NormqPCR: Functions for normalisation of RT-qPCR data

James Perkins and Matthias Kohl
University of Malaga (Spain) / Furtwangen University (Germany)

November 1, 2022

Contents

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 houskeeping 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 appearence 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: `Ccl20.Rn00570287_m1` we have these readings for the different samples:

```
> exprs(qPCRBatch.taqman)["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       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)["Ccl20.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)["Ccl20.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 contructed 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 samplenames vector
> contM <- cbind(a,b)
> colnames(contM) <- c("case","control") # set the names of each sample type
> rownames(contM) <- sampleNames(qPCRBatch.taqman) # set row names
> 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, wheras 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 (see Table 3 in Vand02)

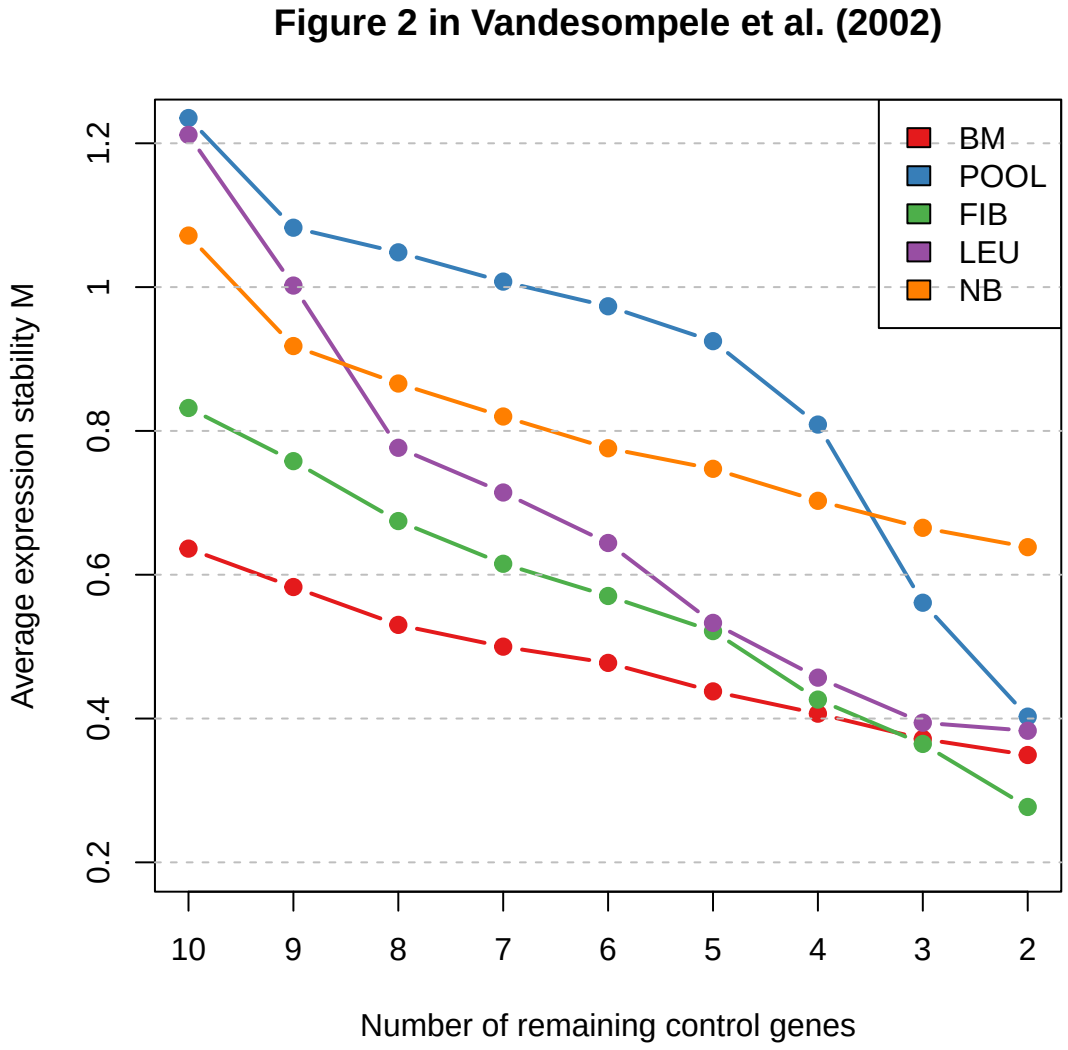
```
> ranks <- data.frame(c(1, 1:9), res.BM$ranking, res.POOL$ranking,
+                     res.FIB$ranking, res.LEU$ranking,
+                     res.NB$ranking)
> names(ranks) <- c("rank", "BM", "POOL", "FIB", "LEU", "NB")
> ranks
```

```
   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 (see Figure 2 in Vand02).

