Title: Future-proofing code: Principles of coding for plant ecophysiology with {photosynthesis} as a case study

Authors: Joseph R. Stinziano^{1,*}, Cassaundra Roback¹, Demi Gamble², Bridget K. Murphy^{3,4}, Patrick J. Hudson¹, Christopher D. Muir⁵

Affiliations:

¹Department of Biology, University of New Mexico, Albuquerque, NM, USA 87131-0001

²CSIRO Agriculture and Food, Myall Vale, NSW, Australia

³Department of Biology, University of Toronto Mississauga, Mississauga, ON, Canada

⁴Graduate Program in Cell and Systems Biology, University of Toronto, Toronto, ON, Canada

⁵School of Life Sciences, University of Hawai'i at Manoa, Honolulu, HI, USA

***Corresponding Author:** email: josephstinziano@gmail.com; phone: +1 (226) 678-1670

ORCiDs:

Joseph R. Stinziano: 0000-0002-7628-4201

Cassaundra Roback: none

Demi Gamble: 0000-0001-5951-4119

Bridget K. Murphy: 0000-0002-3574-6764

Patrick J. Hudson: 0000-0001-7759-2321

Chris D. Muir: 0000-0003-2555-3878

Word count: 3,999

Figures: 6

Tables: 1

Supporting Information: 3

Article Type: Viewpoint

26 **Summary**

- 27 • Plant physiological ecology is founded on a rich body of physical and chemical theory, but
28 it is challenging to connect theory with data in unambiguous, analytically rigorous, and
29 reproducible ways. Custom scripts written in computer programming languages (coding)
30 enable plant ecophysiolgists to model plant processes and fit models to data
31 reproducibly using advanced statistical techniques. Since most ecophysiolgists lack
32 formal programming education, we have yet to adopt a unified set of coding principles
33 and standards that could make coding easier to learn, use, and modify.
- 34 • We outline principles and standards for coding in plant ecophysiology to develop: 1)
35 standardized nomenclature, 2) consistency in style, 3) increased modularity/extensibility
36 for easier editing and understanding; 4) code scalability for application to large datasets,
37 5) documented contingencies for code maintenance; 6) documentation to facilitate user
38 understanding; and 7) extensive tutorials for biologists new to coding to rapidly become
39 proficient with software.
- 40 • We illustrate these principles using a new R package, {photosynthesis}, designed to
41 provide a set of analytical tools for plant ecophysiology.
- 42 • Our goal with these principles is to future-proof coding efforts to ensure new advances
43 and analytical tools can be rapidly incorporated into the field, while ensuring software
44 maintenance across scientific generations.

45 *Keywords: Gas exchange, hydraulics, photosynthesis, modeling, stomatal conductance,*
46 *curve fitting, software, R*

47

Background

Computer coding is becoming an increasingly important skill in biological research (Sayres *et al.*, 2018), especially within plant ecophysiology. However, formal computer science training is rarely part of formal training in biology programs (Carey & Papin, 2018). Instead, many biologists receive informal training or are self-taught (Baker, 2017). While formal computer science training may teach the capacity to write fast and efficient software, self-taught biologists may code in an entirely contingent manner, i.e. using techniques they come across first and understand best rather than the fastest or most efficient method (Baker, 2017). This disconnect in coding skill and training background can make it difficult for biologists to modify efficient programs (or even code written by other biologists) to account for a new understanding of biological processes. In other words, sophisticated code (by trained programmers) is efficient, but difficult to modify by biologists for new uses.

So why code at all? Coding allows for consistent, reproducible, and transparent and scalable analyses of scientific data, while at the same time minimizing human work-hours compared to using pre-packaged software. For example, Sharkey *et al.* (2007) have an Excel spreadsheet-based method for fitting photosynthetic CO₂ response (*A-C_i*) curves (also see Bellasio *et al.*, 2016) - this process can take several minutes per curve and involves a substantial amount of subjective decision-making (e.g. 'eye-balling' where transitions between CO₂- and RuBP-limited photosynthesis occur). Likewise, analysis of pressure-volume curves for hydraulic parameters is usually done via an Excel spreadsheet-based method (Sack *et al.*, 2003), which is also time-consuming, requires subjective decisions, and spreadsheets are usually not published with manuscripts, obscuring methodology. The total workload is time per spreadsheet multiplied by the number of curves, which can be inefficient in large studies. Cryptic changes in the spreadsheets can occur without a record of the change, potentially leading to compounding errors. Furthermore, spreadsheet tools often break, requiring a fresh, unaltered spreadsheet to be used for each CO₂ response curve. Another option, provided by Gu *et al.* (2010) (leafweb.org) provides an online service that analyses *A-C_i* curves, however in this case, the analysis is a black-box and could be misused by users lacking an understanding of the fitting process, and the data are stored on a government server which may cause some users discomfort. Meanwhile, Duursma (2015) developed an R package, {plantecophys}, that can obtain similar outputs to the Sharkey *et al.* (2007) fitting tools in seconds, with far fewer subjective decisions that can easily be outlined in the code used in

the fitting process, while providing a similar, but transparent approach as in Gu et al. (2010). Even with the {plantecophys} package, methods are not fully transparent or reproducible unless authors publish their code, which is still rare in plant ecophysiology (but see Kumarathunge et al., 2019 for an example of published code). Coding may also streamline integration between theory and data analysis, especially for complex mathematical formulations that require computationally intensive numerical methods, a common situation in plant ecophysiology. Ideally, we would like a workflow in which we state our assumptions mathematically, derive empirical predictions, and test those predictions or estimate parameters with data. This workflow is relatively accessible because open-source, research-grade computer algebra systems like SymPy (Meuer *et al.*, 2017) and numerical solvers are part of or can be readily integrated with programming languages that are widely used for data manipulation and analysis, such as R (R Core Team, 2020), Python (Python Software Foundation), or Julia (Bezanson *et al.* 2017).

