PCRedux package - an overview

The PCRedux package authors

2020-06-29

Contents

The PDF version of this document is available online.

1 Aims of the Project

The focus of this study is the development of statistical and bioinformatical algorithms for the `PCRedux` software (version 1.0.6). This software can be used to automatically calculate putative predictors (*features*) from qPCR amplification curves. A predictor herein refers to a quantifiable *informative* property of an amplification curve, employable for data mining, machine learning applications and classification tasks.

1.1 Analysis of Sigmoid-Shaped Curves for Data Mining and Machine Learning Applications

The following sections describe `PCRedux` regarding the analysis, numerical description and predictor calculation from a sigmoid curve. A predictor herein refers to a quantifiable *informative* property of a sigmoid curve. The predictors (subsection 1.2), sometimes referred to as descriptors, can be used for applications such as machine learning and automatic classification (e. g., negative or positive amplification).

The availability of classified amplification curve data sets and technologies for the classification of amplification curves is of high importance to train and validate models. This is dealt with in subsection 1.4 and subsection 1.3, respectively.

1.1.1 Principles of Amplification Curve Data Analysis and Predictor Calculation

The shape of a positive amplification curve is in most cases sigmoidal. Many factors such as the sample quality, qPCR chemistry, and technical problems (e. g., sensor errors) contribute to various curve shapes (Ruijter et al. 2014). The curvature of the amplification curve can be used as a quality measure. For example, fragmentation, inhibitors and sample material handling errors during the extraction can be identified. The kinetic of fluorescence emission is proportional to the quantity of the synthesized DNA. Typical amplification curves have three phases.

1. **Ground phase**: This phase occurs during the first cycles of the PCR, where the fluorescence emission is in most cases flat. Here, noise but no product formation is detected by the sensor system and the PCR product signal is an insignificantly small component of the total signal. This is often referred to as base-line or background signal. Apparently, there is only a phase shift or no signal at all, primarily due to the limited sensitivity of the instrument. Even in a perfect PCR reaction (double amplification per cycle), qPCR instruments cannot detect the fluorescence signal from the amplification. Fragmentation, inhibitors and sample handling errors would result in a prolonged ground phase. Nevertheless, this may indicate some typical properties of the qPCR system or probe system. In many instruments, this phase is used to determine the base-line level for the calculation of the Cycle threshold (Ct). The Ct value is considered statistically relevant as an increase outside of the noise range (threshold) when coming from the amplicon. In some qPCR systems, a flat amplification signal is expected in this phase. Slight deviations from this trend are presumably due to changes (e. g., disintegration of probes) in the fluorophores. Background correction algorithms are often used here to ensure that flat amplification curves without slope are generated. However, this can result in errors and inevitably leads to a loss of information via the waveform of the raw data (Nolan, Hands, and Bustin 2006). The slope, level and variance of this phase can serve as predictors.
2. **Exponential phase**: This phase follows the ground phase and is also called *log-linear phase*. It is characterized by a strong increase of the emitted fluorescence as the DNA amount roughly doubles in each cycle under ideal conditions and when the amount of the synthesized fluorescent labeled PCR product is high enough to be detected by the sensor system. This phase is used for the calculation of the quantification point (Cq) and curve specific amplification efficiency. The most important measurement from qPCRs is the Cq, which signifies the PCR cycle for which the fluorescence exceeds a `threshold value`. However, there is an ongoing debate as to what a significant and robust threshold value is. An overview and performance comparison of Cq methods is given in Ruijter et al. (2013). There are several mathematical methods to calculate the Cq.
    - The 'classical' threshold value (cycle threshold, Ct) is the intersection between a manually defined straight horizontal line with the quasi-linear phase in the exponential amplification phase (**??**A & B). This simple to implement method requires that amplification curves are properly baselined prior to analysis. The Ct method makes the assumption that the amplification efficiency (~ slope in the log-linear phase) is equal across all compared amplification curves (Ruijter et al. 2013). Evidently, this is not always case as exemplified in Figure 1C. The Ct method is widely used presumably due to the familiarity of users with this approach (e. g., chemical analysis procedures). However, this method is statistically unreliable (Ruijter et al. 2013; Spiess et al. 2015, 2016).

Moreover, the Ct method gives no stable in predictions if different users are given the same data set to be analyzed. *Therefore, this method is not used within the* ***PCRedux*** *package.*

- Another Cq method uses the maximum of the second derivative (SDM) (Rödiger et al. 2015) (**??**C). In all cases, the Cq value can be used to calculate the concentration of target sequence in a sample (low Cq →high target concentration). In contrast, negative or ambiguous amplification curves loosely resemble noise. This noise may appear linear or exhibit a curvature similar to a specific amplification curve (**??**). This however, may result in faulty interpretation of the amplification curves. Fragmentation, inhibitors and sample handling errors would decrease the slope of the amplification curve (Spiess, Feig, and Ritz 2008; Ritz and Spiess 2008). The slope and its variation can be considered as predictors. Since the Cq depends on the initial template amount and amplification efficiency, there is no immediate use of the Cq as an predictor.

3. **Plateau phase**: This phase follows the exponential phase and is a consequence of the exhaustion of limited reagents (incl. primers, nucleotides, enzyme activity) in the reaction vessel, limiting the amplification reaction, so that the theoretical maximum amplification efficiency (doubling per cycle) no longer prevails. This turning point, and the progressive limitation of resources, finally leads to a plateau. In the plateau phase, there is in some cases a signal decrease called *hook effect* (**??** and (Barratt and Mackay 2002; Isaac 2009; Burdukiewicz et al. 2018)). The slope (*hook effect*), level and variation can be considered as predictors.

If the amplification curve has only a slight positive slope and no perceptible/measureable exponential phase, it can be assumed that the amplification reaction did not occur (Figure 1B). Causes may include poor specificity of the PCR primers (non-specific PCR products), degraded sample material, degraded probes or detector failures. If a lot of input DNA is present in a sample, the amplification curve starts to increase in early PCR cycles (1 - 12 cycles). Some PCR devices have a software that corrects this feature without rechecking, resulting in an amplification curve with a negative trend.

The discussed phases are considered as regions of interest (ROI). As an example, the *ground phase* is in the head area, while the *plateau phase* is in the tail area. The *exponential phase* is located between these two ROIs.

The amplification curve shape, the amplification efficiency and the Cq value are important measures to judge the outcome of a qPCR reaction.

1.2 Data Analysis Functions of the `PCRedux` Package

The `PCRedux` package contains functions for analyzing amplification curves. In the following, these are distinguished into helper functions (subsection 1.5) and analysis functions (subsubsection 1.2.1).

1.2.1 Amplification Curve Analysis Functions of the `PCRedux` package

On the basis of these observations, **concepts** for predictors (*features*) were developed and implemented in algorithms to describe amplification curves. The functions described in the following are aimed for experimental studies. It is important to note that the concepts for the predictors proposed herein emerged by a *critical reasoning* process and *domain knowledge* of the `PCRedux` package creator. The aim of the package is to propose a set of predictors, functions and data for an independent research.

1.2.2 `pcrfit_single()` and `encu()`- Functions to Calculate Predictors from an Amplification Curve

The following sections give a concise description of the algorithms used to calculate predictor vectors by the `pcrfit_single()` function. Based on considerations and experience, the algorithms of the `pcrfit_single()` function are restricted to ROIs (Figure 1) to calculate specific predictors.

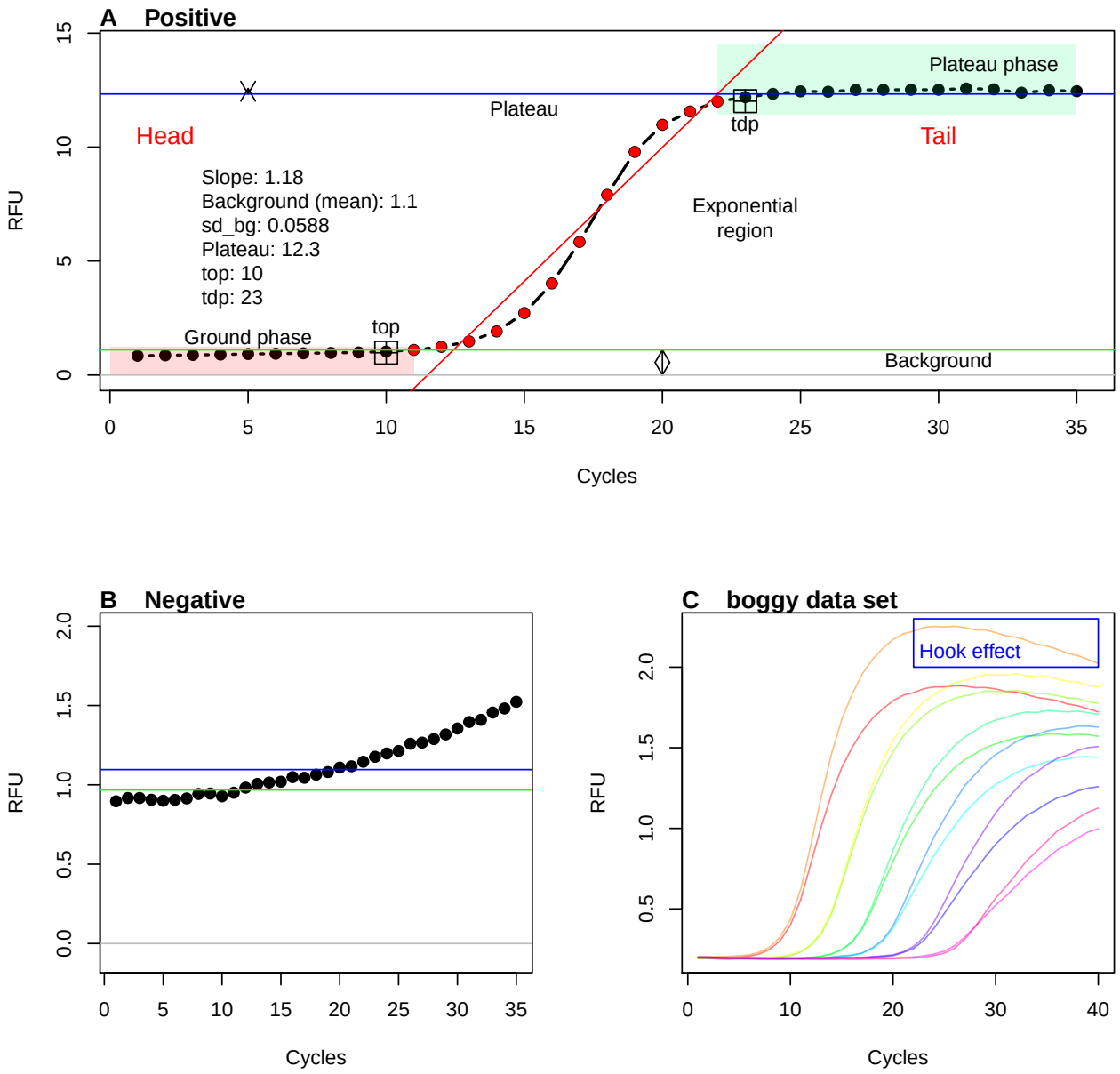

Figure 1: Phases of amplification curves as Region of Interest (ROI). For amplification curves, the fluorescence signal (RFU, relative fluorescence units) of the reporter dye is plotted against the cycle number. Positive amplification curves possess three ROIs: ground phase, exponential phase and plateau phase. These ROIs can be used to determine predictors such as the takedown point ('tdp') or the standard deviation within the ground phase ('sd_bg'). The exponential range (red dots) is used to determine the Cq values and amplification efficiency (not shown). A linear regression model (red) can be used to calculate the slope in this region. B) PCRs without amplification reaction usually show a flat (non-sigmoides) signal. C) The exponential phase of PCR reactions can vary greatly depending on the DNA starting quantity and other factors. Amplification curves that appear in later cycles often have a lower slope in the exponential phase.

The `encu()` function is a wrapper for the `pcrfit_single()` function. `encu()` can be used to process large records of amplification curve data arranged in columns. The progress of processing is displayed in the form of a progress bar and the estimated run-time. Additionally, `encu()` allows to specify which monitoring chemistry (e. g., DNA binding dye, sequence specific probes) and which thermo-cycler was used. Ruijter et al. (2014) demonstrated that the monitoring chemistry and the type of input DNA (single stranded, double stranded) are important when analysing qPCR data, because they have an influence on the shape of the amplification curve. For simplicity, the documentation will describe the `pcrfit_single()` only.

The algorithms are divided into the following broad categories:

- algorithms that determine slopes, signal levels,
- algorithms that determine turning points and
- algorithms that determine areas.

The algorithms in

- `earlyreg()` (subsubsection 1.2.9),
- `head2tailratio()` (subsubsection 1.2.10),
- `hookreg()`& `hookregNL()` (subsubsection 1.2.11) and
- `mblrr()` (subsubsection 1.2.12),
- `autocorrelation_test()` (subsubsection 1.2.13)

were implemented as standalone functions to make them available for other applications.

1.2.3 Amplification Curve Preprocessing

The `pcrfit_single()` function performs preprocessing steps before each calculation, including checking whether an amplification curve contains missing values. Missing values (NA) are measuring points in a data set where no measured values are available or have been removed arbitrarily. NAs may occur if no measurement has been carried out (e. g., defective detector) or lengths of the vectors differ (number of cycles) between the observations. Such missing values are automatically imputed by spline interpolation as described in Rödiger, Burdukiewicz, and Schierack (2015).

All values of an amplification curve are normalized to their 99% quantile. The normalization is used to equalize the amplitudes differences of amplification curves from thermo-cyclers (sensor technology, software processing) and detection chemistries. To compare amplification curves from different thermo-cyclers, the values should always be scaled systematically using the same method. Although there are other normalization methods (e. g., minimum-maximum normalization, see Rödiger, Burdukiewicz, and Schierack (2015), the normalization by the 99% quantile preserves the information about the level of the background phase. A normalization to the maximum is not used to avoid strong extenuation by outliers. The data in **??**D show that the `maxRFU` values after normalization are approximately 1. There is no statistical significant difference between `maxRFU` values of positive and negative amplification curves.

Selected algorithms of the `pcrfit_single()` function use the `CPP()` [`chipPCR`] function to preprocess (e. g., base-lining, smoothing, imputation of missing values) the amplification curves. Further details are given in Rödiger, Burdukiewicz, and Schierack (2015). Until package version 0.2.6-4 was the `visdat_pcrfit()` part of the package. `visdat_pcrfit()` was used for visualizing the content of data from an analysis with the `pcrfit_single()` function. There are other more powerful packages such as `visdat` by Tierney (2017), `assertr` by Fischetti (2019) and `xray` by Seibelt (2017).

During the analysis, several values are determined to describe the amplitude of an amplification curve. The resulting potential predictors are `minRFU` (minimum of the amplification curve, which is determined at the 1% quantile to minimize the influence of outliers), `init2` (the initial template fluorescence from an exponential model) and `fluo` (raw fluorescence value at the second derivative maximum). The `minRFU`, `init2` and `fluo` values differ significantly between negative and positive amplification curves (**??**C, E & F).

1.2.4 Handling of Missing Predictors

Missing values (NA) can occur if a calculation of a predictor is impossible (e. g., if a logistic function cannot be adapted to noisy raw data). The lack of a predictor is nevertheless an useful information (no predictor calculate $\mapsto$ amplification curves deviate from sigmoid shape). The NAs were left unchanged in the `PCRedux` package up to version 0.2.5-1. Since version 0.2.6 the NAs are replaced by numerical values (e. g., total number of cycles) or factors (e. g., *lNA* for non-fitted model). Under the term "imputation", there are a number of procedures based on statistical methods (e. g., neighboring median, spline interpolation) or on user-defined rules (Williams 2009; Cook and Swayne 2007; Hothorn and Everitt 2014). Rules are mainly used in the functions of `PCRedux` to relieve the user from the decision as to how to deal with missing values. For example, slope parameters of a model are set to zero when it cannot be determined. The disadvantage is that rules do not necessarily concur to real world values.

1.2.5 Multi-parametric Models for Amplification Curve Fitting

Both the `pcrfit_single()` function and the `encu()` function use four multi-parametric models based on the findings of Spiess, Feig, and Ritz (2008) and Ritz and Spiess (2008). The `pcrfit_single()` function starts by adjusting a seven-parameter model since this adapts *easier* and more frequent to a data set (Figure 2).

- **l7**:

$$f(x) = c + k1 \cdot x + k2 \cdot x^2 + \frac{d - c}{(1 + exp(b(log(x) - log(e))))^f} \tag{1}$$

From that model, the `pcrfit_single()` function estimates the variables `b_slope` and `c_intercept`, describing the slope and the y-intercept. The number of iterations required to adapt the model is also stored. That value is returned by the `pcrfit_single()` function as `convInfo_iteratons`. The higher the `convInfo_iteratons` value, the more iterations are necessary to converge from the start parameters (**??**L). A low `convInfo_iteratons` value is an indicator for

- a sigmoid curve shape or
- close start parameters.

High iterations numbers imply

- noisy amplification curves or
- non-sigmoid amplification curves.

The amplification curve fitting process continues with the four-parameter model (*l4*, Equation 2). This is followed by a model with five parameters (*l5*, Equation 3) and six parameters (*l6*, Equation 4).

- **l4**:

$$f(x) = c + \frac{d - c}{1 + exp(b(log(x) - log(e)))} \tag{2}$$

- **l5**:

$$f(x) = c + \frac{d - c}{(1 + exp(b(log(x) - log(e))))^f} \tag{3}$$

- **l6**:

$$f(x) = c + k \cdot x + \frac{d - c}{(1 + exp(b(log(x) - log(e))))^f} \tag{4}$$

The optimal model is selected on the basis of the Akaike information criterion and used for all further calculations. The `pcrfit_single()` function returns `qPCRmodel` as a factor (*l4*, *l5*, *l6*, *l7*). In case no model could be fitted, an *lNA* is returned.

The model is an indicator of the amplification curve shape. Model with many parameters deviate more from an ideal sigmoid model. For instance, a four-parameter model, unlike the six-parameter model, does not have a linear component. A negative linear slope in the plateau phase is an indicator of a *hook effect* (Burdukiewicz et al. 2018).