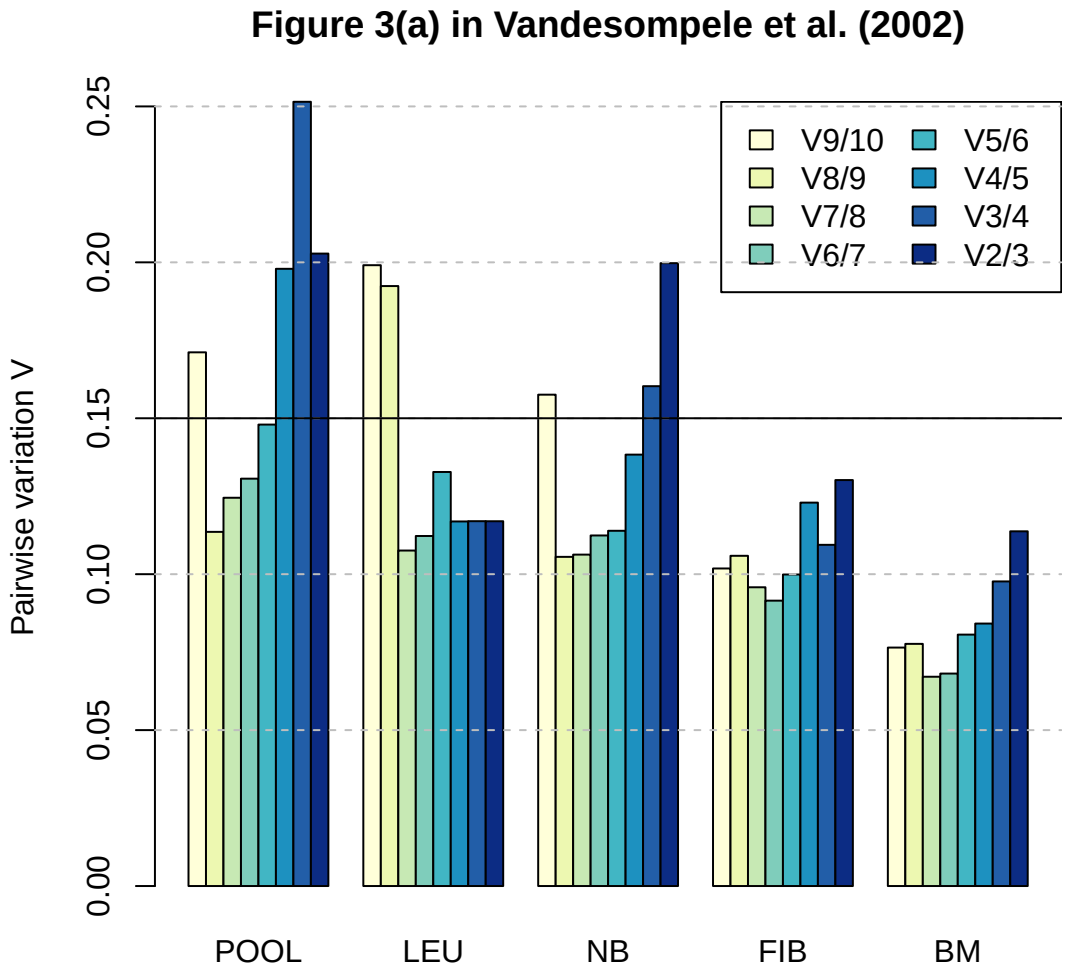
```
> library(RColorBrewer)
> mypalette <- brewer.pal(5, "Set1")
> matplot(cbind(res.BM$meanM, res.POOL$meanM, res.FIB$meanM,
+               res.LEU$meanM, res.NB$meanM), type = "b",
+         ylab = "Average expression stability M",
+         xlab = "Number of remaining control genes",
+         axes = 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)
```



