# RTqPCR: Functions for analysis and normalisation of pre-processing RT-qPCR data

Navneet Phogat and Matthias Kohl
Institute of Precision Medicine
Hochschule Furtwangen University, Germany

May 16, 2017

## Contents

## 1 Introduction

The package "RTqPCR" provides methods for the normalization and analysis of pre-processing real-time quantitative RTqPCR data. The this vignette we describe and demonstrate the available functions.The package "RTqPCR" is designed on the basis of experimental data of LC480 light cycler and Mx3005P RT-qPCR. If the user wants to implement the package for any other RTqPCR, then, user may change the data into one of the two formats of LC480 light cycler and Mx3005P RTqPCR. The package "RTqPCR" is divided into six parts: First, we show how the user may read the fluorescence and sample information raw data files and compute Cq values and amplification efficiencies. Second, we demonstrate, how the user may deal with Cq values and efficiencies above user-chosen threshold values, replace the particular values and deal with non-detects in Cq values and efficiencies. Third, how the user may combine technical replicates and compute standard deviation of Cq values and amplification efficiencies within replicates. Forth, how the user may compute delta Cq values and standard deviation of delta Cq, compute delta delta Cq values, standard deviation of delta delta Cq values, fold concentration (log values) based on delta delta Cq values and visualisation of the results of fold concentration in the form of bar plots too. Fifth, how the user may compute the relative expression ratio based on the three different methods - Method by Roche, Method by Pfaffl etal., ddCq (delta delta Cq) method and may visualise the results in the form of bar plots too. Sixth part is about auxiliary functions.

User may load the "RTqPCR" package.

```
> library(RTqPCR)
```

## 2 Reading the raw fluorescent data

The `read.RTqPCR` function is based on `[ReadqPCR]read.LC480` function from the "ReadqPCR" package from `Biocondcutor` software in R software and `read.Mx3005P` function. Details of `read.LC480` function is present in "ReadqPCR" package. The detail of `read.Mx3005P` is present in section `Auxiliary Function`. The `read.RTqPCR` function reads in the .txt Tab delimited file of raw fluorescent data of both LC480 light cycler and Mx3005P RTqPCR and populate an object of class `RTqPCRBatch` with new slots. Following is the example to read in the raw fluorescent data of LC480 light cycler through `read.RTqPCR` function:

```
> #library(ReadqPCR) # load the ReadqPCR library
> path <- system.file("exData", package = "RTqPCR")
> LC480.example <- file.path(path, "LC480_Example.txt")
> rtData.LC480 <- read.RTqPCR(LC480.example, PCRtype = "LC480")
> rtData.LC480 ## to express the overview of fluorescent data
```

```
RTqPCRBatch (storageMode: lockedEnvironment)
assayData: 45 features, 96 samples
  element names: exprs
protocolData: none
phenoData
  sampleNames: A1 A2 ... H12 (96 total)
  varLabels: Sample position Sample name Program number Segment number
  varMetadata: labelDescription
featureData
  featureNames: 1 2 ... 45 (45 total)
  fvarLabels: Cycle number Acquisition time Acquisition temperature
  fvarMetadata: labelDescription
experimentData: use 'experimentData(object)'
Annotation:
```

```
> head(exprs(rtData.LC480[,1:5])) ## to express the first five fluorescent data
```

```
     A1    A2    A3    A4    A5
1 15.58 16.93 27.32 16.60 16.88
2 15.53 16.96 27.69 16.68 16.91
3 15.49 16.93 27.56 16.67 16.95
4 15.47 16.97 27.62 16.63 16.87
5 15.47 16.88 27.56 16.65 16.92
6 15.41 16.90 27.55 16.65 16.87
```

```
> head(pData(rtData.LC480)) ## to express the pheno data
```

```
   Sample position Sample name Program number Segment number
A1              A1    Sample_1              2              3
A2              A2    Sample_2              2              3
A3              A3    Sample_2              2              3
A4              A4    Sample_3              2              3
A5              A5    Sample_4              2              3
A6              A6    Sample_4              2              3
```

```
> head(fData(rtData.LC480)) ## to express the feature data

  Cycle number Acquisition time Acquisition temperature
1            1           691600                   71.79
2            2           770133                   71.64
3            3           848716                   71.64
4            4           927233                   71.67
5            5          1005816                   71.67
6            6          1084316                   71.67
```

The example to read in the raw fluorescent data of Mx3005P RTqPCR through `read.RTqPCR` function is as follows:

```
> Mx3005P.example <- file.path(path, "Mx3005P_Example.txt")
> rtData.Mx <- read.RTqPCR(Mx3005P.example, PCRtype = "Mx3005P")
> rtData.Mx ## to express the overview of fluorescent data

RTqPCRBatch (storageMode: lockedEnvironment)
assayData: 45 features, 34 samples
  element names: exprs
protocolData: none
phenoData
  sampleNames: 14 15 ... 78 (34 total)
  varLabels: Well Ramp/Plateaue ... mspSegment (5 total)
  varMetadata: labelDescription
featureData
  featureNames: 1 2 ... 45 (45 total)
  fvarLabels: Cycle# Temperature
  fvarMetadata: labelDescription
experimentData: use 'experimentData(object)'
Annotation:

> head(exprs(rtData.Mx[,1:5])) ## to express the first five fluorescent data

    14   15   16   17   18
1 4827 4910 4900 4874 5127
2 4708 4801 4780 4766 5003
3 4662 4762 4721 4711 4934
4 4633 4757 4705 4688 4933
5 4634 4748 4708 4697 4928
6 4615 4749 4705 4693 4939

> head(pData(rtData.Mx)) ## to express the pheno data
```

```
   Well Ramp/Plateaue Ramp/Plateaue#   Dye mspSegment
14   14             P              2 Campy          2
15   15             P              2 Campy          2
16   16             P              2 Campy          2
17   17             P              2 Campy          2
18   18             P              2 Campy          2
20   20             P              2 Campy          2
```

```
> head(fData(rtData.Mx)) ## to express the feature data
```

```
  Cycle# Temperature
1      1        59.6
2      2        59.6
3      3        59.6
4      4        59.6
5      5        59.6
6      6        59.6
```

For other RTqPCR machines, the user needs to save the raw fluorescent data into the provided .txt Tab delimited format files of LC480 or Mx3005P.

