCR experiment.*

Description

encu (ENcode CUrves) is a function to calculate numerous features of a large amplification curve data set. The `pcrfit_single` is performing the analysis for a single process.

Usage

```
encu(data, detection_chemistry = NA, device = NA)
```

Arguments

`data`	is the data set containing the cycles and fluorescence amplitudes.
`detection_chemistry`	contains additional meta information about the detection chemistry (e.g., probes, intercalating dye) that was used.
`device`	contains additional meta information about the qPCR system that was used.

Value

gives a `data.frame` vector (S3 class, type of `list`) as output for features

The output of the encu function is identical to the `pcrfit_single` function.

Author(s)

Stefan Roediger, Michal Burdukiewcz

Examples

```
library(qpcR)

# Calculate curve features of an amplification curve data. Note that not all
# available CPU cores are used. If need set "all" to use all available cores.
# In this example the testdat data set from the qpcR package is used.
# The samples F1.1 and F1.2 are positive amplification curves. The samples
# F1.3 and F1.4 are negative.
res_encu <- encu(testdat[, 1:3])
res_encu
```

head2tailratio — *A function to calculate to head to tail ratio of amplification curve data from a quantitative PCR experiment*

Description

`head2tailratio` is a function to calculate the ratio of the head and the tail of a quantitative PCR amplification curve. In this test, only the head (first six cycles) and the tail (last six cycles) form the region of interest (ROI).

Usage

```
head2tailratio(y, normalize = FALSE, slope_normalizer = FALSE, verbose = FALSE)
```

Arguments

`y`	is the cycle dependent fluorescence amplitude (y-axis).
`normalize`	is a logical parameter, which indicates if the amplification curve.
`slope_normalizer`	is a logical parameter, which indicates if the head2tailratio should be normalized to the slope of the ROI.
`verbose`	is a logical parameter, which indicates if all the values, parameters and coefficients of the analysis should be shown.

Value

gives a `numeric` (S3 class, type of `double`) as output for the head to tail ratio

Author(s)

Stefan Roediger, Michal Burdukiewcz

Examples

```
library(qpcR)

# calculate head to tail ratio on amplification curve data
res_head2tailratio <- sapply(2:ncol(competimer), function(i) {
   head2tailratio(y=competimer[, i], normalize=TRUE, slope_normalizer=TRUE)
})

res_head2tailratio_cluster <- kmeans(res_head2tailratio, 3)$cluster

matplot(competimer[, 1], competimer[, -1], xlab="Cycle", ylab="RFU",
      main="competimer data set", type="l", lty=1, col=res_head2tailratio_cluster, lwd=2)
```

hookreg — *A function to calculate the slope and intercept of an amplification curve data from a quantitative PCR experiment at the end of the data stream.*

Description

`hookreg` is a function to calculate the slope and intercept of an amplification curve data from a quantitative PCR experiment. The idea is that a strong negative slope at the end of an amplification curve is indicative for a hook effect (see Barratt and Mackay 2002).

Usage

```
hookreg(
  x,
  y,
  normalize = TRUE,
  sig.level = 0.0025,
  CI.level = 0.9975,
  robust = FALSE
)
```

Arguments

`x`	is the cycle numbers (x-axis).
`y`	is the cycle dependent fluorescence amplitude (y-axis).
`normalize`	is a logical parameter indicating if the data should be normalized to the 0.999 quantile
`sig.level`	defines the significance level to test for a significant regression
`CI.level`	confidence level required for the slope
`robust`	is a logical parameter indicating if the data should be analyzed be a robust linear regression (`lmrob`).

Value

gives a `numeric` (S3 class, type of `double`) as output for the detection of a hook

Author(s)

Stefan Roediger, Michal Burdukiewcz

References

K. Barratt, J.F. Mackay, *Improving Real-Time PCR Genotyping Assays by Asymmetric Amplification*, J. Clin. Microbiol. 40 (2002) 1571–1572. doi:10.1128/JCM.40.4.1571-1572.2002.

