

---

# **platelib Documentation**

***Release 0.1.5-alpha***

**Emil Dandanell Agerschou**

**Jun 06, 2018**



# CONTENTS

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Summary . . . . .	1
1.2	Disclaimer . . . . .	1
1.3	Read the docs . . . . .	1
<b>2</b>	<b>Installation</b>	<b>3</b>
2.1	Prerequisites . . . . .	3
2.2	Download . . . . .	3
2.3	Install . . . . .	3
2.4	Upgrade . . . . .	3
2.5	Uninstall . . . . .	3
<b>3</b>	<b>Support</b>	<b>5</b>
<b>4</b>	<b>Cookbook</b>	<b>7</b>
4.1	Reading in data . . . . .	7
4.2	Accessing data . . . . .	7
4.3	Plotting data . . . . .	8
<b>5</b>	<b>platelib</b>	<b>9</b>
5.1	platelib package . . . . .	9
<b>6</b>	<b>Contribute</b>	<b>13</b>
6.1	More tools . . . . .	13
6.2	Building . . . . .	13
6.3	Planned improvements . . . . .	14
	<b>Python Module Index</b>	<b>15</b>
	<b>Index</b>	<b>17</b>



## INTRODUCTION

### 1.1 Summary

platelib is an attempt to make common tasks when working with kinetic platereader, especially amyloid aggregation, data easy and compatible with Python.

### 1.2 Disclaimer

The state of this repository is one of very early development, the code is not elegant, there most likely are bugs so **Use with caution!**

### 1.3 Read the docs

A brief overview is given below, for more detailed information see the docs directory and subfolders, or even better have a look at the source :)



## INSTALLATION

### 2.1 Prerequisites

- `python`
- `pip`
- `git` (optional)

### 2.2 Download

Either download the [source distribution](#) directly or use:

```
git https://github.com/edager/platelib
```

### 2.3 Install

Go to directory where the platelib source file is (`platelib/dist/` if `git` was used) and run the following in the terminal:

```
pip install platelib-X.X.tar.gz
```

Where `X.X` should be replaced by the version that was downloaded.

### 2.4 Upgrade

Go to directory where the platelib source file is (`platelib/dist/` if `git` was used) and run the following in the terminal:

```
pip install --upgrade platelib-X.X.tar.gz
```

Where `X.X` should be replaced by the version that was downloaded.

### 2.5 Uninstall

Simply go to the terminal and run:

```
pip uninstall platelib
```



## SUPPORT

- **For questions related to usage:**
  - [Stack Overflow](#) using the tags `python` and `matplotlib`
- **For questions related to bugs or enhancements:**
  - [GitHub](#) using the issue system



## 4.1 Reading in data

The main functionality of `platelib` is the `read_plate` function that allows for reading in `platerreader` data from kinetic experiments into a common framework namely into the `Plate_data` class.

If an equal number of replicates per sample were prepared this can be specified (default is 3):

```
p = read_plate('path/to/file', replicates=5)
```

It can be specified which direction the replicates were loaded onto the plate where `'hori'` (horizontal) means towards increasing numbers and `'vert'` is towards increasing letters (default is `'hori'`):

```
p = read_plate('path/to/file', rep_direction='vert')
```

**NOTE that the replicates have to be next to each other!**

Alternatively it can be specified which wells contains replicates:

```
p = read_plate('path/to/file', named_samples=[['B03', 'D07'], ['B02', 'E06', 'G12']])
```

Data from Tecan platereaders can be read in as (default is `'bmg'`):

```
p = read_plate('path/to/file', platerreader='tecan')
```

**NOTE that this functionality has not been fully tested yet!**

As well as from BMG platereaders either where the data has prior been transposed `True` such that well data are in column format or in row format `False` (default is `True`):

```
p = read_plate('path/to/file', transposed=False)
```

Note that it's automatically detected if several measurements (*e.g.*) were made per time-point (see [Accessing data](#))

The time unit can also be specified which as either `'seconds'`, `'minutes'`, `'hours'`, or `'days'` will carry along into indexes if exported and to unit of x-axis if plotted (default is `'hours'`):

```
p = read_plate('path/to/file', time_unit='days')
```

## 4.2 Accessing data

The `Plate_data` class allows for different ways of accessing the data

Through index:

```
p[1]
```

Through index slice:

```
p[:3]
```

Through well name:

```
p['B02']
```

Through list of well names:

```
p[['B02', 'C03', 'D04']]
```

Retrieved as a pandas.DataFrame with wellnames as column names and time points as index:

```
df = Plate_data.to_a_dataframe()
```