## 3 Reading the sample information data

read.RTqPCRSampleInfo function is based on `Read.LC480SampleInfo` and `Read.Mx3005PSampleInfo` functions. The `read.RTqPCRSampleInfo` function populates sample information data into the resulting `AnnotatedDataFrame` format. The example to read in the .txt Tab delimited files of sample information data of LC480 light cycler through `read.RTqPCRSampleInfo` function, is as follows:

```
> SampleInfoLC480 <- file.path(path, "LC480_example_SampleInfo.txt")
> samInfoLC480 <- read.RTqPCRSampleInfo(SampleInfoLC480, PCRtype = "LC480")
> samInfoLC480  ## to express the overview of the sample information data
```

```
An object of class 'AnnotatedDataFrame'
  rowNames: 1 2 ... 96 (96 total)
  varLabels: Sample position Sample name ... Combined sample and target
    type (9 total)
  varMetadata: labelDescription
```

```
> head(pData(samInfoLC480))
```

```
  Sample position Sample name Replicate of Filter combination       Target name
1              A1    Sample_1           A1            465-510 negativ Kontrolle
2              A2    Sample_2           A2            465-510               goi
3              A3    Sample_2           A3            465-510                hk
4              A4    Sample_3           A1            465-510 negativ Kontrolle
5              A5    Sample_4           A2            465-510               goi
6              A6    Sample_4           A3            465-510                hk
  Sample Pref color Concentration Efficiency Combined sample and target type
1         $00FF8000            NA          2                 Target Negative
2             clRed            NA          2                  Target Unknown
3         $0030D700            NA          2                     Ref Unknown
4         clFuchsia            NA          2                 Target Negative
5            clGray            NA          2                  Target Unknown
6         $0012D7FA            NA          2                     Ref Unknown
```

To read in the .txt Tab delimited file of sample information data of Mx3005P, designed on the basis of given sample format in exData of "RTqPCR" package, the example is illustrated below:

```
> SampleInfoMx <- file.path(path, "Mx3005P_example_SampleInfo.txt")
> samInfoMx <- read.RTqPCRSampleInfo(SampleInfoMx, PCRtype = "Mx3005P")
> samInfoMx ## to express the overview of the sample information data
```

```
An object of class 'AnnotatedDataFrame'
  rowNames: 1 2 ... 34 (34 total)
  varLabels: Well Replicate of Target name Combined sample and target
    type
  varMetadata: labelDescription
```

```
> head(pData(samInfoMx))
```

```
  Well Replicate of Target name Combined sample and target type
1   14           A1         goi                  Target Unknown
2   15           A2         goi                  Target Unknown
3   16           A3          hk                     Ref Unknown
4   17           A1         goi                  Target Unknown
5   18           A2         goi                  Target Unknown
6   20           A3          hk                     Ref Unknown
```

## 4 Computing the Cq (Cycle Threshold) and amplification efficiencies

The `CqEffs` function compute the Cq values, amplification efficiencies and standard errors of amplification efficiencies for LC480 light cycler and Mx3005P RTqPCR from raw fluorescent data. The `CqEffs` function works on object of class `RTqPCRBatch` and populate the final results into an object of class `RTqPCRBatch` with new slots. CqEffs function is based on the function `[qpcR]pcrbatch` from `"qpcR"`. The default method of computation is the sigmoidal model. To compute the Cq values and amplification efficiencies, first of all we need to merge the fluorescent and sample information data. Following is the example to merge the fluorescent and sample information data of LC480 light cycler:

```
> ## To merge the fluorescent and sample information data
> merge.LC480<-merge(rtData.LC480,samInfoLC480)
> ## to express the overview of merge data
> merge.LC480

RTqPCRBatch (storageMode: lockedEnvironment)
assayData: 45 features, 96 samples
  element names: exprs
protocolData: none
phenoData
  sampleNames: 1 2 ... 96 (96 total)
  varLabels: Sample position Sample name ... Combined sample and target
    type (11 total)
  varMetadata: labelDescription
featureData
  featureNames: 1 2 ... 45 (45 total)
  fvarLabels: Cycle number Acquisition time Acquisition temperature
  fvarMetadata: labelDescription
experimentData: use 'experimentData(object)'
Annotation:
```

Following is the example to compute the CqValues, amplification efficiencies and standard errors of amplification efficiencies of LC480 light cycler data by default sigmoidal method:

```
> ## To compute the CqValues and amplification efficiencies by sigmoidal model
> res.LC <- CqEffs(merge.LC480, PCRtype = "LC480", baseline = "none")

> #to see the overview of data
> res.LC
```

```
RTqPCRBatch (storageMode: lockedEnvironment)
assayData: 96 features, 1 samples
  element names: effs, exprs, se.effs
protocolData: none
phenoData: none
featureData
  featureNames: 1 *2* ... 96 (96 total)
  fvarLabels: Sample position Sample name ... sig.model (35 total)
  fvarMetadata: labelDescription
experimentData: use 'experimentData(object)'
Annotation:
```

```
> #to express the first five CqValues
> exprs(res.LC)[1:5]
```

```
[1] 45.00    NA 24.87 29.37    NA
```

```
> #to express the first five amplification efficiencies
> effs(res.LC)[1:5]
```

```
[1] 1.0024280        NA 0.6576489 1.0003550        NA
```

```
> #to express the first five standard errors of amplification efficiencies
> se.effs(res.LC)[1:5]
```

```
[1] 4.560011e-02           NA 6.429260e+03 3.701013e-02           NA
```

To express all the Cq values, amplification efficiencies and standard errors in amplification efficiencies, user can implement the following code:

```
> ## To express all Cq values
> exprs(res.LC)
> ## To express all amplification efficiencies
> effs(res.LC)
> ## To express standard errors in amplification efficiencies
> se.effs(res.LC)
```

