AMERICAN SOCIETY FOR MICROBIOLOGY
Microbiology® Resource Announcements
DATABASES AND SOFTWARE

sparse-growth-curve: a Computational Pipeline for Parsing Cellular Growth Curves with Low Temporal Resolution

Chuankai Cheng,[a] J. Cameron Thrash[a]

[a]Department of Biological Sciences, University of Southern California, Los Angeles, California, USA

ABSTRACT Here, we introduce a Python-based repository, sparse-growth-curve, a software package designed for parsing cellular growth curves with low temporal resolution. The repository uses cell density and time data as the input, automatically separates different growth phases, calculates the exponential growth rates, and produces multiple graphs to aid in interpretation.

The cellular growth curve arising from cell density variation over time is an important phenotype for cell physiology and systems biology studies (1). Improving the throughput of growth rate calculations from growth curves benefits from automated curve-fitting algorithms, but most existing software has been developed for data with high temporal resolution (2–8). Such data are usually generated by automated optical density measurements (5), which apply only to a limited set of organisms that can achieve high cell densities. Absolute cell count quantification (cells per milliliter) can be labor-intensive for high-temporal-resolution growth curves when methods like flow cytometry or microscopy are used. Therefore, we have developed a growth-curve-fitting pipeline specifically for characterizing sparse data.

The sparse-growth-curve method takes an array of cell density X_i and the corresponding acquisition time t_i (see the template Excel file at GitHub, https://github.com/thrash-lab/sparse-growth-curve) as the input (i.e., typical growth data) (Fig. 1A). The method differentiates growth phases and returns a piecewise linear fit on a logarithmic scale (Fig. 1B). The growth phases are determined through decision tree regression (Fig. 1C), based on defining each growth phase as a period with similar instantaneous growth rates (see https://github.com/thrash-lab/sparse-growth-curve/blob/main/detailed_method.md for detailed methods). The result is automated detection of exponential phase in sparse data and calculation of growth rates according to linear regression.

The software package can analyze multiple growth curves from different replicates, strains, or conditions (e.g., salinity, temperature, or substrate). The output includes multiple products, including (i) plots of the raw growth data on a logarithmic scale, including the best-fit linear regressions for exponential phase; (ii) box plots of growth rates separated by treatment and colored by replicates; (iii) growth rate calculations in both doubling and specific growth rates, with output in an Excel file; and (iv) multistrain/replicate plots of growth rates for experimental designs with continuous variables (e.g., temperature) (Fig. 1D).

Data availability. sparse-growth-curve is available on GitHub (https://github.com/thrash-lab/sparse-growth-curve) under MIT license. The code can be run directly in the browser through Google Colaboratory (https://colab.research.google.com/notebooks/intro.ipynb) without any configuration required on a local machine. The code in iPython Notebook v5.5.0 format illustrates the detailed steps of the methods. There are three separate notebooks, i.e., (i) a notebook showing how a single growth curve (X_i versus t_i) (Fig. 1A) is fitted (Fig. 1B); (ii) a notebook showing how a single Excel file with multiple growth curves is parsed and characterized; and (iii) a

Citation Cheng C, Thrash JC. 2021. sparse-growth-curve: a computational pipeline for parsing cellular growth curves with low temporal resolution. Microbiol Resour Announc 10:e00296-21. https://doi.org/10.1128/MRA.00296-21.

Editor Irene L. G. Newton, Indiana University, Bloomington

Address correspondence to J. Cameron Thrash, thrash@usc.edu.