Examples

```
library(qpcR)

default.par <- par(no.readonly = TRUE)
# Calculate slope and intercept on noise (negative) amplification curve data
# for the last eight cycles.
res_hook <- data.frame(sample=colnames(boggy)[-1],
                       t(sapply(2:ncol(boggy), function(i) {
                       hookreg(x=boggy[, 1], y=boggy[, i])})))
res_hook

data_colors <- rainbow(ncol(boggy[, -1]), alpha=0.5)
cl <- kmeans(na.omit(res_hook[, 2:3]), 2)$cluster

par(mfrow=c(1,2))
matplot(x=boggy[, 1], y=boggy[, -1], xlab="Cycle", ylab="RFU",
 main="boggy Data Set", type="l", lty=1, lwd=2, col=data_colors)
 legend("topleft", as.character(res_hook$sample), pch=19,
         col=data_colors, bty="n")

plot(res_hook$intercept, res_hook$slope, pch=19, cex=2, col=data_colors,
 xlab="intercept", ylab="Slope",
 main="Clusters of Amplification Curves with an Hook Effect-like Curvature\nboggy Data Set")
 points(res_hook$intercept, res_hook$slope, col=cl, pch=cl, cex=cl)
 legend("topright", c("Strong Hook effect", " Weak Hook effect"), pch=c(1,2), col=c(1,2), bty="n")
 text(res_hook$intercept, res_hook$slope, res_hook$sample)

par(default.par)
```

---

hookregNL — *hookregNL - A function to calculate the slope of amplification curves in the tail region*

Description

hookregNL is a function to calculate the slope and intercept of an amplification curve from a quantitative PCR experiment. The idea is that a strong negative slope at the end of an amplification curve is indicative for a hook effect (see Barratt and Mackay 2002). In contrast to hookreg fits this function a sex-parameter model to the amplification curve and extracts the coefficient, which describes the slope.

Usage

```
hookregNL(x, y, plot = FALSE, level = 0.995, simple = TRUE, manualtrim = 5)
```

Arguments

x	is the cycle numbers (x-axis).
y	is the cycle dependent fluorescence amplitude (y-axis).

`plot`	is a logical parameter indicating if the data should be plotted, Default: FALSE.
`level`	the confidence level required, Default: 0.99.
`simple`	is a logical parameter. If TRUE (default) only the slope, confidence interval and decisions are shown as output
`manualtrim`	is the number of cycles that should be removed from the background. (`data.frame`). If FALSE, a `list` including the 6-parameter model is the output.

Value

gives a `numeric` (S3 class, type of `double`) as output for the detection of a hook

Author(s)

Andrej-Nikolai Spiess, Stefan Roediger, Michal Burdukiewcz

References

K. Barratt, J.F. Mackay, *Improving Real-Time PCR Genotyping Assays by Asymmetric Amplification*, J. Clin. Microbiol. 40 (2002) 1571–1572. doi:10.1128/JCM.40.4.1571-1572.2002.

Examples

```
library(qpcR)

# Analyze data from the boggy data set for potential hook effect like
# curvature
# has hook
res <- hookregNL(boggy[, 1], boggy[, 2])
res

# has no hook
res <- hookregNL(boggy[, 1], boggy[, 12])
res
```

`humanrater2` *Human Rater 2.0*

Description

Launches graphical user interface for the manual annotation of large amplification curve data sets, similarly to the `humanrater` function.

Usage

```
humanrater2()
```

Value

No return value, called for side effects

Warning

Any ad-blocking software may cause malfunctions.

`mblrr` *A function to perform a Local Robust Regression in Ranges defined by Qunantile-filtering*

Description

`mblrr` is a function to perform the Median based Local Robust Regression (mblrr) from a quantitative PCR experiment. In detail, this function attempts to break the amplification curve in two parts (head (~background) and tail (~plateau)). Subsequent, a robust linear regression analysis (`lmrob`) is preformed individually on both parts. The rational behind this analysis is that the slope and intercept of an amplification curve differ in the background and plateau region.