To compute the Cq values, amplification efficiencies and standard error of amplification efficiencies of LC480 light cycler data by fit exponential method:

```
> res.LC1<- CqEffs(merge.LC480, PCRtype = "LC480", Effmethod = "expfit",baseline = "none")
> res.LC1
> exprs(res.LC1)[1:5]
> effs(res.LC1)[1:5]
> se.effs(res.LC1)[1:5]
> exprs(res.LC1)
> effs(res.LC1)
> se.effs(res.LC1)
```

To compute the Cq values, amplification efficiencies and standard error of amplification efficiencies of LC480 light cycler data by window of linearity method:

```
> res.LC2<- CqEffs(merge.LC480, PCRtype = "LC480", Effmethod = "sliwin", baseline = "none")
> res.LC2
> exprs(res.LC2)[1:5]
> effs(res.LC2)[1:5]
> se.effs(res.LC2)[1:5]
> exprs(res.LC2)
> effs(res.LC2)
> se.effs(res.LC2)
```

To compute the Cq values, amplification efficiencies and standard errors of amplification efficiencies of LC480 light cycler data by linear regression of efficiency method:

```
> res.LC3<- CqEffs(merge.LC480, PCRtype = "LC480", Effmethod = "LRE", baseline = "none")
> res.LC3
> exprs(res.LC3)[1:5]
> effs(res.LC3)[1:5]
> se.effs(res.LC3)[1:5]
> exprs(res.LC3)
> effs(res.LC3)
> se.effs(res.LC3)
```

To compute the Cq values and amplification efficincies for Mx3005P RTqPCR, user needs to implement the `"Mx3005"` in argument `PCRtype` of `CqEffs` function. Remaining arguments will remain same as implemented above for computation of Cq values and amplification efficiencies for LC480 light cycler data. For example, like LC480 light cycler, here we first merge the fluorescent and sample information data of Mx3005P RTqPCR and then, compute the Cq values, amplification efficiencies and standard errors of amplification efficiencies for Mx3005 RTqPCR. Following is the example to merge the data:

```
> ## To merge the fluorescent and sample information data
> merge.Mx<-merge(rtData.Mx,samInfoMx)
```

Following is the example to compute Cq values and amplification efficiencies for Mx3005P RTqPCR by default sigmoidal model:

```
> ## To compute the CqValues and amplification efficiencies
> ##by default method (sigmoidal model)
> res.Mx <- CqEffs(merge.Mx, PCRtype = "Mx3005P", baseline = "none")

> #To express the overview of data
> res.Mx

RTqPCRBatch (storageMode: lockedEnvironment)
assayData: 34 features, 1 samples
  element names: effs, exprs, se.effs
protocolData: none
phenoData: none
featureData
  featureNames: 1 2 ... 34 (34 total)
  fvarLabels: Well Ramp/Plateaue ... sig.model (32 total)
  fvarMetadata: labelDescription
experimentData: use 'experimentData(object)'
Annotation:

> #to express the first five CqValues
> exprs(res.Mx)[1:5]

[1] 35.39 30.93 26.17 23.10 19.79

> #to express the first five amplification efficiencies
> effs(res.Mx)[1:5]

[1] 1.097644 1.093488 1.100702 1.108600 1.104106

> #to express the first five standard errors of efficiencies
> se.effs(res.Mx)[1:5]

[1] 34.12339 30.02550 44.42449 39.89076 37.30819
```

To visualise all the CqValues and amplification efficiencies and standard errors of amplification efficiencies, user can implement following code:

```
> ## To express all Cq values
> exprs(res.Mx)
> ## To express all amplification efficiencies
> effs(res.Mx)
> ## To express all standard errors in amplification efficiencies
> se.effs(res.Mx)
```

Following is the example to compute the Cq values, amplification efficiencies and standard errors of amplification efficiencies for Mx3005P RTqPCR by fit exponential method:

```
> ## To compute the CqValues and amplification efficiencies by fit exponential method
> res.Mx1 <- CqEffs(merge.Mx, PCRtype = "Mx3005P", Effmethod = "expfit", baseline = "none")
> res.Mx1
> exprs(res.Mx1)[1:5]
> effs(res.Mx1)[1:5]
> se.effs(res.Mx1)[1:5]
> exprs(res.Mx1)
> effs(res.Mx1)
> se.effs(res.Mx1)
```

Following is the example to compute the Cq values, amplification efficiencies and standard errors of amplification efficiencies for Mx3005P RTqPCR by window of liearity method:

```
> # To compute the CqValues and amplification efficiencies by window of linearity (sliwin)
> res.Mx2 <- CqEffs(merge.Mx, PCRtype = "Mx3005P", Effmethod = "sliwin", baseline = "none")
> res.Mx2
> exprs(res.Mx2)[1:5]
> effs(res.Mx2)[1:5]
> se.effs(res.Mx2)[1:5]
> exprs(res.Mx2)
> effs(res.Mx2)
> se.effs(res.Mx2)
```

Following is the example to compute the Cq values, amplification efficiencies and standard errors of amplification efficiencies for Mx3005P RTqPCR by linear regression of efficiency method:

```
> res.Mx3 <- CqValues(merge.Mx, PCRtype = "Mx3005P", Effmethod = "LRE", baseline = "none")
> res.Mx3
> exprs(res.Mx3)[1:5]
> effs(res.Mx3)[1:5]
> se.effs(res.Mx3)[1:5]
> exprs(res.Mx3)
> effs(res.Mx3)
> se.effs(res.Mx3)
```

## 5 Replacing the values and removing the non-detects of Cq values and amplification efficiencies

This part is further divided into four different subsections, based on the functions. The subsections are described in detail as follows:

### 5.1 ReplaceAboveCutOff

The function `ReplaceAboveCutOff` is specifically designed to replace the Cq values and amplification efficiencies above a cut off level by `NA`. The cut off level for Cq values and efficiencies is defined by the user based on the values and his experience with analysis of RT-qPCR data. Even then, it is recommended that the user should set the Cq cut off as a non-negative value below 40 and amplification efficiency cut off as non negative value below 2. The user can define the cutoff value for Cq and amplification efficiency based on his experimental data. The `ReplaceAboveCutOff` function uses the object of class `RTqPCRBatch` and returns the resulting data as an object of class `RTqPCRBatch`. The output of `CqEffs`, `ReplaceNAs`, `ReplaceValue`, `NonDetects` and `ReplaceAboveCutOff` can be implemented as an input for the `ReplaceAboveCutOff` function. Following is the example to implement the `ReplaceAboveCutOff` function:

```
> Cqeffs.cutoff <- ReplaceAboveCutOff(res.LC,NewVal = NA, Cqcutoff = 37, effscutoff = 2)
> ## to visualise the overview of the resulting data
> Cqeffs.cutoff
```

```
RTqPCRBatch (storageMode: lockedEnvironment)
assayData: 96 features, 1 samples
  element names: effs, exprs, se.effs
protocolData: none
phenoData: none
featureData
  featureNames: 1 *2* ... 96 (96 total)
  fvarLabels: Sample position Sample name ... sig.model (35 total)
  fvarMetadata: labelDescription
experimentData: use 'experimentData(object)'
Annotation:
```

```
> ## to express the first five Cq values
> exprs(Cqeffs.cutoff)[1:5]
```

```
[1]    NA    NA 24.87 29.37    NA
```

```
> ## to express the first five amplification efficiencies
> effs(Cqeffs.cutoff)[1:5]
```

```
[1] 1.0024280        NA 0.6576489 1.0003550        NA
```

## 5.2 ReplaceValue

The `ReplaceValue` function provides an opportunity to replace any value of Cq and amplification efficiency by a desired value, including `NA` too. This function is very useful, when because of the experimental error, manual error, instrumental error or software error the Cq and amplification efficiency for a negative sample is produced. In this case, the user can replace the values of the specific negative sample by non detects (NA). This function is also of greater use, if the user wants to replace any specific non detects by a value, may be by the value of another perfectly expressed technical replicate. The `ReplaceValue function` works on an object of class `RTqPCRBatch` and returns the resulting data too in the format of class `RTqPCRBatch`. The input for the `ReplaceValue` function can be output of any of the functions, including `CqEffs`, `ReplaceNAs`, `NonDetects`, `ReplaceAboveCutOff`, `ReplaceValue`. Following is the example of implementation of `ReplaceValue` function:

```
> Cqeffs.replace <- ReplaceValue(res.LC, NewCq = 36, Neweffs = 1, Cqrow = 24, effsrow = 24)
> ## to visualise the overview of the resulting data
> Cqeffs.replace

RTqPCRBatch (storageMode: lockedEnvironment)
assayData: 96 features, 1 samples
  element names: effs, exprs, se.effs
protocolData: none
phenoData: none
featureData
  featureNames: 1 *2* ... 96 (96 total)
  fvarLabels: Sample position Sample name ... sig.model (35 total)
  fvarMetadata: labelDescription
experimentData: use 'experimentData(object)'
Annotation:

> ## to visualise the replaced value of Cq
> exprs(Cqeffs.replace)[24]

[1] 36

> ## to visualise the replaced value of amplification efficiency
> effs(Cqeffs.replace)[24]

[1] 1
```

## 5.3 ReplaceNAs

The `ReplaceNAs` function is designed for replacing the non detects by a defined value. The defined value depends on the user. The `ReplaceNAs` function works on the object of class `RTqPCRBatch` and returns the resulting data too in the same format of class `RTqPCR-Batch`. The output of any of the functions, including `CqEffs`, `ReplaceAboveCutOff`, `Re-placeValue`, `NonDetects` can be implemented as an input for the `ReplaceNAs` function. Following is an example to execute `ReplaceNAs` function:

```
> Cqeffs.NA <- ReplaceNAs(res.LC, NewCqNA = 30, NeweffsNA = 1)
> ## to visualise the overview of the resulting data
> Cqeffs.NA

RTqPCRBatch (storageMode: lockedEnvironment)
assayData: 96 features, 1 samples
  element names: effs, exprs, se.effs
protocolData: none
phenoData: none
featureData
  featureNames: 1 *2* ... 96 (96 total)
  fvarLabels: Sample position Sample name ... sig.model (35 total)
  fvarMetadata: labelDescription
experimentData: use 'experimentData(object)'
Annotation:

> ## to express first five Cq values
> exprs(Cqeffs.NA)[1:5]

[1] 45.00 30.00 24.87 29.37 30.00

> ## to express first five amplification efficiencies
> effs(Cqeffs.NA)[1:5]

[1] 1.0024280 1.0000000 0.6576489 1.0003550 1.0000000
```

## 5.4 NonDetects

The non detects in the Cq values and amplification efficiencies can be replaced by using the `NonDetects` function. This function is based on replacing the non detects within the technical replicates based on the mean or median values of the technical replicates of the sample. In the case of a sample, if all the technical replicates are present as non detects (NAs), then the non detects for that sample will not be replaced. It will replace the non detects in a sample, if atleast one of the technical replicate of a sample has expressed. The

NonDetects function works on the object of class RTqPCRBatch and returns the resulting data too in the same format of class RTqPCRBatch. The output of any of the functions, including CqEffs, ReplaceAboveCutOff, ReplaceValue can be implemented as an input for the NonDetects function. The default method of replacing the non detects is mean. Following is the example to replace the non detects on the basis of mean value within technical replicates and display of the first ten Cq and amplification efficiencies:

```
> Cqeffs.nondetects <- NonDetects(res.LC)
> ## to express the overview of the resulting data
> Cqeffs.nondetects
```

```
RTqPCRBatch (storageMode: lockedEnvironment)
assayData: 96 features, 1 samples
  element names: effs, exprs
protocolData: none
phenoData: none
featureData
  featureNames: 1 *2* ... 96 (96 total)
  fvarLabels: Sample position Sample name ... sig.model (35 total)
  fvarMetadata: labelDescription
experimentData: use 'experimentData(object)'
Annotation:
```

```
> ## to express the first ten Cq values
> exprs(Cqeffs.nondetects)[1:10]
```

```
 [1] 45.00    NA 24.87 29.37    NA 12.36  2.49    NA 20.69  2.49
```

```
> ## to express the first ten amplification efficiencies
> effs(Cqeffs.nondetects)[1:10]
```

```
 [1] 1.0024280        NA 0.6576489 1.0003550        NA 0.5977603 1.0000910
 [8]        NA 0.5882534 1.0000910
```