Although coding can speed up large analyses, reduce errors, make analyses reproducible, and integrate theory with data, robust code that can be understood and reused by other scientists is not easy. First, one must learn one or more programming languages (e.g. R, Python, Matlab, Julia), which can involve shallow learning curves. Second, even though coding one's own analysis can make it easier to catch errors associated with inappropriate use of black-box statistical software using proprietary programs, one must still understand the assumptions and limitations of statistical techniques and conceptual tools. Finally, code can be as unique as someone's handwriting, which can make it difficult even for an experienced programmer to make sense of a "transparent" analysis unless there is sufficient annotation within the code.

In this perspective, we propose seven principles of coding tailored to the specific needs of the plant ecophysiology research community. Other scientific fields often emphasize speed. However, given the typical scale of ecophysiological datasets (~MB, i.e. small-batch, artisanal datasets), the computer power of modern laptops (~GB of RAM, ~GHz of processing power), computational speed is usually not a major limitation. Instead, ecophysiolgists often need to estimate parameters derived from complex biophysical/chemical models. In this domain, code flexibility and modularity are more important than speed. Furthermore, flexibility and modularity in code would enhance the sustainability of software after publication, which can

be an issue (Prlić & Procter, 2012). Here we demonstrate coding standards and principles designed for plant ecophysiology using a new R package called {photosynthesis} designed for the plant ecophysiology community. This perspective is intended to provide guidance for plant ecophysiolgists who are thinking about or starting to code their workflows, especially using R. We recognize that many scientists in this field are adept coders who have already honed their practices through experience. We hope this piece spurs experienced coders to share “best practices” to less-experienced peers and expand the principles below to other languages besides R.

Description

Principles of Coding

The overarching concept we propose is “future-proofing code”. Obviously not every possible discovery and need within a scientific field can be predicted, but the code can be written to allow easy modification and accommodation of the source code as the science progresses. Meta-programming in R provides a powerful tool for writing meta-functions (e.g. functions with functions as arguments, generalized arguments, etc.) that can easily process newly written code into a standardized output without the need for ever modifying the meta-function itself (Wickham, 2019). Such an approach helps to write modular code that is easy to modify and understand, while minimizing interdependencies between functions. For example, fitting a particular model across many groups could be done with a dedicated function (e.g. `fitacis` from {plantecophys}), however this would require a dedicated, multi-group fitting function for each type of response or model. In contrast, a meta-function could be used that simply iterates any input model across all groups within a dataset, producing an output that is standardized across models.

Freely available resources already exist for good coding practices in R packages and can be applied to R scripts as well, primarily from the efforts of Hadley Wickham (Wickham, 2014, 2015, 2016, 2017, 2019; Wickham & Golemund, 2016). As well, guides to best practices for scientific computing exist (see Wilson et al. 2014 for a list of best practices). Here we propose principles of coding for plant ecophysiology that, if implemented, would circumvent some of the common coding issues encountered when modifying the code of others, reduce the learning difficulty for naïve coders, and make software maintenance much easier:

1. Standardized nomenclature for variables and functions

- 142 2. Consistent style
- 143 3. Modularity and extensibility
- 144 4. Scalability
- 145 5. Documented contingencies
- 146 6. Documentation
- 147 7. Extensive tutorials

148 *Principle 1: standardized nomenclature*

149 Names vary wildly between functions with published code and data (for example “gmeso” is
150 used for mesophyll conductance in {plantecophys}, while most data and literature
151 representations use “ g_m ”), and even amongst instruments within the same company (e.g. for
152 net CO₂ assimilation, “A” is used in the Li-Cor 6800 and “PHOTO” is used in the Li-Cor 6400).
153 By standardizing the construction of variable and function names within the field, inter-
154 individual code readability would be greatly enhanced and the burden for learning how to use
155 new packages and functions or testing published code would be reduced. For example, g is
156 always in reference to conductance, where a subscript term would then describe the physical
157 pathway (e.g. s for stomata, c for cuticle, or m for mesophyll) as well as the gas (e.g. c for
158 CO₂, w for water vapor). For example, g_{sw} would mean stomatal conductance to water vapor.
159 Standardizing nomenclature across both mathematical models and data files can also
160 streamline theory-data integration, but this also requires standard translation between
161 mathematical and computer notation, which is beyond our scope here.

162 For example, in {photosynthesis}, every function is named in a descriptive manner: e.g.
163 `fit_t_response` fits specified temperature responses model to data, while `fit_gs_model`
164 fits specified models of stomatal conductance. Variable names are also standardized: e.g.
165 “T_leaf” always means leaf temperature in degrees Kelvin (K), “A_net” always means net CO₂
166 assimilation in $\mu\text{mol m}^{-2} \text{s}^{-1}$. In this regard, standard units should also be imposed in the
167 analysis (e.g. in R via the {units} package (Pebesma et al., 2016)), to remove any ambiguities
168 when interpreting the output. To allow for differences in variable names from the raw data
169 (e.g. from using different machines), the “varnames” list is used to translate input names (note
170 that this convention is adopted from {plantecophys} (Duursma, 2015)). We propose adopting
171 Wickham’s (Wickham, 2019) style in that functions that *do* something have a verb name, e.g.

172 `fit_aci_response`, while functions that act as objects within other functions (e.g. stomatal
173 conductance models) should have a noun name, e.g. `gs_model`.

