Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

SiTree: A framework to implement single-tree simulators

Clara Antón-Fernández [*], Rasmus Astrup

Division of Forest and Forest Resources, NIBIO (Norwegian Institute for Bioeconomy Research), Høgskoleveien 8, 1433 Ås, Norway

ARTICLE INFO

Article history:
Received 4 August 2021
Received in revised form 23 November 2021
Accepted 25 November 2021

Keywords:
Forest simulator
Forest growth modeling
Individual-tree
R package
Forest growth and yield

ABSTRACT

SiTree is a flexible, cross-platform, open-source framework for individual-tree simulators intended to facilitate accurate and flexible analyses of forest growth and yield, or more generally forest dynamics simulations. *SiTree* provides generic functionality to build customized individual-tree simulators using additional user-written code. In the forestry literature there are a wide variety of individual models that describe the different parts of forest growth and dynamics and new models are continuously developed and published. The aim of *SiTree* is to provide a broad community of R-users within forestry with an easily adaptable individual-tree simulator framework and an easily accessible tool for testing and combining new and existing models describing parts of forest growth dynamics.

Code metadata

Current code version	0.1-12
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-21-00145
Code Ocean compute capsule	https://codeocean.com/capsule/3789289/tree
Legal Code License	GPL-3
Code versioning system used	git
Software code languages, tools, and services used	R
Compilation requirements, operating environments & dependencies	R ($\geq$ 3.5), data.table, ggplot2
If available Link to developer documentation/manual	https://cran.r-project.org/web/packages/sitree/sitree.pdf
Support email for questions	caf@nibio.no

Software metadata

Current software version	0.1-12
Permanent link to executables of this version	https://github.com/cantonfe/SiTree
Legal Software License	GPL-3
Computing platforms/Operating Systems	Any operating system supporting R
Installation requirements	R ($\geq$ 3.5), data.table, ggplot2
If available, link to user manual - if formally published include a reference to the publication in the reference list	https://github.com/cantonfe/SiTree
Support email for questions	caf@nibio.no

1. Motivation and significance

Forest growth simulators (FGSs) are a fundamental tool to inform decisions in forest management at various spatial and temporal scales. FGSs have been used for more than 50 years to examine the effects of different treatment scenarios, and to help determine the best management solutions in practical forest planning.

A FGS, following the definition of Hasenauer [1], is the implementation of a forest growth model (FGM) into a software for prediction and scenario analysis. A FGM is the biometric and mathematical representation of growth processes [1]. A FGM should at least represent growth and mortality [2] through a

[*] Corresponding author.
E-mail address: caf@nibio.no (Clara Antón-Fernández).

https://doi.org/10.1016/j.softx.2021.100925

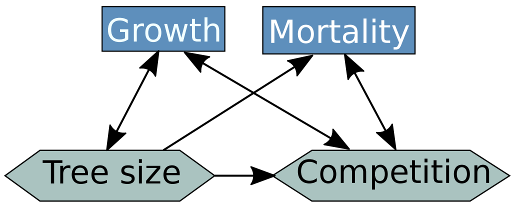

Fig. 1. Simplified description of the interactions between two of the components of a FGS. Arrows indicate the direction of the interaction. Forest growth sub-models are indicated in blue, and dependent variables in green. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

growth sub-model and a mortality sub-model. These sub-models estimate growth and mortality for a tree or stand for a certain period, for example, 5 years. Typically, this sub-models depend on variables such as tree size, or competition (for example, density). Tree size and competition variables are at the same time affecting and affected by the growth and mortality sub-models (Fig. 1). For example, growth affects tree size, which in turn affects both the competition exerted by the tree to the surrounding trees and the growth of the tree in the next period. These complex interactions between variables and sub-models makes it impossible to project variables independently from each other beyond the first period.

To estimate the state of a stand for longer periods the FGM is applied recurrently, that is, it is applied serially to calculate the temporal development of the stand. Consistently applying the FGM, while at the same time keeping track of all the variables during the full simulation in a robust way, can be an arduous and error-prone task. FGSs have been developed to help with this task.

