R/Bioconductor
Center for Genomic Sciences
Universidad Nacional Autónoma de México

Daniela Azucena García Soriano, dgarcia@lcg.unam.mx
Yuvia Alhelí Pérez Rico, yperez@lcg.unam.mx

October 23, 2009

Abstract

HTqPCR - high-throughput qPCR analysis in R and Bioconductor

1 What is HTqPCR for?

Introduction

The package HTqPCR is designed for the analysis of cycle threshold (Ct) values from quantitative real-time PCR data. The main areas of functionality comprise data import, quality assessment, normalisation, data visualisation, and testing for statistical significance in Ct values between different features (genes, miRNAs).

```
> library("HTqPCR")
```

Introduction

The package employs functions from other packages of the Bioconductor project ref:bioc. Dependencies include Biobase, RColorBrewer, limma, statmod, affy and gplots.

Two qPCRset test objects are included in the package: one containing raw data, and the other containing processed values.

```
> data(qPCRraw)
> data(qPCRpros)
> class(qPCRraw)

[1] "qPCRset"
attr(,"package")
[1] ".GlobalEnv"
```

2 Reading in the raw data

General data format

The input consists of tab-delimited text files containing the Ct values for a range of genes.

The package comes with example input files (from Applied Biosystem's TLDA cards), along with a text file listing sample file names and biological conditions.

```
> path <- system.file("exData", package = "HTqPCR")
> head(read.delim(file.path(path, "files.txt")))
```

```
         File  Treatment
1 sample1.txt    Control
2 sample2.txt LongStarve
3 sample3.txt LongStarve
4 sample4.txt    Control
5 sample5.txt     Starve
6 sample6.txt     Starve
```

```
> files <- read.delim(file.path(path, "files.txt"))
> raw <- readCtData(files = files$File, path = path)
```

.The qPCRset object looks like:

```
> show(raw)
```

```
An object of class "qPCRset"
Size:  384 features, 6 samples
Feature types:                 Endogenous Control, Target
Feature names:                 Gene1 Gene2 Gene3 ...
Feature classes:
Feature categories:         OK, Undetermined
Sample names:                sample1 sample2 sample3 ...
```

widthheightSchunk

```
SDS 2.3 RQ Results        1.2
 Filename        Testscreen analys all.sdm
 Assay Type        RQ Study
 EmbeddedFile        FileA
 Run DateTime        Fri May 15 17:10:28 BST 2009
 Operator
 ThermalCycleParams
 EmbeddedFile        FileB
 Run DateTime        Sat May 16 10:36:09 BST 2009
 Operator
 ThermalCycleParams
 EmbeddedFile        FileC
 Run DateTime        Sun May 17 13:21:05 BST 2009
 Operator
 ThermalCycleParams

 #          Plate        Pos        Flag        Sample        Detector        Task        Ct
 1        Control        A1        Passed        Sample01        Gene1        Endogenous Control
 2        Control        A2        Passed        Sample01        Gene2        Target        33.94

 [1] "SDS 2.3 RQ Results\t1.2"
 [2] "Filename\tTestscreen analys all.sdm"
 [3] "Assay Type\tRQ Study"
 [4] "EmbeddedFile\tFileA"
```

```
 [5] "Run DateTime\tFri May 15 17:10:28 BST 2009"
 [6] "Operator\t"
 [7] "ThermalCycleParams\t"
 [8] "EmbeddedFile\tFileB"
 [9] "Run DateTime\tSat May 16 10:36:09 BST 2009"
[10] "Operator\t"
[11] "ThermalCycleParams\t"
[12] "EmbeddedFile\tFileC"
[13] "Run DateTime\tSun May 17 13:21:05 BST 2009"
[14] "Operator\t"
[15] "ThermalCycleParams\t"
[16] ""
[17] "#  \tPlate\tPos\tFlag\tSample\tDetector\tTask\tCt\tDelta Ct\tAvg Delta Ct\t?Ct SE\tDelta De
[18] "1\tControl\tA1\tPassed\tSample01\tGene1\tEndogenous Control\t11.463166\t\t\t\t\t\t\tFALSE
[19] "2\tControl\tA2\tPassed\tSample01\tGene2\tTarget\t33.949196\t\t22.479778\t0.26758063\t0\t1\t
[20] "3\tControl\tA3\tPassed\tSample01\tGene3\tTarget\t27.956657\t\t16.195972\t0.14037517\t0\t1\t
```

