ReadqPCR: Functions to load RT-qPCR data into R

James Perkins
University College London

November 1, 2011

Contents

1 Introduction

The package `"ReadqPCR"` contains different functions for reading qPCR data into R.

As well as the functions to read in the data, `"ReadqPCR"` contains the `qPCRBatch` class definition. The data output by these RT-qPCR systems is in the form of cycle threshold, or Ct values, which represents the number of cycles of amplification needed in order to detect the expression of a given gene from a sample.

`"ReadqPCR"` is designed to be complementary to 2 other R modules: QCqPCR and NormqPCR, which are intended for (respectively) the quality control and normalisation of qPCR data. It must be installed before the other two modules.

2 read.qPCR

`read.qPCR` allows the user to read in qPCR data and populate a `qPCRBatch` R object (see section `qPCRBatch`) using their own data matrix. The format of the data file should be tab delimited and have the following columns, the first two of which are optional (although they should either be provided together, or not at all):

Well Optional, this represents the position of the detector on a plate. This information, if given, will be used to check the plates are of the same size and will also be used in order to plot a representation of the card to look for spatial effects and other potential problems. Both Well number and Plate ID must be present to enable a plate to be plotted.

Plate ID Optional, this is an identifier for the plate on which an experiment was performed. It is not possible to have duplicate plate IDs with the same Well number. Neither is it possible to have Plate Ids without Well numbers. Both Well number and Plate ID must be present to enable a plate to be plotted.

Sample The sample being analysed. Each sample must contain the same detectors in order to combine and compare samples effectively and to form a valid expression set matrix.

Detector This is the identifier for the gene being investigated. The Detectors must be identical for each sample.

Ct This is the cycle threshold for a given detector in the corresponding sample.

The generic function `read.qPCR` is called to read in the qPCR file. It is similar to the `read.affybatch` function of the `"affy"` package, in that it reads a file and automatically populates an R object, `qPCRBatch` described below. However it is different in that the file is user formatted. In addition, unlike `read.affybatch`, and also unlike the `read.taqman` function detailed below, only one file may be read in at a time.

If `Well` and `Plate ID` information are given, then these are used to populate the `exprs.well.order`, a new `assayData` slot introduced in the `qPCRBatch` object, as detailed below in section `qPCRBatch`.

So for the `qPCR.example.txt` file, in directory `exData` of this library, which contains `Well` and `Plate ID` information, as well as the mandatory `Sample`, `Detector` and `Ct` information, we can read in the data as follows.

```
> library(ReadqPCR) # load the ReadqPCR library
> path <- system.file("exData", package = "ReadqPCR")
> qPCR.example <- file.path(path, "qPCR.example.txt")
> qPCRBatch.qPCR <- read.qPCR(qPCR.example)
```

`qPCRBatch.qPCR` will be a `qPCRBatch` object with an exprs and exprs.well.order, as well as a phenoData slot which gets automatically populated in the same way as when using `read.affybatch`. More detail is given in the `qPCRBatch` section below.

`read.qPCR` can deal with technical replicates. If the same detector and sample identifier occurs more than once, the suffix `_TechRep.n` is concatenated 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 for a qPCR file with 2 technical replicates and 8 detectors per replicate, with one replicate per plate, the detector names would be amended as follows:

```
> 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"
```

The reason for appending the suffix when technical replicates are encountered is in order to populate the `exprs` and `exprs.well.order` slots correctly and keep them to the `assayData` format. It also allows the decisions on how to deal with the analysis and combination of technical replicates to be controlled by the user, either using the `"NormqPCR"` package, or potentially some other function that takes `assayData` format R objects as input.

3 read.taqman

`read.taqman` allows the user to read in the data output by the Sequence Detection Systems (SDS) software which is the software used to analyse the Taqman Low Density Arrays. This data consists of the header section, which gives some general information about the experiment, run date etc., followed by the raw Cts values detected by the software, followed by summary data about the experiment. `read.taqman` is a generic function, and is called in a way similar to the `read.affybatch` function of the `"affy"` package.

```
> taqman.example <- file.path(path, "example.txt")
> qPCRBatch.taq <- read.taqman(taqman.example)
```

Currently the SDS software only allows up to 10 plates to be output onto one file. read.taqman allows any number of SDS output files to be combined to make a single `qPCRBatch`, as long as they have matching detector identifiers.

```
> path <- system.file("exData", package = "ReadqPCR")
> taqman.example <- file.path(path, "example.txt")
> taqman.example.second.file <- file.path(path, "example2.txt")
> qPCRBatch.taq.two.files <- read.taqman(taqman.example,
+                                taqman.example.second.file)
```

