Analyse RT–PCR data with *ddCt*

Jitao David Zhang, Rudolf Biczok and Markus Ruschhaupt

October 26, 2021

Abstract

Quantitative real–time PCR (qRT–PCR or RT–PCR for short) is a laboratory technique based on the polymerase chain reaction, and is commonly used to amplify and quantify a targeted nucleotide molecule simultaneously. The data analysis of qRT–PCR experiments can be divided in subsequent steps: importing the data, setting reference sample(s) and housekeeping gene(s), (optional) filtering, applying algorithm for the relative expression, and finally reporting the results in the form of text and figure. *ddCt* package implements the $2^{-\Delta\Delta C_T}$ algorithm in a pipeline as described above and performs end–to–end analysis of qRT-PCR experiments in R and Bioconductor. In this vignette we introduce the usage of the pipeline[1].

This vignette introduces the analysis of RT–PCR data with *ddCt* by a step–to–step guide through the pipeline. The idea is that the user could read the vignette, modify the parameters to her/his need, and then use the `Sweave` function in R to perform the analysis. The `Sweave` function converts the '.Rnw' version into a LaTeX file while processing the RT–PCR data in the background. It gives the same result as if the user invokes the pipeline in the way described in the section 3.1.

The vignette begins with a short description of the values to be calculated by the *ddCt*. Then it explains the paramters to be specified by the user in depth.

1 Introduction

Several values are calculated during pipeline by Sweaving this document. The most important ones are explained here in the order of the execution. To simply the discussion, the values discussed are calculated without efficiencies (see below).

1. The mean or median of the technical replicates for one gene–sample pair combination is the C_T value.

2. For a sample A the C_T value of the housekeeping gene (or the median of the C_T values of all housekeeping genes) is subtracted from the corresponding C_T value of a gene $Gene1$. The result is the dC_T value for $Gene1$ and sample A. This value is used for the *t-test* and the *Wilcoxon test*.

[1] For more general information about qRT–PCR and the $2^{-\Delta\Delta C_T}$ algorithm, please follow the another vignette *rtPCR* released along the package *ddCt*.

3. For a gene *Gene*1 the dC_T value of the reference sample (or the mean of the dC_T values of all reference samples) is subtracted from the corresponding dC_T value of *Gene*1 and a sample B. This is called the ddC_T value for *Gene*1 and sample B.

4. The transformation $x \rightarrow 2^{-x}$ is applied to each ddC_T value. The resulting value is called 'exprs' (used to be named as 'level', these two words are exchangable in this document).

Additionally, for each value an error is calculated. This is based on the standard error of the mean (*S.E.M*) if the mean is used to summarize the replicates. If the median is used for summarization of the individual Ct values, the MAD[2] divided by the square root of replicate number is used as an error estimate.

2 Prerequisites

The package *ddCt* requires the latest version of R to perform properly. Please check the DESCRIPTION file along the package for the requirement of packages.

To run this document and perform the analysis, you need a tab-delimited plain text file containing the individual Ct values of your experiment. This file is usually exported from the software used to measure the Ct values, for example from SDM©Software of Roche Coop.. It is important that you export the individual Ct values, since normally the software combines the replicates of the measurements to one value. See the file *extdata/Experiment1.txt* in the directory of the *ddCt* package for an example.

Once the file is ready you can go on to set parameters.

3 Using *ddCt*

3.1 Invoke the pipeline

To use the functionality of the *ddCt* package, one can

- Writting R scripts with the functions implemented in the *ddCt*. It requires basic programming skills but provides the maximum flexibility. See the vignette *rtPCR* for a short example and the manual pages of the functions in the package.

- Calling the *ddCt* script in the *scripts* sub–directory in the *ddCt* installation package. This script can be invoked by either typing in the command line, or by modifying the parameters in this vigenette and then run the `Sweave` function. Actually this vignette is just a wrapper of the *ddCt* script which describes the parameters. In the vigette we discuss this approach.

