# Package 'PCRedux'

November 20, 2017

**Type** Package

**Title** Quantitative Polymerase Chain Reaction (qPCR) Machine Learning Helper Tool

**Version** 0.2.5-1

**Date** 2017-11-20

**Description** Extracts features from amplification curve data of quantitative Polymerase Chain Reactions (qPCR) (Pabinger, Stephan, Stefan Roediger, Albert Kriegner, Klemens Vierlinger, and Andreas Weinhauusel (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 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

**URL** https://github.com/devSJR/PCRedux

**BugReports** https://github.com/devSJR/PCRedux/issues

**Depends** R (>= 3.3.3)

**Imports** bcp, changepoint, chipPCR, ecp, fda.usc, FFTrees, magrittr, MBmca, pbapply, plotly, pracma, qpcR, robustbase, stats, testthat, utils, visdat, zoo

**Suggests** data.table, drc, dplyr, knitr, RDML, readxl, rmarkdown, xtable

**NeedsCompilation** no

**VignetteBuilder** knitr

**RoxygenNote** 6.0.1

**Author** Stefan Roediger [cre, aut], Michal Burdukiewicz [aut], Andrej-Nikolai Spiess [aut], Konstantin A. Blagodatskikh [aut]

**Maintainer** Stefan Roediger <stefan.roediger@b-tu.de>

**Repository** CRAN

**Date/Publication** 2017-11-20 18:14:46 UTC

# R topics documented:



---

`PCRedux-package` *PCRedux - quantitative PCR machine learning helper tool*

---

## Description

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

## Machine learning

In machine learning and statistics, classification is the task to identify a new observation which is assigned to a set of categories. A foundation are training data set, which containing observations with known memberships of the categories. In the context of sigmoid amplification curves this could be an assignment into "negative", "ambiguous" or "positive" classes. Basically, a set of descriptors (features of the curvature) is need to perform an assignment to a class. The PCRedux package contains function for feature extraction and human rated amplification curve with classes.

## 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])
```

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 either a correlation coefficient in case the result is significant at a given significance level, or a "n.s." (non-significant) of no correlation could be determined. Noise (negative) amplification curves usually do not exhibit any autocorrelation and will therefore be "n.s.".

## Usage

```
autocorrelation_test(y, n = 3, sig.level = 0.01)
```

## Arguments

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

## Author(s)

Stefan Roediger, Michal Burdukiewcz

## See Also

as.zoo, lag, cor.test

## Examples

```
# Test for autocorrelation in amplification curve data
# Load the libraries magrittr for pipes and qpcR for the data
library(magrittr)
library(qpcR)
# 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 <- data.frame(ac=ifelse(res_ac=="n.s.", 0, 1),
                        hr=human_rating) %>% as.matrix
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)
```

---

`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" -> "ambiguos", "negaive", "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 ocurring class or a summary.

## Author(s)

Stefan Roediger, Michal Burdukiewcz

## Examples

```
# First example
# Enter a string of arbritary 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
library(data.table)
library(magrittr)
filename <- system.file("decision_res_testdat.csv", package = "PCRedux")
my_data <- as.data.frame(fread(filename))
head(my_data)

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

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).

## Usage

```
earlyreg(x, y, range = 6, 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. |

### Author(s)

Stefan Roediger, Michal Burdukiewcz

### See Also

lmrob coefficients

### Examples

```
# Calculate slope and intercept on noise (negative) amplification curve data
# for the cycles 2 to 10 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=6)
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

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

## Author(s)

Stefan Roediger, Michal Burdukiewcz

## Examples

```
# 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.

library(qpcR)
res_encu <- encu(testdat[, 1:4])
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. |

## Author(s)

Stefan Roediger, Michal Burdukiewcz

## Examples

```
# calculate head to tail ratio on amplification curve data

library(qpcR)

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 = 0.005, CI.level = 0.995,
  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). |

## 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

```
# Calculate slope and intercept on noise (negative) amplification curve data
# for the last eight cycles.

library(qpcR)
library(magrittr)

 res_hook <- sapply(2:ncol(boggy), function(i) {
     hookreg(x=boggy[, 1], y=boggy[, i])}) %>% t %>%
     data.frame(sample=colnames(boggy)[-1],.)
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)
```

---

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 = 0.99, 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 reomoved from the background. (data.frame). If FALSE, a list including the 6-parameter model is the output. |

## 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.

## See Also

pcrfit confint

## Examples

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

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

mblrr *A function to perform a Qunantile-filter based Local Robust Regression*

## 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_less* is the intercept of the head region, *mblrr_slope_less* is the slope of the head region, *mblrr_cor_less* is the coefficient of correlation of the head region, *mblrr_intercept_more* is the intercept of the tail region, *mblrr_intercept_more* is the slope of the tail region, *mblrr_cor_more* is the coefficient of correlation of the tail region

