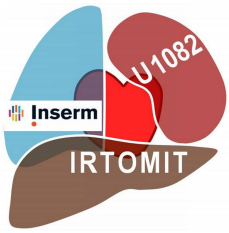
EasyqpcR: low-throughput real-time quantitative PCR data analysis

Sylvain Le Pape
IRTOMIT-INSERM U1082 (Poitiers, France)



August 26, 2015

Contents

1 Introduction 2

2 Amplification efficiency calculation 3

3 Calculation of the expression values from one or multiple qPCR run(s) 4

4 Calculation of the calibration factors 6

5 Attenuation of inter-run variations 7

6 Aggregation of multiple independent biological replicates from the same experiment 8

7 RStudio and gWidgets: tools to facilitate your qPCR data analysis 11

8 How to analyse qPCR data with EasyqpcR when samples and genes are spread across runs ? 15

1 Introduction

The package "EasyqpcR" has been created to facilitate the analysis of real-time quantitative RT-PCR data. This package contains five functions (`badCt, nrmData, calData, totData, slope`). In this manuscript, we describe and demonstrate how we can use this package. The last section presents how we can use the free R GUI `RStudio` and the `gWidgets` package created by John Verzani in order to facilitate the qPCR data analysis by a graphical user interface.

2 Amplification efficiency calculation

In this section, we describe how we can use the `slope` function of this package. As an example, we have 2 genes (gene 1 and gene 2) and 5 samples in triplicates (control group 1 and control group 2, treatment group 1 and treatment group 2, calibrator). We want to calculate the amplification efficiency of these two genes:

```
library(EasyqpcR)

data(Efficiency_calculation)

slope(data=Efficiency_calculation, q=c(1000, 100 ,10, 1, 0.1),
    r=3, na.rm=TRUE)

$Efficiency
              E
Gene.1 1.989779
Gene.2 1.869559
```

You can put the returned values into a vector to use it (without the need to type every amplification efficiency) in the next functions.

```
efficiency <- slope(data=Efficiency_calculation, q=c(1000, 100 ,10, 1, 0.1),
    r=3, na.rm=TRUE)
```

3 Calculation of the expression values from one or multiple qPCR run(s)

We describe the calculation of the normalization factors, the relative quantities, the normalized relative quantities, and the normalized relative quantities scaled to the control group of your choice using the method of Hellemans et al (2007) [2]. We have a set of three qPCR runs, each run representing an independent biological replicate. The raw data of these runs can be found in the `data` folder of this package. The no template control and no amplification control have been discarded in order to facilitate the understanding of the workflow. The limiting step is that you need to put the control samples on the top of the data frame otherwise, the algorithm will not work correctly. Firstly, we load the datasets:

```
data(qPCR_run1,qPCR_run2,qPCR_run3)

str(c(qPCR_run1,qPCR_run2,qPCR_run3))

List of 15
 $ Samples: Factor w/ 5 levels "Calibrator","Control 1",..: 2 2 2 3 3 3 4 4 4 5 ...
 $ RG1    : num [1:15] 19.6 19.3 19.5 19.6 19.4 ...
 $ RG2    : num [1:15] 18.5 18.4 18.5 18.6 18.7 ...
 $ TG     : num [1:15] 26.3 26.1 26.2 26.1 26.1 ...
 $ TGb    : num [1:15] 16.3 16.5 16.7 16.8 17 ...
 $ Samples: Factor w/ 5 levels "Calibrator","Control 1",..: 2 2 2 3 3 3 4 4 4 5 ...
 $ RG1    : num [1:15] 20.7 20.7 20.5 20.8 21 ...
 $ RG2    : num [1:15] 19.7 19.6 19.5 20.2 20.2 ...
 $ TG     : num [1:15] 27.5 27.3 27.3 27.5 27.7 ...
 $ TGb    : num [1:15] 17.5 17.7 17.7 18.2 18.6 ...
 $ Samples: Factor w/ 5 levels "Calibrator","Control 1",..: 2 2 2 3 3 3 4 4 4 5 ...
 $ RG1    : num [1:15] 20.7 20.7 20.5 20.5 20.6 ...
 $ RG2    : num [1:15] 19.7 19.6 19.5 19.8 19.8 ...
 $ TG     : num [1:15] 27.5 27.3 27.3 27.2 27.3 ...
 $ TGb    : num [1:15] 17.5 17.7 17.7 17.8 18.1 ...
```

Each dataset contains 15 observations: 5 samples (2 control groups, 2 treatment groups, 1 calibrator) in triplicates. There are 4 genes: 2 reference genes (RG1 and RG2) and 2 target genes (TG and Tgb). In order to facilitate the understanding of the `nrmData` function, I suggest you to read its man page by typing `?nrmData` in your R session.

Concerning the reference genes, I suggest you to use the `selectHKgenes` function of the `SLqPCR` package from Matthias Kohl [1].

In order to avoid the inter-run variations, we have used a calibrator (one is the minimum recommended, more is better). Thus, we have to calculate the calibration factor for each

gene. We have to include the normalized relative quantities of our calibrator in an object:

```
## Isolate the calibrator NRQ values of the first biological replicate

aa <- nrmData(data=qPCR_run1 , r=3, E=c(2, 2, 2, 2),
       Eerror=c(0.02, 0.02, 0.02, 0.02), nSpl=5,
       nbRef=2, Refposcol=1:2, nCTL=2,
       CF=c(1, 1, 1, 1), CalPos=5, trace=FALSE, geo=TRUE, na.rm=TRUE)[[3]]

## Isolate the calibrator NRQ values of the first biological replicate

bb <- nrmData(data=qPCR_run2 , r=3, E=c(2, 2, 2, 2),
           Eerror=c(0.02, 0.02, 0.02, 0.02), nSpl=5,
           nbRef=2, Refposcol=1:2, nCTL=2,
           CF=c(1, 1, 1, 1), CalPos=5, trace=FALSE, geo=TRUE, na.rm=TRUE)[[3]]

## Isolate the calibrator NRQ values of the first biological replicate

cc <- nrmData(data=qPCR_run3 , r=3, E=c(2, 2, 2, 2),
       Eerror=c(0.02, 0.02, 0.02, 0.02), nSpl=5,
       nbRef=2, Refposcol=1:2, nCTL=2,
       CF=c(1, 1, 1, 1), CalPos=5, trace=FALSE, geo=TRUE, na.rm=TRUE)[[3]]
```

Now, we have to run the `calData` function.