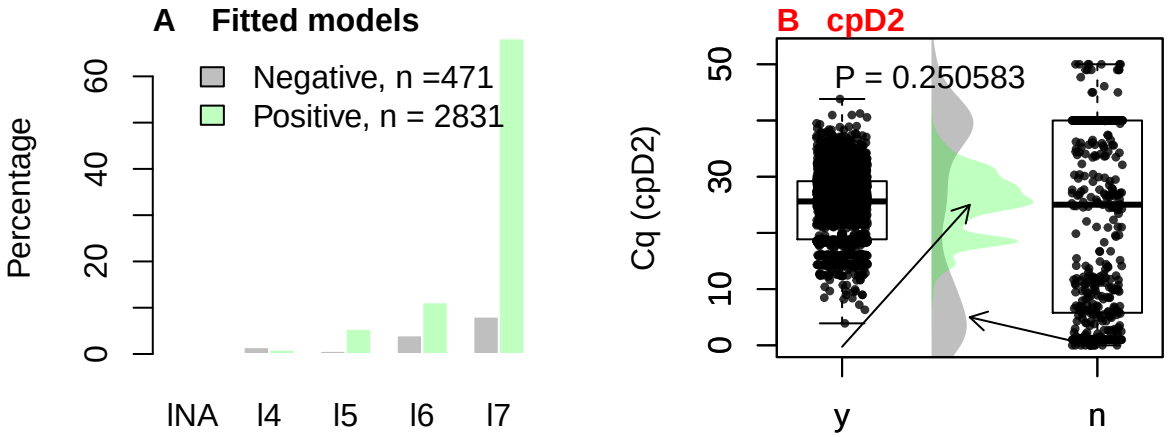

Figure 2: Frequencies of the fitted multiparametric models and Cq values. The amplification curves (n = 3302) of the 'data_sample' data set were analyzed with the encu() function. The amplification curves have been stratified according to their classes (negative: grey, positive: green). A) The optimal multiparametric model was selected for each amplification curve based on the Akaike information criterion. lNA stands for 'no model' and l4 . . . l7 for a model with four to seven parameters. B) All Cq values were calculated from optimal multiparametric models. Cqs of positive amplification curves accumulate in the range between 15 and 30 PCR cycles (50%). For the negative amplification curves, the Cqs are distributed over the entire span of the cycles. Note: The Cqs of the negative amplification curves are false-positive!

1.2.6 `winklR()` - A function to calculate the central angle based on the first and the second derivative of an amplification curve data

`winklR()` is a function to calculate the in the trajectory of the first negative and the second negative derivatives maxima and minima (**??**) of an amplification curve data from a quantitative PCR experiment. For the determination of the angle, the origin is the maximum of the first derivative. On this basis, the vectors to the approximate minimum and maximum of the second derivatives are determined. The vectors result from the relation of the maximum of the first derivative to the minimum of the second derivative and from the maximum of the first derivative to the maximum of the second derivative. In a simple trigonometric approach, the scalar product of the two vectors is formed first. Then the absolute values are calculated and multiplied by each other. Finally, the value is converted into an angle with the cosine. The assumption is that flat (negative amplification curves) have a large angle and sigmoid (positive amplification curves) have a smaller angle. Another assumption is that this angle is independent of the rotation of the amplification curve. This means that systematic off-sets, such as those caused by incorrect background correction, are of no consequence. The cycles to be analyzed is defined by the user. The output contains the angle and the coordinates of the minima and maxima.

1.2.7 Quantification Points, Ratios and Slopes

The `pcrfit_single()` function calculates `cpD1` and `cpD2` and uses them for further analysis. Both the `cpD1` and `cpD2` value are used to describe the amplification reaction quantitatively. For example, low `cpD1` and

cpD2 values (< 5 cycles) indicate that the PCR reaction was negative or that the amount of input DNA was to high Figure 2B. Since the pcrfit_single() function gives the parameters of all models (subsubsection 1.2.5) they are part of the feature set for completeness (**??**). In particular, the results of the five-parameter function and of the seven-parameter function are reported.

Further predictors from the pcrfit_single() function are:

- eff is the optimized PCR efficiency found within a sliding window (**??**C). A linear model of cycles versus log(Fluorescence) is fit within a sliding window (for details see sliwin() [qpcR] function). The comparison of positive and negative amplification curves in **??**A demonstrates that the classes are significantly different from each other. The eff values differ significantly between negative and positive amplification curves (**??**A).
- sliwin is the PCR efficiency by the 'window-of-linearity' method (Spiess, Feig, and Ritz 2008) (**??**B). The sliwin values differ significantly between negative and positive amplification curves (**??**B).
- cpDdiff is the difference between the first (cpD1) and the second derivative maximum cpD2 ($cpDdiff = cpD1 - cpD2$) **from the fitted model** (**??**C). Provided that a model can be exactly fitted, the estimates of the difference are reliable. Higher cpDdiff values indicate a negative amplification reaction or a very low amplification efficiency. The comparison of positive and negative amplification curves in **??**C demonstrates that the classes are significantly different from each other. In the event that the cpDdiff value cannot be determined (NA), it is replaced by zero. The cpDdiff values differ significantly between negative and positive amplification curves (**??**C).
- cpD2_range is the absolute value of the difference between the minimum and the maximum of the second derivative maximum ($cpD2\_range = |cpD2m - cpD2|$) from the diffQ2() function (**no model fitted**) (Figure 3E). The cpD2_range value does not require an adjustment of a multiparametric model. The approximate first and second derivatives are determined using a five-point stencil (Rödiger, Burdukiewicz, and Schierack 2015). The comparison of positive and negative amplification curves in **??**E shows that the classes differ significantly from each other. In the event that the cpD2_range value cannot be determined (NA), it is replaced by zero. The cpD2_range values differ significantly between negative and positive amplification curves (**??**E).
- cpD2_approx is the approximate second derivative maximum. In most cases the value should be close to the cpD2. Deviations indicate noise in the data, negative amplification curves or positive amplification curves that deviate from a typical sigmoid amplification curve.
- cpD2_ratio is is the ratio between the the approximate second derivative maximum cpD2_approx and the second derivative maximum cpD2 ($cpD2_ratio = cpD2/cpD2_approx$). In the event that the cpD2_ratio value cannot be determined (NA, Inf), it is replaced by zero. Provided that a model can be exactly fitted and that the function has little noise the ratio between both values should be close to 1. Note: Empirical data suggest that an interval of 0.85 and 1.1 indicates positive amplification curves. These can be set to 1. Values outside this interval indicate non-sigmoidal (e. g.„ negative) amplification curves. These can be set to 0.

- bg.stop is the end of the ground phase and amp.stop is the end of the exponential phase estimated by the bg.max() [chipPCR] function (Rödiger, Burdukiewicz, and Schierack 2015). A graphical presentation of the locations in the amplification curve are shown in Figure 3. The bg.stop and bg.stop values differ significantly between negative and positive amplification curves (**??**J & K).

- top is the *takeoff point* as proposed by Tichopad et al. (2003). The top is calculated using externally studentized residuals, which tested to be an outlier in terms of the t-distribution. The top signifies to first PCR cycle entering the exponential phase. tdp is the *takedown point*. This is an implementation in the pcrfit_single() function, which uses the rotated $f(x) \mapsto f_1(f(x))$ and flipped $g(x) = -(x)$ amplification curve for calculation. Figure 1A describes the location of top and tdp. The position (f.top, f.tdp) on the ordinate is also determined from these points. If an amplification curve is negative or neither top nor tdp can be calculated, then top & tdp will be assigned the number of cycles and f.top & f.tdp the value 1. The distribution of top, tdp, f.top and f.tdp is shown in **??**F-I. The top, tdp, f.top and f.tdp values differ significantly between negative and positive amplification curves. Potentially they enable a qualitative classification of the amplification reaction. An interesting aspect is that the positive f.top values are markedly lower than the negative f.top values The same applies

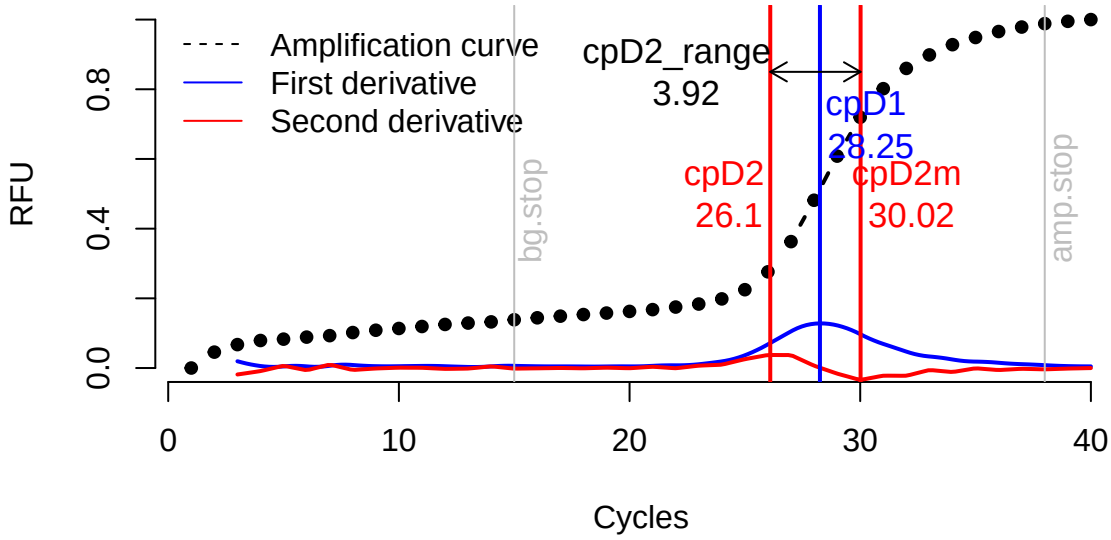

Figure 3: Location of the predictors 'cpD2_range', 'bg.start', 'bg.stop' within an amplification curve. The minimum (cpD2m) and maximum (cpD2) of the second derivative were calculated numerically using the diffQ2() function. This function also returns the maximum of the first derivative (cpD1). The 'cpD2_range' is defined as $cpD2\_range = |cpD2 - cpD2m|$. Large 'cpD2_range' values indicate a low amplification efficiency or a negative amplification reaction. The predictor 'bg.start' is an estimate for the end of the ground phase. 'bg.start' is an approximation for the onset of the plateau phase.

inversely to the `tdp` values. In this way, amplification curves can be classified according to these values.

- `peaks_ratio` is based on a sequential chaining of functions. The `diffQ()` [`MBmca`] function determines numerically the first derivative of an amplification curve. This derivative is passed to the `mcaPeaks()` [`MBmca`] function. In the output all minima and all maxima are contained. The ranges are calculated from the minima and maxima. The Lagged Difference is determined from the ranges of the minima and maxima. Finally, the ratio of the differences (maximum/minimum) is calculated. The `peaks_ratio` values differ significantly between negative and positive amplification curves (**??**B).
- `loglin_slope` is calculated from the slope determined by a linear model of the data points from the cycle dependent fluorescence at the minimum of the second derivative and maximum of the second derivative (Figure 4), provided that the locations of the minimum of the second derivative and the maximum of the second derivative yield a *suitable* interval. As a precaution, the algorithm checks, for example, whether the distance between the minimum of the second derivative and the maximum of the second derivative is not more than nine PCR cycles. Failing this, the `loglin_slope` value is set to zero (no slope), as in the example of Figure 4. The `loglin_slope` values differ significantly between negative and positive amplification curves (**??**D).

The predictor `loglin_slope` is used in the following to test if the slope within this ROI can be used to distinguish positive and negative amplification curves. The hypothesis is that positive amplification curves have a higher `loglin_slope` than negative amplification curves. As shown in **??**D, there is a statistically significant difference between positive and negative amplification curves.

- `sd_bg` is the standard deviation from the first PCR cycle to the takeoff point (Figure 1A). Manufacturers of thermo-cyclers use different sensors and data processing algorithms. The same applies to the detection chemistry used in experiments **??**. The signal variation in the ground phase differs between the different systems (**??**D). If no takeoff point can be determined from an amplification curve, the value for `sd_bg` is calculated from the first to the eighth PCR cycle. The results for the predictor `sd_bg` were broken down by the thermo-cycler and the output of the amplification reaction (negative, positive). It can be seen that the signal variation between the thermo-cyclers seems to be different. There is also a difference between negative and positive amplification curves Figure 5. The `sd_bg` values differ significantly

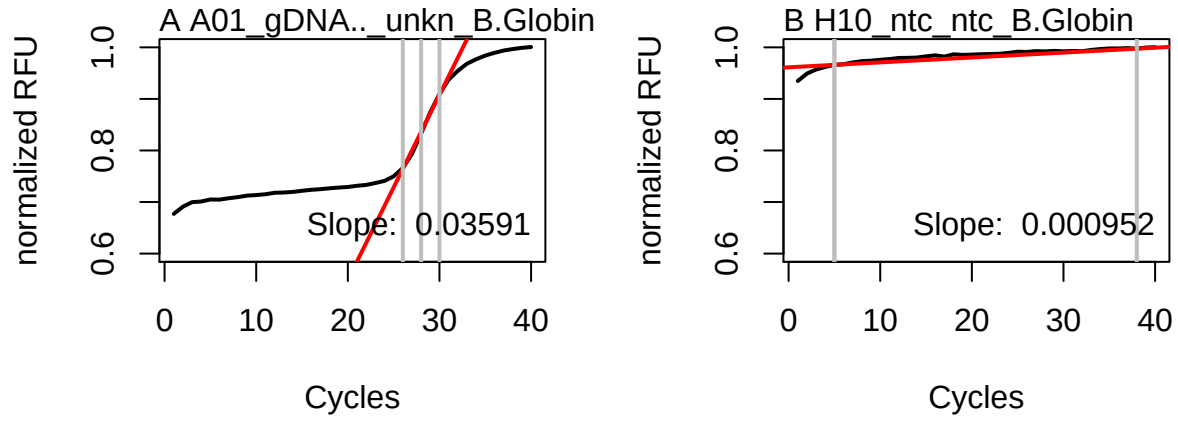

Figure 4: Concept of the 'loglin_slope' predictor. The algorithm determines the fluorescence values of the raw data at the approximate positions of the maximum off the first derivative, the minimum of the second derivative and the maximum of the second derivative, which are in the exponential phase of the amplification curve. The data were taken from the 'RAS002' data set. A linear model is created from these parameter sets and the slope is determined. A) Positive amplification curves have a clearly positive slope. B) Negative amplification curves usually have a low, sometimes negative slope.

between negative and positive amplification curves (??J).

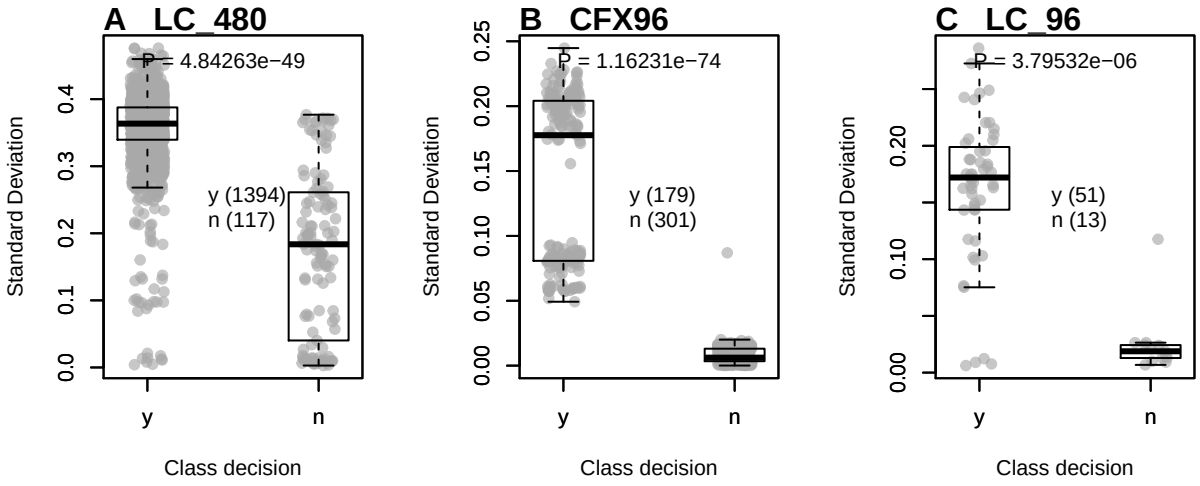

Figure 5: Standard deviation in the ground phase of various qPCR devices. The 'sd_bg' predictor was used to determine if the standard deviation between thermo-cyclers and between positive and negative amplification curves was different. The standard deviation was determined from the fluorescence values from the first cycle to the takeoff point. If the takeoff point could not be determined, the standard deviation from the first cycle to the eighth cycle was calculated. The Mann-Whitney test was used to compare the medians of the two populations (y, positive; n, negative). The differences were significant for A) LC_480 (Roche), B) CFX96 (Bio-Rad) and C) LC96 (Roche).

1.2.8 Integration of the `amptester()` function in `PCRedux`

`amptester_polygon` is another method to calculate the area under an amplification curve. `amptester_polygon`[1] is part of the `amptester()` [`chipPCR`] package (Rödiger, Burdukiewicz, and Schierack 2015). In contrast

[1]This predictor is determined from the points in an amplification curve (like a polygon, in particular non-convex polygons) in a 'clockwise' order. The sum over the edges result in a positive value if the amplification curve is 'clockwise' and is negative if the curve is 'counter-clockwise'.

to the implementation in `amptester()`, `amptester_polygon` has values normalized to the total number of cycles, thereby allowing comparable predictions (**??**Ds).