Second, we plot the pairwise variation for each cell type (see 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")
```



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)
> Colon

qPCRBatch (storageMode: lockedEnvironment)
assayData: 13 features, 40 samples
  element names: exprs
protocolData: none
phenoData
  sampleNames: I459N 90 ... I-C1056T (40 total)
  varLabels: Sample.no. Classification
  varMetadata: labelDescription
featureData
  featureNames: UBC UBB ... TUBA6 (13 total)
  fvarLabels: Symbol Gene.name
  fvarMetadata: labelDescription
experimentData: use 'experimentData(object)'
Annotation: Table 1 in Andersen et al. (2004)

> Class <- pData(Colon)[,"Classification"]
> res.Colon <- stabMeasureRho(Colon, group = Class, log = FALSE)
> sort(res.Colon) # see Table 3 in Andersen et al (2004)

      UBC      GAPD      TPT1       UBB     TUBA6     RPS13
0.1821707 0.2146061 0.2202956 0.2471573 0.2700641 0.2813039
     NACA      CFL1      SUI1      ACTB      CLTC     RPS23
0.2862397 0.2870467 0.3139404 0.3235918 0.3692880 0.3784909
 FLJ20030
0.3935173

> data(Bladder)
> Bladder

qPCRBatch (storageMode: lockedEnvironment)
assayData: 14 features, 28 samples
  element names: exprs
protocolData: none
phenoData
```

```
  sampleNames: 335-6 1131-1 ... 1356-1 (28 total)
  varLabels: Sample.no. Grade
  varMetadata: labelDescription
featureData
  featureNames: ATP5B HSPCB ... FLJ20030 (14 total)
  fvarLabels: Symbol Gene.name
  fvarMetadata: labelDescription
experimentData: use 'experimentData(object)'
Annotation: Table 1 in Andersen et al. (2004)
```

```
> grade <- pData(Bladder)[,"Grade"]
> res.Bladder <- stabMeasureRho(Bladder, group = grade,
+                               log = FALSE)
> sort(res.Bladder)
```

```
    HSPCB      TEGT     ATP5B       UBC     RPS23     RPS13
0.1539598 0.1966556 0.1987227 0.2033477 0.2139626 0.2147852
     CFL1  FLJ20030      TPT1       UBB     FLOT2      GAPD
0.2666129 0.2672918 0.2691553 0.2826051 0.2960429 0.3408742
   S100A6      ACTB
0.3453435 0.3497295
```

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

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

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

```
      UBC      GAPD      TPT1       UBB     TUBA6     RPS13
0.1821707 0.2146061 0.2202956 0.2471573 0.2700641 0.2813039
     NACA      CFL1      SUI1      ACTB      CLTC     RPS23
0.2862397 0.2870467 0.3139404 0.3235918 0.3692880 0.3784909
 FLJ20030
0.3935173
     GAPD      TPT1       UBB      NACA      CFL1     RPS13
0.1375298 0.1424519 0.1578360 0.1657364 0.1729069 0.1837057
    TUBA6      SUI1      ACTB     RPS23  FLJ20030      CLTC
0.1849021 0.2065531 0.2131651 0.2188277 0.2359623 0.2447588
     TPT1      NACA       UBB     RPS13      CFL1     TUBA6
0.1108474 0.1299802 0.1356690 0.1411173 0.1474242 0.1532953
 FLJ20030      SUI1      ACTB     RPS23      CLTC
0.1583031 0.1586250 0.1682972 0.1686139 0.1926907
       UBB      TUBA6       ACTB       CFL1      RPS13       SUI1
0.09656546 0.09674897 0.10753445 0.10830099 0.11801680 0.12612399
      CLTC       NACA   FLJ20030      RPS23
0.12773131 0.13422958 0.14609897 0.16530522
     RPS13       SUI1      TUBA6       NACA   FLJ20030       CFL1
0.09085973 0.09647829 0.09943424 0.10288912 0.11097074 0.11428399
      ACTB      RPS23       CLTC
0.11495336 0.12635109 0.13286210
      ACTB      TUBA6       CFL1   FLJ20030       NACA       CLTC
0.09215478 0.09499893 0.09674032 0.10528784 0.10718604 0.10879846
      SUI1      RPS23
0.11368091 0.13134766
      SUI1       NACA   FLJ20030      RPS23      TUBA6       CFL1
0.08281504 0.08444905 0.08922236 0.09072667 0.10559279 0.10993755
      CLTC
0.13142181
      NACA       CFL1      TUBA6   FLJ20030       CLTC      RPS23
0.08336046 0.08410148 0.09315528 0.09775742 0.10499056 0.10554332
```

```
      CFL1      TUBA6       CLTC   FLJ20030      RPS23
0.07222968 0.07722737 0.08440691 0.09831958 0.12735605
  FLJ20030      TUBA6       CLTC      RPS23
0.08162006 0.08189011 0.10705192 0.11430674
     TUBA6       CLTC      RPS23
0.06978897 0.08069582 0.13702726
     CLTC     RPS23
0.1199009 0.1245241
         1          2          3          4          5          6
     "UBC"     "GAPD"     "TPT1"      "UBB"    "RPS13"     "ACTB"
         7          8          9         10         11         12
    "SUI1"     "NACA"     "CFL1" "FLJ20030"    "TUBA6"     "CLTC"
```

