gcplyr: an R package for microbial growth curve data analysis

Michael Blazanin

Abstract

Characterization of microbial growth is of both fundamental and applied interest. Modern platforms can automate collection of high-throughput microbial growth curves, necessitating the development of computational tools to handle and analyze these data to produce insights. However, existing tools are limited. Many use parametric analyses that require mathematical assumptions about the microbial growth characteristics. Those that use non-parametric or model-free analyses often can only quantify a few traits of interest, and none are capable of importing and reshaping all known growth curve data formats. To address this gap, here I present a newly-developed R package: gcplyr. gcplyr can flexibly import growth curve data in every known format, and reshape it under a flexible and extendable framework so that users can design custom analyses or plot data with popular visualization packages. gcplyr can also incorporate metadata and generate or import experimental designs to merge with data. Finally, gcplyr carries out model-free and non-parametric analyses, extracting a broad range of clinically and ecologically important traits, including initial density, lag time, growth rate and doubling time, carrying capacity, diauxie, area under the curve, extinction time, and more. In sum, gcplyr makes scripted analysis of growth curve data in R straightforward, streamlines common data wrangling and analysis steps, and easily integrates with common visualization and statistical analyses.

Introduction

Characterization of microbial population growth dynamics has been of fundamental and applied interest since nearly the dawn of microbiology (1). From bacterial interactions with antimicrobials to the effects of mutations in yeast, growth curves are a ubiquitous technique to study microbial growth. Indeed, modern automated platforms, including plate readers, can collect high-throughput growth data over time on hundreds of samples simultaneously. Yet, this data-generation capacity has outpaced the development of computational tools to handle and analyze microbial growth data, presenting new challenges.

First and foremost, data are rarely output in the ideal format for analysis, visualization, or publication. Reorganizing data manually can be tedious and fraught with the potential for introduction of errors. Moreover, since output files vary between different plate readers, scripted reorganization may require tailored code for each output format. Despite this, existing software tools provide limited utilities for streamlined data wrangling and reorganization (Table S1).

Once data are reorganized, scientists face the challenge of converting raw data into quantitative microbial traits. Typically, plate readers measure the optical density of a microbial culture, which corresponds to the density of the population. To convert optical density measures over time into a quantitative microbial trait, many groups have developed software with graphical user interfaces (Table S1, (2–12)). Graphical user interfaces make the tools easy to use, with little or no programming. However, these programs limit the degree to which users can customize their analyses, carry out analyses beyond the options built into the software, or integrate their analyses with scripted approaches to visualization or statistics.

In addition, many of the most popular computational tools use parametric analyses of microbial growth curve data (Table S1, (2, 3, 5, 7, 9, 11–19)). Parametric analyses fit a mathematical model of microbial population growth to observed data, then extract the fitted parameter values to quantify traits. While this approach is useful, it has drawbacks. Namely, users must choose a model to fit, making specific error-prone mathematical assumptions about the form the growth data should take. Moreover, users must then validate that their data meet the assumptions of the model, and verify that optimization algorithms converged on appropriate fits to their data. Most challengingly, some growth dynamics may not fit any known models of microbial growth, leaving researchers with few to no options using programs built on parametric analyses.

Given these drawbacks, in recent years some groups have developed software to analyze microbial growth data with non-parametric or model-free approaches (Table S1, (2, 4, 6, 8, 10, 13–15, 17–24)). These analyses make no specific mathematical assumptions about the form of the growth data, instead extracting parameters of interest directly from the data itself or from non-parametrically smoothed transformations of the data. However, the vast majority of these tools can only quantify a few traits of interest, and all of them have limited data wrangling and reorganization capabilities.

To address these shortcomings, I present my newly-released R package, gcplyr. gcplyr is a software package that can flexibly import growth curve data in all of the instrument output formats of which I am aware. gcplyr is built in R, a popular scripting language for scientific data analysis and visualization. gcplyr provides a framework for data reshaping that is flexible and extendable so that users can easily run custom analyses or integrate gcplyr with existing visualization packages. gcplyr also allows incorporation

of metadata and experimental design information. Finally, gcplyr facilitates model-free and non-parametric analyses, extracting a number of traits of interest, including initial density, lag time, growth rate and doubling time, carrying capacity, diauxie, area under the curve, extinction time, and more. All of these functionalities are extensively documented in tutorials and the user manual, such that only a basic working knowledge in R is sufficient to use the package.

Results

Implementation

gcplyr is an open-source R package available on CRAN and GitHub, with source code available under the permissive MIT License. Within R, gcplyr functions make it easy to import data, merge them with experimental design information, smooth and calculate derivatives as necessary, and analyze curves to produce a number of metrics (Fig 1). These metrics can be easily plotted using ggplot2, combined with other non-growth curve experimental data, and statistically analyzed in R. gcplyr is usable with a basic working knowledge of the R coding language, or by following the available step-by-step tutorials.

