Analysis of the qPCR dataset

Frédéric Schütz

31/03/2022

File import

We first need to import the dataset into R. This can be done in several ways; one of them is use the readxl package in order to read the Excel file (.xlsx) directly into R; another one is to first save the file in text format (e.g. CSV) before reading it into R.

Directly reading the Excel file

This is usually recommended, because it involves little manual operations or data convertions. However, when the Excel file contains complex formatting (e.g. merged cells), it may be difficult to recover the data directly in R.

```
library(readxl)
qPCR <- read_excel("qPCR.xlsx")
```

```
## New names:
## * `` -> ...7
## * `` -> ...8
```

```
# This produces a Tibble; if you do not work with Tibbles,
# you can convert it back to a data frame
data_excel <- as.data.frame(qPCR)
head(data_excel)
```

```
##   sample.ID group   gene    ct   dct average dct ...7           ...8      ddct
## 1        99  test HSP7B' 17.24    NA          NA   NA           <NA>        NA
## 2        20  WT C    AKT 16.49 -0.70  -0.7400000   NA WT C vs WT T : -1.107778
## 3        20  WT C    AKT 16.44 -0.96          NA   NA           <NA>        NA
## 4        20  WT C    AKT 16.65 -0.56          NA   NA           <NA>        NA
## 5        21  WT C    AKT  16.5 -0.85  -0.5666667   NA           <NA>        NA
## 6        21  WT C    AKT 16.74 -0.43          NA   NA           <NA>        NA
##   2^(-ddct)
## 1        NA
## 2  2.155134
## 3        NA
## 4        NA
## 5        NA
## 6        NA
```

```
summary(data_excel)
```

```
##    sample.ID        group               gene               ct
##  Min.   :20.00   Length:76          Length:76          Length:76
##  1st Qu.:23.00   Class :character   Class :character   Class :character
##  Median :26.00   Mode  :character   Mode  :character   Mode  :character
```

```
##  Mean   :29.37
##  3rd Qu.:29.00
##  Max.   :99.00
##
##       dct           average dct        ...7             ...8
##  Min.   :-0.9600   Min.   :-0.9433   Mode:logical   Length:76
##  1st Qu.:-0.4050   1st Qu.:-0.3842   NA's:76        Class :character
##  Median : 0.3950   Median : 0.3983                  Mode  :character
##  Mean   : 0.3792   Mean   : 0.3792
##  3rd Qu.: 0.8650   3rd Qu.: 0.8317
##  Max.   : 2.0300   Max.   : 1.8633
##  NA's   :40        NA's   :64
##       ddct          2^(-ddct)
##  Min.   :-1.108   Min.   :2.155
##  1st Qu.:-1.108   1st Qu.:2.155
##  Median :-1.108   Median :2.155
##  Mean   :-1.108   Mean   :2.155
##  3rd Qu.:-1.108   3rd Qu.:2.155
##  Max.   :-1.108   Max.   :2.155
##  NA's   :75       NA's   :75
```

```
# Some missing values are indicated by "?
data_excel[data_excel$ct=="?", "ct" ] <- NA
```

The data still needs to be cleaned up; see below.

Exporting to CSV format first

Another solution is to export the file from Excel in CSV format (or an equivalent text-based format), and then read it then into R, using the read.table or read.csv functions.

Particular care should be given to the specification of the file format (column separator, presence of header, etc). In our case, the file was saved in tab-separated columns with a header, but your file may be in a different format.

It is worth noting that some entries in the file contains single quote characters ('), for example **HSP7B'** and **hERRa1 mRNA, 3' end**. By default, some R functions such as read.table() will interpret this as a quote, and will mangle the output; in this case, you will need to specifically indicate that no quote character should be interpreted as such.

Preparing and cleaning the data

After having entered the data into R, we still need to clean it and prepare it for analysis. First, we select only the columns we are interested in (and remove those that do not contain raw data, but calculations done in Excel).