Usage

```
mblrr(x, y, sig.level = 0.01, normalize = FALSE)
```

Arguments

`x`	is the cycle numbers (x-axis).
`y`	is the cycle dependent fluorescence amplitude (y-axis).
`sig.level`	is the significance level for the correlation test.
`normalize`	is a logical parameter, which indicates if the amplification curve data should be normalized to the 99 percent quantile of the amplification curve.

Details

mblrr_intercept_bg is the intercept of the head region, *mblrr_slope_bg* is the slope of the head region, *mblrr_cor_bg* is the coefficient of correlation of the head region, *mblrr_intercept_pt* is the intercept of the tail region, *mblrr_intercept_pt* is the slope of the tail region, *mblrr_cor_pt* is the coefficient of correlation of the tail region

Value

gives a `numeric` (S3 class, type of `double`) as output for the regressed regions

Author(s)

Stefan Roediger, Michal Burdukiewcz

Examples

```
library(qpcR)

# Perform an mblrr analysis on noise (negative) amplification data of qPCR data
# with 35 cycles.
mblrr(x=boggy[, 1], y=boggy[, 2], normalize=TRUE)
```

`PCRedux_datasets` *The datasets implemented in PCRedux*

Description

A compilation of datasets for method evaluation/comparison.

Usage

```
data_sample
RAS002
RAS002_decisions
kbqPCR
decision_res_kbqPCR
```

Details

data_sample
Setup: Amplification curve data were analyzed with the encu() and the decision_modus() functions.
Details:
Data sets: batsch1, boggy, C126EG595, competimer, dil4reps94, guescini1, karlen1, lievens1, reps384, rutledge, testdat, vermeulen1, VIMCFX96_60, stepone_std.rdml, RAS002.rdml, RAS003.rdml, HCU32_aggR.csv, lc96_bACTXY.rdml.

RAS002
Setup: Amplification curve data of the RAS002.rdml data set.
Details:
Data sets: RAS002.rdml.

RAS002_decisions
Setup: Classes of the amplification curves from the RAS002.rdml data set.
Details:
Data sets: decision_res_RAS002.csv.

Author(s)

Stefan Roediger

References

Roediger, S., Burdukiewicz, M., Spiess, A.-N. & Blagodatskikh, K. Enabling reproducible real-time quantitative PCR research: the RDML package. *Bioinformatics* (2017). doi:10.1093/bioinformatics/btx528
Roediger, S., Burdukiewicz, M. & Schierack, P. chipPCR: an R package to pre-process raw data of amplification curves. *Bioinformatics* 31, 2900–2902 (2015)
Ritz, C. & Spiess, A.-N. qpcR: an R package for sigmoidal model selection in quantitative real-time polymerase chain reaction analysis. *Bioinformatics* 24, 1549–1551 (2008).

Examples

```
## 'data_sample' dataset.
head(data_sample)

## 'RAS002.rdml' dataset as rda file.
head(RAS002)
head(RAS002_decisions)
```

---

pcrfit_single — *pcrfit_single - A function to extract features from an amplification curve*

Description

The pcrfit_single is responsible for the extraction of features from amplification curve data. The function can be used for custom functions for a paralleled analysis of amplification curve data. An example is given in the vignette.

Usage

```
pcrfit_single(x)
```

Arguments

`x`	is the data set containing the fluorescence amplitudes.

Details

Details can be found in the vignette.