Or as a (C)omma (S)eperated (V)aribles file with the first line being (time unit + ) well names and the first column are the time points:

```
Plate_data.to_a_csv('path/to/file.csv')
```

## 4.3 Plotting data

The data is plotted according to replicates, and subtitles can be added (default is *None*):

```
p.plot(titles=['condition 1', 'conditions 2'])
```

It can be specified whether all plots should have its own y-axis, whether all plots should have the same (default is *True*):

```
p.plot(sharey=False)
```

If several measurements were made per time-point it can be specified whether all measurements should be plotted or not (default is *True*):

```
p.plot(plot_multi=False)
```

## 5.1 platelib package

### 5.1.1 Submodules

#### 5.1.2 platelib.fitfun module

```

platelib.fitfun.exp_rise(t, a, b, k)
platelib.fitfun.fit_fun(func, df, bounds=([0, 0, 0, 0], [1, 50, 10, 65]))
class platelib.fitfun.fit_plate(data, replicates, multi_chrom)
    Bases: platelib.plateread.Plate_data
platelib.fitfun.linear(t, a, b)
platelib.fitfun.quadratic(t, a, b, c)
platelib.fitfun.sigmoid(x, y0, L, k, x_half)
platelib.fitfun.sigmoidal_auto(t, a, b, k)

```

#### 5.1.3 platelib.plateread module

```

class platelib.plateread.Plate_data(data, replicates, multi_chrom)
    Class for containing data from a platereader assay.

```

##### Parameters

- **data** – Pandas DataFrame with time points as index and wells as columns
- **replicates** – Positive integer of replicates, assuming equal number of replicates of all samples

```
plot (titles=None, sharey=True, plot_multi=True, return_fig=False)
```

Plots the number of samples i.e. replicates/wells in the data set.

##### Parameters

- **titles** – List-like object of subtitles
- **sharey** – Bool, if True all y-axis limits will be identical, if False y-axis limits are given by matplotlib defaults.
- **plot\_multi** – Bool, if several different measurements are present per time point, only plot the first one, if False plots all of the values.

- **return\_fig** – Bool, if True returns a figure object, if False only plot the data

**to\_a\_csv** (*path, one\_per\_multi\_c=False*)

Returns the data as a pandas dataframe with times as indexes

**Parameters**

- **path** – String of path to store output
- **one\_per\_multi\_c** – Bool, if ‘True’ one measurement per dataframe will be exported otherwise all will be exported in one file.

**to\_a\_dataframe** (*one\_per\_multi\_c=False*)

Returns the data as a list of pandas dataframe(s) with times as indexes

**Parameters** **one\_per\_multi\_c** – Bool, if ‘True’ one measurement per dataframe will be exported, if False all will be exported in one dataframe.

**platelib.plateread.add\_lines** (*fig, hori=False, vert=False*)

Adds horizontal and figure lines to each plot in a subplot

**Parameters**

- **fig** – matplotlib figure object
- **hori** – Float, y position where the horizontal will be at
- **vert** – Float, x position where the horizontal will be at

**platelib.plateread.named\_order** (*named\_samples, df*)

Reorder dataframe columns according to the specified order, returns dataframe and list of replicates.

**Parameters**

- **named\_samples** – List of lists, one list per samples containing all replicates
- **df** – Pandas DataFrame

**platelib.plateread.read\_plate** (*filename, replicates=3, rep\_direction='hori', time\_unit='hours', named\_samples=[], platereader='bmg', transposed=True*)

Reads in data from a CSV file from a BMG or Tecan platereader and returns a platedata object.

:param filename:String, path to filename :param replicates: Positive integer, The number of replicates per sample. :param rep\_direction: String, directions replicates is in. Only ‘hori’ and ‘vert’ are accepted directions where ‘hori’ if replicates are going from left to righth ‘vert’ from replicates going from top to bottom. :param time\_unit: String, time unit one would like to have, accepted values are: ‘seconds’, ‘minutes’, ‘hours’, ‘days’ :param named\_samples: List of lists, where each list should correpond to a sample and contain all replicates of it :param platereader: String, The plate reader used to collect the data. Only ‘bmg’ and ‘tecan’ are accepted platereaders :param transposed: Bool, specifies wether the wells are in column (True) or row format (False).