When using the *ddCt* script, One can call the *ddCt* script in the *scripts* sub–directory in the *ddCt* installation package (*PKG_DIR* in the following samples) via the *Rscript* command in the command line[3]:

[2]median absolute deviation, defined as $MAD = median|x_i - \tilde{x}|$, where $\tilde{x}$ is the median of x. Its relationship with standard deviation σ can be expressed as $\sigma \approx 1.4826MAD$

[3]The examples shown here are for Debian Linux system, on the windows system the path separator has to be modified, however the functionality is the same

```
Rscript PKG_DIR/scripts/ddCt.R -inputFile="PKG_DIR/extdata/Experiment1.txt" ....
```

This command-line approach does not need to invoke a R session and is easy to be built into automatic analysis pipelines. It is however hard to know all the possible parameters and not easy to be used by users who are not familiar with the command line.

Alternatively, one could load the script via the source function in the R prompt. If you do so, you have to pass the parameters as a list to the `ddCtExec` function, which is an end–to–end function combining data import, analysis and report:

```
> scFile <- system.file("scripts/ddCt.R", package="ddCt")
> inputFile <- system.file("extdata/Experiment1.txt", package="ddCt")
> source(scFile)
> ddCtExec(list(inputFile=inputFile, ...))
```

The parameters for the ddCtExec function are all equal with parameters in the Rscript call. In this example, we use the second way to invoke the *ddCt* package.

```
> source(system.file("scripts", "ddCt.R", package="ddCt"))
```

3.2 Parameters

Table 1 on the page 3 illustrates the usable parameters in the ddCt.R script. Not all parameters have to be set or they already have default values. Only `inputFile`, `referenceGene` and `referenceSample` are required parameters.

Table 1: List of parameters

Parameter	Description	Required	Default
inputFile	a valid SDS input file	Yes	`NULL`
referenceGene	houskeeping gene to use	Yes	`NULL`
referenceSample	calibration sample to use	Yes	`NULL`
loadPath	directory, from which the input files are loaded	No	Working directory
savePath	output directory	No	Working directory
confFile	load parameters from this R script	No	`NULL`
sampleAnnotationFile	optional annotation file	No	`NULL`
columnForGrouping	columns used to group output	No	`NULL`
onlyFromAnnotation	only use annotated samples	No	`FALSE`
geneAlias	replace specified gene names	No	`NULL`
sampleAlias	replace specified sample names	No	`NULL`
threshold	threshold value	No	40
mode	the used algorithm mode	No	median

Continued on next page

Table 1 – continued from previous page

Parameter	Description	Required	Default
plotMode	the kind of values which schould be ploted (Ct, level, ...)	No	c("level","Ct")
algorithm	use ddCt or ddCtWihtE algorithm	No	ddCt
efficiencies	efficience vaue for each gene	No	NULL
efficienciesError	error value	No	NULL
genesRemainInOutput	show only these specified genes in output	No	NULL
samplesRemainInOutput	show only these specified genessamples in output	No	NULL
genesNotInOutput	dont show these specified genes in output	No	NULL
samplesNotInOutput	dont show these specified genessamples in output	No	NULL
groupingBySamples	plot output by	No	TRUE
plotPerObject	generate one plot per kind of value (Ct, level, ...)	No	TRUE
groupingForPlot	group also in plot	No	NULL
groupingColor	The color for each bar	No	c("#E41A1C","#377EB8","#4DAF4A", "#984EA3","#FF7F00")
cutoff	cutoff value	No	NULL
brewerColor	color for the brewer	No	c("Set3","Set1","Accent", "Dark2","Spectral","PuOr","BrBG")
legend	show legend	No	TRUE
ttestPerform	perform a TTest	No	FALSE
ttestCol	color of the TTest	No	NULL
pairsCol	color of the pairs	No	NULL
samplesRemainInTTest	use this samples for TTest	No	NULL
samplesNotInTTest	dont use this samples for TTest	No	NULL
samplesRemainInCor	use this samples for correlation plot	No	NULL
samplesNotInCor	dont use this samples for correlation plot	No	NULL