4 Calculation of the calibration factors

Here, we describe how to use the `calData` function. In the continuation of what has been done before, we have three objects containing the NRQ values of the calibrator(s) and we now have to calculate the calibration factors for each gene:

```
## Calibration factor calculation

e <- calData(aa)

f <- calData(bb)

g <- calData(cc)
```

5 Attenuation of inter-run variations

Now, we have the calibration factors, we can calculate the expression value without the obsession of the inter-run variability:

```
nrmData(data=qPCR_run1 , r=3, E=c(2, 2, 2, 2),
       Eerror=c(0.02, 0.02, 0.02, 0.02), nSpl=5,
       nbRef=2, Refposcol=1:2, nCTL=2,
       CF=e, CalPos=5, trace=FALSE, geo=TRUE, na.rm=TRUE)

nrmData(data=qPCR_run2 , r=3, E=c(2, 2, 2, 2),
       Eerror=c(0.02, 0.02, 0.02, 0.02), nSpl=5,
       nbRef=2, Refposcol=1:2, nCTL=2,
       CF=f, CalPos=5, trace=FALSE, geo=TRUE, na.rm=TRUE)

nrmData(data=qPCR_run3 , r=3, E=c(2, 2, 2, 2),
       Eerror=c(0.02, 0.02, 0.02, 0.02), nSpl=5,
       nbRef=2, Refposcol=1:2, nCTL=2,
       CF=g, CalPos=5, trace=FALSE, geo=TRUE, na.rm=TRUE)
```

Remark: The validity of IRCs must be interpreted with care: two or more IRCs must be used to control if the IRCs measure the technical variation between the runs with the same extent the `calData` value divided by each calibrator NRQ value must be sensitively equal). If this ratio is really different, you must exclude the highly variable IRC in **all** the qPCR runs.

6 Aggregation of multiple independent biological replicates from the same experiment

In this section, we will discuss about the final function of this package `totData`. In some research fields, the reproducibility of an observation can be tough (notably in the stem cells field). An algorithm published by Willems et al. (2008) [3] attenuates the high variations between independent biological replicates which have the same tendency in order to draw relevant statistical conclusions. This algorithm has been inputed in this function for the scientists experiencing this kind of issue.

```
## Isolate the NRQs scaled to control of the first biological replicate

a1 <- nrmData(data=qPCR_run1 , r=3, E=c(2, 2, 2, 2),
       Eerror=c(0.02, 0.02, 0.02, 0.02), nSpl=5,
       nbRef=2, Refposcol=1:2, nCTL=2,
       CF=e, CalPos=5, trace=FALSE, geo=TRUE, na.rm=TRUE)[1]

## Isolate the NRQs scaled to control of the second biological replicate

b1 <- nrmData(data=qPCR_run2 , r=3, E=c(2, 2, 2, 2),
       Eerror=c(0.02, 0.02, 0.02, 0.02), nSpl=5,
       nbRef=2, Refposcol=1:2, nCTL=2,
       CF=f, CalPos=5, trace=FALSE, geo=TRUE, na.rm=TRUE)[1]

## Isolate the NRQs scaled to control of the third biological replicate

c1 <- nrmData(data=qPCR_run3 , r=3, E=c(2, 2, 2, 2),
       Eerror=c(0.02, 0.02, 0.02, 0.02), nSpl=5,
       nbRef=2, Refposcol=1:2, nCTL=2,
       CF=g, CalPos=5, trace=FALSE, geo=TRUE, na.rm=TRUE)[1]

## Data frame transformation

a2 <- as.data.frame(a1)
b2 <- as.data.frame(b1)
c2 <- as.data.frame(c1)

## Aggregation of the three biological replicates

d2 <- rbind(a2, b2, c2)
```

Finally, we use the final function `totData` and indicate that we want to use the transformation algorithm published by Willems et al. (2008) [3] followed by the linearization process:

```
totData(data = d2, r = 3, base=2, nSpl = 15, nCTL = 2, na.rm = TRUE)

$`Linearized Standardized data according to Willems procedure (2008)`
                  RG1       RG2         TG       TGb
Control 1    0.9433579 1.0600430  0.1290449 1.1462944
Control 2    1.0059474 0.9940878  0.1438302 0.9264764
Treatment 1  1.0700425 0.9345424  7.6004540 0.6442379
Treatment 2  1.0413523 0.9602898 55.3140468 1.2780037
Calibrator   0.9456916 1.0574272  0.1281545 1.1436448
Control 11   0.9618646 1.0396474  0.1310156 1.1604854
Control 21   1.0754922 0.9298068  0.1401095 0.9077403
Treatment 11 1.0417225 0.9599486  7.2437660 0.6465487
Treatment 21 0.9900799 1.0100195 57.1363145 1.2314170
Calibrator1  0.9372524 1.0669485  0.1316233 1.1923187
Control 12   0.9415964 1.0620261  0.1285689 1.1797731
Control 22   1.0480369 0.9541648  0.1419923 0.9010735
Treatment 12 1.0560850 0.9468935  7.9156554 0.6483932
Treatment 22 1.0190487 0.9813074 53.8240524 1.2297147
Calibrator2  0.9415964 1.0620261  0.1285689 1.1797731

$Mean
                  RG1       RG2         TG       TGb
Control 1   0.9488956 1.0538568  0.1295388 1.1621034
Control 2   1.0427648 0.9589890  0.1419692 0.9117002
Treatment 1 1.0558867 0.9470714  7.5816478 0.6463910
Treatment 2 1.0166098 0.9836616 55.4082818 1.2461793
Calibrator  0.9415072 1.0621268  0.1294397 1.1717294

$`Lower Confidence Interval 95%`
                  RG1       RG2         TG       TGb
Control 1   0.9214998 1.0234306  0.1263639 1.1211444
Control 2   0.9590033 0.8819568  0.1374217 0.8796572
Treatment 1 1.0212860 0.9160365  6.7898488 0.6412387
Treatment 2 0.9546123 0.9236734 51.4424166 1.1803433
Calibrator  0.9310810 1.0503649  0.1248363 1.1103335

$`Upper Confidence Interval 95%`
```

```
                  RG1       RG2         TG       TGb
Control 1   0.9771058 1.0851875  0.1327935 1.2045587
Control 2   1.1338423 1.0427493  0.1466672 0.9449104
Treatment 1 1.0916595 0.9791576  8.4657826 0.6515847
Treatment 2 1.0826337 1.0475457 59.6798886 1.3156874
Calibrator  0.9520501 1.0740204  0.1342129 1.2365202
```