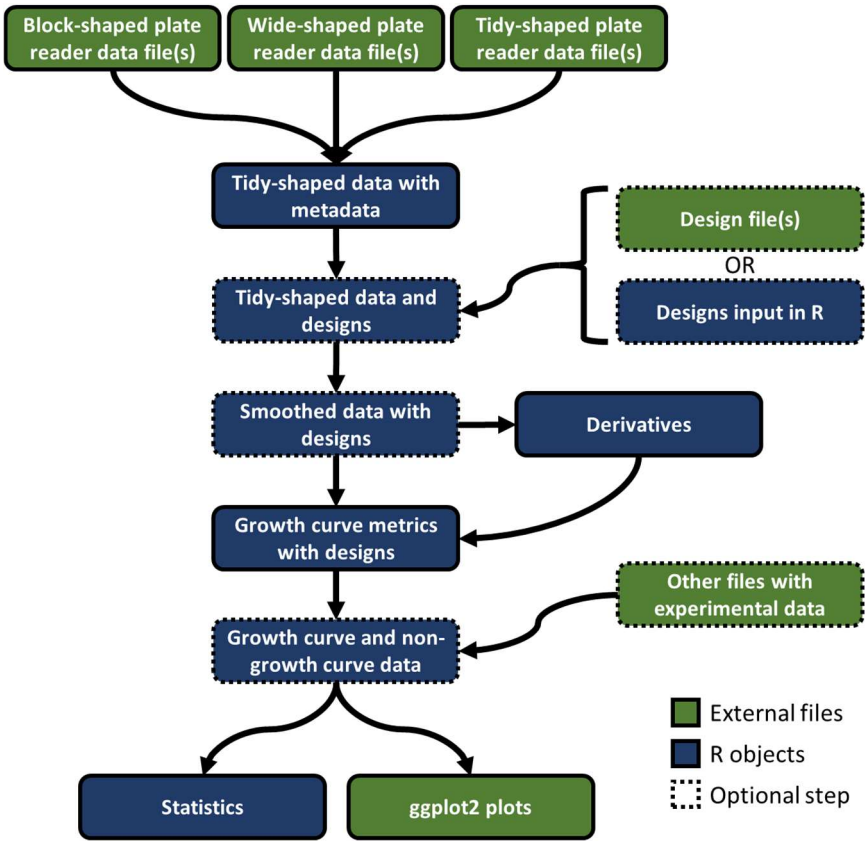

Fig 1. Workflow to use gcplyr to analyze microbial growth curve data. gcplyr functions import and reshape data files into tidy-shaped data, then merge them with imported or user-input experimental design information. Data and designs can then be smoothed and have derivatives calculated to extract growth curve metrics. Metrics can be easily merged with other non-growth curve experimental data before statistical analyses and visualization.

Data reshaping

gcplyr can import a variety of input data formats and then reshape them into a uniform format ideal for subsequent graphic and analysis steps. Every export format I am aware of from plate readers or other similar instruments can be imported by gcplyr (Fig 2). This is an especially useful improvement for block-shaped data, where data are arranged to match the physical layout of the plate it was read from and each timepoint is saved separately. I am not aware of any previous software that could parse block-shaped data files without requiring a custom script; gcplyr streamlines this process with a single function. Once data are imported into R, they are reshaped into a 'tidy' format (also known as 'long' format) (25). Tidy-shaped data have all observations in a single column, with each unique datapoint with its own row, and additional columns specifying the timepoint, well, and any added experimental design information (e.g. bacterial strain). Tidy-shaped data is the best layout for most analyses (25), is consistent with requirements of data repositories like Dryad, and is the expected input for a number of popular R packages (26). By transforming data into a tidy-shape, gcplyr makes it easy for users to also visualize their data using ggplot2 (27), manipulate their data using dplyr (28), or apply any of the other tidyverse packages to their data (26).

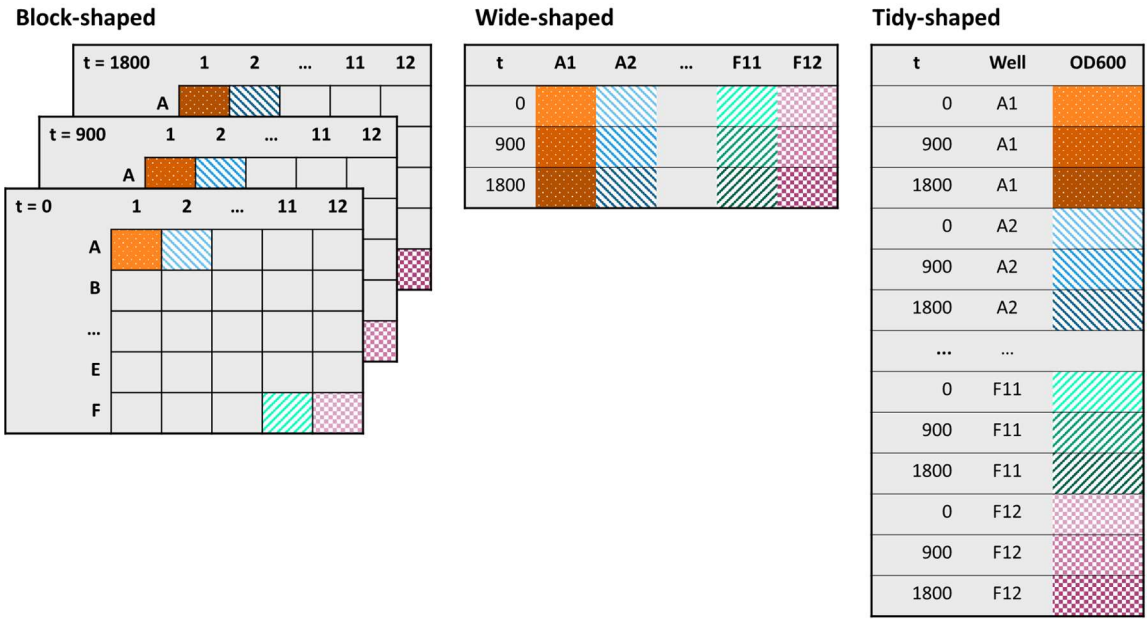

Fig 2. Common microbial growth curve data export formats. Block-shaped data are organized to match the physical layout of the multi-well plate from which they were generated, with each timepoint having its own block. Wide shaped data contain one column for the timepoint and one column for each well from a plate, with each row corresponding to a different timepoint. Tidy-shaped data feature one column for the timepoint, one column for the well identifier, and one column containing all the observations, so each unique data point (well-by-timepoint combination) has its own row. In tidy-shaped data, additional columns (not shown) can contain experimental design information, other data, or both.

Incorporation of metadata and experimental designs

gcplyr can also incorporate metadata and experimental designs with growth curve data. Metadata from input files can be incorporated during file reading. Experimental design information can be incorporated in one of two ways. First, gcplyr can read designs from user-generated spreadsheet files. Second, users can directly input experimental design information. gcplyr functions can also output designs to files for reference or inclusion in a laboratory notebook.

