mpxtractor: A flexible R package to process data from different plate reader machines

by Martin Banchero

Student number: 2589681

Major internship thesis Bioinformatics and Systems biology

Specialization: Systems Biology

Course: XM_0071(36 EC)

Supervisors AIMMS/VU

Supervisor: Douwe Molenaar PhD

mpxtractor: A flexible R package to process data from different plate reader machines

Abstract

Nowadays, high throughput experiments using plate reader machines are used frequently. These devices allow massive parallel experiments that generate large volumes of raw data. To analyze this type of data automated systems are required to extract relevant results. However, there are considerable unresolved issues in this type of technology. Here I focus on those concerns to data analysis and the lack of the standards to integrate raw data from plate reader machines. To address these problems, Here I introduce an R package call mpxtractor to deal with large raw data from plate reader machines, capable of extract metadata and combine this information with experimental design files generating tidy data [1] . Also mpxtractor is capable of producing a quick visualization of experimental designs contained in the layout files, and plot growth curves for each micro plate well. The package focuses on SpectraMax [2], MultiscanGO [3], FluorStar [4] plate reader machines.
The results shown that the package can produce similar results to those obtained from custom scripts but in a faster and more concise way. The main goal of mpxtractor then is to help to reduce the time to analyzed data leaving more time to focus in downstream data analysis within high throughput experiments using plate reader machines.

1 Introduction

In these days it is possible to perform experiments testing several samples simultaneously as a part of high-throughput experimentation. This can be done by automated devices like plate reader machines. These machines can measure simultaneous changes across multiple samples . The measured changes can be for example absorbance, luminescence, fluorescence and expressed in output files as optical densities or fluorescence. Although these machines allow the automated analysis of hundreds of samples, there are two major problems to consider, technical errors, and the large volume of raw data[5]. While the technical errors are intrinsically part of experimental processes, for example missing values can be generated, the raw data must be processed in order to extract useful information. Furthermore, each plate reader can generate output files with different formats depending on settings that the user selects

generating a lack of standardization in the output file format. This adds to the task of data analysis one more degree of complexity. The combination of all these factors produces a shortfall of generalization in the strategy to follow during the data analysis of plate reader data, contributing to the lack of experimental reproducibility and making the task of data analysis tedious and time-consuming.
Several methods were developed in recent years to deal with plate reader data. Most of these focus on specific parts of the data analysis pipeline, for example, plater [6] . This package is able to read experimental design files or layout files and also the measurements obtained by the plate reader. Although this package is robust there is a lack of automation, the data from the raw files has to be copy-pasted into a file before using it as input for one of the functions of the package. Other packages like cellHTS [7] receive processed data as input, this means without including the metadata.
As mentioned before standard output files are needed, same reader machines produce different output file formats that are difficult to parse, and also produce different file extensions like txt, csv, or xlsx making the integration of different sources of information very tedious. The main example of this is that the output files do not contain any information about the experimental design. This information usually is stored in layout files designed by the experimentalist. These files contain a description of the location and content of each well indicating if is it a control or a sample and including control variables, like concentration or pH.
At this point is clear that having standard outputs files from microplate reader machines can lead to a faster and robust way to combine different sources of information.
In order to attend the issues mentioned before the main goal of this project is to develop a flexible R package call **mpxtractor** to deal with the data produced by plate readers. In our laboratory these are spectraMax[2], multiscanGO [3] and Fluorstar [4] . The main feature of **mpxctractor** lies in the transformation of raw data in a txt file format into a tibble object, which is a modern version of a data frame and stored with a tidy data format [1] . In principle, this tidy format allows a reduction of the time needed for data analysis and is also easier to combine with other sources of information i.e experimental design files or layout files. Furthermore, tibbles brings more options for re-formatting and analysis of the data available in tidyverse collection of packages for data analysis.

The package **mpxtractor** contains one import and wrangling function for each type of reader machine mentioned before. It also contains a function to read multiple files which is implemented as a wrapper around the import functions.
Another characteristic of mpxtractor is that it can generate quick visualizations of layout files in a plate format, which can be printed to be used in the lab. This function works as a wrapper with the function read_layout_file(), which is similar to the function read_table() from the plater package [6]. The package also has functions for the computation and graphical representation of growth rate curves per well mimicking a plate-shaped format. The growth rates are calculated using a Savitzky-Golay filter [8] implemented in R. This filter allows an improvement of the graphical output by smoothing the data, and also computes the growth rates by taking the first-order derivative of the smoothed data.
The steps followed by the package are shown in **Figure 1**. The pipeline starts with the import and wrangling of raw data files with a *.txt* format and generates a tibble which can be combined with experimental design files in a *.csv* format. Also, the layout files can be read directly into a tibble. All the tibbles can be exported by the user. Furthermore, the layout files and the growth curves can be plotted and exported in order to obtain a quick visualization.
In short, it is expected that the **mpxtractor** package allows users to focus on more specific methods downstream in the data analysis pipeline rather than spending time on reading the layout files and raw data, and integrating these to obtain a data object suitable for further analysis.

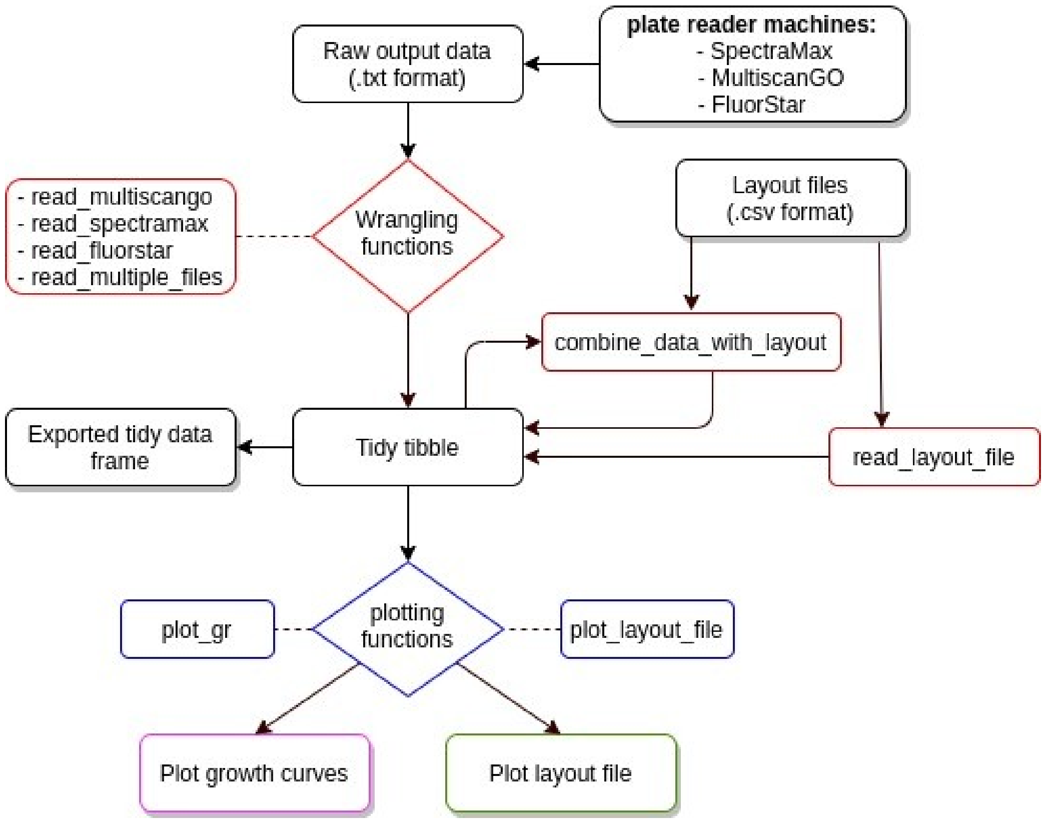


Figure 1: Flowchart mpxtractor. In red are shown the wrangling functions, in blue plotting functions. In magenta the outputs of ***plot_gr()*** and in green the output of ***plot_layout_file().*** Note that the tidy data frame can be exported. The input files for mpxtractor are *.txt* for data files and *.csv* for layout files.