FGMs and forest dynamics modeling have a long tradition in forestry [2]. Individual-tree simulators (ITS), which implement individual-tree FGM models, keep track of the development over time and fate of all trees in the simulation. ITS keep track of thousands to millions of alive, dead, and harvested trees through a simulation that can be as long as several centuries. This is a cumbersome task that can be taxing in terms of computer resources and time. That is why most ITS are written in a combination of languages such as Fortran, C, C++, or Java [for example, see 3,4]. This makes the simulations faster but at the cost of transparency and flexibility. Manipulation of the code is harder because it requires specific knowledge of one or several programming languages and it is often only a small number of designated programmers in a single research group that can make changes to the code or add/test a new model. When a new FGM sub-model describing a single process (for example, mortality) has been developed, the sub-model will normally have to be evaluated in conjunction with the exiting sub-models in order to understand the new sub-model performance and effect on the predicted forest dynamics. Hence, development and testing of new sub-models requires access and ability to reprogram the simulator.

Forestry researchers often use R [5] to perform statistical data analysis and to develop new FGM sub-models [for example, see 6,7]. With *SiTree* researchers can easily incorporate their new sub-models with existing models in a FGS. Researchers can then investigate the impacts of these models on forest growth and dynamics without having to learn a new programming language or having to depend on a programmer to translate their sub-models into a different programming language. Further, the `sitree` package is intended to be a collaborative effort that makes new models and functions easily available for the forestry community. While the `sitree` package will stay as the core of the framework, we have created the sister package `sitreeE` [8], SiTree Extension, intended to share new models and functions (for example, growth sub-models, biomass functions, competition indexes) with the forest research community. We also encourage researchers that develop FGM functions in R, to share their work in *SiTreeE*, so the rest of the community can use, test, and validate the new functions in a straight forward way.

SiTree is a framework that provides functionality to implement single tree simulators which can be used to test new sub-models that might be later implemented in a specialized FGS, or it could also be useful for building individual-tree simulators for research specific projects such as doctoral or post-doctoral studies. *SiTree* can also be used to implement a full forest growth and yield simulator, such as in [9], including, for example, ad-hoc management options, climate induced changes in productivity, and/or disturbances effects. Functions built for Norwegian conditions of the main sub-models are provided as example. These sub-models are not intended to be used as a valid Norwegian FGS, but as a showcase of the capabilities of *SiTree*.

Simulations produced using *SiTree* are currently and actively being used to inform policy decisions [for example, see 10], as well as in research [for example, see 9,11].

The next section describes the software we have developed, while the following section (Section 3) provides an example of how to use *SiTree*.

2. Software description

SiTree is fully written in R, and its latest version (0.1-12) is downloadable from the Comprehensive R Archive network (CRAN) at https://cran.r-project.org/package=sitree.

SiTree requires both tree-level data and plot/stand-level data to run a simulation. In *SiTree* a tree must be defined by a minimum of three variables, two that are time-dependent, for example, diameter and height, and one which is time independent, for example, tree species, but additional tree characteristics, such as crown ratio or vigor, can be added. Plot/stand-level data contains information at the plot and stand-level, such as plot size, elevation, climatic variables, or latitude. Although the data examples included in *SiTree* use diameter at breast height (*dbh*) in mm, and tree height in dm, the time-dependent variables could be in any other unit (for example, inches) or be any other allometric variable such as crown ratio, leaf area index or above-ground biomass.

The simulation starts at `t0`, which corresponds to the state of the trees at the time of the inventory (the tree data provided), and the forecasted periods have consecutive names, that is, `t1`, `t2`, etc. The length of the period is defined in the `sitree()` function in `period.length`, but it is the same length throughout the simulation, that is, all periods have the same length. *SiTree* returns all tree and plot data (`plot.data`), including diameter and height for all periods and for all trees (dead, alive, and harvested), and tree species. Through `plot.data` `sitree()` returns also time-varying plot-level simulation results, such as management (if management has been defined).