SDS output will not necessarily contain plate identifiers, in which case a numeric identifier will be generated, which will increment for each plate, depending on the order of the

plates within the SDS files. This is important for filling the `exprs.well.order` slot of the `qPCRBatch`, which is used for assessing the quality of different arrays, using the `"QCqPCR"` package, as explained in section `qPCRBatch` and in the vignette for `"QCqPCR"`.

read.taqman can also deal with technical replicates. If the same detector and sample identifier occurs more than once, the suffix `_TechRep.n` will be concatenated 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 the order of occurence in the taqman data file. So for a taqman file with 4 technical replicates of 96 detectors per sample, with one sample per plate, the detector names would be amended as follows:

```
> taqman.example.tech.reps <- file.path(path, "exampleTechReps.txt")
> qPCRBatch.taq.tech.reps <- read.taqman(taqman.example.tech.reps)
> rownames(exprs(qPCRBatch.taq.tech.reps))[1:8]

[1] "ACE.Hs00174179_m1_TechReps.1"
[2] "ACE.Hs00174179_m1_TechReps.2"
[3] "ACE.Hs00174179_m1_TechReps.3"
[4] "ACE.Hs00174179_m1_TechReps.4"
[5] "AT1R.AGTR1..Hs00241341_m1_TechReps.1"
[6] "AT1R.AGTR1..Hs00241341_m1_TechReps.2"
[7] "AT1R.AGTR1..Hs00241341_m1_TechReps.3"
[8] "AT1R.AGTR1..Hs00241341_m1_TechReps.4"
```

As with read.qPCR, the motivation for appending the suffix when technical replicates are encountered is in order to populate the `exprs` and `exprs.well.order` slots correctly and keep them to the `assayData` format. Again it allows the decisions on how to deal with the analysis of technical replicates to be controlled by the user, either using the `"NormqPCR"` package, or otherwise.

4 qPCRBatch

`qPCRBatch` is an S4 class, designed to store information on the Ct raw values which represents the relative gene expression for a given sample, phenotypic information on the different samples which enable the user to compare expression accross different conditions or cell lines, and information on the spatial location of the different detectors used to measure Ct. This is achieved by making `qPCRBatch` an an extension of eSet, which means we can recycle slots such as exprs and pData, and by introducing a new `assyData` slot. Here is an example of what a `qPCRBatch` looks like. note the similarity to `eSet`:

```
> qPCRBatch.taq
```

```
qPCRBatch (storageMode: lockedEnvironment)
assayData: 96 features, 8 samples
  element names: exprs, exprs.well.order
protocolData: none
phenoData
  sampleNames: fp1.day3.v fp2.day3.v ... fp.8.day.3.mia (8 total)
  varLabels: sample
  varMetadata: labelDescription
featureData: none
experimentData: use 'experimentData(object)'
Annotation:
```

pData will be filled automatically if no data is given, in a way analagous to read.affybatch:

```
> pData(qPCRBatch.taq)
```

```
               sample
fp1.day3.v          1
fp2.day3.v          2
fp5.day3.mia        3
fp6.day3.mia        4
fp.3.day.3.v        5
fp.4.day.3.v        6
fp.7.day.3.mia      7
fp.8.day.3.mia      8
```

In addition there is a new slot, `exprs.well.order` which extends the `assayData` slot used for `exprs()`. It has the same dimensions as `exprs` (as every instance of `assayData` must). The cells contain further details on the position on the arrays where the different meaurements were taken.

The data provided by this slot can be used in order to identify certain problems with arrays, perhaps due to spatial effects and other problems with the microfluidics technology that is used by many of these systems (see `"QCqPCR"` for more details).

This is conceptually similar to the cdf file information being stored in the `AffyBatch` class, which contains information on the spatial layout of features on an affy chip. However it differs since it allows for different arrays within the same `affyBatch` object to have different layouts to each other. This information can be viewed using the `exprs.well.order()` function and is later used in the `"QCqPCR"` package in order to produce pseudoplots of the qPCR cards, in a method analagous to the pseudo-images produced by `"affyPLM"`.

When using `read.taqman`, if the input file includes identifiers for the different arrays in the experiment, the identifiers will be of the format `<plate.id>-<plate.position>`. However if no names are given for the different plates, `"ReadqPCR"` will assign them a

numeric identifier, which increments depending on the order of plates in the original file. When several input files are given, as in the case of SDS files, the order in which they are supplied as arguments to the `read.taqman` function will be mirrored in the order of the numeric identifiers for the different plates. However, to minimise confusion, we recommend the useR giving the plates their own unique identifiers where possible.