2 Methods

The functions of **mpxtractor** package are divided in two groups (see **Figure 1**), one group of functions is involved in data wrangling to deal with raw data. The other group of functions is for plotting and data visualization.
To demonstrate the functionalities of these functions I used data generated during different experiments using plate readers at the lab. The original files containing raw data were processed (not shown) to

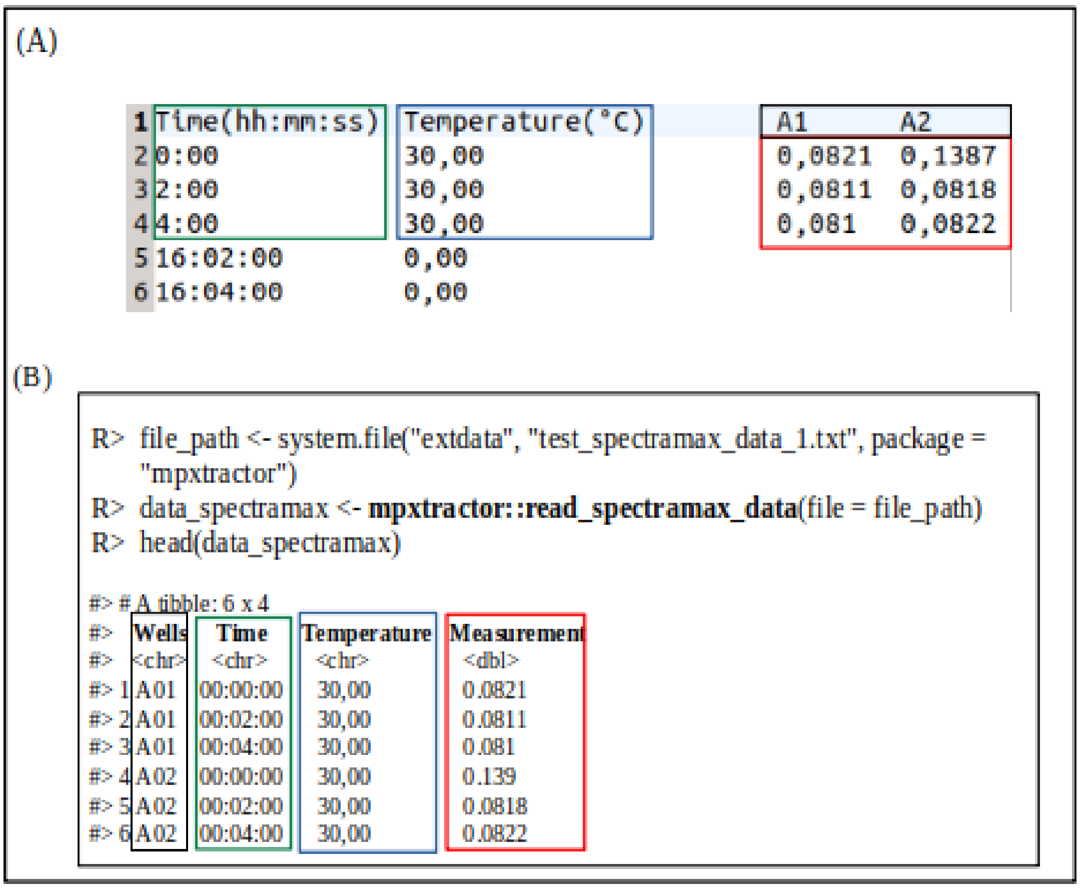
keep a reasonable size to show some examples. These small data files are available through the package repository[9].

2.1 Import and wrangling functions

The mpxtractor package provide one function to read raw data specific for each reader machine, this are *read_spectramax_data()*, *read_multiscango_data()* and *read_fluorstar_data()*. Furthermore, to read multiple files the function *read_multiple_files() can be used. For this function,* the type of machine that generates the files must be given as an argument of the function. Also, there is a function *combine_data_with_layout()* to combine raw files with layout files. Moreover, the function *read_layout_file()* can be use to read the layout file into a tibble. Next, each of these functions is explained in detail.

2.1.1 read_spectramax_data()

This function reads output files (.txt) produced by spectraMax readers [2] . In **Figure 2.a** the caption of the raw file is shown. In **Figure 2.b** the code to run the function is followed by a few lines of the output data frame. Note that the Well ids are transformed by zero-padding between the letter and the number, this is done mainly to avoid R to sort wells by using alphabetical order. The time format is also modified to a more complete time representation in hh:mm:ss. The measurements are gathered using the function *gather()* from tidyr package, with Wells as key and Measurement as value.


(A)

```
1 Time(hh:mm:ss)  Temperature(°C)          A1       A2
2 0:00            30,00                    0,0821   0,1387
3 2:00            30,00                    0,0811   0,0818
4 4:00            30,00                    0,081    0,0822
5 16:02:00        0,00
6 16:04:00        0,00
```

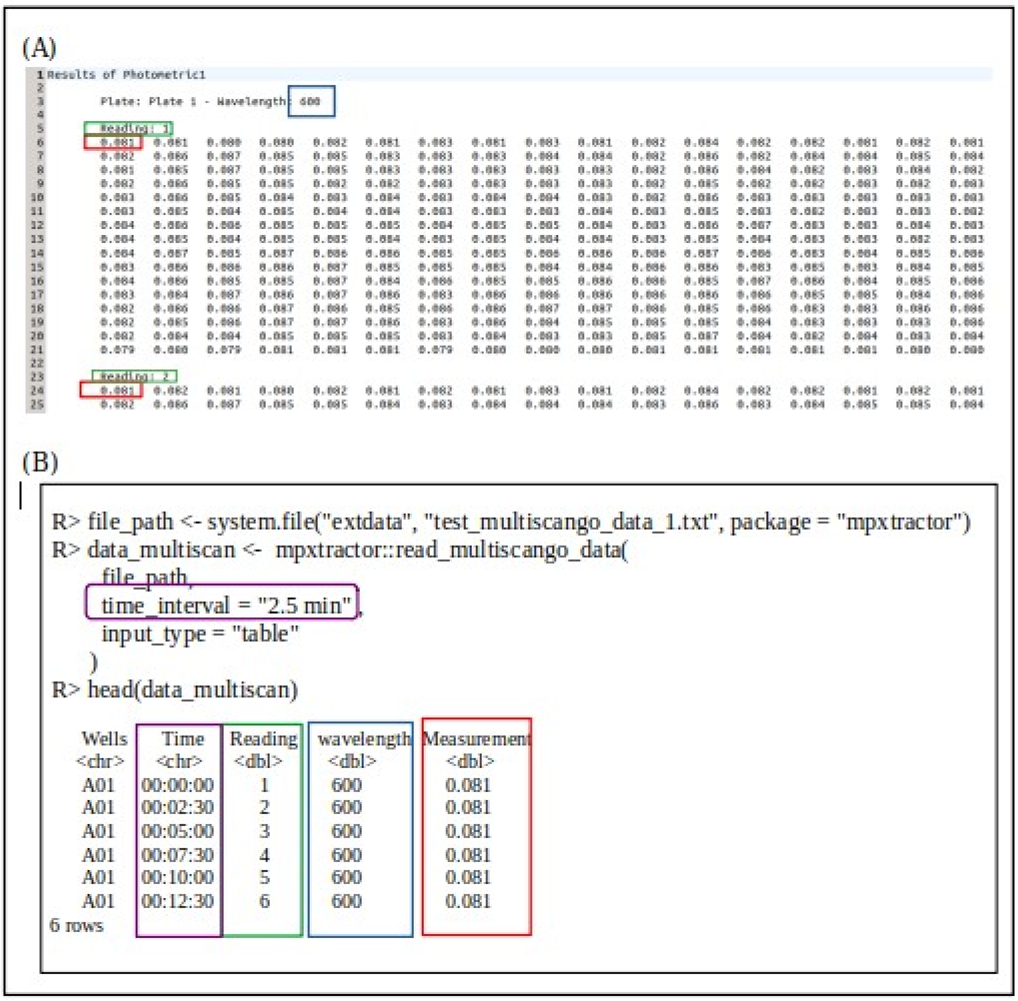
(B)

```
R> file_path <- system.file("extdata", "test_spectramax_data_1.txt", package =
   "mpxtractor")
R> data_spectramax <- mpxtractor::read_spectramax_data(file = file_path)
R> head(data_spectramax)

#> # A tibble: 6 x 4
#>   Wells  Time      Temperature  Measurement
#>   <chr>  <chr>     <chr>        <dbl>
#> 1 A01    00:00:00  30,00        0.0821
#> 2 A01    00:02:00  30,00        0.0811
#> 3 A01    00:04:00  30,00        0.081
#> 4 A02    00:00:00  30,00        0.139
#> 5 A02    00:02:00  30,00        0.0818
#> 6 A02    00:04:00  30,00        0.0822
```

Figure 2: read_spectramax_data function. (A) Caption of raw data file. **(B)** R code example to run *read_spectramax_data()* follow by the print of few lines of the output data frame. The different colors show the origin of the data stored in the data frame.

2.1.2 read_multiscango_data()

This function reads two types of output files (.txt) produced by multiscanGO plate readers [3]. The type of output file is defined by the user before export the raw data files from multiscango machines. One format is *table,* this shows the measurements for each reading in a microplate shape. In **Figure 3.a** the caption of the table raw file is shown. In **Figure 3.b** the code to run the function table format followed by a few lines of the output data frame. Note that the argument time_interval in magenta is needed because of the absence of this value in the metadata of the table raw file. This argument has to be in minutes for the sake of simplicity. Also, the argument has to be provided by the user. The time series is generated by multiplying the time interval by the maximum number of readings minus one.



