ModelBark

v1.0

GI en Especies Leñosas (WooSP)
Dpto. Sistemas y Recursos Naturales
ETSI Montes, Forestal y del Medio Natural
Universidad Politécnica de Madrid

https://github.com/ggfhf/

Table of contents

Disclaimer

The software package ModelBark is available for free download from the GitHub software repository (https://github.com/GGFHF/ModelBark) under GNU General Public License v3.0.

Introduction

ModelBark is a software application to simulate outer bark growth of woody plant species using a simple mathematical model based on the idea of cellular automata (Wolfram, 1983). The application simulates the evolution of cellular components of a secondary stem radius taking into consideration the position of the cells of each type (xylem, vascular cambium, active and inactive phloem, phellogen and phellem) and the mechanical stimuli produce by secondary growth.

ModelBark facilitates testing hypothesis related to bark development mediated by mechanical stimuli in woody plant species and is able to produce different bark types (thick, thin or rhytidome-like barks) by using different parameter combinations.

This manual briefly presents the rationale of the mathematical model, installation instructions and instructions to run the application. A more detailed descritption of the mathematical model and a case study exemplifying the capabilities of ModelBark are available at:

Gutiérrez Climent Á, Nuño JC, López de Heredia U, Soto Á. MODELBARK: a toy model to study bark formation in woody species. *In silico* Plants (Submitted)

ModelBark rationale

ModelBark centers on the mechanical induction of phellogen activation, considering that the meristematic activity of the vascular cambium, responsible for producing secondary xylem and phloem, pushes outward all surrounding tissues, resulting in diametrical growth and consequently, an expansion of the stem (or root) circumference. Notably, the growth of xylem exerts pressure on the vascular cambium itself, causing it to expand through anticlinal divisions. Consider a cell located at a distance r_0 from the center of the section, on a circumference of length $2\pi r_0$. As time progresses, the activity of the vascular cambium leads to a radial increase of Δr; to accommodate this growth, the circumference must also increase in $2\pi\Delta r$. This expansion generates tangential tension, according to Hooke's law, expressed as $k2\pi\Delta r$,where k represents the elasticity constant of the tissue. While cellular elasticity and cohesion between neighboring cells resist this tension, it may eventually result in tissue rupture. We propose that when this tension surpasses a certain threshold, it triggers the activation of a phellogen at a more internal position.

In our model, we not only take into account the phellogen's capacity to endure the pressure, including its ability to increase circumference through anticlinal divisions, but also the contribution of outer cells. These cells can partially counteract the outward pressure stemming from vascular cambium activity through elasticity and cohesion. Nevertheless, when the resultant pressure surpasses a threshold once more, it triggers the induction of a new, more internally located phellogen, and the cycle continues.

ModelBark is a one-dimensional Cellular Automaton (CA) representing a single stem radius, comprised solely of axial cells. Programming a CA typically involves defining several attributes. The "space" refers to the collection of cells upon which a metric is defined and can be conceptualized as a one-dimensional matrix (sequence) of increasing size, analogous to a stem radius. In our model, we locate the center of the stem at the leftmost end of the CA sequence.

Each cell within the space can adopt a finite set of states or values that are updated discretely, beginning from an initial configuration consisting solely of a vascular cambium cell. For simplicity, we focus solely on secondary tissues, disregarding any preexisting primary cells. The states of the cells, representing different tissues, are mapped to integer values (**Table 1**):

Table 1. Cell types and user-defined parameters considered in the model.

Cell type	**Code**	**Location**
Xylem	0	To the right of vascular cambium
Vascular cambium	1	-
Active phloem	0	Between vascular cambium and phellogen
Phellogen	2	-
Phellem	3, 4, 5, 6, 7, …	-
Inactive phloem	0	To the right of a periderm
Parameter	**Description**	
β	Division rate of the phellogen	
r	Expected proportion of xylem.	
a	Mechanical coefficient for xylem	
b	Mechanical coefficient for active phloem	
c	Mechanical coefficient for phellem (suber)	
d	Mechanical coefficient for inactive phloem	
K	Threshold for new phellogen onset (eq. 2)	
P	Location of the new phellogen (distant percentile of active phloem)	
ML	Maximum radius length (in cells). End of the simulation	