Without plate names:

```
> head(exprs.well.order(qPCRBatch.taq))
```

```
                    fp1.day3.v fp2.day3.v fp5.day3.mia fp6.day3.mia
Actb.Rn00667869_m1   "1-38"     "1-134"    "1-230"      "1-326"
Adipoq.Rn00595250_m1 "1-61"     "1-157"    "1-253"      "1-349"
Adrbk1.Rn00562822_m1 "1-66"     "1-162"    "1-258"      "1-354"
Agtrl1.Rn00580252_s1 "1-91"     "1-187"    "1-283"      "1-379"
Alpl.Rn00564931_m1   "1-10"     "1-106"    "1-202"      "1-298"
B2m.Rn00560865_m1    "1-37"     "1-133"    "1-229"      "1-325"
                    fp.3.day.3.v fp.4.day.3.v fp.7.day.3.mia fp.8.day.3.mia
Actb.Rn00667869_m1   "2-38"       "2-134"      "2-230"        "2-326"
Adipoq.Rn00595250_m1 "2-61"       "2-157"      "2-253"        "2-349"
Adrbk1.Rn00562822_m1 "2-66"       "2-162"      "2-258"        "2-354"
Agtrl1.Rn00580252_s1 "2-91"       "2-187"      "2-283"        "2-379"
Alpl.Rn00564931_m1   "2-10"       "2-106"      "2-202"        "2-298"
B2m.Rn00560865_m1    "2-37"       "2-133"      "2-229"        "2-325"
```

With plate names:

```
> taqman.example.plateNames <- file.path(path, "exampleWithPlateNames.txt")
> qPCRBatch.taq.plateNames <- read.taqman(taqman.example.plateNames)
> head(exprs.well.order(qPCRBatch.taq.plateNames))
```

```
                    gp.1.day.3.v gp.2.day.3.v gp.5.day.3.mia gp.6.day.3.mia
Actb.Rn00667869_m1   "PlateA-38"  "PlateA-134" "PlateA-230"   "PlateA-326"
Adipoq.Rn00595250_m1 "PlateA-61"  "PlateA-157" "PlateA-253"   "PlateA-349"
Adrbk1.Rn00562822_m1 "PlateA-66"  "PlateA-162" "PlateA-258"   "PlateA-354"
Agtrl1.Rn00580252_s1 "PlateA-91"  "PlateA-187" "PlateA-283"   "PlateA-379"
Alpl.Rn00564931_m1   "PlateA-10"  "PlateA-106" "PlateA-202"   "PlateA-298"
B2m.Rn00560865_m1    "PlateA-37"  "PlateA-133" "PlateA-229"   "PlateA-325"
                    gp.3.day.3.v gp.4.day.3.v gp.7.day.3.mia gp.8.day.3.mia
Actb.Rn00667869_m1   "PlateB-38"  "PlateB-134" "PlateB-230"   "PlateB-326"
Adipoq.Rn00595250_m1 "PlateB-61"  "PlateB-157" "PlateB-253"   "PlateB-349"
Adrbk1.Rn00562822_m1 "PlateB-66"  "PlateB-162" "PlateB-258"   "PlateB-354"
Agtrl1.Rn00580252_s1 "PlateB-91"  "PlateB-187" "PlateB-283"   "PlateB-379"
```

```
Alpl.Rn00564931_m1   "PlateB-10"  "PlateB-106" "PlateB-202"   "PlateB-298"
B2m.Rn00560865_m1    "PlateB-37"  "PlateB-133" "PlateB-229"   "PlateB-325"
```