(A)

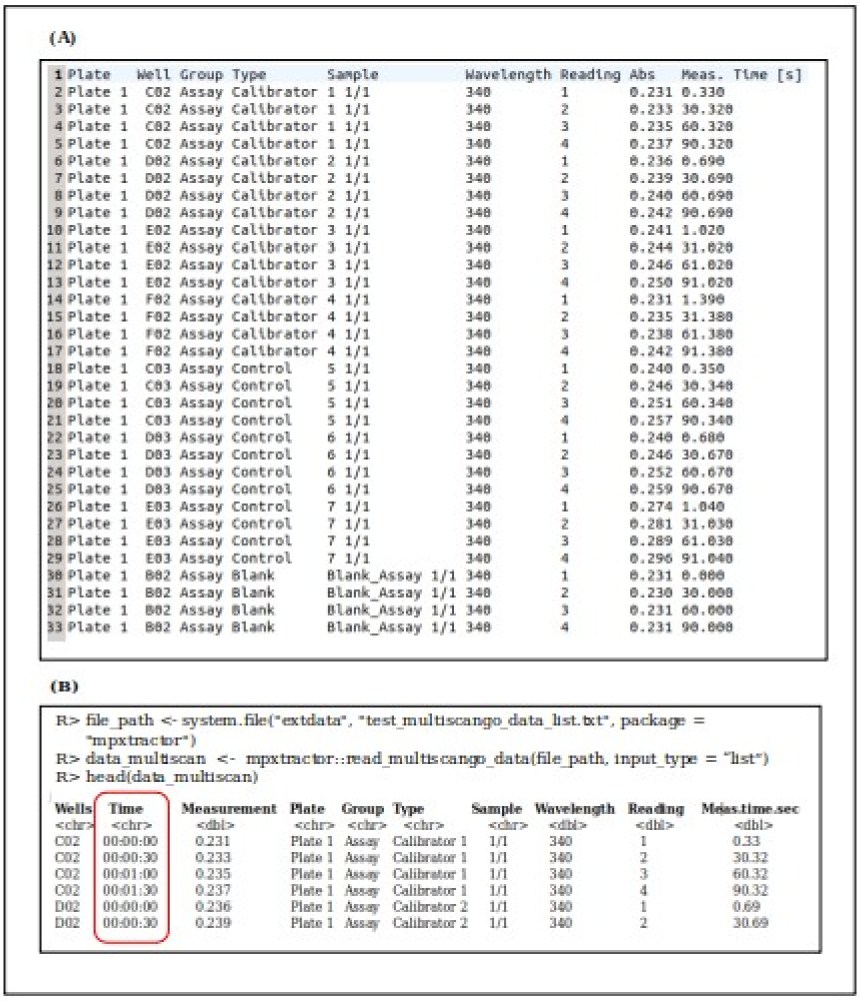
```
Results of Photometric1

        Plate: Plate 1 - Wavelength: 600

    Reading: 1
        0.081  0.081  0.080  0.080  0.082  0.081  0.083  0.081  0.083  0.081  0.082  0.084  0.082  0.082  0.081  0.082  0.081
        0.082  0.086  0.087  0.085  0.085  0.083  0.083  0.083  0.084  0.084  0.082  0.086  0.082  0.084  0.084  0.085  0.084
        0.081  0.085  0.087  0.085  0.085  0.083  0.083  0.083  0.083  0.083  0.082  0.086  0.084  0.082  0.083  0.084  0.082
        0.082  0.086  0.085  0.085  0.082  0.082  0.083  0.083  0.083  0.083  0.082  0.085  0.082  0.082  0.083  0.082  0.083
        0.083  0.086  0.085  0.084  0.083  0.084  0.083  0.084  0.084  0.083  0.082  0.086  0.083  0.083  0.083  0.083  0.083
        0.083  0.085  0.084  0.085  0.084  0.084  0.083  0.083  0.083  0.084  0.083  0.085  0.083  0.082  0.083  0.083  0.082
        0.084  0.086  0.086  0.085  0.085  0.085  0.084  0.085  0.085  0.084  0.083  0.086  0.087  0.083  0.083  0.084  0.083
        0.084  0.085  0.084  0.085  0.085  0.084  0.083  0.085  0.084  0.084  0.083  0.085  0.084  0.083  0.083  0.082  0.083
        0.084  0.087  0.085  0.087  0.086  0.086  0.085  0.085  0.086  0.086  0.086  0.087  0.086  0.083  0.084  0.085  0.086
        0.083  0.086  0.086  0.086  0.087  0.085  0.085  0.085  0.084  0.084  0.086  0.086  0.083  0.085  0.083  0.084  0.085
        0.084  0.086  0.085  0.085  0.087  0.084  0.086  0.085  0.085  0.086  0.086  0.085  0.087  0.086  0.084  0.085  0.086
        0.083  0.084  0.087  0.086  0.087  0.086  0.083  0.086  0.086  0.086  0.086  0.086  0.086  0.085  0.085  0.084  0.086
        0.082  0.086  0.086  0.087  0.086  0.085  0.086  0.086  0.087  0.087  0.086  0.085  0.086  0.083  0.083  0.086  0.086
        0.082  0.085  0.086  0.087  0.087  0.086  0.083  0.086  0.084  0.085  0.085  0.085  0.084  0.083  0.083  0.083  0.086
        0.082  0.084  0.084  0.085  0.085  0.085  0.083  0.084  0.083  0.083  0.085  0.087  0.084  0.082  0.084  0.083  0.084
        0.079  0.080  0.079  0.081  0.081  0.081  0.079  0.080  0.080  0.080  0.081  0.081  0.081  0.081  0.081  0.080  0.080

    Reading: 2
        0.081  0.082  0.081  0.080  0.082  0.081  0.082  0.081  0.083  0.081  0.082  0.084  0.082  0.082  0.081  0.082  0.081
        0.082  0.086  0.087  0.085  0.085  0.084  0.083  0.084  0.084  0.084  0.083  0.086  0.083  0.084  0.085  0.085  0.084
```

(B)

```
R> file_path <- system.file("extdata", "test_multiscango_data_1.txt", package = "mpxtractor")
R> data_multiscan <-  mpxtractor::read_multiscango_data(
     file_path,
     time_interval = "2.5 min",
     input_type = "table"
   )
R> head(data_multiscan)

   Wells    Time    Reading  wavelength  Measurement
   <chr>    <chr>    <dbl>     <dbl>       <dbl>
    A01   00:00:00     1        600        0.081
    A01   00:02:30     2        600        0.081
    A01   00:05:00     3        600        0.081
    A01   00:07:30     4        600        0.081
    A01   00:10:00     5        600        0.081
    A01   00:12:30     6        600        0.081
6 rows
```

Figure 3: read_multiscango_data function. (A) Caption of raw file table format. **(B)** R code example to run *read_multiscango_data()* follow by the print of few lines of the output data frame. The different colors show the origin of the data stored in the data frame.

The other possible output file format of multiscango is list. Compared to the table format this file contains more information, i.e time at which the measurements are done in seconds, well id, type indicating the content of the well, for example if is a blank or control. **Figure 4.a** shows the caption of the raw file with format list. **Figure 4.b** shows the code to run the multiscan function taking the raw data files in list format followed by few lines of the output data frame.


(A)

```
Plate    Well Group Type       Sample            Wavelength Reading Abs   Meas. Time [s]
Plate 1  C02 Assay Calibrator 1 1/1              340        1       0.231 0.330
Plate 1  C02 Assay Calibrator 1 1/1              340        2       0.233 30.320
Plate 1  C02 Assay Calibrator 1 1/1              340        3       0.235 60.320
Plate 1  C02 Assay Calibrator 1 1/1              340        4       0.237 90.320
Plate 1  D02 Assay Calibrator 2 1/1              340        1       0.236 0.690
Plate 1  D02 Assay Calibrator 2 1/1              340        2       0.239 30.690
Plate 1  D02 Assay Calibrator 2 1/1              340        3       0.240 60.690
Plate 1  D02 Assay Calibrator 2 1/1              340        4       0.242 90.690
Plate 1  E02 Assay Calibrator 3 1/1              340        1       0.241 1.020
Plate 1  E02 Assay Calibrator 3 1/1              340        2       0.244 31.020
Plate 1  E02 Assay Calibrator 3 1/1              340        3       0.246 61.020
Plate 1  E02 Assay Calibrator 3 1/1              340        4       0.250 91.020
Plate 1  F02 Assay Calibrator 4 1/1              340        1       0.231 1.390
Plate 1  F02 Assay Calibrator 4 1/1              340        2       0.235 31.380
Plate 1  F02 Assay Calibrator 4 1/1              340        3       0.238 61.380
Plate 1  F02 Assay Calibrator 4 1/1              340        4       0.242 91.380
Plate 1  C03 Assay Control    5 1/1              340        1       0.240 0.350
Plate 1  C03 Assay Control    5 1/1              340        2       0.246 30.340
Plate 1  C03 Assay Control    5 1/1              340        3       0.251 60.340
Plate 1  C03 Assay Control    5 1/1              340        4       0.257 90.340
Plate 1  D03 Assay Control    6 1/1              340        1       0.240 0.680
Plate 1  D03 Assay Control    6 1/1              340        2       0.246 30.670
Plate 1  D03 Assay Control    6 1/1              340        3       0.252 60.670
Plate 1  D03 Assay Control    6 1/1              340        4       0.259 90.670
Plate 1  E03 Assay Control    7 1/1              340        1       0.274 1.040
Plate 1  E03 Assay Control    7 1/1              340        2       0.281 31.030
Plate 1  E03 Assay Control    7 1/1              340        3       0.289 61.030
Plate 1  E03 Assay Control    7 1/1              340        4       0.296 91.040
Plate 1  B02 Assay Blank      Blank_Assay 1/1 340        1       0.231 0.000
Plate 1  B02 Assay Blank      Blank_Assay 1/1 340        2       0.230 30.000
Plate 1  B02 Assay Blank      Blank_Assay 1/1 340        3       0.231 60.000
Plate 1  B02 Assay Blank      Blank_Assay 1/1 340        4       0.231 90.000
```

(B)