Following is the example to replace the non detects on the basis of median value within technical replicates and display of the first ten Cq and amplification efficiencies:

```
> Cqeffs.nondetects1 <- NonDetects(res.LC, Calc = "Median")
> ## to express the overview of the resulting data
> Cqeffs.nondetects1
> ## to express the first ten Cq values
> exprs(Cqeffs.nondetects1)[1:10]
> ## to express the first ten amplification efficiencies
> effs(Cqeffs.nondetects1)[1:10]
```

Important Note: Before starting the further steps, we need to look on our final Cq values and amplification efficiencies, particularly of reference samples. The values of all the technical replicates of reference samples and positive calibrators of target and reference samples should not be NA, because reference samples are very important to compute the final fold concentration and relative expression ratio. If the final computed values of fold concentration and relative expression ratio has high number of NA, then the user needs to look the Cq values and amplification efficiencies of reference samples. The user can change the particular value by implementing the `ReplaceValue` function.

## 6 Combining the replicates, based on Cq values and amplification efficiencies

User can combine technical replicates of Cq values by default method (mean) and compute their standard deviations within replicates, through following code:

```
> ## Combine technical replicates of Cq values by default method (mean)
> Cqreps <- CombineTechReps(res.LC)
> ## to visualise the overview of the resulting data
> Cqreps

RTqPCRBatch (storageMode: lockedEnvironment)
  element names:
protocolData: none
phenoData: none
featureData
  featureNames: 1 2 ... 30 (30 total)
  fvarLabels: ID Cq ... Combined sample and target type (5 total)
  fvarMetadata: labelDescription
experimentData: use 'experimentData(object)'
Annotation:

> ## to visualise the resulting data
> fData(Cqreps)[,c("ID", "Cq","sd.Cq")]

      ID    Cq  sd.Cq
1      1 37.19 11.052
2      3 18.62  8.846
3  **7**  2.49     NA
4      9 20.93  0.332
5     15 18.44  0.245
6     21 18.50     NA
```

```
7  **22** 14.96 19.742
8      23 25.70     NA
9      27 12.60 16.405
10 **32**  7.94     NA
11     34  1.87     NA
12     36 13.43     NA
13 **37** 11.00     NA
14     42 20.09     NA
15     45 12.61     NA
16     51 17.96  8.224
17     54 11.33  6.527
18     58 17.67  0.594
19     60 19.55  3.437
20     63 19.30     NA
21     66 17.14  3.824
22     67 10.26  6.237
23 **74** 29.30     NA
24 **80** 10.48  0.396
25 **84** 26.82     NA
26     86 30.32  1.054
27     88 13.58     NA
28     91 12.28  3.745
29     94  6.91     NA
30 **95**  8.05     NA
```

```
> ## to visualise the resulting data along with the other information of replicates
> fData(Cqreps)
```

```
       ID    Cq  sd.Cq       Target name Combined sample and target type
1       1 37.19 11.052 negativ Kontrolle                 Target Negative
2       3 18.62  8.846                hk                     Ref Unknown
3   **7**  2.49     NA negativ Kontrolle                 Target Negative
4       9 20.93  0.332                hk                     Ref Unknown
5      15 18.44  0.245                hk                     Ref Unknown
6      21 18.50     NA                hk                     Ref Unknown
7  **22** 14.96 19.742               goi           Target PosCalibrator
8      23 25.70     NA               goi                  Target Unknown
9      27 12.60 16.405                hk                     Ref Unknown
10 **32**  7.94     NA               goi                  Target Unknown
11     34  1.87     NA               goi           Target PosCalibrator
12     36 13.43     NA                hk                     Ref Unknown
```

```
13 **37** 11.00     NA               goi            Target PosCalibrator
14     42 20.09     NA                hk                    Ref Unknown
15     45 12.61     NA                hk                    Ref Unknown
16     51 17.96  8.224                hk                    Ref Unknown
17     54 11.33  6.527                hk                    Ref Unknown
18     58 17.67  0.594                hk               Ref PosCalibrator
19     60 19.55  3.437                hk                    Ref Unknown
20     63 19.30     NA                hk                    Ref Unknown
21     66 17.14  3.824                hk                    Ref Unknown
22     67 10.26  6.237                hk               Ref PosCalibrator
23 **74** 29.30     NA               goi                 Target Unknown
24 **80** 10.48  0.396               goi                    Ref Unknown
25 **84** 26.82     NA                hk                    Ref Unknown
26     86 30.32  1.054               goi                 Target Unknown
27     88 13.58     NA               goi            Target PosCalibrator
28     91 12.28  3.745               goi                 Target Unknown
29     94  6.91     NA               goi                 Target Unknown
30 **95**  8.05     NA               goi                 Target Unknown
```