The information we want are in columns 1 to 4, but it is preferable to select using the column names. This is more robust, as it will still work if the order of the columns change in the original data file. Also, if one column does not exist, it will yield an error, while selecting columns based on their position will always work even if we select the wrong columns (in which case it will fail silently).

```
# We use one of the datasets we read earlier
data <- data_excel

# Equivalent to selecting data[, 1:4], but less fragile
data <- data[, c("sample.ID", "group", "gene", "ct")]
```

```
# We remove all the samples labelled as "test"
data <- data[ data$group!="test", ]

# We have already changed the "?" that indicated a missing value into
# an NA above, but there is as least another problem in the data.

# The Ct columns contains issues; we can find what they are by
# trying to convert the column to a numeric variable and check which
# numbers were not converted:
converted <- as.numeric(data$ct)
which(is.na(converted))
```

```
## integer(0)
```

```
data$ct[which(is.na(converted))]
```

```
## character(0)
```

```
# We have a value 17,21 (with a comma instead of a decimal point),
# which we can correct in the following way:
data$ct[data$ct == "17,21"] <- 17.21

# And we then convert back to numeric
data$ct <- as.numeric(data$ct)

summary(data)
```

```
##    sample.ID        group               gene                ct
##  Min.   :20.00   Length:72          Length:72          Min.   :16.30
##  1st Qu.:22.75   Class :character   Class :character   1st Qu.:17.15
##  Median :25.50   Mode  :character   Mode  :character   Median :17.29
##  Mean   :25.50                                         Mean   :17.43
##  3rd Qu.:28.25                                         3rd Qu.:17.65
##  Max.   :31.00                                         Max.   :19.18
```

```
# It works now
```

We can now get some more information from the data. The column "group" contains both the genotype and the treatment information; we will need to have them as separate variables for the analysis. We will avoid doing this manually (for example in the spreadsheet), and will see several was for doing this with R.

```
# Using base R: split the group information at the space, producing
# a list containing vectors of the two elements (genotype and treatment)
splitname <- strsplit(data$group, " ")
# use do.call and rbind to make the list elements into a matrix, and
# give names to the columns
newcolumns <- do.call(rbind, splitname)
colnames(newcolumns) <- c("genotype", "treatment")
# Add the new columns to the original data
data_1 <- cbind(data, newcolumns)

# do.call() is not a very well-known R command; there are other ways
# to perform the same task, but the R code is not trivial either way
# For example, we can use unlist to vectorize the list, and then put
# it into a vector
newcolumns <- matrix(unlist(splitname), ncol=2, byrow=TRUE)
colnames(newcolumns) <- c("genotype", "treatment")
```

```
data_1 <- cbind(data, newcolumns)

# We can also use the separate function in the tidyr package
# Note: the pipe operator |> requires R version 4.1 (or above)
# alternatively, you can use the legacy %>% operator available in
# package magrittr
library(tidyr)
data_2 <- data |>
  separate(group, c("Genotype", "Treatment"), " ", remove=FALSE)
# Note that in this case, we do need to specify the remove=FALSE
# option, in order to keep the original variable (which we will need
# later)
```

At this step, we have information about the AKT and GAPDH genes. We will not perform any normalization (using GAPDH) for now, so we will keep the AKT data only

```
# We select one of the data frame we created above
data <- data_1

dataAKT <- data[data$gene=="AKT",]
```

Taking care of technical replicates

It would be incorrect to use the technical replicates as if they were biological replicates – we may then see false positive results, as any statistical test will be conducted over an artifically high number of samples.

With only 3 technical replicates per biological sample, there isn't much else we can than average the technical replicates into a single value. While doing this, we reduce the (technical) variability of our samples, so that we will still get better results than if we had only had a single technical replicate for each biological replicate.

```
# Aggregating the technical replicates with base R and the
# "aggregate" command:
agdata <- aggregate(dataAKT$ct,
                    list(sample.ID=dataAKT$sample.ID,
                         group=dataAKT$group,
                         genotype=dataAKT$genotype,
                         treatment=dataAKT$treatment),
                    FUN=mean)
names(agdata)[4] <- "meanCt"

# Doing the same with the dplyr package and the group_by and
# summarize commands
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
## The following objects are masked from 'package:stats':
##
##     filter, lag
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
agdata <- as.data.frame( dataAKT |>
  group_by(`sample.ID`, group, gene, genotype, treatment) |>
```

```
  summarize(meanCt = mean(ct, na.rm = TRUE)))
```

```
## `summarise()` has grouped output by 'sample.ID', 'group', 'gene', 'genotype'.
## You can override using the `.groups` argument.
```