7 RStudio and gWidgets: tools to facilitate your qPCR data analysis

To facilitate the use of R, a free software named *RStudio* has been created [4]. This interface allows (among other things) easy data importation/exportation. In the same spirit of having an interface for using R, John Verzani has published a package `gWidgets` which has the great advantage to easily create a graphical user interface [5] for the function you want. In this last section, we will present how we can use these tools to facilitate the qPCR data analysis.

To begin, we must choose our workspace directory by typing this in your R session: **setwd(gfile(type='selectdir '))**. You will see the opening of a window and you will just need to define your workspace directory. Then, we have to import some datasets in our R session. *RStudio* allows an easy data importation (see 1).

This can be done by following these steps:

1. uncompress the csv file (qPCR_run1.csv) in the data folder
2. move it to inst/extdata

Then, you just need to type this in your R session:

```
file <- system.file("extdata", "qPCR_run1.csv", package="EasyqpcR")

qPCR_run1 <- read.table(file, header=TRUE, sep="", dec=".")

qPCR_run1
```

```
       Samples   RG1   RG2    TG   TGb
1    Control 1 19.62 18.54 26.32 16.32
2    Control 1 19.32 18.35 26.12 16.48
3    Control 1 19.48 18.49 26.24 16.68
4    Control 2 19.63 18.63 26.08 16.75
5    Control 2 19.40 18.72 26.12 16.98
6    Control 2 19.35 18.51 26.23 16.89
7  Treatment 1 19.21 18.45 20.32 17.12
8  Treatment 1 19.32 18.49 20.15 17.23
9  Treatment 1 19.09 18.66 20.26 17.33
10 Treatment 2 19.63 18.75 17.65 16.65
11 Treatment 2 19.46 18.69 17.75 16.54
12 Treatment 2 19.51 18.92 17.60 16.40
13  Calibrator 19.88 18.91 26.69 16.69
14  Calibrator 19.78 18.72 26.49 16.85
15  Calibrator 19.85 18.86 26.63 17.05
```

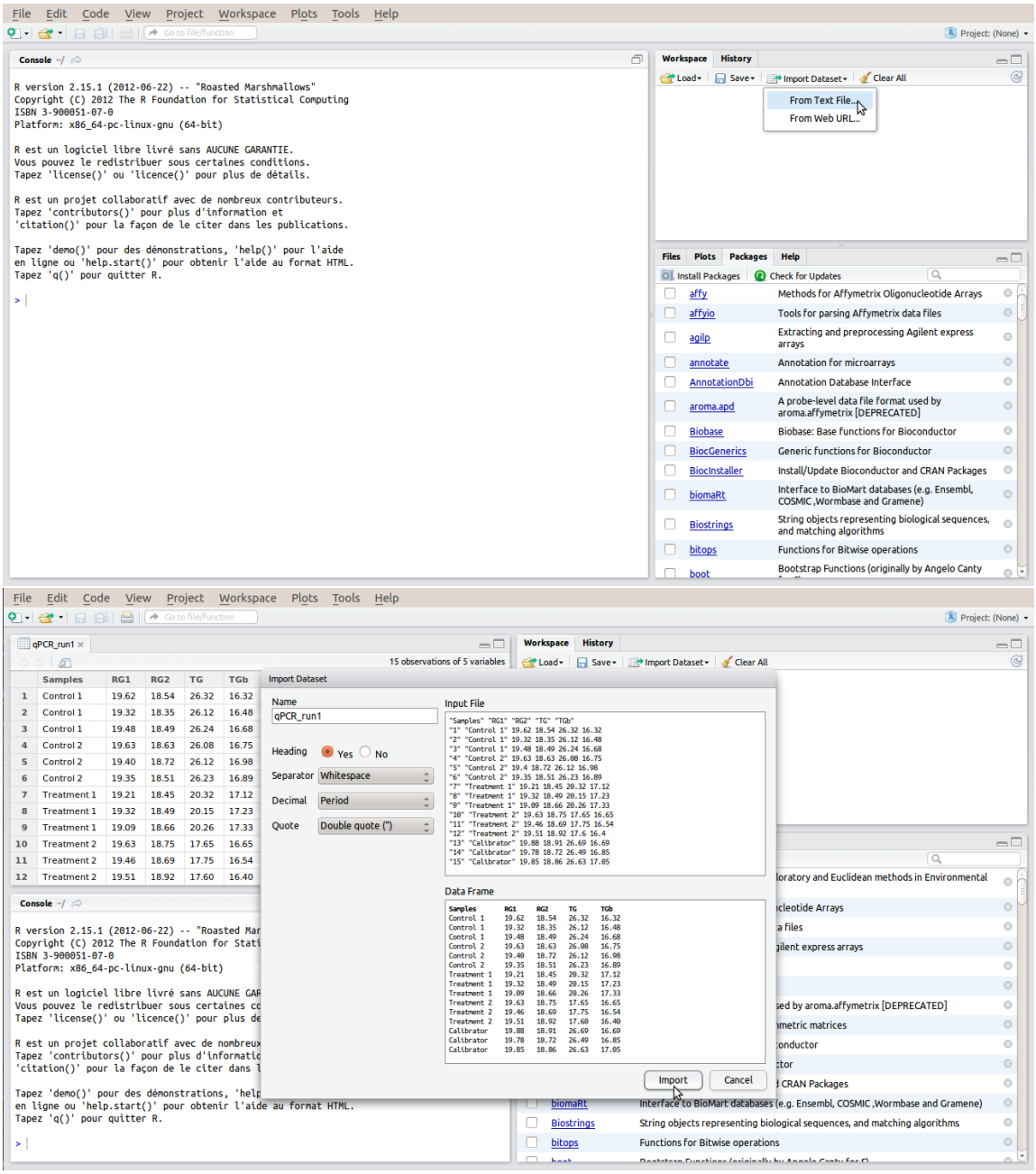


Figure 1: *Data importation in RStudio*

After data importation, you must control if your qPCR technical replicates satisfy your