Characterizing microbial growth

To quantify attributes of growth data without model fitting, gcplyr computes metrics of interest from density data and its derivatives. gcplyr can calculate both the derivative and per-capita derivative of density data (Fig 3). gcplyr can then identify features in the density data and its derivatives to quantify traits of interest, including lag time, growth rate, doubling time, and carrying capacity (Table 1, Fig 4A). gcplyr also has the novel ability to detect and quantify diauxic growth (Table 1, Fig 4B), a trait that is common in many microbes but difficult to analyze using existing software tools.

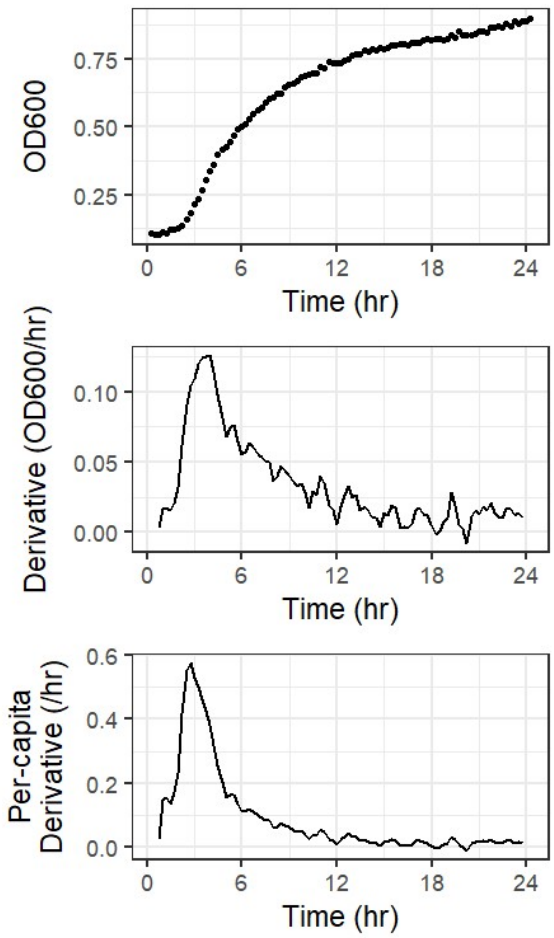

Fig 3. Example of a growth curve, its derivative, and its per-capita derivative. Using an example experimental bacterial growth curve, gcplyr was used to calculate the derivative and per-capita derivative by fitting a linear regression to rolling windows of the plain or log-transformed density values, respectively. These calculations used a window 75 minutes (five data points) wide.

Table 1. Examples of microbial growth traits that can be quantified using gcplyr. Each trait is listed with a brief description of the computational approach used by gcplyr to quantify the trait.

Trait	Description	gcplyr method
Growth rate	The maximum rate of exponential growth per unit time	Maximum of per-capita derivative
Doubling time	The amount of time it would take for the population to double in size when growing exponentially	Minimum of doubling time, which is log(2) divided by per-capita derivative
Area under the curve	An overall measure of bacterial growth	Area under the curve of density
Lag time	The amount of time that passes between the start of a growth curve and the beginning of exponential growth	Projection of maximum growth rate fit back to starting density
Carrying capacity	The maximum density that can be reached in the environment	Maximum of density
Diauxic shift	The time when the microbe metabolically shifts from growing on one resource to growing on another (less-favored) one	Time of the first local minima of derivative
Growth rate during diauxie	The maximum rate of exponential growth per unit time post diauxic shift	Maximum of per-capita derivative after diauxic shift
Doubling time during diauxic shift	The amount of time for it would take for the population to double in size when growing exponentially post diauxic shift	Minimum of doubling time after diauxic shift
Peak density	The peak density reached before declining towards extinction, most frequently used to quantify interactions with antagonists, like between bacteria and phages	Density of first local maxima of density
Peak time	The time when density peaks before declining towards extinction, most frequently used to quantify interactions with antagonists, like between bacteria and phages	Time of first local maxima of density
Near-extinction time	The time when density falls below some threshold that denotes near or complete extinction of the population	Time when density first drops below threshold