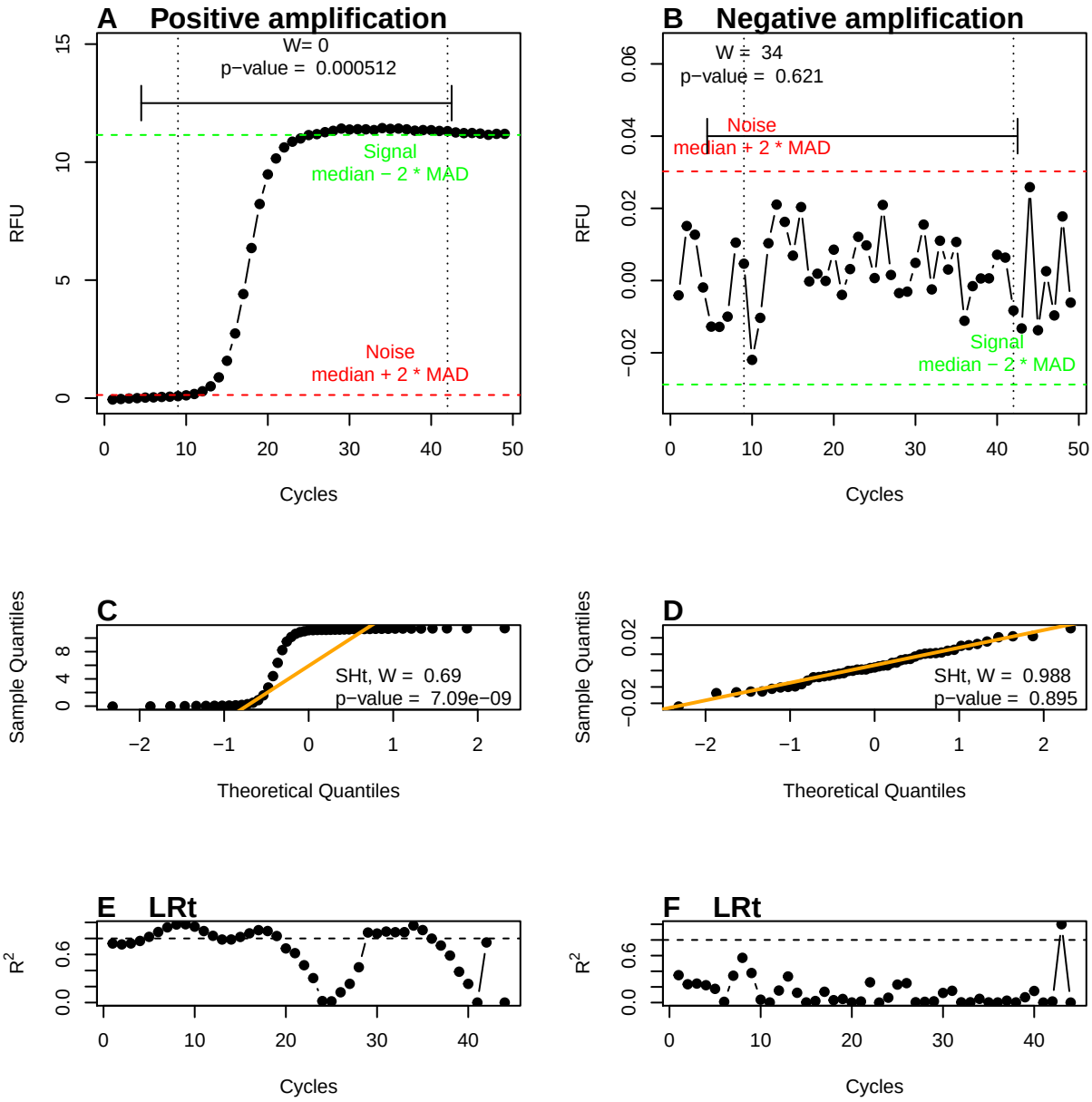

Figure 6: Analysis of amplification curves with the "amptester()" function. A & B) The threshold test (THt) is based on the Wilcoxon ranksum test and compares 20% of the fluorescence values of the ground phase with 15% of the plateau phase. In the example, a significant difference ($p = 0.000512$) was found for the positive amplification curve. However, this did not apply to the negative amplification curve ($p = 0.621$). C & D) A Q-Q diagram is used to graphically compare two probability distributions. In this study the probability distribution of the amplification curve was compared with a theoretical normal distribution. The orange line is the theoretically normal quantil-quantile plot that passes through the probabilities of the first and third quartiles. The Shapiro-Wilk test (SHt) of normality checks whether the underlying measurement data of the amplification curve is significantly normal distributed. Since the p-value of $7.09e^{-9}$ of the positive amplification curve is $\alpha \leq 5e^{-4}$, the null hypothesis is rejected. However, this does not apply to the negative amplification curve ($p = 0.895$). E & F) The linear regression test (LRt) calculates the coefficient of determination (R^2) using an ordinary least square regression where all measured values are integrated into the model in a cycle-dependent manner. Experience shows that the non-linear part of an amplification curve has a R^2 smaller than 0.8, which is also shown in the example.

1.2.9 `earlyreg()` - A Function to Calculate the Slope and Intercept in the Ground Phase of an Amplification Curve

The signal height and the slope in the first cycles (1 - 10) of amplification curves are potentially useful because some qPCR systems calibrate themselves by fluorescence intensity of the first cycles. This is noticeable as strong signal changes which appear spontaneously between the first and second cycle (e. g., **??**B). Furthermore, the signal level can be used to determine which background signal is present and whether the ground phase already has a slope. Moreover, characteristics of the detection probe system are noticeable (see subsubsection 1.1.1). From the slope, it may be deduced whether amplification has already started (see subsubsection 1.1.1).

Consequently, the `earlyreg()` function was developed. This function uses an ordinary least squares linear regression within a limited number of cycles. As ROI, the first 10 cycles were defined. This restriction is based on the developers experience, suggesting that during the first ten cycles only a significant increase in signal strength can be measured within few qPCRs. However, `earlyreg()` does not ignore the first cycle, as many thermo-cyclers use this cycle for sensor calibration. Extreme values are therefore included. As standard, the next nine amplitude values are used for the linear regression. The number of cycles can also be adjusted via the parameter `range`. Since all amplification curves are normalized to the 99%-percentile, comparability between the background signals and the slopes is ensured. The output of the `earlyreg()` function is:

- `slope_bg`, which is the slope of the ordinary least squares linear regression model,
- `intercept_bg`, which is the intercept of the linear model and
- `sigma_bg`, which is the square root of the estimated variance of the random error.

The `slope_bg`, `intercept_bg` and `sigma_bg` values differ significantly between negative and positive amplification curves (**??**G-I).

The following example illustrates possible usage of `earlyreg()`. For that purpose, amplification curves from the `C127EGHP` data set were analysed (Figure 7A). In figure Figure 7A the amplification curves for all cycles are shown. Next, the `earlyreg()` function was used to determine the `slope_bg`, `intercept_bg` and `sigma_bg` in the range of the first ten PCR cycles. The results were used in a cluster analysis using k-means clustering, demonstrating that the slope seems to be an indicator of differences between the amplification curves. The first 8 cycles were colored according to their cluster (Figure 7B). After cluster analysis, the same could also be observed (Figure 7D-F). Hence, it can be postulated that the slope in the background phase is useful for the amplification curve classification.

```
library(PCRedux)

# box_cox() function for the Box-Cox transformation of data

box_cox <- function(x, lambda = 1, offset = 0) {
    if (lambda == 0) {
        log(x + offset)
    } else
    {
        ((x + offset)^lambda - 1)/lambda
    }
}

# Load the C127EGHP data set
data <- chipPCR::C127EGHP[, -1]

# Normalize each amplification curve to their 0.99 percentile and use the
# earlyreg() function to determine the slope and intercept of the first
# cycles 'user_range'
```

```r
user_range <- 8

res_earlyreg <- do.call(rbind, lapply(2L:ncol(data), function(i) {
  earlyreg(x = data[, 1], y = data[, i], range = user_range, normalize = TRUE)
})) %>% box_cox(.)

# Label the observation with their names
rownames(res_earlyreg) <- substr(colnames(data)[-1], 1, 10)

# Show the first five lines of the res_earlyreg data matrix

head(res_earlyreg)
```

```
##      intercept      slope      sigma
## EG1 -0.9996982 -1.0000257 -0.9994170
## EG2 -0.9999187 -0.9999724 -0.9994349
## EG3 -1.0001748 -0.9999236 -0.9995357
## EG4 -1.0000410 -0.9999487 -0.9994961
## EG5 -1.0001467 -0.9999259 -0.9995627
## EG6 -1.0003437 -0.9998825 -0.9995230
```

```r
# Perform k-means clustering on the res_earlyreg data matrix
cl <- kmeans(res_earlyreg, centers = 2)

# Plot the results
# Use x_roi (cycles) and rfu_range (RFU values) to limit the
# range for the detailed plot of first cycles.

x_roi <- 1:(user_range + 1)
rfu_range <- range(data[x_roi, -1])

# Create graphic device for the plot(s)
layout(matrix(c(1, 2, 1, 2, 3, 3), 3, 2, byrow = TRUE))

# Plot of raw amplification curves

matplot(
    data[, 1], data[, -1], ylim = range(data[, -1]), pch = 19, lty = 1,
        type = "l", xlab = "Cycles", ylab = "RFU", main = "", col = "grey"
)
mtext("A", cex = 1, side = 3, adj = 0, font = 2)
abline(v = c(1,user_range))
text(3, range(data[, -1])[2], "ROI")

# Detailed plot of the first cycles and the clusters according
# to the k-means clustering
# Define some user colors (blue: EvaGreen, orange: Hydrolysis probes)
colors <- c(
    adjustcolor("blue", alpha.f = 0.5),
            adjustcolor("orange", alpha.f = 0.8)
)

matplot(NA, NA, xlim = range(x_roi), ylim = rfu_range, xlab = "Cycles",
        ylab = "RFU", main = "")
```

```
for(i in 1L:length(unique(cl$cluster))) {
cl_id <- which(cl[["cluster"]] == i) + 1
par(new=TRUE)
matplot(data[x_roi, 1], xlab = "", ylab = "", xaxt = "n", yaxt = "n",
        data[x_roi, cl_id], ylim = rfu_range, pch = 19, lty = 1, type = "l",
        col = colors[i])
}


mtext("B", cex = 1, side = 3, adj = 0, font = 2)
abline(v = c(1,user_range))
text(3, rfu_range[2], "ROI")
legend("bottomleft", c("Cluster 1", "Cluster 2"), pch = 15, cex = 1.2,
       col = colors, bty = "n")

# Overview of clusters and corresponding detection chemistry

eghp <- rep(0.7, length(cl$cluster))
names(eghp) <- names(cl$cluster)

barplot(eghp, las = 2, col = colors[cl[["cluster"]]],
        border = "white", xlab = "", ylab = "",
        yaxt = "n", ylim = c(0,2.2), cex.axis = 0.7)
mtext("C", cex = 1, side = 3, adj = 0, font = 2)
legend("topleft", c("Cluster 1", "Cluster 2"), pch = 15, col = colors,
       bty = "n", box.col =  "white", cex = 0.9)
arrows(0.5, 2, 38, 2, angle = 90, code = 3)
arrows(39, 2, 76.5, 2, angle = 90, code = 3)
text(c(18.75, 57.5), c(1.6, 1.6), c("EvaGreen", "Hydrolysis probes"))
```

1.2.10 `head2tailratio()` - A Function to Calculate the Ratio of the Head and the Tail of a Quantitative PCR Amplification Curve

The ratios from the ground and plateau phase can be used to search for patterns in amplification curves. Positive amplification curves have different slopes and intercepts at the start (head, background region) and the end (tail, plateau region) of the amplification curve. Hence, these regions are potentially useful to extract a predictor for amplification curve classification. Negative amplification curves (no slope) are assumed to have a ratio of about 1. In contrast, positive amplification curves should have a ratio of less than 1.

The n-dimensional space of all predictor variables $X_{1,2...n}$ is also called feature space. In the present study the feature space was extended by domain knowledge using known features. The `head2tailratio()`-function is an example for this. Here the feature $X_3 \equiv head2tail\_ratio$ could be calculated by determining the ratio of the $X_1 \equiv$ *fluorescence intensity in the head region* and $X_2 \equiv$ *fluorescence intensity in the tail region* of a quantitative PCR amplification curve ($X_3 = \frac{X_1}{X_2}$). As ROI, the areas in the ground phase (head) and plateau phases (tail) are used (Figure 1A). For the calculation, the median from the first six data points of the amplification curve and the median from the last six data points are used. The determination of six data points in both regions was made on the basis of *empirical experience*. As a rule, no significant increase in amplification signals can be measured in the first six cycles and in the last six cycles (where the amplification curve usually transitions into the plateau). This assumption is sometimes violated (e. g., *hook effect*) and might lead to false estimates.

```
library(PCRedux)
```

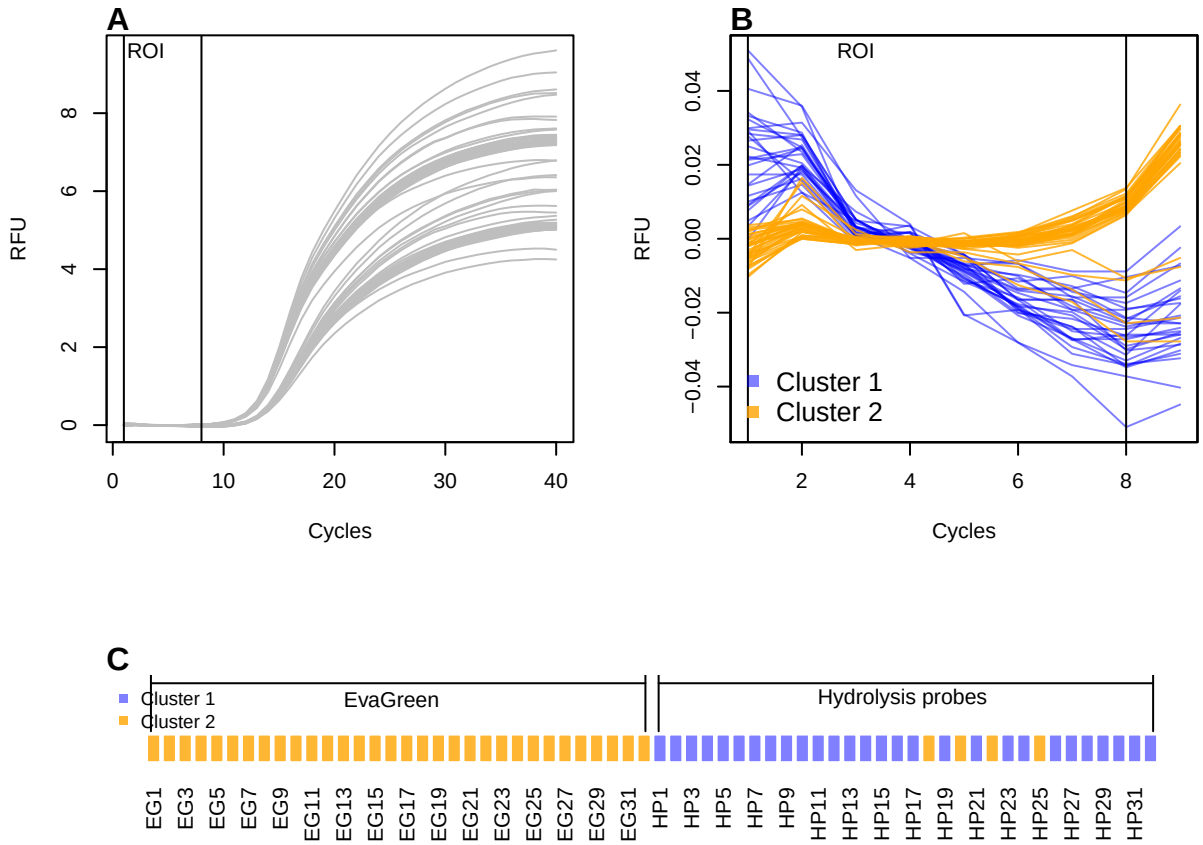

Figure 7: Analysis of the ground phase with the earlyreg() function and the 'C127EGHP' data set (n = 64 amplification curves). This data set consists of 32 samples, which were simultaneously monitored with the intercalator EvaGreen or hydrolysis probes. A) All amplification curves possess slightly different slopes and intercepts in the first cycles of the ground phase (ROI: Cycles 1 to 8). Both the slope and the intercept of each amplification curve were used for cluster analysis (k-means, Hartigan-Wong algorithm, number of centers $k = 2$). B) The amplification curves were assigned to five clusters, depending on their slope and their intersection (red, black). C) Finally, the clusters were associated to the detection chemistries (EvaGreen (EG) or hydrolysis probes (HP)).

```r
# Load the RAS002 amplification curve data set and assign it to the object data

data <- RAS002

# Load the RAS002 decision data set and assign it to the object data_decisions
data_decisions <- RAS002_decisions

# Calculate the head2tailratio of all amplification curves

res_head2tailratio <- lapply(2L:ncol(data), function(i) {
  head2tailratio(
    y = data[, i], normalize = TRUE, slope_normalizer = TRUE,
    verbose = TRUE
  )
})

# Fetch all values of the head2tailratio analysis for a later comparison
# by a boxplot.

res <- sapply(1L:length(res_head2tailratio), function(i)
  res_head2tailratio[[i]]$head_tail_ratio)

data_normalized <- cbind(
  data[, 1],
  sapply(2L:ncol(data), function(i) {
    data[, i] / quantile(data[, i], 0.99)
  })
)

# Assign colors to the classes (n: black, y: green).
colors <- as.character(factor(
  data_decisions, levels = c("y", "n"),
  labels = c(
      adjustcolor("green", alpha.f = 0.5), adjustcolor("black", alpha.f = 0.5))
))

res_wilcox.test <- stats::wilcox.test(res ~ data_decisions)
```

The amplification curves in Figure 8 show a signal increase within the first three cycles, and those in Figure 1C have a negative slope in the tail. The median is used to minimize the influence of outliers.

```r
# Plot the results of the analysis
#
# Position and plot parameters
h <- max(na.omit(res))
h_text <- rep(h * 0.976, 2)

# Create graphic device for the plot(s)
layout(matrix(c(1,1,2), 1, 3, byrow = TRUE))

matplot(
  data_normalized[, 1], data_normalized[, -1],
  xlab = "Cycles", ylab = "normalized RFU", main = "",
  type = "l", lty = 1, lwd = 2, col = colors
```

```
)
for (i in 1L:(ncol(data_normalized) - 1)) {
  points(
    res_head2tailratio[[i]]$x_roi, res_head2tailratio[[i]]$y_roi,
    col = colors[i], pch = 19, cex = 1.5
  )
  abline(res_head2tailratio[[i]]$fit, col = colors[i], lwd = 2)
}
mtext("A", cex = 1, side = 3, adj = 0, font = 2)

# Boxplot of the head2tail ratios of the positive and negative
# amplification curves.

boxplot(res ~ data_decisions, col = unique(colors), ylab = "Head to Tail Ratio")

lines(c(1, 2), rep(h * 0.945, 2))
text(1.5, h_text, paste0("P = ", signif(res_wilcox.test[["p.value"]])),
     cex = 1)

mtext("B", cex = 1, side = 3, adj = 0, font = 2)
```