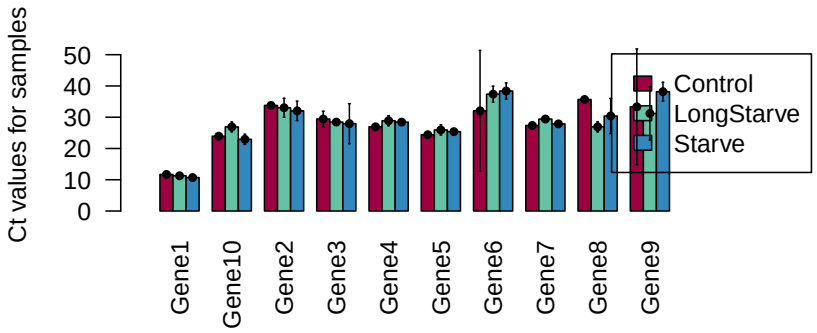
3 Data visualisation

Overview of Ct values across all groups

To get a general overview of the data the (average) Ct values for a set of features across all samples or different condition groups can be displayed.

widthheightSchunk

```
> par(mfrow = c(2, 1))
> g <- featureNames(raw)[1:10]
> plotCtOverview(raw, genes = g, xlim = c(0, 50), groups = files$Treatment,
+     conf.int = TRUE, ylim = c(0, 55))
```

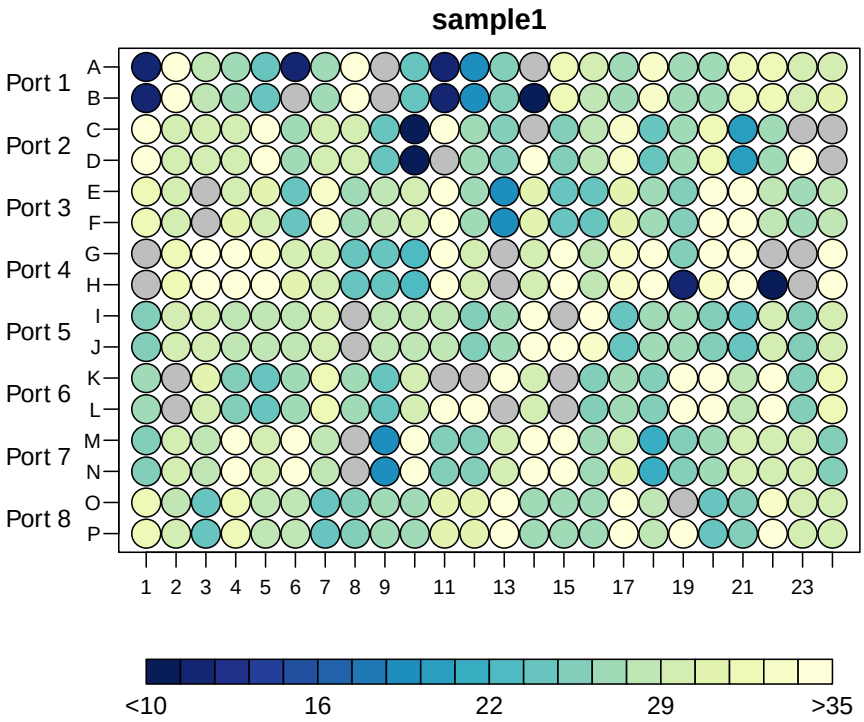


Spatial layout

When the features are organised in a particular spatial pattern, it is possible to plot the Ct values or other characteristics of the features using this layout.