As the meristems undergo division, the length of the CA increases. At any given step t, the length of the CA, denoted as L(t), is the sum of cells across all tissues (assuming equal cell sizes) at that step, expressed by equation (1):

$$L(t)=Nx(t)+Nvc(t)+Np(t)+Ncc(t)+Ns(t)+Nip(t) \quad \text{(eq.1)}$$

Nx(t) --> number of xylem cells

Nvc(t) --> number of vascular cambium cells (considered as one)

Np(t) --> number of active phloem cells

Ncc(t) --> number of phellogen (cork cambium) cells (also considered as one)

Ns(t) --> number of phellem (suber) cells

Nip(t) --> number of non-functional (inactive) phloem cells.

Initially, the CA consists of a single vascular cell, yielding a length of L(0)=1. CA growth will stop when a maximum cell number (defined by the user) is achieved. The CA expansion occurs via two probabilistic transitions governed by their corresponding functions:

Function 1: Vascular cambium activity

A probabilistic function adds a vascular cell (coded as 0) either to the left (as a xylem cell) or to the right (as a phloem cell) of the initial vascular cambium (coded as 1). At each simulation step, a new vascular cell is generated. This new cell stochastically corresponds to either a xylem or a phloem cell based on a user-defined proportion r=Nx/Np. In reality, both the division rate of the vascular cambium and the ratio of xylem to phloem cells produced vary with the age and diameter of the stem. Nevertheless, for the sake of simplicity, ModelBark assumes both parameters as constant throughout the simulation.

Function 2: Onset of the first and successive phellogens.

The following transition function (eq. 2) governs the activation of phellogens:

$$F(t) = \frac{(1+c*Ns(t) + d*Nip(t))}{(a*Nx(t)+ b*Np(t))} \qquad \text{(eq.2)}$$

Nx, **Np**, **Ns**, and **Nip** are as defined in eq.1. a, b, c and d are coefficients accounting for the mechanical stress exerted on the phellogen by:

a --> xylem.

b --> active phloem cells (inner cells).

c --> phellem

d --> inactive phloem cells (outer cells).

When the function F(t) falls below a user-defined threshold K, the first phellogen is inserted at a user-defined position as the furthest *p* percentile within the active phloem, denoted as [2].

Phloem cells located outside the phellogen are subsequently considered phloem. inactive The initial appearance of a phellogen leads to an increase in the *F(t)* value due to the inactivation of some phloem cells (and their relocation in the numerator). Subsequently, whenever $F(t) \leq K$, a new phellogen, is inserted at the same distal percentile of active phloem. At this juncture, the previous phellogen is deemed inactive. Successive phellems produced whenever $F(t) \leq K$ are consecutively coded as [3], [4], [5], [6], and so forth (Table 1).

Once the phellogen appears, it will undergo division at each simulation step with a user-defined probability β. Phellogen divisions exclusively yield phellem cells outward, thereby pushing out on the outer tissues.

Installation

ModelBark was programmed in Python 3.8.8. and features a user-friendly GUI that functions on any computer with an operating system supporting Python 3 (including Linux/Unix, Microsoft Windows, Mac OS X, and other platforms), provided the following dependencies are installed: Tkinter, NumPy, Matplotlib, Pandas, and Pillow.

ModelBark installation

ModelBark is available from the GitHub software repository in the URL https://github.com/GGFHF/ModelBark, and it is distributed under GNU General Public License Version 3 (Figure 1).