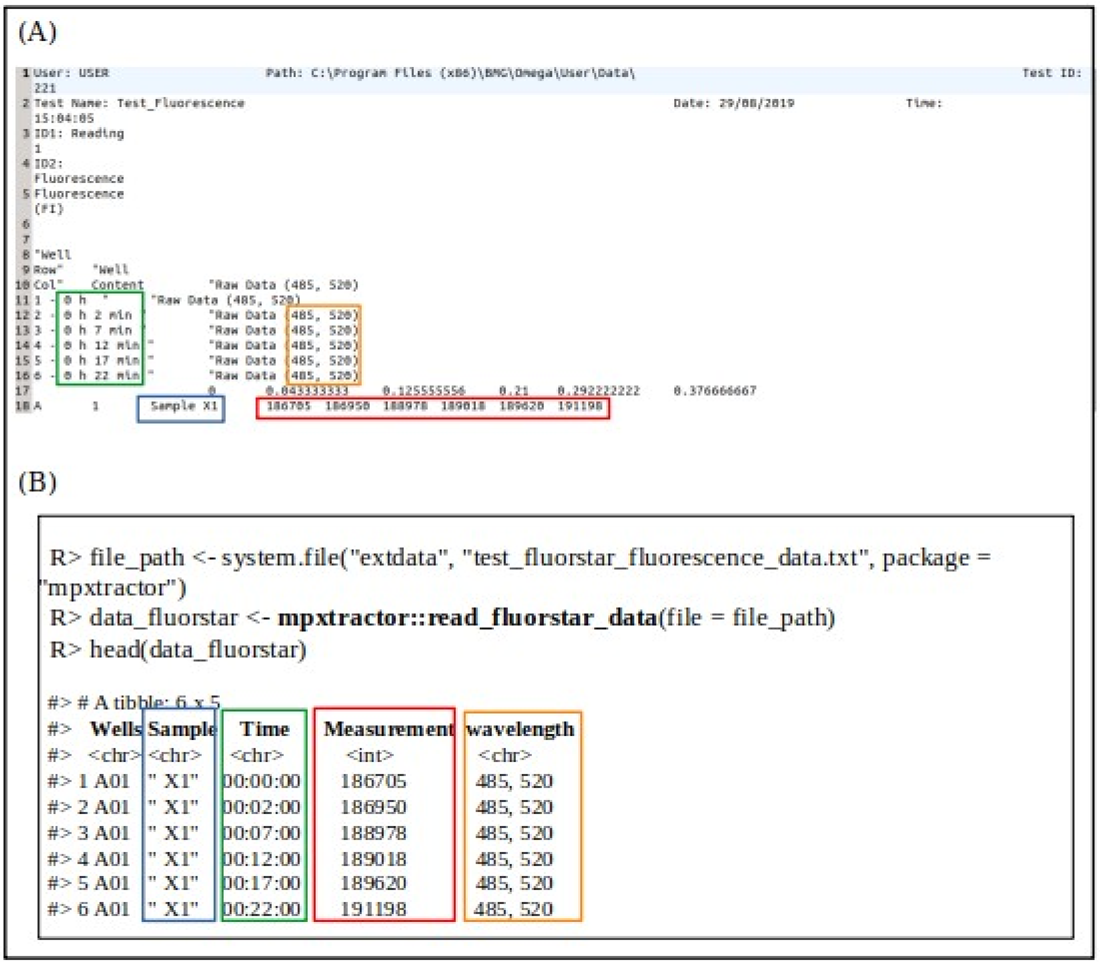
```
R> file_path <- system.file("extdata", "test_multiscango_data_list.txt", package =
    "mpxtractor")
R> data_multiscan  <-  mpxtractor::read_multiscango_data(file_path, input_type = "list")
R> head(data_multiscan)

Wells  Time      Measurement  Plate    Group  Type           Sample  Wavelength  Reading  Meas.time.sec
<chr>  <chr>     <dbl>        <chr>    <chr>  <chr>          <chr>   <dbl>       <dbl>    <dbl>
C02    00:00:00  0.231        Plate 1  Assay  Calibrator 1   1/1     340         1        0.33
C02    00:00:30  0.233        Plate 1  Assay  Calibrator 1   1/1     340         2        30.32
C02    00:01:00  0.235        Plate 1  Assay  Calibrator 1   1/1     340         3        60.32
C02    00:01:30  0.237        Plate 1  Assay  Calibrator 1   1/1     340         4        90.32
D02    00:00:00  0.236        Plate 1  Assay  Calibrator 2   1/1     340         1        0.69
D02    00:00:30  0.239        Plate 1  Assay  Calibrator 2   1/1     340         2        30.69
```

Figure 4: multiscango list data format. **(A)** Caption of raw file list format. **(B)** R code example to run *read_multiscango_data()* follow by the print of few lines of the output data frame. In red the new column with time in format hh:mm:ss.

In this function the missing values, in the case to be present in column "Measurement" are imputed by taking the mean between the two elements surrounding the center using the R package imputeTS [10].

2.1.3 read_fluorstar_data()

This function reads output files (.txt) produced by fluorstar readers [4]. In **Figure 5.a** the caption of the raw input file is shown. In **Figure 5.b** the code to run the function follow by few lines of the output data frame. The function generates a new time format, in hh:mm:ss using the time data in the file and also transformed the format of Well ids by zero-padding between the letter and the number.



(A)

```
User: USER                    Path: C:\Program Files (x86)\BMG\Omega\User\Data\                          Test ID:
221
Test Name: Test_Fluorescence                                            Date: 29/08/2019           Time:
15:04:05
ID1: Reading
1
ID2:
Fluorescence
Fluorescence
(FI)


"Well
Row"    "Well
Col"    Content          "Raw Data (485, 520)
1 - 0 h   "       "Raw Data (485, 520)
2 - 0 h 2 min "          "Raw Data (485, 520)
3 - 0 h 7 min "          "Raw Data (485, 520)
4 - 0 h 12 min "         "Raw Data (485, 520)
5 - 0 h 17 min "         "Raw Data (485, 520)
6 - 0 h 22 min "         "Raw Data (485, 520)
                         0    0.043333333    0.125555556    0.21    0.292222222    0.376666667
A       1      Sample X1     186705  186950  188978  189018  189620  191198
```

(B)

```
R> file_path <- system.file("extdata", "test_fluorstar_fluorescence_data.txt", package =
"mpxtractor")
R> data_fluorstar <- mpxtractor::read_fluorstar_data(file = file_path)
R> head(data_fluorstar)

#> # A tibble: 6 x 5
#>   Wells Sample   Time     Measurement  wavelength
#>   <chr> <chr>    <chr>      <int>        <chr>
#> 1 A01   " X1"   00:00:00    186705       485, 520
#> 2 A01   " X1"   00:02:00    186950       485, 520
#> 3 A01   " X1"   00:07:00    188978       485, 520
#> 4 A01   " X1"   00:12:00    189018       485, 520
#> 5 A01   " X1"   00:17:00    189620       485, 520
#> 6 A01   " X1"   00:22:00    191198       485, 520
```

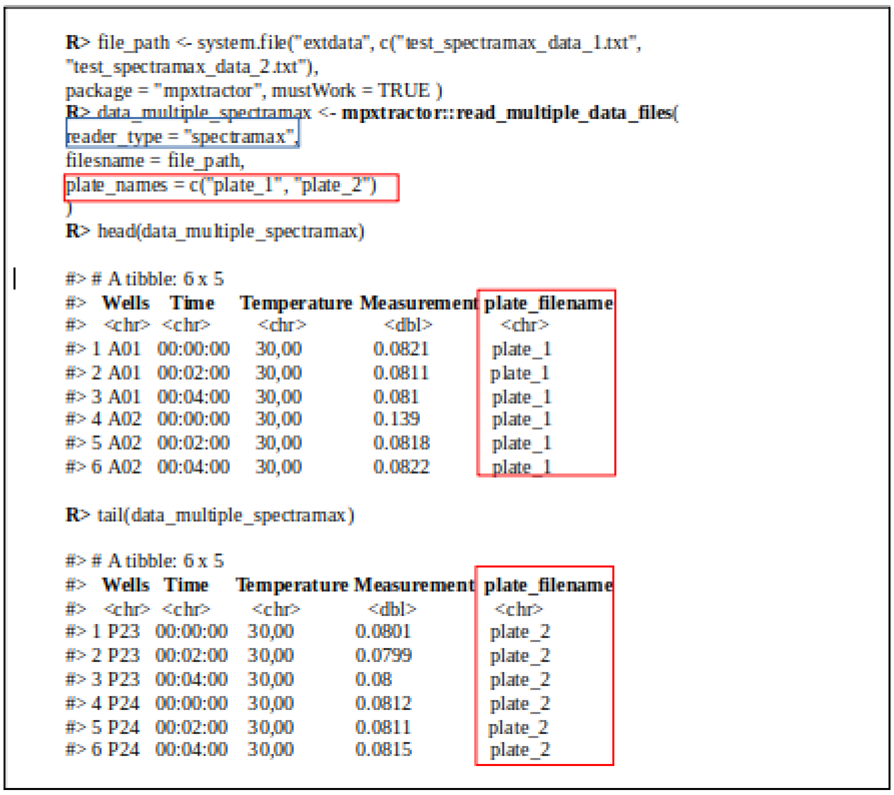
Figure 5: read_fluorstar_data(): (A) Caption of raw file. **(B)** R code example to run *read_fluorstar_data()* follow by the print of few lines of the output data frame. The different colors show the origin of the data store in the data frame.