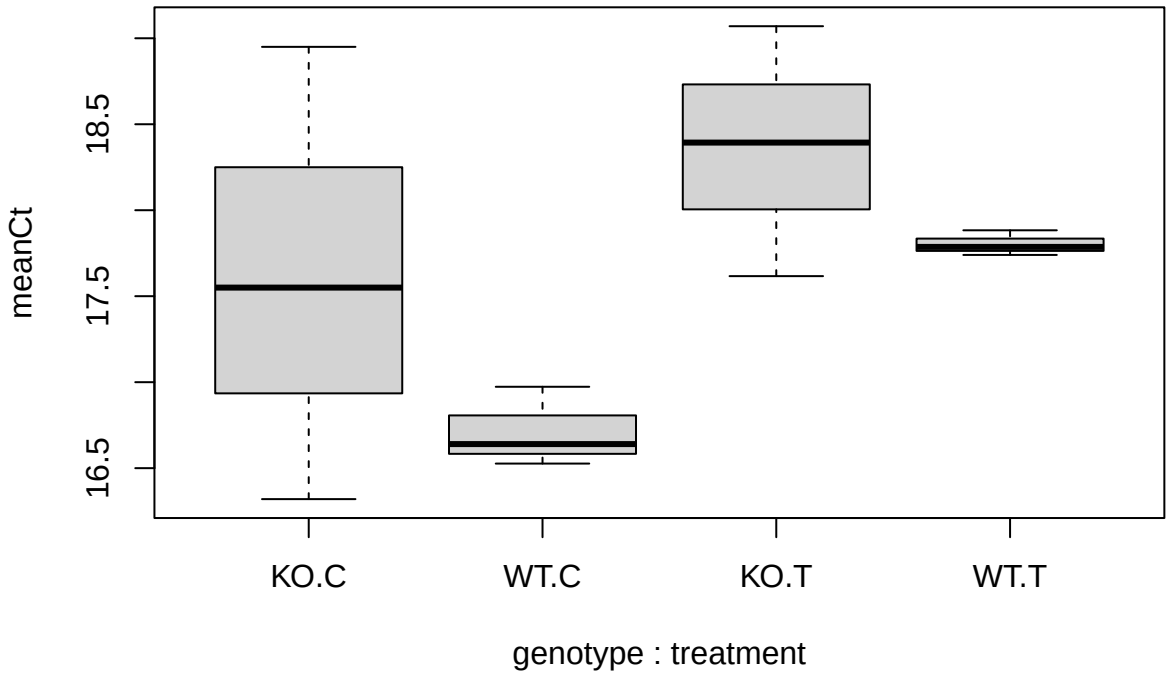
Note that, while the two resulting data frames will mostly contain the same data, the rows will not necessarily be in the same order.

Data analysis

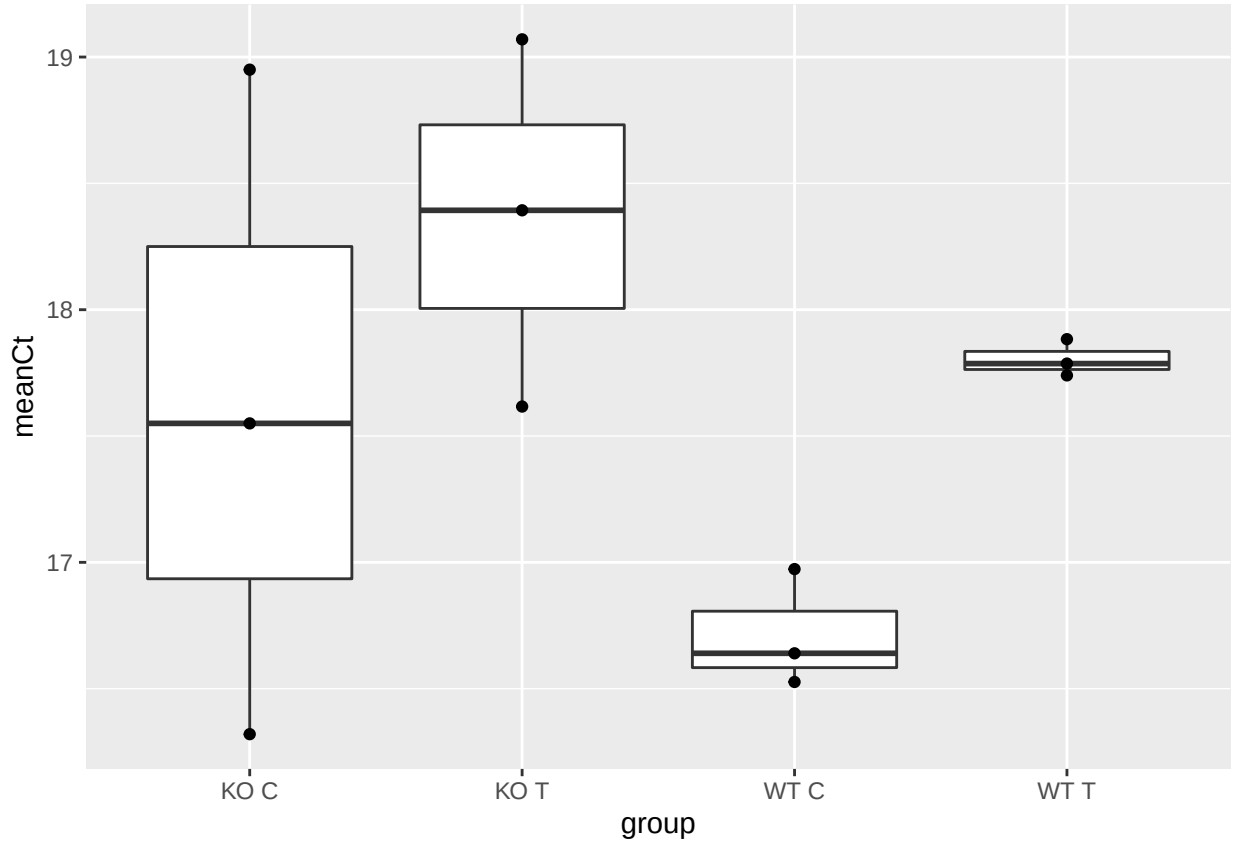
We can start looking at the data in each group. A boxplot (done with base R graphics) is a typical way to represent such data; however, it is not the best way to represent group containing only 3 data points; a second graph (made with ggplot2) shows both the boxplot and the individual points.

```
boxplot(meanCt ~ genotype*treatment, data=agdata)

library(ggplot2)
```