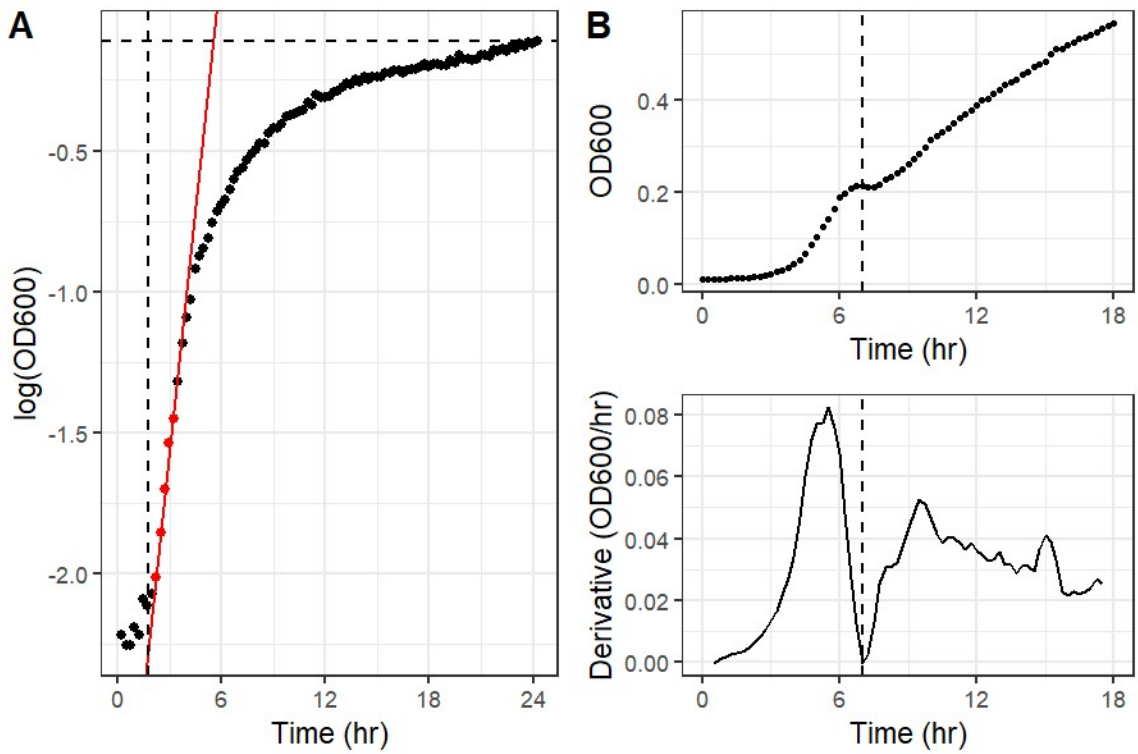

Fig 4. Example of a growth curve demonstrating calculation of microbial traits. **A.** Depiction of the calculation of lag time, maximum growth rate, and maximum density reached in 24 hours on an example experimental bacterial growth curve. Maximum growth rate was determined by finding the maximum of the per-capita growth rate (slope of the red line originating from the red points). Lag time was calculated as the point where the red line intersects with the initial density on log-transformed axes (vertical dashed line). Maximum density in 24 hours was simply the maximum density (horizontal dashed line). **B.** Depiction of the identification of diauxic growth in an example experimental bacterial growth curve. Diauxie was identified by using gcplyr's local extrema finding function to find a local minimum in the derivative curve, yielding an accurate identification of the time when the diauxic shift occurred (vertical dashed line).

<u>Characterizing bacterial growth in the presence of phages</u>

Microbial growth curves can also be used to characterize interactions between bacteria and antagonists. One frequent application of this is to quantify interactions with lytic phages (29–42), a use-case which existing software has limited capacity to handle. In these curves, the phage densities cannot be directly visualized, but changes in the bacterial density can be. The bacterial density tends to initially increase, before peaking and declining due to phage lysis. Metrics like peak density, time until near-extinction, and area under the curve can be used to, for example, compare the susceptibility of different bacterial strains to a focal phage (33, 34, 37–39, 41). gcplyr can directly calculate all three metrics (Fig 5).

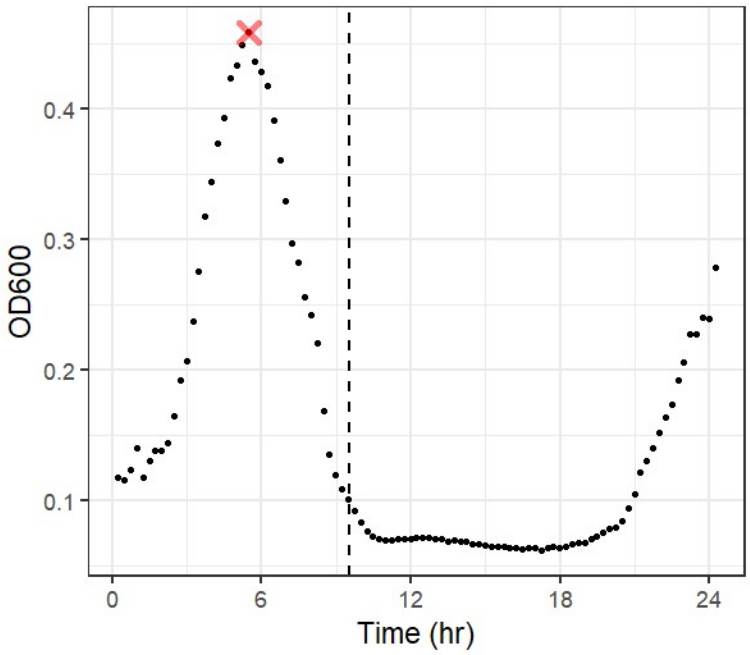

Fig 5. Example of an experimental growth curve of bacteria in the presence of lytic phages. Peak density and near-extinction time act as proxy measures for the bacterial strain's sensitivity to the phage. Peak density (red "X") was identified by using gcplyr's local extrema finding function to identify a local maximum in the density data. Near-extinction time (vertical dashed line) was identified by finding the first point the OD600 dropped below 0.1 using gcplyr's threshold-detection function.

Dealing with noise in growth curve data

One frequent challenge with experimental growth curve data is the presence of noise. Noise can obscure underlying traits of interest. gcplyr currently has two methods to deal with noise: 1) smoothing raw density data, and 2) using fitting during derivative calculations.

In many cases, noise can be smoothed directly from the density data. gcplyr implements several well-established smoothing algorithms (Fig 6A), including moving average, moving median, LOESS (43–46), and GAM (47–50). Each of these algorithms is tunable by user-set parameters, and they can be applied individually or in sequence.

In some cases, smoothing density data itself may not be necessary or sufficient. In particular, derivatives, especially per-capita derivatives, are often sensitive to experimental noise. By default, gcplyr calculates derivatives for each pair of subsequent points. However, to deal with noisy data, gcplyr can calculate derivatives by fitting a linear regression with a rolling window of multiple points (Fig 6B). The rolling regression approach has been implemented elsewhere, and can improve accuracy and reduce the effects of noise on calculated derivatives (21).