Value

Output Description

"cpD1"	maximum of the first derivative curve
"cpD2"	maximum of the second derivative curve
"cpD2_approx"	maximum of the second derivative curve calculated by the approximate derivative
"cpD2_ratio"	a value calculated from the ratio between cpD2 and cpD2_approx
"eff"	qPCR amplification efficiency

"sliwin"	qPCR amplification efficiency according the the 'window-of-linearity' method by Ruijter et al. (200
"cpDdiff"	absolute difference between cpD1 and cpD2
"loglin_slope"	slope determined by a linear model of the data points from the minimum and maximum of the secon
"cpD2_range"	cycle difference between the maximum and the minimum of the second derivative curve
"top"	takeoff point. When no top can be determined, the tob value is set to the first cycle number.
"f.top"	fluorescence at takeoff point. When no f.tdp can be determined, the f.tdp value is set to the RFU val
"tdp"	takes the maximum fluorescence subtracted by reverse values of the fluorescence and calculates the
"f.tdp"	fluorescence at tdp point. When no f.tdp can be determined, the f.tdp value is set to the RFU value a
"bg.stop"	estimates the end (cycle) the amplification curve background based on the bg.max function and nor
"amp.stop"	estimates the end (cycle) of the amplification curve based in the bg.max function and normalizes it t
"b_slope"	Is the slope of the seven parameter model
"b_model_param"	Is the b model parameter of the model optimally fitted according to the AIC
"c_model_param"	Is the c model parameter of the model optimally fitted according to the AIC
"d_model_param"	Is the d model parameter of the model optimally fitted according to the AIC
"e_model_param"	Is the e model parameter of the model optimally fitted according to the AIC
"f_model_param"	Is the f model parameter of the model optimally fitted according to the AIC
"f_intercept"	Is the intercept of the seven parameter model
"convInfo_iteratons"	Number of iterations needed to fit the 7 parameter model
"qPCRmodel"	non-linear model determined for the analysis
"qPCRmodelRF"	non-linear model determined for the analysis of the reversed amplification curve
"minRFU"	minimum of fluorescence amplitude
"maxRFU"	maximum of fluorescence amplitude
"init2"	initial template fluorescence from an exponential model
"fluo"	raw fluorescence value at the point defined by cpD2
"slope_bg"	slope of the first cycles
"k1_model_param"	Is the k1 model parameter of the seven parameter model
"k2_model_param"	Is the k2 model parameter of the seven parameter model
"intercept_bg"	intercept of the first cycles
"sigma_bg"	sigma of background
"sd_bg"	standard deviation of the background (ground phase) region (start to takeoff point)
"head2tail_ratio"	ratio between the signal of the background and tail region
"mblrr_intercept_bg"	the value of the intercept in the estimated background region of the amplification curve
"mblrr_slope_bg"	the value of the slope in the estimated background region of the amplification curve
"mblrr_cor_bg"	the value of the linear correlation coefficient in the estimated background region of the amplificatior
"mblrr_intercept_pt"	the value of the intercept in the estimated plateau phase of the amplification curve
"mblrr_slope_pt"	the value of the slope in the estimated plateau phase of the amplification curve
"mblrr_cor_pt"	the value of the linear correlation coefficient in the estimated plateau phase of the amplification curv
"polyarea"	area of a polygon given by the vertices in the vectors cycles and fluorescence
"peaks_ratio"	Takes the estimate approximate local minimums and maximums
"autocorrelation"	is a value of autocorrelation of a gain curve from a quantitative PCR experiment
"cp_e.agglo"	agglomerative hierarchical estimate for multiple change points
"amptester_shapiro"	tests based on the Shapiro-Wilk normality test if the amplification curve is just noise
"amptester_lrt"	performs a cycle dependent linear regression and determines if the coefficients of determination dev
"amptester_rgt"	Resids growth test (RGt) tests if fluorescence values in a linear phase are stable
"amptester_tht"	Threshold test (THt) takes the first 20 percent and the last 15 percent of any input data set and perfo
"amptester_slt"	Signal level test compares 1. the signals by a robust "sigma" rule by median + 2 * mad and 2. by co
"amptester_polygon"	pco test (pco) determines if the points in an amplification curve (like a polygon, in particular non-co
"amptester_slope.ratio"	SlR uses the inder function to find the approximated first derivative maximum, second derivative mi

