

NormqPCR: Functions for normalisation of RT-qPCR data

James Perkins and Matthias Kohl
University College London (UK) / Jena University Hospital (Germany)

March 31, 2012

Contents

| | | |
|----------|--|-----------|
| 1 | Introduction | 1 |
| 2 | Combining technical replicates | 2 |
| 3 | Dealing with undetermined values | 3 |
| 4 | Selection of most stable reference/housekeeping genes | 5 |
| 4.1 | geNorm | 5 |
| 4.2 | NormFinder | 11 |
| 5 | Normalization by means of reference/housekeeping genes | 14 |
| 5.1 | ΔCq method using a single housekeeper | 14 |
| 5.2 | ΔCq method using a combination of housekeeping genes | 15 |
| 5.3 | $2^{-\Delta\Delta Cq}$ method using a single housekeeper | 16 |
| 5.4 | $2^{\Delta\Delta Cq}$ method using a combination of housekeeping genes | 19 |

1 Introduction

The package "NormqPCR" provides methods for the normalization of real-time quantitative RT-PCR data. In this vignette we describe and demonstrate the available functions. Firstly we show how the user may combine technical replicates, deal with undetermined values and deal with values above a user-chosen threshold. The rest of the vignette is split into two distinct sections, the first giving details of different methods to select the best housekeeping gene/genes for normalisation, and the second showing how to use the selected housekeeping gene(s) to produce $2^{-\Delta Cq}$ normalised estimators and $2^{-\Delta\Delta Cq}$ estimators of differential expression.

2 Combining technical replicates

When a raw data file read in using `read.qPCR` contains technical replicates, they are dealt with by concatenating the suffix `_TechRep.n` to the detector name, where `n` in 1, 2...`N` is the number of the replication in the total number of replicates, `N`, based on order of appearance in the qPCR data file.

So if we read in a file with technical replicates, we can see that the detector/feature names are thus suffixed:

```
> library(ReadqPCR) # load the ReadqPCR library
> library(NormqPCR)
> path <- system.file("exData", package = "NormqPCR")
> qPCR.example.techReps <- file.path(path, "qPCR.techReps.txt")
> qPCRBatch.qPCR.techReps <- read.qPCR(qPCR.example.techReps)
> rownames(exprs(qPCRBatch.qPCR.techReps))[1:8]

[1] "gene_aj_TechReps.1" "gene_aj_TechReps.2" "gene_al_TechReps.1"
[4] "gene_al_TechReps.2" "gene_ax_TechReps.1" "gene_ax_TechReps.2"
[7] "gene_bo_TechReps.1" "gene_bo_TechReps.2"
```

It is likely that before continuing with the analysis, the user would wish to average the technical replicates by using the arithmetic mean of the raw Cq values. This can be achieved using the `combineTechReps` function, which will produce a new `qPCRBatch` object, with all tech reps reduced to one reading:

```
> combinedTechReps <- combineTechReps(qPCRBatch.qPCR.techReps)
> combinedTechReps
```

```
qPCRBatch (storageMode: lockedEnvironment)
assayData: 8 features, 3 samples
  element names: exprs
protocolData: none
phenoData
  sampleNames: one three two
  varLabels: sample
  varMetadata: labelDescription
featureData: none
experimentData: use 'experimentData(object)'
Annotation:
```

3 Dealing with undetermined values

When an RT-qPCR experiment does not produce a reading after a certain number of cycles (the cycle threshold), the reading is given as undetermined. These are represented in `qPCRBatch` objects as `NA`. Different users may have different ideas about how many cycles they wish to allow before declaring a detector as not present in the sample. There are two methods for the user to decide what to do with numbers above a given cycle threshold:

First the user might decide that anything above 38 cycles means there is nothing present in their sample, instead of the standard 40 used by the `taqman` software. They can replace the value of all readings above 38 as `NA` using the following:

Firstly read in the `taqman` example file which has 96 detectors, with 4 replicates for `mia` (case) and 4 non-`mia` (control):

```
> path <- system.file("exData", package = "NormqPCR")
> taqman.example <- file.path(path, "/example.txt")
> qPCRBatch.taqman <- read.taqman(taqman.example)
```

We can see that for the detector: `Cc120.Rn00570287_m1` we have these readings for the different samples:

```
> exprs(qPCRBatch.taqman)["Cc120.Rn00570287_m1",]

      fp1.day3.v      fp2.day3.v      fp5.day3.mia      fp6.day3.mia      fp.3.day.3.v
              NA              NA             35.74190             34.05922             35.02052
fp.4.day.3.v fp.7.day.3.mia fp.8.day.3.mia
              NA             35.93689             36.57921
```

We can now use the `replaceAboveCutOff` method in order to replace anything above 35 with `NA`:

```
> qPCRBatch.taqman.replaced <- replaceAboveCutOff(qPCRBatch.taqman, newVal = NA,
+ cutOff = 35)
> exprs(qPCRBatch.taqman.replaced)["Cc120.Rn00570287_m1",]

      fp1.day3.v      fp2.day3.v      fp5.day3.mia      fp6.day3.mia      fp.3.day.3.v
              NA              NA              NA             34.05922             NA
fp.4.day.3.v fp.7.day.3.mia fp.8.day.3.mia
              NA              NA              NA
```

It may also be the case that the user wants to get rid of all `NA` values, and replace them with an arbitrary number. This can be done using the `replaceNAs` method. So if the user wanted to replace all `NAs` with 40, it can be done as follows:

```
> qPCRBatch.taqman.replaced <- replaceNAs(qPCRBatch.taqman, newNA = 40)
> exprs(qPCRBatch.taqman.replaced)["Cc120.Rn00570287_m1",]
```

| | | | | |
|--------------|----------------|----------------|--------------|--------------|
| fp1.day3.v | fp2.day3.v | fp5.day3.mia | fp6.day3.mia | fp.3.day.3.v |
| 40.00000 | 40.00000 | 35.74190 | 34.05922 | 35.02052 |
| fp.4.day.3.v | fp.7.day.3.mia | fp.8.day.3.mia | | |
| 40.00000 | 35.93689 | 36.57921 | | |

In addition, the situation sometimes arises where some readings for a given detector are above a given cycle threshold, but some others are not. The user may decide for example that if a given number of readings are NAs, then all of the readings for this detector should be NAs. This is important because otherwise an unusual reading for one detector might lead to an inaccurate estimate for the expression of a given gene.