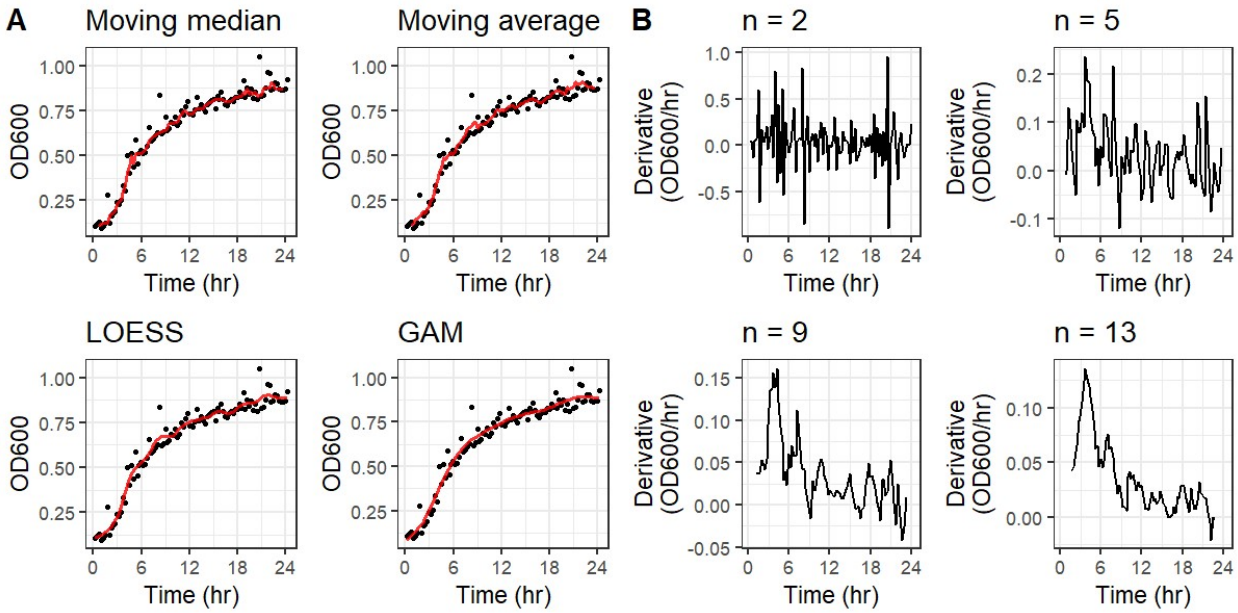

Fig 6. gcplyr methods for dealing with noisy data. Example data is an experimental bacterial growth curve with added simulated noise. **A.** Example of growth curve with added noise demonstrating smoothing algorithms. Noisy density data (points) was smoothed with each algorithm (red line) using gcplyr's data smoothing function. Moving median and moving average were smoothed using windows of 5 data points (75 minutes), LOESS was used with a span of 0.2, and GAM was used with 20 knots. **B.** Example of fitting during derivative calculation to reduce the effects of noise. The derivative of the data points in A was calculated using gcplyr's derivative calculation function with rolling windows of 2, 5, 9, or 13 data points (30, 75, 135, or 195 minutes).

Discussion

Modern technology has accelerated the generation of high-throughput microbial growth data, necessitating new computational tools capable of handling and analyzing this data to produce insights. Here I introduced gcplyr, a new R package built specifically to address this need. gcplyr can flexibly import growth curve data in every format of which I am aware, combine data with experimental design information, reshape data for analysis and use with other popular R packages, and compute a number of growth curve metrics using model-free and non-parametric analyses.

gcplyr improves on existing computational tools by implementing improved data wrangling capabilities. gcplyr's input format requirements are less restrictive and more flexible than existing tools, freeing users from the need to reformat files manually. Moreover, gcplyr's capacity for data reorganization goes well beyond that of existing tools (Table S1). This data organization framework provides a number of benefits over existing implementations:

1. It allows users to integrate as many pieces of experimental design information as desired.
2. It allows users to easily integrate their growth curve analyses with existing visualization and statistics packages in R.
3. It allows users to merge growth curve data analyses with other sources of data.

4. It allows users to leverage the general-use functions in gcplyr and other packages to generate custom analyses to identify unique features in their data.

Additionally, gcplyr improves on existing computational tools by providing an expanded array of traits that can be quantified using model-free and non-parametric approaches. Previous tools had fit parametric mathematical models of microbial growth to observed data (2, 3, 5, 7, 9, 11–19), an approach that requires validation of model assumptions and fails when data do not fit the chosen model. Instead, gcplyr and some recent tools quantify traits directly from the data itself (2, 4, 6, 8, 10, 13–15, 17–24). In comparison to most existing tools, gcplyr expands on the number of possible traits to be quantified and facilitates analyses of a greater diversity of growth curve phenomena.

Growth curves are a widespread experimental approach in the microbial sciences. From bacterial interactions with antimicrobials to investigating the effects of genetic manipulations in yeast, growth curves are used to study microbial growth. However, until now, tools capable of wrangling and doing model-free analysis of growth curve data were limited. By enabling these steps, gcplyr lubricates high-quality model-free analysis for a wide range of applied and fundamental research on microbial growth.

Materials and Methods

Availability and dependencies of gcplyr

gcplyr is written in the open-source R programming language (46), and is available for free to use. gcplyr can be installed from the centralized CRAN repository (https://CRAN.R-project.org/package=gcplyr) with the built-in install.packages function, or from GitHub (https://github.com/mikeblazanin/gcplyr). Installation requirements will continue to adjust as active development on gcplyr continues, but are always listed on the CRAN page. gcplyr has been written to minimize the number of external dependencies, thus simplifying the installation.

Documentation

Extensive documentation for gcplyr is available online (https://mikeblazanin.github.io/gcplyr/) and in the vignettes and user manual bundled with installations of gcplyr, including tutorials that walk through each step of an analysis.

Inputs.

gcplyr can import files in any tabular file format (e.g. .csv, .xls, .xlsx, .tsv). gcplyr can also import data and design files regardless of whether they are block-shaped, wide-shaped, or tidy-shaped, which includes every instrument output format I am aware of (Fig 2).

Data reshaping.

Once data is imported, gcplyr extends the functionality of dplyr (28) to transform block-shaped and wide-shaped data and designs into tidy-shaped data and designs.