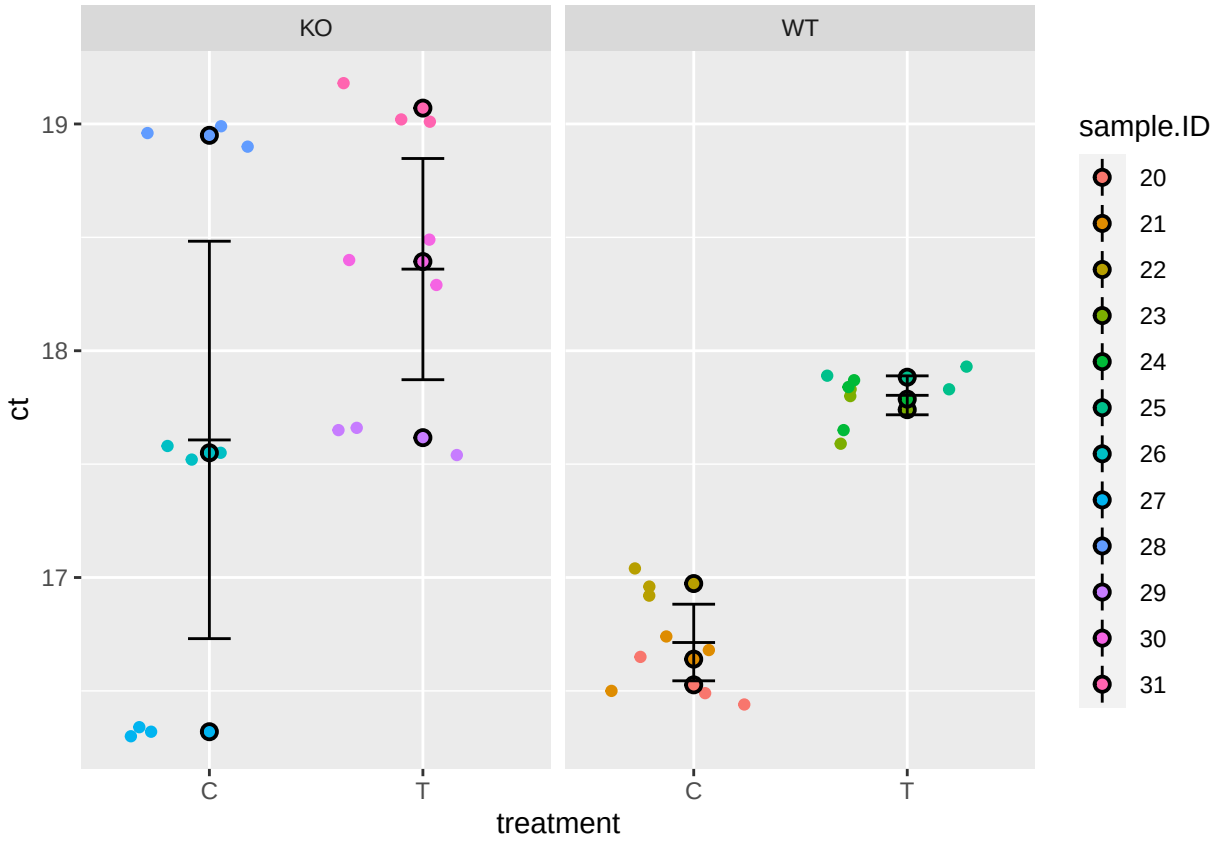

```
scatterplot <- ggplot( aes(x=group, y=meanCt), data=agdata)
scatterplot + geom_boxplot() + geom_point()
```