This process will necessarily be separate for the different sample types, since you might expect a given gene to show expression in one sample type compared to another. Therefore it is necessary to designate the replicates per sample type using a contrast matrix. It is also necessary to make a sampleMaxMatrix which gives a maximum number of NAs allowed for each sample type.

So in the example file above we two sample types, with 4 biological replicates for each, the contrastMatrix and sampleMaxMatrix might be constructed like this:

```
> sampleNames(qPCRBatch.taqman)
```

```
[1] "fp1.day3.v"      "fp2.day3.v"      "fp5.day3.mia"    "fp6.day3.mia"
[5] "fp.3.day.3.v"    "fp.4.day.3.v"    "fp.7.day.3.mia"  "fp.8.day.3.mia"
```

```
> a <- c(0,0,1,1,0,0,1,1) # one for each sample type, with 1 representing
> b <- c(1,1,0,0,1,1,0,0) # position of sample type in the sampleNames vector
> contM <- cbind(a,b)
> colnames(contM) <- c("case","control") # then give the names of each sample type
> rownames(contM) <- sampleNames(qPCRBatch.taqman) # and the rows of the matrix
> contM
```

| | case | control |
|----------------|------|---------|
| fp1.day3.v | 0 | 1 |
| fp2.day3.v | 0 | 1 |
| fp5.day3.mia | 1 | 0 |
| fp6.day3.mia | 1 | 0 |
| fp.3.day.3.v | 0 | 1 |
| fp.4.day.3.v | 0 | 1 |
| fp.7.day.3.mia | 1 | 0 |
| fp.8.day.3.mia | 1 | 0 |

```
> sMaxM <- t(as.matrix(c(3,3))) # now make the contrast matrix
> colnames(sMaxM) <- c("case","control") # make sure these line up with samples
> sMaxM
```

```
      case control
[1,]    3      3
```

More details on contrast matrices can be found in the limma manual, which requires a similar matrix when testing for differential expression between samples.

For example, if the user decides that if at least 3 out of 4 readings are NAs for a given detector, then all readings should be NA, they can do the following, using the `makeAllNewVal` method:

```
> qPCRBatch.taqman.replaced <- makeAllNewVal(qPCRBatch.taqman, contM, sMaxM, newVal=NA)
```

Here you can see for the Ccl20.Rn00570287_m1 detector, the control values have been made all NA, whereas before 3 were NA and one was 35. However the case values have been kept, since they were all below the NA threshold. It is important to filter the data in this way to ensure the correct calculations are made downstream when calculating variation and other parameters.

```
> exprs(qPCRBatch.taqman.replaced)["Ccl20.Rn00570287_m1",]
```

| | | | | |
|--------------|----------------|----------------|--------------|--------------|
| fp1.day3.v | fp2.day3.v | fp5.day3.mia | fp6.day3.mia | fp.3.day.3.v |
| NA | NA | 35.74190 | 34.05922 | NA |
| fp.4.day.3.v | fp.7.day.3.mia | fp.8.day.3.mia | | |
| NA | 35.93689 | 36.57921 | | |

4 Selection of most stable reference/housekeeping genes

This section contains two subsections containing different methods for the selection of appropriate housekeeping genes.

4.1 geNorm

We describe the selection of the best (most stable) reference/housekeeping genes using the method of Vandesompele et al (2002) [3] (in the sequel: Vand02) which is called *geNorm*. We first load the package and the data

```
> options(width = 68)
> data(geNorm)
> str(exprs(geNorm.qPCRBatch))
```

```

num [1:10, 1:85] 0.0425 0.0576 0.1547 0.1096 0.118 ...
- attr(*, "dimnames")=List of 2
..$ : chr [1:10] "ACTB" "B2M" "GAPD" "HMBS" ...
..$ : chr [1:85] "BM1" "BM2" "BM3" "BM4" ...

```

We start by ranking the selected reference/housekeeping genes. The geNorm algorithm implemented in function `selectHKs` proceeds stepwise; confer Section “Materials and methods” in Vand02. That is, the gene stability measure M of all candidate genes is computed and the gene with the highest M value is excluded. Then, the gene stability measure M for the remaining gene is calculated and so on. This procedure is repeated until two respectively, `minNrHK` genes remain.

```

> tissue <- as.factor(c(rep("BM", 9), rep("FIB", 20), rep("LEU", 13),
+                       rep("NB", 34), rep("POOL", 9)))
> res.BM <- selectHKs(geNorm.qPCRBatch[,tissue == "BM"], method = "geNorm",
+                     Symbols = featureNames(geNorm.qPCRBatch), minNrHK = 2, log = FALSE)

```

| | | | | | |
|-----------|-----------|-----------|-----------|-----------|-----------|
| HPRT1 | YWHAZ | RPL13A | UBC | GAPD | SDHA |
| 0.5160313 | 0.5314564 | 0.5335963 | 0.5700961 | 0.6064919 | 0.6201470 |
| TBP | HMBS | B2M | ACTB | | |
| 0.6397969 | 0.7206013 | 0.7747634 | 0.8498739 | | |
| HPRT1 | RPL13A | YWHAZ | UBC | GAPD | SDHA |
| 0.4705664 | 0.5141375 | 0.5271169 | 0.5554718 | 0.5575295 | 0.5738460 |
| TBP | HMBS | B2M | | | |
| 0.6042110 | 0.6759176 | 0.7671985 | | | |
| HPRT1 | RPL13A | SDHA | YWHAZ | UBC | GAPD |
| 0.4391222 | 0.4733732 | 0.5243665 | 0.5253471 | 0.5403137 | 0.5560120 |
| TBP | HMBS | | | | |
| 0.5622094 | 0.6210820 | | | | |
| HPRT1 | RPL13A | YWHAZ | UBC | SDHA | GAPD |
| 0.4389069 | 0.4696398 | 0.4879728 | 0.5043292 | 0.5178634 | 0.5245346 |
| TBP | | | | | |
| 0.5563591 | | | | | |
| HPRT1 | RPL13A | UBC | YWHAZ | GAPD | SDHA |
| 0.4292808 | 0.4447874 | 0.4594181 | 0.4728920 | 0.5012107 | 0.5566762 |
| UBC | RPL13A | HPRT1 | YWHAZ | GAPD | |
| 0.4195958 | 0.4204997 | 0.4219179 | 0.4424631 | 0.4841646 | |
| RPL13A | UBC | YWHAZ | HPRT1 | | |
| 0.3699163 | 0.3978736 | 0.4173706 | 0.4419220 | | |
| UBC | RPL13A | YWHAZ | | | |
| 0.3559286 | 0.3761358 | 0.3827933 | | | |

```

RPL13A      UBC
0.3492712 0.3492712