Incorporating experimental design information.

gcplyr can import experimental designs from user-created tabular files, or can generate designs within R. Designs generated within R can include as many experimental design fields as desired, and can be output to a spreadsheet file. Regardless of how designs are generated, they can then be merged with data.

Calculation of derivatives.

Growth curve metrics can be calculated in a model-free way by identifying features of the density data and its derivatives. To facilitate this, gcplyr can calculate both plain and per-capita derivatives. Plain derivatives are calculated simply as the slope of the density data over time. Per-capita derivatives can be calculated as the plain derivative divided by the density, or as the slope of the log-transformed density over time. By default, gcplyr uses each pair of subsequent points to calculate derivatives. However, gcplyr can also calculate derivatives by fitting a linear regression to all points within a window centered at each data point, with user-set parameters determining the width of the window. The package documentation discusses best practices for setting derivative parameters and calculating derivatives.

Smoothing data.

Model-free analyses can be sensitive to experimental noise in growth curve data. To smooth noise in raw density data, gcplyr implements several well-established smoothing functions, including moving average, moving median, loess (43–46), and GAM (47–50). Smoothing functions are tuned by user-set smoothness parameters. The package documentation discusses best practices for setting smoothness parameters and smoothing data.

Analysis functions.

A number of growth curve metrics can be quantified by identification of local extrema (Table 1). gcplyr includes a function that can identify local extrema by iteratively searching a window centered at each data point for the maximum and minimum data in the window until the algorithm converges. User-set parameters determine the width and height of the window, tuning the sensitivity of the algorithm to narrower and shallower peaks and valleys, respectively. The package documentation discusses best practices for setting local extrema finding parameters and identifying local extrema.

Threshold-crossing events represent another important class of growth curve metrics. For example, the near-extinction time of a bacterial population can be estimated as the time required for optical density to drop below a threshold value (Table 1). To identify such events, gcplyr includes a general threshold-crossing finding function that finds times when the data or derivative crosses the user-defined threshold in the direction(s) specified by the user.

gcplyr also includes some specialized analysis functions for specific metrics. Lag time is calculated using an established approach (51, 52): finding the maximum per-capita growth rate, then projecting that rate as a tangent line from the point of maximum growth until it intersects with the initial density on a log-transformed y-axis (Fig 4). Area under the curve is simply calculated using the trapezoid rule to get an exact definite integral.

Example microbial growth data

To generate experimental microbial growth curve data for examples, *Pseudomonas fluorescens* SBW25 (53) was grown at 28 °C in King's B media: 10g/L LP0037 Oxoid Bacteriological Peptone, 15 g/L glycerol, 1.5 g/L potassium phosphate, and 0.6 g/L magnesium sulfate. Bacteria were inoculated in a total volume

of 200 µL to an initial density of 10^5 cfu/mL, then grown shaking in a BioTek Epoch 2 microplate spectrophotometer. For the phage-bacteria co-culture growth curve, phage Phi2 (54) was inoculated at an initial MOI of 0.01.

For experimental microbial growth curve data demonstrating diauxic growth, I pulled growth curve data of ancestral *Pseudomonas fluorescens* SBW25 from (55). To summarize their methods, 4 µL of an overnight bacterial culture was added to a 96 well plate with 146 µL of modified King's B media: 2.5g/L LP0037 Oxoid Bacteriological Peptone, 3.75 g/L glycerol, 0.75 g/L potassium phosphate, and 0.3 g/L magnesium sulfate. Bacteria were grown overnight at 29°C while shaking with the OD600 read every 15 minutes.

All data and code to generate the plots in this paper are available at https://github.com/mikeblazanin/gcplyr/tree/master/manuscript.

Acknowledgements

Thanks to Paul Turner for supporting me while I worked on the project. Thanks to Jyot Antani, Alita Burmeister, Noah Houpt, and Jordan Lewis for feedback on drafts of this paper. Thanks to Emma Vasen for collecting the example growth curve data. Thanks to a number of alpha and beta testers who provided feedback on the package before release.

References

1. Monod J. 1949. The Growth of Bacterial Cultures. Annual Review of Microbiology 3:371–394.

2. Wirth NT, Funk J. 2023. QurvE: Robust and User-Friendly Analysis of Growth and Fluorescence Curves. https://CRAN.R-project.org/package=QurvE.

3. Coutin NPJ, Giaever G, Nislow C. 2020. Interactively AUDIT Your Growth Curves with a Suite of R Packages. G3 10:933–943.

4. Swain PS, Stevenson K, Leary A, Montano-Gutierrez LF, Clark IBN, Vogel J, Pilizota T. 2016. Inferring time derivatives including cell growth rates using Gaussian processes. 1. Nat Commun 7:13766.

5. Bukhman YV, DiPiazza NW, Piotrowski J, Shao J, Halstead AGW, Bui MD, Xie E, Sato TK. 2015. Modeling Microbial Growth Curves with GCAT. Bioenergy Research 8:1022–1030.

6. Fernandez-Ricaud L, Kourtchenko O, Zackrisson M, Warringer J, Blomberg A. 2016. PRECOG: A tool for automated extraction and visualization of fitness components in microbial growth phenomics. BMC Bioinformatics 17:249.

7. Huang L. 2014. IPMP 2013 — A comprehensive data analysis tool for predictive microbiology. International Journal of Food Microbiology 171:100–107.

8. Jung PP, Christian N, Kay DP, Skupin A, Linster CL. 2015. Protocols and Programs for High-Throughput Growth and Aging Phenotyping in Yeast. PLoS ONE 10:e0119807.

9. Liu Y, Wang X, Liu B, Yuan S, Qin X, Dong Q. 2021. Microrisk Lab: An Online Freeware for Predictive Microbiology. Foodborne Pathogens and Disease 18:607–615.