We can also look at a more complex graph, containing all the technical replicates:

```
dataAKT$sample.ID <- as.factor(dataAKT$sample.ID)
scatterplot <- ggplot( aes(x=treatment, y=ct), data=dataAKT)
scatterplot + geom_jitter(aes(col=sample.ID), height=0) +
  stat_summary(aes(fill=sample.ID), fun=mean, shape=21) +
  stat_summary(fun=mean, geom="crossbar", width=0.2, fatten=1) +
  stat_summary(fun.data="mean_cl_normal", geom="errorbar", width=0.2)+
  facet_wrap(~genotype)
```

```
## Warning: Removed 6 rows containing missing values (geom_segment).
## Removed 6 rows containing missing values (geom_segment).
```



We can see that the variability among technical replicates is very low (compared to the variability among the biological replicates).

We can notice some differences between the groups - not only between the means, but also in variances: the KO groups have a large variance, while the WT groups have a much smaller variance.

Statistical tests

We can now answer the questions that we were asking.

Is there a significant difference between C and T in the WT ?

We can perform a Student t-test in the wild-type mice, between the control and treated groups. With only 3 samples in each group, it is impossible to assess whether common assumptions for the t-test are satisfied. The normality assumption, however, is unlikely to be an issue with such continuous data, as the t-test is robust to departures from normality. Inegality of variance is not an issue either, as the test used is the Welch t-test, which accounts for this possible inegality. The most important assumption, independance of the observations, can not be tested; we must rely on the experimental design of the experiment to make sure that the samples were independant.

The result of the t-test is:

```
t.test(agdata[agdata$genotype=="WT" & agdata$treatment=="C", "meanCt"],
       agdata[agdata$genotype=="WT" & agdata$treatment=="T", "meanCt"])
```

```
##
##  Welch Two Sample t-test
##
```

```
## data:  agdata[agdata$genotype == "WT" & agdata$treatment == "C", "meanCt"] and agdata[agdata$genotype
## t = -7.7557, df = 2.3927, p-value = 0.00949
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -1.6088816 -0.5711184
## sample estimates:
## mean of x mean of y
##  16.71333  17.80333
```

This indicates that the data is unlikely to be coming from groups with the same mean, so that there is evidence that the means of the two groups are different. The mean for the treated group is 1.09 cycles higher than for the control group, corresponding to a reduction in expression of 2.12 fold changes (95% CI for the difference between the means: [-1.06, -0.57]).

Is there a difference in the effect of the treatment between WT and KO ?

We can perform an ANOVA on the data, in order to estimate the effect of the genotype, the treatment and the interaction between the two; the interaction will allow us to answer the question.

```
# Combined effect of treatment and genotype
summary(aov(meanCt ~ genotype*treatment, data=agdata))
```

```
##                    Df Sum Sq Mean Sq F value Pr(>F)
## genotype            1  1.577  1.5769   2.719 0.1378
## treatment           1  2.548  2.5484   4.394 0.0693 .
## genotype:treatment  1  0.085  0.0850   0.147 0.7118
## Residuals           8  4.640  0.5799
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The ANOVA does not seem to indicate that the interaction is significant; indeed, none of the parameters indicate some significance (except maybe the treatment).

However, there seems to be a paradox between the two analysis - the t-test yielded a significant result, and if any difference between the groups is significant, the ANOVA should show it as well.

If we look at the boxplot we plotted above, we noticed that there was a large difference in variance between the groups: the WT groups showed a lower variance, which makes the t-test significant despite the low number of observations.

However, the ANOVA assumes that all groups have the same variance, and the global variance is an average (a pool) of the variance in each groups. The large variance in the KO groups leads to an inflated global variance, which leads the ANOVA not to show any significant variable.