```

```

> res.POOL <- selectHKs(geNorm.qPCRBatch[,tissue == "POOL"], method = "geNorm",
+                       Symbols = featureNames(geNorm.qPCRBatch), minNrHK = 2,
+                       trace = FALSE, log = FALSE)
> res.FIB <- selectHKs(geNorm.qPCRBatch[,tissue == "FIB"], method = "geNorm",
+                      Symbols = featureNames(geNorm.qPCRBatch), minNrHK = 2,
+                      trace = FALSE, log = FALSE)
> res.LEU <- selectHKs(geNorm.qPCRBatch[,tissue == "LEU"], method = "geNorm",
+                      Symbols = featureNames(geNorm.qPCRBatch), minNrHK = 2,
+                      trace = FALSE, log = FALSE)
> res.NB <- selectHKs(geNorm.qPCRBatch[,tissue == "NB"], method = "geNorm",
+                     Symbols = featureNames(geNorm.qPCRBatch), minNrHK = 2,
+                     trace = FALSE, log = FALSE)

```

We obtain the following ranking of genes (cf. Table 3 in Vand02)

```

> ranks <- data.frame(c(1, 1:9), res.BM$ranking, res.POOL$ranking,
+                     res.FIB$ranking, res.LEU$ranking,
+                     res.NB$ranking)
> names(ranks) <- c("rank", "BM", "POOL", "FIB", "LEU", "NB")
> ranks

```

| | rank | BM | POOL | FIB | LEU | NB |
|----|------|--------|--------|--------|--------|--------|
| 1 | 1 | RPL13A | GAPD | GAPD | UBC | GAPD |
| 2 | 1 | UBC | SDHA | HPRT1 | YWHAZ | HPRT1 |
| 3 | 2 | YWHAZ | HMBS | YWHAZ | B2M | SDHA |
| 4 | 3 | HPRT1 | HPRT1 | UBC | GAPD | UBC |
| 5 | 4 | GAPD | TBP | ACTB | RPL13A | HMBS |
| 6 | 5 | SDHA | UBC | TBP | TBP | YWHAZ |
| 7 | 6 | TBP | RPL13A | SDHA | SDHA | TBP |
| 8 | 7 | HMBS | YWHAZ | RPL13A | HPRT1 | ACTB |
| 9 | 8 | B2M | ACTB | B2M | HMBS | RPL13A |
| 10 | 9 | ACTB | B2M | HMBS | ACTB | B2M |

Remark 1:

Since the computation is based on gene ratios, the two most stable control genes in each cell type cannot be ranked.

We plot the average expression stability M for each cell type (cf. Figure 2 in Vand02).

```

> library(RColorBrewer)
> mypalette <- brewer.pal(5, "Set1")

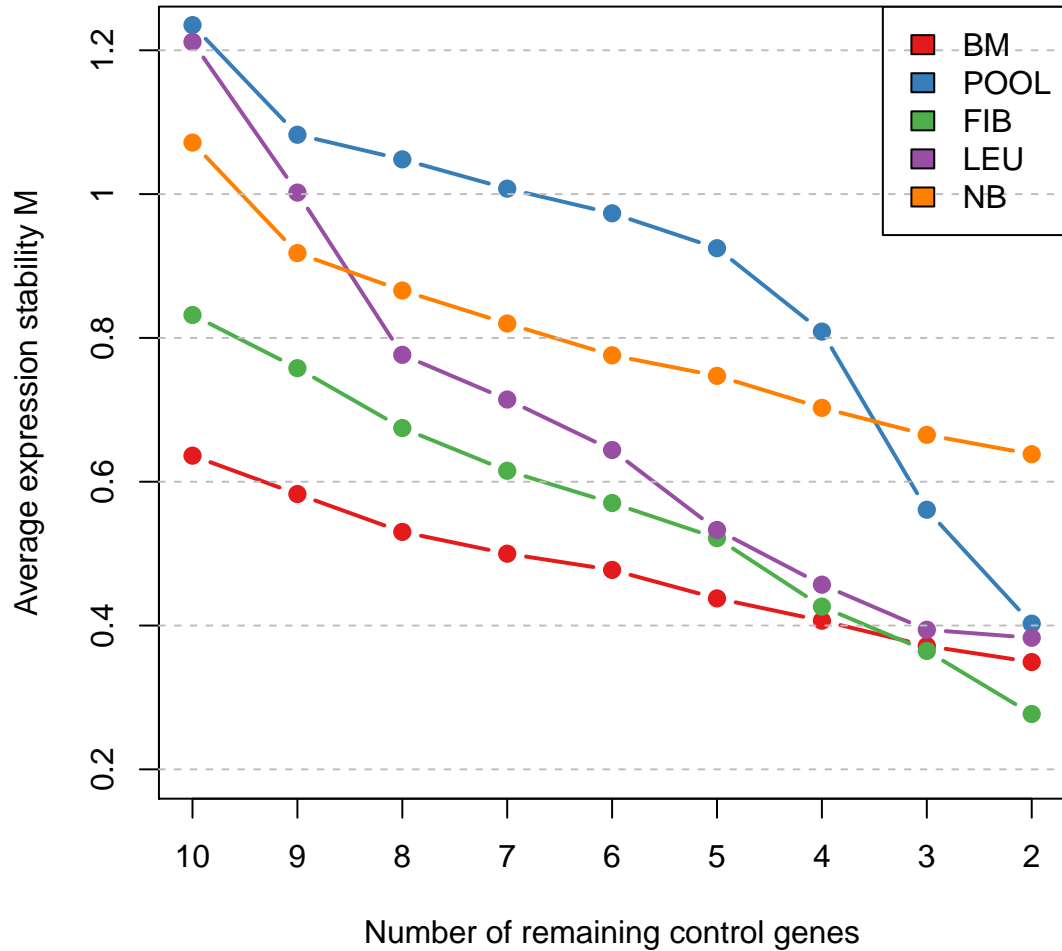
```

```

> matplot(cbind(res.BM$meanM, res.POOL$meanM, res.FIB$meanM,
+               res.LEU$meanM, res.NB$meanM), type = "b",
+         ylab = "Average expression stability M",
+         xlab = "Number of remaining control genes",
+         axes = FALSE, pch = 19, col = mypalette,
+         ylim = c(0.2, 1.22), lty = 1, lwd = 2,
+         main = "Figure 2 in Vandesompele et al. (2002)")
> axis(1, at = 1:9, labels = as.character(10:2))
> axis(2, at = seq(0.2, 1.2, by = 0.2), labels = seq(0.2, 1.2, by = 0.2))
> box()
> abline(h = seq(0.2, 1.2, by = 0.2), lty = 2, lwd = 1, col = "grey")
> legend("topright", legend = c("BM", "POOL", "FIB", "LEU", "NB"),
+       fill = mypalette)