**platelib.plateread.read\_tecan** (*filename*)

Reads in untransposed data from a tecan platereader and returns a pandas DataFrame object.

**Parameters** **filename** – String, path to filename

**platelib.plateread.read\_transposed\_bmg** (*filename*)

Reads in transposed data from a BMG platereader and returns a pandas DataFrame object.

**Parameters** **filename** – String, path to filename.

**platelib.plateread.read\_untransposed\_bmg** (*filename*)

Reads in untransposed data from a BMG platereader and returns a pandas DataFrame object.

**Parameters** **filename** – String, path to filename.

`platelib.plateread.search_start(filename)`

Find start of data region and returns the line number by finding the line that starts with “Well”.

**Parameters** `filename` – String, path to filename.

`platelib.plateread.to_time_units(df, time_unit)`

Convert the index a dataframe into the time unit specified, returns dataframe.

**Parameters**

- **df** – Pandas DataFrame
- **time\_unit** – String, allowed values are ‘seconds’, ‘minutes’, ‘hours’, and ‘days’.

`platelib.plateread.vert_order(replicates, df)`

Reorder dataframe column to vertical order, returns dataframe.

**Parameters**

- **replicates** – List of positive integer number of replicates.
- **df** – Pandas DataFrame

## 5.1.4 Module contents





## CONTRIBUTE

Contributions are more than welcome, please raise an issue on the [github](#) page highlighting the bug/extension/compatibilities before doing a pull request.

### 6.1 More tools

Apart from the tools listed in [Installation](#) the following is needed:

- Unix-like system
- [git](#)
- [pandoc](#)

### 6.2 Building

You have made some wicked cool changes to the source code or the documentation that you want to share with the world, awesome!

Now there's just a few steps before they can be incorporated into the platelib master branch

#### 6.2.1 Changing the version number

The versioning scheme of platelib should be done in reasonable accordance with the so called [Semantic versioning](#) where X.Y.Z should be read as MAJOR.MINOR.PATCH.

The version number has to be changed in the two files `setup.py` and `docs/source/conf.py`

#### 6.2.2 Create new source distribution

Go to the docs folder and run:

```
./full_make.sh
```

If no errors occurred it can be uploaded to your local branch and a pull request can be made.

## 6.3 Planned improvements

This is as much a wish-list as literally planned improvements:

- Plotting
  - Plotting of data from several plates in some sensible way.
- Fitting
  - Local fitting of traces in plate
  - Global fitting of traces in plate
- Statistical analysis
  - Goodness-of-fit
  - Variance along traces, among replicates, and between conditions
- Python 3.X compatibility
- PyPI availability

## PYTHON MODULE INDEX

### p

platelib, [11](#)

platelib.fitfun, [9](#)

platelib.plateread, [9](#)



## INDEX

### A

`add_lines()` (in module `platelib.plateread`), [10](#)

### E

`exp_rise()` (in module `platelib.fitfun`), [9](#)

### F

`fit_fun()` (in module `platelib.fitfun`), [9](#)

`fit_plate` (class in `platelib.fitfun`), [9](#)

### L

`linear()` (in module `platelib.fitfun`), [9](#)

### N

`named_order()` (in module `platelib.plateread`), [10](#)

### P

`Plate_data` (class in `platelib.plateread`), [9](#)

`platelib` (module), [11](#)

`platelib.fitfun` (module), [9](#)

`platelib.plateread` (module), [9](#)

`plot()` (`platelib.plateread.Plate_data` method), [9](#)

### Q

`quadratic()` (in module `platelib.fitfun`), [9](#)

### R

`read_plate()` (in module `platelib.plateread`), [10](#)

`read_tecan()` (in module `platelib.plateread`), [10](#)

`read_transposed_bmg()` (in module `platelib.plateread`), [10](#)

`read_untransposed_bmg()` (in module `platelib.plateread`),  
[10](#)

### S

`search_start()` (in module `platelib.plateread`), [10](#)

`sigmoid()` (in module `platelib.fitfun`), [9](#)

`sigmoidal_auto()` (in module `platelib.fitfun`), [9](#)

### T

`to_a_csv()` (`platelib.plateread.Plate_data` method), [10](#)

`to_a_dataframe()` (`platelib.plateread.Plate_data` method),  
[10](#)

`to_time_units()` (in module `platelib.plateread`), [11](#)

### V

`vert_order()` (in module `platelib.plateread`), [11](#)