174 *Principle 2: consistent style*

175 Consistent coding style makes reading code easier - certain conventions, e.g. commenting
176 what the *next* line of code does, can make it easier to understand code documentation. Our
177 preference is for the 'tidy style', which applies to both data and code structure, and much else
178 (see the *The tidyverse Style Guide*: <https://style.tidyverse.org/>). For data, tidy style advocates
179 that each column is a variable, and each row is an observation, since R is particularly suited
180 for this style of data structure. Popular R packages like {dplyr} (Wickham *et al.*, 2020) and
181 {tidyr} (Wickham and Henry, 2020) facilitate tidy data and many other packages, like
182 {photosynthesis}, use them for consistent style. For code, computers do not care about style,
183 as long as it is correctly formatted, but for humans reading code, adherence to well-designed
184 style can be helpful, especially for beginners trying to learn from others. A benefit of tidy style
185 in particular is that R packages {styler} (Müller and Walthert, 2020), {lintr} (Hester *et al.* 2020),
186 and {formatR} (Xie, 2019) can automate conformity to style. Ideally, a consistent style would
187 be adopted across the field, however this may be too rigid. Style can be highly personal, and
188 many experienced coders likely have developed their own style, formal or informal, that works
189 for them. Our proposal is geared for beginning coders who are looking for guidance on an
190 established and easy-to-implement style. At the very least, a consistent style *within* a project
191 will make it easier to read, understand, and modify the code.

192 *Principle 3: modularity and extensibility*

193 Arguably, code written for plant ecophysiologists, whether formally trained in coding or not,
194 should be written in a modular manner, much like Lego bricks, where one component (e.g.
195 Arrhenius function) can be easily swapped with another (e.g. peaked Arrhenius function), or
196 extended (e.g. hypothetical mechanistic temperature response model). Note that this will
197 increase apparent complexity of software packages by creating more function files. However,
198 it will make adding, subtracting, or modifying code modules easier for researchers who need
199 to make on-the-fly changes to code as new biological processes are discovered or old ones
200 re-evaluated. To achieve modularity in the structure of photosynthesis, we used principles of
201 meta-programming to develop a set of key functions for processing data and running quality
202 control checks: `fit_many`, `analyze_sensitivity`, `compile_data`, and `print_graphs`.

Both `fit_many` and `analyze_sensitivity` can be run with any function within and outside of `{photosynthesis}` to run multiple curve fits or sensitivity analyses on assumed input parameters. Meanwhile, `compile_data` is used for processing the list outputs from `fit_many` into a form usable for further analyses and export from R, and `print_graphs` is used to export all graphs from a list as either .jpeg or compiled as a .pdf.

For curve fitting functions with multiple models (e.g. temperature responses, g_s models), we use a basic function (e.g. `fit_t_response`), which contains fitting procedures for each of the seven temperature response models in the package. Meanwhile, a `t_functions` file contains all the temperature response functions. To extend the capabilities and add in a new temperature response model, we simply need to add the new model to `t_functions`, and the fitting procedure to `fit_t_response`. This principle of function building increases the extensibility of the code, while consistent style and standardized nomenclature provide the rules for writing the extended components.

Modularity also applies to modelling. The `{photosynthesis}` functions `photo` and `photosynthesis` model C_3 photosynthesis using the Faquhar-von Caemmerer-Berry biochemical model (Farquhar *et al.* 1980). To account for temperature dependence, a user can specify leaf temperature, or they can provide additional inputs (e.g. air temperature, leaf size, wind speed, etc.) to model leaf temperature using energy balance in the R package `{tealeaves}` (Muir 2019). Both `{photosynthesis}` and `{tealeaves}` packages are modular in that they can work independently or be readily integrated (**Methods S1**). Ideally, future modeling packages would add modules to model environmental and plant parameters either on their own or integrated with these tools.

Principle 4: scalability

A major advantage in using code to analyze data is the ability to scale up an analysis to reduce time spent on repetitive tasks, allowing the same model to be fit across groups within a dataset using a consistent method. For this, our `fit_many` function and the principles of meta-programming are how we achieve scalability within the package. Rather than writing functions for each type of model or curve, we have a single multiple fitting function, sensitivity analysis function, and printing function. R even has generic functions for scaling such as `apply` (base R language) and `map` (`{purrr}` package [Henry & Wickham, 2020]) which can be

233 easily parallelized for speed (e.g. {parallel} and {furrr} [Vaughn & Dancho, 2018] packages).
 234 This makes it easy to scale a new function within the software to a large dataset.

235 *Principle 5: documented contingencies*

236 By documenting which functions are dependent on one another, it becomes easier to
 237 troubleshoot issues when modifying code and to pre-empt issues when adding or replacing a
 238 component. For example, `fit_aq_response` depends on `aq_response` - if we want to
 239 change from the non-rectangular hyperbola model to a rectangular hyperbolic model, then
 240 `fit_aq_response` needs to be modified in addition to `aq_response`. To document
 241 contingencies, we created a function, `check_dependencies`, which uses {pkgnet} (Burns et
 242 al., 2020) to generate an html report that automatically documents R package inter-
 243 dependencies and function inter-dependencies. This is particularly useful when adding,
 244 subtracting, or modifying functions in the package, as it allows planning to minimize issues
 245 that could break code.

246 *Principle 6: documentation*

247 Code annotations allow a new user to readily understand what a line of code is doing, how it
 248 is doing it, and why the code is written in a particular way. By providing exhaustive line-by-line
 249 annotation of a function, a new user can more rapidly understand the blueprint of the function.
 250 This is especially useful for R scripts and code hosted on GitHub (unfortunately, comments
 251 are erased from code upon submission to CRAN). For example, in `fit_t_response`, we
 252 outline the need for running looped iterations for the starting values of nonlinear least squares
 253 curve-fitting (Fig. 1). In the case of R packages hosted on CRAN, R documentation files
 254 provide information on how to use a function, though as a terser set of instructions as per
 255 CRAN policies (<https://cran.r-project.org/doc/manuals/r-devel/R-exts.html>).