"hookreg_hook"	estimate of hook effect like curvature
"hookreg_hook_slope"	estimate of slope of the hook effect like curvature
"hookreg_hook_delta"	Estimated value for the number of cycles from the qPCR cycle where the hook effect was determine
"central_angle"	shows the central angle calculated from the maximum and minimum of the second derivatives, with
"sd_bg"	shows the standard deviation of the fluorescence in the ground phase
"number_of_cycles"	Number of cylces
"direction"	test if the maximum of the first derivative is positive or negative
"range"	outputs the difference of fluorescence between 0.99 and 0.01 percentile. The value thus corresponds
"polyarea_trapz"	calculates trapezoidal integration. The calculation stops when the difference from one step to the ne
"cor"	is the value of the correlation coefficient from a linear correlation analysis according to Pearson bet
"res_coef_pcrfit.b"	is the parameter from the adjustment with a nonlinear (sigmoid) four-parametric model which descr
"res_coef_pcrfit.c"	is the parameter from the adjustment with a nonlinear (sigmoid) four-parametric model which descr
"res_coef_pcrfit.d"	is the parameter from the adjustment with a nonlinear (sigmoid) four-parametric model which descr
"res_coef_pcrfit.e"	is the parameter from the adjustment with a nonlinear (sigmoid) four-parametric model, which descr
"fitAIC"	is the value of the Akaike's second-order corrects Information Criterion, which was determined on a
"fitIter"	Number of iterations needed to fit the 4 parameter model
"segment_x"	Adjusts a regression model with segmented (linear) relationships between fluorescence and PCR cyc
"segment_U1.x"	Adjusts a regression model with segmented (linear) relationships between fluorescence and PCR cyc
"segment_U2.x"	Adjusts a regression model with segmented (linear) relationships between fluorescence and PCR cyc
"segment_psi1.x"	Adjusts a regression model with segmented (linear) relationships between fluorescence and PCR cyc
"segment_psi2.x"	Adjusts a regression model with segmented (linear) relationships between fluorescence and PCR cyc
"sumdiff"	describes proportion of cycles x in which the fluorescence signal of x is smaller than in x+1
"poly_1"	is a value of a third-order polynomial a + b*x + c*x^2 + d*x^3 is fitted to the curve data, where the
"poly_2"	is a value of a third-order polynomial a + b*x + c*x^2 + d*x^3 is fitted to the curve data, where the
"poly_3"	is a value of a third-order polynomial a + b*x + c*x^2 + d*x^3 is fitted to the curve data, where the
"poly_4"	is a value of a third-order polynomial a + b*x + c*x^2 + d*x^3 is fitted to the curve data, where the
"window_Win_1"	The complete curve trajectory is segmented into 10 equidistant windows by fitting an interpolating s
"window_Win_2"	The complete curve trajectory is segmented into 10 equidistant windows by fitting an interpolating s
"window_Win_3"	The complete curve trajectory is segmented into 10 equidistant windows by fitting an interpolating s
"window_Win_4"	The complete curve trajectory is segmented into 10 equidistant windows by fitting an interpolating s
"window_Win_5"	The complete curve trajectory is segmented into 10 equidistant windows by fitting an interpolating s
"window_Win_6"	The complete curve trajectory is segmented into 10 equidistant windows by fitting an interpolating s
"window_Win_7"	The complete curve trajectory is segmented into 10 equidistant windows by fitting an interpolating s
"window_Win_8"	The complete curve trajectory is segmented into 10 equidistant windows by fitting an interpolating s
"window_Win_9"	The complete curve trajectory is segmented into 10 equidistant windows by fitting an interpolating s
"window_Win_10"	The complete curve trajectory is segmented into 10 equidistant windows by fitting an interpolating s
"sd_plateau"	describes the standard deviation in the late phase of an amplification curve (last five cycles). With id

gives a `data.frame` (S3 class, type of `list`) as output for the curve features

Author(s)