Received 22 March 2021
Accepted 16 April 2021
Published 13 May 2021

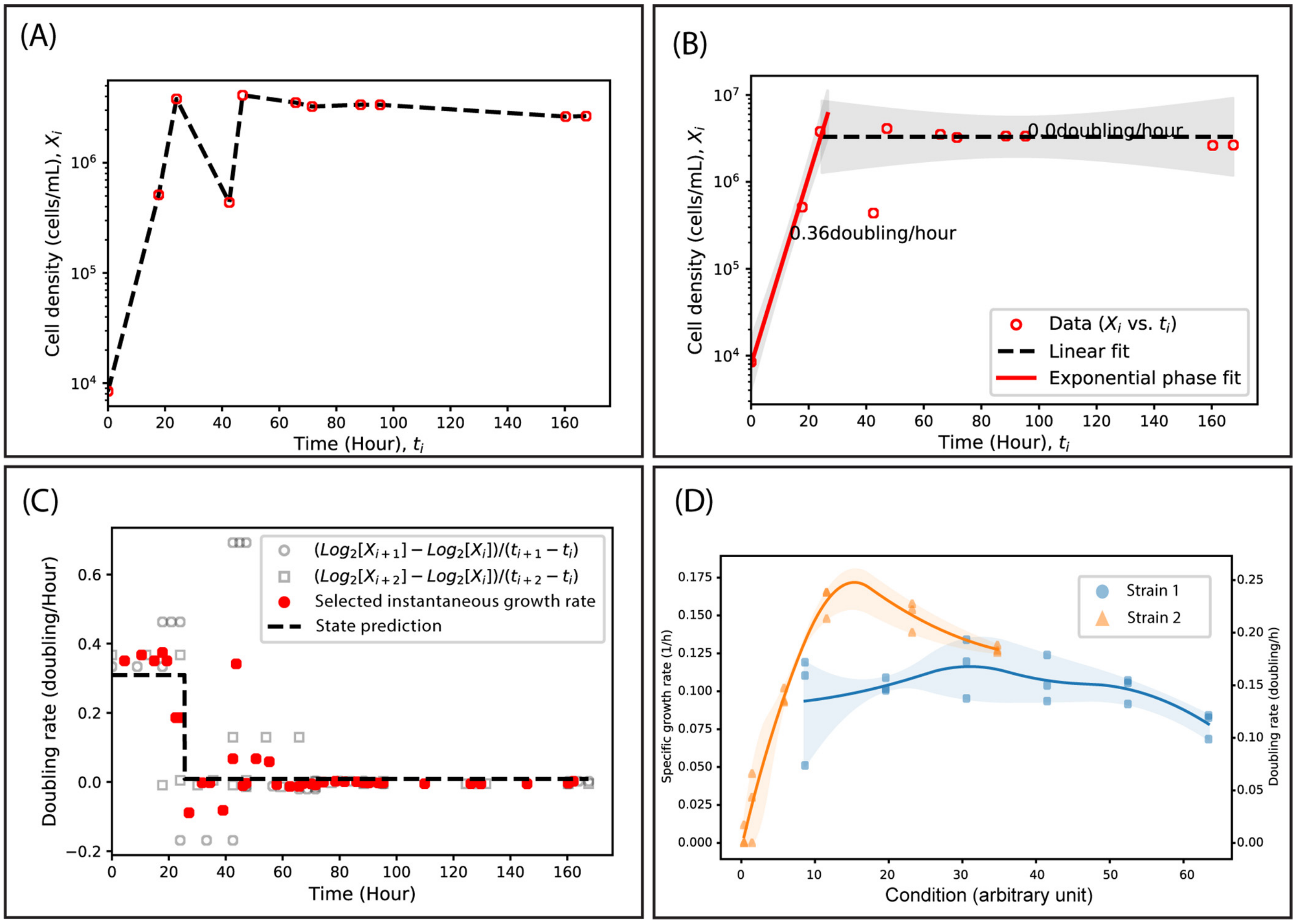

FIG 1 Overview of the process within sparse-growth-curve. (A) The input for the method (red open circles connected by dashed lines) is cell density X_i versus time t_i. Here, we use an imperfect and very sparse curve to illustrate the functionality of the software. (B) The output of the method is piecewise linear regression for each phase. The exponential phase is highlighted in red. The doubling rates are labeled. (C) The subdomains of the piecewise fit in plot B are determined through decision tree regression. Here, gray open circles and squares represent the instantaneous doubling rates calculated from the input data. Red solid squares represent the noise-removed rates versus time for decision tree regression. The state prediction (dashed line) ends up as a step function, and each step corresponds to a growth phase. (D) An extended feature of the sparse-growth-curve package is that the package imports experimental growth curves of multiple strains under multiple conditions. Exponential growth rates for all of the growth curves are calculated. The conversion of doubling rate γ to specific growth rate λ is $\lambda = \ln(2)\gamma$. The solid lines are the interpolated connection mean growth rates, and the shaded areas indicate the range between the maximum and minimum growth rates.

notebook that can handle multiple Excel files (people may want to put growth curves collected from different experimental runs in separate files) and analyze them all in a batch. The Python v3.7.10 (9) code relies on multiple packages (which are automatically installed by running the script in the Google Colab notebooks), including NumPy v1.19.5 (10) for numerical analysis, SciPy v1.4.1 (11) and Scikit-learn v0.22.2 (12) for statistical analysis, Pandas v1.1.5 (13) for handling table format data, Matplotlib v3.2.2 (14) for data visualization, and XlsxWriter v1.3.9 (https://github.com/jmcnamara/XlsxWriter) for writing Excel files. Note that, as time goes on, the package versions will be updated within the Google Colab environment.