2.1.4 read_multiple_files()

The purpose of this function is to combine multiple raw files into a tidy data frame. This function is a wrap around the wrangling functions explained in the previous sub-sections. It generates a list of data frames that are bound to each other using the dplyr function bind_rows().

There are several parameter options in this function to load the raw files into R, for example, to read the files in one directory using

dirFiles, *file_pattern* pattern to match filenames, *filesname* names of the files to be read. Also, the type of plate reader has to be specified in the argument *reader_type,* this is important to call the proper import and wrangling function.
In case that the reader_type is multiscango, there are two extra parameters, depending on the type of multiscango raw file one can be skipped. For raw files with table format, *time_interval* is needed, this a string indicating the size of the time interval followed by the word min, ie. "2 min". This argument has to be always in minutes because for many raw files used as reference to develop this function, the time intervals are in the order of minutes. The last parameter to read several multiscango files is *input_type,* this is a not *NULL* parameter, and specifies whether the input file of multiscango is in table or list format.
The code to read two raw files is shown in **Figure 6**. Note that the name of the file can be added with the parameter *plate_names*, to keep the reference to the original files.

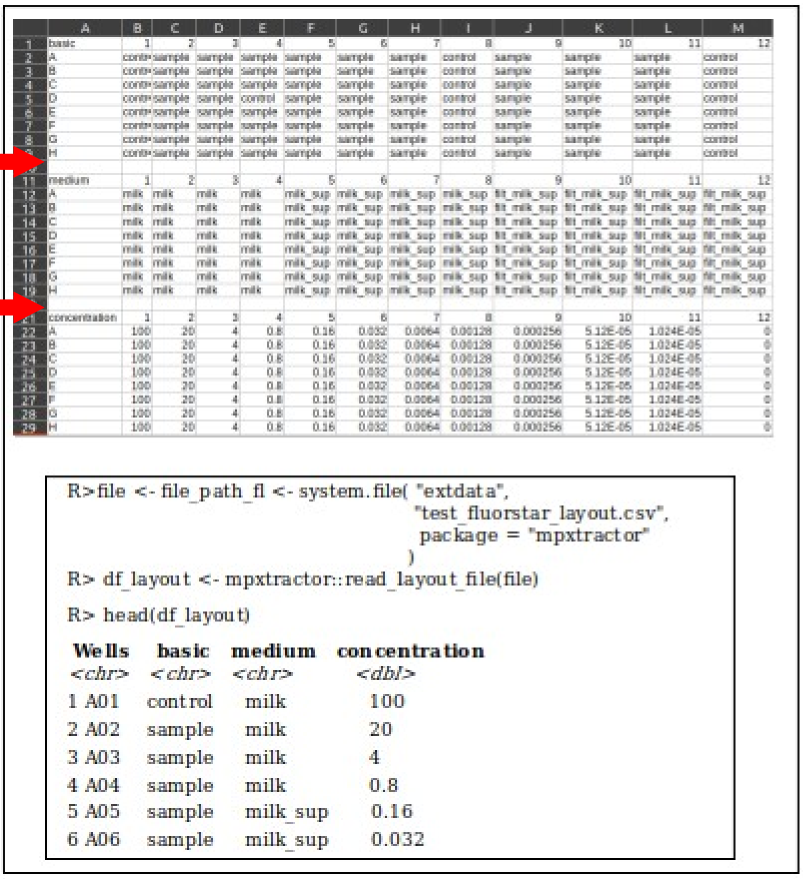


```
R> file_path <- system.file("extdata", c("test_spectramax_data_1.txt",
"test_spectramax_data_2.txt"),
package = "mpxtractor", mustWork = TRUE )
R> data_multiple_spectramax <- mpxtractor::read_multiple_data_files(
reader_type = "spectramax",
filesname = file_path,
plate_names = c("plate_1", "plate_2")
)
R> head(data_multiple_spectramax)

#> # A tibble: 6 x 5
#>   Wells  Time      Temperature Measurement plate_filename
#>   <chr>  <chr>       <chr>        <dbl>       <chr>
#> 1 A01   00:00:00   30,00        0.0821      plate_1
#> 2 A01   00:02:00   30,00        0.0811      plate_1
#> 3 A01   00:04:00   30,00        0.081       plate_1
#> 4 A02   00:00:00   30,00        0.139       plate_1
#> 5 A02   00:02:00   30,00        0.0818      plate_1
#> 6 A02   00:04:00   30,00        0.0822      plate_1

R> tail(data_multiple_spectramax)

#> # A tibble: 6 x 5
#>   Wells  Time      Temperature Measurement plate_filename
#>   <chr>  <chr>       <chr>        <dbl>       <chr>
#> 1 P23   00:00:00   30,00        0.0801      plate_2
#> 2 P23   00:02:00   30,00        0.0799      plate_2
#> 3 P23   00:04:00   30,00        0.08        plate_2
#> 4 P24   00:00:00   30,00        0.0812      plate_2
#> 5 P24   00:02:00   30,00        0.0811      plate_2
#> 6 P24   00:04:00   30,00        0.0815      plate_2
```

Figure 6: read_multiple_files() function. Note in blue the type of plate reader is specified as argument. In red the name referencing the raw file.

2.1.5 read_layout_files()

This function can be used to extract information from the layout files describing the experimental design into a tibble. The function receives a .csv file as its input. This means that one file represents only one plate and there is a layout for each of the variables used in the experiment. The separation between variables is one row. One example of the layout file is shown in **Figure 7.a**. The function read_layout_files() is similar to the function read_plate() from the *plater* package [6] with some modifications. The output data frame can be observed in **Figure 7.b** alongside with one code example.



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	basic	1	2	3	4	5	6	7	8	9	10	11	12
2	A	contr	sample	sample	sample	sample	sample	sample	control	sample	sample	sample	control
3	B	contr	sample	sample	sample	sample	sample	sample	control	sample	sample	sample	control
4	C	contr	sample	sample	sample	sample	sample	sample	control	sample	sample	sample	control
5	D	contr	sample	sample	control	sample	sample	sample	control	sample	sample	sample	control
6	E	contr	sample	sample	sample	sample	sample	sample	control	sample	sample	sample	control
7	F	contr	sample	sample	sample	sample	sample	sample	control	sample	sample	sample	control
8	G	contr	sample	sample	sample	sample	sample	sample	control	sample	sample	sample	control
9	H	contr	sample	sample	sample	sample	sample	sample	control	sample	sample	sample	control
10													
11	medium	1	2	3	4	5	6	7	8	9	10	11	12
12	A	milk	milk	milk	milk	milk_sup	milk_sup	milk_sup	milk_sup	filt_milk_sup	filt_milk_sup	filt_milk_sup	filt_milk_sup
13	B	milk	milk	milk	milk	milk_sup	milk_sup	milk_sup	milk_sup	filt_milk_sup	filt_milk_sup	filt_milk_sup	filt_milk_sup
14	C	milk	milk	milk	milk	milk_sup	milk_sup	milk_sup	milk_sup	filt_milk_sup	filt_milk_sup	filt_milk_sup	filt_milk_sup
15	D	milk	milk	milk	milk	milk_sup	milk_sup	milk_sup	milk_sup	filt_milk_sup	filt_milk_sup	filt_milk_sup	filt_milk_sup
16	E	milk	milk	milk	milk	milk_sup	milk_sup	milk_sup	milk_sup	filt_milk_sup	filt_milk_sup	filt_milk_sup	filt_milk_sup
17	F	milk	milk	milk	milk	milk_sup	milk_sup	milk_sup	milk_sup	filt_milk_sup	filt_milk_sup	filt_milk_sup	filt_milk_sup
18	G	milk	milk	milk	milk	milk_sup	milk_sup	milk_sup	milk_sup	filt_milk_sup	filt_milk_sup	filt_milk_sup	filt_milk_sup
19	H	milk	milk	milk	milk	milk_sup	milk_sup	milk_sup	milk_sup	filt_milk_sup	filt_milk_sup	filt_milk_sup	filt_milk_sup
20													
21	concentration	1	2	3	4	5	6	7	8	9	10	11	12
22	A	100	20	4	0.8	0.16	0.032	0.0064	0.00128	0.000256	5.12E-05	1.024E-05	0
23	B	100	20	4	0.8	0.16	0.032	0.0064	0.00128	0.000256	5.12E-05	1.024E-05	0
24	C	100	20	4	0.8	0.16	0.032	0.0064	0.00128	0.000256	5.12E-05	1.024E-05	0
25	D	100	20	4	0.8	0.16	0.032	0.0064	0.00128	0.000256	5.12E-05	1.024E-05	0
26	E	100	20	4	0.8	0.16	0.032	0.0064	0.00128	0.000256	5.12E-05	1.024E-05	0
27	F	100	20	4	0.8	0.16	0.032	0.0064	0.00128	0.000256	5.12E-05	1.024E-05	0
28	G	100	20	4	0.8	0.16	0.032	0.0064	0.00128	0.000256	5.12E-05	1.024E-05	0
29	H	100	20	4	0.8	0.16	0.032	0.0064	0.00128	0.000256	5.12E-05	1.024E-05	0