threshold variation value (0.5 classically):

```
badCt(data=qPCR_run1, r=3, threshold=0.5, na.rm=TRUE)

$`Bad replicates localization`
     row col

$`Mean of the Cq`
                 RG1      RG2       TG      TGb
Control 1   19.47333 18.46000 26.22667 16.49333
Control 2   19.46000 18.62000 26.14333 16.87333
Treatment 1 19.20667 18.53333 20.24333 17.22667
Treatment 2 19.53333 18.78667 17.66667 16.53000
Calibrator  19.83667 18.83000 26.60333 16.86333

$`Standard error of the Cq`
                  RG1        RG2         TG        TGb
Control 1   0.08666667 0.05686241 0.05811865 0.10413666
Control 2   0.08621678 0.06082763 0.04484541 0.06691620
Treatment 1 0.06641620 0.06437736 0.04977728 0.06064468
Treatment 2 0.05044249 0.06887993 0.04409586 0.07234178
Calibrator  0.02962731 0.05686241 0.05925463 0.10413666
```

Here, there is no bad replicates, but as an example, we will set the threshold value to 0.2 and see what it returns:

```
badCt(data=qPCR_run1, r=3, threshold=0.2, na.rm=TRUE)

$`Bad replicates localization`
            row col
Control 1     1   2
Control 2     2   2
Treatment 1   3   2
Control 2     2   3
Treatment 1   3   3
Treatment 2   4   3
Calibrator    5   4
Control 1     1   5
Control 2     2   5
Treatment 1   3   5
Treatment 2   4   5
```

```
Calibrator    5   5

$`Mean of the Cq`
                RG1      RG2       TG      TGb
Control 1   19.47333 18.46000 26.22667 16.49333
Control 2   19.46000 18.62000 26.14333 16.87333
Treatment 1 19.20667 18.53333 20.24333 17.22667
Treatment 2 19.53333 18.78667 17.66667 16.53000
Calibrator  19.83667 18.83000 26.60333 16.86333

$`Standard error of the Cq`
                  RG1        RG2         TG        TGb
Control 1   0.08666667 0.05686241 0.05811865 0.10413666
Control 2   0.08621678 0.06082763 0.04484541 0.06691620
Treatment 1 0.06641620 0.06437736 0.04977728 0.06064468
Treatment 2 0.05044249 0.06887993 0.04409586 0.07234178
Calibrator  0.02962731 0.05686241 0.05925463 0.10413666
```

There are some bad replicates (according to the example threshold value). Now, we want to easily remove technical error (no more than one in qPCR technical triplicates), we just need to use the `gdfnotebook` function of the `gWidgets` package by typing this in the R session: **gdfnotebook(cont=TRUE)**, and then choose which dataset you want to edit (qPCR_run1, here). After saving it (on an other name in order to easily reproduce your analysis with the same raw data, by example: qPCR_run1 is the raw dataset, and after removing technical replicates you can save it under the name qPCR_run1_cor). Or, you can edit your dataset directly in your spreadsheet and save it under an other name.

Finally, to easily analyze your qPCR data, you just will need to type this in your R session for each function of the package: **ggenericwidget(`function`,cont=TRUE)**, where `function` has to be replaced by `nrmData`, `calData`, `totData`, or `badCt`. This can also be done with the command lines described above.

8 How to analyse qPCR data with EasyqpcR when samples and genes are spread across runs ?

All the previous examples showed how to perform qPCR data analysis when all the samples were present for each gene in each run. Here we present the procedure to follow when we have too much samples to be contained in each run, thus when samples and genes are spread across different runs.

Here are some examples of plate designs:

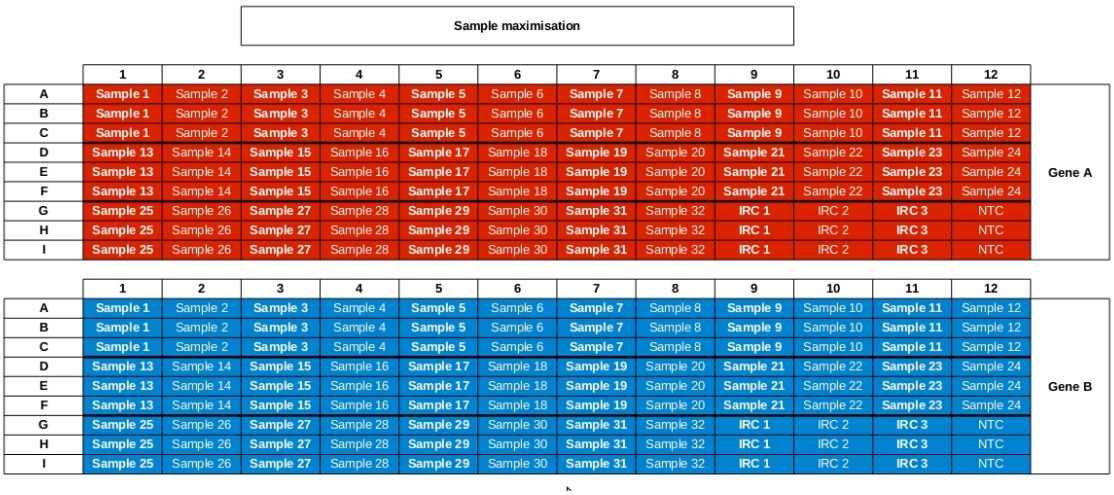


Figure 2: *Sample maximisation strategy*

Sample maximisation

Run 1

	1	2	3	4	5	6	7	8	9	10	11	12
A	**Sample 1**	Sample 1	**Sample 1**	Sample 2	**Sample 2**	Sample 2	**Sample 3**	Sample 3	**Sample 3**	Sample 4	**Sample 4**	Sample 4
B	**IRC 1.1**	IRC 1.1	**IRC 1.1**	IRC 2.1	**IRC 2.1**	IRC 2.1	**IRC 3.1**	IRC 3.1	**IRC 3.1**			
C	**Sample 1**	Sample 1	**Sample 1**	Sample 2	**Sample 2**	Sample 2	**Sample 3**	Sample 3	**Sample 3**	Sample 4	**Sample 4**	Sample 4
D	**IRC 1.1**	IRC 1.1	**IRC 1.1**	IRC 2.1	**IRC 2.1**	IRC 2.1	**IRC 3.1**	IRC 3.1	**IRC 3.1**			
E	**Sample 1**	Sample 1	**Sample 1**	Sample 2	**Sample 2**	Sample 2	**Sample 3**	Sample 3	**Sample 3**	Sample 4	**Sample 4**	Sample 4
F	**IRC 1.1**	IRC 1.1	**IRC 1.1**	IRC 2.1	**IRC 2.1**	IRC 2.1	**IRC 3.1**	IRC 3.1	**IRC 3.1**			
G	**NTC 1.1**	NTC 1.1	**NTC 1.1**	NTC 2.1	**NTC 2.1**	NTC 2.1	**NTC 3.1**	NTC 3.1	**NTC 3.1**			
H												

Run 2

	1	2	3	4	5	6	7	8	9	10	11	12
A	**Sample 5**	Sample 5	**Sample 5**	Sample 6	**Sample 6**	Sample 6	**Sample 7**	Sample 7	**Sample 7**	Sample 8	**Sample 8**	Sample 8
B	**IRC 1.2**	IRC 1.2	**IRC 1.2**	IRC 2.2	**IRC 2.2**	IRC 2.2	**IRC 3.2**	IRC 3.2	**IRC 3.2**			
C	**Sample 5**	Sample 5	**Sample 5**	Sample 6	**Sample 6**	Sample 6	**Sample 7**	Sample 7	**Sample 7**	Sample 8	**Sample 8**	Sample 8
D	**IRC 1.2**	IRC 1.2	**IRC 1.2**	IRC 2.2	**IRC 2.2**	IRC 2.2	**IRC 3.2**	IRC 3.2	**IRC 3.2**			
E	**Sample 5**	Sample 5	**Sample 5**	Sample 6	**Sample 6**	Sample 6	**Sample 7**	Sample 7	**Sample 7**	Sample 8	**Sample 8**	Sample 8
F	**IRC 1.2**	IRC 1.2	**IRC 1.2**	IRC 2.2	**IRC 2.2**	IRC 2.2	**IRC 3.2**	IRC 3.2	**IRC 3.2**			
G	**NTC 1.2**	NTC 1.2	**NTC 1.2**	NTC 2.2	**NTC 2.2**	NTC 2.2	**NTC 3.2**	NTC 3.2	**NTC 3.2**			
H												

Run 3

	1	2	3	4	5	6	7	8	9	10	11	12
A	**Sample 9**	Sample 9	**Sample 9**	Sample 10	**Sample 10**	Sample 10	**Sample 11**	Sample 11	**Sample 11**	Sample 12	**Sample 12**	Sample 12
B	**IRC 1.3**	IRC 1.3	**IRC 1.3**	IRC 2.3	**IRC 2.3**	IRC 2.3	**IRC 3.3**	IRC 3.3	**IRC 3.3**			
C	**Sample 9**	Sample 9	**Sample 9**	Sample 10	**Sample 10**	Sample 10	**Sample 11**	Sample 11	**Sample 11**	Sample 12	**Sample 12**	Sample 12
D	**IRC 1.3**	IRC 1.3	**IRC 1.3**	IRC 2.3	**IRC 2.3**	IRC 2.3	**IRC 3.3**	IRC 3.3	**IRC 3.3**			
E	**Sample 9**	Sample 9	**Sample 9**	Sample 10	**Sample 10**	Sample 10	**Sample 11**	Sample 11	**Sample 11**	Sample 12	**Sample 12**	Sample 12
F	**IRC 1.3**	IRC 1.3	**IRC 1.3**	IRC 2.3	**IRC 2.3**	IRC 2.3	**IRC 3.3**	IRC 3.3	**IRC 3.3**			
G	**NTC 1.3**	NTC 1.3	**NTC 1.3**	NTC 2.3	**NTC 2.3**	NTC 2.3	**NTC 3.3**	NTC 3.3	**NTC 3.3**			
H												

Figure 3: *Gene maximisation strategy 1/3*

	1	2	3	4	5	6	7	8	9	10	11	12	
A	Sample 13	Sample 13	Sample 13	Sample 14	Sample 14	Sample 14	Sample 15	Sample 15	Sample 15	Sample 16	Sample 16	Sample 16	Run 4
B	IRC 1.4	IRC 1.4	IRC 1.4	IRC 2.4	IRC 2.4	IRC 2.4	IRC 3.4	IRC 3.4	IRC 3.4				
C	Sample 13	Sample 13	Sample 13	Sample 14	Sample 14	Sample 14	Sample 15	Sample 15	Sample 15	Sample 16	Sample 16	Sample 16	
D	IRC 1.4	IRC 1.4	IRC 1.4	IRC 2.4	IRC 2.4	IRC 2.4	IRC 3.4	IRC 3.4	IRC 3.4				
E	Sample 13	Sample 13	Sample 13	Sample 14	Sample 14	Sample 14	Sample 15	Sample 15	Sample 15	Sample 16	Sample 16	Sample 16	
F	IRC 1.4	IRC 1.4	IRC 1.4	IRC 2.4	IRC 2.4	IRC 2.4	IRC 3.4	IRC 3.4	IRC 3.4				
G	NTC 1.4	NTC 1.4	NTC 1.4	NTC 2.4	NTC 2.4	NTC 2.4	NTC 3.4	NTC 3.4	NTC 3.4				
H													