In case of the `Bladder` dataset the two top ranked genes are HSPCB and RPS13; see Figure 1 in And04.

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

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

5 Normalization by means of reference/housekeeping genes

5.1 ΔCq method using a single housekeeper

The ΔCq 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 postive values only, `2^` 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:

```
> hkgs<-c("Actb-Rn00667869_m1", "B2m-Rn00560865_m1", "Gapdh-Rn99999916_s1")
> 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   -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 $2^{-\Delta\Delta Cq}$ method using a single housekeeper

It is possible to use the $2^{-\Delta\Delta Cq}$ 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, contrastM=contM, case="interestingPhenotype",
+                             control="wildTypePhenotype", statCalc="geom", hkgCalc="arith")
> 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,
+                                hkg=hkg, contrastM=contM, case="interestingPhenotype",
+                                control="wildTypePhenotype", paired=FALSE, statCalc="geom",
+                                hkgCalc="arith")
> 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)
> hkgs<-c("Actb-Rn00667869_m1", "B2m-Rn00560865_m1", "Gapdh-Rn99999916_s1")
> ddCq.gM.taqman <- deltaDeltaCq(qPCRBatch = qPCRBatch.taqman, maxNACase=1, maxNAControl=1,
+                                hkgs=hkgs, contrastM=contM, case="interestingPhenotype",
+                                control="wildTypePhenotype", statCalc="arith", hkgCalc="ari
> 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)
> hkgs<-c("Actb-Rn00667869_m1", "B2m-Rn00560865_m1", "Gapdh-Rn99999916_s1")
> ddAvgCq.gM.taqman <-deltaDeltaCq(qPCRBatch = qPCRBatch.taqman, maxNACase=1, maxNAControl=1
+                                  hkgs=hkgs, contrastM=contM, case="interestingPhenotype",
+                                  control="wildTypePhenotype", paired=FALSE, statCalc="arit
+                                  hkgCalc="arith")
> 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

5.5 Compute NRQs

THIS FUNCTION IS STILL EXPERIMENTAL!

We load a dataset including technical replicates.

```
> path <- system.file("exData", package = "ReadqPCR")
> qPCR.example <- file.path(path, "qPCR.example.txt")
> Cq.data <- read.qPCR(qPCR.example)
```

We combine the technical replicates and in addition compute standard deviations.

```
> Cq.data1 <- combineTechRepsWithSD(Cq.data)
```

We load efficiencies for the dataset and add them to the dataset.

```
> Effs <- file.path(path, "Efficiencies.txt")
> Cq.effs <- read.table(file = Effs, row.names = 1, header = TRUE)
> rownames(Cq.effs) <- featureNames(Cq.data1)
> effs(Cq.data1) <- as.matrix(Cq.effs[,"efficiency",drop = FALSE])
> se.effs(Cq.data1) <- as.matrix(Cq.effs[,"SD.efficiency",drop = FALSE])
```

Now we can compute normalized relative quantities for the dataset where we consider two of the included features as reference/housekeeping genes.

