Introduction to using gcplyr

Mike Blazanin

Contents

Getting started

`gcplyr` is a package that implements a number of functions to make it easier to import, manipulate, and analyze microbial growth from data collected in multiwell plate readers ("growth curves"). Without `gcplyr`, importing and analyzing plate reader data can be a complicated process that has to be tailored for each experiment, requiring many lines of code. With `gcplyr` many of those steps are now just a single line of code.

This document gives an introduction of how to use `gcplyr` for each step of a growth curve analysis.

To get started, you need your growth curve data file saved to your computer (.csv, .xls, .xlsx, or any other format that can be read by `read.table`).

Users often want to combine their data with some information on the experimental design of their plate(s). You can save this information into a tabular file as well, or you can just keep it handy to enter directly in `R` (see `vignette("gc03_incorporate_designs")`).

Let's get started by loading `gcplyr`. We're also going to load a couple other packages we'll need.

```
library(gcplyr)
#> ##
#> ## gcplyr (Version 1.12.0, Build Date: 2025-07-28)
#> ## See http://github.com/mikeblazanin/gcplyr for additional documentation
#> ## Please cite software as:
#> ##   Blazanin, Michael. gcplyr: an R package for microbial growth
#> ##   curve data analysis. BMC Bioinformatics 25, 232 (2024).
#> ##   https://doi.org/10.1186/s12859-024-05817-3
#> ##

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
library(ggplot2)
```

A quick demo of gcplyr

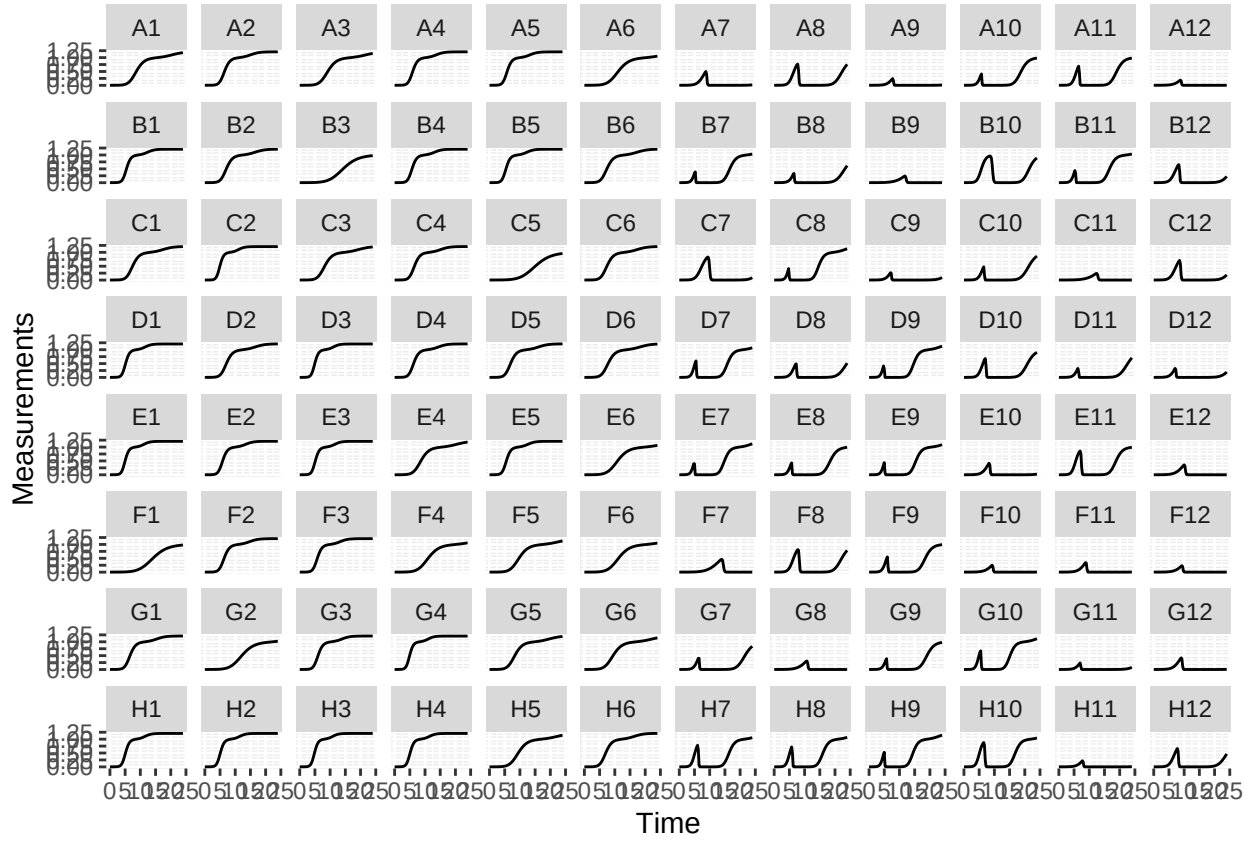
Before digging into the details, here's a simple demonstration of what a final `gcplyr` script can look like. This script:

1. imports data from files created by a plate reader
2. combines it with design files created by the user
3. calculates the lag time, maximum growth rate, maximum density, and area-under-the-curve

Don't worry about understanding all the details of how the code works right now. Each of these steps is explained in depth in later articles.

```
#For the purposes of this demo, we have to create our example data and
# design files. Normally, the data file would be created by a plate reader, and
# the design file would be created by you, the user

#Generate our example data file, widedata.csv
make_example(vignette = 1, example = 1)
#> Files have been written
#> [1] "./widedata.csv"

#Generate our example design files, Bacteria_strain.csv and Phage.csv
make_example(vignette = 1, example = 2)
#> Files have been written
#> [1] "./Bacteria_strain.csv" "./Phage.csv"

# Read in our data
data_wide <- read_wides(files = "widedata.csv")

# Transform our data to be tidy-shaped
data_tidy <-
  trans_wide_to_tidy(wides = data_wide, id_cols = c("file", "Time"))

# Convert our time into hours
data_tidy$Time <- as.numeric(data_tidy$Time)/3600

# Import our designs
designs <- import_blockdesigns(files = c("Bacteria_strain.csv", "Phage.csv"))
#> Inferred 'into' column names as: Bacteria_strain, Phage

# Merge our designs and data
data_merged <- merge_dfs(data_tidy, designs)
#> Joining with `by = join_by(Well)`

#Set up the Well column so they plot in the correct order
data_merged$Well <-
```

```
  factor(data_merged$Well,
         levels = paste0(rep(LETTERS[1:8], each = 12), 1:12))

#Plot the data
ggplot(data = data_merged, aes(x = Time, y = Measurements)) +
  geom_line() +
  facet_wrap(~Well, nrow = 8, ncol = 12)
```




```
# Voila! 8 lines of code and all your data is imported & plotted!

# Calculate the per-capita growth rate over time in each well
data_merged <- mutate(
  group_by(data_merged, Well),
  percap_deriv = calc_deriv(y = Measurements, x = Time, percapita = TRUE,
                            blank = 0, window_width_n = 5))

# Calculate four common metrics of bacterial growth:
#  the lag time, saving it to a column named lag_time
#  the maximum growth rate, saving it to a column named max_percap
#  the maximum density, saving it to a column named max_dens
#  the area-under-the-curve, saving it to a column named 'auc'
data_sum <- summarize(
  group_by(data_merged, Well, Bacteria_strain, Phage),
  lag_time = lag_time(x = Time, y = Measurements, deriv = percap_deriv,
                      blank = 0),
```

```
  max_percap = max(percap_deriv, na.rm = TRUE),
  max_dens = max(Measurements),
  auc = auc(y = Measurements, x = as.numeric(Time)))
#> `summarise()` has grouped output by 'Well', 'Bacteria_strain'. You can override using
#> the `.groups` argument.

# Print some of the values
head(data_sum)
#> # A tibble: 6 x 7
#> # Groups:   Well, Bacteria_strain [6]
#>   Well  Bacteria_strain Phage    lag_time max_percap max_dens   auc
#>   <fct> <chr>           <chr>       <dbl>      <dbl>    <dbl> <dbl>
#> 1 A1    Strain 1        No Phage     2.11      1.00      1.18  15.9
#> 2 A2    Strain 2        No Phage     1.74      1.31      1.21  19.3
#> 3 A3    Strain 3        No Phage     2.14      0.915     1.15  15.1
#> 4 A4    Strain 4        No Phage     1.68      1.43      1.21  20.1
#> 5 A5    Strain 5        No Phage     1.67      1.47      1.21  20.3
#> 6 A6    Strain 6        No Phage     2.41      0.789     1.05  12.8

#Set up the Well column so they plot in the correct order
data_sum$Well <- factor(data_sum$Well,
                        levels = paste0(rep(LETTERS[1:8], each = 12), 1:12))

#Plot lag time
ggplot(data = data_sum) +
  geom_text(aes(label = round(lag_time, 2), x = 1, y = 1)) +
  facet_wrap(~ Well, ncol = 12) +
  labs(title = "Lag time by well") +
  theme(axis.title = element_blank(),
        axis.text = element_blank(),
        axis.ticks = element_blank())
```