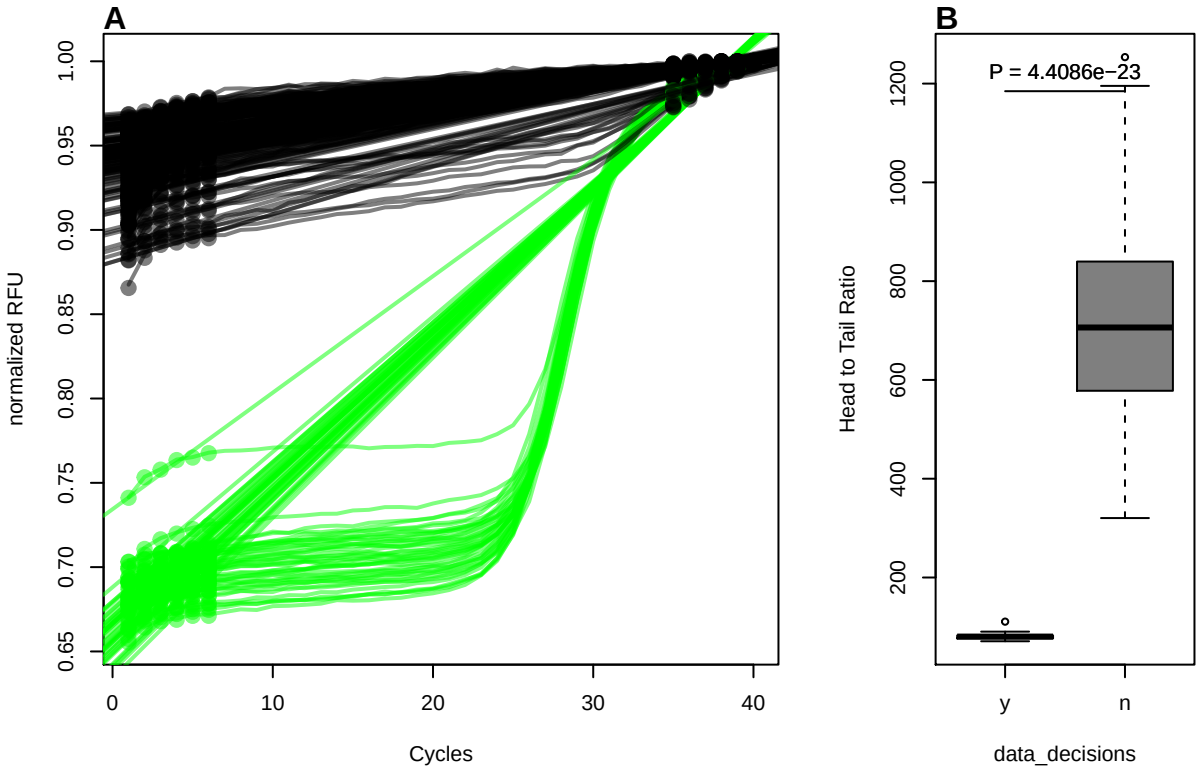

Figure 8: Ratio between the head and the tail of a quantitative PCR amplification curve. A) Plot of quantile normalized amplification curves from the 'RAS002' data set. Data points used in the head and and tail are highlighted by circles. The intervals for the Robust Linear Regression are automatically selected using the 25% and 75% quantiles. Therefore, not all data points are used in the regression model. The straight line is the regression line from the robust linear model. The slopes of the positive and negative amplification curves differ. B) Boxplot for the comparison of the *head/tail* ratio. Positive amplification curves have a lower ratio than negative curves. The difference between the classes is significant.

In subsubsection 1.1.1 and in **??**, it was shown that negative amplification curves may have a slope with a positive or negative sign. There is no consent in the literature and among peers how to deal with this during the processing. One solution is to include the slope as factor in the ratio calculation. The `head2tailratio()` function uses a linear model that calculates the slope between the ground and plateau phases. If the slope of the model is significant, then the ratio from the head and tail is normalized to this slope. This requires setting the `slope_normalizer` parameter in the `head2tailratio()` function. By default, this parameter is not set.

The `head2tail_ratio` values differ significantly between negative and positive amplification curves (**??**K).

1.2.11 `hookreg()` and `hookregNL()` - Functions to Detect Hook Effect-like Curvatures

`hookreg()` and `hookregNL()` are functions to detect amplification curves bearing a *hook effect* (Barratt and Mackay 2002) or negative slope at the end of the amplification curve. Both functions calculate the slope and intercept of an amplification curve data. The assumption is that a strong negative slope at the end of an amplification curve is indicative for a *hook effect*. `hookreg()` and `hookregNL()` are part of a peer-reviewed publication (Burdukiewicz et al. 2018). For this reason, the functions will not be discussed here.

1.2.12 `mblrr()` - A Function to Perform the Quantile-filter Based Local Robust Regression

`mblrr()` is a function to perform the **m**edian **b**ased **l**ocal **r**obust **r**egression (mblrr) from a quantitative PCR experiment. In detail, this function attempts to break the amplification curve in two ROIs (head (~background) and tail (~plateau)). As opposed to the `earlyreg()` function, the `mblrr()` function does not use a fixed interval. Instead, the `mblrr()` function dynamically determines cut points for each amplification curve. It was defined that:

- the 25% quantile is the value for which 25% of all values are smaller than this value.
- the 75% quantile is the value for which 75% of all values are greater than this value.

Subsequent, a robust linear regression analysis (`lmrob()`) is preformed individually on both regions of the amplification curve. The rationale behind this analysis is that the slope and intercept of an amplification curve differ in the background and plateau region. This is also shown by the simulations in **??**A-C. In the example shown below, the observations "P01.W19", "P06.W35", "P33.W66", "P65.W90", "P71.W23" and "P87.W01" were arbitrarily selected for demonstration purposes Figure 9. Another example is shown in Figure 15A. Those amplification curves have a slight negative trend in the base-line region and a positive trend in the plateau region.

The correlation coefficient[2] is a measure to quantify the dependence on variables (e. g., number of cycles, signal height). The correlation coefficient is always between -1 and 1, with a value close to -1 describing a strong-negative dependency and close to 1 describing a strong-positive dependency; if the value is 0, there is no dependency between the variables. The most frequently used correlation coefficient to describe a linear dependency is the Pearson correlation coefficient r. The correlation coefficient can be used as a predictor. Similar data structures have similar correlation coefficients. However, variables that are not strongly correlated can also be important for modeling.

```
library(PCRedux)

# Select four amplification curves from the RAS002 data set
amplification_curves <- c(2, 3, 4, 5, 44, 45)
data <- RAS002[, c(1, amplification_curves)]

# Load the decision_res_htPCR.csv data set from a csv file.
filename <- system.file("decision_res_RAS002.csv", package = "PCRedux")
decision_res <- read.csv(filename)
```

[2]Product moment correlation coefficient (Pearson)

```r
# Overview of the amplicon curve classifications
res_decision <- decision_res[amplification_curves - 1, -c(3,4)]

res_decision
```

```
##                       RAS002 test.result.1 conformity
## 1  A01_gDNA.._unkn_B.Globin             y       TRUE
## 2     A01_gDNA.._unkn_HPRT1             n       TRUE
## 3  A02_gDNA.._unkn_B.Globin             y       TRUE
## 4     A02_gDNA.._unkn_HPRT1             n       TRUE
## 43 B10_gDNA.._unkn_B.Globin             n       TRUE
## 44    B10_gDNA.._unkn_HPRT1             n       TRUE
```

```r
# Plot the regions and the linear regression line in the
# amplification curve plot

colors <- c(
            adjustcolor("blue", alpha.f = 0.5),
            adjustcolor("orange", alpha.f = 0.8)
           )

# Create graphic device for the plot(s)
par(mfrow = c(3, 2))

for (i in 2L:ncol(data)) {
  x <- data[, 1]
  y_tmp <- data[, i] / quantile(data[, i], 0.99)
  res_q25 <- y_tmp < quantile(y_tmp, 0.25)
  res_q75 <- y_tmp > quantile(y_tmp, 0.75)
  res_q25_lm <- try(
    suppressWarnings(lmrob(y_tmp[res_q25] ~ x[res_q25])),
    silent = TRUE
  )
  res_q75_lm <- try(
    suppressWarnings(lmrob(y_tmp[res_q75] ~ x[res_q75])),
    silent = TRUE
  )

  plot(x, y_tmp, xlab = "Cycles", ylab = "RFU (normalized)",
    main = "", type = "b", pch = 19)

  mtext(paste0(LETTERS[i - 1], "   ", colnames(data)[i]), cex = 1,
        side = 3, adj = 0, font = 2)
  legend("topleft", paste0(ifelse(res_decision[i - 1, 2] == "n",
                                  "negative", "positive")),
         bty = "n")
  abline(res_q25_lm, col = colors[1])
  points(x[res_q25], y_tmp[res_q25], cex = 2.5, col = colors[1])
  abline(res_q75_lm, col = colors[2])
  points(x[res_q75], y_tmp[res_q75], cex = 2.5, col = colors[2])
}
```

Finally, the results of the analysis were printed in a tabular format.

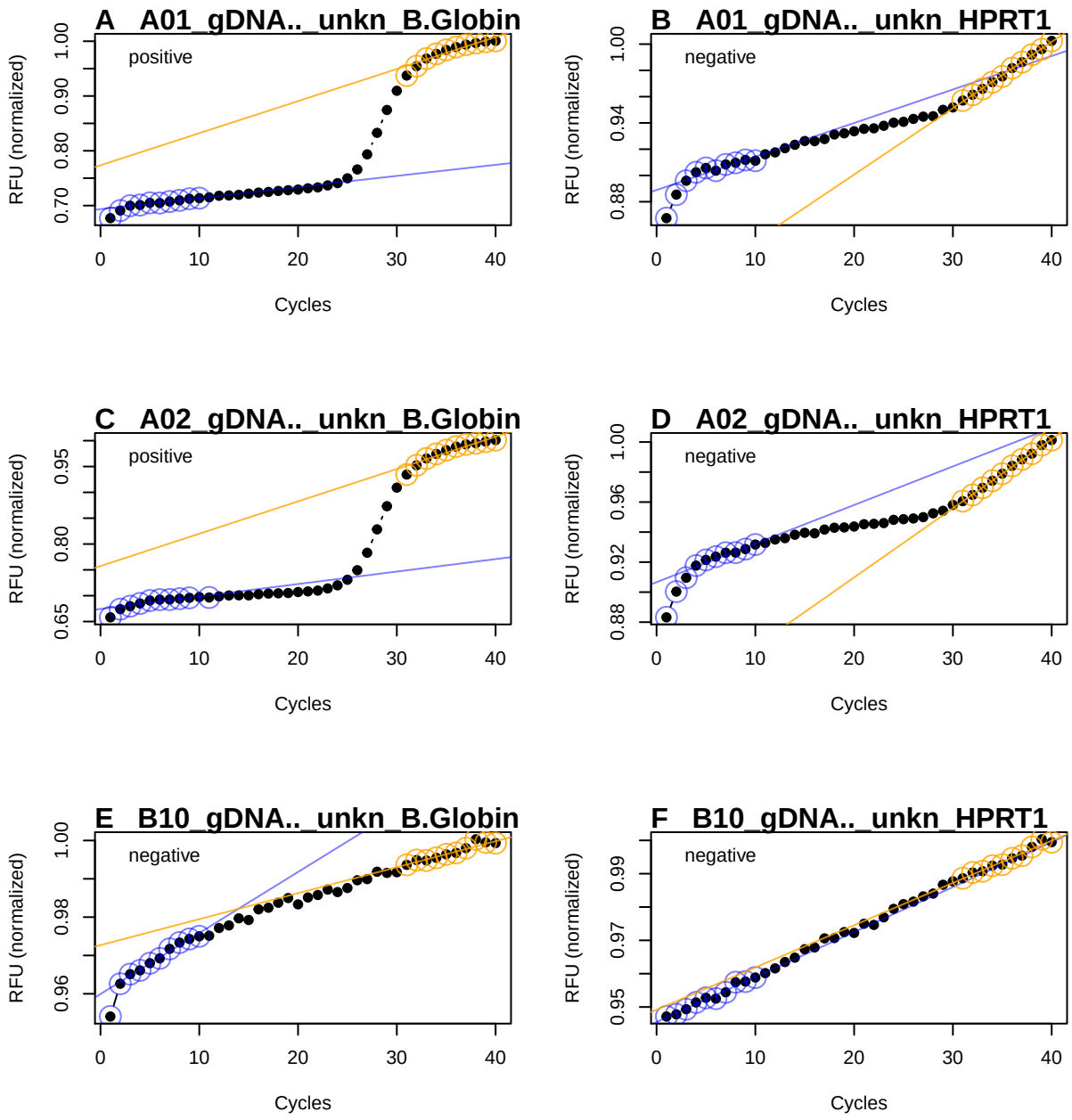

Figure 9: Robust local regression to analyze amplification curves. The amplification curves were arbitrarily selected from the 'RAS002' data set. In the qPCR setup, the target genes beta globin (B. globin) and HPRT1 were simultaneously measured in a PCR cavity using two specific hydrolysis probes (duplex qPCR). Both positive (A, C, E) and negative (B, D, F) amplification curves were used. The amplification curves are normalized to the 99% quantile. The differences in slopes and intercepts (blue and orange lines and dots). The mblrr() function is presumably useful for data sets which are accompanied by noise and artifacts.

```r
# Load the xtable library for an appealing table output
library(xtable)

# Analyze the data via the mblrr() function

res_mblrr <- do.call(cbind, lapply(2L:ncol(data), function(i) {
  suppressMessages(mblrr(
    x = data[, 1], y = data[, i],
    normalize = TRUE
  )) %>% data.frame()
}))
colnames(res_mblrr) <- colnames(data)[-1]

# Transform the data for a tabular output and assign the results to the object
# output_res_mblrr.

output_res_mblrr <- res_mblrr %>% t()

# The output variable names of the mblrr() function are rather long. For better
# readability the variable names were changed to "nBG" (intercept of the head
# region), "mBG" (slope of the head region), "rBG" (Pearson correlation of head
# region), "nTP" (intercept of the tail region), "mTP" (slope of tail region),
# "rBG" (Pearson correlation of the tail region)

colnames(output_res_mblrr) <- c(
  "nBG", "mBG", "rBG",
  "nTP", "mTP", "rTP"
)

print(xtable(
  output_res_mblrr, caption = "Selected results of predictors from the mblrr()
             function. nBG, intercept of head region; mBG, slope of head region;
             rBG, Pearson correlation of head region; nTP, intercept of tail
             region; mTP, slope of tail region; rBG, Pearson correlation of
             tail region",
  label = "tablemblrrintroduction"
), comment = FALSE, caption.placement = "top")
```

Table 1: Selected results of predictors from the mblrr() function. nBG, intercept of head region; mBG, slope of head region; rBG, Pearson correlation of head region; nTP, intercept of tail region; mTP, slope of tail region; rBG, Pearson correlation of tail region

	nBG	mBG	rBG	nTP	mTP	rTP
A01_gDNA.._unkn_B.Globin	0.69	0.00	0.91	0.77	0.01	0.94
A01_gDNA.._unkn_HPRT1	0.89	0.00	0.87	0.80	0.01	1.00
A02_gDNA.._unkn_B.Globin	0.67	0.00	0.88	0.76	0.01	0.95
A02_gDNA.._unkn_HPRT1	0.91	0.00	0.90	0.82	0.00	1.00
B10_gDNA.._unkn_B.Globin	0.96	0.00	0.95	0.97	0.00	0.96
B10_gDNA.._unkn_HPRT1	0.95	0.00	0.99	0.95	0.00	0.98

In another example, the results from the `mblrr()` function were combined with human classifications (positive, negative) to apply them in an analysis with Fast and Frugal Trees (FFTrees). FFTrees belong to class of simple decision rules. DT's are a classic approach to machine learning (Quinlan 1986). Here relatively simple algorithms and simple tree structures are used to create a model. A general introduction to decision trees

is given in (Quinlan 1986; Luan, Schooler, and Gigerenzer 2011). In many situations, FFTrees make fast decisions based on a few predictors (N = 1 - 5). In this example six predictors were used for the analysis.

The `FFTrees` package (Phillips et al. 2017) provides an implementation for the `R` statistical computing language. All that is needed for the present example are:

- the data assessed by the `mblrr()` function,
- the classification of the amplification curve data by a human,
- and a standard formula, which looks like $outcome \sim var1 + var2 + \ldots$ along with the data arguments. The function `FFTrees()` returns a fast and frugal tree object. This rich object contains the underlying trees and many classification statistics (similar to subsubsection 1.5.1). In the following example, the `RAS002` data set was used.

```
# Load the xtable library for an appealing table output
library(FFTrees)
library(PCRedux)

# The RAS002 amplification curves were analyzed with the mblrr() function
# to save computing time and the.results of this analysis are stored in the
# `data_sample` data set.

data <- data_sample[data_sample$dataset == "RAS002", c("mblrr_intercept_bg",
                                                       "mblrr_slope_bg",
                                                       "mblrr_cor_bg",
                                                       "mblrr_intercept_pt",
                                                       "mblrr_slope_pt",
                                                       "mblrr_cor_pt")]

# The output variable names of the mblrr() function are rather long. For better
# readability the variable names were changed to "nBG" (intercept of head
# region), "mBG" (slope of head region), "rBG" (Pearson correlation of head
# region), "nTP" (intercept of tail region), "mTP" (slope of tail region),
# "rBG" (Pearson correlation of tail region).


res_mblrr <- data.frame(
    class = as.numeric(as.character(factor(RAS002_decisions,
                                           levels = c("y", "n"),
                                           label = c(1, 0)))),
  data
)

colnames(res_mblrr) <- c("class", "nBG", "mBG", "rBG", "nTP", "mTP", "rTP")

res_mblrr.fft <- suppressMessages(
            FFTrees(formula = class ~., data = res_mblrr)
            )
```

Figure 10 shows the Fast and Frugal Trees by using the predictors nBG (intercept of head region), mBG (slope of head region), rBG (Pearson correlation of head region), nTP (intercept of tail region), mTP (slope of tail region), and rBG (Pearson correlation of tail region).

`R` offers several packages like `party` (Hothorn, Hornik, and Zeileis 2006), `rpart` (Therneau, Atkinson, and Ripley 2017) and `Rattle` (Williams 2009) for creating and visualizing decision trees.

