qPCR data analysis

Daniel Hammarström

IDR4000

updated: 2020-11-13

Background

- Reverse transcriptase followed by quantitative polymerase chain reaction (RT-qPCR or qPCR) can be used to quantify abundance of RNA from a biological sample.
- This is the technique most commonly used to determine **targeted gene expression**

- Others techniques include northern blotting (old), micro arrays (specific probes, many target) and RNA-seq (many targets)

One-step vs. two-step qPCR

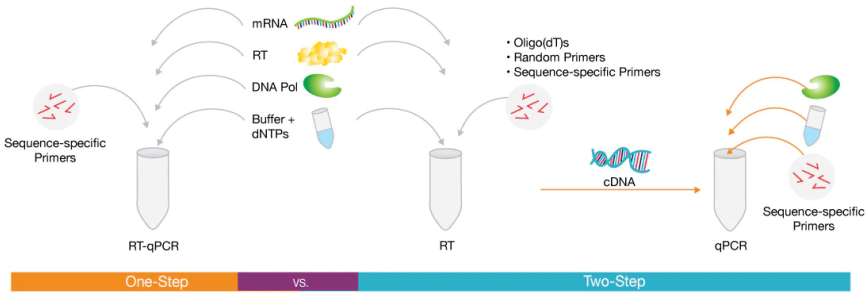


Image source: thermofischer.com

cDNA synthesis: we used a mixture of random hexamer and Oligo dT primers

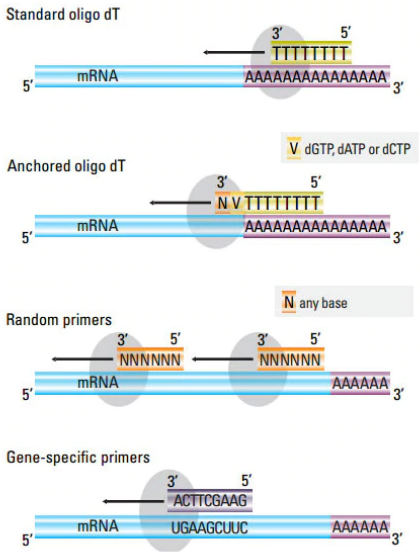
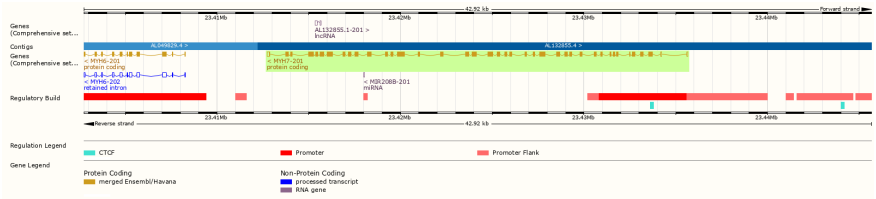


Image source: thermofischer.com

The qPCR reaction leads to exponential amplification of target cDNA

- mRNA sequences are known from mapping of the human genome.
- Specific primers are designed to capture a specific gene.



Amplification of genomic DNA is avoided in primer design.

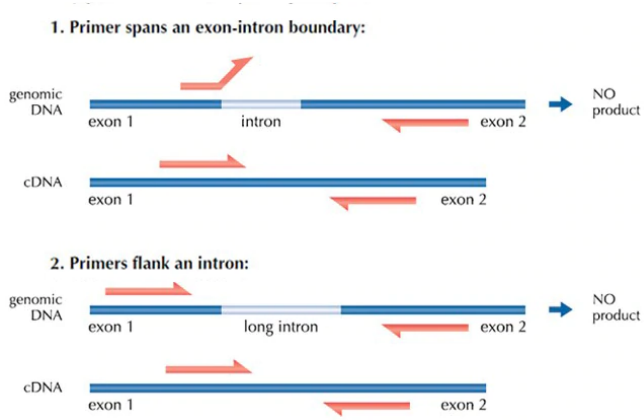


Image source: thermofischer.com

A simulation

Products doubles with each cycle.