256 Enough metadata and commenting should be provided for a new user to understand how to
 257 use the written code (which can be an issue that affects widespread use of a program,
 258 Mangul *et al.* 2019).

259 *Principle 7: extensive tutorials*

260 As with any tool, software will only be used if potential users can understand how it works.
 261 Extensive tutorials, while providing function-by-function examples of how to use the software,
 262 should also incorporate basic data-wrangling examples and explanations of why a given

approach to data analysis is used in the field. The benefits of this approach include: making the code easier to adopt into your own analysis, making it easier for new coders to learn enough of the language to use the package effectively, and help trainees learn the appropriate theory behind the measurements and analytical approach. The net effect should be to increase the inclusivity of the field by reducing barriers to success since not all individuals will have equal access to workshops or experienced colleagues.

{photosynthesis}: the future-proofed R package for plant ecophysiology

We built a package containing analytical tools for plant ecophysiology (Stinziano *et al.*, 2020), embedding our coding principles into the package itself. The R package contains functions for fitting photosynthetic CO₂ (Farquhar *et al.*, 1980; von Caemmerer, 2000; Gu *et al.*, 2010; Duursma, 2015) and light response curves (Marshall & Biscoe, 1980), temperature responses of biological processes (Arrhenius, 1915; Medlyn *et al.*, 2002; Kruse *et al.*, 2006; Heskell *et al.*, 2016; Liang *et al.*, 2018), light respiration (Kok, 1956; Laisk, 1977; Yin *et al.*, 2009, 2011; Walker & Ort, 2016), mesophyll conductance (Harley *et al.*, 1992), stomatal conductance models (Ball *et al.*, 1987; Leuning, 1995; Medlyn *et al.*, 2011), pressure-volume curves (Tyree & Hammel, 1972; Koide *et al.*, 2000; Sack *et al.*, 2003), hydraulic vulnerability curves (Pammenter & van der Willigen, 1998; Ogle *et al.*, 2009), and sensitivity analyses (Table 1; Table S2). It also contains functions for modeling C₃ photosynthesis using the Farquhar-von Caemmerer-Berry biochemical model (Farquhar *et al.*, 1980). The default kinetic parameters for gas exchange fitting procedures are taken from *Nicotiana tabacum* (Bernacchi *et al.*, 2001, 2002). A comprehensive illustration of how to use the package can be found in the vignette of the package (**Notes S1**, “photosynthesis-curve-fitting-sensitivity-analyses.rmd”).

The package is specifically designed to accommodate new analytical tools and discoveries and be easily maintained by new users. Nonlinear curve fitting procedures use the `nlsLM` function from `{minpack.lm}` (Elzhov *et al.*, 2016), which provides a more robust fitting procedure for non-linear functions than the base R `nls` function. Graphical outputs are provided using `{ggplot2}` (Wickham, 2016). Meta-functions were constructed with the tools provided for generalizing functions and arguments in `{rlang}` (Henry & Wickham, 2019).

The principles of modularity and metaprogramming have been used to substantially reduce code interdependencies within the software. For example, the `fitaci` function from `{plantecophys}` has over 30 function dependencies (Fig. 2a). By applying our principles, we

were able to reduce this to just 4 function dependencies (Fig. 2b), by reengineering the fitting procedure and eliminating redundant functions and code.

Example Dataset

To demonstrate the fitting functions of the package, we use a combination of data collected for the package and previously published data. A CO₂ by light response curve and CO₂ by temperature response curve were collected in sunflower (*Helianthus annuum*) grown in a rooftop greenhouse at the University of New Mexico (35.0843°N, 106.6198°W, 1587 m a.s.l., 18.3 to 21.1/15.6 to 21.1 °C day/night temperature with daily irradiances of 600 to 1,200 μmol m⁻² s⁻¹). CO₂ response curves were measured at irradiances of 1,500, 150, 375, 125, 100, 75, 50, and 25 μmol m⁻² s⁻¹ at a T_{leaf} of 25 °C. CO₂ response curves were also measured at T_{leaf} of 17.5, 20, 22.5, 25, 27.5, 30, 32.5, 35, 37.5, and 40 °C at an irradiance of 1,500 μmol m⁻² s⁻¹. Data to demonstrate hydraulic vulnerability curve fitting methods were drawn from (Hudson *et al.* 2018), while data for leaf pressure/volume analysis come from an as-yet unpublished dataset collected at the University of New Mexico. Below we illustrate some of the functionality of the package. We refer potential users to the package vignette for more worked examples (**Notes S1**, “photosynthesis-curve-fitting-sensitivity-analyses.rmd”).

Photosynthetic light response curve fitting

The `fit_aq_response` function returns a list containing the fitted light response model, model parameters, and a graph showing the model fit to the data (Fig. 3a).

Photosynthetic CO₂ response curve-fitting

The `fit_aci_response` function returns a list containing the fitted parameters, a data frame with the modelled data output, and a graph showing the model fit to the data (Fig. 3b).

Photosynthetic temperature response curve fitting

A series of temperature response functions can be fit using the package, with the outputs including the fitted model, model parameters, and a graph (Fig. 3c).

Fitting g_m using the variable J method

The `fit_g_mc_variableJ` function using the method of Harley *et al.* (1992) using chlorophyll fluorescence and gas exchange data to estimate g_{mc} . Both g_{mc} and $\delta C_o/\delta A$ are calculated, where $\delta C_o/\delta A$ between 10 and 50 are deemed to be “reliable” (Harley *et al.*,

1992), and an average g_{mc} value is estimated based on the reliable values. This makes it relatively easy to assess the reliability of g_{mc} estimates (Fig. 4).