Processes that drive tree and stand development such as disturbances, resource availability, or site productivity can also be included in *SiTree*. For example, changes in site productivity driven by fertilization or climate change can be calculated as a modifier (`fn.modif`) function and tracked in `plot.data` as a time-varying variable. Disturbances that affect tree survival, such as bark beetle attacks, can be included in the mortality sub-model in combination with competition-induced mortality. Seed dispersal and seedling survival can be calculated in the management function (`fn.management`) and tracked in `plot.data` until they are considered trees and added to the tree list. Although *SiTree* is fully written in R, user-defined functions can include procedures written in the C, C++, .Net, Python or FORTRAN languages thanks to the capabilities of R to integrate these languages.

SiTree stores data in R “reference class” objects. Reference class objects are mutable, so the typical behavior of R objects “copy on modify” does not apply. This keeps memory usage at bay, allowing for updating and manipulating potentially large tree datasets in a memory efficient way, and making it possible to run long simulations with large datasets. There are two types of reference class objects in *SiTree*: `trList` for alive trees and `trListDead` for removed and dead trees.

In *SiTree* maximum flexibility is one of the main goals, implying that none of the main functions are hard-coded and all functions can be selected, added or modified by the user. To make the task of addition and writing the most frequently used functions (growth, mortality, recruitment, and management) easier, we have provided examples for all the sub-models and vignettes that explain the main functionalities of *SiTree*.

2.1. Software architecture

The outline of the flow of execution within `sitree()` is shown in Fig. 2. `sitree()` first runs some basic checks on the input data. It then creates the tree list (object of class `trList`) and then enters the loop for the main simulation. The loop runs from 0 to (`n.periods - 1`) periods. The loop starts by calculating external modifiers, if supplied. Then it calculates the common variables that are needed for the simulation. The next step is to define management. Management would usually include the removal of trees (for example, thinning, clear cuts, shelterwood harvest). Other type of management that affect plot/stand-level characteristics, such as fertilization should be calculated in the `fn.modif` function. Once external modifiers, common variables, management, growth, mortality, and recruitment (which potentially includes natural and artificial regeneration) are calculated `sitree()` applies the changes. First, growth of diameter and height is applied to the live trees, then dead trees that are not scheduled for harvesting are moved to the `dead.trees` list, then removed trees are moved to the `removed.trees` list, and finally the new trees from ingrowth are added to the `tr` list for alive trees. Growth of the dead/harvested trees between the last period that they were alive and their harvest/death is calculated with the `dead.trees.growth` function. By default, the growth before the harvest/death of the tree is assumed to be half of the growth for the whole period, but the `dead.trees.growth` function can be modified.

The `sitree()` function returns a list with four elements, `live`, `dead`, `removed`, and `plot.data`. `live` contains all measurements of all trees while they were alive in the simulation. For measurements where trees were not alive the values for diameter and height are zero. `dead` and `removed` elements contain all measurements of the dead/removed trees while they were alive, and their estimated dbh and height at harvest/death.

3. Illustrative examples

In this section we provide an example of *SiTree* simulations. Further examples are available in the package’s vignettes.

3.0.1. Simulation of a whole stand

We forecast the development of a stand in the west coast of Norway through the next 60 years using the `stand.west.tr` and `stand.west.st` data sets available in *SiTree*. Since the stand is around 1000 m^2 and all the functions that we are using are developed for 250 m^2 plots, the stand is divided into 4 plots of approximately 250 m^2 . For this example, we choose no management and no external modifier function. We run the simulation as follows:

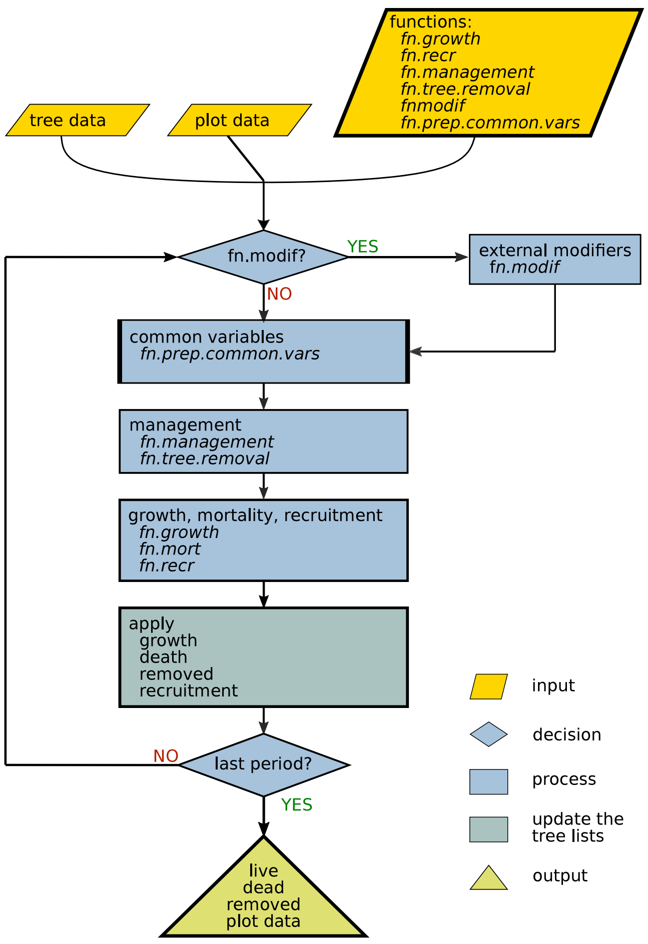

Fig. 2. Flowchart showing the workflow in *SiTree*. Inputs are indicated in yellow, processes and conditional operations (decisions) in blue, processes that modify the tree lists for alive, dead, and removed trees in dark green, and outputs in light green. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

```
library(sitree)
result.sitree <- sitree (tree.df   = stand.west.tr,
                         stand.df  = stand.west.st,
                         functions = list(
                           fn.growth     = '
                               grow.dbhinc.hgtinc',
                           fn.mort       = 'mort.B2007'
                             ,
                           fn.recr       = '
                               recr.BBG2008',
                           fn.management = NULL,
                           fn.tree.removal = NULL,
                           fn.modif      = NULL,
                           fn.prep.common.vars = '
                               prep.common.vars.fun'
                         ),
                         n.periods = 12,
                         period.length = 5,
                         mng.options = NA,
                         print.comments = FALSE,
                         fn.dbh.inc     = 'dbhi.BN2009',
                         fn.hgt.inc     = 'height.korf'
                       )
```

We can take a look at some basic graphs (Fig. 3): development of basal area, number of stems, number of new dead trees, and height of the 10 tallest trees, using the `sitree.summary` function (see Appendix A for code). The `sitree.summary` function assumes that dbh is given in mm, height in dm, and that the plot data include a variable named `plot.size.m2` which contains