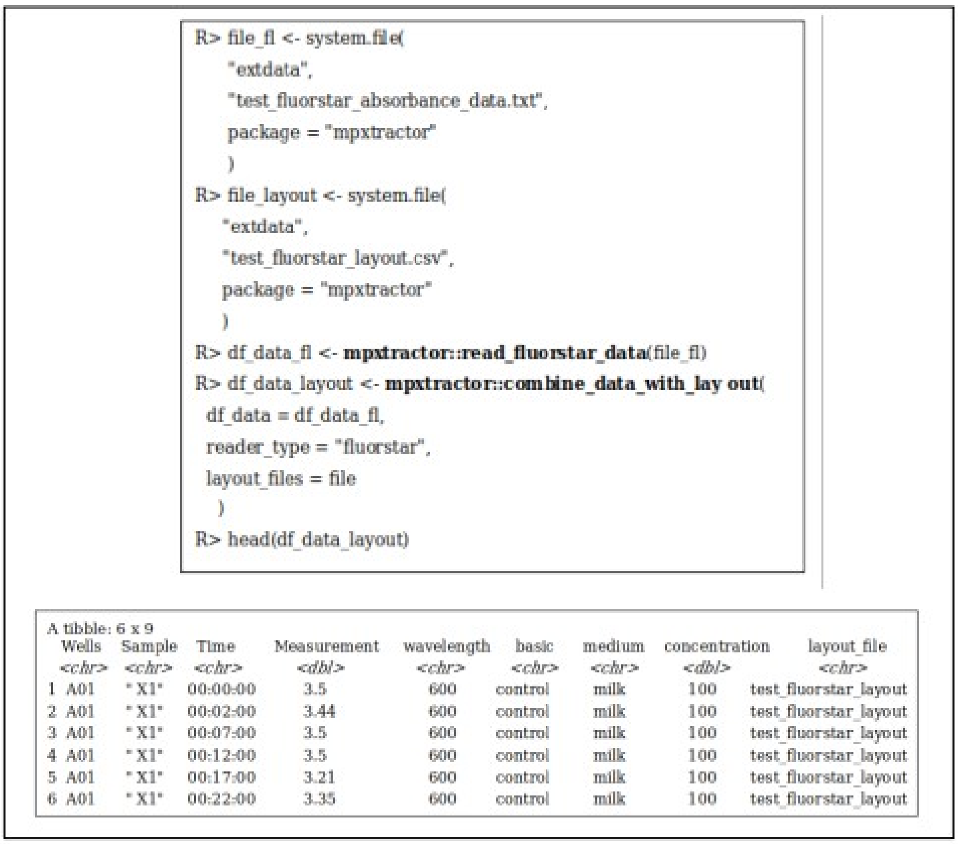
```
R>file <- file_path_fl <- system.file( "extdata",
                                       "test_fluorstar_layout.csv",
                                       package = "mpxtractor"
                                      )
R> df_layout <- mpxtractor::read_layout_file(file)
R> head(df_layout)
 Wells  basic   medium   concentration
 <chr>  <chr>   <chr>    <dbl>
1 A01   control  milk      100
2 A02   sample   milk      20
3 A03   sample   milk      4
4 A04   sample   milk      0.8
5 A05   sample   milk_sup  0.16
6 A06   sample   milk_sup  0.032
```

Figure 7: read_layout_file(). (A) Layout file properly formatted. The layout file represents only one plate and there is a layout for each of the variables used in the experiment Note that there is only one empty row between the layouts of each variable (red arrows). **(B)** R code example to run *read_layout_file()* follow by the print of few lines of the output data frame.

The output data frame shown in **Figure 7.b** can be combined with tidy data frames generated by the wrangling functions explained in the previous sections. This can be performed by the mpxtractor function ***combine_data_with_layout()*** that is explained in the following section.

2.1.6 combine_data_with_layout()

This function receives as an argument a data frame (data) which is produced by one of the functions (read_spectramax_data, read_multiscango_data, read_fluorstar_data or read_multiple_files). Also, the type of machine (spectramax, multiscango, or fluorstar) has to be specified alongside the path to the layout files.
In **Figure 8.a** the code to use the function is shown. Note that in this example the *df_data* argument receives the data frame shown in **Figure 5.b.** The layout file provided in this example is the one used for the example shown in **Figure 7.a.** In **Figure 8.b** is shown the the output data frame.



```
R> file_fl <- system.file(
     "extdata",
     "test_fluorstar_absorbance_data.txt",
     package = "mpxtractor"
     )
R> file_layout <- system.file(
    "extdata",
    "test_fluorstar_layout.csv",
    package = "mpxtractor"
    )
R> df_data_fl <- mpxtractor::read_fluorstar_data(file_fl)
R> df_data_layout <- mpxtractor::combine_data_with_lay out(
  df_data = df_data_fl,
  reader_type = "fluorstar",
  layout_files = file
   )
R> head(df_data_layout)
```

```
A tibble: 6 x 9
  Wells  Sample  Time      Measurement  wavelength  basic    medium  concentration  layout_file
  <chr>  <chr>   <chr>     <dbl>        <chr>       <chr>    <chr>   <dbl>          <chr>
1 A01    " X1"   00:00:00  3.5          600         control  milk    100            test_fluorstar_layout
2 A01    " X1"   00:02:00  3.44         600         control  milk    100            test_fluorstar_layout
3 A01    " X1"   00:07:00  3.5          600         control  milk    100            test_fluorstar_layout
4 A01    " X1"   00:12:00  3.5          600         control  milk    100            test_fluorstar_layout
5 A01    " X1"   00:17:00  3.21         600         control  milk    100            test_fluorstar_layout
6 A01    " X1"   00:22:00  3.35         600         control  milk    100            test_fluorstar_layout
```

Figure 8: combine_data_layout(). In (A) is shown the code to run the function. In **(B)** the output data frame.

2.2 Plotting functions

As mentioned above mpxtractor possess two functions to visualize data. One of them is plot_layout_file(), to visualize experimental layout. The other is plot_gr_microplate() to visualize growth curves in the format of a 8x12 plate.

2.2.1 plot_layout_file()

This function can take several arguments to improve flexibility. The first argument is a .csv layout file with the proper format as is shown in **Figure 7.a.** The idea of the design of this function was taken from other implementation [11] .

The function plot_layout_file() possesses the following arguments that are required to identify the different conditions, like *var_shape* and *var_colour*. Here it is important to remark that to obtain a better visualization the argument *var_shape* should contain less than six different factors. This is related to different shapes available in R. While *var_color* is limitless.

The parameter *add_conc* takes a boolean value, and if is true the concentration values are added to the plot.

The file shown in **Figure 7.a.** is used to create the plot in **Figure 9**.

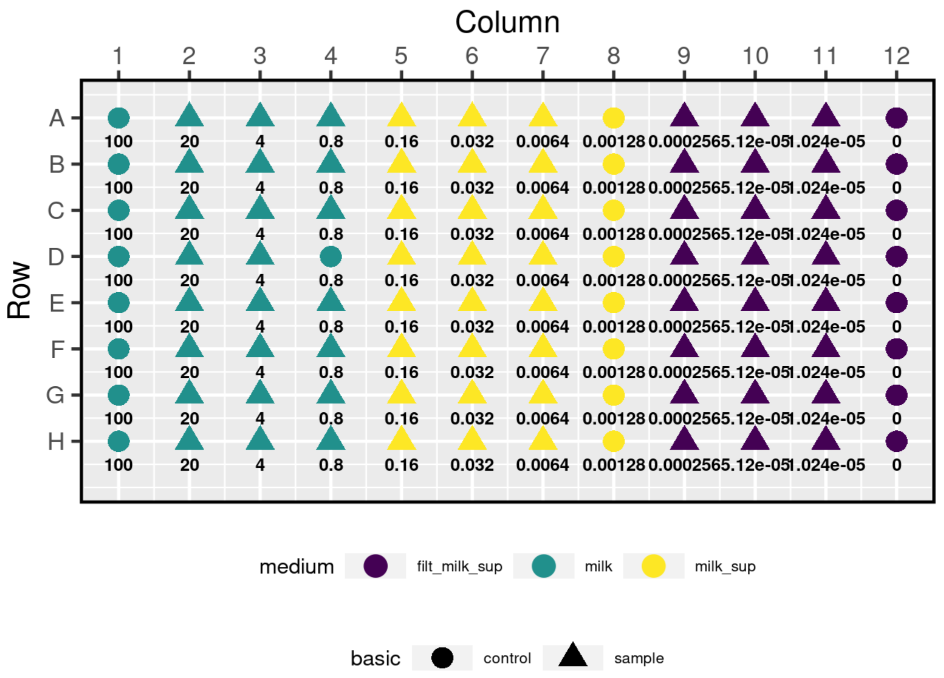
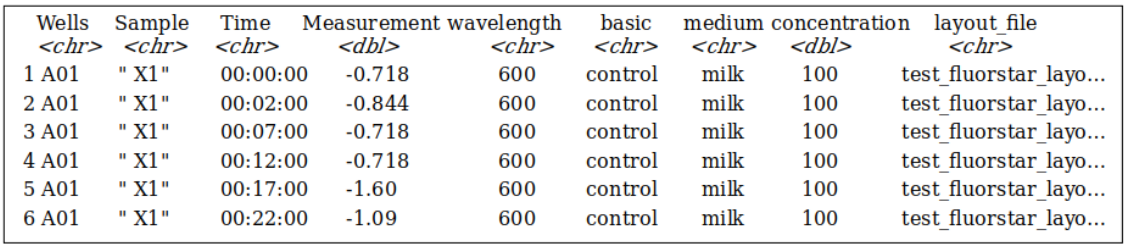


Figure 9: plot_layout_file function. The plot of the layout file corresponding to a 96 well plate with dimensions of 8x12. For the variable medium, there are three factors, milk, milk supernatant(milk_sup), and filtered milk supernatant (filt_milk_sup) and is used in the var_color parameter. The variable basic contains the control and samples and is represented by the parameter var_shape.