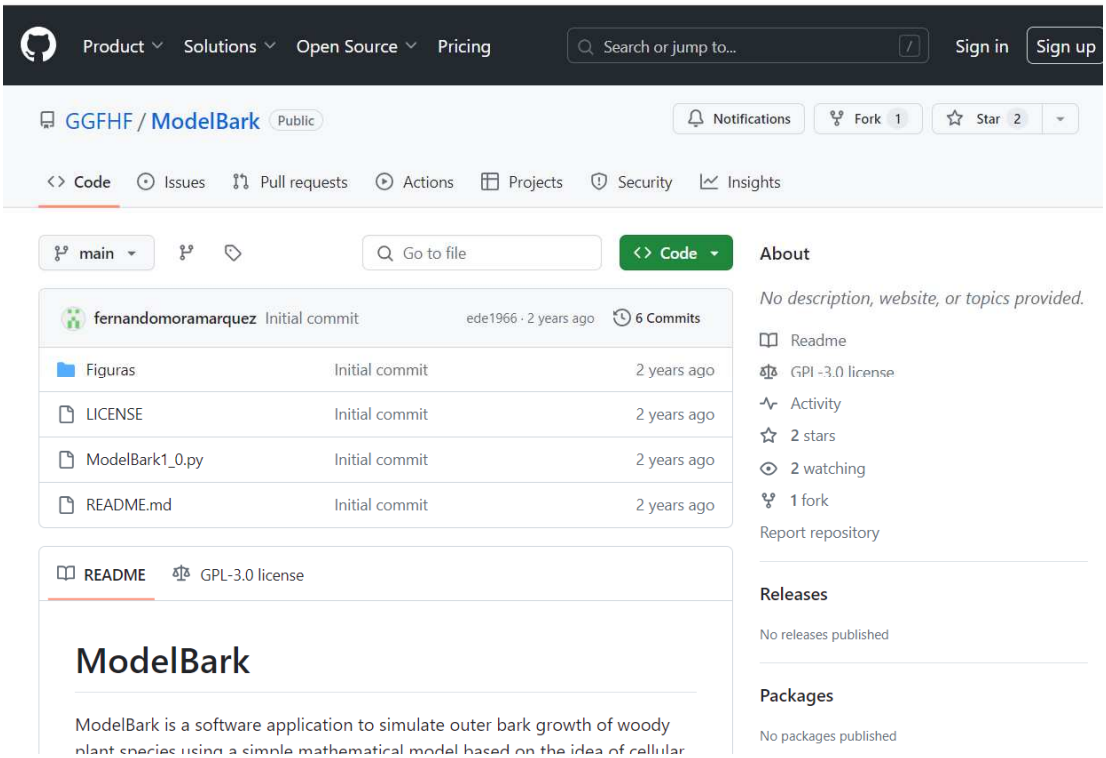

Figure 1. ModelBark home at GitHub software repository.

To download ModelBark, click in *Code* and in the pup-up window click in *Download ZIP*.

To install ModelBark on Linux and macOS, download simply decompress the ModelBark-main.zip into a directory, typing the following command in a terminal window:

```
$ unzip ModelBark-main.zip
```

Then, the execution permissions of the programs must be set by using this command:

```
$ chmod u+x *.py
```

To work properly the directory that hosts ModelBark1_0.py requires an additional empty directory called Figuras (already available in the compressed file).

For Microsoft Windows, simply unzip ModelBark-main.zip in the usual way.

ModelBark dependencies

ModelBark was programmed in Python3, and runs in any computer with an OS that allows for Python 3: Linux/Unix, Microsoft Windows, Mac OS X and other platforms. If Linux is the OS of your computer, Python will be already installed. In other cases, it can be downloaded from https://www.python.org/. For Windows, you can download both Python versions from the official website (https://www.python.org/), or use one of the several distributions that include Python along with other software packages for standard bioinformatic analysis. We recommend installing Anaconda (a version corresponding to Python 3.8.8 or higher). Anaconda is a free cross-platform for Microsoft Windows, Linux and macOS (https://www.continuum.io/). The installation instructions for Anaconda are available on its web site.

To work properly, ModelBark needs the following Python modules:

- Tkinter: https://docs.python.org/3.8/library/tk.html
- PIL: https://pypi.org/project/Pillow/
- Numpy: https://numpy.org/
- Matplotlib: https://matplotlib.org/
- Pandas: https://pandas.pydata.org/

Ubuntu 24.04 LTS (Noble Numbat) / Ubuntu 22.04 (Jammy Jellyfish)