```


Figure 2 in Vandesompele et al. (2002)



Second, we plot the pairwise variation for each cell type (cf. Figure 3 (a) in Vand02)

```
> mypalette <- brewer.pal(8, "YlGnBu")
> barplot(cbind(res.POOL$variation, res.LEU$variation, res.NB$variation,
+             res.FIB$variation, res.BM$variation), beside = TRUE,
+       col = mypalette, space = c(0, 2),
+       names.arg = c("POOL", "LEU", "NB", "FIB", "BM"),
+       ylab = "Pairwise variation V",
+       main = "Figure 3(a) in Vandesompele et al. (2002)")
```

```

> legend("topright", legend = c("V9/10", "V8/9", "V7/8", "V6/7",
+                               "V5/6", "V4/5", "V3/4", "V2/3"),
+       fill = mypalette, ncol = 2)
> abline(h = seq(0.05, 0.25, by = 0.05), lty = 2, col = "grey")
> abline(h = 0.15, lty = 1, col = "black")

```

Figure 3(a) in Vandesompele et al. (2002)



Remark 2:

Vand02 recommend a cut-off value of 0.15 for the pairwise variation. Below this bound the inclusion of an additional housekeeping gene is not required.

4.2 NormFinder

The second method for selection reference/housekeeping genes implemented in package is the method derived by [1] (in the sequel: And04) called *NormFinder*.

The ranking contained in Table 3 of And04 can be obtained via

```
> data(Colon)
> str(exprs(Colon.qPCRBatch))

num [1:13, 1:40] 922 1167 920 818 1409 ...
- attr(*, "dimnames")=List of 2
..$ : chr [1:13] "ACTB" "CFL1" "CLTC" "FLJ20030" ...
..$ : chr [1:40] "C.C108T" "C.C215T" "C.C242T" "C.C415T" ...

> group <- pData(Colon.qPCRBatch)[,"Group"]
> res.Colon <- stabMeasureRho(Colon.qPCRBatch, group = group,
+                             log = FALSE)
> sort(res.Colon) # cf. Table 3 in Andersen et al (2004)

      TPT1      UBC      SUI1      GAPD      CFL1      TUBA6
0.1414763 0.1610925 0.1730781 0.1767370 0.1771489 0.1775287
      RPS13      UBB      NACA      CLTC      RPS23      ACTB
0.1869378 0.1885640 0.1919831 0.2029681 0.2201116 0.2329093
      FLJ20030
0.2568652

> data(Bladder)
> str(exprs(Bladder.qPCRBatch))

num [1:14, 1:28] 3041 4453 2465 1329 6102 ...
- attr(*, "dimnames")=List of 2
..$ : chr [1:14] "ACTB" "ATP5B" "CFL1" "FLJ20030" ...
..$ : chr [1:28] "X1014.1" "X1131.1" "X1164.1" "X1206.1" ...

> group <- pData(Bladder.qPCRBatch)[,"Group"]
> cat("test1")

test1

> res.Bladder <- stabMeasureRho(Bladder.qPCRBatch, group = group,
+                               log = FALSE)
> cat("test test")

test test
```

```
> sort(res.Bladder)
```

| | | | | | |
|------------|------------|------------|------------|------------|------------|
| HSPCB | ATP5B | RPS23 | CFL1 | UBC | TEGT |
| 0.08078175 | 0.10094343 | 0.12881495 | 0.13119714 | 0.13363952 | 0.14265863 |
| RPS13 | TPT1 | FLJ20030 | GAPD | UBB | FLOT2 |
| 0.15495774 | 0.16566827 | 0.17352074 | 0.18887425 | 0.19256415 | 0.20147640 |
| ACTB | S100A6 | | | | |
| 0.22197915 | 0.24452110 | | | | |

Of course, we can also reproduce the geNorm ranking also included in Table 3 of And04.

```
> selectHKs(Colon.qPCRBatch, log = FALSE, trace = FALSE,
+           Symbols = featureNames(Colon.qPCRBatch))$ranking
```

| | | | | | |
|------------|--------|---------|--------|--------|--------|
| 1 | 1 | 3 | 4 | 5 | 6 |
| "RPS23" | "TPT1" | "RPS13" | "SUI1" | "UBC" | "GAPD" |
| 7 | 8 | 9 | 10 | 11 | 12 |
| "TUBA6" | "UBB" | "NACA" | "CFL1" | "CLTC" | "ACTB" |
| 13 | | | | | |
| "FLJ20030" | | | | | |

```
> selectHKs(Bladder.qPCRBatch, log = FALSE, trace = FALSE,
+           Symbols = featureNames(Bladder.qPCRBatch))$ranking
```

| | | | | | |
|---------|----------|---------|------------|---------|--------|
| 1 | 1 | 3 | 4 | 5 | 6 |
| "CFL1" | "UBC" | "ATP5B" | "HSPCB" | "GAPD" | "TEGT" |
| 7 | 8 | 9 | 10 | 11 | 12 |
| "RPS23" | "RPS13" | "TPT1" | "FLJ20030" | "FLOT2" | "UBB" |
| 13 | 14 | | | | |
| "ACTB" | "S100A6" | | | | |