Stefan Roediger, Michal Burdukiewcz

References

M. Febrero-Bande, M.O. de la Fuente, others, *Statistical computing in functional data analysis: The R package fda.usc*, Journal of Statistical Software. 51 (2012) 1–28. http://www.jstatsoft.org/v51/i04/

A.-N. Spiess, C. Deutschmann, M. Burdukiewicz, R. Himmelreich, K. Klat, P. Schierack, S. Roediger, Impact of Smoothing on Parameter Estimation in Quantitative DNA Amplification Experiments, Clinical Chemistry. 61 (2015) 379–388. doi:10.1373/clinchem.2014.230656.

S. Roediger, A. Boehm, I. Schimke, Surface Melting Curve Analysis with R, *The R Journal*. 5 (2013) 37–53. http://journal.r-project.org/archive/2013-2/roediger-bohm-schimke.pdf.

S. Roediger, M. Burdukiewicz, K.A. Blagodatskikh, P. Schierack, R as an Environment for the Reproducible Analysis of DNA Amplification Experiments, *The R Journal*. 7 (2015) 127–150. http://journal.r-project.org/archive/2015-1/RJ-2015-1.pdf.

S. Pabinger, S. Roediger, A. Kriegner, K. Vierlinger, A. Weinhauusel, A survey of tools for the analysis of quantitative PCR (qPCR) data, *Biomolecular Detection and Quantification*. 1 (2014) 23–33. doi:10.1016/j.bdq.2014.08.002.

S. Roediger, M. Burdukiewicz, P. Schierack, *chipPCR: an R package to pre-process raw data of amplification curves*, *Bioinformatics*. 31 (2015) 2900–2902. doi:10.1093/bioinformatics/btv205.

Examples

```
# Load the chipPCR package and analyze from the C126EG685 the first qPCR run
# "A01" (column 2).
library(chipPCR)
res <- pcrfit_single(C126EG685[, 2])
```

---

performeR *Performance analysis for binary classification*

Description

This function performs an analysis sensitivity and specificity to asses the performance of a binary classification test. For further reading the studies by Brenner and Gefeller 1997, James 2013 by Kuhn 2008 are a good starting point.

Usage

```
performeR(sample, reference)
```

Arguments

`sample`	is a vector with logical decisions (0, 1) of the test system.
`reference`	is a vector with logical decisions (0, 1) of the reference system.

Details

TP, true positive; FP, false positive; TN, true negative; FN, false negative

Sensitivity - TPR, true positive rate TPR = TP / (TP + FN)

Specificity - SPC, true negative rate SPC = TN / (TN + FP)

Precision - PPV, positive predictive value PPV = TP / (TP + FP)

Negative predictive value - NPV NPV = TN / (TN + FN)

Fall-out, FPR, false positive rate FPR = FP / (FP + TN) = 1 - SPC

False negative rate - FNR FNR = FN / (TN + FN) = 1 - TPR

False discovery rate - FDR FDR = FP / (TP + FP) = 1 - PPV

Accuracy - ACC ACC = (TP + TN) / (TP + FP + FN + TN)

F1 score F1 = 2TP / (2TP + FP + FN)

Likelihood ratio positive - LRp LRp = TPR/(1-SPC)

Matthews correlation coefficient (MCC) MCC = (TP*TN - FP*FN) / sqrt(TN + FP) * sqrt(TN+FN))

Cohen's kappa (binary classification) kappa=(p0-pc)/(1-p0)

r (reference) is the trusted label and s (sample) is the predicted value

	r=1	r=0
s=1	a	b
s=0	c	d

$$n = a + b + c + d$$

pc=((a+b)/n)((a+c)/n)+((c+d)/n)((b+d)/n)

po=(a+d)/n

Value

gives a `data.frame` (S3 class, type of `list`) as output for the performance

Author(s)

Stefan Roediger, Michal Burdukiewcz

References

H. Brenner, O. Gefeller, others, Variation of sensitivity, specificity, likelihood ratios and predictive values with disease prevalence, *Statistics in Medicine*. 16 (1997) 981–991.