Hydraulic vulnerability curve fitting

The `fit_hydra_vuln_curve` fits hydraulic vulnerability data using both a sigmoidal and Weibull function. Outputs include model fits, parameters and a graph (Fig. 5a).

Pressure-volume curves

The `fit_pv_curve` fits pressure-volume curves, returning parameters such as relative water content and water potential at turgor loss points, relative capacitance at full turgor, and others. Outputs include parameters and graphs (Fig. 5bc).

Sensitivity analyses

Both `analyze_sensitivity` and `compute_sensitivity` are used in combination for sensitivity analyses. `analyze_sensitivity` allows up to two assumed parameters to be varied in a fitting function, while `compute_sensitivity` runs two types of local sensitivity calculations based on a user-defined reference value: parameter effect (Bauerle et al., 2014) and control coefficient (Capaldo & Pandis, 1997). We can look at the impact of varying g_m and Γ^* at 25 °C on fitted V_{cmax} (Fig. 6). We can see that g_m and Γ^* at 25 °C have an orthogonal impact on V_{cmax} , with Γ^* having a stronger control than g_m on V_{cmax} .

Moving forward - standardized practices and code editors

It is not easy to rewrite software, and we are not arguing as such. Rather, going forward as a community, we argue that we should adopt a set of coding principles and guidelines to create code as flexible as the biology we study. We present the R package, {photosynthesis}, as an example of these principles and guidelines. The consequences of this are not to be understated: it will be easier for new trainees and beginner coders to learn, understand, and write code for the community; and it will be easier to tailor existing code to our projects.

The drawback is that code may run more slowly: however, in cases where this actually matters (e.g. eddy flux covariance, bioinformatics, big data) computational speed may take precedence over flexibility. In ecophysiology, our datasets are often small enough that even complex analyses may only take 1 hr on one computer core of a multi-core system – as a community we can afford slower-running code for greater flexibility and ease-of-understanding, especially as this could save days or weeks of coding to write a desired

353 analysis. Our code should be as flexible as, and easier to understand, than the biology it
354 describes.

355 However, providing code according to these standards is not sufficient - we also need code-
356 competent editorial staff for journals who can properly review and test submitted code to
357 ensure that it runs as intended. In some cases, code for a published dataset does not work
358 even after comprehensive modification (Stinziano, pers. comm.). Standardized coding
359 practices will help to reduce the burden on code editors by making it easier to read and
360 understand code submissions.

361 **Acknowledgements**

362 JRS would like to thank Dr. David T. Hanson for use of gas exchange equipment and
363 greenhouse space, and Dayan Fuentes for assisting in collecting gas exchange data. CDM is
364 supported by the National Science Foundation (1929167).

365 **Conflict of Interests**

366 The authors declare no conflicts of interest.

367 **Author Contributions**

368 All authors contributed to study design and testing of the software tools. JRS, PJH, and CDM
369 wrote the software. JRS and CDM wrote the manuscript with input from all coauthors. CR,
370 DG, and BKM contributed equally to the manuscript.

371 **Data Availability Statement**

372 All data and code used in the manuscript are available at
373 <https://github.com/cdmuir/photosynthesis>.

374 **Supporting Information**

375 **Methods S1 – Description of variables used in {photosynthesis}**

376 **Tables S2 – Table of other utility functions in {photosynthesis}.**

377 **Notes S1 – The {photosynthesis} R package tar.gz file.**

378

379 References

- 380 **Arrhenius S. 1915.** Quantitative laws in biological chemistry. Bell.
- 381 **Baker M. 2017.** Scientific computing: code alert. *Nature* **541**:563-565.
- 382 **Ball JT, Woodrow IE, Berry JA. 1987.** A model predicting stomatal conductance and its
383 contribution to the control of photosynthesis under different environmental conditions, in
384 Progress in Photosynthesis Research, Proceedings of the VII International Congress on
385 Photosynthesis, vol. 4, edited by I. **Biggins**, pp. 221–224, Martinus Nijhoff, Dordrecht,
386 Netherlands.
- 387 **Bauerle WL, Daniels AB, Barnard DM. 2014.** Carbon and water flux responses to
388 physiology by environment interactions: a sensitivity analysis of variation in climate on
389 photosynthetic and stomatal parameters. *Climate Dynamics* **42**:2539-2554.
- 390 **Bellasio C, Beerling J, Griffiths H. 2016.** An Excel tool for deriving key photosynthetic
391 parameters from combined gas exchange and chlorophyll fluorescence: theory and practice.
392 *Plant, Cell & Environment* **39**:1180-1197.
- 393 **Bernacchi CJ, Singaas EL, Pimentel C, Portis AR, Long SP. 2001.** Improved temperature
394 response functions for models of rubisco-limited photosynthesis. *Plant Cell & Environment*
395 **24**:253-259.
- 396 **Bernacchi CJ, Portis AR, Nakano H, von Caemmerer S, Long SP. 2002.** Temperature
397 response of mesophyll conductance. Implications for the determination of rubisco enzyme
398 kinetics and for limitations to photosynthesis in vivo. *Plant Physiology* **130**:1992-1998.
- 399 **Burns B, Lamb J, Qi J. 2020.** pkgnet: Get Network Representation of an R Package. R
400 package version 0.4.1. <https://CRAN.R-project.org/package=pkgnet>
- 401 **Capaldo KP, Pandis SN. 1997.** Dimethylsulfide chemistry in the remote marine atmosphere:
402 evaluation and sensitivity analysis of available mechanisms. *Journal of Geophysical Research*
403 **102**:23251-23267
- 404 **Carey MA, Papin JA. 2018.** Ten simple rules for biologists learning to program. *PLoS*
405 *Computational Biology* **14**:e1005871.