As we are often interested in more than one reference/housekeeping gene we also implemented a step-wise procedure of the NormFinder algorithm explained in Section “Average control gene” in the supplementary information of And04. This procedure is available via function `selectHKs`.

```
> group <- pData(Colon.qPCRBatch)[,"Group"]
> selectHKs(Colon.qPCRBatch,
+           log = FALSE, trace = TRUE, group = group,
+           Symbols = featureNames(Colon.qPCRBatch), minNrHKs = 12,
+           method = "NormFinder")$ranking
```

| | | | | | |
|------------|------------|------------|------------|------------|------------|
| TPT1 | UBC | SUI1 | GAPD | CFL1 | TUBA6 |
| 0.1414763 | 0.1610925 | 0.1730781 | 0.1767370 | 0.1771489 | 0.1775287 |
| RPS13 | UBB | NACA | CLTC | RPS23 | ACTB |
| 0.1869378 | 0.1885640 | 0.1919831 | 0.2029681 | 0.2201116 | 0.2329093 |
| FLJ20030 | | | | | |
| 0.2568652 | | | | | |
| UBC | TUBA6 | GAPD | SUI1 | CFL1 | RPS13 |
| 0.1031043 | 0.1114774 | 0.1142132 | 0.1152734 | 0.1185269 | 0.1200290 |
| CLTC | NACA | UBB | ACTB | RPS23 | FLJ20030 |
| 0.1232217 | 0.1297298 | 0.1301249 | 0.1384574 | 0.1445223 | 0.1471372 |
| NACA | CFL1 | SUI1 | GAPD | UBB | TUBA6 |
| 0.08888919 | 0.09117470 | 0.09249714 | 0.09290098 | 0.09697394 | 0.09763415 |
| RPS13 | CLTC | RPS23 | FLJ20030 | ACTB | |
| 0.09769254 | 0.10351175 | 0.10424815 | 0.11516759 | 0.11855899 | |
| TUBA6 | GAPD | CLTC | CFL1 | SUI1 | RPS13 |
| 0.07604126 | 0.07857074 | 0.08012378 | 0.08118840 | 0.08124209 | 0.08381455 |
| ACTB | FLJ20030 | UBB | RPS23 | | |
| 0.08704531 | 0.09214267 | 0.09281511 | 0.09926029 | | |
| GAPD | CFL1 | SUI1 | CLTC | RPS13 | UBB |
| 0.07052455 | 0.07075470 | 0.07199446 | 0.07486103 | 0.07540652 | 0.07649412 |
| FLJ20030 | RPS23 | ACTB | | | |
| 0.08378769 | 0.08455423 | 0.08599148 | | | |
| SUI1 | RPS13 | CFL1 | UBB | CLTC | FLJ20030 |
| 0.06745408 | 0.06754705 | 0.06822068 | 0.06887908 | 0.07101164 | 0.07230004 |
| RPS23 | ACTB | | | | |
| 0.08041376 | 0.08261883 | | | | |
| RPS13 | CFL1 | UBB | CLTC | FLJ20030 | RPS23 |
| 0.06496047 | 0.06559937 | 0.06648983 | 0.06742791 | 0.06847696 | 0.07708310 |
| ACTB | | | | | |
| 0.07803569 | | | | | |
| CFL1 | CLTC | UBB | FLJ20030 | ACTB | RPS23 |
| 0.06270147 | 0.06274136 | 0.06632289 | 0.06830593 | 0.07273986 | 0.07379364 |
| CLTC | UBB | FLJ20030 | ACTB | RPS23 | |
| 0.06142205 | 0.06258837 | 0.06323820 | 0.07119490 | 0.07415997 | |
| UBB | RPS23 | FLJ20030 | ACTB | | |
| 0.04975140 | 0.06153268 | 0.06260266 | 0.07634498 | | |
| FLJ20030 | RPS23 | ACTB | | | |
| 0.05658949 | 0.06211800 | 0.06395867 | | | |
| RPS23 | ACTB | | | | |
| 0.06731095 | 0.07412848 | | | | |
| 1 | 2 | 3 | 4 | 5 | 6 |

```

      "TPT1"      "UBC"      "NACA"      "TUBA6"      "GAPD"      "SUI1"
        7         8         9         10         11         12
"RPS13"      "CFL1"      "CLTC"      "UBB" "FLJ20030"      "RPS23"

> group <- pData(Bladder.qPCRBatch)[,"Group"]
> selectHKs(Bladder.qPCRBatch, group = group,
+           log = FALSE, trace = FALSE,
+           Symbols = featureNames(Bladder.qPCRBatch), minNrHKs = 13,
+           method = "NormFinder")$ranking

      1         2         3         4         5         6
"HSPCB"      "ATP5B"      "RPS23"      "CFL1"      "UBC"      "TEGT"
      7         8         9         10         11         12
"RPS13"      "TPT1"      "GAPD"      "UBB" "FLJ20030"      "FLOT2"
     13
"S100A6"

```

5 Normalization by means of reference/housekeeping genes

5.1 ΔC_q method using a single housekeeper

The ΔC_q method normalises detectors within a sample by subtracting the cycle time value of the housekeeper gene from the other genes. This can be done in NormqPCR as follows:

for the example dataset from "ReadqPCR" we must first read in the data:

```

> path <- system.file("exData", package = "NormqPCR")
> taqman.example <- file.path(path, "example.txt")
> qPCR.example <- file.path(path, "qPCR.example.txt")
> qPCRBatch.taqman <- read.taqman(taqman.example)

```

We then need to supply a housekeeper gene to be subtracted:

```

> hkgs<-"Actb-Rn00667869_m1"
> qPCRBatch.norm <- deltaCq(qPCRBatch = qPCRBatch.taqman, hkgs = hkgs, calc="arith")
> head(exprs(qPCRBatch.norm))