M. Kuhn, Building Predictive Models in R Using the caret Package, *Journal of Statistical Software*. 28 (2008). doi:10.18637/jss.v028.i05.

G. James, D. Witten, T. Hastie, R. Tibshirani, An Introduction to Statistical Learning, *Springer New York, New York, NY*, (2013). doi:10.1007/978-1-4614-7138-7.

Examples

```
# Produce some arbitrary binary decisions data
# test_data is the new test or method that should be analyzed
# reference_data is the reference data set that should be analyzed
test_data <- c(0,0,0,0,0,0,1,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1)
reference_data <- c(0,0,0,0,1,1,1,1,0,1,0,1,0,1,0,1,0,1,0,1,1,1,1,1)

# Plot the data of the decisions
plot(1:length(test_data), test_data, xlab="Sample", ylab="Decisions",
     yaxt="n", pch=19)
axis(2, at=c(0,1), labels=c("negative", "positive"), las=2)
points(1:length(reference_data), reference_data, pch=1, cex=2, col="blue")
legend("topleft", c("Sample", "Reference"), pch=c(19,1),
        cex=c(1.5,1.5), bty="n", col=c("black","blue"))

# Do the statistical analysis with the performeR function
performeR(sample=test_data, reference=reference_data)
```

---

qPCR2fdata — *A helper function to convert amplification curve data to the fdata format.*

Description

qPCR2fdata is a helper function to convert qPCR data to the functional fdata class as proposed by Febrero-Bande & de la Fuente (2012). This function prepares the data for further analysis with the fda.usc package, which includes utilities for functional data analysis (e.g., Hausdorff distance).

Usage

```
qPCR2fdata(data, preprocess = FALSE)
```

Arguments

data	is a data set containing the amplification cycles (1. column) and the fluorescence (subsequent columns).
preprocess	is a logical parameter (default FALSE). If TRUE, the CPP function from the chipPCR package (Roediger et al. 2015) is used to pre-process the data (e.g., imputation of missing values). and the fluorescence (subsequent columns).

Value

gives an fdata object (S3 class, type of list) as output for a converted amplification curve.

Author(s)

Stefan Roediger, Michal Burdukiewcz

References

M. Febrero-Bande, M.O. de la Fuente, others, *Statistical computing in functional data analysis: The R package fda.usc*, Journal of Statistical Software. 51 (2012) 1–28. http://www.jstatsoft.org/v51/i04/

S. Roediger, M. Burdukiewicz, P. Schierack, *chipPCR: an R package to pre-process raw data of amplification curves*, Bioinformatics. 31 (2015) 2900–2902. doi:10.1093/bioinformatics/btv205.

Examples

```
library(qpcR)
library(fda.usc)

default.par <- par(no.readonly = TRUE)
# Calculate slope and intercept on noise (negative) amplification curve data
# for the last eight cycles.
# Convert the qPCR data set to the fdata format
res_fdata <- qPCR2fdata(testdat)

# Extract column names and create rainbow color to label the data
res_fdata_colnames <- colnames(testdat[-1])
data_colors <- rainbow(length(res_fdata_colnames), alpha=0.5)

# Plot the converted qPCR data
par(mfrow=c(1,2))
plot(res_fdata, xlab="cycles", ylab="RFU", main="testdat", type="l",
                   lty=1, lwd=2, col=data_colors)
legend("topleft", as.character(res_fdata_colnames), pch=19,
         col=data_colors, bty="n", ncol=2)

# Calculate the Hausdorff distance (fda.usc) package and plot the distances
# as clustered data.

res_fdata_hclust <- metric.hausdorff(res_fdata)
plot(hclust(as.dist(res_fdata_hclust)), main="Clusters of the amplification\n
   curves as calculated by the Hausdorff distance")
par(default.par)
```

`run_PCRedux` *PCRedux app*

Description

A graphical user interface for computing the properties of amplification curves. Take a look at the vignette to learn more about the different ways to start the app.