2.2.1 plot_gr_microplate()

The main objective of *plot_gr_microplate()* function is to obtain one quick overview of the performed experiment plotting growth rate over a microplate shaped plot. This function takes as input a tidy data frame which is generated by the wrangling function *combine_data_with_layout()* explained in section 2.1.6. In **Figure 10** the is shown the data frame after background correction and logarithmic transformation of the measurements.



```
   Wells   Sample    Time    Measurement wavelength   basic   medium concentration  layout_file
  <chr>   <chr>    <chr>        <dbl>        <chr>    <chr>    <chr>     <dbl>          <chr>
1 A01     " X1"    00:00:00     -0.718        600     control   milk      100     test_fluorstar_layo...
2 A01     " X1"    00:02:00     -0.844        600     control   milk      100     test_fluorstar_layo...
3 A01     " X1"    00:07:00     -0.718        600     control   milk      100     test_fluorstar_layo...
4 A01     " X1"    00:12:00     -0.718        600     control   milk      100     test_fluorstar_layo...
5 A01     " X1"    00:17:00     -1.60         600     control   milk      100     test_fluorstar_layo...
6 A01     " X1"    00:22:00     -1.09         600     control   milk      100     test_fluorstar_layo...
```

Figure 10: Example of input data frame of plot_gr_microplate(). In this case the column Measurement was preprocessed by background correction followed by log transform.

This function calculates growth curves through the internal function *compute_growth_rates()* that contains an implementation in R of Savitzky-Golay filter [8] . The filter is implemented in the function *sgolayfilt() f*rom the R package *signal* [12] . This filter is a particular type of low pass filter that is used to smooth the data and is needed here to improve the graphical output due to a large number of data points and random noise. It can be used to compute smoothed first order derivatives to obtain the growth rates.
The way that the Savitzky-Golay filter works is using a moving window average to fit a polynomial function through the data points. In the auxiliary function *compute_growth_rates()* the parameter window size(ws) is used to obtain the window length by dividing ws over the time interval. One import remark here is that the function *sgolayfilt()* in this case is used for time series which requires a constant time step with no missing values. In case of measurements with missing values these are imputed by taking the mean of the surrounding values, using the R package imputeTS [10] .
Once the growth curves were calculated one subplot for each well is generated and stored as a graphical object (grob) with the

coordinates of the background plot that represent the micro plate-shaped [13]. The code showing the arguments of this function is shown in **Figure 11.a.** The output is shown in **Figure 11.b.**

(A)

```
R> plot_gr <- mpxtractor::plot_gr_microplate(
     df_data = df_corrected,
     var_gr = "Measurement",
     ws = "2hs", cond_to_col = "medium"
     )

R> ggplot2::ggsave(
      filename = "fluorstar_report.png",
      plot = plot_gr,
      width = 25,
      height = 20,
      units = "cm"
      )
```

(B)

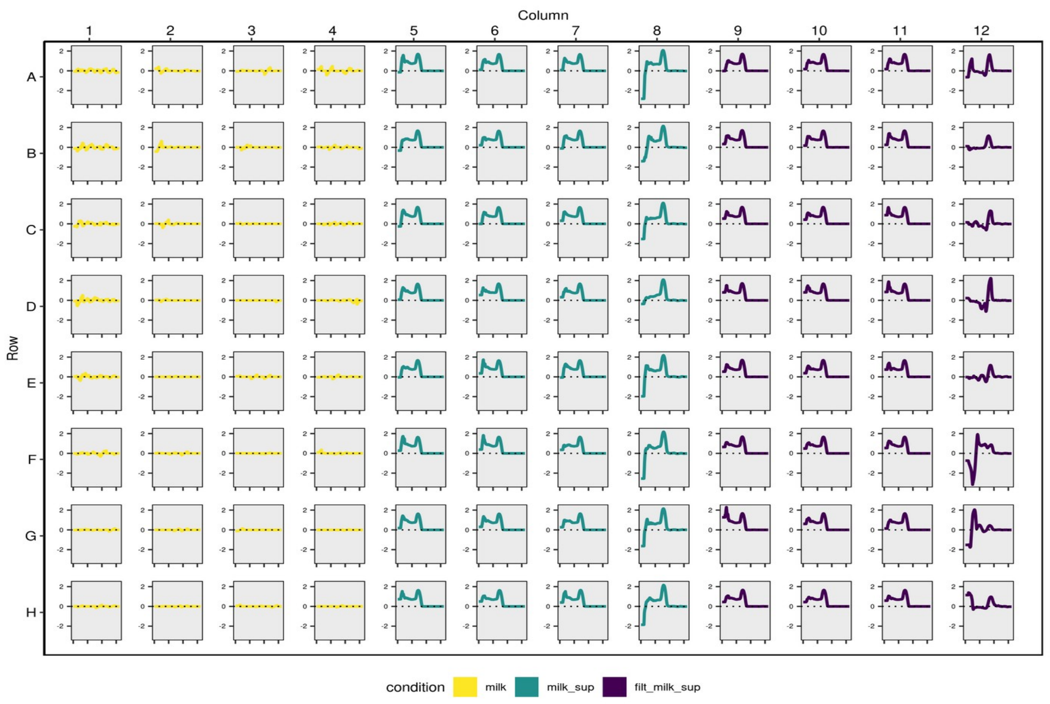


Figure 11: plot_gr_microplate(). In (A) the code to generate the plot. Note the ws is 2 hs, the condition to colour the subplots is Medium. In (B) the growth curves plot over the micro plate frame.

3 Results

3.1 Parameter settings to validate results

To validate the growth rates calculated by mpxctractor one validation was performed for each of the different plate reader machines. The output of mpxtractor for each machine was compared against the raw and processed data obtained by Sieze Douwenga, one of the PhD students in the laboratory (data not shown). In one case the growth rates were calculated using the implementation of the Savitzky-Golay filter in R and the other using Python. Also, in both implementations the same arguments were used, see **Table1.**
It is important to remark that the raw files of the validation data were preprocessed performing background correction followed by log transformation for each well, and that the same preprocess was applied when using mpxtractor to obtain comparable results.

Table 1: Arguments Savitzky-Golay filter. Here are shown the values and symbols for each argument in R and Python used to performed the validation of growth rates.

Type of argument	Value	R symbol	Python symbol
Window length = window size / diff time	wl= (2/ 0.0333) hs	n=60	window_length=60
Filter order	1	p=1	polyorder=1
Derivative order	1	m=1	derivative=1

3.2 Validation of results

The results shown in this section correspond to those obtained with mpxtractor function *compute_growth_rates()* which also can be used as an auxiliary function. This function uses the *sgolayfilt()* function from the signal package [12] in R to compute growth rates. The values in column *log(Abs)* are log values of the measurements after background correction.

3.2.1 Validation of data generated with the SpectraMax reader

The comparison between mpxtractor and the validation data for spectramax readers can be observed in **Table 2**. This table shows the log transform of the absorbance and the corresponding growth rates. From time point 0.0 to 1.10 the growth rates of mpxtractor are the same as the reference data, but from time point 1.13 there are -inf in the validation data until the time point 2.13, from this time point until time point 3.133 there are inf. Also, looking at the time point 2.133 and the value in log(Abs) there is an -inf. Here, the measured absorbance was below the value used for background correction. The python implementation of the Savitzky-Golay filter in python takes that -inf and generates growth rates with +/- inf. Since the window size was two hours, the growth rates in a one-hour interval before and after the time point 2.133 can not be calculated. On the other hand, mpxtractor, when it finds -/+ inf values in a measurement column, in this case *log(Abs)*, will impute the -Inf by taking the mean between the two values surrounding the this time point.

Table 2: spectramax validation results. Highlighted in orange are the growth rate values generated by the python implementation in the validation set. In green the time points from which the +/- inf are generated in the validation data. Note that from 1.133 to 2.133 the value highlighted in gray is an imputed value by the mean of the values one step ahead and one step behind.

		mpxtractor data		**Validation data**	
Wells	**Time(hs)**	**log(Abs)**	**Growth rate**	**log(Abs)**	**Growth rate**
A01	0.0000	-7.264430	-0.238977	-7.264430	-0.238978
:	:	:	:	:	:
A01	1.1000	-6.907755	-0.384584	-6.907755	-0.384584
A01	**1.1333**	-6.907755	-0.412485	-6.907755	**-Inf**
:	**:**	**:**	**:**	**:**	**:**
A01	2.1000	-7.600902	0.033795	-7.600902	**-Inf**
A01	**2.1333**	**-7.600902**	0.031808	**-Inf**	**-Inf**
A01	2.1666	-7.600902	0.007807	-7.600902	**Inf**
:	**:**	**:**	**:**	**:**	**:**
A01	**3.1333**	-7.130899	0.357646	-7.130899	**Inf**
A01	3.1666	-7.600902	0.339011	-7.600902	0.339011
A01	26.1333	-6.645391	0.032598	-6.645391	0.032599

3.2.2 Validation of data generated with the MultiscanGO reader

The next validation results correspond to multiscango output data. In **Table 3** the results are shown. In this case, the value of absorbance remains constant until the time point 0.6666, this means that from time point 0.00 until 0.666 there are -Inf values, and all the values for these time points are imputed. The implementation of Savitzky-Golay in R in case of finding all missing values within the window size automatically increases the window until finding two non-missing values. On the other hand, the implementation in python assigns a NaN.