Lag time by well

A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12
2.11	1.74	2.14	1.68	1.67	2.41	1.42	1.21	1.38	1.19	1.2	1.54
B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12
1.58	1.9	2.76	1.66	1.56	1.97	1.13	1.33	1.64	1.2	1.11	1.38
C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12
1.91	1.47	1.98	1.79	2.56	1.97	1.31	1.06	1.31	1.26	1.34	1.38
D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12
1.51	1.86	1.46	1.75	1.84	1.99	1.09	1.28	1.05	1.24	1.28	1.36
E1	E2	E3	E4	E5	E6	E7	E8	E9	E10	E11	E12
1.48	1.69	1.54	2.11	1.61	2.35	1.06	1.22	1.12	1.42	1.16	1.47
F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12
2.51	1.79	1.63	2.41	2.25	2.41	1.38	1.28	1.15	1.54	1.48	1.52
G1	G2	G3	G4	G5	G6	G7	G8	G9	G10	G11	G12
1.78	2.47	1.74	1.45	2.06	2.11	1.24	1.48	1.26	1.04	1.4	1.33
H1	H2	H3	H4	H5	H6	H7	H8	H9	H10	H11	H12
1.52	1.52	1.43	1.5	2.26	1.87	1.1	1.1	1.03	1.08	1.52	1.3

```
#Plot growth rate
ggplot(data = data_sum) +
  geom_text(aes(label = round(max_percap, 2), x = 1, y = 1)) +
  facet_wrap(~ Well, ncol = 12) +
  labs(title = "Maximum growth rate by well") +
  theme(axis.title = element_blank(),
        axis.text = element_blank(),
        axis.ticks = element_blank())
```

Maximum growth rate by well

A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12
1	1.31	0.92	1.43	1.47	0.79	1	1.31	0.92	1.43	1.47	0.79
B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12
1.56	1.22	0.62	1.49	1.54	1.17	1.56	1.22	0.62	1.49	1.54	1.17
C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12
1.14	1.7	1.03	1.31	0.57	1.17	1.14	1.7	1.03	1.31	0.57	1.17
D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12
1.63	1.2	1.69	1.37	1.22	1.1	1.63	1.2	1.69	1.37	1.22	1.1
E1	E2	E3	E4	E5	E6	E7	E8	E9	E10	E11	E12
1.68	1.46	1.65	1	1.53	0.79	1.68	1.46	1.65	1	1.53	0.79
F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12
0.62	1.36	1.46	0.79	0.9	0.78	0.62	1.36	1.46	0.79	0.9	0.78
G1	G2	G3	G4	G5	G6	G7	G8	G9	G10	G11	G12
1.28	0.7	1.43	1.69	1.04	0.89	1.28	0.7	1.43	1.69	1.04	0.89
H1	H2	H3	H4	H5	H6	H7	H8	H9	H10	H11	H12
1.66	1.65	1.71	1.64	0.94	1.22	1.66	1.65	1.71	1.64	0.94	1.22

```
#Plot Maximum density
ggplot(data = data_sum) +
  geom_text(aes(label = round(max_dens, 2), x = 1, y = 1)) +
  facet_wrap(~ Well, ncol = 12) +
  labs(title = "Maximum density by well") +
  theme(axis.title = element_blank(),
        axis.text = element_blank(),
        axis.ticks = element_blank())
```