4 Setting parameters

4.1 Input and output directories

First you have to specify the directory where your data is stored (`loadPath`) and the directory where the results are supposed to be stored (`savePath`). You can specify the path relatively to the directory you are in when starting the script in R.

```
> params <- list(loadPath = system.file("extdata", package="ddCt"),
+                savePath = getwd())
```

4.2 Input files

Then you have to specify the name of the exported file from the RT–PCR device containing the individual Ct values (`inputFile`). Optionally, you may specify a file containing additional annotation data related to the samples (`sampleAnnotationFile`). Refer to the file 'SampleAnno.txt' released along the package as an example. The file should be a tab-delimited with several columns and one row for each sample. The first column must has the column name of 'Sample' and include the sample names. Also a valid annotation file is in the form of Table 2:

Sample	Variable1	Variable2	Variable3
Sample2	1	0	1
Sample1	1	1	1

Table 2: Simple structure of an annotation file

This file with additional information is important if you want to perform the t–test, if you use special colors for groups of samples, or if you want the samples to be grouped according to one column of the sample annotation file that can be specified through the parameter `columnForGrouping`. If none of these is important, just set all parameters to `NULL`. This also holds true for other parameters described below: setting `NULL` will make the *ddCt* neglect the parameter. If you have a sample annotation file and want to include only samples from this file into your final object, you can set the parameter 'useOnlySamplesFromAnno'.

```
> params$confFile <- system.file("scripts", "ddCt.conf", package="ddCt")
> params$inputFile <- c("Experiment1.txt")
> params$sampleAnnotationFile = NULL
> params$columnForGrouping    = NULL
> params$onlyFromAnnotation   = FALSE
```

4.3 Change gene names and sample names (optional)

You may change gene names and sample names, for instance to make the figure report of the data pretty. These can be done in the *ddCt* package.

In the following example, 'Gene1' will become 'Gene A', 'Gene4' will become 'Gene B', and so on. In the final plot, 'Gene A' will be plotted first, then 'Gene B' followed by 'Gene C', 'HK1' and finally 'HK2', which is the result of default ordering or factor. If you want to rename genes, **all** genes have to be included into this list, otherwise an error will be raised.

The rename of the samples follow a similar way by setting the variable `sampleAlias` in the `params` list.

```
> #params$geneAlias   = c("Gene1"="Gene A",
> #                       "Gene4"="Gene B",
```

```
> #                          "Gene5"="Gene C",
> #                          "Gene2"="HK1",
> #                          "Gene3"="HK2")
> #params$sampleAlias = NULL
```

Attention: If you rename genes or samples, the new names must be used for the parameters in the settings below.
In the commald-line you can set sutch a parameter in this way:

```
Rscript ddCt.R -geneAlias=Gene1="Gene A",Gene2="Gene B" ...
```

4.4 Housekeeping genes and reference samples

To calculate relative expression, one has to specify the housekeeping gene(s) and the reference sample(s). In this sample, one reference sample and two housekeeping genes are used. If more than one object is specified, the names have to be given as shown in this example.

If you specify more than one housekeeping gene, the software will use the mean of the C_t values of the housekeeping genes for the normalization. If you use more than one sample, the algorithm uses the mean of the chosen samples as the reference.

```
> params$referenceGene <- c("Gene1", "Gene2")
> params$referenceSample <- c("Sample1")
```

The comand-line version would look like this:

```
Rscript ddCt.R -referenceGene=Gene1,Gene2 ...
```

You may set a threshold to filter the C_T values, which is used to set an upper boundary of the C_T value to be considered. Any C_T value above this threshold will be treated as 'undetermined'.

```
> params$threshold <- 40
```