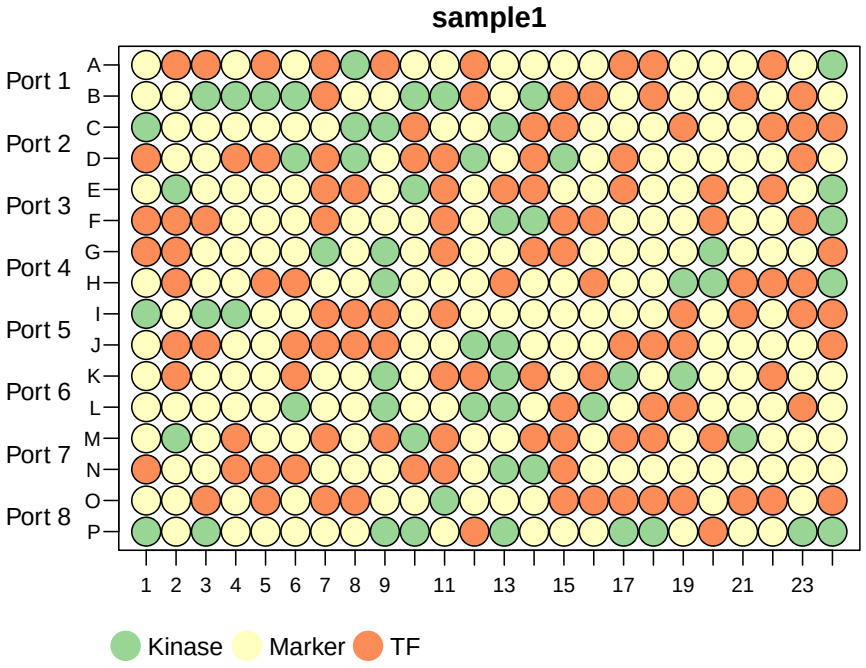
widthheightSchunk

```
> plotCtCard(raw, col.range = c(10, 35), well.size = 2.6)
> featureClass(raw) <- factor(c("Marker", "TF", "Kinase")[sample(c(1,
+     1, 2, 2, 1, 3), 384, replace = TRUE)])
```



widthheightSchunk

```
> plotCtCard(raw, plot = "class", well.size = 2.6)
```



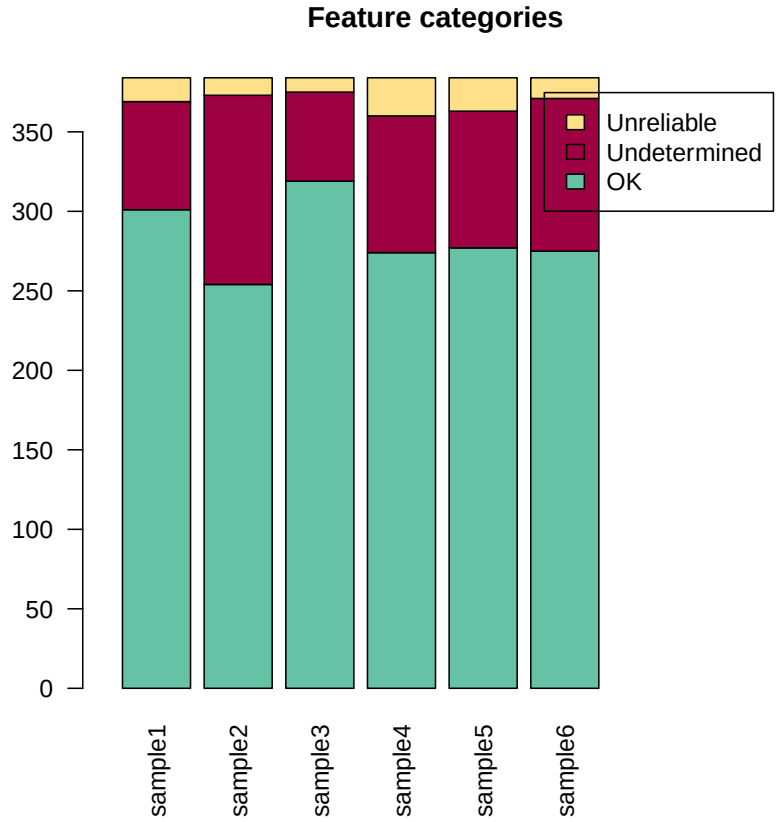
4 Feature categories and filtering

Each Ct values in HTqPCR has an associated feature category. This is an important component to indicate the reliability of the qPCR data. Aside from the "OK" indicator, there are two other categories: "Undetermined" is used to flag Ct values above a user-selected threshold, and "Unreliable" indicates Ct values that are either so low as to be estimated by the user to be problematic, or that arise from deviation between individual Ct values across replicates.