ACKNOWLEDGMENTS

This work was supported by the Simons Foundation (Early Career Investigator in Marine Microbial Ecology and Evolution Award) and NSF Biological Oceanography Program grants (OCE-1747681 and OCE-1945279) to J.C.T.

We thank Jordan T. Coelho, Shelby J. Barnes, Nicole R. Ratib, V. Celeste Lanclos, and Brittany D. Bennet for testing and providing feedback on the software.

REFERENCES

1. Schaechter M. 2006. From growth physiology to systems biology. Int Microbiol 9:157–161.
2. 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. https://doi.org/10.1186/s12859-016-1134-2.
3. Sprouffske K, Wagner A. 2016. Growthcurver: an R package for obtaining interpretable metrics from microbial growth curves. BMC Bioinformatics 17:172. https://doi.org/10.1186/s12859-016-1016-7.
4. Cuevas DA, Edwards RA. 2017. PMAnalyzer: a new Web interface for bacterial growth curve analysis. Bioinformatics 33:1905–1906. https://doi.org/10.1093/bioinformatics/btx084.
5. Tonner PD, Darnell CL, Engelhardt BE, Schmid AK. 2017. Detecting differential growth of microbial populations with Gaussian process regression. Genome Res 27:320–333. https://doi.org/10.1101/gr.210286.116.
6. 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 Comput Biol 16:e1008366. https://doi.org/10.1371/journal.pcbi.1008366.
7. Midani FS, Collins J, Britton RA. 2020. AMiGA: software for automated analysis of microbial growth assays. Cold Spring Harbor Laboratory, Cold Spring Harbor, NY.
8. Tong M, French S, El Zahed SS, Ong WK, Karp PD, Brown ED. 2020. Gene dispensability in *Escherichia coli* grown in thirty different carbon environments. mBio 11:e02259-20. https://doi.org/10.1128/mBio.02865-20.
9. Van Rossum G, Drake FL, Jr. 1995. Python tutorial. Centrum voor Wiskunde en Informatica, Amsterdam, Netherlands.
10. Harris CR, Millman KJ, van der Walt SJ, Gommers R, Virtanen P, Cournapeau D, Wieser E, Taylor J, Berg S, Smith NJ, Kern R, Picus M, Hoyer S, van Kerkwijk MH, Brett M, Haldane A, Del Río JF, Wiebe M, Peterson P, Gérard-Marchant P, Sheppard K, Reddy T, Weckesser W, Abbasi H, Gohlke C, Oliphant TE. 2020. Array programming with NumPy. Nature 585:357–362. https://doi.org/10.1038/s41586-020-2649-2.
11. Virtanen P, Gommers R, Oliphant TE, Haberland M, Reddy T, Cournapeau D, Burovski E, Peterson P, Weckesser W, Bright J, van der Walt SJ, Brett M, Wilson J, Millman KJ, Mayorov N, Nelson ARJ, Jones E, Kern R, Larson E, Carey CJ, Polat İ, Feng Y, Moore EW, VanderPlas J, Laxalde D, Perktold J, Cimrman R, Henriksen I, Quintero EA, Harris CR, Archibald AM, Ribeiro AH, Pedregosa F, van Mulbregt P, SciPy 1.0 Contributors. 2020. SciPy 1.0: fundamental algorithms for scientific computing in Python. Nat Methods 17:261–272. https://doi.org/10.1038/s41592-019-0686-2.
12. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay É. 2011. Scikit-learn: machine learning in Python. J Mach Learn Res 12:2825–2830.
13. McKinney W. 2010. Data structures for statistical computing in Python, p 51–56. *In* Proceedings of the 9th Python in Science Conference. SciPy Organizers, Austin, TX. https://doi.org/10.25080/Majora-92bf1922-00a.
14. Hunter JD. 2007. Matplotlib: a 2D graphics environment. Comput Sci Eng 9:90–95. https://doi.org/10.1109/MCSE.2007.55.