Type the following commands in a terminal window to install pip3 and the Python 3 required modules modules Tk, PIL, Numpy, Pandas and Matplotlib if necessary:

```
Python 3 - pip:

      $ sudo apt install --yes python3-pip

Python - Tkinter:

      $ sudo apt install --yes python3-tk

      $ sudo apt install --yes python3-tk-dbg

      $ sudo apt install --yes python3-ttkthemes

      $ sudo apt install --yes python3-tktreectrl

Python - Python Imaging Library:

      $ sudo apt install --yes python3-pil

      $ sudo apt install --yes python3-pil.imagetk

Python - NumPy:

      $ sudo apt install --yes python3-numpy

      $ sudo apt install --yes python3-numpydoc

Python - Pandas:
```

```
$ sudo apt install --yes python3-pandas

$ sudo apt install --yes python3-pandas-lib

$ sudo apt install --yes python-pandas-doc
```

```
Python - Matplotlib:

$ sudo apt install --yes python3-matplotlib

$ sudo apt install --yes python3-matplotlib-venn

$ sudo apt install --yes python-matplotlib-data

$ sudo apt install --yes python-matplotlib-doc
```

macOS 13.10

First, install Homebrew and wget command, if necessary, typing the following commands in the terminal window:

$ /usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"

$ brew install wget

Now download the Anaconda software file, e.g. the version 2022.10, typing the command:

$ wget https://repo.anaconda.com/archive/Anaconda3-2022.10-MacOSX-x86_64.sh

and provide execution permission to this file and run it typing the commands:

$ chmod u+x Anaconda3-2022.10-MacOSX-x86_64.sh

$ ./Anaconda3-2022.10-MacOSX-x86_64.sh

During the Anaconda installation, read the Anaconda End User License Agreement, accept the license terms and indicate the location where Anaconda will be installed. Later, review that the PATH variable incorporates the directory where Python is. So, if Anaconda installation directory is Anaconda3_path, edit .bash_profile file in the user's home directory and check the directory Anaconda3_path/bin. is added in the PATH variable through a sentence like this: export PATH=Anaconda3_path/bin:$PATH The appearance of the file .bash_profile must be similar to the file shown in the Figure 2.

```
# >>> conda initialize >>>
# !! Contents within this block are managed by 'conda init' !!
__conda_setup="$('/Users/user/anaconda3/bin/conda' 'shell.zsh' 'hook' 2> /dev/null)"
if [ $? -eq 0 ]; then
    eval "$__conda_setup"
else
    if [ -f "/Users/user/anaconda3/etc/profile.d/conda.sh" ]; then
        . "/Users/user/anaconda3/etc/profile.d/conda.sh"
    else
        export PATH="/Users/user/anaconda3/bin:$PATH"
    fi
fi
unset __conda_setup
# <<< conda initialize <<<
```

Figure 2. An example of the file macOS *.zshrc*.

It is also possible to install the Anaconda distribution using a graphical installer available at the url https://www.anaconda.com/products/distribution/.

The Anaconda installation already includes the python modules required by ModelBark.

Microsoft Windows 10 (64-bits)

In order to have Python 3 in a Windows computer, we can install the Anaconda distribution. There is a Anaconda graphical installer for Windows in the url https://www.anaconda.com/products/distribution/.

Next, you must review the window "Edit environment variable" in "System Properties" and verify that the following directories are declared as PATH variables:

- *Anaconda3_path*
- *Anaconda3_path*\Library\mingw-w64\bin
- *Anaconda3_path*\Library\usr\bin
- *Anaconda3_path*\Library\bin
- *Anaconda3_path*\Scripts

Anaconda3_path is the directory where Anaconda3 is installed (the default directory proposed in the installation was C:\ProgramData\Anaconda3, but it could be changed).