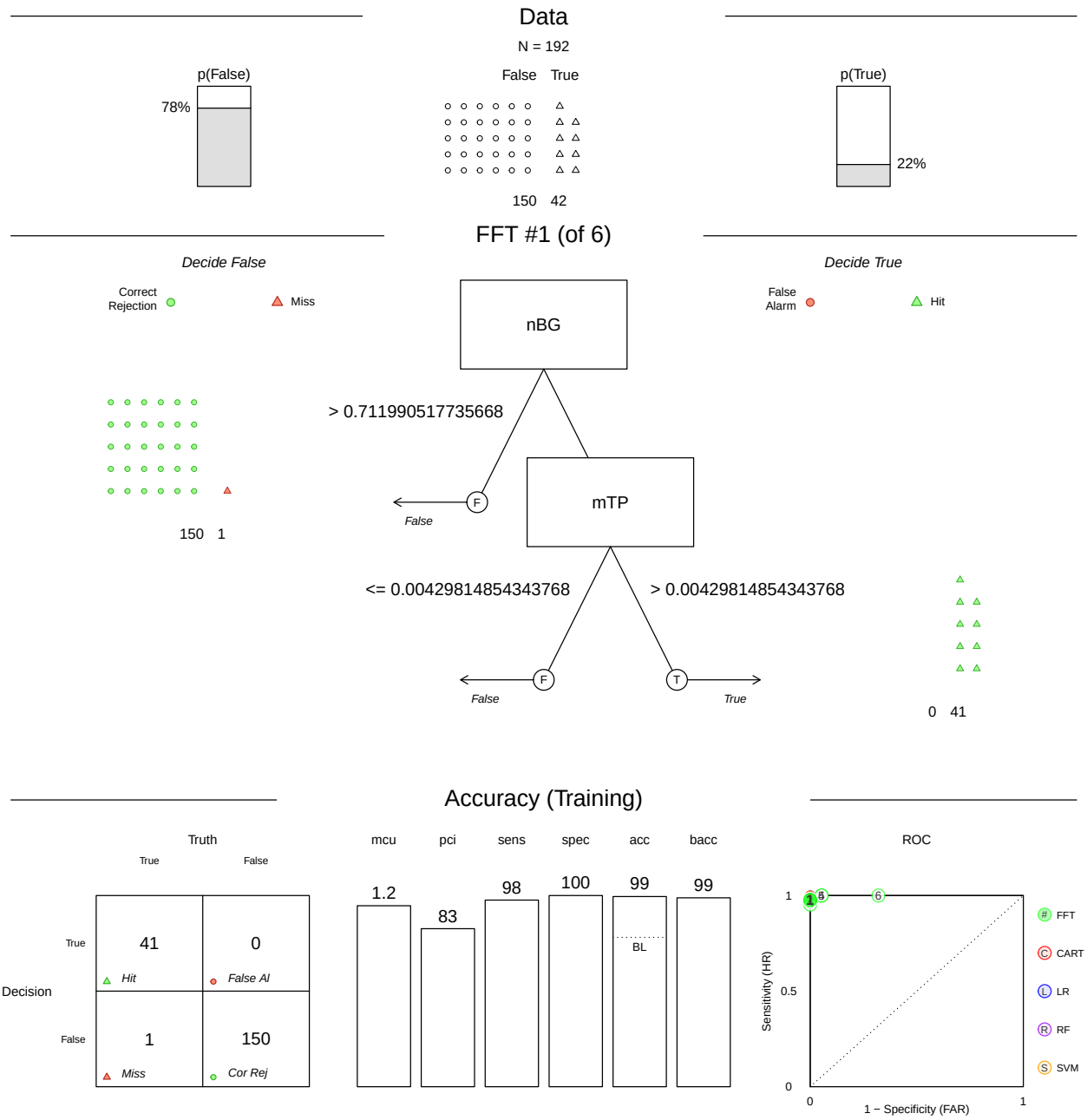

Figure 10: Visualization of decisions in Fast and Frugal Trees after data analysis of amplification curves via the mblrr() function. **Top row** 'Data') Overview of the data set, stating the total number of observations (N = 192) and percentage of positive (22%) and negative (78%) amplification curves. **Middle row** 'FFT #1 (of 6)') Decision Tree with the number of observations classified at each level of the tree. For the analysis, six predictors (nBG, intercept of head region; mBG, slope of head region; rBG, Pearson correlation of head region; nTP, intercept of tail region; mTP, slope of tail region; rBG, Pearson correlation of tail region) have been used for the analysis. After two tree levels (nBG, nTP), the decision tree is created, where all positive amplification curves (N = 40) are correctly classified. Two observations are classified as false-negative in the negative amplification curves. **Lower row** 'Performance') The FFTrees() [FFTrees] function determines several performance statistics. For the training data, there is a classification table on the left side showing the relationship between tree 'decision' and the 'truth'. The correct rejection ('Cor Rej') and 'Hit' are the right decisions. 'Miss' and false alarm ('False Al') are wrong decisions. The centre shows the cumulative tree performance in terms of mean of used cues ('mcu'), Percent of ignored cues ('pci'), sensitivity ('sens'), specificity ('spec'), accuracy ('acc') and weighted Accuracy ('wacc'). The receiver operating characteristic (ROC) curve on the right-hand side compares the performance of all trees in the FFTrees object. The system also displays the performance of the fast frugal trees ('#', green), CART ('C', red), logistical regression ('L', blue), random forest ('R', violet) and the support vector machine ('S', yellow).

1.2.13 `autocorrelation_test()` - A Function to Detect Positive Amplification Curves

Autocorrelation analysis is a technique that is used in the field of time series analysis. It can be used to reveal regularly occurring patterns in one-dimensional data (Spiess et al. 2016). Autocorrelation measures the correlation of a signal $f(t)$ with itself shifted by some time delay $f(t - \tau)$.

The `autocorrelation_test()` function coerces the amplification curve data to an object of class "zoo" (`zoo` package) as indexed totally ordered observations. Then follows the computation of a lagged version of the amplification curve data. The shifting of the amplification curve data is based on the number of observations (number of cycles 'c') with the following τ.

Number of Cycles (c)	τ
$c \leq 35$	8
$35 > c \leq 40$	10
$40 < c \leq 45$	12
$c > 45$	14

This is followed by a significance test for correlation between paired observations (original & lagged amplification curve data). The hypothesis is that paired observations of positive amplification curves exhibit significant correlation (`stats::cor.test`, significance level is 0.01) in contrast to negative amplification curves (noise). The application of the `autocorrelation_test()` function is shown in the following example.

In addition, the decisions file `decision_res_RAS002.csv` from the user was analyzed for the most frequent decision (modus) using the `decision_modus()` function (subsubsection 1.4.3).

1.2.14 Frequentist and Bayesian Change Point Analysis

Change point analysis (CPA) encompasses methods to identify or estimate single or multiple locations of distributional changes in a series of data points indexed in time order. A change herein refers to a statistical property. CPA is used for example in econometrics and bioinformatics (Killick and Eckley 2014; Erdman, Emerson, and others 2007). Several change point algorithms exist, such as the binary segmentation algorithm (Scott and Knott 1974). In change point analysis one assumes independent ordered observations $x_1, x_2, \ldots, x_n \in \mathbb{R}^d$ (N. A. James and Matteson 2013). In the case of qPCR, this is simply the cycle-dependent fluorescence, used to create k homogeneous subsets of unknown size (Erdman, Emerson, and others 2007). While frequentist methods make an estimation of the parameter at the location (e. g., mean) of the change points at specific points, change point analysis using Bayesian method produces a probability for the occurrence of a change point at certain points. For the analysis of the amplification curves, it was hypothesized that the number of change points differs between positive (sigmoidal) and negative (noise) amplification curves.

The `pcrfit_single()` function uses two independent approaches for change point analysis. These are the `bcp()` [`bcp`] (Erdman, Emerson, and others 2007) and the `e.agglo()` [`ecp`] function (N. A. James and Matteson 2013). The `e.agglo()` function performs a non-parametric change point analysis based on agglomerative hierarchical estimation and is useful to "detect changes within the marginal distributions" (N. A. James and Matteson 2013). Measurements from the qPCR systems typically show noise that has rapidly changing components. Differentiators amplify these rapidly changing noise components (Rödiger, Böhm, and Schimke 2013). Therefore, the first derivation of the amplification curve was used for both change point analyses. It was assumed for the change point analysis of amplification curves, that this leads to larger differences between positive and negative amplification curves. An example is shown on Figure 11. In contrast the `bcp()` [`bcp`] function performs a change point analysis based on a Bayesian approach. This method can detect changes in the mean of independent Gaussian observations. As a result, the analysis returns the posterior probability of a change point at each x_i. An example is shown on Figure 11. Both the change point analysis methods provide additional information to distinguish positive and negative amplification curves **??**E & F).

1.2.15 Frequentist Approaches to Test the Class of an Amplification Reaction and Application of the `amptester()` Predictors

A part of `pcrfit_single()` is the `amptester()` [`chipPCR`] function, which contains tests to determine whether an amplification curve is positive or negative. The input values for the function differ due to the different preprocessing steps in the `pcrfit_single()` function. Therefore, the concepts of the tests are briefly described below.

- The first test, designated as SHt, is based on this Shapiro-Wilk test of normality. This relatively simple procedure can be used to check whether the underlying population of a sample (amplification curve) is significantly ($\alpha \leq 5e-04$) normal distributed. The name of the output of the `pcrfit_single()` function is `amptester_shapiro`.
- The second test is the *Resids growth test* (RGt), which tests if the fluorescence values in linear phase are stable. Whenever no amplification occurs, fluorescence values quickly deviate from linear model. Their standardized residuals will be strongly correlated with their value. For real amplification curves, the situation is much more stable. Noise (meaning deviations from linear model) in background do not correlate strongly with the changes in fluorescence. The decision is based on the threshold value (here 0.5). The output is binary coded (negative = 0, positive = 1). The output name of the `pcrfit_single()` function is `amptester_rgt`.
- The third test is the *Linear Regression test* (LRt). This test determines the coefficient of determination (R^2) by an ordinary least squares linear (OLS) regression. The R^2 are determined from a run of $\sim 15\%$ range of the data. If a sequence of more than six R^2s larger than 0.8 is found, a nonlinear signal is plausible. This is somewhat counter-intuitive, because R^2 of nonlinear data should be low. The output is binary coded (negative = 0, positive = 1). The output name of the `pcrfit_single()` function is `amptester_lrt`.

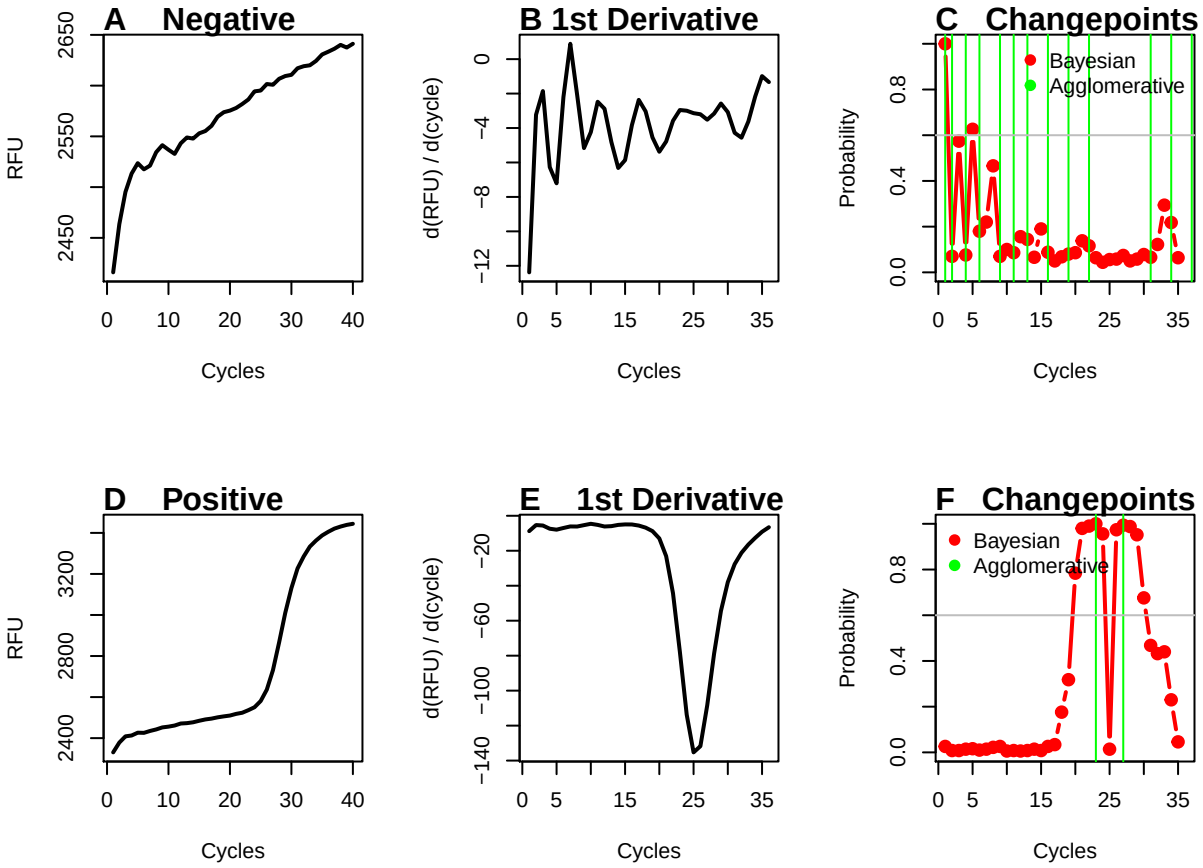

Figure 11: Bayesian and energy agglomerative change point analysis on negative and positive amplification curves. An analysis of a negative and a positive amplification curve from the 'RAS002' data set was performed using the pcrfit_single() function. In this process, the amplification curves were analysed for change points using Bayesian change point analysis and energy agglomerative clustering. A) The negative amplification curve has a base signal of approximately 2450 RFU and only a small signal increase to 2650 RFU. There is a clear indication of the signal variation (noise). B) The first negative derivative amplifies the noise so that some peaks are visible. C) The change point analysis shows changes in energy agglomerative clustering at several positions (green vertical line). The Bayesian change point analysis rarely exceeds a probability of 0.6 (grey vert line). D) The positive amplification curve has a lower base signal ($\sim$ 2450 RFU) and increases up to the 40th cycle ($\sim$ 3400 RFU). A sigmoid shape of the curve is visible. E) The first negative derivation of the positive amplification curve shows a distinctive peak with a minimum at cycle 25. F) The change point analysis in energy agglomerative clustering shows changes (green vertical line) only at two positions. The Bayesian change point analysis shows a probability higher than 0.6 (grey horizontal line) at several positions.

- The fourth test is called *Threshold test* (THt), based on the Wilcoxon rank sum test. As a simple rule the first 20% (head) and the last 15% (tail) of an amplification curve are used as input data. From this, a one-sided Wilcoxon rank sum tests of the head versus the tail is performed ($\alpha \leq 1e-02$). The output is binary coded (negative = 0, positive = 1). The output name of the `pcrfit_single()` function is `amptester_tht`.
- The fifth test is called *Signal level test* (SLt). The test compares the signals of the head and the tail by a robust "sigma" rule (median + 2 * MAD) and the comparison of the head/tail ratio. If the returned value is less than 1.25 (25 percent), then the amplification curve is likely negative. The output is binary coded (negative = 0, positive = 1). The output name of the `pcrfit_single()` function is `amptester_slt`.
- The sixth test is called *Polygon test* (pco). The pco test determines if the points in an amplification curve (like a polygon) are in a "clockwise" order. The sum over the edges result in a positive value if the amplification curve is "clockwise" and is negative if the curve is counter-clockwise. Experience states that noise is positive and "true" amplification curves are "highly" negative. In contrast to the implementation in the `amptester()` function, the result is normalized by a division to the number of PCR cycles. The output is numeric. The output name of the `pcrfit_single()` function is `amptester_polygon`.
- The seventh test is the *Slope Ratio test* (SlR). This test uses the approximated first derivative maximum, the second derivative minimum and the second derivative maximum of the amplification curve. Next, the raw fluorescence at the approximated second derivative minimum and the second derivative maximum are taken from the original data set. The fluorescence intensities are normalized to the maximum fluorescence of this data and then employed in a linear regression, using the estimated slope. The output is numeric and the output name of the `pcrfit_single()` function is `amptester_slope.ratio`.

1.3 Classified Amplification Curve Datasets

Amplification curves from different sources (e. g., detection chemistries, thermo-cyclers) were manually classified with the `humanrater()` function (subsubsection 1.4.1) or with the `tReem()` function (subsubsection 1.4.2). Raw amplification curve data were exported as comma separated values or in the Real-time PCR Data Markup Language (RDML) format via the `RDML` package. RDML is human readable data exchange format for qPCR experiments. A detailed description can be found in Rödiger et al. (2017). The following code section describes the import of an RDML file from the `PCRedux` package. The RDML file contains amplification curve data of a duplex qPCR (HPV 16 & HPV 18) performed in the CFX96 (Bio-Rad).

At least the following datasets with a dichotomous decision (positive, negative) are included.