	1	2	3	4	5	6	7	8	9	10	11	12	
A	Sample 17	Sample 17	Sample 17	Sample 18	Sample 18	Samplle 18	Sample 19	Sample 19	Sample 19	Sample 20	Sample 20	Sample 20	Run 5
B	IRC 1.5	IRC 1.5	IRC 1.5	IRC 2.5	IRC 2.5	IRC 2.5	IRC 3.5	IRC 3.5	IRC 3.5				
C	Sample 17	Sample 17	Sample 17	Sample 18	Sample 18	Samplle 18	Sample 19	Sample 19	Sample 19	Sample 20	Sample 20	Sample 20	
D	IRC 1.5	IRC 1.5	IRC 1.5	IRC 2.5	IRC 2.5	IRC 2.5	IRC 3.5	IRC 3.5	IRC 3.5				
E	Sample 17	Sample 17	Sample 17	Sample 18	Sample 18	Samplle 18	Sample 19	Sample 19	Sample 19	Sample 20	Sample 20	Sample 20	
F	IRC 1.5	IRC 1.5	IRC 1.5	IRC 2.5	IRC 2.5	IRC 2.5	IRC 3.5	IRC 3.5	IRC 3.5				
G	NTC 1.5	NTC 1.5	NTC 1.5	NTC 2.5	NTC 2.5	NTC 2.5	NTC 3.5	NTC 3.5	NTC 3.5				
H													

	1	2	3	4	5	6	7	8	9	10	11	12	
A	Sample 21	Sample 21	Sample 21	Sample 22	Sample 22	Sample 22	Sample 23	Sample 23	Sample 23	Sample 24	Sample 24	Sample 24	Run 6
B	IRC 1.6	IRC 1.6	IRC 1.6	IRC 2.6	IRC 2.6	IRC 2.6	IRC 3.6	IRC 3.6	IRC 3.6				
C	Sample 21	Sample 21	Sample 21	Sample 22	Sample 22	Sample 22	Sample 23	Sample 23	Sample 23	Sample 24	Sample 24	Sample 24	
D	IRC 1.6	IRC 1.6	IRC 1.6	IRC 2.6	IRC 2.6	IRC 2.6	IRC 3.6	IRC 3.6	IRC 3.6				
E	Sample 21	Sample 21	Sample 21	Sample 22	Sample 22	Sample 22	Sample 23	Sample 23	Sample 23	Sample 24	Sample 24	Sample 24	
F	IRC 1.6	IRC 1.6	IRC 1.6	IRC 2.6	IRC 2.6	IRC 2.6	IRC 3.6	IRC 3.6	IRC 3.6				
G	NTC 1.6	NTC 1.6	NTC 1.6	NTC 2.6	NTC 2.6	NTC 2.6	NTC 3.6	NTC 3.6	NTC 3.6				
H													

Figure 4: *Gene maximisation strategy 2/3*

	1	2	3	4	5	6	7	8	9	10	11	12	
A	Sample 25	Sample 25	Sample 25	Sample 26	Sample 26	Sample 26	Sample 27	Sample 27	Sample 27	Sample 28	Sample 28	Sample 28	Run 7
B	IRC 1.7	IRC 1.7	IRC 1.7	IRC 2.7	IRC 2.7	IRC 2.7	IRC 3.7	IRC 3.7	IRC 3.7				
C	Sample 25	Sample 25	Sample 25	Sample 26	Sample 26	Sample 26	Sample 27	Sample 27	Sample 27	Sample 28	Sample 28	Sample 28	
D	IRC 1.7	IRC 1.7	IRC 1.7	IRC 2.7	IRC 2.7	IRC 2.7	IRC 3.7	IRC 3.7	IRC 3.7				
E	Sample 25	Sample 25	Sample 25	Sample 26	Sample 26	Sample 26	Sample 27	Sample 27	Sample 27	Sample 28	Sample 28	Sample 28	
F	IRC 1.7	IRC 1.7	IRC 1.7	IRC 2.7	IRC 2.7	IRC 2.7	IRC 3.7	IRC 3.7	IRC 3.7				
G	NTC 1.7	NTC 1.7	NTC 1.7	NTC 2.7	NTC 2.7	NTC 2.7	NTC 3.7	NTC 3.7	NTC 3.7				
H													

	1	2	3	4	5	6	7	8	9	10	11	12	
A	Sample 29	Sample 29	Sample 29	Sample 30	Sample 30	Sample 30	Sample 31	Sample 31	Sample 31	Sample 32	Sample 32	Sample 32	Run 8
B	IRC 1.8	IRC 1.8	IRC 1.8	IRC 2.8	IRC 2.8	IRC 2.8	IRC 3.8	IRC 3.8	IRC 3.8				
C	Sample 29	Sample 29	Sample 29	Sample 30	Sample 30	Sample 30	Sample 31	Sample 31	Sample 31	Sample 32	Sample 32	Sample 32	
D	IRC 1.8	IRC 1.8	IRC 1.8	IRC 2.8	IRC 2.8	IRC 2.8	IRC 3.8	IRC 3.8	IRC 3.8				
E	Sample 29	Sample 29	Sample 29	Sample 30	Sample 30	Sample 30	Sample 31	Sample 31	Sample 31	Sample 32	Sample 32	Sample 32	
F	IRC 1.8	IRC 1.8	IRC 1.8	IRC 2.8	IRC 2.8	IRC 2.8	IRC 3.8	IRC 3.8	IRC 3.8				
G	NTC 1.8	NTC 1.8	NTC 1.8	NTC 2.8	NTC 2.8	NTC 2.8	NTC 3.8	NTC 3.8	NTC 3.8				
H													

Figure 5: *Gene maximisation strategy 3/3*

As described in the work of Hellemans et al (2007) [2], the sample maximisation strategy does not need inter-run calibration factor because there is no samples spread over runs (all the samples for each gene are contained in one plate). Unless you have to many samples to be contained in one run even with sample maximisation strategy, you will have to perform the inter-run variation correction presented below.

In gene maximisation strategy the samples and genes are spread across runs. Thus, it is necessary to calibrate the NRQs with a run-specific and gene-specific calibration factor.