```
> res <- ComputeNRQs(Cq.data1, hkgs = c("gene_az", "gene_gx"))
> ## NRQs
> exprs(res)
```

```
            caseA     caseB  controlA  controlB
gene_ai 1.9253072 1.3586729 0.6479659 0.8749479
gene_az 1.0567118 1.1438982 1.0331980 0.9134997
gene_bc 1.1024935 0.7193500 0.7030487 1.2140836
gene_by 1.5102316 0.9573047 0.7527082 1.6008850
gene_dh 1.2982037 1.0722522 0.9623335 0.9392871
gene_dm 0.6590246 1.1690720 1.2475372 0.9366210
gene_dq 0.7541955 0.7036408 0.8327917 1.6165326
```

```
gene_dr 2.2192305 1.0581211 0.7026411 0.6900584
gene_eg 0.9366671 0.5800339 0.8313720 1.1848856
gene_er 0.5269062 0.9375427 0.6953326 2.3195978
gene_ev 1.4622280 2.3457021 0.9038912 1.1454535
gene_fr 1.4954763 1.6200792 0.9641192 0.7295680
gene_fw 0.6944248 0.8051075 1.5698382 0.7978611
gene_gx 0.9463318 0.8742037 0.9678687 1.0946911
gene_hl 1.0009372 1.4015267 0.7683665 0.7713712
gene_il 1.4632019 1.2595559 0.7216891 0.9318860
gene_iv 1.7263335 1.2275001 1.5464212 0.8881605
gene_jr 0.8984351 0.9834026 0.8754813 0.6637941
gene_jw 1.4655948 0.9340184 1.0505200 1.5504136
gene_qs 0.6730225 0.7610418 1.0665938 3.5329891
gene_qy 0.5287127 1.5722670 1.0615326 3.3252907
gene_rz 0.8690600 1.5588299 0.7287288 1.4812753
gene_sw 0.5975288 1.2406438 0.6982954 1.6007333
gene_vx 0.6942254 0.7168408 2.0253177 1.3190943
gene_xz 0.7668030 1.0218209 0.6136038 1.6729352

> ## SD of NRQs
> se.exprs(res)

            caseA     caseB  controlA  controlB
gene_ai 1.3996554 0.8787290 0.4855882 1.0034912
gene_az 0.6832730 0.7601966 0.8971054 0.6031927
gene_bc 0.7225348 0.4746146 0.9626570 1.0478385
gene_by 1.1522746 0.6116269 0.6088836 2.0409211
gene_dh 1.2483072 0.7889984 0.6165041 0.9767947
gene_dm 0.4711409 0.7780238 0.8476053 0.7294405
gene_dq 0.7023561 0.4849899 0.5813310 1.4067670
gene_dr 1.4407662 1.0804211 0.4543153 0.5149367
gene_eg 0.7355269 0.5497433 0.5588801 1.0938601
gene_er 0.4301195 0.6119514 0.4471454 1.5115897
gene_ev 1.0094209 2.4267114 0.6337126 0.7782519
gene_fr 1.6760391 1.1119157 0.6226081 0.5040967
gene_fw 0.5041070 0.9131565 1.1153268 0.7234551
gene_gx 0.6046042 0.9027816 0.6713914 1.7394961
gene_hl 0.7633174 0.9123997 1.0000329 0.5005813
gene_il 1.4621406 0.9540445 0.5678634 0.6067147
gene_iv 1.2668346 0.8039841 0.9995225 0.8171996
gene_jr 0.5749672 0.6786989 0.5595295 0.4405919
```

```
gene_jw 0.9626606 0.7890401 0.7194378 1.1512512
gene_qs 0.5830335 0.5309990 0.6828952 2.8253799
gene_qy 0.5947918 1.1294199 0.6794829 2.1713340
gene_rz 0.5846751 1.7926435 0.4911506 2.1424580
gene_sw 0.6284440 0.8062083 0.9638307 1.8398593
gene_vx 0.5285361 1.0126959 1.3861226 0.8683886
gene_xz 0.5231477 0.9270275 0.3972901 1.3643840
```

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 $2^{\Delta\Delta Ct}$ 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 averiging of multiple internal control genes. Genome Biology 2002, 3(7):research0034.1-0034.11 http://genomebiology.com/2002/3/7/research/0034/ 5

[4] 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