- decision_res_batsch1.csv
- decision_res_batsch2.csv
- decision_res_batsch3.csv
- decision_res_batsch4.csv
- decision_res_batsch5.csv
- decision_res_boggy.csv
- decision_res_C126EG595.csv
- decision_res_C127EGHP.csv
- decision_res_C316.amp.csv
- decision_res_C317.amp.csv
- decision_res_C60.amp.csv
- decision_res_CD74.csv
- decision_res_competimer.csv
- decision_res_dil4reps94.csv
- decision_res_guescini1.csv
- decision_res_guescini2.csv
- decision_res_HCU32_aggR.csv
- decision_res_htPCR.csv
- decision_res_karlen1.csv
- decision_res_karlen2.csv
- decision_res_karlen3.csv
- decision_res_lc96_bACTXY.csv
- decision_res_lievens1.csv
- decision_res_lievens2.csv
- decision_res_lievens3.csv
- decision_res_RAS002.csv
- decision_res_RAS003.csv
- decision_res_reps2.csv
- decision_res_reps384.csv
- decision_res_reps3.csv
- decision_res_reps.csv
- decision_res_rutledge.csv
- decision_res_stepone_std.csv
- decision_res_testdat.csv
- decision_res_vermeulen1.csv
- decision_res_vermeulen2.csv
- decision_res_VIMCFX96_60.csv

```
library(RDML)
# Load the RDML package and use its functions to import the amplification curve
#  data
library(RDML)
filename <- system.file("RAS002.rdml", package = "PCRedux")
```

```
raw_data <- RDML$new(filename = filename)
```

The following example shows the export of the `RAS002.rdml` file from the RDML format to the csv format.

```
# Export the RDML data from the PCRedux package as the objects RAS002 and RAS003.
library(RDML)
library(PCRedux)

RAS002 <- data.frame(RDML$new(paste0(
    path.package("PCRedux"), "/", "RAS002.rdml"))$GetFData()
    )

# The obbject RAS002 can be stored in the working directory as CSV file with
# the name RAS002_amp.csv.
write.csv(RAS002, "RAS002_amp.csv", row.names = FALSE)
```

RDML data file	Device	Target gene	Detection chemistry
RAS002.rdml	CFX96	HPV16, HPV18, HPRT1	TaqMan
RAS003.rdml	CFX96	HPV16, HPV18, HPRT1	TaqMan
hookreg.rdml	Bio-Rad	various	TaqMan, DNA binding dyes

1.4 Technologies for Amplification Curve Classification and Classified Amplification Curves

Many machine learning concepts exist. One method is supervised machine learning, where the goal is to derive a property from user-defined (classified) training data. Categories such as negative, ambiguous or positive are assigned depending on the form of the amplification curve. An extensive literature research showed that there are no openly accessible classified amplification curve data sets. Open Data is meant in the sense that data are freely available, free of charge, free to use and that data can be republished, without restrictions from copyright, patents or other mechanisms of control (Kitchin 2014).

Therefore, a large number of records with amplification curves and their classification (negative, ambiguous, positive) were added to the `PCRedux` package.

For the amplification curves in Table 4, a dichotomous classification was performed (roughly sigmoid or negative amplification reaction with a flat curve shape). Consequently, this does not rule out

- if an unspecific amplification product has been synthesized,
- if a contamination has been amplified or
- if only primer-dimers have been amplified.

To answer this question, other methods such as agarose gel electrophoresis need to be used.

1.4.1 Manual Amplification Curve Classification

For machine learning and method validation, it was important to classify the amplification curves individually. In Rödiger, Burdukiewicz, and Schierack (2015), the `humanrater()` [`chipPCR`] function was described. This function was developed to help the user during the classification of amplification curves and melting curves. The user has to define classes, which get assigned to an amplification curve after an expert has entered the class in input mask.

> By convention, class labels were specified as e. g., negative (“n”), ambiguous (“a”), positive (“y”) in the `PCRedux` package.

Table 4: Classified amplification curve data sets. Decision data sets in `PCRedux`: table with results of manual classification as comma separated values. qPCR data set: name of original amplification curve data set. Package: name of the `R` package containing the amplification curves. Device: is the device used to measure the amplification reaction. `Note: The original data sets contain information about the detection chemistry used within the corresponding qPCR experiments.` AB, Applied Biosystems.

Decision Datasets in PCRedux	qPCR Dataset	Package	Device
decision_res_RAS002.csv	RAS002.rdml	`PCRedux`	CFX96, Bio-Rad
decision_res_RAS003.csv	RAS003.rdml	`PCRedux`	CFX96, Bio-Rad
decision_res_batsch1.csv	batsch1	`qpcR`	Light Cycler 1.0, Roche
decision_res_batsch2.csv	batsch2	`qpcR`	Light Cycler 1.0, Roche
decision_res_batsch3.csv	batsch3	`qpcR`	Light Cycler 1.0, Roche
decision_res_batsch4.csv	batsch4	`qpcR`	Light Cycler 1.0, Roche
decision_res_batsch5.csv	batsch5	`qpcR`	Light Cycler 1.0, Roche
decision_res_lc96_bACTXY.csv	lc96_bACTXY.rdml	`RDML`	Light Cycler 1.0, Roche
decision_res_boggy.csv	boggy	`qpcR`	Light Cycler 96, Roche
decision_res_C126EG595.csv	C126EG595	`chipPCR`	Chromo4, Bio-Rad
decision_res_C127EGHP.csv	C127EGHP	`chipPCR`	iQ5, Bio-Rad
decision_res_C316.amp.csv	C316.amp	`chipPCR`	iQ5, Bio-Rad
decision_res_C317.amp.csv	C317.amp	`chipPCR`	iQ5, Bio-Rad
decision_res_C60.amp.csv	C60.amp	`chipPCR`	iQ5, Bio-Rad
decision_res_CD74.csv	CD74	`chipPCR`	iQ5, Bio-Rad
decision_res_competimer.csv	competimer	`qpcR`	Light Cycler 480, Roche
decision_res_dil4reps94.csv	dil4reps94	`qpcR`	CFX384, Bio-Rad
decision_res_guescini1.csv	guescini1	`qpcR`	Light Cycler 480, Roche
decision_res_guescini2.csv	guescini2	`qpcR`	Light Cycler 480, Roche
decision_res_htPCR.csv	htPCR	`qpcR`	Biomark HD, Fluidigm
decision_HCU32_aggR.csv	HCU32_aggR.csv	`PCRedux`	VideoScan
decision_res_karlen1.csv	karlen1	`qpcR`	ABI Prism 7700, AB
decision_res_karlen2.csv	karlen2	`qpcR`	ABI Prism 7700, AB
decision_res_karlen3.csv	karlen3	`qpcR`	ABI Prism 7700, AB
decision_res_lievens1.csv	lievens1	`qpcR`	ABI7300, ABI
decision_res_lievens2.csv	lievens2	`qpcR`	ABI7300, ABI
decision_res_lievens3.csv	lievens3	`qpcR`	ABI7300, ABI
decision_res_reps.csv	reps	`qpcR`	MXPro3000P, Stratagene
decision_res_reps2.csv	reps2	`qpcR`	MXPro3000P, Stratagene
decision_res_reps3.csv	reps3	`qpcR`	MXPro3000P, Stratagene
decision_res_reps384.csv	reps384	`qpcR`	CFX384, Bio-Rad
decision_res_rutledge.csv	rutledge	`qpcR`	Opticon 2, MJ Research
decision_res_stepone_std.csv	stepone_std	`RDML`	StepOne, AB
decision_res_testdat.csv	testdat	`qpcR`	Light Cycler 1.0, Roche
decision_res_vermeulen1.csv	vermeulen1	`qpcR`	Light Cycler 480, Roche
decision_res_vermeulen2.csv	vermeulen2	`qpcR`	Light Cycler 480, Roche
decision_res_VIMCFX96_60.csv	VIMCFX96_60	`chipPCR`	CFX96, Bio-Rad
decision_res_kbqPCR	kbqPCR	`PCRedux`	CFX96, Bio-Rad

All amplification curve data sets listed in Table 4 were classified in interactive, semi-blinded sessions. `humanrater()` [`chipPCR`] was set to randomly select individual amplification curves. All data sets were manually classified at least three times. The `htPCR` data set (??B) was classified eight times in total (see Figure 12). Most of the amplification curves are neither unequivocal classifiable as positive or negative.

```
# Suppress messages and load the packages for reading the data of the classified
# amplification curves.
library(PCRedux)

# Load the decision_res_htPCR.csv data set from a csv file.
filename <- system.file("decision_res_htPCR.csv", package = "PCRedux")
decision_res_htPCR <- read.csv(filename)

# Create graphic device for the plot(s)
par(mfrow = c(2, 4))
for (i in 2L:9) {
    data_tmp <- table(as.factor(decision_res_htPCR[, i]))

    barplot(data_tmp, col = adjustcolor("grey", alpha.f = 0.5),
            xlab = "Class", ylab = "Counts", border = "white")
    text(c(0.7, 1.9, 3.1), rep(quantile(data_tmp, 0.25), 3),
         data_tmp, srt = 90)
    mtext(LETTERS[i - 1], cex = 1, side = 3, adj = 0, font = 2)
}
```

This approach is well suited and has been applied to classify a variety of amplification curves during the development of the `PCRedux` package. From experience this is time-consuming and tiring for large data sets, especially when the amplification curves are similar in shape. A high similarity between amplification curves exists, for example, in replicates and negative controls.

1.4.2 `tReem()` - A Function for Shape-based Group-wise Classification of Amplification Curves

The similarity of amplification curves can be used to form groups of similar shapes. The amplification curves in the groups can then be classified in bulk. In this way, a higher throughput can be achieved, a concept yet not described for the analysis of qPCR data in the literature.

The `tReem()` function was developed to perform a *shape-based group classification*. To use the `tReem()` function, the first column must contain the qPCR cycles and all subsequent columns must contain the amplification curves. Two measures of similarity are used within the `tReem()` function.

- In the first measure (default), the Pearson *correlation coefficients* (r) are determined in pairs for all combinations of the amplification curves. The correlation coefficient is a statistical measure to describe the strength of the correlation between two or more variables. The correlation coefficient r is regarded as distance between the amplification curves. r is a dimensionless value and only takes values between -1 and 1. If $r = -1$, there is a maximum reciprocal relationship. If $r = 0$ there is no correlation between the two variables. If $r = 1$, there is a maximum rectified correlation.
- In the second measure, the *Hausdorff distance* is used to determine the similarity between amplification curves. The Hausdorff distance is "the maximum of the distances from a point in any of the sets to the nearest point in the other set" (Rote 1991; Herrera et al. 2016). The amplification curves are converted within the `tReem()` function using the `qPCR2data()` function.

Both methods process the distances in the same steps. This involves the calculation of the distance matrix using the Euclidean distances of all distance measures to determine the distance between the lines of the data matrix.

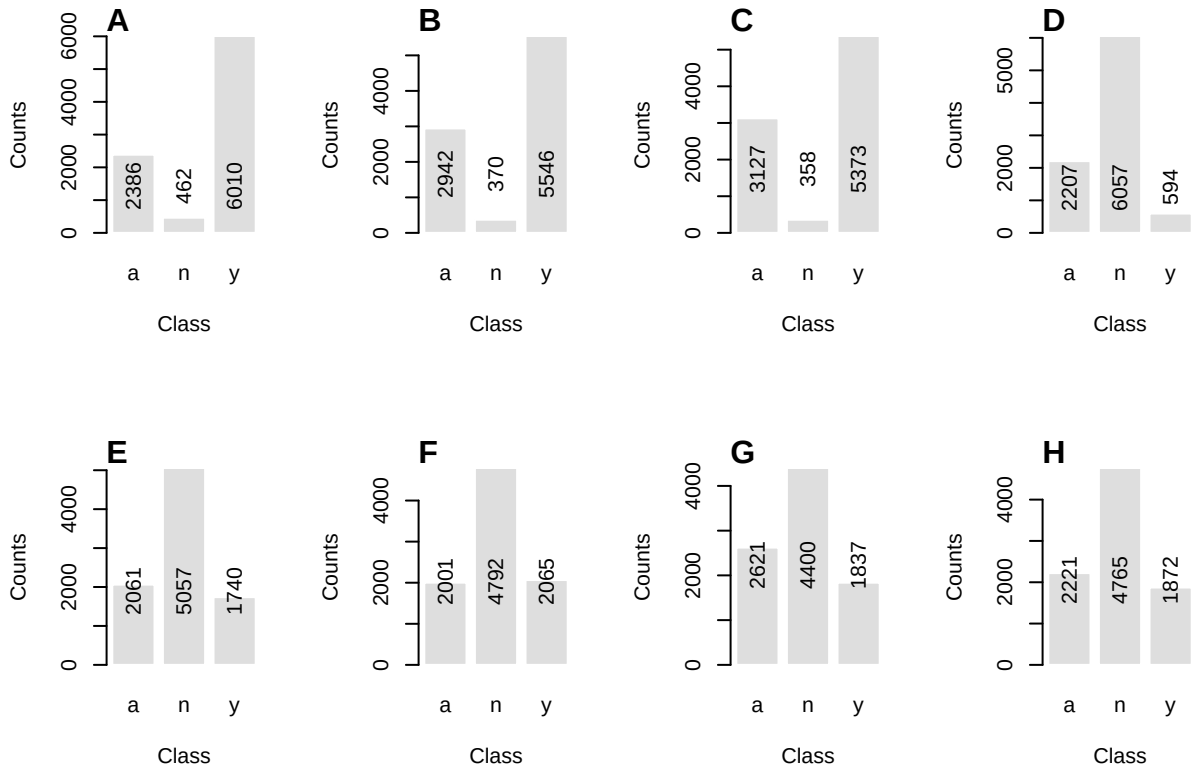

Figure 12: Variations in the classification of amplification curves. A prerequisite for the development of machine-learning models is the availability of manually classified amplification curves. Amplification curves (n = 8858) from the 'htPCR' data set have been classified by one user eight times at different points over time (classes: ambiguous (a), positive (y) or negative (n)). During this process, the amplification curves were presented in random order. The example shows that different (subjective) class mappings may occur for the same data set. While only a few amplification curves were classified as negative in the first three classification cycles (A-C), their proportion increased almost tenfold in later classification cycles (D-H).

This is used to perform a hierarchical cluster analysis. In the last step, the cluster is divided into groups based on a user-defined k value. For example, two groups are created for $k = 2$. If the amplification curves shapes are highly diverse, a larger k should be used. After a chain of processing steps presents the `tReem()` function a series of plots with grouped of amplification curves. The corresponding classes can then be assigned to the groups of amplification curves by the user using an input mask.

Grouping the amplification curves with the Pearson correlation coefficient as a distance measure is usually faster than the Hausdorff distance. The Hausdorff distance is an approximation of a shape metrics to define similarity measures between shapes. (Charpiat, Faugeras, and Keriven 2003).

```
# Classify amplification curve data by correlation coefficients (r)
data <- qpcR::testdat

classification_result <- tReem(data[, 1:15], k = 3)
classification_result
```

1.4.3 `decision_modus()` - A Function to Get a Decision (Modus) from a Vector of Classes

For the systematic statistical analysis of classification data sets, the `decision_modus()` function has been developed. This allows the most common decision (mode) to be determined. The mode is useful to consolidate large collections of different decisions into a single (most frequent) decision.

> Observed:a, a, a, a, a, n, n, n $\rightarrow$ frequencies 5 x a, 3 x n $\rightarrow$ mode: a. Since the class names are known, they only have to be interpreted by the user (e. g., "a",n,"y" -> "ambivalent","negative","positive").

A manual classification was performed out for the `htPCR` data set (for an example plot **??**B) with the `humanrater()` function. The classification of each amplification curve was performed eight times at different time points since many of the amplification curves did not resemble optimal curvatures (e. g., **??**). It is likely that the amplification curve (P06.W47, **??**) is considered as ambiguous or even positive (positive $\leftrightarrow$ ambivalent) by the users.

Table 5 shows from a total of 8858 amplification curves the first 25 lines classified as negative (*conformity=TRUE*) and the first 25 lines classified as positive. Since in total, the curves were classified eight times (`test.result.1` ... `test.result.8`) a whole of 70864 amplification curves was analysed. In this classification experiment the amplification curves have been classified differently in 94.5% of the cases (e. g., line 1 "P01. W01").

The `decision_modus()` function was applied to the record `decision_res_htPCR.csv` with all classification rounds (columns 2 to 9) and the mode was determined for each amplitude curve Figure 13.

```
# Use decision_modus() to go through each row of all classifications done by
# a human.

# Determine the number of observations where all classifications were
# the same (conformity == TRUE).
conformity <- decision_res_htPCR[["conformity"]]

# List all classifications.
dec <- lapply(1L:nrow(decision_res_htPCR), function(i) {
    decision_modus(decision_res_htPCR[i, 2:9])[1]
}) %>% unlist()

# Show statistic of the decisions
summary(dec)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
```

Table 5: Results of the ‘htPCR‘ data set classification. All amplification curves of the ‘htPCR‘ data set were classified as ‘negative‘, ‘ambiguous‘ and ‘positive‘ by individuals in eight analysis cycles (‘test.result.1‘ ... ‘test.result.8‘). If an amplification curve was always classified with the same class, the last column (‘conformity‘) shows ‘TRUE‘. As an example, the table shows 25 amplification curves with consistent classes and 25 amplification curves with differing classes (‘conformity = FALSE‘).

htPCR	test.result.1	test.result.2	test.result.3	test.result.4	test.result.5	test.result.6	test.result.7	test.result.8	conformity
P01.W01	y	a	a	n	n	n	a	n	FALSE
P01.W02	a	y	a	n	n	n	n	n	FALSE
P01.W03	y	a	a	n	n	n	n	n	FALSE
P01.W04	y	a	a	n	n	n	n	n	FALSE
P01.W05	y	a	a	n	n	n	a	n	FALSE
P01.W06	a	a	a	n	n	n	n	n	FALSE
P01.W07	a	a	a	n	n	n	n	n	FALSE
P01.W08	y	a	a	n	n	n	n	n	FALSE
P01.W09	y	a	a	n	n	n	n	n	FALSE
P01.W10	n	a	a	n	n	n	n	n	FALSE
P01.W11	y	y	a	y	y	y	a	y	FALSE
P01.W12	a	a	a	n	n	n	n	n	FALSE
P01.W13	y	a	y	n	n	n	n	n	FALSE
P01.W14	y	a	y	n	n	n	n	n	FALSE
P01.W15	y	a	y	n	n	n	n	n	FALSE
P01.W16	y	a	a	n	n	n	n	n	FALSE
P01.W17	y	a	a	n	n	n	n	n	FALSE
P01.W18	a	a	a	n	n	n	n	n	FALSE
P01.W19	y	a	a	n	n	n	a	n	FALSE
P01.W20	y	a	a	y	a	a	y	a	FALSE
P01.W21	a	a	a	n	n	n	n	n	FALSE
P01.W22	y	a	a	n	n	n	n	n	FALSE
P01.W23	y	a	a	n	n	n	n	n	FALSE
P01.W24	y	a	a	n	n	n	a	n	FALSE
P01.W25	y	a	a	n	n	n	n	n	FALSE
P01.W58	n	n	n	n	n	n	n	n	TRUE
P02.W09	y	y	y	y	y	y	y	y	TRUE
P02.W19	y	y	y	y	y	y	y	y	TRUE
P02.W31	y	y	y	y	y	y	y	y	TRUE
P02.W41	y	y	y	y	y	y	y	y	TRUE
P02.W72	y	y	y	y	y	y	y	y	TRUE
P02.W81	y	y	y	y	y	y	y	y	TRUE
P02.W84	n	n	n	n	n	n	n	n	TRUE
P03.W09	y	y	y	y	y	y	y	y	TRUE
P03.W21	y	y	y	y	y	y	y	y	TRUE
P03.W22	y	y	y	y	y	y	y	y	TRUE
P03.W31	y	y	y	y	y	y	y	y	TRUE
P03.W32	y	y	y	y	y	y	y	y	TRUE
P03.W39	y	y	y	y	y	y	y	y	TRUE
P03.W56	y	y	y	y	y	y	y	y	TRUE
P03.W59	y	y	y	y	y	y	y	y	TRUE
P03.W68	y	y	y	y	y	y	y	y	TRUE
P03.W72	y	y	y	y	y	y	y	y	TRUE
P03.W73	y	y	y	y	y	y	y	y	TRUE
P04.W19	y	y	y	y	y	y	y	y	TRUE
P04.W31	y	y	y	y	y	y	y	y	TRUE
P04.W66	y	y	y	y	y	y	y	y	TRUE
P04.W81	y	y	y	y	y	y	y	y	TRUE
P04.W91	y	y	y	y	y	y	y	y	TRUE
P05.W20	y	y	y	y	y	y	y	y	TRUE

```
##   1.000   1.000   2.000   1.779   2.000   3.000    1179
```

Another usage mode of `decision_modus()` is to set the parameter as `max_freq=FALSE`. This option specifies the number of all classifications.

```
library(PCRedux)
# Decisions for observation P01.W06
res_dec_P01.W06 <- decision_modus(decision_res_htPCR[
which(decision_res_htPCR[["htPCR"]] == "P01.W06"),
                                  2L:9
], max_freq = FALSE)
print(res_dec_P01.W06)
```

```
##   variable freq
## 1        a    3
## 2        n    5
```

The amplification curve `P01. W06` was classified as a=3 times and as n=5 times. Therefore, the decision would turn `negative`.