Maximum density by well

A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12
1.18	1.21	1.15	1.21	1.21	1.05	0.5	0.76	0.24	0.98	0.98	0.19
B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12
1.21	1.21	0.98	1.21	1.21	1.21	1.03	0.62	0.24	0.96	1.03	0.65
C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12
1.21	1.21	1.2	1.21	0.96	1.21	0.83	1.13	0.26	0.86	0.24	0.71
D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12
1.21	1.21	1.21	1.21	1.21	1.2	1.07	0.52	1.13	0.9	0.71	0.33
E1	E2	E3	E4	E5	E6	E7	E8	E9	E10	E11	E12
1.21	1.21	1.21	1.18	1.21	1.06	1.12	0.99	1.09	0.42	0.99	0.36
F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12
0.98	1.21	1.21	1.06	1.13	1.05	0.47	0.81	0.99	0.25	0.34	0.23
G1	G2	G3	G4	G5	G6	G7	G8	G9	G10	G11	G12
1.21	1.02	1.21	1.21	1.19	1.14	0.84	0.31	0.98	1.11	0.24	0.42
H1	H2	H3	H4	H5	H6	H7	H8	H9	H10	H11	H12
1.21	1.21	1.21	1.21	1.14	1.21	1.05	1.07	1.14	1.04	0.22	0.66

```
#Plot AUC
ggplot(data = data_sum) +
  geom_text(aes(label = round(auc, 2), x = 1, y = 1)) +
  facet_wrap(~ Well, ncol = 12) +
  labs(title = "Area under the curve by well") +
  theme(axis.title = element_blank(),
        axis.text = element_blank(),
        axis.ticks = element_blank())
```

Area under the curve by well

A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12
15.91	19.27	15.13	20.08	20.31	12.77	1.07	3.52	0.44	5.57	5.99	0.39
B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12
20.88	18.33	9.49	20.42	20.91	17.83	7.71	1.8	0.69	6.15	7.77	1.72
C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12
17.76	21.65	16.6	19.2	8.74	17.82	2.65	9.91	0.56	3.4	0.73	1.91
D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12
21.33	18.29	21.64	19.62	18.56	17.22	8.85	1.8	9.88	4.28	2.17	0.81
E1	E2	E3	E4	E5	E6	E7	E8	E9	E10	E11	E12
21.58	20.22	21.36	15.94	20.71	13.02	9.71	5.94	9.05	0.83	6.95	0.86
F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12
9.63	19.45	20.32	12.88	14.51	12.66	1.68	3.91	6.11	0.54	0.72	0.51
G1	G2	G3	G4	G5	G6	G7	G8	G9	G10	G11	G12
19.06	11.64	19.97	21.67	16.44	14.86	3.11	0.79	5.31	9.78	0.47	0.92
H1	H2	H3	H4	H5	H6	H7	H8	H9	H10	H11	H12
21.38	21.41	21.79	21.42	14.83	18.43	8.84	8.98	10.29	8.86	0.4	2.16

What's next?

In the example here, we've shown how each step of a `gcplyr` workflow is only one or a few lines of code. In the following pages, we've explained each of these steps in depth. To start, we'll learn how to import our data into `R` and transform it into a convenient format.

1. Introduction: `vignette("gc01_gcplyr")`
2. **Importing and reshaping data: `vignette("gc02_import_reshape")`**
3. Incorporating experimental designs: `vignette("gc03_incorporate_designs")`
4. Pre-processing and plotting your data: `vignette("gc04_preprocess_plot")`
5. Processing your data: `vignette("gc05_process")`
6. Analyzing your data: `vignette("gc06_analyze")`
7. Dealing with noise: `vignette("gc07_noise")`
8. Best practices and other tips: `vignette("gc08_conclusion")`
9. Working with multiple plates: `vignette("gc09_multiple_plates")`
10. Using make_design to generate experimental designs: `vignette("gc10_using_make_design")`