In addition, a mixture of files with and without plate identifiers is possible.

```
> taqman.example <- file.path(path, "example.txt")
> taqman.example.plateNames <- file.path(path, "exampleWithPlateNames.txt")
> qPCRBatch.taq.mixedPlateNames <- read.taqman(taqman.example,
+                                     taqman.example.plateNames)
> head(exprs.well.order(qPCRBatch.taq.mixedPlateNames))

                    fp1.day3.v fp2.day3.v fp5.day3.mia fp6.day3.mia
Actb.Rn00667869_m1   "1-38"     "1-134"    "1-230"      "1-326"
Adipoq.Rn00595250_m1 "1-61"     "1-157"    "1-253"      "1-349"
Adrbk1.Rn00562822_m1 "1-66"     "1-162"    "1-258"      "1-354"
Agtrl1.Rn00580252_s1 "1-91"     "1-187"    "1-283"      "1-379"
Alpl.Rn00564931_m1   "1-10"     "1-106"    "1-202"      "1-298"
B2m.Rn00560865_m1    "1-37"     "1-133"    "1-229"      "1-325"
                    fp.3.day.3.v fp.4.day.3.v fp.7.day.3.mia fp.8.day.3.mia
Actb.Rn00667869_m1   "2-38"       "2-134"      "2-230"        "2-326"
Adipoq.Rn00595250_m1 "2-61"       "2-157"      "2-253"        "2-349"
Adrbk1.Rn00562822_m1 "2-66"       "2-162"      "2-258"        "2-354"
Agtrl1.Rn00580252_s1 "2-91"       "2-187"      "2-283"        "2-379"
Alpl.Rn00564931_m1   "2-10"       "2-106"      "2-202"        "2-298"
B2m.Rn00560865_m1    "2-37"       "2-133"      "2-229"        "2-325"
                    gp.1.day.3.v gp.2.day.3.v gp.5.day.3.mia gp.6.day.3.mia
Actb.Rn00667869_m1   "PlateA-38"  "PlateA-134" "PlateA-230"   "PlateA-326"
Adipoq.Rn00595250_m1 "PlateA-61"  "PlateA-157" "PlateA-253"   "PlateA-349"
Adrbk1.Rn00562822_m1 "PlateA-66"  "PlateA-162" "PlateA-258"   "PlateA-354"
Agtrl1.Rn00580252_s1 "PlateA-91"  "PlateA-187" "PlateA-283"   "PlateA-379"
Alpl.Rn00564931_m1   "PlateA-10"  "PlateA-106" "PlateA-202"   "PlateA-298"
B2m.Rn00560865_m1    "PlateA-37"  "PlateA-133" "PlateA-229"   "PlateA-325"
                    gp.3.day.3.v gp.4.day.3.v gp.7.day.3.mia gp.8.day.3.mia
Actb.Rn00667869_m1   "PlateB-38"  "PlateB-134" "PlateB-230"   "PlateB-326"
Adipoq.Rn00595250_m1 "PlateB-61"  "PlateB-157" "PlateB-253"   "PlateB-349"
Adrbk1.Rn00562822_m1 "PlateB-66"  "PlateB-162" "PlateB-258"   "PlateB-354"
Agtrl1.Rn00580252_s1 "PlateB-91"  "PlateB-187" "PlateB-283"   "PlateB-379"
Alpl.Rn00564931_m1   "PlateB-10"  "PlateB-106" "PlateB-202"   "PlateB-298"
B2m.Rn00560865_m1    "PlateB-37"  "PlateB-133" "PlateB-229"   "PlateB-325"
```

If the files to be combined do not have matching detector names, or if duplicate sample or plate names are given, read.taqman will stop and give an error message.

When reading in qPCR files with read.qPCR, exprs.well.order will be populated as long as Well and Plate ID columns are given in the input file, otherwise the exprs.well.order slot will be NULL.

So when plate ID and Well data are given:

```
> head(exprs.well.order(qPCRBatch.qPCR))
```

```
                   caseA caseB controlA controlB
gene_ai_TechReps.1  A-18  A-43     B-18     B-43
gene_ai_TechReps.2  C-18  C-43     D-18     D-43
gene_az_TechReps.1  A-23  A-48     B-23     B-48
gene_az_TechReps.2  C-23  C-48     D-23     D-48
gene_bc_TechReps.1   A-6  A-31      B-6     B-31
gene_bc_TechReps.2   C-6  C-31      D-6     D-31
```

And when they are not:

```
> qPCR.example.noPlateOrWell <- file.path(path, "qPCR.noPlateOrWell.txt")
> qPCRBatch.qPCR.noPlateOrWell <- read.qPCR(qPCR.example.noPlateOrWell)
> exprs.well.order(qPCRBatch.qPCR.noPlateOrWell)
```

```
NULL
```

Once a qPCRBatch has been populated it is theoretically possible to use any tool which takes as it's input an exprs set matrix. However it is important to bear in mind the values are not raw expression values but Ct values, and a lower the Ct will indicate a higher expression level for a given transcript in the sample. Also it is important to note that when normalising, the amount is relative and is intended to be compared to another condition or tissue type in order to look for differential expression between condition; the technology is not designed to give absolute quantification.