You can see below an example of this windows (Figure 3):

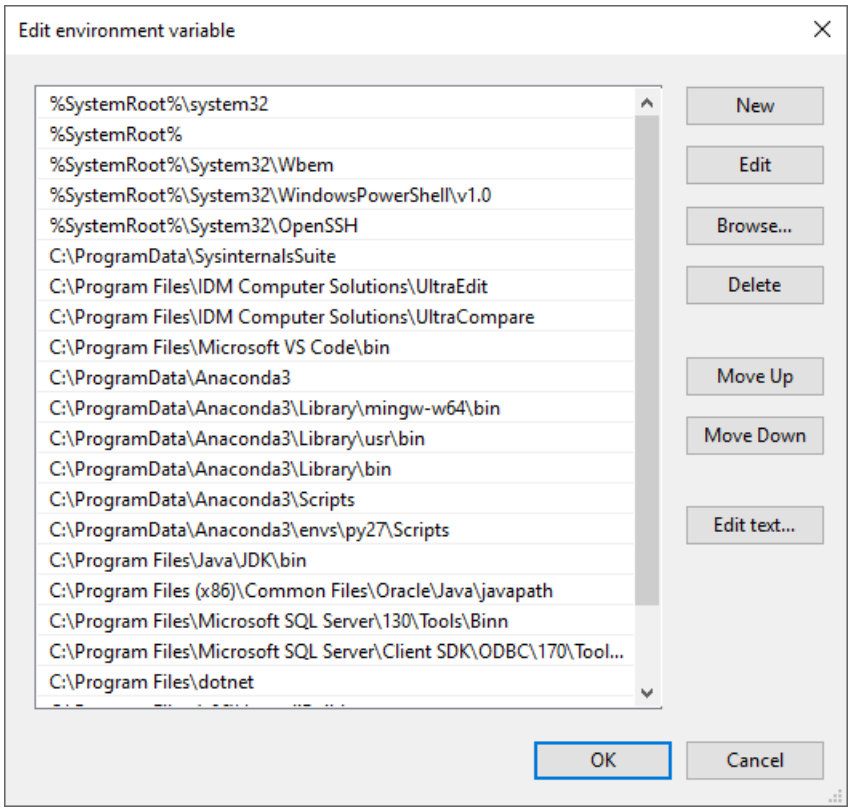

Figure 3. Environment variable window on Windows 10.

The Anaconda installation already includes the python modules required by ModelBark.

Running ModelBark

On Ubuntu or macOS computers, ModelBark starts typing the following command in a terminal window in the directory where the package of ModelBark is downloaded:

```
$ ./ModelBark_v1.0.py
```

On Windows computer, the command to start ModelBark is:

```
> python ModelBark_v1.0.py
```

Initial appearance of ModelBark at application startup is shown in Figure 4.

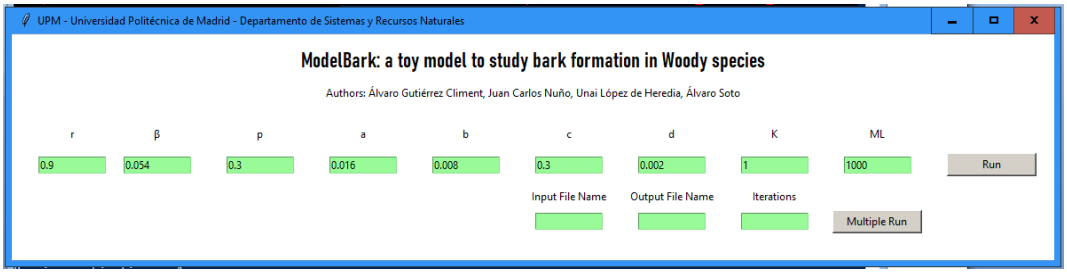

Figure 4. Aspect of the ModelBark interface at startup.

ModelBark can operate in two ways:

(1) Single Run mode. Simulating a single cellular automata using a one-by-one parameter input mode, where users directly input parameter values into corresponding GUI boxes and directly obtain the result by clicking Run (Figure 4).

The output generated by ModelBark for a single simulation includes the parameters utilized, statistical information (such as the number of cells of each type), the final numerical vector produced, and a graphical representation thereof. Additionally, two plots are provided: one illustrating the evolution of cell counts and another depicting the evolution of the transition function *F(t)* throughout the simulation (Figure 5).