10. Olsen B, Murakami CJ, Kaeberlein M. 2010. YODA: Software to facilitate high-throughput analysis of chronological life span, growth rate, and survival in budding yeast. BMC Bioinformatics 11:141.

11. Veríssimo A, Paixão L, Neves AR, Vinga S. 2013. BGFit: Management and automated fitting of biological growth curves. BMC Bioinformatics 14.

12. Vervier K, Browne HP, Lawley TD. 2019. CarboLogR: a Shiny/R application for statistical analysis of bacterial utilisation of carbon sources. bioRxiv https://doi.org/10.1101/695676.

13. Sprouffske K, Wagner A. 2016. Growthcurver: An R package for obtaining interpretable metrics from microbial growth curves. BMC Bioinformatics 17:17–20.

14. Petzoldt T. 2022. growthrates: Estimate Growth Rates from Experimental Data. https://CRAN.R-project.org/package=growthrates.

15. Vaas LAI, Sikorski J, Hofner B, Fiebig A, Buddruhs N, Klenk HP, Göker M. 2013. Opm: An R package for analysing OmniLog® phenotype microarray data. Bioinformatics 29:1823–1824.

16. Garre A, Koomen J, Besten H den, Zwietering M. 2022. biogrowth: Modelling of Population Growth. https://CRAN.R-project.org/package=biogrowth.

17. Cuevas DA, Edwards RA. 2017. PMAnalyzer: A new web interface for bacterial growth curve analysis. Bioinformatics 33:1905–1906.

18. Cuevas DA, Garza D, Sanchez SE, Rostron J, Henry CS, Vonstein V, Overbeek RA, Segall A, Rohwer F, Dinsdale EA, Edwards RA. 2016. Elucidating genomic gaps using phenotypic profiles. F1000Research https://doi.org/10.12688/f1000research.5140.2.

19. Osthege M, Tenhaef N, Zyla R, Müller C, Hemmerich J, Wiechert W, Noack S, Oldiges M. 2022. bletl - A Python package for integrating BioLector microcultivation devices in the Design-Build-Test-Learn cycle. Engineering in Life Sciences 22:242–259.

20. Midani FS, Collins J, Britton RA. 2021. AMiGA: Software for Automated Analysis of Microbial Growth Assays. mSystems 6:e00508-21.

21. Hall BG, Acar H, Nandipati A, Barlow M. 2014. Growth rates made easy. Molecular Biology and Evolution 31:232–238.

22. Tonner PD, Darnell CL, Bushell FML, Lund PA, Schmid AK, Schmidler SC. 2020. A Bayesian non-parametric mixed-effects model of microbial growth curves. PLOS Computational Biology 16:e1008366.

23. Tonner PD, Darnell CL, Engelhardt BE, Schmid AK. 2017. Detecting differential growth of microbial populations with Gaussian process regression. Genome Res 27:320–333.

24. Hemmerich J, Wiechert W, Oldiges M. 2017. Automated growth rate determination in high-throughput microbioreactor systems. BMC Research Notes 10:617.

25. Wickham H. Tidy Data. Journal of Statistical Software 10.

26. Wickham H, Averick M, Bryan J, Chang W, McGowan LD, François R, Grolemund G, Hayes A, Henry L, Hester J, Kuhn M, Pedersen TL, Miller E, Bache SM, Müller K, Ooms J, Robinson D, Seidel DP, Spinu V, Takahashi K, Vaughan D, Wilke C, Woo K, Yutani H. 2019. Welcome to the tidyverse. Journal of Open Source Software 4:1686.

27. Wickham H. 2016. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York. https://ggplot2.tidyverse.org.

28. Wickham H, François R, Henry L, Müller K. 2023. dplyr: A Grammar of Data Manipulation. https://CRAN.R-project.org/package=dplyr.

29. Sørensen PE, Ng DYK, Duchateau L, Ingmer H, Garmyn A, Butaye P. 2021. Classification of In Vitro Phage–Host Population Growth Dynamics. 12. Microorganisms 9:2470.

30. Maillard J-Y, Beggs T s., Day M j., Hudson R a., Russell A d. 1996. The use of an automated assay to assess phage survival after a biocidal treatment. Journal of Applied Bacteriology 80:605–610.

31. Glonti T, Pirnay J-P. 2022. In Vitro Techniques and Measurements of Phage Characteristics That Are Important for Phage Therapy Success. 7. Viruses 14:1490.

32. Martinez-Soto CE, Cucić S, Lin JT, Kirst S, Mahmoud ES, Khursigara CM, Anany H. 2021. PHIDA: A High Throughput Turbidimetric Data Analytic Tool to Compare Host Range Profiles of Bacteriophages Isolated Using Different Enrichment Methods. 11. Viruses 13:2120.

33. Rajnovic D, Muñoz-Berbel X, Mas J. 2019. Fast phage detection and quantification: An optical density-based approach. PLOS ONE 14:e0216292.

34. Ceballos RM, Stacy CL. 2020. Quantifying relative virulence: when μ max fails and AUC alone just is not enough. J Gen Virol 102:jgv001515.

35. Rajnovic D, Mas J. 2020. Fluorometric detection of phages in liquid media: Application to turbid samples. Analytica Chimica Acta 1111:23–30.

36. Borin JM, Avrani S, Barrick JE, Petrie KL, Meyer JR. 2021. Coevolutionary phage training leads to greater bacterial suppression and delays the evolution of phage resistance. Proc Natl Acad Sci USA 118:e2104592118.

37. Xie Y, Wahab L, Gill J. 2018. Development and Validation of a Microtiter Plate-Based Assay for Determination of Bacteriophage Host Range and Virulence. Viruses 10:189.

38. Konopacki M, Grygorcewicz B, Dołęgowska B, Kordas M, Rakoczy R. 2020. PhageScore: A simple method for comparative evaluation of bacteriophages lytic activity. Biochemical Engineering Journal 161:107652.

