Data Creation for Umbrella - Tutorial

Bart K. M. Jacobs

6 april 2017

==========

This tutorial gives a quick overview on how to create an RData file that can be used as input for the Umbrella procedure.

This example uses .csv files as input. With minor changes, it could also work on plain text formats. For all other formats, consider either saving the data as .csv or read up on ways to import data in R or RStudio.

We worked with Bio-Rad's QuantaSoft to extract our data. In QuantaSoft, fluorescence intensities of individual partitions can be exported under Setup -> Options -> Export Amplitude and Cluster Data. This creates a .csv file for every partition set (which is a well for Bio-Rad's system). In this tutorial, we start from such a group of .csv files that share their name with the exception of an indication of the partition set.

Download the 6 .csv files with names starting with `Experiment42_Date031415_SomeTechnicalDetails` in the same directory. Then, change the path in the following code so it refers to this directory before executing it.

```
setwd("C:/temp")
```

Note that the pattern of the filenames is typically:

- Some technical information
- Partition set identifier
- the word "Amplitude"

The partition set identifier for QuantaSoft is a letter followed by two digits.

The following code starts by creating an empty list. It then cycles through all the .csv files. It makes a separate R object (a dataframe or a data vector) for each partition set, and then adds this R object to the list. The final result is a list that has a single entry for each partition set and can be used as input for the Umbrella procedure.

The following is the full code for single channel data, where the data are assumed to be in the first column.

```
Tutorial.DataCreate.List1d <- list()
ltr <- c("A","B","C","D","E","F","G","H")
for(i in 1:2){
for(j in 1:3){
fname <- paste("Experiment42_Date031415_SomeTechnicalDetails_",ltr[i],sprintf("%02d",j),
               "_Amplitude.csv",sep="")
name <- paste("Well_",ltr[i],sprintf("%02d",j),sep="")
try(assign(name,read.csv(fname,header=T)))
try(singlepartset <- list(get(name)[,1]))
try(names(singlepartset) <- name)
try(Tutorial.DataCreate.List1d <- append(Tutorial.DataCreate.List1d,singlepartset))
try(rm(singlepartset))
}}
rm(name,fname)
rm(ltr,i,j)
```

We can check that the list looks correct:

```
str(Tutorial.DataCreate.List1d)
```

```
## List of 6
##  $ Well_A01: num [1:20000] 2086 1821 1744 2198 8712 ...
##  $ Well_A02: num [1:20000] 1827 8848 1710 2018 1530 ...
##  $ Well_A03: num [1:20000] 2139 1720 2018 2376 7485 ...
##  $ Well_B01: num [1:20000] 1919 1843 2268 2334 2020 ...
##  $ Well_B02: num [1:20000] 2159 2012 2182 1485 2086 ...
##  $ Well_B03: num [1:20000] 1725 1989 1763 2270 2042 ...
```

The following is the full code for multi-channel data, where the intensities are assumed to be in the first columns. An optional clustering may be included as well. Note that Umbrella itself can currently only deal with one-dimensional and two-dimensional data.

```
Tutorial.DataCreate.ListNd <- list()
ltr <- c("A","B","C","D","E","F","G","H")
for(i in 1:2){
for(j in 1:3){
fname <- paste("Experiment42_Date031415_SomeTechnicalDetails_",ltr[i],sprintf("%02d",j),
               "_Amplitude.csv",sep="")
name <- paste("Well_",ltr[i],sprintf("%02d",j),sep="")
try(assign(name,read.csv(fname,header=T)))
try(singlepartset <- list(get(name)))
try(names(singlepartset) <- name)
try(Tutorial.DataCreate.ListNd <- append(Tutorial.DataCreate.ListNd,singlepartset))
try(rm(singlepartset))
}}
rm(name,fname)
rm(ltr,i,j)
```

We can also check that this list looks correct:

```
str(Tutorial.DataCreate.ListNd)
```

```
## List of 6
##  $ Well_A01:'data.frame':    20000 obs. of  2 variables:
##   ..$ Ch1.Amplitude: num [1:20000] 2086 1821 1744 2198 8712 ...
##   ..$ Ch2.Amplitude: num [1:20000] 2000 8079 8130 2191 2011 ...
##  $ Well_A02:'data.frame':    20000 obs. of  2 variables:
##   ..$ Ch1.Amplitude: num [1:20000] 1827 8848 1710 2018 1530 ...
##   ..$ Ch2.Amplitude: num [1:20000] 7012 2249 1594 8209 8029 ...
##  $ Well_A03:'data.frame':    20000 obs. of  2 variables:
##   ..$ Ch1.Amplitude: num [1:20000] 2139 1720 2018 2376 7485 ...
##   ..$ Ch2.Amplitude: num [1:20000] 1666 1897 7974 2106 6326 ...
##  $ Well_B01:'data.frame':    20000 obs. of  2 variables:
##   ..$ Ch1.Amplitude: num [1:20000] 1919 1843 2268 2334 2020 ...
##   ..$ Ch2.Amplitude: num [1:20000] 2119 1757 2315 1829 2280 ...
##  $ Well_B02:'data.frame':    20000 obs. of  2 variables:
##   ..$ Ch1.Amplitude: num [1:20000] 2159 2012 2182 1485 2086 ...
##   ..$ Ch2.Amplitude: num [1:20000] 2105 1694 1759 1889 2155 ...
##  $ Well_B03:'data.frame':    20000 obs. of  2 variables:
##   ..$ Ch1.Amplitude: num [1:20000] 1725 1989 1763 2270 2042 ...
##   ..$ Ch2.Amplitude: num [1:20000] 1904 1915 1841 1809 1659 ...
```

Step by step, the code does the following:

- Create an empty list

- Create a variable with letters. ltr[1] is now the first letter, i.e. A
- Loop over the different letters. Here, we only have A and B, so i goes from 1 to 2.
- Loop over the different numbers. We have from 1 to 3 here. Note that in this case the same numbers appear for each letter.
  If this is not the case, the code still works if all numbers that appear are included. All nonexisting combinations will get skipped and several error messages will be shown for each nonexisting combination. The final result will still be a list with all existing combinations.

Then, for each partition set the following happens:

- Create an object (`fname`) that stores the name of the file.
- Create an object (`name`) that stores the (future) name of the partition set in the list.
- Read the .csv file and assign the data to an object that can be called from the name object. E.g.: if name = "Well_A01", then the object Well_A01 now stores the data from this partition set. The `try` part of the command makes sure that the code doesn't stop at this time when in case the file doesn't exist. If you use this code for other data: change the name according to the file name at hand.
- Extract the data in the format that we want. This is the only step where the single channel and multichannel extraction are different.
- Assign a name to the data
- Add the data to the list

The last steps are about removing objects are are not needed, or shouldn't exist anymore, later.

This code can be copied as is for fluorescence intensities exported from QuantaSoft. We may extend this tutorial with file formats from other systems in the future. Feel free to contact us in meanwhile.