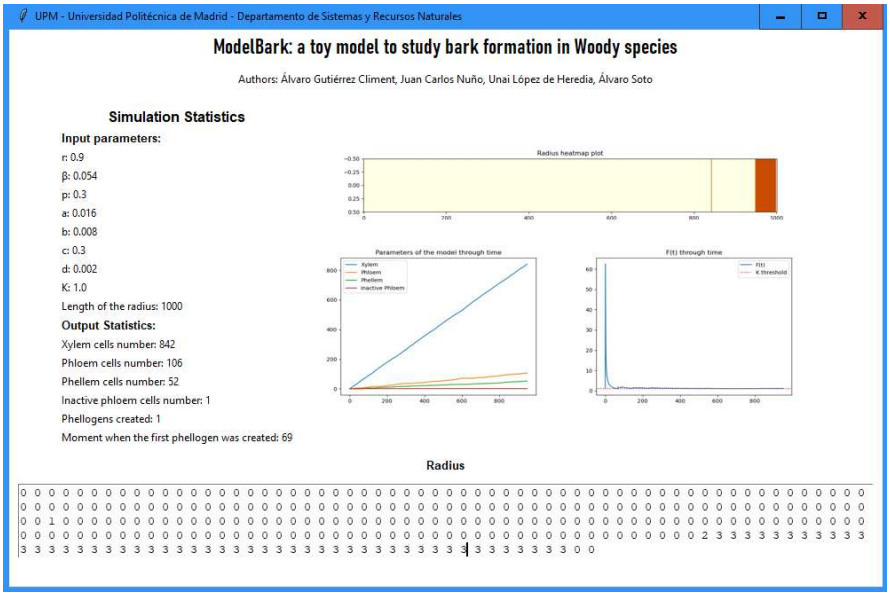

Figure 5. Output displayed in Single Run Mode.

(2) Multiple Run mode. The input consists of a multiple parameter combination in .csv format. An example of the input format for two parameter combinations is available with the distribution for testing purposes (*Multiple-parameter-input-test.csv*). The path to the input file (without extension) and to the output file must be indicated, along with the number of iterations for each parameter combination, before clicking the Multiple run button.

The output consists of a .csv file containing all the CA produced by running the Multiple Run mode. An example of the output file that generated 40 CA is available with the distribution for testing purposes (*ModelBark-output-test.csv*).This file can be post-processed with a R script (*MultipleRunsCircularOutput.R*), to generate circular plots in .pdf format (Figure 6). The script is easy to run and only requires indicating the working directory, the ModelBark output .csv file name and a name for the circular plot where indicated.

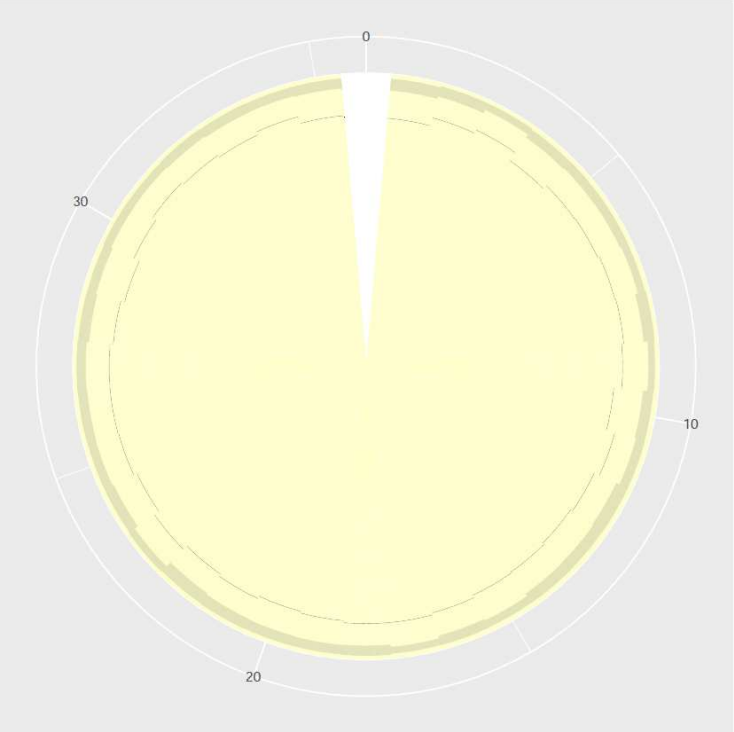

Figure 6. Circular plot generated for 40 CA with the R script MultipleRunsCircularOutput.R.