And next step you have to specify if you want to use the 'mean' or the 'median' to summarize the individual $C_T$values for a gene/sample combination. The median is often more appropriate when replicate number is large since it is more robust.

```
> params$mode = "median"
```

4.5 Efficiencies (optional)

You may include amplification efficiency for each gene (see the vignette *rtPCR* for more background information). There is also the possibility to include error estimates for the efficiencies (for example the standard deviation). These estimates will be used for the error calculation. These efficiencies can be specified in the following way:

```
> params$efficiencies       = c("Gene1"=1.9, "Gene2"=2, ...)
```

where "Gene1" represents the gene and 1.9 the efficience which is used for this gene
If you use efficiencies, only the raw Ct value and the final 'level' is calculated. There are no 'dCt' or 'ddCt' values. Hence no t-test can be performed if efficiencies are used. In this example we do not use efficiencies.

```
> #params$efficiencies       = c("Gene A"=1.9,"Gene B"=1.8,"HK1"=2,"Gene C"=2,"HK2"=2)
> #params$efficienciesError  = c("Gene A"=0.01,"Gene B"=0.1,"HK1"=0.05,"Gene C"=0.01,"HK2"=0.
```

4.6 Plot parameter (optional)

The following parameters are used to change the final plot. First you have to specify what you want to plot. Here you can specify either or both of the following two choices:

- level - For each gene and sample the relative expression to the reference line is plotted
- Ct - the raw, unnormalized but merged Ct values are plotted

```
> params$plotMode = c("level","Ct")
```

There may be cases where experiments want to exlcude certain gene and/or sample in the plot. Or sometimes one merely wants to plot a small fraction of genes and/or samples. One could set options in the parameter. For example the following example excludes the "NTC" sample in the plot:

```
> #REMAIN
> params$genesRemainInOutput   = NULL
> params$samplesRemainInOutput = NULL
> #NOT
> params$genesNotInOutput   = NULL
> params$samplesNotInOutput = NULL
```

Do you want the final plot to be drawn in a way such that the samples are plotted next to each other ?

```
> params$groupingBySamples = FALSE
```

set the variable plotPerObject to FALSE if one does not need one plot for each pair.

```
> params$plotPerObject = TRUE
```

This is often useful if you have many genes or samples. Depending on the parameter groupingBySamples either each gene (groupingBySamples=TRUE) will have its own plot, or each sample (groupingBySamples=FALSE) will have its own plot.

If you have a single plot for each individual gene, then you may color the sample bars according to one parameter of your sample annotation file (if you have specified such a file in the beginning of this script). You may also specify the colors (maybe with the help of *RColorBrewer*).

```
> #params$groupingForPlot = NULL
> #params$groupingColor   = c("#E41A1C","#377EB8","#4DAF4A","#984EA3","#FF7F00")
```

You may specify a `cutoff` for the y axis for all plots. Then all plots have the same scale.

```
> #params$cutoff = NULL
```

With the parameter `brewerColor` you can specify color sets you want to use in order to color the individual bars. For additional information have a look at the description of the `RColorBrewer` package.

```
> #params$brewerColor = c("Set3","Set1","Accent","Dark2","Spectral","PuOr","BrBG")
```

Set `legende` as `TRUE` in case you want the legend along the plot. Sometimes the plot will be messed up if the legend is plotted, so please have a try.

```
> params$legende  = TRUE
```

4.7 t-test and Wilcoxon test specification (optional)

You may perform either *t–test* or *Wilcoxon test* if you have specified a sample annotation file above. **Attention:** The *t–test* and *Wilcoxon test* will not work if efficiencies are used for the calculation.

```
> params$ttestPerform = FALSE
```

If you want to do a *t–test/Wilcoxon test*, you have to specify the name of the column of your sample information file that includes the group information needed for the tests. If there are more than two groups in this column, tests for each possible pair of groups are performed.

```
> #params$ttestCol = NULL
```