Combine technical replicates of amplification efficiencies and compute their standard deviation, by method of mean:

```
> ## Combine technical replicates of amplification efficiencies by default method (mean)
> Effsreps <- CombineTechReps(res.LC, cRepCq = FALSE)
> ## to visualise the overview of the resulting data
> Effsreps
```

```
RTqPCRBatch (storageMode: lockedEnvironment)
  element names:
protocolData: none
phenoData: none
featureData
  featureNames: 1 2 ... 30 (30 total)
  fvarLabels: ID effs ... Combined sample and target type (7 total)
  fvarMetadata: labelDescription
experimentData: use 'experimentData(object)'
Annotation:
```

```
> ## to visualise the resulting data
> fData(Effsreps)[,c("ID", "effs","sd.effs","Cq","sd.Cq")]
```

```
       ID     effs  sd.effs    Cq  sd.Cq
1       1    1.001 0.001466 37.19 11.052
```

```
2        3    0.628 0.042348 18.62  8.846
3    **7**    1.000       NA  2.49     NA
4        9    0.604 0.022614 20.93  0.332
5       15    0.760 0.293195 18.44  0.245
6       21    1.100       NA 18.50     NA
7   **22**    1.024 0.033736 14.96 19.742
8       23 2209.600       NA 25.70     NA
9       27    0.361 0.382880 12.60 16.405
10  **32**    0.281       NA  7.94     NA
11      34    0.879       NA  1.87     NA
12      36    0.663       NA 13.43     NA
13  **37**    0.359       NA 11.00     NA
14      42    1.096       NA 20.09     NA
15      45    0.511       NA 12.61     NA
16      51    0.915 0.367514 17.96  8.224
17      54    0.766 0.308475 11.33  6.527
18      58    0.822 0.381787 17.67  0.594
19      60    0.896 0.275894 19.55  3.437
20      63    1.097       NA 19.30     NA
21      66    0.675 0.065095 17.14  3.824
22      67    0.721 0.109550 10.26  6.237
23  **74**    0.978       NA 29.30     NA
24  **80**    0.988 0.000168 10.48  0.396
25  **84**    0.992       NA 26.82     NA
26      86    1.047 0.264689 30.32  1.054
27      88    1.132       NA 13.58     NA
28      91    1.047 0.118480 12.28  3.745
29      94    0.979       NA  6.91     NA
30  **95**    0.894       NA  8.05     NA
```

```
> ## to visualise the resulting data along with the other information of replicates
> fData(Effsreps)
```

```
        ID     effs  sd.effs    Cq  sd.Cq       Target name
1        1    1.001 0.001466 37.19 11.052 negativ Kontrolle
2        3    0.628 0.042348 18.62  8.846                hk
3    **7**    1.000       NA  2.49     NA negativ Kontrolle
4        9    0.604 0.022614 20.93  0.332                hk
5       15    0.760 0.293195 18.44  0.245                hk
6       21    1.100       NA 18.50     NA                hk
7   **22**    1.024 0.033736 14.96 19.742               goi
```

```
8       23 2209.600        NA 25.70     NA                goi
9       27    0.361 0.382880 12.60 16.405                  hk
10 **32**    0.281        NA  7.94     NA                goi
11      34    0.879        NA  1.87     NA                goi
12      36    0.663        NA 13.43     NA                 hk
13 **37**    0.359        NA 11.00     NA                goi
14      42    1.096        NA 20.09     NA                 hk
15      45    0.511        NA 12.61     NA                 hk
16      51    0.915 0.367514 17.96  8.224                  hk
17      54    0.766 0.308475 11.33  6.527                  hk
18      58    0.822 0.381787 17.67  0.594                  hk
19      60    0.896 0.275894 19.55  3.437                  hk
20      63    1.097        NA 19.30     NA                 hk
21      66    0.675 0.065095 17.14  3.824                  hk
22      67    0.721 0.109550 10.26  6.237                  hk
23 **74**    0.978        NA 29.30     NA                goi
24 **80**    0.988 0.000168 10.48  0.396                 goi
25 **84**    0.992        NA 26.82     NA                 hk
26      86    1.047 0.264689 30.32  1.054                 goi
27      88    1.132        NA 13.58     NA                goi
28      91    1.047 0.118480 12.28  3.745                 goi
29      94    0.979        NA  6.91     NA                goi
30 **95**    0.894        NA  8.05     NA                goi
   Combined sample and target type
1                 Target Negative
2                     Ref Unknown
3                 Target Negative
4                     Ref Unknown
5                     Ref Unknown
6                     Ref Unknown
7            Target PosCalibrator
8                  Target Unknown
9                     Ref Unknown
10                 Target Unknown
11           Target PosCalibrator
12                    Ref Unknown
13           Target PosCalibrator
14                    Ref Unknown
15                    Ref Unknown
16                    Ref Unknown
17                    Ref Unknown
```

```
18               Ref PosCalibrator
19                     Ref Unknown
20                     Ref Unknown
21                     Ref Unknown
22               Ref PosCalibrator
23                  Target Unknown
24                     Ref Unknown
25                     Ref Unknown
26                  Target Unknown
27            Target PosCalibrator
28                  Target Unknown
29                  Target Unknown
30                  Target Unknown
```

Combine technical replicates based on Cq, by method of median:

```
> ## Combine technical replicates of Cq values by default method (mean)
> Cqreps1 <- CombineTechReps(res.LC, calc = "Median")
> ## to visualise the overview of the resulting data
> Cqreps1
> ## to visualise the resulting data
> fData(Cqreps1)[,c("ID", "Cq","sd.Cq")]
> ## to visualise the resulting data along with the other information of replicates
> fData(Cqreps1)
```

Combine technical replicates based on amplification efficiencies, by method of median:

```
> ## Combine technical replicates of amplification efficiencies by default method (mean)
> Effsreps1 <- CombineTechReps(res.LC, calc = "Median", cRepCq = FALSE)
> ## to visualise the overview of the resulting data
> Effsreps1
> ## to visualise the resulting data
> fData(Effsreps1)[,c("ID", "effs","sd.effs","Cq","sd.Cq")]
> ## to visualise the resulting data along with the other information of replicates
> fData(Effsreps1)
```

Combine technical replicates, based on Cq values by method of geometric mean:

```
> ## Combine technical replicates of Cq values by default method (mean)
> Cqreps2 <- CombineTechReps(res.LC, calc = "Geom")
> ## to visualise the overview of the resulting data
> Cqreps2
> ## to visualise the resulting data
> fData(Cqreps2)[,c("ID", "Cq","sd.Cq")]
> ## to visualise the resulting data along with other information of replicates
> fData(Cqreps2)
```

Combine technical replicates, based on amplification efficiencies by method of geometric mean:

```
> ## Combine technical replicates of Cq values by Geometric mean method
> Effsreps2 <- CombineTechReps(res.LC, calc = "Geom", cRepCq = FALSE)
> ## to visualise the overview of the resulting data
> Effsreps2
> ## to visualise the resulting data
> fData(Effsreps2)[,c("ID", "effs","sd.effs","Cq","sd.Cq")]
> ## to visualise the resulting data along with other information of replicates
> fData(Effsreps2)
```

## 7 Compute delta Cq, delta delta Cq values and fold concentration of the Target samples

This step is further divided into two parts: i) Computing delta Cq values of the combined technical replicates, based on the Cq values. ii) Computing the delta delta Cq values and log of fold concentration of the samples. Both the parts are illustrated in further subsections.