The quantification cycle is the primary outcome of a qPCR experiment

- The cycle where fluorescence reaches above the background noise.
- May be set as an arbitrary threshold based on baseline noise
- Can also be calculated from modeling the relationship between cycles and fluorescence

Installing qpcrpal

```
install.package(qpcR) # qpcR package

library(remotes)
remotes::install_github("dhammarstrom/qpcrpal", build_vignettes = TRU
```


Using qpcrpal

- qpcrpal can be used to analyze "multi-plate" experiments

```
library(qpcR)
library(qpcrpal)
library(parallel)

qpcr.dat <- read_quant5("./data/qpcr_data/Group2.xls",
                        skip = 47)

models <- model_qpcr(qpcr.dat)

analyzed.models <- analyze_models(models)
```


Normalize data

- If we have loaded different amount of RNA in the cDNA synthesis we are comparing apples and oranges when comparing different samples
- To overcome this we use *reference genes* to normalize the data
- Normalization is important as a bad *reference gene* will reduce an experiments ability to detect changes in gene expression.

Normalized data

$$Normalized\ Expression = \frac{Target\ gene}{Reference\ gene}$$

Target gene and reference gene expression has to be *linearized* in calculations. Relative abundance:

$$Target\ gene\ expression = Efficiency^{-Cq}$$

$$Target\ gene\ expression = 2^{-Cq}$$

$$Target\ gene\ expression = 2^{-25} = 0.0000000298$$

$$Normalized\ Expression = \frac{2^{-25}}{2^{-20}}$$

Between sample changes

We often calculate the fold-change between samples as the outcome of studies. For example between trained and un-trained biopsies.

$$Relative\ expression = \frac{Trained}{Untrained}$$

In practice this is done with log-transformed data

$$Relative\ expression = log(\frac{Trained}{untrained})$$

giving the log-differences between groups. Transforming to the linear scale will give the fold-change.

In R

```
analyzed.models %>%
  dplyr::select(ID, cpD2) %>%
  filter(cpD2 > 10) %>%
  separate(ID, into = c("sample", "cDNA", "condition", "target"), sep
  filter(sample %in% c("C1", "C2"),
         target %in% c("MyHC2X F5R5",
                       "B2m F5R5")) %>%
  mutate(target = gsub(" F5R5", "", target)) %>%
  pivot_wider(names_from = target, values_from = cpD2) %>%

  mutate(norm = (2^-MyHC2X) / (2^-B2m)) %>%

  dplyr::select(sample, norm) %>%
  pivot_wider(names_from = sample, values_from = norm) %>%
  mutate(relative.expression = C2/C1) %>%
  print()
```

```
## # A tibble: 1 x 3
##      C1    C2 relative.expression
##   <dbl> <dbl>               <dbl>
## 1  32.7  11.7               0.358
```


Alternatives to reference gene normalization

- Gene family normalization can be used to remove differences in sample preparations
- Genes from a functional family of genes (e.g. myosin heavy chains) can be expressed as a percentage of their total expression (Ellefsen & Stensløkken 2010).

```
analyzed.models %>%
  dplyr::select(ID, cpD2) %>%
  filter(cpD2 > 10) %>%
  separate(ID, into = c("sample", "cDNA", "condition", "target"), sep

  dplyr::select(sample, target, cpD2) %>%

  filter(sample %in% c("C1", "C2"),
         target %in% c("MyHC2X F5R5",
                       "MyHC2A F5R5",
                       "MyHC1 F1R1")) %>%
  group_by(sample) %>%
  mutate(linear = 2^-cpD2,
         rel.expression = linear / sum(linear)) %>%
  print()
```



```
## # A tibble: 6 x 5
## # Groups:   sample [2]
##   sample target       cpD2   linear rel.expression
##   <chr>  <chr>       <dbl>    <dbl>          <dbl>
## 1 C1     MyHC1 F1R1   28.9 1.94e- 9         0.173
## 2 C1     MyHC2A F5R5  26.8 8.27e- 9         0.734
## 3 C1     MyHC2X F5R5  29.8 1.05e- 9         0.0931
## 4 C2     MyHC1 F1R1   31.2 4.11e-10         0.236
## 5 C2     MyHC2A F5R5  29.8 1.08e- 9         0.619
## 6 C2     MyHC2X F5R5  31.9 2.53e-10         0.145
```