widthheightSchunk

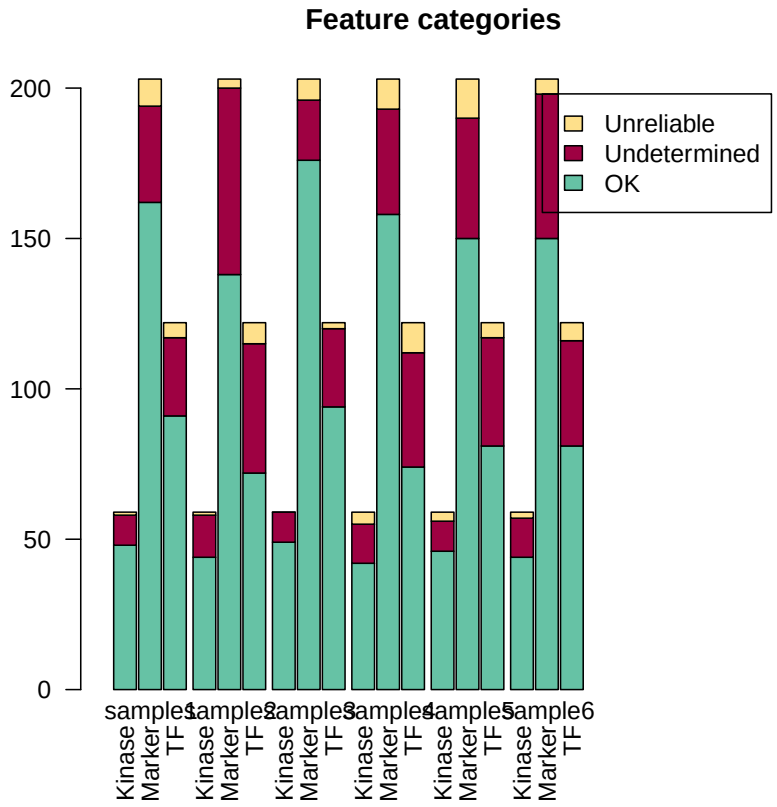
```
> raw.cat <- setCategory(raw, groups = files$Treatment, quantile = 0.8)

Categories after Ct.max and Ct.min filtering:
             sample1 sample2 sample3 sample4 sample5 sample6
OK               313     264     327     295     296     286
Undetermined      68     119      56      86      86      96
Unreliable         3       1       1       3       2       2
Categories after standard deviation filtering:
             sample1 sample2 sample3 sample4 sample5 sample6
OK               301     254     319     274     277     275
Undetermined      68     119      56      86      86      96
Unreliable        15      11       9      24      21      13
```

```
> plotCtCategory(raw.cat)
```