If you want to perform a paired *t–test/Wilcoxon test*, you have to specify a column of your sample information file that contains information describing the pairing of the samples. A pair of samples must have the same number/parameter in that column. Please have a look at the example.

```
> #params$pairsCol = NULL
```

You can specify whether you want to exclude some samples from the t-test. Here we again want to exclude the 'NTC' sample.

```
> #params$samplesRemainInTTest          = NULL
> #params$samplesNotInTTest         = NULL
```

4.8 Housekeeping gene correlation (optional)

If more than one housekeeping gene is used, the correlation (*Pearson* or *Spearman*) for each pair is calculated, together with a correlation plot. You may specify some samples that are not supposed to be used for the correlation calculation, for example negative control. In the default setting those samples are excluded that are also excluded from the plot.

```
> #params$samplesRemainInCor = NULL
> #params$samplesNotInCor    = NULL
```

4.9 Execution

Now all parameters have been set and you can pass them to the ddCtExec function

```
> ddCtExec(params)
```

At last all you have to do is to start R, change the directory if you like, and then type the following:

```
> Sweave("rtPCR-usage.Rnw")
```

The output is stored in a directory called Result_YOU_INPUT_FILENAMES

5 Results

Various files are created during the calculation process. The most important files are the following:

- A tab-delimited text file containing all calculated values for each gene/sample combination, e.g. Ct+error, dCt+error, ddCt+error, level+error. **Name of the file:** 'allValues' followed by the name of the file containing the individual Ct values and extension.
- A .pdf file including the plots, either of the 'levels' or the raw Ct values (depending on your choice) **Name of the file:** 'Level' or 'Ct' followed by 'Result', the name of the file containing the individual Ct values and the extension. In this example the name is 'LevelResultTest.pdf'.
- A tab-delimited text file containing the level and the error of the level. This file can be used in Excel to create own plots with error bars. **Name of the file:** 'LevelPlusError' followed by the name of the file containing the individual Ct values and extension.
- A tab-delimited text file containing the results from the t-test and the Wilcoxon test (if these tests have been performed). **Name of the file:** 'ttest' followed by the name of the groups used in the test, followed by the name of the file containing the individual Ct values and extension. In this example we have: 'ttestG1G2Test.txt'.
- An R object containing all calculated values for each gene/sample combination, e.g. Ct+error, dCt+error, ddCt+error, level+error. Furthermore the object includes additional sample information. **Name of the file:** 'Result' followed by the name of the file containing the individual Ct values and extension.

There are additional tab-delimited text files and .html files containing 'Ct','dCt', 'ddCt' and 'level' information. All this information is also included in the 'allValues' file, but in a different format.

6 Session Info

The script has been running in the following session:

- R version 4.1.1 (2021-08-10), `x86_64-pc-linux-gnu`
- Locale: `LC_CTYPE=en_US.UTF-8`, `LC_NUMERIC=C`, `LC_TIME=en_GB`, `LC_COLLATE=C`, `LC_MONETARY=en_US.UTF-8`, `LC_MESSAGES=en_US.UTF-8`, `LC_PAPER=en_US.UTF-8`, `LC_NAME=C`, `LC_ADDRESS=C`, `LC_TELEPHONE=C`, `LC_MEASUREMENT=en_US.UTF-8`, `LC_IDENTIFICATION=C`
- Running under: `Ubuntu 20.04.3 LTS`
- Matrix products: default
- BLAS: `/home/biocbuild/bbs-3.14-bioc/R/lib/libRblas.so`
- LAPACK: `/home/biocbuild/bbs-3.14-bioc/R/lib/libRlapack.so`
- Base packages: base, datasets, grDevices, graphics, methods, stats, utils
- Other packages: Biobase 2.54.0, BiocGenerics 0.40.0, RColorBrewer 1.1-2, ddCt 1.50.0, lattice 0.20-45
- Loaded via a namespace (and not attached): compiler 4.1.1, grid 4.1.1, tools 4.1.1, xtable 1.8-4