## Author(s)

Stefan Roediger, Michal Burdukiewcz

## Examples

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

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

| | |
|---|---|
| "eff" | qPCR amplification efficiency |
| "cpD1" | maximum of the first derivative curve |
| "cpD2" | maximum of the second derivative curve |
| "fluo" | raw fluorescence value at the point defined by cpD2 |
| "init1" | initial template fluorescence from the sigmoidal model |
| "init2" | initial template fluorescence from an exponential model |
| "top" | takeoff point |
| "f.top" | fluorescence at takeoff point |
| "resLRE" | PCR efficiency by the 'linear regression of efficiency' method |
| "ressliwin" | PCR efficiency by the 'window-of-linearity' method |
| "cpDdiff" | difference between cpD1 and cpD2 |
| "slope_background" | slope of the first cycles |
| "intercept_background" | intercept of the first cycles |
| "polyarea" | area of a polygon given by the vertices in the vectors cycles and fluorescence |
| "changepoint.e.agglo" | agglomerative hierarchical estimate for multiple change points |
| "changepoint.bcp" | change point by Bayesian analysis methods |
| "qPCRmodel" | non-linear model determined for the analysis |
| "amptester_shap.noisy" | tests based on the Shapiro-Wilk normality test if the amplification curve is just noise |
| "amptester_lrt.test" | performs a cycle dependent linear regression and determines if the coefficients of deter |
| "amptester_rgt.dec" | Resids growth test (RGt) tests if fluorescence values in a linear phase are stable |
| "amptester_tht.dec" | Threshold test (THt) takes the first 20 percent and the last 15 percent of any input data s |
| "amptester_slt.dec" | Signal level test compares 1. the signals by a robust "sigma" rule by median + 2 * mad |

| | |
|---|---|
| "amptester_polygon" | pco test (pco) determines if the points in an amplification curve (like a polygon, in parti |
| "amptester_slope.ratio" | SlR uses the inder function to find the approximated first derivative maximum, second d |
| "minRFU" | minimum of fluorescence amplitude (percentile 0.01) |
| "maxRFU" | maximum of fluorescence amplitude (percentile 0.99) |
| "bg.start_normalized" | takes the start (cycle) the amplification curve background based on the bg.max function |
| "bg.stop_normalized" | estimates the end (cycle) the amplification curve background based on the bg.max funct |
| "amp.stop_normalized" | estimates the end (cycle) of the amplification curve based in the bg.max function and no |
| "head_to_tail_ratio" | |
| "autocorellation" | |
| "mblrr_intercept_less" | |
| "mblrr_slope_less" | |
| "mblrr_cor_less" | |
| "mblrr_intercept_more" | |
| "mblrr_slope_more" | |
| "mblrr_cor_more" | |
| "hookreg_hook" | estimate of hook effect like curvature |
| "mcaPeaks_minima_maxima_ratio" | Takes the estimate approximate local minimums and maximums |
| "diffQ2_slope" | slope determined by a linear model of the data points from the minimum and maximum |
| "diffQ2_Cq_range" | cycle difference between the maximum and the minimum of the second derivative 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/

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.


## See Also

bcp bg.max,amptester,smoother e.agglo diffQ,mcaPeaks,diffQ2 head2tailratio,earlyreg,hookreg,hookregNL,mblr
polyarea pcrfit,takeoff,LRE,sliwin,efficiency diff quantile

## 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((TP+FP)*(TP+FN)*(TN+FP)*(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

## 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.

## See Also

sensitivity specificity negPredValue

## 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)
reference_data <- c(0,0,0,0,1,1,1,1,0,1,0,1,0,1,0,1,0,1,1,1,1,1)

# 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 preprocess the data (e.g., imputation of missing values). and the fluorescence (subsequent columns). |

## 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

```
# Calculate slope and intercept on noise (negative) amplification curve data
# for the last eight cycles.
# Load additional packages for data and pipes.
library(qpcR)
library(fda.usc)
library(magrittr)

# 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 <- testdat[-1] %>% colnames()
```

```
data_colors <- rainbow(length(res_fdata_colnames), alpha=0.5)

# Plot the converted qPCR data
par(mfrow=c(1,2))
res_fdata %>% plot(., 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")
```

---

`visdat_pcrfit` *Visualizing the content of data from an analysis with the pcrfit_single function*

---

## Description

gives a ‘ggplot‘ or interactive ‘ggplotly‘ object (default) of a ’pcrfit_single’ data frame. The function is based on the vis_dat function by Tierney (2017). The colored indicators show the class (e.g., missing values, factors).

## Usage

```
visdat_pcrfit(data, type = "all", interactive = TRUE)
```

## Arguments

| | |
|---|---|
| `data` | contains the result from an analysis with the pcrfit_single function |
| `type` | specifies of all or only a subset of results should be shown, Default: ’all’. Alternatives ’qpcR’ or ’amptester’ |
| `interactive` | is a logical parameter, which indicates if the plot should be interactive, Default: TRUE |

## Details

type ’all’ shows all results from the analysis by the pcrfit_single function.

## Author(s)

Stefan Roediger, Michal Burdukiewcz

## References

N. Tierney, visdat: Visualising Whole Data Frames, *The Journal of Open Source Software*. 2 (2017). doi:10.21105/joss.00355.

## See Also

ggplotly vis_dat

## Examples

```
# 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.
library(qpcR)
# take the samples F1.1 (positive) and F1.3 (negative) for this example.

test_data <- testdat[, c(1,2,4)]

# Plot the amplification curves

matplot(test_data[, 1], test_data[, -1], xlab="Cycle", ylab="RFU",
        main="testdat data set", type="l", lty=1, lwd=2, col=1:2)
legend("topleft", paste(colnames(test_data)[-1], c("pos", "neg")),
       pch=19, col=1:2)

# Analyze the amplification curves with the pcrfit_single function
res_1 <- cbind(runs="F1.1", pcrfit_single(test_data[, 2]))
res_2 <- cbind(runs="F1.3", pcrfit_single(test_data[, 3]))
res <- rbind(F1.1=res_1, F1.3=res_2)

# Show all results in an interactive plot
visdat_pcrfit(res)
```

# Index