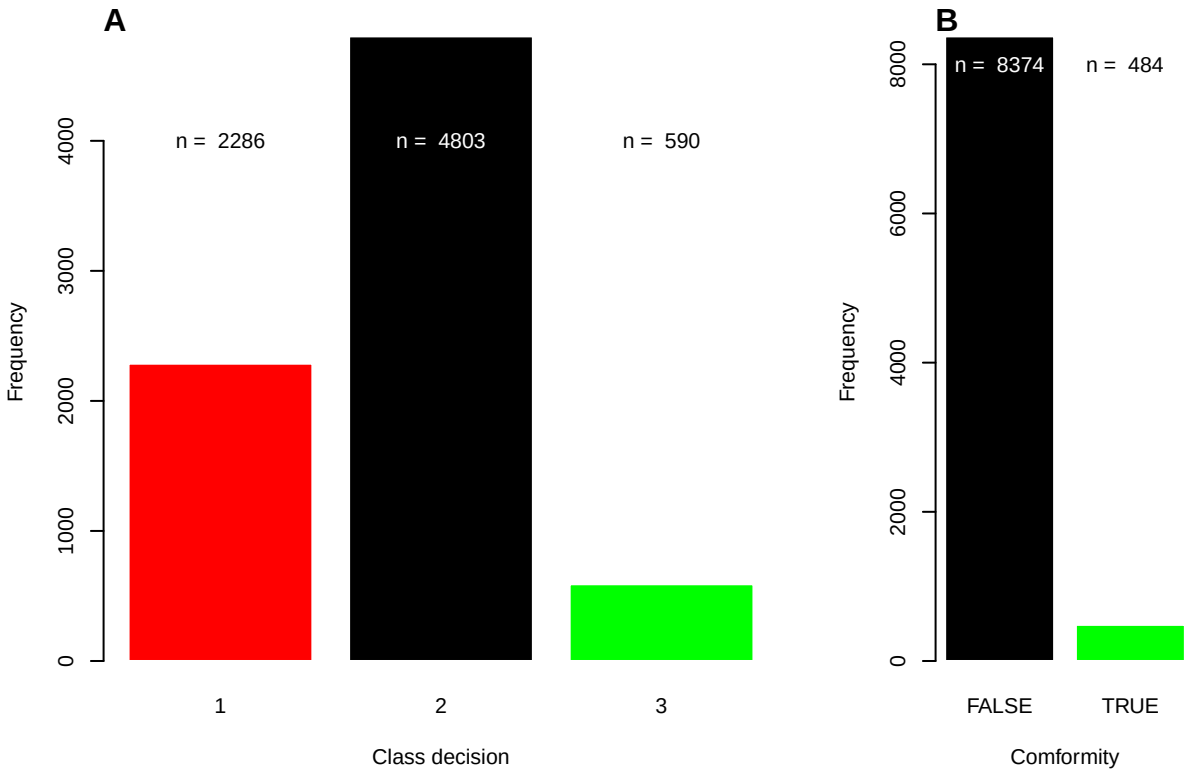

Figure 13: Frequency of amplification curve classes and conformity in the 'htPCR' data set. The 'htPCR' data set was classified by hand eight times. Due to the unusual amplification curve shape and input errors during classification, many amplification curves were classified differently. A) Frequency of negative (black), ambiguous (red) and positive (green) amplification curves in the 'htPCR' data set. The combined number of ambiguous and negative amplification curves appears to be higher, than the number of positive amplification curves. B) The number of observations where all classification cycles made the same decision (conformity == TRUE) accounts for only 5% of the total number of observations. TRUE, all classes of the amplification curve matched. FALSE, at least one in eight observations had a different class.

1.4.4 PCRedux-app

`PCRedux-app` is a web server, based on the shiny technology (Chang et al. 2019) wrapped around the `encu()` function (subsubsection 1.2.2). An user can upload qPCR data and download obtained amplification curve features.

There are different ways to use the function.

- Through `RScript` (Scripting Front-End for R):
  - Enter the command `Rscript -e 'PCRedux::run_PCRedux()'` in a console and copy the pasted URL in a browser.
- Through Graphical User Interfaces:
  - The function can be started directly in `RStudio` or `RKWard` by:

```
# run the Shiny app
PCRedux::run_PCRedux()
```

Table 6: Measures for performance analysis for binary classification. TP, true positive; FP, false positive; TN, true negative; FN, false negative

Measure	Formula
Sensitivity - TPR, true positive rate	$TPR = \frac{TP}{TP+FN}$
Specificity - SPC, true negative rate	$SPC = \frac{TN}{TN+FP}$
Precision - PPV, positive predictive value	$PPV = \frac{TP}{TP+FP}$
Negative predictive value - NPV	$NPV = \frac{TN}{TN+FN}$
Fall-out, FPR, false positive rate	$FPR = \frac{FP}{FP+TN} = 1 - SPC$
False negative rate - FNR	$FNR = \frac{FN}{TN+FN} = 1 - TPR$
False discovery rate - FDR	$FDR = \frac{FP}{TP+FP} = 1 - PPV$
Accuracy - ACC	$ACC = \frac{(TP+TN)}{(TP+FP+FN+TN)}$
F1 score - F1	$F1 = \frac{2TP}{(2TP+FP+FN)}$
Matthews correlation coefficient - MCC	$MCC = \frac{(TP*TN-FP*FN)}{\sqrt{(TP+FP)*(TP+FN)*(TN+FP)*(TN+FN)}}$
Likelihood ratio positive - LRp	$LRp = \frac{TPR}{1-SPC}$
Cohen"s kappa (binary classification)	$\kappa = \frac{p_0-p_c}{1-p_0}$

1.5 Helper Functions of the `PCRedux` Package

1.5.1 `performeR()` - Performance Analysis for Binary Classification

Statistical modeling and machine learning are powerful but expose a risk to the user by introducing an unexpected bias. This may lead to an overestimation of the performance. The assessment of the performance by sensitivity and specificity is fundamental to characterize a classifier or screening test (G. James et al. 2013). Sensitivity is the percentage of true decisions that are identified and specificity is the percentage of negative decisions that are correctly identified (Table 6). An example for the application of the `performeR()` function is shown in subsubsection 1.2.13.

1.5.2 `qPCR2fdata()` - A Helper Function to Convert Amplification Curve Data to the `fdata` Format

`qPCR2fdata()` is a helper function to convert amplification curve data to the functional `fdata` class (Febrero-Bande and Oviedo de la Fuente 2012). The `fdata` format is used for functional data analysis to determine the similarity measures between amplification curves shapes by the Hausdorff distance. Similarity herein refers to the difference in spatial location of two *objects* (e. g., amplification curves). Objects with a close distance are presumably more similar. For single objects (e. g., points) one can use a vector distance, such as the Euclidean distance (Herrera et al. 2016).

The `qPCR2fdata()` function takes a `data.frame` containing the amplification cycles (first column) and the fluorescence amplitudes (subsequent columns) as input.

Noise and missing values may affect the analysis adversely. Therefore, an instance of the `CPP()` [`chipPCR`] function (Rödiger, Burdukiewicz, and Schierack 2015) was integrated in `qPCR2fdata()`. If *preprocess=TRUE* in `qPCR2fdata()`, then all curves are smoothed (Savitzky-Golay smoother), missing values are imputed and outliers in the ground phase get removed as described in Rödiger, Burdukiewicz, and Schierack (2015).

The following example illustrates a hierarchical cluster analysis of the `testdat` data set. The amplification curves of the `testdat` data set remained as raw data or were preprocessed (smoothed). Subsequent, the amplification curves were converted by the `qPCR2fdata()`. The converted data were subjected to cluster analysis (Hausdorff distance). This method uses the elements of a proximity matrix to generate a dendrogram. The dendrogram can then be used to further analyze the clusters. There are methods to determine the

number of clusters automatically k (Cook and Swayne 2007). However, for simplicity, the number of clusters was determined visually.

```
# Calculate the Hausdorff distance of the amplification curves
# cluster the curves.
# Load additional packages for data and pipes.
library(fda.usc)

data <- qpcR::testdat

# Convert the qPCR data set to the fdata format
# Use unprocessed data from the testdat data set
res_fdata <- qPCR2fdata(data)

# Extract column names and create rainbow color to label the data
columnames <- data[-1] %>% colnames(.)
colors <- rainbow(length(columnames), alpha = 0.5)

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

res_fdata_hclust <- metric.hausdorff(res_fdata)
res_hclust <- hclust(as.dist(res_fdata_hclust))
```

The distance based on the Hausdorff metric was already calculated in the next steps involving the `cutree()` [`stats`] function to split the dendrogram into smaller junks. *A priori* it was defined that two classes (*positive* & *negative*) are expected. Therefore, the *group* parameter was set to k=2 in the `cutree()`.

```
# Cluster of the unprocessed amplification curves
res_cutree <- cutree(res_hclust, k = 2)
res_cutree <- factor(res_cutree)
levels(res_cutree) <- list(y = "1", n = "2")
```

The dendrogram shows that

- the observations are correctly assigned to a cluster of positive or negative amplification curves and that
- the shift of the Cq (late increase of the fluorescence) is reflected in the positive cluster (Figure 14).

```
# Plot the converted qPCR data
# Create graphic device for the plot(s)
par(mfrow = c(1, 2))

plot(res_fdata, xlab = "Cycles", ylab = "RFU", main = "", type = "l",
        lty = 1, lwd = 2, col = colors
)
legend(
    "topleft", paste0(as.character(columnames), ": ", res_cutree),
       pch = 19, col = colors, bty = "n", ncol = 2, cex = 0.7
)
mtext("A", cex = 1, side = 3, adj = 0, font = 2)

plot(res_hclust, main = "", xlab = "", sub="")
text(c(5.5,18), rep(6.5,2), c("negative", "positive"),
     col = c("black", "green"), cex = 0.9, srt = 90)
mtext("B", cex = 1, side = 3, adj = 0, font = 2)
```

This workflow can be used to cluster amplification curve data according to similar shape. Classification

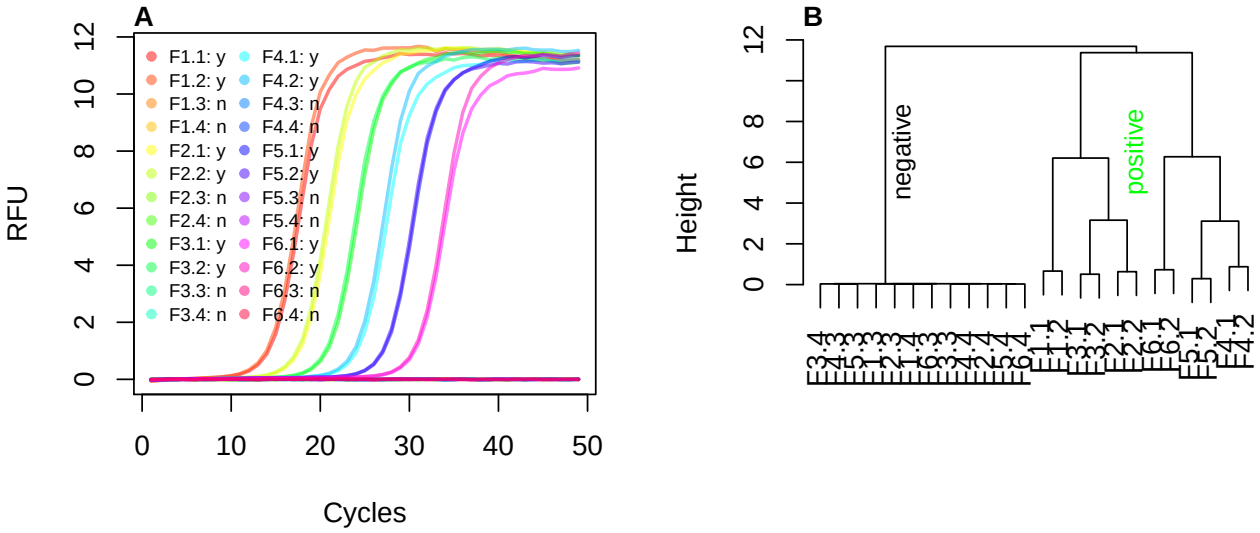

Figure 14: Shape-based clustering of amplification curves. A) The clustering of the amplification curves of the 'testdat' data set (A) was based on the Hausdorff distance. B) The amplification curves were converted with the qPCR2fdata() function, and the Hausdorff distance of the curves was determined by cluster analysis. There were no errors in distinguishing between negative (n) and positive (y) amplification curves.

tasks can be preformed in batches of amplification curves. The calculation of the distances is a computation expensive step dependent on the number of amplification curves.

The following example illustrates the usage for the `HCU32_aggR.csv` data set from the VideoScan platform with 32 heating and cooling units (equivalent of 32 PCR vessels). In this experiment, the bacterial gene *aggR* from *E. coli* was amplified in 32 replicate qPCR reactions. Details of the experiment are described in the manual of the `PCRedux` package. The ambition was to test if the 32 amplification curves of the qPCR reaction are identical. As before, the data were processed with the `qPCR2fdata()` function and compared by the Hausdorff distance. Ideally, the amplification curves form only few clusters.

```
# Calculate slope and intercept on positive amplification curve data from the
# VideoScan 32 cavity real-time PCR device.
# Use the fda.usc package for functional data analysis
library(fda.usc)

# Load the qPCR data from the HCU32_aggR.csv data set
# Convert the qPCR data set to the fdata format

filename <- system.file("HCU32_aggR.csv", package = "PCRedux")
data_32HCU <- read.csv(filename)

res_fdata <- qPCR2fdata(data_32HCU)
# Extract column names and create rainbow color to label the data
columnames <- data_32HCU[-1] %>% colnames(.)
colors <- rainbow(length(columnames), alpha = 0.55)
```

In advance the Cq values were calculated by the following code:

```
# Load the qpcR package to calculate the Cq values by the second derivative
# maximum method.
library(qpcR)

res_Cq <- sapply(2L:ncol(data_32HCU), function(i) {
```

```
    efficiency(pcrfit(data_32HCU, cyc = 1, fluo = i, model = l6))
})

data.frame(
    obs = colnames(data_32HCU)[-1],
          Cq = unlist(res_Cq["cpD2", ]), eff = unlist(res_Cq["eff", ])
)

#        Results
#
# obs    Cq      eff
# 1      A1 14.89 1.092963
# 2      B1 15.68 1.110480
# 3      C1 15.63 1.111474
# ...
# 30     F4 15.71 1.109634
# 31     G4 15.70 1.110373
# 32     H4 15.73 1.117827
```

Next, the amplification curves (Figure 15A), the differences between base-line region and plateau region (Figure 15B), the correlation between the Cq value and amplification efficiency (Figure 15C) and the clusters based on the Hausdorff distance were taken into account.