We will now describe how to handle this issue. Follow these steps:

1. uncompress the csv file (Gene_maximisation.csv) in the data folder
2. move it to inst/extdata

Then, you just have to do this:

```
filebis <- system.file("extdata", "Gene_maximisation.csv", package="EasyqpcR")

Gene_maximisation <- read.table(filebis, header=TRUE, sep=";", dec=",")
```

First, you have to analyze if there are some bad replicates:

```
badCt(data=Gene_maximisation, r=3, threshold=0.5, na.rm=FALSE)[1]

$`Bad replicates localization`
          row col
Sample 15  15   2
```

```
NTC 1       36   2
NTC 2       40   2
NTC 3       44   2
NTC 5       52   2
NTC 7       60   2
NTC 8       64   2
Sample 18   18   3
NTC 1       36   3
NTC 2       40   3
NTC 3       44   3
NTC 4       48   3
NTC 7       60   3
NTC 8       64   3
NTC 1       36   4
NTC 2       40   4
NTC 3       44   4
NTC 4       48   4
NTC 7       60   4
NTC 8       64   4
```

Do not worry about the NTC (no template control), they are not needed in qPCR data analysis (but they have to present at least a Ct difference > 5 Ct compared to the samples). Here, you can use the `gdfnotebook` function of the `gWidgets` package.

After having removed the two aberrant values (RG2, Sample 15, Ct = 22.0691567571 ; and RG3, Sample 18, Ct = 19.0232804823), rename the data frame as Gene_maximisation_cor.

Now, we remove the NTC values:

```
fileter <- system.file("extdata", "Gene_maximisation_cor.csv",
    package="EasyqpcR")

Gene_maximisation_cor <- read.table(fileter, header=TRUE, sep=";", dec=",")

Gene_maximisation_cor1 <- Gene_maximisation_cor[-c(106:108, 118:120, 130:132,
 142:144, 154:156, 166:168, 178:180, 190:192),]

rownames(Gene_maximisation_cor1) <- c(1:168)
```

Now, we will calculate run-specific calibration factors:

```
calr1 <- nrmData(data = Gene_maximisation_cor1, r=3, E=c(2, 2, 2),
    Eerror=c(0.02, 0.02, 0.02), nSpl=56, nbRef=2, Refposcol=1:2,
    nCTL=16, CF=c(1, 1, 1), CalPos=c(33:56), trace = FALSE, geo = TRUE,
    na.rm = TRUE)[[3]][1:3,]

calr2 <- nrmData(data = Gene_maximisation_cor1, r=3, E=c(2, 2, 2),
    Eerror=c(0.02, 0.02, 0.02), nSpl=56, nbRef=2, Refposcol=1:2, nCTL=16,
    CF=c(1, 1, 1), CalPos=c(33:56), trace = FALSE, geo = TRUE,
    na.rm = TRUE)[[3]][4:6,]

calr3 <- nrmData(data = Gene_maximisation_cor1, r=3, E=c(2, 2, 2),
    Eerror=c(0.02, 0.02, 0.02), nSpl=56, nbRef=2, Refposcol=1:2, nCTL=16,
    CF=c(1, 1, 1), CalPos=c(33:56), trace = FALSE, geo = TRUE,
    na.rm = TRUE)[[3]][7:9,]

calr4 <- nrmData(data = Gene_maximisation_cor1, r=3, E=c(2, 2, 2),
    Eerror=c(0.02, 0.02, 0.02), nSpl=56, nbRef=2, Refposcol=1:2,
    nCTL=16, CF=c(1, 1, 1), CalPos=c(33:56), trace = FALSE, geo = TRUE,
    na.rm = TRUE)[[3]][10:12,]

calr5 <- nrmData(data = Gene_maximisation_cor1, r=3, E=c(2, 2, 2),
    Eerror=c(0.02, 0.02, 0.02), nSpl=56, nbRef=2, Refposcol=1:2, nCTL=16,
    CF=c(1, 1, 1), CalPos=c(33:56), trace = FALSE, geo = TRUE,
    na.rm = TRUE)[[3]][13:15,]

calr6 <- nrmData(data = Gene_maximisation_cor1, r=3, E=c(2, 2, 2),
    Eerror=c(0.02, 0.02, 0.02), nSpl=56, nbRef=2, Refposcol=1:2, nCTL=16,
    CF=c(1, 1, 1), CalPos=c(33:56), trace = FALSE, geo = TRUE,
    na.rm = TRUE)[[3]][16:18,]

calr7 <- nrmData(data = Gene_maximisation_cor1, r=3, E=c(2, 2, 2),
    Eerror=c(0.02, 0.02, 0.02), nSpl=56, nbRef=2, Refposcol=1:2,
    nCTL=16, CF=c(1, 1, 1), CalPos=c(33:56), trace = FALSE, geo = TRUE,
    na.rm = TRUE)[[3]][19:21,]

calr8 <- nrmData(data = Gene_maximisation_cor1, r=3, E=c(2, 2, 2),
    Eerror=c(0.02, 0.02, 0.02), nSpl=56, nbRef=2, Refposcol=1:2, nCTL=16,
    CF=c(1, 1, 1), CalPos=c(33:56), trace = FALSE, geo = TRUE,
    na.rm = TRUE)[[3]][22:24,]
```

```
e <- calData(calr1)

f <- calData(calr2)

g <- calData(calr3)

h <- calData(calr4)

i <- calData(calr5)

j <- calData(calr6)

k <- calData(calr7)

l <- calData(calr8)
```

To respect the calculation of the NRQs which need the whole samples to take into account the whole variability, we have to apply the `nrmData` function on the whole samples. But we will do it for each run with the correction by the run-specific calibration factor and after each inter-run variation correction we isolate the corresponding CNRQs (*i.e.* the NRQs corrected by the specific CF), for example:

We perform inter-run variation correction on the whole samples by the CF of the first run, which corresponds to the samples 1 to 4 and IRC 1.1, 2.1, 3.1. But, the CF of the first is not the correct CF for the second run and for any other run. Thus, we isolate (after inter-run variation correction on the whole samples by the CF of the first run) the samples concerned by this specific CF which are the samples 1 to 4 and IRC 1.1, 2.1, 3.1. And we do it for each run-specific CF. Then we isolate the NRQs of the samples 5 to 8 and IRC 1.2, 2.2, 3.2 corrected by the CF of the second run, etc...

```
m <- nrmData(data = Gene_maximisation_cor1, r=3, E=c(2, 2, 2),
    Eerror=c(0.02, 0.02, 0.02), nSpl=56, nbRef=2, Refposcol=1:2, nCTL=16,
    CF=e, CalPos=c(33:35), trace = FALSE, geo = TRUE,
    na.rm = TRUE)[[2]][c(1:4,33:35),]


n <- nrmData(data = Gene_maximisation_cor1, r=3, E=c(2, 2, 2),
    Eerror=c(0.02, 0.02, 0.02), nSpl=56, nbRef=2, Refposcol=1:2, nCTL=16,
    CF=f, CalPos=c(36:56), trace = FALSE, geo = TRUE,
    na.rm = TRUE)[[2]][c(5:8,36:38),]
```

```
o <- nrmData(data = Gene_maximisation_cor1, r=3, E=c(2, 2, 2),
    Eerror=c(0.02, 0.02, 0.02), nSpl=56, nbRef=2, Refposcol=1:2, nCTL=16,
    CF=g, CalPos=c(36:56), trace = FALSE, geo = TRUE,
    na.rm = TRUE)[[2]][c(9:12,39:41),]

p <- nrmData(data = Gene_maximisation_cor1, r=3, E=c(2, 2, 2),
    Eerror=c(0.02, 0.02, 0.02), nSpl=56, nbRef=2, Refposcol=1:2, nCTL=16,
    CF=h, CalPos=c(33:35), trace = FALSE, geo = TRUE,
    na.rm = TRUE)[[2]][c(13:16,42:44),]


q <- nrmData(data = Gene_maximisation_cor1, r=3, E=c(2, 2, 2),
    Eerror=c(0.02, 0.02, 0.02), nSpl=56, nbRef=2, Refposcol=1:2, nCTL=16,
    CF=i, CalPos=c(36:56), trace = FALSE, geo = TRUE,
    na.rm = TRUE)[[2]][c(17:20,45:47),]

r <- nrmData(data = Gene_maximisation_cor1, r=3, E=c(2, 2, 2),
    Eerror=c(0.02, 0.02, 0.02), nSpl=56, nbRef=2, Refposcol=1:2, nCTL=16,
    CF=j, CalPos=c(36:56), trace = FALSE, geo = TRUE,
    na.rm = TRUE)[[2]][c(21:24,48:50),]

s <- nrmData(data = Gene_maximisation_cor1, r=3, E=c(2, 2, 2),
    Eerror=c(0.02, 0.02, 0.02), nSpl=56, nbRef=2, Refposcol=1:2, nCTL=16,
    CF=k, CalPos=c(33:35), trace = FALSE, geo = TRUE,
    na.rm = TRUE)[[2]][c(25:28,51:53),]


t <- nrmData(data = Gene_maximisation_cor1, r=3, E=c(2, 2, 2),
    Eerror=c(0.02, 0.02, 0.02), nSpl=56, nbRef=2, Refposcol=1:2, nCTL=16,
    CF=l, CalPos=c(36:56), trace = FALSE, geo = TRUE,
    na.rm = TRUE)[[2]][c(29:32,54:56),]

## Aggregation of all the CNRQs

u <- rbind(m, n, o, p, q, r, s, t)
```

Explanation of what we have done before: we have isolated the corresponding NRQs corrected by the specific CF (remember that after correction by the specific CF , we talk in terms of CNRQs and not anymore of NRQs):

- Samples 1 to 4 and IRC 1.1, 2.1, 3.1 are corrected by e

- Samples 5 to 8 and IRC 1.2, 2.2, 3.2 are corrected by f
- Samples 9 to 12 and IRC 1.3, 2.3, 3.3 are corrected by g
- Samples 13 to 16 and IRC 1.4, 2.4, 3.4 are corrected by h
- Samples 17 to 20 and IRC 1.5, 2.5, 3.5 are corrected by i
- Samples 21 to 24 and IRC 1.6, 2.6, 3.6 are corrected by j
- Samples 25 to 28 and IRC 1.7, 2.7, 3.7 are corrected by k
- Samples 29 to 32 and IRC 1.8, 2.8, 3.8 are corrected by l

Do not worry about the `nCTL` parameter, because in gene maximisation (or sample maximisation if there are too many samples), the CTL samples are not present in all the runs, so you will have to perform the scaling to control after.

Note that in this case where the control samples are not present in all the runs, the nCTL parameter is not relevant and we only take into account the NRQs corrected by the calibration factor (CNRQs). Thus, to have nicer graphs, you will need to perform a scaling to your control group by doing a geometric mean of the CNRQs of your control samples and divide all the CNRQs by this geometric mean. Here the control group is composed by the samples 1 to 16, thus:

```
ctlgroup <- u[c(1:4,8:11,15:18,22:25),]

ctlgeom <- colProds(ctlgroup)^(1/dim(ctlgroup)[1])
ctlgeom1 <- (as.data.frame(ctlgeom)[rep(1:(ncol(u)), each = nrow(u)), ])
ctlgeom2 <- as.data.frame(matrix(ctlgeom1, ncol = ncol(u), byrow = FALSE))

CNRQs_scaled_to_group <- u/ctlgeom2
```

References

[1] Kohl, M. SLqPCR: Functions for analysis of real-time quantitative PCR data at SIRS-Lab GmbH R package (2007) SIRS-Lab GmbH, Jena 4

[2] Jan Hellemans, Geert Mortier, Anne De Paepe, Frank Speleman and Jo Vandesompele. (2007). qBase relative quantification framework and software for management and automated analysis of real-time quantitative PCR data. Genome Biology 2007, 8:R19 (doi:10.1186/gb-2007-8-2-r19) http://genomebiology.com/2007/8/2/R19 4, 18

[3] Erik Willems Luc Leyns, Jo Vandesompele. Standardization of real-time PCR gene expression data from independent biological replicates. Analytical Biochemistry 379 (2008) 127-129 (doi:10.1016/j.ab.2008.04.036). http://www.sciencedirect.com/science/article/pii/S0003269708002649 8, 9

[4] RStudio: Integrated development environment for R (Version 0.96.330) [Computer software]. Boston, MA. Retrieved August 6, 2012. http://www.rstudio.org/ 11

[5] John Verzani. Based on the iwidgets code of Simon Urbanek and suggestions by Simon Urbanek and Philippe Grosjean and Michael Lawrence. gWidgets: gWidgets API for building toolkit-independent, interactive GUIs R package version 0.0-50 (2012) http://CRAN.R-project.org/package=gWidgets 11