### 7.1 Compute delta Cq values

The delta Cq value can be computed by implementing the `DeltaCq` function on the combined Cq values of technical replicates. These combined Cq values can be produced from the `CombineTechReps` function. The object of class `RTqPCRBatch`, which is an output of `CombineTechReps` function, as the input and will populate an object of class `RTqPCRBatch` with new slots as an output. The following code can be implemented to calculate the delta Cq values:

```
> ## Combine technical replicates of Cq values by default method (mean)
> deltaCq <- DeltaCq(Cqreps, Ref = "hk")
> ## to visualise the overview of the resulting data
> deltaCq

RTqPCRBatch (storageMode: lockedEnvironment)
  element names:
protocolData: none
phenoData: none
```

```
featureData
  featureNames: 8 10 ... 27 (11 total)
  fvarLabels: ID deltaCq ... Combined sample and target type (5 total)
  fvarMetadata: labelDescription
experimentData: use 'experimentData(object)'
Annotation:
```

```
> ## To visualise the results of delta Cq
> fData(deltaCq)
```

```
       ID   deltaCq sd.deltaCq Target name Combined sample and target type
8      23   8.03429         NA         goi                  Target Unknown
10 **32**  -9.72571         NA         goi                  Target Unknown
23 **74**  11.63429         NA         goi                  Target Unknown
26     86  12.65429       6.07         goi                  Target Unknown
28     91  -5.38571       7.06         goi                  Target Unknown
29     94 -10.75571         NA         goi                  Target Unknown
30 **95**  -9.61571         NA         goi                  Target Unknown
7  **22**  -2.70571      20.63         goi            Target PosCalibrator
11     34 -15.79571         NA         goi            Target PosCalibrator
13 **37**  -6.66571         NA         goi            Target PosCalibrator
27     88  -4.08571         NA         goi            Target PosCalibrator
```

### 7.2 Compute delta delta Cq values and fold concentration

The `DeltaDeltaCqAll` function can be implemented to compute the delta delta Cq values and log of fold concentration of all the samples. The `DeltaDeltaCqAll` function will take an object of class `RTqPCRBatch` as an input and will populate an object of class `RTqPCRBatch` as an output. The ouptput of `DeltaCq` function will act as the input for the function `DeltaDeltaCqAll`.The delta delta Cq values, standard deviation of delta delta Cq and fold concentration can be computed through following implementation:

```
> ## Compute delta delta Cq values and it's standard deviation and fold conentration
> deltadeltaCq <- DeltaDeltaCqAll(deltaCq)
> ## to visualise the overview of the resulting data
> deltadeltaCq
> ## to visualise the resulting data
> fData(deltadeltaCq)
```

## 8 Compute relative expression ratio

The function `NRQeffsAll` compute the relative expression ratio of Target by three methods of Roche, Pfaffl et al. and ddcq. The `NRQeffsAll` computes on the output of `CombineTechReps` function, where the technical replicates are combined on the basis of amplification efficiencies. Here is the example to implement the `NRQeffsAll` function:

```
> ## Compute delta delta Cq values and it's standard deviation and fold conentration
> exp.ratio <- NRQeffsAll(Effsreps, y= "hk")
> ## to visualise the overview of the resulting data
> exp.ratio
```

```
RTqPCRBatch (storageMode: lockedEnvironment)
  element names:
protocolData: none
phenoData: none
featureData
  featureNames: 1 2 ... 7 (7 total)
  fvarLabels: ID Roche Method Pfaffl Method delta delta Cq Method
  fvarMetadata: labelDescription
experimentData: use 'experimentData(object)'
Annotation:
```

```
> ## to visualise the resulting data
> fData(exp.ratio)
```

```
      ID Roche Method Pfaffl Method delta delta Cq Method
1     23     -199.269     -118.1844                 10.64
2 **32**        8.714       -3.0624                 -1.67
3 **74**       -0.713        0.4215                 13.13
4     86       -2.757       -0.9171                 13.84
5     91       -1.929       -0.0885                  1.34
6     94       -1.218       -0.0731                 -2.39
7 **95**       -0.463       -0.2580                 -1.60
```

## 9 Auxiliary functions

This section is divided into five subsections, based on the functions:

### 9.1 read.Mx3005P

The `read.Mx3005P` function reads in the .txt Tab delimited files of raw fluorescence data of Mx3005P RTqPCR and populates an object of class `RTqPCR`. This function is used in the `read.RTqPCR` function. An example to implement the `read.Mx3005P` function is as follows:

```
> path <- system.file("exData", package = "RTqPCR")
> Mx3005P.example <- file.path(path, "Mx3005P_Example.txt")
> rtData <- read.Mx3005P(Mx3005P.example)
> ##to express the overview of resulting data
> rtData
```

```
RTqPCRBatch (storageMode: lockedEnvironment)
assayData: 45 features, 34 samples
  element names: exprs
protocolData: none
phenoData
  sampleNames: 14 15 ... 78 (34 total)
  varLabels: Well Ramp/Plateaue ... mspSegment (5 total)
  varMetadata: labelDescription
featureData
  featureNames: 1 2 ... 45 (45 total)
  fvarLabels: Cycle# Temperature
  fvarMetadata: labelDescription
experimentData: use 'experimentData(object)'
Annotation:
```

```
> ## to express the phenoData of cycData
> head(pData(rtData))
```

```
   Well Ramp/Plateaue Ramp/Plateaue#   Dye mspSegment
14   14             P              2 Campy          2
15   15             P              2 Campy          2
16   16             P              2 Campy          2
17   17             P              2 Campy          2
18   18             P              2 Campy          2
20   20             P              2 Campy          2
```

```
> ## to express the fData of cycData
> head(fData(rtData))
```

```
  Cycle# Temperature
1      1        59.6
2      2        59.6
3      3        59.6
4      4        59.6
5      5        59.6
6      6        59.6
```