```

| | fp1.day3.v | fp2.day3.v | fp5.day3.mia |
|----------------------|------------|------------|--------------|
| Actb.Rn00667869_m1 | 0.000000 | 0.000000 | 0.000000 |
| Adipoq.Rn00595250_m1 | 0.016052 | -0.116520 | 2.933523 |
| Adrbk1.Rn00562822_m1 | NA | NA | 6.566628 |
| Agtrl1.Rn00580252_s1 | 4.899380 | 5.035841 | 6.397364 |
| Alpl.Rn00564931_m1 | 12.531942 | 11.808657 | 13.035166 |

| | | | |
|----------------------|----------------|----------------|--------------|
| B2m.Rn00560865_m1 | 0.741558 | 0.890717 | 2.040470 |
| | fp6.day3.mia | fp.3.day.3.v | fp.4.day.3.v |
| Actb.Rn00667869_m1 | 0.000000 | 0.000000 | 0.000000 |
| Adipoq.Rn00595250_m1 | 2.540987 | -0.178971 | -0.563263 |
| Adrbk1.Rn00562822_m1 | 6.642561 | NA | NA |
| Agtrl1.Rn00580252_s1 | 5.680837 | 5.220796 | 4.425364 |
| Alpl.Rn00564931_m1 | 12.239549 | 12.394802 | 11.772896 |
| B2m.Rn00560865_m1 | 2.234605 | 0.505516 | 0.877598 |
| | fp.7.day.3.mia | fp.8.day.3.mia | |
| Actb.Rn00667869_m1 | 0.000000 | 0.000000 | |
| Adipoq.Rn00595250_m1 | 2.458509 | 2.736475 | |
| Adrbk1.Rn00562822_m1 | 3.737100 | 6.873568 | |
| Agtrl1.Rn00580252_s1 | 4.794776 | 5.345202 | |
| Alpl.Rn00564931_m1 | 12.110000 | 12.255186 | |
| B2m.Rn00560865_m1 | 1.927563 | 1.903269 | |

This returns a new `qPCRBatch`, with new values in the `exprs` slot. This will be compatible with many other bioconductor and R packages, such as `heatmap`.

Note these numbers might be negative. For further analysis requiring positive values only, 2^{\cdot} can be used to transform the data into $2^{\Delta CT}$ values.

5.2 ΔCq method using a combination of housekeeping genes

If the user wishes to normalise by more than one housekeeping gene, for example if they have found a more than one housekeeping gene using the `NormFinder/geNorm` algorithms described above, they can. This is implemented by calculating the average of these values to form a "pseudo-housekeeper" which is subtracted from the other values. So using the same dataset as above, using housekeeping genes `GAPDH`, `Beta-2-microglobulin` and `Beta-actin`, the following steps would be taken:

```
> hkg<-c("Actb-Rn00667869_m1", "B2m-Rn00560865_m1", "Gapdh-Rn99999916_s1")
> qPCRBatch.norm <- deltaCq(qPCRBatch = qPCRBatch.taqman, hkg = hkg, calc="arith")
> head(exprs(qPCRBatch.norm))
```

| | | | |
|----------------------|--------------|--------------|--------------|
| | fp1.day3.v | fp2.day3.v | fp5.day3.mia |
| Actb.Rn00667869_m1 | -1.2998917 | -1.2816963 | -1.380296 |
| Adipoq.Rn00595250_m1 | -1.2838397 | -1.3982163 | 1.553227 |
| Adrbk1.Rn00562822_m1 | NA | NA | 5.186332 |
| Agtrl1.Rn00580252_s1 | 3.5994883 | 3.7541447 | 5.017068 |
| Alpl.Rn00564931_m1 | 11.2320503 | 10.5269607 | 11.654870 |
| B2m.Rn00560865_m1 | -0.5583337 | -0.3909793 | 0.660174 |
| | fp6.day3.mia | fp.3.day.3.v | fp.4.day.3.v |

| | | | |
|----------------------|----------------|----------------|------------|
| Actb.Rn00667869_m1 | -1.5106197 | -1.1644617 | -1.1714227 |
| Adipoq.Rn00595250_m1 | 1.0303673 | -1.3434327 | -1.7346857 |
| Adrbk1.Rn00562822_m1 | 5.1319413 | NA | NA |
| Agtrl1.Rn00580252_s1 | 4.1702173 | 4.0563343 | 3.2539413 |
| Alpl.Rn00564931_m1 | 10.7289293 | 11.2303403 | 10.6014733 |
| B2m.Rn00560865_m1 | 0.7239853 | -0.6589457 | -0.2938247 |
| | fp.7.day.3.mia | fp.8.day.3.mia | |
| Actb.Rn00667869_m1 | -1.323712 | -1.286277 | |
| Adipoq.Rn00595250_m1 | 1.134797 | 1.450198 | |
| Adrbk1.Rn00562822_m1 | 2.413388 | 5.587291 | |
| Agtrl1.Rn00580252_s1 | 3.471064 | 4.058925 | |
| Alpl.Rn00564931_m1 | 10.786288 | 10.968909 | |
| B2m.Rn00560865_m1 | 0.603851 | 0.616992 | |

5.3 $2^{-\Delta\Delta C_q}$ method using a single housekeeper

It is possible to use the $2^{-\Delta\Delta C_q}$ method for calculating relative gene expression between two sample types. Both the same well and the separate well methods as detailed in [2] can be used for this purpose, and will produce the same answers, but with different levels of variation. By default detectors in the same sample will be paired with the housekeeper, and the standard deviation used will be that of the differences between detectors and the housekeepers. However, if the argument `paired=FALSE` is added, standard deviation between case and control will be calculated as $s = \sqrt{s_1^2 + s_2^2}$, where s_1 is the standard deviation for the detector readings and s_2 is the standard deviation the housekeeper gene readings. The latter approach is not recommended when the housekeeper and genes to be compared are from the same sample, as is the case when using the taqman cards, but is included for completeness and for situations where readings for the housekeeper might be taken from a separate biological replicate (for example in a *post hoc* manner due to the originally designated housekeeping genes not performing well), or for when NormqPCR is used for more traditional qPCR where the products undergo amplifications from separate wells.

for the example dataset from "ReadqPCR" we must first read in the data:

```
> path <- system.file("exData", package = "NormqPCR")
> taqman.example <- file.path(path, "example.txt")
> qPCR.example <- file.path(path, "qPCR.example.txt")
> qPCRBatch.taqman <- read.taqman(taqman.example)
```

