An (opinionated) introduction to R for biologists

Analysing qPCR and longitudinal data using R

Carlos Ronchi

2022-08-01

Table of contents

Welcome

For a more gentle introduction to R and its basics, check the book Hands-On Programming with R.

If you are familiar with R already and want to go deeper with the tidyverse functions, take a look at the book R for Data Science.

Preface

Part I

First steps

1 Prerequisites

2 Loading your data

The first step in any data analysis is to open up your favorite software and load the data up. You are probably familiar with user interfaces, where you click some buttons, you upload your excel file and *voilá*, data is ready to analyse.

In this chapter I will teach you how to format your qPCR data so you can load it easily to R. I will also show you how to load the data itself.

> ⚠ Warning
>
> If you are using RStudio, you can also click some buttons and load your data, without writing any code. I strongly recommend you to **not** do this. In the next sections I will show how you should be doing.

2.1 Formats, formats and more formats...

If you are here, you are probably used to excel and the format xlsx. R can open these files, however, it is easier to run your analysis if you have your data in the *csv* format. For qPCR specially, each qPCR machine, the excel sheet will be different and not in a standard format. Usually it is a table with several cells containing information regarding your run and other parameters.

Given your fresh table from the qPCR machine, we will extract some columns. In qPCR experiments, usually there are three technical replicates for your samples. This is reflected in how the data is saved.

An example table is shown in the data folder.

2.2 How to load your data

If you saved your table as a csv file, then we have several options to load the file. You can use the function `read.csv` or `read.table` from base R to open the tables. The way to use these functions is shown below.

```
qpcr <- read.csv("../data/qPCR.csv")
qpcr <- read.table("../data/qPCR.csv", sep = ",", header = TRUE)
```

Note that for `read.table` we need to specify the separator, in this case it is a comma (",") and we also specify `HEADER = TRUE`. This means the first row of your table is the header of your data frame. In general if you have a csv file, it is easier to run `read.csv`.

2.3 Checking the data

When loading the data, it is very important to check if it was imported successfully or any problem happened. A package that we will be using throughout this book is `dplyr`. Within `dplyr` we have some functions that help us deal data frames in a very intuitive way. The first function we will use here to check the data is `glimpse`. We load the package here, but it is best practice to load all the packages you will be using at the beginning of your analysis.

```
# we now use the function glimpse on our dataframe
glimpse(qpcr)
```

```
Rows: 72
Columns: 4
$ sample_id <int> 20, 20, 20, 21, 21, 21, 22, 22, 22, 23, 23, 23, 24, 24, 24, ~
$ group     <chr> "WT C", "WT C", "WT C", "WT C", "WT C", "WT C", "WT C", "WT ~
$ gene      <chr> "AKT", "AKT", "AKT", "AKT", "AKT", "AKT", "AKT", "AKT", "AKT~
$ ct        <dbl> 16.49, 16.44, 16.65, 16.50, 16.74, 16.68, 16.96, 16.92, 17.0~
```

The function `glimpse` displays a summary of each column of your dataframe and their data types. We see that `sample_id` is an integer, `group` and `gene` are characters and finally the `ct` values are doubles, meaning they are numeric. So far the data looks good.

Another way to use the function `glimpse` and other functions from the `dplyr` package is using the pipe `%>%`. The way to interpret the pipe is the following. Given a dataframe, we pipe the dataframe to the next function. The syntax is shown in the code block below.

```
qpcr %>% glimpse
```

```
Rows: 72
Columns: 4
$ sample_id <int> 20, 20, 20, 21, 21, 21, 22, 22, 22, 23, 23, 23, 24, 24, 24, ~
$ group     <chr> "WT C", "WT C", "WT C", "WT C", "WT C", "WT C", "WT C", "WT ~
```

```
$ gene      <chr> "AKT", "AKT", "AKT", "AKT", "AKT", "AKT", "AKT", "AKT", "AKT~
$ ct        <dbl> 16.49, 16.44, 16.65, 16.50, 16.74, 16.68, 16.96, 16.92, 17.0~
```

The output is the same as before, the difference is that we can use other functions and chain them together. For example, if we want to see only the first 5 rows of the table and then select the row with maximum ct value, we can use the function `head` followed by `slice_max`.

```
qpcr %>%
    head %>%
    slice_max(order_by = ct) # specify the column by passing the name without quotes
```

```
  sample_id group gene    ct
1        21  WT C  AKT 16.74
```

There is a flow of code when chaining functions like this. By also writing code like this, it is easier to modify or add new steps to the chain.

2.4 Checkpoint

We now use `readRDS` to save our current matrix to load it up in the next chapter. The checkpoints are not necessary. As this is a simple analysis, you can run everything in a single script and loaded in your memory. The good thing about checkpoints is that you can save heavy calculations so you do not need to perform them again.

```
saveRDS(qpcr, "../checkpoints/loading/qpcr.rds")
```

> **Tip**
>
> `readRDS` is a function that lets you save R objects into your computer. It is extremely handy when you want to save expensive calculations or continue your analysis later. Note you are only saving one object.

> **Tip**
>
> If you are using quarto markdown or Rmarkdown, there is a chunk option that you can use to not rerun that chunk, namely the `cache` option.

For each checkpoint, we save them in a specific folder for each chapter inside `checkpoints`, in the root folder. This ensures we know where the data was generated by pointing to the name of the chapter.

3 Cleaning your data

In this chapter I will show how you can clean your data before you start any analysis. Doing this before anything else will save you a lot of time. It is very frequent that you need to change a variable name or create a new column in your dataframe. By performing the cleaning and modifications before doing any plotting, it helps to ensure you will be using clean data and it will make your life easier.

3.1 Loading libraries and files

As discussed before, we start by loading the libraries we will use. Later I will describe what the `stringr` and `janitor` library do.

```
library(dplyr)
library(stringr)
library(janitor)
```

And we load up our qPCR data saved from the checkpoint. To do this we use the function `readRDS`. For this we only specify the path to where the checkpoint was saved.

```
qpcr <- readRDS("../checkpoints/loading/qpcr.rds")
```

3.2 Formatting your column names

Describe the use of janitor here.

3.3 Changing values in a column

describe the use of mutate from dplyr and the functions from stringr.

3.4 Checkpoint

explain what is being saved and where

Part II

Exploring

4 Preparing the data

Describe data needs to be in special format to do stuff in R, explain pivoting. Explain groupping as well, which makes it easier to do average calculations

4.1 Loading libraries and files

```
library(dplyr)
library(tidyr)
```

4.2 Pivoting

explain pivoting and why it is useful. mention plotting and filtering

4.3 Grouping data

explain the use of group_by from dplyr. show how useful it is by summarizing and filtering based on groups.

4.4 Checkpoint

save summarized data and pivotted data.

5 Plotting

Part III

Modeling

6

7

Part IV

Wrap up

8

Conclusion

References