Table 3: multiscango validation results. Highlighted in orange the growth rate values generated by the python implementation in the validation set. In green the time points from which the NaN and Inf are generated in the validation data. In blue the cells containing the background value. The value highlighted in gray are the imputed values by the mean of the first two non-missing values .

		mpxtractor data		Validation data	
Wells	**Time(hs)**	**log(Abs)**	**Growth rate**	**log(Abs)**	**Growth rate**
A01	**0.000000**	**-6.907755**	0.6355529	-Inf	NaN
:	:	:	:	:	:
A01	**0.666667**	**-6.907755**	0.6355529	**-Inf**	NaN
A01	0.708333	-6.907755	0.6355529	-6.907755	NaN
:	:	:	:	:	:
A01	0.958333	-6.907755	0.6355529	-6.907755	**NaN**
A01	1.000000	-6.907755	0.6355529	-6.907755	**Inf**
:	:	:	:	:	:
A01	**1.666667**	-6.214608	0.8386713	-6.214608	**Inf**
A01	1.708333	-5.809143	0.8284531	-5.809143	0.828453
:	:	:	:	:	:
A01	41.625000	-4.710531	-0.0173069	-4.710531	-0.0173069

3.2.3 Validation of data generated with the Fluorstar reader

The comparison between mpxtractor and the validation data for fluorstar readers can be observed in **Table 4**. In this case, due to the correction of the time series, the starting time point for **mpxtractor** data is 0.0333. This is because between the starting time point 0.00 and the second time point there is a 2 minutes interval. To correct this and keep the global time interval of 5 minutes, the starting time point 0.00 was removed, see **Figure 10.** In the validation data, the *t*ime series is generated by multiplying the time interval by the maximum number of readings minus one.

From time point 0.0 to 1.25 of the validation data (in green) the growth rates are the same, but from time point 1.333 there are -inf in the validation data until the time point 2.333, from this time point until time point 3.333 there are inf. Also looking at the time point 2.333 and the value in log(Abs) there is an -inf, corresponding to absorbances below the minimum value that is used for background correction.

As mentioned in **section 3.2.1** the implementation of savgol filter in python takes that -inf to generate growth rates with +/- inf, this is because there are no values to generate the growth rates. On the other hand, mpxtractor imputes the data by computing the mean between the two values surrounding the missing value.

Table 4: Fluorstar validation results. Highlighted in orange the growth rate values generated by the python implementation in the validation set. In green the time points from which the +/- inf are generated in the validation data. The value highlighted in gray is an imputed value by the mean of the values one step ahead and one step behind.

	mpxtractor data			Validation data		
Wells	**Time(hs)**	**log(Abs)**	**Growth rate**	**Time(hs)**	**log(Abs)**	**Growth rate**
A01	0.03333	-0.84373	-0.07818094	0.00000	-0.843738	-0.078181
:	:	:	:	:	:	:
A01	1.28333	-1.175061	0.1089509	1.25000	-1.175061	0.108951
A01	1.36666	-0.717644	0.06203398	**1.3333**	-0.717645	**-inf**
:	:	:	:	:	:	:
A01	2.28333	-1.161871	-0.05384393	2.25000	-1.16187	**-inf**
A01	2.36666	**-1.010418**	0.057739	**2.3333**	**-inf**	**-inf**
A01	2.45000	-0.858965	0.00011882	2.41666	-0.858966	**inf**
:	**:**	**:**	**:**	**:**	**:**	**:**
A01	3.36666	-0.999400	0.06919943	**3.3333**	-0.999401	**inf**
A01	3.45000	-0.857078	0.08525832	3.41666	-0.857079	0.085258
:	:	:	:	:	:	:
A01	16.1166	-0.797175	-0.1556584	16.0833	-0.797175	0.073194
A01	--	--	--	20.4166	-0.920299	0.057739

4 Conclusion & Discussion

During the development of mpxtractor the lack of standards of output files produced by plate reader machines was evident. For the three plate readers used to build mpxtractor many different format files can be exported depending on the user's choice. To improve standardization mpxtractor receives only two types of files, .txt for data files and .csv for layout files. For example, if the input files are not in a correct format the package reports an error. Wrong input files can be generated by the users if they do not export from the plate readers the output file as .txt format. This can happen for example if the user copy-pastes the data from the reader software interface. Another example is referred to multiscanGO output files with table format (see **Figure 3).**, These files contain as metadata only the number of readings, wavelength, and plate number. When using these types of files the user has to add the time interval between readings as an argument to the import function. The other option to export output files from the multiscango reader is as *list*. This is a file with more complete information, containing the time points, wavelength, sample number, groups and number of readings .
In the case of the FluorStar machine mpxtractor processes the output files containing as metadata wavelengths, time, and the measurements which can either be absorbance or fluorescence. SpectraMax output files contain time, temperature and the measurements.

The flexibility of mpxctractor lies in the different input raw data files that can process from three different plate readers, making it possible to use the package for an acceptable number of different plate machines. The results showed that the growth rates obtained by **mpxtractor** were similar to those obtained by other researchers (see **table 2,   3, and 4**).
Using the data available it was possible to discover several technical errors that are important to be aware of, like missing values in multiscango files. To deal with this, **mpxtractor** imputes the missing value by taking the mean of the surrounding data points and returning a warning.
Other technical errors can be found in Fluorstar machines associated with time points, the time interval is not uniform, this means that between some time points the time interval is different than in others.

This issue has to be addressed by the user to plot the growth curves otherwise mpxtractor returns an error and the process stops.

Although mpxtractor deals with standard raw data files contributing to the standardization of high throughput experiments, there are a few considerations to take into account. For example, the lack of broad testing. The robustness of mpxtractor has not been rigorously tested because many situations were not considered during the development of the package. One approach to improve this is through users' reports of bugs in the package repository.

5 Acknowledgments

I would like to thank Douwe Molenaar for giving me the chance to carry out this project under his supervision and for giving me space to develop this project more independently. To all the Ph.D. students from the Systems Biology laboratory at the VU who share their data used to build mpxtractor, Laura Guilherme, Dennis Postma, and specially Sieze Douwenga for his willingness to help me through insightful discussions.
Also, I would like to thank my family and my friends for the continued support from both sides of the world.

References

[1] H. Wickham, “Tidy data,” *J. Stat. Softw.*, vol. 59, no. 10, pp. 1–23, Sep. 2014.

[2] M. Devices, “SpectraMax Plus 384 Microplate Reader | Molecular Devices.” [Online]. Available: https://www.moleculardevices.com/sites/default/files/en/assets/data-sheets/br/spectramax-plus-384-microplate-reader.pdf. [Accessed: 23-Jul-2020].

[3] U. Manual, “Thermo Scientific Multiskan GO.” [Online]. Available: https://assets.thermofisher.com/TFS-Assets/LCD/manuals/D01521~.pdf. [Accessed: 23-Jul-2020].

[4] “FLUOstar Omega Microplate Reader - BMG LABTECH.” [Online]. Available: https://www.bmglabtech.com/fluostar-omega/?gclid=CjwKCAjw5vz2BRAtEiwAbcVIL1gsU23RHnNXkk9MJVHjLjORtVjQe3sIJ1l7cp5YA38qmxnD18LnLBoCrC8QAvD_BwE. [Accessed: 09-Jun-2020].

[5] Czarnik, A W; Mei and H-Y, *Comprehensive Medicinal Chemistry II*. 2007.

[6] S. M. Hughes, “plater: Read, Tidy, and Display Data from Microtiter Plates.”

[7] M. Boutros, L. P. Brás, and W. Huber, “Analysis of cell-based RNAi screens,” *Genome Biol.*, vol. 7, no. 7, p. R66, Jul. 2006.

[8] A. Savitzky and M. J. E. Golay, “Smoothing and Differentiation of Data by Simplified Least Squares Procedures.,” *Anal. Chem.*, vol. 36, no. 8, pp. 1627–1639, Jul. 1964.

[9] “MartinBanchero/mpxtractor: ##!Beta Version of mpxtractor## Extract data from microplate reader output files into a tidy dataframe. Plot layout files and plot growth rates over microplate frame.” [Online]. Available: https://github.com/MartinBanchero/mpxtractor. [Accessed: 09-Jun-2020].

[10] “Package ‘imputeTS’ Title Time Series Missing Value Imputation,” 2019.

[11] “Brian Connelly | Plotting Microtiter Plate Maps.” [Online]. Available: https://bconnelly.net/posts/plotting-microtiter-plate-maps/. [Accessed: 10-Jun-2020].

[12] “signal package | R Documentation.” [Online]. Available: https://www.rdocumentation.org/packages/signal/versions/0.7-6. [Accessed: 11-Jun-2020].

[13] “Embedding subplots in ggplot2 graphics.” [Online]. Available: https://aosmith.rbind.io/2019/04/22/embedding-subplots/. [Accessed: 11-Jun-2020].