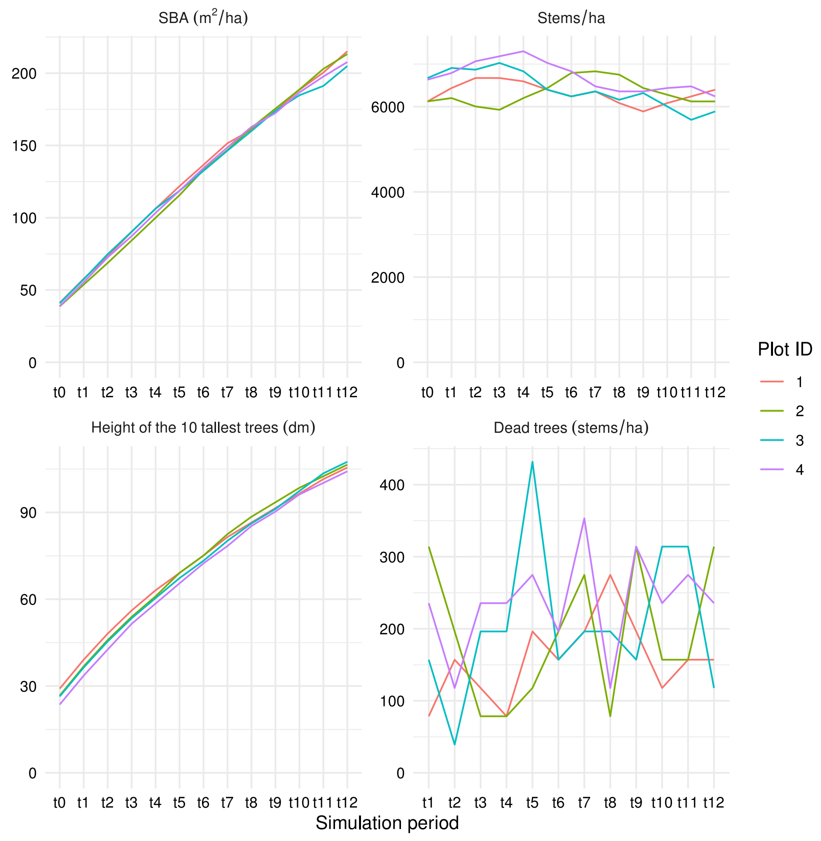

Fig. 3. Summary plots including average height of the 10 tallest trees (top left), number of dead trees per ha (top right), stand basal area (bottom left), and number of trees per ha (bottom right). The variable displayed in the *y*-axis is described in the caption of each panel. The *x*-axis is, for all panels, the simulation period. For example, `t0` corresponds to the initial data, the inventory data, and `t1` corresponds to the first cycle of the simulation. The plots have been produced using the `sitree.summary()` function.

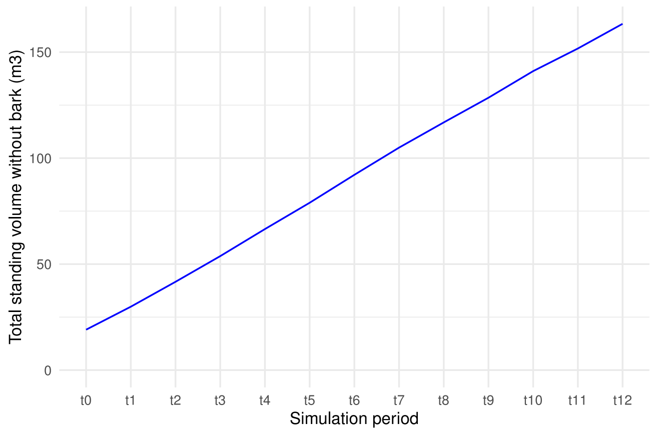

Fig. 4. Development of the stand volume for the next 60 years (12 periods of 5 years).

the plot size in squared meters, which is used to calculate the variables per hectare, for example, number of stems per ha.

The plots passed by `sitree.summary()` can be further modified using the `ggplot2` package capabilities. We could also use some of the functions available in *SiTreeE* to calculate, for example, standing volume for each of the 4 plots, see Fig. 4.

4. Impact

SiTree offers researchers a framework to easily implement their forest growth models using exclusively R, a popular language among forest growth modelers. It also allows to easily exchange forest growth sub-models, which allows for effortless testing of new or alternative sub-models within the simulator.

Many of the existing FGSs are closed-source [for example, see 3] and most of them are written in languages that most researchers within forestry are not comfortable with (for example, SILVA [12], which uses C++). Although there are some FGSs implemented as R packages (for example, r3PG [13]), or that have R wrappers (for example, rFVS https://sourceforge.net/p/open-fvs/wiki/rFVS), to our knowledge only two other FGSs frameworks exist: Capsis [14], which is written in Java, and SIMO [15], which is driven by external XML files. Because *SiTree* is entirely written in R it reduces the burden for forest growth modelers of testing new sub-models within an existing FGS, and it also reduces the learning curve of building a new simulator which is robust, fast, flexible, and memory efficient.