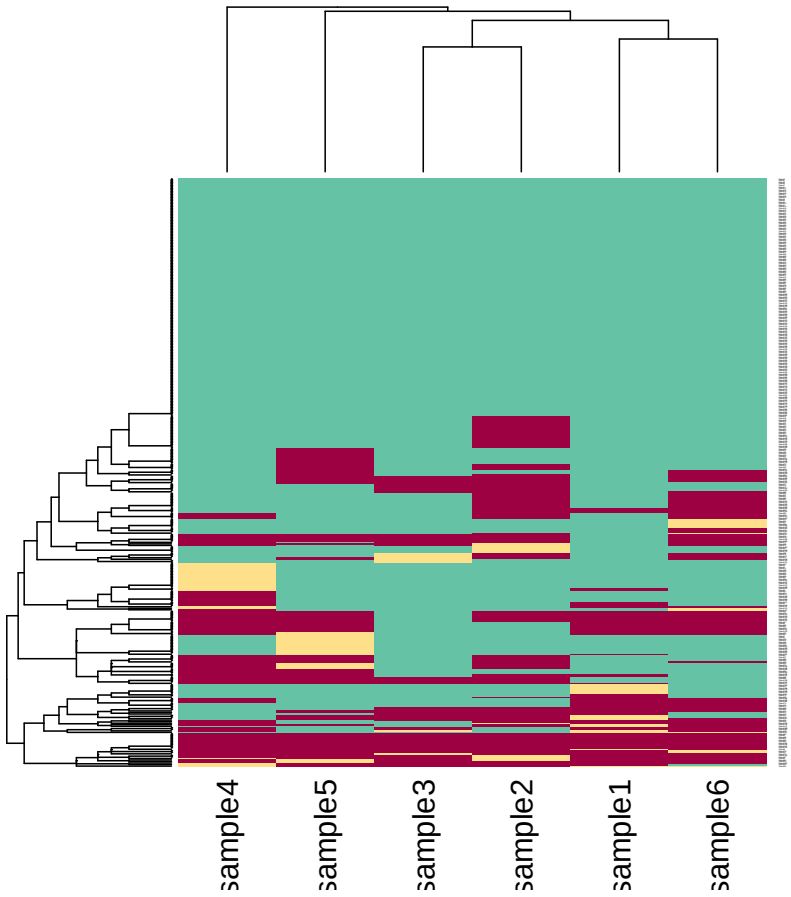


widthheightSchunk

```
> plotCtCategory(raw.cat, stratify = "class")
```




```
> plotCtCategory(raw.cat, by.feature = TRUE, cexRow = 0.1)
```



5 Data Normalisation

Normalisation

Four different normalisation methods are currently implemented in HTqPCR. Two of these.

quantile Will make the distribution of Ct values more or less identical across samples.

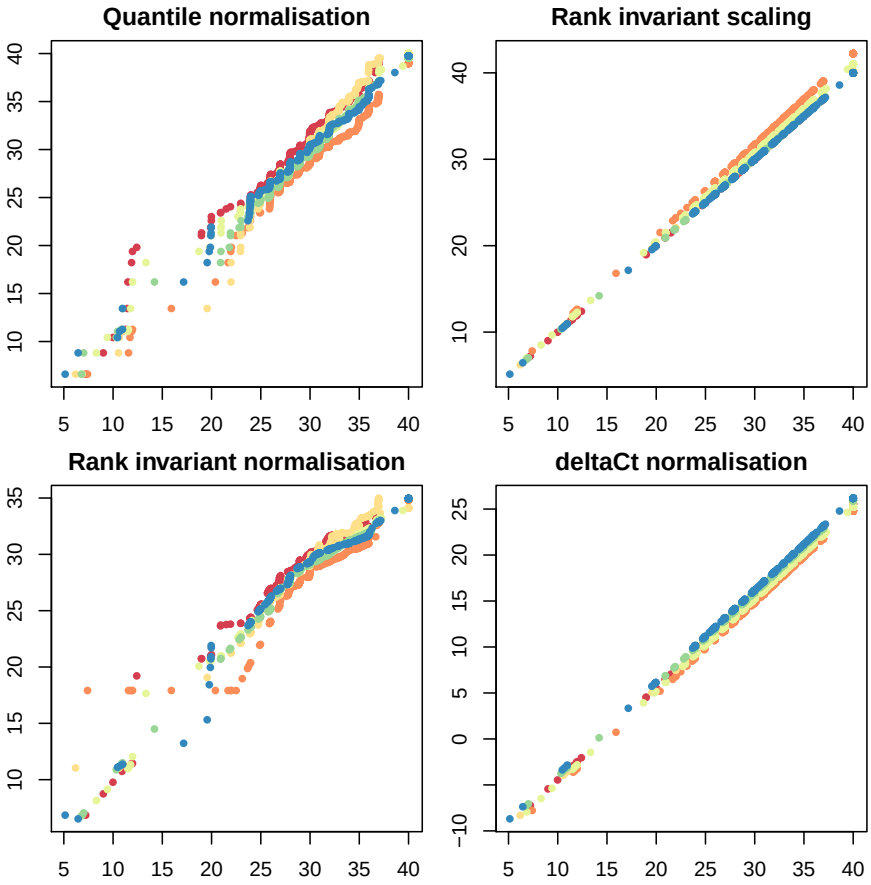
norm.rankinvariant Computes all rank-invariant sets of features between pairwise comparisons of each sample against a reference, such as a pseudo-mean. The rank-invariant features are used as a reference for generating a smoothing curve, which is then applied to the entire sample.

scale.rankinvariant Also computes the pairwise rank-invariant features, but then takes only the features found in a certain number of samples, and used the average Ct value of those as a scaling factor for correcting all Ct values.

deltaCt Calculates the standard deltaCt values, i.e. subtracts the mean of the chosen controls from all other values in the feature set.