`deltaDeltaCq` also requires a contrast matrix. This is to contain columns which will be used to specify the samples representing `case` and `control` which are to be compared, in a similar way to the "limma" package. these columns should contain 1s or 0s which refer to the samples in either category:


```
> contM <- cbind(c(0,0,1,1,0,0,1,1),c(1,1,0,0,1,1,0,0))
> colnames(contM) <- c("interestingPhenotype","wildTypePhenotype")
> rownames(contM) <- sampleNames(qPCRBatch.taqman)
> contM
```

| | interestingPhenotype | wildTypePhenotype |
|----------------|----------------------|-------------------|
| fp1.day3.v | 0 | 1 |
| fp2.day3.v | 0 | 1 |
| fp5.day3.mia | 1 | 0 |
| fp6.day3.mia | 1 | 0 |
| fp.3.day.3.v | 0 | 1 |
| fp.4.day.3.v | 0 | 1 |
| fp.7.day.3.mia | 1 | 0 |
| fp.8.day.3.mia | 1 | 0 |

We can now normalise each sample by a given housekeeping gene and then look at the ratio of expression between the case and control samples. Results show (by column): 1) Name of gene represented by detector. 2) Case ΔCq for the detector: the average cycle time for this detector in the samples denoted as "case" - the housekeeper cycle time. 3) the standard deviation for the cycle times used to calculate the value in column 2). 4) Control ΔCq for the detector: the average cycle time for this detector in the samples denoted as "controller", or the "callibrator" samples - the housekeeper cycle time. 5) The standard deviation for the cycle times used to calculate the value in column 4). 6) $2^{-\Delta\Delta Cq}$ - The difference between the ΔCq values for case and control. We then find 2^{-} of this value. 7) and 8) correspond to 1 s.d. either side of the mean value, as detailed in [2].

```
> hkg <- "Actb-Rn00667869_m1"
> ddCq.taqman <- deltaDeltaCq(qPCRBatch = qPCRBatch.taqman, maxNACase=1, maxNAControl=1, hkg=hkg)
> head(ddCq.taqman)
```

| | ID | $2^{-dCt.interestingPhenotype}$ |
|---|----------------------|---------------------------------|
| 1 | Actb.Rn00667869_m1 | 1.000e+00 |
| 2 | Adipoq.Rn00595250_m1 | 1.587e-01 |
| 3 | Adrbk1.Rn00562822_m1 | 2.602e-02 |
| 4 | Agtrl1.Rn00580252_s1 | 2.300e-02 |
| 5 | Alpl.Rn00564931_m1 | 1.892e-04 |
| 6 | B2m.Rn00560865_m1 | 2.464e-01 |

| | interestingPhenotype.sd | $2^{-dCt.wildTypePhenotype}$ |
|---|-------------------------|------------------------------|
| 1 | 0.000e+00 | 1.000e+00 |
| 2 | 2.280e-02 | 1.171e+00 |
| 3 | 3.266e-02 | NA |
| 4 | 1.014e-02 | 3.434e-02 |

| | | |
|---|-----------------------------|---|
| 5 | 4.770e-05 | 2.298e-04 |
| 6 | 2.498e-02 | 5.965e-01 |
| | wildTypePhenotype.sd | 2 ⁻ ddCt 2 ⁻ ddCt.min 2 ⁻ ddCt.max |
| 1 | 0.000e+00 1 | NA NA |
| 2 | 2.131e-01 0.135541545192243 | NA NA |
| 3 | NA + | NA NA |
| 4 | 8.584e-03 0.669721905042939 | NA NA |
| 5 | 6.107e-05 0.823327272466571 | NA NA |
| 6 | 7.668e-02 0.413128242070071 | NA NA |

We can also average the taqman data using the separate samples/wells method . Here standard deviation is calculated separately and then combined, as described above. Therefore the pairing of housekeeper with the detector value within the same sample is lost. This can potentially increase variance.

```
> hkg <- "Actb-Rn00667869_m1"
> ddCqAvg.taqman <- deltaDeltaCq(qPCRBatch = qPCRBatch.taqman, maxNACase=1, maxNAControl=1,
> head(ddCqAvg.taqman)
```

| | ID | 2 ⁻ dCt.interestingPhenotype |
|---|-----------------------------|---|
| 1 | Actb.Rn00667869_m1 | 1.000e+00 |
| 2 | Adipoq.Rn00595250_m1 | 1.587e-01 |
| 3 | Adrbk1.Rn00562822_m1 | 2.602e-02 |
| 4 | Agtrl1.Rn00580252_s1 | 2.300e-02 |
| 5 | Alpl.Rn00564931_m1 | 1.892e-04 |
| 6 | B2m.Rn00560865_m1 | 2.464e-01 |
| | interestingPhenotype.sd | 2 ⁻ dCt.wildTypePhenotype |
| 1 | 0.000e+00 | 1.000e+00 |
| 2 | 2.280e-02 | 1.171e+00 |
| 3 | 3.266e-02 | NA |
| 4 | 1.014e-02 | 3.434e-02 |
| 5 | 4.770e-05 | 2.298e-04 |
| 6 | 2.498e-02 | 5.965e-01 |
| | wildTypePhenotype.sd | 2 ⁻ ddCt 2 ⁻ ddCt.min 2 ⁻ ddCt.max |
| 1 | 0.000e+00 1 | NA NA |
| 2 | 2.131e-01 0.135541545192243 | NA NA |
| 3 | NA + | NA NA |
| 4 | 8.584e-03 0.669721905042939 | NA NA |
| 5 | 6.107e-05 0.823327272466571 | NA NA |
| 6 | 7.668e-02 0.413128242070071 | NA NA |