5. Conclusions

SiTree is a flexible, cross-platform, open-source individual-tree simulator framework intended to facilitate accurate and flexible analyses of forest growth and yield. We hope that it will provide a useful simulation framework to the large R-based forest growth and yield community and that the framework will be expanded with many new functions.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.softx.2021.100925.

References

[1] Hasenauer H, editor. Sustainable forest management: growth models for Europe. Berlin Heidelberg: Springer-Verlag; 2006.

[2] Weiskittel AR, Hann DW, Kershaw Jr. JA, Vanclay JK. Forest growth and yield modeling. John Wiley & Sons; 2011.

[3] Burkhart HE, Amateis RL, Westfall JA, Daniels RF. PTAEDA4.0: simulation of individual tree growth, stand development and economic evaluation in Loblolly pine plantations. Blacksburg, VA: Virginia Tech University; 2008, p. 27, https://fmrc.frec.vt.edu/content/dam/fmrc_frec_vt_edu/documents/ptaeda4.0manual.pdf.

[4] Nagel J. TreeGrOSS: Tree growth open source software—a tree growth model component. Göttingen, Germany: Niedersächsischen Forstlichen Versuchsanstalt, , Abteilung Waldwachstum; 2003, URL http://treegross.sourceforge.net/treegross.pdf.

[5] Team RC. R: A language and environment for statistical computing. Vienna, Austria: R Foundation for Statistical Computing; 2020, URL https://www.R-project.org/.

[6] Hall KB, Stape J, Bullock BP, Frederick D, Wright J, Scolforo HF, et al. A growth and yield model for eucalyptus benthamii in the Southeastern United States. For Sci 2020;66(1):25–37. http://dx.doi.org/10.1093/forsci/fxz061.

[7] Robinson AP, Hamann JD. Forest Analytics with R. New York, NY: Springer New York; 2011.

[8] Anton Fernandez C. SiTreeE: SiTree extensions. 2021, R package version 0.0-7, URL https://CRAN.R-project.org/package=sitreeE.

[9] Bright RM, Allen M, Antón-Fernández C, Belbo H, Dalsgaard L, Eisner S, et al. Evaluating the terrestrial carbon dioxide removal potential of improved forest management and accelerated forest conversion in Norway. Global Change Biol 2020;26(9):5087–105. http://dx.doi.org/10.1111/gcb.15228.

[10] Søgaard G, Alfredsen G, Antón-Fernández C, Astrup RA, Blom HH, Clarke N, Eriksen R, et al. Klimakur 2030–beskrivelse av utvalgte klimatiltak knyttet til skog. 2020, NIBIO Rapport, URL http://hdl.handle.net/11250/2639345, Publisher: NIBIO.

[11] Majasalmi T, Allen M, Antón-Fernández C, Astrup R, Bright RM. A simple grid-based framework for simulating forest structural trajectories linked to transient forest management scenarios in fennoscandia. Clim Change 2020. http://dx.doi.org/10.1007/s10584-020-02742-1.

[12] Pretzsch H, Biber P, Ďurský J. The single tree-based stand simulator SILVA: construction, application and evaluation. In: National and Regional Climate Change Impact Assessments in the Forestry Sector, Forest Ecol Manag In: National and Regional Climate Change Impact Assessments in the Forestry Sector, 2002;162(1):3–21.http://dx.doi.org/10.1016/S0378-1127(02)00047-6,

[13] Trotsiuk V, Hartig F, Forrester DI. r3PG – an r package for simulating forest growth using the 3-PG process-based model. Methods Ecol Evol 2020;11(11):1470–5. http://dx.doi.org/10.1111/2041-210X.13474.

[14] Dufour-Kowalski S, Courbaud B, Dreyfus P, Meredieu C, de Coligny F. Capsis: an open software framework and community for forest growth modelling. Ann For Sci 2012;69(2):221–33. http://dx.doi.org/10.1007/s13595-011-0140-9.

[15] Rasinmäki J, Mäkinen A, Kalliovirta J. SIMO: An adaptable simulation framework for multiscale forest resource data. Comput Electron Agric 2009;66(1):76–84. http://dx.doi.org/10.1016/j.compag.2008.12.007.