Sources of biological sequences

```
> q.norm <- normalizeCtData(raw.cat, norm = "quantile")
> sr.norm <- normalizeCtData(raw.cat, norm = "scale.rank")

Scaling Ct values
        Using rank invariant genes: Gene1 Gene29
        Scaling factors: 1.00 1.06 1.00 1.03 1.00 1.00

> nr.norm <- normalizeCtData(raw.cat, norm = "norm.rank")

Normalizing Ct values
        Using rank invariant genes:
        sample1: 75 rank invariant genes
        sample2: 33 rank invariant genes
        sample3: 48 rank invariant genes
        sample4: 69 rank invariant genes
        sample5: 18 rank invariant genes
        sample6: 67 rank invariant genes

> d.norm <- normalizeCtData(raw.cat, norm = "deltaCt", deltaCt.genes = c("Gene1",
+     "Gene60"))

Calculating deltaCt values
        Using control gene(s): Gene1 Gene60
        Card 1:         Mean=14.45        Stdev=4.25
        Card 2:         Mean=15.19        Stdev=5.27
        Card 3:         Mean=14.50        Stdev=5.8
        Card 4:         Mean=14.79        Stdev=4.79
        Card 5:         Mean=14.07        Stdev=5.32
        Card 6:         Mean=13.82        Stdev=4.75

> col <- rep(brewer.pal(6, "Spectral"), each = 384)
> par(mfrow = c(2, 2), mar = c(2, 2, 2, 1))
> plot(exprs(raw), exprs(q.norm), pch = 20, main = "Quantile normalisation",
+     col = col)
> plot(exprs(raw), exprs(sr.norm), pch = 20, main = "Rank invariant scaling",
+     col = col)
> plot(exprs(raw), exprs(nr.norm), pch = 20, main = "Rank invariant normalisation",
+     col = col)
> plot(exprs(raw), exprs(d.norm), pch = 20, main = "deltaCt normalisation",
+     col = col)
```



6 Filtering and subsetting the data

Filtering and subsetting the data

At any point during the analysis it's possible to filter out both individual features or groups of features that are either deemed to be of low quality, or not of interest for a particular aspect of the analysis. This can be done using any of the feature characteristics that are included in the featureNames, featureType, featureClass and/or featureCategory slots of the data object. Likewise, the qPCRset object can be turned into smaller subsets, for example if only a particular class of features are to be used, or some samples should be excluded.

```
> nr.norm[1:10, ]

An object of class "qPCRset"
Size:  10 features, 6 samples
Feature types:                 Endogenous Control, Target
Feature names:                 Gene1 Gene2 Gene3 ...
Feature classes:         Kinase, Marker, TF
Feature categories:         OK, Unreliable, Undetermined
Sample names:                sample1 sample2 sample3 ...
```

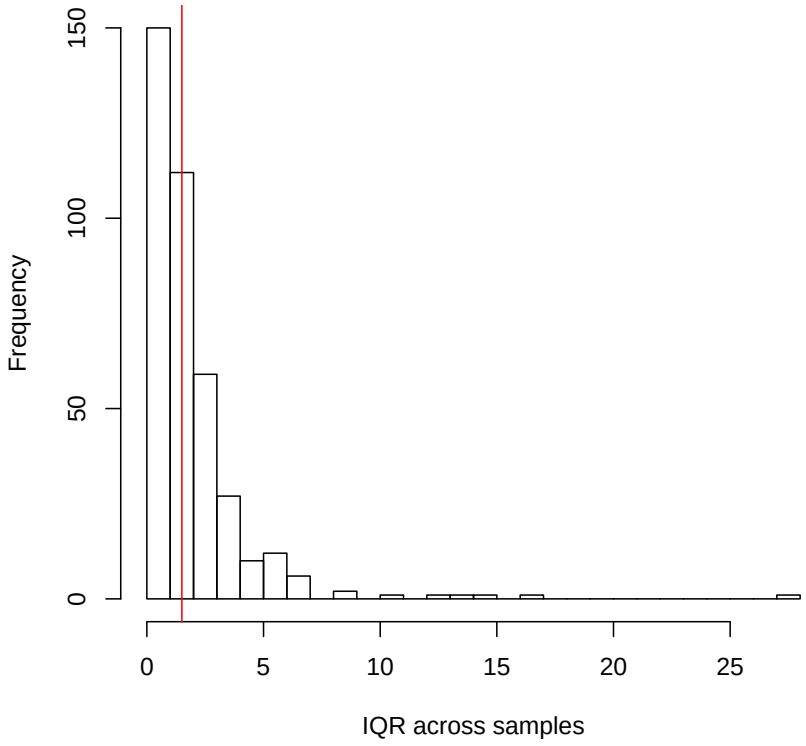
```
> nr.norm[, c(1, 3, 5)]

An object of class "qPCRset"
Size:  384 features, 3 samples
Feature types:                 Endogenous Control, Target
Feature names:                 Gene1 Gene2 Gene3 ...
Feature classes:         Kinase, Marker, TF
Feature categories:         OK, Unreliable, Undetermined
Sample names:                sample1 sample3 sample5 ...
```