406 **Duursma RA. 2015.** Plantecophys – an R package for analyzing and modeling leaf gas
407 exchange data. *PLoS ONE* **10**:e0143346.

408 **Elzhov TV, Mull KM, Spiess A-N, Bolker B. 2016.** minpack.lm: R interface to the
409 Levenberg-Marquardt nonlinear least-squares algorithm found in MINPACK, plus support for
410 bounds. R package version 1.2-1. <https://CRAN.R-project.org/package=minpack.lm>

411 **Farquhar GD, von Caemmerer S, Berry JA. 1980.** A biochemical model of photosynthetic
412 CO₂ assimilation in leaves of C₃ species. *Planta* **149**: 78-90.

413 **Gu L, Pallardy SG, Tu K, Law BE, Wullschlegel SD. 2010.** Reliable estimation of
414 biochemical parameters from C₃ leaf photosynthesis-intercellular carbon dioxide response
415 curves. *Plant Cell & Environment* **33**:1852-1874.

416 **Harley PC, Loreto F, Di Marco G, Sharkey TD. 1992.** Theoretical considerations when
417 estimating mesophyll conductance to CO₂ flux by analysis of the response of photosynthesis
418 to CO₂. *Plant Physiology* **98**:1429 - 1436.

419 **Henry L, Wickham H. 2019.** Rlang: functions for base types and core R and 'Tidyverse'
420 features. R package version 0.4.2. <https://CRAN.R-project.org/package=rlang>.

421 **Henry L, Wickham H. 2020.** purrr: Functional Programming Tools. R package version 0.3.4.
422 <https://CRAN.R-project.org/package=purrr>

423 **Heskel MA, O'Sullivan OS, Reich PB, Tjoelker MG, Weerasinghe LK, Penillard A,**
424 **Egerton JJG, Creek D, Bloomfield KJ, Xiang J, Sinca F, Stangl ZR, la Torre AM, Griffin**
425 **KL, Huntingford C, Hurry V, Meir P, Turnbull MH, Atkin OK. 2016.** Convergence in the
426 temperature response of leaf respiration across biomes and plant functional types.
427 *Proceedings of the National Academy of Sciences USA* **113**:3832-3837

428 **Hester J, Angly F, Hyde R. 2020.** lintr: A 'Linter' for R Code. R package version 2.0.1.
429 <https://CRAN.R-project.org/package=lintr>

430 **Hudson PJ, Limousin JM, Krofcheck DJ, Boutz AL, Pangle RE, Gehres N, McDowell**
431 **NG, Pockman WT. 2018.** Impacts of long-term precipitation manipulation on hydraulic

432 architecture and xylem anatomy of piñon and juniper in Southwest USA. *Plant Cell &*
433 *Environment* **41**: 421-435.

434 **Koide RT, Robichaux RH, Morse SR, Smith CM. 2000.** Plant water status, hydraulic
435 resistance and capacitance. In: Plant Physiological Ecology: Field Methods and
436 Instrumentation (eds RW Pearcy, JR Ehleringer, HA Mooney, PW Rundel), pp. 161-183.
437 Kluwer, Dordrecht, the Netherlands

438 **Kok B. 1956.** On the inhibition of photosynthesis by intense light. *Biochimica et Biophysica*
439 *Acta* **21**: 234–244

440 **Laisk A. 1977.** Kinetics of photosynthesis and photorespiration in C3 plants. Nauka, Moscow.

441 **Leuning R. 1995.** A critical appraisal of a coupled stomatal- photosynthesis model for C3
442 plants. *Plant Cell & Environment* **18**:339-357

443 **Liang LL, Arcus VL, Heskell MA, O'Sullivan OS, Weerasinghe LK, Creek D, Egerton JGG,**
444 **Tjoelker MG, Atkin OK, Schipper LA. 2018.** Macromolecular rate theory (MMRT) provides a
445 thermodynamics rationale to underpin the convergent temperature response in plant leaf
446 respiration. *Global Change Biology* **24**:1538-1547

447 **Mangul S, Martin LS, Eskin E, Blekhman R. 2019.** Improving the usability and archival
448 stability of bioinformatics software. *Genome Biology* **20**:47.

449 **Marshall B, Biscoe P. 1980.** A model for C3 leaves describing the dependence of net
450 photosynthesis on irradiance. *Journal of Experimental Botany* **31**:29-39

451 **Medlyn BE, Dreyer E, Ellsworth D, Forstreuter M, Harley PC, Kirschbaum MUF, Le Roux**
452 **X, Montpied P, Strassmeyer J, Walcroft A, Wang K, Loutstau D. 2002.** Temperature
453 response of parameters of a biochemically based model of photosynthesis. II. A review of
454 experimental data. *Plant Cell & Environment* **25**:1167-1179

455 **Medlyn BE, Duursma RA, Eamus D, Ellsworth DS, Prentice IC, Barton CVM, Crous KY,**
456 **Angelis PD, Freeman M, Wingate L. 2011.** Reconciling the optimal and empirical
457 approaches to modelling stomatal conductance. *Global Change Biology* **17**:2134-2144

458 **Muir CD. 2019.** tealeaves: an R package for modelling leaf temperature using energy
459 budgets. *AoB PLANTS* **11**:plz054

460 **Müller K, Walthert L. 2020.** styler: Non-Invasive Pretty Printing of R Code. R package
461 version 1.3.2. <https://CRAN.R-project.org/package=styler>

462 **Ogle K, Barber JJ, Willson C, Thompson B. 2009.** Hierarchical statistical modeling of xylem
463 vulnerability to cavitation. *New Phytologist* **182**:541-554

464 **Pammenter NW, Van der Willigen CV. 1998.** A mathematical and statistical analysis of the
465 curves illustrating vulnerability of xylem to cavitation. *Tree Physiology* **18**:589-593

466 **Pebesma R, Mailund T, Hiebert J. 2016.** Measurement units in R. *R Journal* **8**:486-494.