## 9.2 Read.LC480SampleInfo

The `Read.LC480SampleInfo` function reads in the .txt Tab delimited file of sample information data of LC480 light cycler and populate an object of class `AnnotatedDataFrame`. This function is a modified function from the `link[ReadqPCR]read.LC480SampleInfo`. One change is introduced in the previously designed function to read all columns, irrespective of the condition whether they are completely filled or empty. Reading all the columns is required to perform the computation of other functions of the work flow. Following is the example to implement the `Read.LC480SampleInfo` function:

```
> LC480.sampleInfo <- file.path(path, "LC480_example_SampleInfo.txt")
> LC480.saminfo <- Read.LC480SampleInfo(LC480.sampleInfo)
> ##To express the overview of sample information data
> LC480.saminfo
```

```
An object of class 'AnnotatedDataFrame'
  rowNames: 1 2 ... 96 (96 total)
  varLabels: Sample position Sample name ... Combined sample and target
    type (9 total)
  varMetadata: labelDescription
```

```
> ##To express the phenodata
> head(pData(LC480.saminfo))
```

```
  Sample position Sample name Replicate of Filter combination       Target name
1              A1    Sample_1           A1            465-510 negativ Kontrolle
2              A2    Sample_2           A2            465-510               goi
3              A3    Sample_2           A3            465-510                hk
4              A4    Sample_3           A1            465-510 negativ Kontrolle
5              A5    Sample_4           A2            465-510               goi
6              A6    Sample_4           A3            465-510                hk
  Sample Pref color Concentration Efficiency Combined sample and target type
1         $00FF8000            NA          2                 Target Negative
2             clRed            NA          2                  Target Unknown
3         $0030D700            NA          2                     Ref Unknown
4         clFuchsia            NA          2                 Target Negative
5            clGray            NA          2                  Target Unknown
6         $0012D7FA            NA          2                     Ref Unknown
```

## 9.3 Read.Mx3005PSampleInfo

The `Read.Mx3005PSampleInfo` function reads in the .txt Tab delimited file of sample information data of Mx3005P RTqPCR and populate an object of class `AnnotatedDataFrame`.

This function is further used in `Read.RTqPCRSampleInfo` function. It's a supplementary function to `Read.RTqPCRSampleInfo` function. Following is the example to implement the `Read.Mx3005PSampleInfo` function:

```
> Mx3005P.sampleInfo <- file.path(path, "Mx3005P_example_SampleInfo.txt")
> Mx3005P.samInfo <- Read.Mx3005PSampleInfo(Mx3005P.sampleInfo)
> ##To express the overview of the sample information data
> Mx3005P.samInfo

An object of class 'AnnotatedDataFrame'
  rowNames: 1 2 ... 34 (34 total)
  varLabels: Well Replicate of Target name Combined sample and target
    type
  varMetadata: labelDescription

> ##To express the phenodata
> head(pData(Mx3005P.samInfo))

  Well Replicate of Target name Combined sample and target type
1   14           A1         goi                  Target Unknown
2   15           A2         goi                  Target Unknown
3   16           A3          hk                     Ref Unknown
4   17           A1         goi                  Target Unknown
5   18           A2         goi                  Target Unknown
6   20           A3          hk                     Ref Unknown
```

## 9.4 DeltaDeltaCq

The `DeltaDeltaCq` function computes the delta delta Cq values and fold concentration of the sample. It is designed as a supplementary to `DeltaDeltaCqAll` function. The output of `DeltaCq` function acts as the input to this function. It acts on an object of class RTqPCR-Batch and returns the resulting data as an object of class `data.frame`. Although, it's an auxilliary function and can be directly implemented through DeltaDeltaCqAll function. Following is the example for implementing the `DeltaDeltaCq` function:

```
> x.deltadeltaCq <- DeltaDeltaCq(deltaCq)
> ## to express the resulting data
> x.deltadeltaCq

      ID deltadeltaCq Fold Conc.[log] sd.deltadeltaCq
1     23        15.35          -10.64              NA
2 **32**        -2.41            1.67              NA
```

```
3 **74**        18.95            -13.14             NA
4     86        19.97            -13.84           21.5
5     91         1.93             -1.34           21.8
6     94        -3.44              2.38             NA
7 **95**        -2.30              1.59             NA
```

### 9.5 NRQeffs

The `NRQeffs` function computes the relative expression ratio and is designed as a supplementary function to the `NRQeffsAll` function. It takes the combined technical replicates, which are produced as an output of `CombineTechReps` function, where they are combined based on amplification efficiency. The `NRQeffs` function acts on an object of class `RTqPCR-Batch` and returns the resulting data as an object of class `data.frame`.It's an auxilliary function to NRQeffsAll, which can be directly implemented through the NRQeffs function.

### 9.6 RTqPCR.dataframe

The `RTqPCR.dataframe` function is designed as a supplementary function to the graphical user interface (GUI) for package`"RTqPCR"`. This GUI is based on the shinyApps. In shinyApps, the information are passed between functions as well as resulting ones are taken out in form of data.frame. So, to handle the functions based on S4 class `RTqPCRBatch`, the `RTqPCR.dataframe` function is designed, which can convert the object of class `data.frame` into object of class`RTqPCRBatch`. It also reveals a way to handle the S4 class object in shinyApp.

## 10 Graphical User Interface (GUI)

```
> #library(shiny)
> #RTqPCR.gui()
```