5.4 $2^{\Delta\Delta Cq}$ method using a combination of housekeeping genes

If the user wishes to normalise by more than one housekeeping gene, for example if they have found a more than one housekeeping gene using the NormFinder/geNorm algorithms described above, they can. This is implemented by calculating the average of these values using the geometric mean to form a "pseudo-housekeeper" which is subtracted from the other values. For the dataset above, using housekeeping genes GAPDH, Beta-2-microglobulin and Beta-actin:

```
> qPCRBatch.taqman <- read.taqman(taqman.example)
> contM <- cbind(c(0,0,1,1,0,0,1,1),c(1,1,0,0,1,1,0,0))
> colnames(contM) <- c("interestingPhenotype","wildTypePhenotype")
> rownames(contM) <- sampleNames(qPCRBatch.taqman)
> hkg<-c("Actb-Rn00667869_m1", "B2m-Rn00560865_m1", "Gapdh-Rn99999916_s1")
> ddCq.gM.taqman <- deltaDeltaCq(qPCRBatch = qPCRBatch.taqman, maxNAcase=1, maxNAcontrol=1,
> head(ddCq.gM.taqman)
```

| | ID | 2 ⁻ dCt.interestingPhenotype | | | |
|---|-------------------------|---|--------------------------|--------------------------|--|
| 1 | Actb.Rn00667869_m1 | 2.594e+00 | | | |
| 2 | Adipoq.Rn00595250_m1 | 4.083e-01 | | | |
| 3 | Adrbk1.Rn00562822_m1 | 4.182e-02 | | | |
| 4 | Agtrl1.Rn00580252_s1 | 5.520e-02 | | | |
| 5 | Alpl.Rn00564931_m1 | 4.767e-04 | | | |
| 6 | B2m.Rn00560865_m1 | 6.367e-01 | | | |
| | interestingPhenotype.sd | 2 ⁻ dCt.wildTypePhenotype | | | |
| 1 | 0.09819 | 2.345e+00 | | | |
| 2 | 0.24929 | 2.713e+00 | | | |
| 3 | 1.45844 | NA | | | |
| 4 | 0.63719 | 7.878e-02 | | | |
| 5 | 0.42589 | 5.242e-04 | | | |
| 6 | 0.05413 | 1.390e+00 | | | |
| | wildTypePhenotype.sd | 2 ⁻ -ddCt | 2 ⁻ -ddCt.min | 2 ⁻ -ddCt.max | |
| 1 | 0.071373 | 1.10638851325547 | 1.034e+00 | 1.184310 | |
| 2 | 0.201905 | 0.150497255530234 | 1.266e-01 | 0.178884 | |
| 3 | NA | + | NA | NA | |
| 4 | 0.333840 | 0.700597907024805 | 4.505e-01 | 1.089636 | |
| 5 | 0.386280 | 0.909381199520663 | 6.769e-01 | 1.221662 | |
| 6 | 0.163975 | 0.457939394245865 | 4.411e-01 | 0.475448 | |

There is also the option of using the mean housekeeper method using shared variance between the samples being compared, similar to the second `deltaDeltaCq` method shown above.

```

> qPCRBatch.taqman <- read.taqman(taqman.example)
> contM <- cbind(c(0,0,1,1,0,0,1,1),c(1,1,0,0,1,1,0,0))
> colnames(contM) <- c("interestingPhenotype","wildTypePhenotype")
> rownames(contM) <- sampleNames(qPCRBatch.taqman)
> hkg<-c("Actb-Rn00667869_m1", "B2m-Rn00560865_m1", "Gapdh-Rn99999916_s1")
> ddAvgCq.gM.taqman <-deltaDeltaCq(qPCRBatch = qPCRBatch.taqman, maxNACase=1, maxNAControl=
> head(ddAvgCq.gM.taqman)

```

| | ID | 2 ⁻ dCt.interestingPhenotype |
|---|----------------------|---|
| 1 | Actb.Rn00667869_m1 | 2.594e+00 |
| 2 | Adipoq.Rn00595250_m1 | 4.083e-01 |
| 3 | Adrbk1.Rn00562822_m1 | 4.182e-02 |
| 4 | Agtrl1.Rn00580252_s1 | 5.520e-02 |
| 5 | Alpl.Rn00564931_m1 | 4.767e-04 |
| 6 | B2m.Rn00560865_m1 | 6.367e-01 |

| | interestingPhenotype.sd | 2 ⁻ dCt.wildTypePhenotype |
|---|-------------------------|--------------------------------------|
| 1 | 0.3849 | 2.345e+00 |
| 2 | 0.4822 | 2.713e+00 |
| 3 | 1.4545 | NA |
| 4 | 0.6905 | 7.878e-02 |
| 5 | 0.5846 | 5.242e-04 |
| 6 | 0.2777 | 1.390e+00 |

| | wildTypePhenotype.sd | 2 ⁻ ddCt | 2 ⁻ ddCt.min | 2 ⁻ ddCt.max |
|---|----------------------|---------------------|-------------------------|-------------------------|
| 1 | 0.3574 | 1.10638851325547 | 8.473e-01 | 1.444684 |
| 2 | 0.2495 | 0.150497255530234 | 1.077e-01 | 0.210221 |
| 3 | NA | + | NA | NA |
| 4 | 0.2813 | 0.700597907024805 | 4.341e-01 | 1.130625 |
| 5 | 0.3689 | 0.909381199520663 | 6.064e-01 | 1.363762 |
| 6 | 0.4576 | 0.457939394245865 | 3.778e-01 | 0.555126 |

TO SHOW EXAMPLE USING GENORM/NORMFINDER DATA

References

- [1] Claus Lindbjerg Andersen, Jens Ledet Jensen and Torben Falck Orntoft (2004). Normalization of Real-Time Quantitative Reverse Transcription-PCR Data: A Model-Based Variance Estimation Approach to Identify Genes Suited for Normalization, Applied to Bladder and Colon Cancer Data Sets CANCER RESEARCH 64, 52455250, August 1, 2004 <http://cancerres.aacrjournals.org/cgi/content/full/64/15/5245>
11

- [2] Kenneth Livak, Thomase Schmittgen (2001). Analysis of Relative Gene Expression Data Using Real-Time Quantitative PCR and the $2^{\Delta\Delta C_t}$ Method. *Methods* 25, 402-408, 2001 <http://www.ncbi.nlm.nih.gov/pubmed/11846609> 16, 17
- [3] Jo Vandesompele, Katleen De Preter, Filip Pattyn, Bruce Poppe, Nadine Van Roy, Anne De Paepe and Frank Speleman (2002). Accurate normalization of real-time quantitative RT-PCR data by geometric averaging of multiple internal control genes. *Genome Biology* 2002, 3(7):research0034.1-0034.11 <http://genomebiology.com/2002/3/7/research/0034/> 5