39. Storms ZJ, Teel MR, Mercurio K, Sauvageau D. 2020. The Virulence Index: A Metric for Quantitative Analysis of Phage Virulence. PHAGE 1:27–36.

40. Cooper CJ, Denyer SP, Maillard JY. 2011. Rapid and quantitative automated measurement of bacteriophage activity against cystic fibrosis isolates of Pseudomonas aeruginosa. Journal of Applied Microbiology 110:631–640.

41. Turner PE, Draghi JA, Wilpiszeski R. 2012. High-throughput analysis of growth differences among phage strains. Journal of Microbiological Methods 88:117–121.

42. Henry M, Biswas B, Vincent L, Mokashi V, Schuch R, Bishop-Lilly KA, Sozhamannan S. 2012. Development of a high throughput assay for indirectly measuring phage growth using the OmniLog TM system. Bacteriophage 2:159–167.

43. Savitzky A, Golay MJ. 1964. Smoothing and differentiation of data by simplified least squares procedures. Analytical chemistry 36:1627–1639.

44. Cleveland WS, Devlin SJ. 1988. Locally weighted regression: an approach to regression analysis by local fitting. Journal of the American statistical association 83:596–610.

45. Cleveland WS. 1979. Robust locally weighted regression and smoothing scatterplots. Journal of the American statistical association 74:829–836.

46. R Core Team. 2022. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria. https://www.R-project.org/.

47. Trevor Hastie, Robert Tibshirani. 1986. Generalized Additive Models. Statistical Science 1:297–310.

48. Wood SN. 2003. Thin-plate regression splines. Journal of the Royal Statistical Society (B) 65:95–114.

49. Wood SN. 2017. Generalized Additive Models: An Introduction with R, 2nd ed. Chapman and Hall/CRC.

50. Wood SN. 2011. Fast stable restricted maximum likelihood and marginal likelihood estimation of semiparametric generalized linear models. Journal of the Royal Statistical Society (B) 73:3–36.

51. Swinnen IAM, Bernaerts K, Dens EJJ, Geeraerd AH, Van Impe JF. 2004. Predictive modelling of the microbial lag phase: a review. International Journal of Food Microbiology 94:137–159.

52. Yates GT, Smotzer T. 2007. On the lag phase and initial decline of microbial growth curves. Journal of Theoretical Biology 244:511–517.

53. Rainey PB, Travisano M. 1998. Adaptive radiation in a heterogeneous environment. Nature 394:69–72.

54. Buckling A, Rainey PB. 2002. Antagonistic coevolution between a bacterium and a bacteriophage. Proceedings of the Royal Society B 269:931–6.

55. Blazanin M, Moore JP, Olsen S, Travisano M. 2023. Fight not flight: parasites drive the bacterial evolution of resistance, not avoidance. bioRxiv https://doi.org/10.1101/2023.04.29.538831.

Table S1. Comparison of gcplyr with other available microbial growth curve analysis software and code.

Software	Citation	Available as	Scriptable	Imports	Data reshaping	Incorporate designs	Plotting	Parametric fitting & analysis	Smoothing	Non-parametric analysis
gcplyr	This paper	R package	Yes	Block, wide, tidy	Yes, into wide or tidy	Yes	With ggplot2	No	Yes	Yes
QurvE	(2)	GUI or R package	Yes	Wide, specialized formats	From specialized formats into wide	Yes	Built-in	Yes	Yes	Yes
growthcurver	(13)	R package	Yes	Wide	No	No	Built-in	Yes	No	Minimally
growthrates	(14)	R package	Yes	Tidy	No	No	With ggplot2	Yes	Yes	Minimally
opm	(15)	R package	Yes	Specialized formats	Yes, into opm-specific class	Yes	Built-in	Yes (but defunct)	Yes	Minimally
AUDIT	(3)	GUI, R-based	No	Tidy, specialized formats	From specialized formats into tidy	Yes	Built-in	Yes	Yes	No
growr (part of AUDIT)	(3)	R package	Yes	Tidy	N/A	N/A	With ggplot2	Yes	Yes	No
mtpview1 (part of AUDIT)	(3)	R package	Yes	Tidy	N/A	N/A	Built-in	N/A	N/A	N/A
biogrowth	(16)	R package	Yes	Wide	No	No	Built-in	Yes	No	No
AMiGA	(20)	Python package or command line	Some	Wide	No	Yes	Built-in	No	Yes	Yes
PMAnalyzer	(17, 18)	bash, R and Python-based	No	Wide	No	No	Built-in	Yes	No	Minimally
GrowthRates	(21)	Command line	No	Wide, specialized formats	Yes	No	No	No	No	Yes
bletl	(19)	Python package	Yes	Specialized formats	Yes	Yes	Built-in	Minimally	Yes	Yes
fitderiv	(4)	GUI or Python package	Some	Wide	No	No	Built-in	No	Yes	Yes
phenom	(22)	Python code	Yes	Wide	No	Yes	No	No	Yes	Yes
B-GREAT	(23)	Python code	Yes	Wide	No	Yes	No	No	Yes	Yes
Hemmerich et al code	(24)	MATLAB code	Yes	One well at a time	No	No	No	No	No	Yes
GCAT	(5)	GUI, R-based	No	Wide	No	Yes	Built-in	Yes	No	No
PRECOG	(6)	GUI	No	Wide	No	No	Built-in	No	Yes	Yes
IPMP 2013	(7)	GUI, Python-based	No	Wide	No	No	Built-in	Yes	No	No
GATHODE	(8)	GUI, Python-based	No	Wide	No	Minimally	Built-in	No	Yes	Yes
Microrisk Lab	(9)	GUI, R-based	No	Wide	No	No	Built-in	Yes	No	No
YODA	(10)	Webpage via server	No	Wide	No	No	No	No	No	Yes
BGFit	(11)	Webpage	No	Wide	No	No	Built-in	Yes	No	No
CarboLogR	(12)	GUI, R-based	No	Specialized formats	No	Yes	Built-in	Yes	No	No