467 **Prlić A, Proctor PB. 2012.** Ten simple rules for the open development of scientific software.
468 *PLoS Computational Biology* **8**:e1002802.

469 **Sack L, Cowan PD, Jaikumar N, Holbrook NM. 2003.** The 'hydrology' of leaves: co-
470 ordination of structure and function in temperate woody species. *Plant, Cell & Environment*
471 **26**: 1343-1356.

472 **Sayres MAW, Hauser C, Sierk M, Robic S, Rosenwald AG, Smith TM, Triplett EW, et al.**
473 **2018.** Bioinformatics core competencies for undergraduate life sciences education. *PLoS*
474 *ONE* **13**:e0196878.

475 **Tyree MT, Hammel HT. 1972.** Measurement of turgor pressure and water relations of plants
476 by pressure bomb technique. *Journal of Experimental Botany* **23**:267

477 **Vaughan D, Dancho M. 2018.** furr: Apply Mapping Functions in Parallel using Futures. R
478 package version 0.1.0. <https://CRAN.R-project.org/package=furr>

479 **von Caemmerer S. 2000.** Biochemical models of leaf photosynthesis. CSIRO Publishing,
480 Collingwood.

481 **Walker BJ, Ort DR. 2015.** Improved method for measuring the apparent CO₂
482 photocompensation point resolves the impact of multiple internal conductances to CO₂ to net
483 gas exchange. *Plant Cell & Environment* **38**:2462- 2474

- 484 **Wickham H. 2019.** Advanced R, 2nd Ed. Chapman & Hall.
- 485 **Wickham H. 2017.** Tidyverse Tidyweb. <http://tidyverse.org/>.
- 486 **Wickham H. 2016.** Tidyverse: easily install and load ‘tidyverse’ packages. [https://CRAN.R-](https://CRAN.R-project.org/package=tidyverse)
- 487 [project.org/package=tidyverse](https://CRAN.R-project.org/package=tidyverse).
- 488 **Wickham H. 2016.** ggplot2: elegant graphics for data analysis. Springer-Verlag New York.
- 489 **Wickham H. 2015.** R Packages: Organize, Test, Document, and Share Your Code. O’Reilly
- 490 Media Inc. Sebastopol, CA, USA.
- 491 **Wickham H. 2014.** Tidy data. *Journal of Statistical Software* **59**:1-23.
- 492 **Wickham H, Golemund G. 2016.** R for Data Science. O’Reilly Media Inc. Sebastopol, CA,
- 493 USA.
- 494 **Wickham H, Henry L. 2020.** tidyr: Tidy Messy Data. R package version 1.1.0.
- 495 <https://CRAN.R-project.org/package=tidyr>
- 496 **Wickham H, François R, Henry L, Müller K. 2020.** dplyr: A Grammar of Data Manipulation.
- 497 R package version 1.0.0. <https://CRAN.R-project.org/package=dplyr>
- 498 **Wilson G, Aruliah DA, Brown CT, Hong NPC, Davis M, Guy RT, Haddock SHD, et al.**
- 499 **2014.** Best practices for scientific computing. *PLoS Biology* **12**:e1001745.
- 500 **Xie Y. 2019.** formatR: format R code automatically. R package version 1.7. [https://CRAN.R-](https://CRAN.R-project.org/package=formatR)
- 501 [project.org/package=formatR](https://CRAN.R-project.org/package=formatR).
- 502 **Yin X, Struik PC, Romero P, Harbinson J, Evers JB, van der Putten PEL, Vos J. 2009.**
- 503 Using combined measurements of gas exchange and chlorophyll fluorescence to estimate
- 504 parameters of a biochemical C3 photosynthesis model: a critical appraisal and a new
- 505 integrated approach applied to leaves in a wheat (*Triticum aestivum*) canopy. *Plant Cell &*
- 506 *Environment* **32**:448-464
- 507 **Yin X, Sun Z, Struik PC, Gu J. 2011.** Evaluating a new method to estimate the rate of leaf
- 508 respiration in the light by analysis of combined gas exchange and chlorophyll fluorescence
- 509 measurements. *Journal of Experimental Botany* **62**: 3489–3499

511 Tables

512 **Table 1. List of {photosynthesis} functions with applications and descriptions.**