Some amplification curves (Figure 15A) had stronger noise, and all curves exhibited a negative non-linear trend and shift in the ground phase. The comparison of the ground phase and the plateau phase showed a difference between the 32 amplification curves. The observations `E1`, `F1` and `H1` were most close in the ground and plateau phase. The comparison of Cq values and amplification efficiency showed that most amplification curves are similar. However, there are also amplification curves that show a greater deviation from the median of all Cq values (Figure 15C). The cluster analysis confirmed the shape similarity (Figure 15D).

```
# Use the fda.usc package for functional data analysis
library(fda.usc)

# To save computing time, the Cq values and amplification efficiencies were
# calculated beforehand and transferred as a hard copy here.

calculated_Cqs <- c(
    14.89, 15.68, 15.63, 15.5, 15.54, 15.37, 15.78, 15.24, 15.94,
    15.88, 15.91, 15.77, 15.78, 15.74, 15.84, 15.78, 15.64, 15.61,
    15.66, 15.63, 15.77, 15.71, 15.7, 15.79, 15.8, 15.72, 15.7, 15.82,
    15.62, 15.71, 15.7, 15.73
)

calculated_effs <- c(
    1.09296326515231, 1.11047987547324, 1.11147389307153, 1.10308929700635,
    1.10012176315852, 1.09136717687619, 1.11871308210321, 1.08006168654712,
    1.09500422011318, 1.1078777171126, 1.11269436700649, 1.10628580163733,
    1.1082009954558, 1.11069683827291, 1.11074914659374, 1.10722949813473,
    1.10754282514113, 1.10098387264025, 1.1107026749644, 1.11599641663658,
    1.11388510347017, 1.11398547396991, 1.09410798249025, 1.12422338092929,
    1.11977386646464, 1.11212436173214, 1.12145338871426, 1.12180879952503,
    1.1080276005651, 1.10963449004393, 1.11037302758388, 1.11782689816295
)

# Plot the converted qPCR data
```

```r
# Create graphic device for the plot(s)

layout(matrix(c(1, 2, 3, 4, 4, 4), 2, 3, byrow = TRUE))


plot(res_fdata, xlab = "Cycles", ylab = "RFU", main = "HCU32_aggR", type = "l",
         lty = 1, lwd = 2, col = colors)

legend("topleft", as.character(columnames), pch = 19, col = colors,
       bty = "n", ncol = 4)
mtext("A", cex = 1, side = 3, adj = 0, font = 2)

# Plot the background and plateau phase.

boxplot(
    data_32HCU[, -1] - apply(data_32HCU[, -1], 2, min),
        col = colors, las = 2, main = "Signal to noise ratio",
        xlab = "Sample", ylab = "RFU"
)
mtext("B", cex = 1, side = 3, adj = 0, font = 2)

# Plot the Cqs and the amplification efficiencies.
# Determine the median of the Cq values and label all Cqs, which a less 0.1 Cqs
# of the median or more then 0.1 Cqs of the median Cq.

plot(
    calculated_Cqs, calculated_effs, xlab = "Cq (SDM)",
     ylab = "eff", main = "Cq vs. Amplification Efficiency",
     type = "p", pch = 19, lty = 1, lwd = 2, col = colors
)

median_Cq <- median(calculated_Cqs)
abline(v = median_Cq)

text(median_Cq + 0.01, 1.085, expression(paste(tilde(x))))
labeled <- c(
    which(calculated_Cqs < median_Cq - 0.1),
             which(calculated_Cqs > median_Cq + 0.1)
)

text(
    calculated_Cqs[labeled], calculated_effs[labeled],
     as.character(columnames)[labeled]
)
mtext("C", cex = 1, side = 3, adj = 0, font = 2)

# Calculate the Hausdorff distance using the fda.usc package and cluster the
# the distances.

res_fdata_hclust <- metric.hausdorff(res_fdata)
cluster <- hclust(as.dist(res_fdata_hclust))

# plot the distances as clustered data and label the leafs with the Cq values
```

```
# and colored dots.

plot(cluster, main = "Clusters of the amplification\n
curves as calculated by the Hausdorff distance", xlab = "", sub="")
mtext("D", cex = 1, side = 3, adj = 0, font = 2)
```

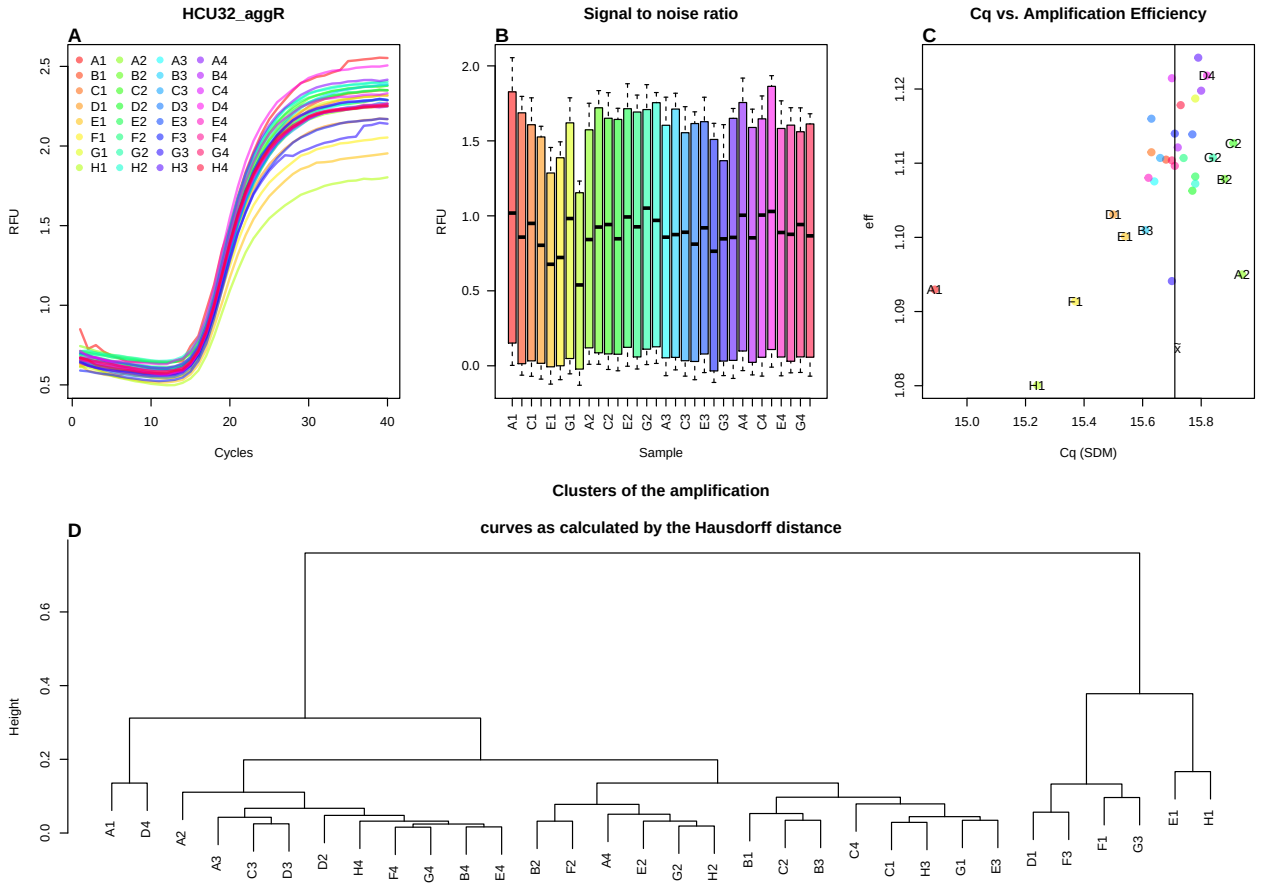

Figure 15: Clustering and variation analysis of amplification curves. The amplification curves of the 32HCU were processed with the qPCR2fdata() function and then processed by cluster analysis (Hausdorff distance). A) Amplification curves were plotted from the raw data. B) Overall, signal-to-noise ratios of the amplification curves between all cavities were similar. C) The Cq values and amplification efficiency were calculated using the efficiency(pcrfit()) [`qpcR`] function. The median Cq is shown as a vertical line. Cqs greater or less than 0. 1 of Cq $\tilde{x}$ are marked with observation labels. D) The cluster analysis showed no specific pattern with respect to the amplification curve signals. It appears that the observations D1, E1, F1, F3, G3 and H1 differ most from the other amplification curves.

The analysis gives an overview on the variation of the amplification curve data.

2 Summary and Conclusions

Extensive amounts of data represent a serious challenge in the analysis of qPCR amplification curves. In a manual classification (e. g. negative, positive), the result is usually characterized by the subjective perception of the experimenter. In addition, the time required for a manual analysis is high. An automatic system for amplification curve analysis might objectify and generalize the decision process. Interestingly, so far most authors focused on the extraction of single predictors from amplification curves. These are mainly the Cq and the amplification efficiency, which are used for the downstream processing such as expression analysis or

genotyping (Pabinger et al. 2014).

Numerous software tools were developed, which deal with theses analytical steps. For example Baebler et al. (2017) published `quantGenius` and Mallona et al. (2017) published `Chainy`. However, none of them attempts to make use of characteristics of the amplification curve. Positive amplification curves usually exhibit a sigmoid shape, consisting of a ground phase, exponential phase, and plateau phase. Negative amplification curves resemble flat noisy signal. As a result, the experienced user is usually able to correctly interpret the curve shape. Similarly, outliers and measurement errors can also be identified. When setting up a qPCR assay, manual data analysis is a useful and necessary approach to familiarize oneself with the properties of the qPCR amplification curves.

It is challenging to analyze and classify amplification curves if they deviate substantially from the sigmoid shape or if their number is no longer feasible for manual analysis. Furthermore, the supposed objectivity of the user must also be questioned. In the scientific environment there is often the temptation - or rather the compulsion - to use all data for publications. As a result, amplification curves of rather poor quality are often provided. In Figure 13 it was demonstrated that a reproducible and objective analysis of amplification curves is not always given.

References

Baebler, Miha Svalina, Marko Petek, Katja Stare, Ana Rotter, Maruša Pompe-Novak, and Kristina Gruden. 2017. "quantGenius: implementation of a decision support system for qPCR-based gene quantification." *BMC Bioinformatics* 18 (1). https://doi.org/10.1186/s12859-017-1688-7.

Barratt, Kevin, and John F. Mackay. 2002. "Improving Real-Time PCR Genotyping Assays by Asymmetric Amplification." *Journal of Clinical Microbiology* 40 (4): 1571–2. https://doi.org/10.1128/JCM.40.4.1571-1572.2002.

Burdukiewicz, Michał, Andrej-Nikolai Spiess, Konstantin A. Blagodatskikh, Werner Lehmann, Peter Schierack, and Stefan Rödiger. 2018. "Algorithms for Automated Detection of Hook Effect-Bearing Amplification Curves." *Biomolecular Detection and Quantification*, October. https://doi.org/10.1016/j.bdq.2018.08.001.

Chang, Winston, Joe Cheng, JJ Allaire, Yihui Xie, and Jonathan McPherson. 2019. *Shiny: Web Application Framework for R*. https://CRAN.R-project.org/package=shiny.

Charpiat, Guillaume, Olivier Faugeras, and Renaud Keriven. 2003. "Shape metrics, warping and statistics." In *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*, 2:II–627. IEEE. http://ieeexplore.ieee.org/abstract/document/1246758/.

Cook, Dianne, and Deborah F. Swayne. 2007. *Interactive and Dynamic Graphics for Data Analysis: With R and GGobi*. 2007 edition. 1st Ser. New York: Springer. https://doi.org/10.1007/978-0-387-71762-3.

Erdman, Chandra, John W. Emerson, and others. 2007. "bcp: an R package for performing a Bayesian analysis of change point problems." *Journal of Statistical Software* 23 (3): 1–13. https://www.jstatsoft.org/article/view/v023i03/v23i03.pdf.

Febrero-Bande, Manuel, and Manuel Oviedo de la Fuente. 2012. "Statistical Computing in Functional Data Analysis: The R Package fda.usc." *Journal of Statistical Software* 51 (4): 1–28. http://www.jstatsoft.org/v51/i04/.

Fischetti, Tony. 2019. *assertr: Assertive Programming for R Analysis Pipelines*. https://CRAN.R-project.org/package=assertr.

Herrera, Francisco, Sebastián Ventura, Rafael Bello, Chris Cornelis, Amelia Zafra, Dánel Sánchez-Tarragó, and Sarah Vluymans. 2016. *Multiple Instance Learning*. Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-47759-6.

Hothorn, Torsten, and Brian S. Everitt. 2014. *A Handbook of Statistical Analyses using R, Third Edition*. 3rd ed. Oakville: Chapman; Hall/CRC.

Hothorn, Torsten, Kurt Hornik, and Achim Zeileis. 2006. "Unbiased Recursive Partitioning: A Conditional Inference Framework." *Journal of Computational and Graphical Statistics* 15 (3): 651–74. https://doi.org/10.1198/106186006X133933.

Isaac, Peter G. 2009. "Essentials of nucleic acid analysis: a robust approach." *Annals of Botany* 104 (2): vi–vi. https://doi.org/10.1093/aob/mcp135.

James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2013. *An Introduction to Statistical Learning*. Vol. 103. Springer Texts in Statistics. New York, NY: Springer New York. https://doi.org/10.1007/978-1-4614-7138-7.

James, Nicholas A., and David S. Matteson. 2013. "ecp: An R package for nonparametric multiple change point analysis of multivariate data." *arXiv Preprint arXiv:1309.3295*. https://arxiv.org/abs/1309.3295.

Killick, Rebecca, and Idris A. Eckley. 2014. "changepoint: An R Package for Changepoint Analysis." *Journal of Statistical Software* 58 (3): 1–19. http://www.jstatsoft.org/v58/i03/.

Kitchin, Rob. 2014. *The data revolution : big data, open data, data infrastructures & their consequences*. Los Angeles, California London: SAGE Publications.

Luan, Shenghua, Lael J. Schooler, and Gerd Gigerenzer. 2011. "A signal-detection analysis of fast-and-frugal trees." *Psychological Review* 118 (2): 316–38. https://doi.org/10.1037/a0022684.

Mallona, Izaskun, Anna Díez-Villanueva, Berta Martín, and Miguel A. Peinado. 2017. "Chainy: an universal tool for standardized relative quantification in real-time PCR." *Bioinformatics*. https://doi.org/10.1093/bioinformatics/btw839.

Nolan, Tania, Rebecca E Hands, and Stephen A Bustin. 2006. "Quantification of mRNA using real-time RT-PCR." *Nature Protocols* 1 (November): 1559. https://doi.org/10.1038/nprot.2006.236.

Pabinger, Stephan, Stefan Rödiger, Albert Kriegner, Klemens Vierlinger, and Andreas Weinhäusel. 2014. "A survey of tools for the analysis of quantitative PCR (qPCR) data." *Biomolecular Detection and Quantification* 1 (1): 23–33. https://doi.org/10.1016/j.bdq.2014.08.002.

Phillips, Nathaniel, Hansjoerg Neth, Jan Woike, and Wolfgang Gaissmaer. 2017. *FFTrees: Generate, Visualise, and Evaluate Fast-and-Frugal Decision Trees*. https://CRAN.R-project.org/package=FFTrees.

Quinlan, J. Ross. 1986. "Induction of decision trees." *Machine Learning* 1 (1): 81–106. https://doi.org/10.1007/BF00116251.

Ritz, Christian, and Andrej-Nikolai Spiess. 2008. "qpcR: an R package for sigmoidal model selection in quantitative real-time polymerase chain reaction analysis." *Bioinformatics* 24 (13): 1549–51. https://doi.org/10.1093/bioinformatics/btn227.

Rote, Günter. 1991. "Computing the minimum Hausdorff distance between two point sets on a line under translation." *Information Processing Letters* 38 (3): 123–27. https://doi.org/10.1016/0020-0190(91)90233-8.

Rödiger, Stefan, Alexander Böhm, and Ingolf Schimke. 2013. "Surface Melting Curve Analysis with R." *The R Journal* 5 (2): 37–53. http://journal.r-project.org/archive/2013-2/roediger-bohm-schimke.pdf.

Rödiger, Stefan, Michał Burdukiewicz, Konstantin A. Blagodatskikh, and Peter Schierack. 2015. "R as an Environment for the Reproducible Analysis of DNA Amplification Experiments." *The R Journal* 7 (2): 127–50. http://journal.r-project.org/archive/2015-1/RJ-2015-1.pdf.

Rödiger, Stefan, Michał Burdukiewicz, and Peter Schierack. 2015. "chipPCR: an R package to pre-process raw data of amplification curves." *Bioinformatics* 31 (17): 2900–2902. https://doi.org/10.1093/bioinformatics/btv205.

Rödiger, Stefan, Michał Burdukiewicz, Andrej-Nikolai Spiess, and Konstantin Blagodatskikh. 2017. "Enabling reproducible real-time quantitative PCR research: the RDML package." *Bioinformatics*, August. https://doi.org/10.1093/bioinformatics/btx528.

Ruijter, Jan M., Peter Lorenz, Jari M. Tuomi, Michael Hecker, and Maurice J. B. van den Hoff. 2014. "Fluorescent-increase kinetics of different fluorescent reporters used for qPCR depend on monitoring chemistry, targeted sequence, type of DNA input and PCR efficiency." *Microchimica Acta*, 1–8. https://doi.org/10.1007/s00604-013-1155-8.

Ruijter, Jan M., Michael W. Pfaffl, Sheng Zhao, Andrej N. Spiess, Gregory Boggy, Jochen Blom, Robert G. Rutledge, et al. 2013. "Evaluation of qPCR curve analysis methods for reliable biomarker discovery: Bias, resolution, precision, and implications." *Methods* 59 (1): 32–46. https://doi.org/10.1016/j.ymeth.2012.08.011.

Scott, A. J., and M. Knott. 1974. "A Cluster Analysis Method for Grouping Means in the Analysis of Variance." *Biometrics* 30 (3): 507. https://doi.org/10.2307/2529204.

Seibelt, Pablo. 2017. *xray: X Ray Vision on your Datasets*. https://CRAN.R-project.org/package=xray.

Spiess, Andrej-Nikolai, Claudia Deutschmann, Michał Burdukiewicz, Ralf Himmelreich, Katharina Klat, Peter Schierack, and Stefan Rödiger. 2015. "Impact of Smoothing on Parameter Estimation in Quantitative DNA Amplification Experiments." *Clinical Chemistry* 61 (2): 379–88. https://doi.org/10.1373/clinchem.2014.230656.

Spiess, Andrej-Nikolai, Caroline Feig, and Christian Ritz. 2008. "Highly accurate sigmoidal fitting of real-time PCR data by introducing a parameter for asymmetry." *BMC Bioinformatics* 9 (1): 221. https://doi.org/10.1186/1471-2105-9-221.

Spiess, Andrej-Nikolai, Stefan Rödiger, Michał Burdukiewicz, Thomas Volksdorf, and Joel Tellinghuisen. 2016. "System-specific periodicity in quantitative real-time polymerase chain reaction data questions threshold-based quantitation." *Scientific Reports* 6 (December): 38951. https://doi.org/10.1038/srep38951.

Therneau, Terry, Beth Atkinson, and Brian Ripley. 2017. *rpart: Recursive Partitioning and Regression Trees*. https://CRAN.R-project.org/package=rpart.

Tichopad, Ales, Michael Dilger, Gerhard Schwarz, and Michael W Pfaffl. 2003. "Standardized determination of real-time PCR efficiency from a single reaction set-up." *Nucleic Acids Research* 31 (20): e122.

Tierney, Nicholas. 2017. "Visdat: Visualising Whole Data Frames." *The Journal of Open Source Software* 2 (16). The Open Journal.

Williams, Graham J. 2009. "Rattle: A Data Mining GUI for R." *The R Journal* 1 (2): 45–55. http://journal.r-project.org/archive/2009-2/RJournal_2009-2_Williams.pdf.