widthheightSchunk

```
> qFilt <- filterCtData(nr.norm, remove.type = "Endogenous Control")

Removed 4 'Endogenous Control' features based on featureType(q).

> qFilt <- filterCtData(nr.norm, remove.name = c("Gene1", "Gene20",
+     "Gene30"))

Removed 8 'Gene1/Gene20/Gene30' features based on featureNames(q).

> qFilt <- filterCtData(nr.norm, remove.class = "Kinase")

Removed 59 'Kinase' features based on featureClass(q).

> qFilt <- filterCtData(nr.norm, remove.type = c("Endogenous Control"),
+     remove.name = c("Gene1", "Gene20", "Gene30"))

Removed 4 'Endogenous Control' features based on featureType(q).
Removed 4 'Gene1/Gene20/Gene30' features based on featureNames(q).

Removed 64 features with >3 'Undetermined' based on featureCategory(q).

Removed 10 features with >5 'Undetermined' based on featureCategory(q).

Removed 207 features with IQR <1.5 based on exprs(q).
```

widthheightSchunk

```
> hist(apply(exprs(nr.norm), 1, IQR), n = 20, main = "", xlab = "IQR across samples")
> abline(v = 1.5, col = 2)
```



7 Quality assesment

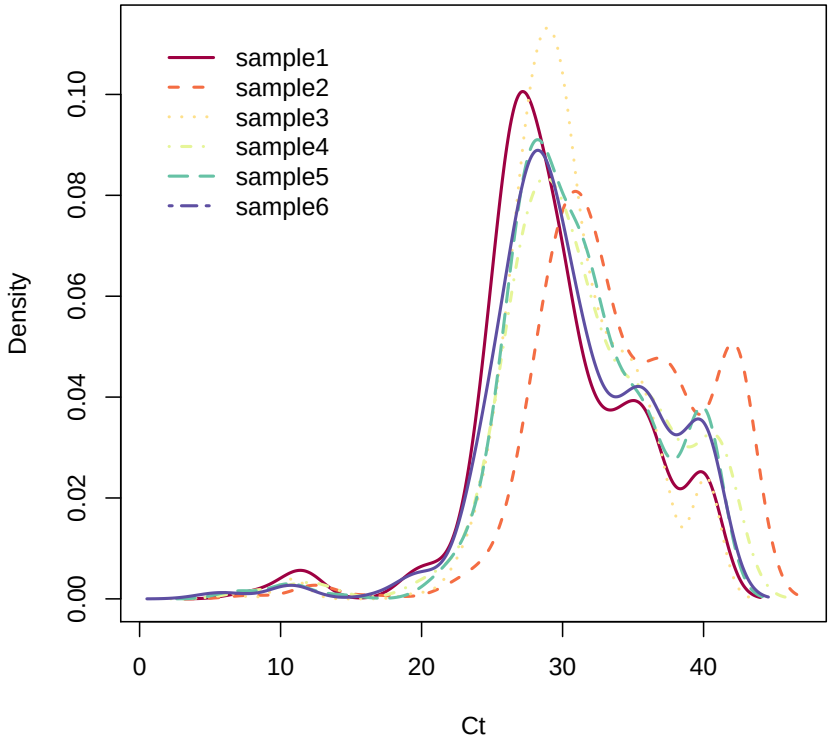
Quality assesment

The overall correlation between different samples can be displayed visually.