Base Functions		
Applications	Function	Description
Gas Exchange	<code>fit_aci_response</code>	Fits $A-C_i$ curves, provides parameters/graphs
Gas Exchange	<code>fit_aq_response</code>	Fits $A-Q$ curves, provides parameters/graphs
Gas Exchange	<code>fit_g_mc_variableJ</code>	Fits g_{mc} , adds g_{mc} and $dCcdA$ to dataframe for reliability checking
Gas Exchange	<code>fit_gs_model</code>	Fits the Ball <i>et al.</i> 1987, Leuning 1995, and Medlyn <i>et al.</i> 2011 models of stomatal conductance, provides parameters/graphs
Hydraulics	<code>fit_hydra_vuln_curve</code>	Fits the sigmoidal and Weibull models to hydraulic vulnerability data, provides parameters/graphs
Hydraulics	<code>fit_PV_curve</code>	Fits pressure volume curves, provides parameters/graphs
Gas Exchange	<code>fit_r_light</code>	Fits r_{light} according to the Kok (1956) method, Yin method (Yin <i>et al.</i> 2009, 2011), or Walker & Ort (2015) method.
Gas Exchange, Biochemistry	<code>fit_t_response</code>	Fits an Arrhenius (Arrhenius 1915), Heskell (Heskell <i>et al.</i> 2016), Kruse (Kruse <i>et al.</i> 2006), Medlyn (Medlyn <i>et al.</i> 2002), MMRT (Hobbs <i>et al.</i> 2013), and quadratic temperature response models, provides parameters/graphs
Modeling	<code>photo</code>	Simulates C_3 photosynthesis over a parameter set
Modeling	<code>make_parameters</code>	A set of functions (e.g. <code>make_enviropar</code> , <code>make_leafpar</code>) that generates the required inputs for <code>photo</code>
Meta-functions & Utilities		
Application	Function	Description
Software modification	<code>check_dependencies</code>	Generates HTML with package and function dependencies
All components	<code>compile_data</code>	Compiles the output from the <code>fit_many</code> function
All components	<code>fit_many</code>	Fits a function many times through a grouping variable
All components	<code>print_graphs</code>	Prints graphs from a list of graphs
All components	<code>sensitivity_analysis</code>	Allows up to 2-factor sensitivity analysis of any function

513

514

515 Figures

```

#Basically, use Arrhenius curve to feed Ea into Topt function start
#Try approach where you start Hd from 1 to 1000 to ensure model fit
#select minimum residual
model <- nlsLM(data = data,
               Par ~ Par25 * t_response_arrhenius(Ea,
                                                    Tleaf = Tleaf),
               start = start,
               lower = c(0, 0),
               upper = c(1e10, 10 * max(data$Par)),
               control = nls.control(maxiter = 100)
)

#Create empty dataframe to fill with 1000 curve fits
model_fm <- as.data.frame(cbind(rep(0, 1000),
                                rep(0, 1000),
                                rep(0, 1000),
                                rep(0, 1000),
                                rep(0, 1000),
                                rep(varnames$Par[[1]], 1000)))

#Assign column names
colnames(model_fm) <- c("Ea", "Hd", "kopt", "Topt", "residual",
                       |"Parameter")

#Make sure variable classes are appropriate

```

517 **Figure 1. Example of coding annotations to explain the given analytical approach.**

518

Figure 3. Gas exchange curve fitting outputs. a) Output from `{fit_aq_response}` showing the data (black points), the model fit (blue line), and the standard error on the model fit (grey region). The light response at a $[\text{CO}_2]$ of $100 \mu\text{mol mol}^{-1}$ is shown. A_{net} : net CO₂ assimilation. b) Graph from `fit_aci_response` showing modelled A_{net} (A_{mod} , black line), CO₂-limited A_{net} (A_c , blue), RuBP regeneration-limited A_{net} (A_j , orange), triose phosphate utilization-limited A_{net} (A_p), and the data (white dots). A_{net} : net CO₂ assimilation; C_i : intercellular CO₂ concentration. c) Output from `fit_t_response` showing the Heskell temperature response of J_{max} . Data are black dots, model fit is the

blue line, and the grey shaded region is the standard error on the model fit. J_{max} : maximum rate of electron transport; T_{leaf} : leaf temperature.

Figure 4. Relationship between g_{mc} estimated through the variable J method and $\delta C_c / \delta A$ to test for reliability. The `fit_g_mc_variableJ` function was used on the CO_2 by light response data in sunflower. g_m : mesophyll conductance; C_c : chloroplastic CO_2 concentration; A : net CO_2 assimilation

Figure 5. a) Example output from `fit_hydra_vuln_curve` showing both model fits overlaid on the data (black dots). *PLC*: percent loss of conductivity; *Pe*: air entry point; *P50*: water potential at 50% *PLC*; *Pmax*: hydraulic failure threshold. b, c) Example output from `fit_pv_curve` showing the b) water mass graph and c) the pressure-volume curve. Grey lines are fit to the linear regions of the data. Ψ : water potential; RWC: relative water content.

Figure 6. Control coefficients of g_m and Γ^* at 25 °C calculated from analyze_sensitivity and compute_sensitivity.

553 Supplementary Information

554 Table S2. List of additional {photosynthesis} functions with applications and 555 descriptions.

Functions		
Applications	Function	Description
Gas Exchange	<code>aq_response</code>	Contains the light response model used by <code>fit_aq_response</code>
Modeling	<code>bake-par</code>	Checks that <code>bake</code> has worked
Modeling	<code>bake</code>	Temperature scales input parameters
Modeling	<code>conductance</code>	Provides calculations for leaf conductances to CO ₂
Modeling	<code>constants</code>	Checks formatting of physical constant inputs
Modeling	<code>enviro-par</code>	Checks formatting of environmental inputs
Modeling	<code>FvCB</code>	Contains the model of Farquhar et al. (1980) for modeling
Gas Exchange	<code>gs_models</code>	Contains all stomatal conductance models that can be fit with <code>fit_gs_model</code>
Gas Exchange	<code>j_calculations</code>	Contains the electron transport models for fitting $A-C_i$ curves
Modeling	<code>leaf-par</code>	Ensures proper formatting of leaf parameter inputs
Modeling	<code>parameter_names</code>	Gets a vector of parameter names
Modeling	<code>photosynthesis</code>	Contains several functions necessary for <code>photo</code>
Gas Exchange	<code>t_functions</code>	Contains all temperature response functions that can be fit with <code>fit_t_response</code>
Modeling	<code>utils</code>	Contains unit conversion functions for modeling

556