Usage

```
run_PCRedux()
```

Value

null.

No return value, called for side effects

Note

Any ad-blocking software may cause malfunctions.

---

tReem — *A function to Group Amplification Curves According to their Shape*

Description

`tReem` is a function to group amplification curves from a quantitative PCR experiment according to their shape. Either the Pearson correlation coefficient or the Hausdorff distance is used as measure. In most cases the grouping based on the Pearson correlation coefficient is sufficient. The grouping based on the Hausdorff distance can be very slow for large data sets.

Usage

```
tReem(data, cor = TRUE, k = 2)
```

Arguments

`data`	is the cycle dependent fluorescence amplitude (y-axis).
`cor`	is a logical parameter. If set true, the Pearson correlation is used as distance measure. If set FALSE the Hausdorff distance will be used.
`k`	an integer scalar or vector with the desired number of groups.

Value

gives a `data.frame` (S3 class, type of `list`) as output for the manual analyzed data

Author(s)

Stefan Roediger, Andrej-Nikolai Spiess

Examples

```
# Classify amplification curve data by Hausdorff distance
library(qpcR)
tReem(testdat[, 1:5])
```

winklR — *winklR: A function to calculate the angle based on the first and the second derivative of an amplification curve data from a quantitative PCR experiment.*

Description

`winklR` is a function to calculate the in the trajectory of the first and the second derivatives maxima and minima of an amplification curve data from a quantitative PCR experiment. For the determination of the angle (central angle), the origin is the maximum of the first derivative. On this basis, the vectors to the minimum and maximum of the second derivative are determined. This means that systematic off-sets, such as those caused by background, are taken into account. The output contains the angle.

Usage

```
winklR(x, y, normalize = FALSE, preprocess = TRUE)
```

Arguments

`x`	is the cycle numbers (x-axis). By default the first ten cycles are removed.
`y`	is the cycle dependent fluorescence amplitude (y-axis).
`normalize`	is a logical parameter, which indicates if the amplification curve data should be normalized to the 99 percent percentile of the amplification curve.
`preprocess`	is a logical parameter, which indicates if the amplification curve data should be smoothed (moving average filter, useful for noisy, jagged data).

Value

gives a `list` (S3 class, type of `list`) as output for the angles from an amplification curve.

Author(s)

Stefan Roediger

Examples

```
# Calculate the angles for amplification curve data from the RAS002 data set
data(RAS002)

# Plot the data
plot(RAS002[, 1],
  y = RAS002[, 2], xlab = "Cycle", ylab = "RFU",
  main = "RAS002 data set", lty = 1, type = "l"
)
res <- winklR(x = RAS002[, 1], y = RAS002[, 2])
res
```

```
plot(rbind(res$origin, res$p1, res$p2), col = c("black", "green", "blue"))

plot(RAS002[, 1],
  y = RAS002[, 7], xlab = "Cycle", ylab = "RFU",
  main = "RAS002 data set", lty = 1, type = "l"
)
res <- winklR(x = RAS002[, 1], y = RAS002[, 7])
res
plot(rbind(res$origin, res$p1, res$p2), col = c("black", "green", "blue"))

res_angles <- unlist(lapply(2:21, function(i) {
  winklR(RAS002[, 1], RAS002[, i])$angle
}))
cdplot(RAS002_decisions[1L:20] ~ res_angles, xlab = "angle", ylab = "decision")
```

Index