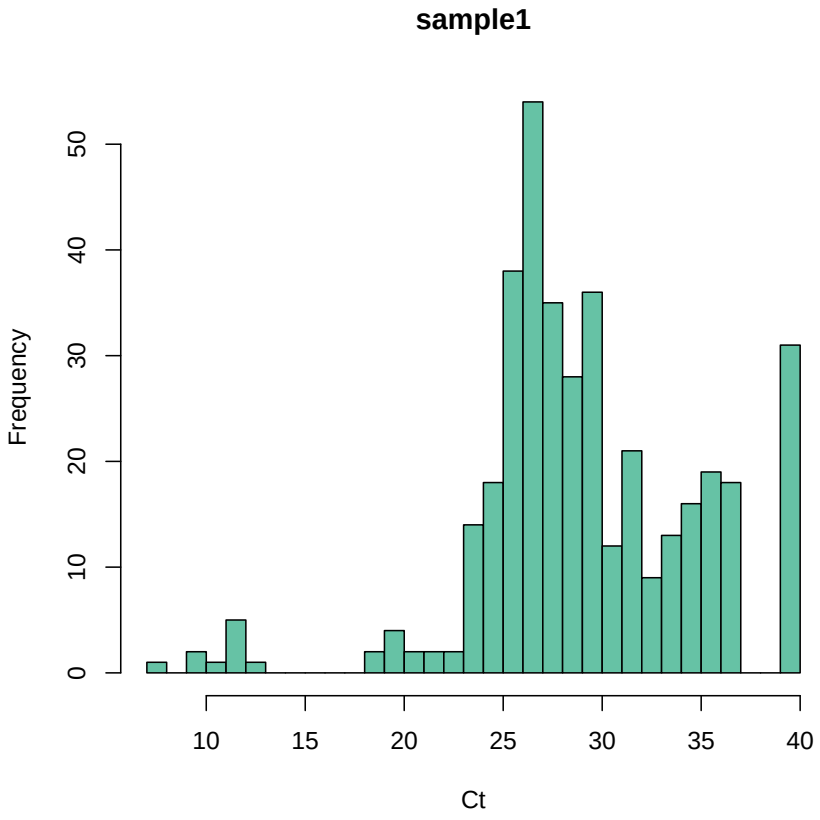
Distribution of Ct values

It may be of interest to examine the general distribution of data both before and after normalisation.

```
> plotCtDensity(sr.norm)
```




```
> plotCtHistogram(sr.norm)
```



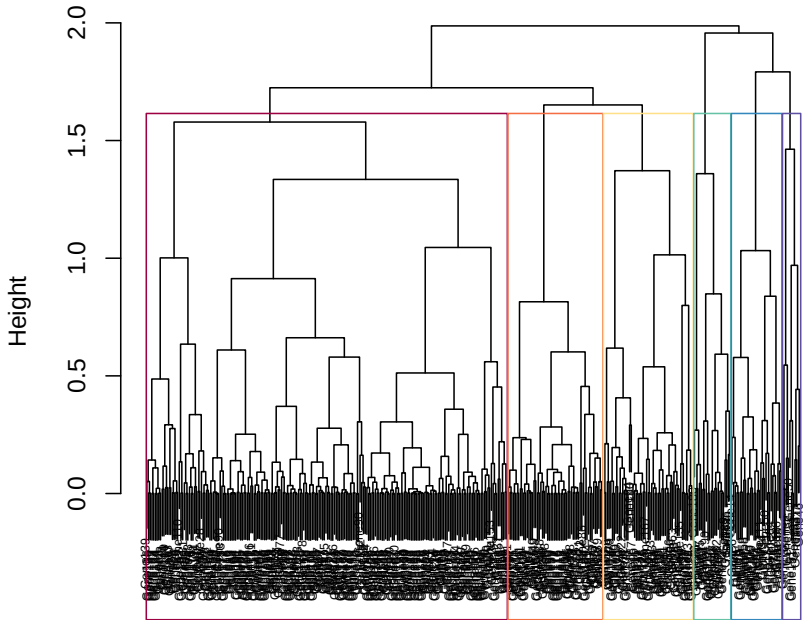
8 Hierarchical clustering

Hierarchical clustering

Both features and samples can be subjected to hierarchical clustering using either Euclidean or Pearson correlation distances, to display similarities and differences within groups of data. Individual subclusters can be selected, either using pre-defined criteria such as number of clusters, or interactively by the user. The content of each cluster is then saved to a list, to allow these features to be extracted from the full data set if desired.

widthheightSchunk

```
> cluster.list <- clusterCt(sr.norm, type = "genes", n.cluster = 6,
+     cex = 0.5)
```



Cluster dendrogram