THIAGO TAGLIALEGNA SALLES

SIMULAÇÃO DE FLORESTAS EQUIÂNEAS

Tese apresentada à Universidade Federal de Viçosa, como parte das exigências do Programa de Pós-Graduação em Ciência Florestal, para obtenção do título de *Doctor Scientiae*.

VIÇOSA
MINAS GERAIS – BRASIL
2016

Ficha catalográfica preparada pela Biblioteca Central da Universidade Federal de Viçosa - Câmpus Viçosa

T
S168s
2016

Salles, Thiago Taglialegna, 1984-
Simulação de florestas equiâneas / Thiago Taglialegna Salles. – Viçosa, MG, 2016.
vii, 138f. : il. (algumas color.) ; 29 cm.

Inclui apêndice.
Orientador: Marcio Lopes da Silva.
Tese (doutorado) - Universidade Federal de Viçosa.
Inclui bibliografia.

1. Sistema de informação geográfica. 2. Florestas - Planejamento. I. Universidade Federal de Viçosa. Departamento de Engenharia Florestal. Programa de Pós-graduação em Ciência Florestal. II. Título.

CDD 22 ed. 526

THIAGO TAGLIALEGNA SALLES

SIMULAÇÃO DE FLORESTAS EQUIÂNEAS

Tese apresentada à Universidade Federal de Viçosa, como parte das exigências do Programa de Pós-Graduação em Ciência Florestal, para obtenção do título de *Doctor Scientiae*.

APROVADA: 14 de dezembro de 2016.

Daniel Henrique Breda Binoti

Carlos Alberto Araújo Júnior

Cibele Hummel do Amaral

Hélio Garcia Leite
(Coorientador)

Márcio Lopes da Silva
(Orientador)

Em algum lugar, algo incrível está esperando para ser descoberto.

(Carl Sagan)

AGRADECIMENTOS

Aos meus pais Ari e Rita, sem os quais nunca teria chegado onde cheguei.

Ao Departamento de Engenharia Florestal, pela oportunidade de cursar o Doutorado e à CAPES, pela concessão de bolsa de estudos.

Aos professores Márcio Lopes e Helio Garcia, pela amizade, pela orientação e por tudo que me permitiram realizar dentro da Universidade Federal de Viçosa.

Ao professor João Carlos Chagas Campos, por apresentar as primeiras ideias que levaram à realização deste trabalho.

Ao pessoal da secretaria de pós-graduação do DEF, Ritinha, Alexandre e Dilson, pelo apoio ao longo dos anos.

Aos professores da banca, Carlos Araújo, Daniel Binoti e Cibele Amaral, pelas sugestões e correções que contribuíram com este trabalho.

Ao Chiquinho da Floresta, pela amizade e apoio dentro do departamento.

Aos amigos do DEF, em especial os companheiros de orientação e pessoal da salinha da pós-graduação, pela colaboração em minhas atividades acadêmicas na UFV.

Aos amigos de Viçosa, pessoal da república e turma sempre presente nos bares, churrascos e afins, pela grande camaradagem.

À dona Cida por ter cuidado da nossa república em Viçosa.

Ao pessoal do LAPEM e da dap Florestal, por confiarem em meu trabalho no período que precedeu meu doutorado.

À Bárbara, melhor companheira do mundo.

MUITO OBRIGADO!

BIOGRAFIA

THIAGO TAGLIALEGNA SALLES, filho de Ari Jaime Salles e Rita de Cássia Taglialegna Salles, nasceu em Paraguaçu, Minas Gerais, em 29 de agosto de 1984.

Em março de 2003, iniciou o curso de Engenharia Florestal na Universidade Federal de Viçosa, Minas Gerais, graduando-se em janeiro de 2008.

Em março de 2008, ingressou no Programa de Pós-Graduação em Ciência Florestal na Universidade Federal de Viçosa, concluindo os requisitos para obtenção do título de *Mestre* em fevereiro de 2010.

Trabalhou como profissional autônomo até novembro de 2012, quando ingressou no Programa de Pós-Graduação em Ciência Florestal, em nível de doutorado, na Universidade Federal de Viçosa, concluindo os requisitos indispensáveis para obtenção do título de *Doutor* em dezembro de 2016.

SUMÁRIO

RESUMO

SALLES, Thiago Taglialegna, D.Sc., Universidade Federal de Viçosa, dezembro de 2016. **Simulação de florestas equiâneas**. Orientador: Márcio Lopes da Silva. Coorientador: Helio Garcia Leite.

Diversos estudos em ciência florestal dependem de um banco de dados que contenha as dimensões das árvores de uma floresta ao longo dos anos. Há casos onde não são necessários dados observados e as características das florestas podem ser obtidas por simulação. Diante disso, este estudo teve como objetivo desenvolver ferramentas para simular monoculturas florestais em nível de árvore para estudos de manejo. Buscou-se criar e testar, em um ambiente SIG, um conjunto de ferramentas para os fins de simulação. O desenvolvimento se deu no programa ArcGIS, por meio de programação em Python, para criar uma *Python toolbox*: um agrupamento de ferramentas que executam um trecho de código para realizar tarefas em dados geográficos. A eficácia das ferramentas foi validada simulando dois plantios conhecidos. O primeiro foi composto de híbridos de *Eucalyptus urophylla* x *Eucalyptus grandis* com arranjo espacial 3,0 x 3,0 m. O segundo foi composto por oito clones de eucalipto, em sistema agroflorestal, com arranjo espacial 10,0 x 4,0 m. Por fim, foi realizado um estudo econômico utilizando dados gerados pelas ferramentas. Dois cenários envolvendo o plantio de eucalipto no espaçamento de 3,0 x 3,0 m, simulado no item anterior, foram comparados: um desbaste simulado (por meio de scripts em Python, no ArcGIS) de 30% da área basal na idade de três anos, com corte da madeira remanescente aos seis anos; e apenas o corte raso na idade de seis anos. Os indicadores utilizados para a comparação foram o VPL, VAE e a TIR. A *toolbox* desenvolvida foi capaz de realizar as simulações desejadas, com resultados consistentes em relação ao crescimento em diâmetro quadrático médio, sobrevivência, média das alturas, volume por hectare e distribuição de diâmetros. Também foi possível reproduzir corretamente os dois plantios de eucalipto descritos na literatura. Por fim, o desbaste e o crescimento pós-desbaste puderam ser simulados corretamente através de programação, com este cenário apresentando melhor resultado econômico em relação ao corte e venda de toda a floresta aos seis anos.

ABSTRACT

SALLES, Thiago Taglialegna, D.Sc., Universidade Federal de Viçosa, December, 2016. **Simulation of even-aged forests**. Adviser: Márcio Lopes da Silva. Co-adviser: Helio Garcia Leite.

Several studies in forest science depend on a database that contains the dimensions of a forest's trees over the years. There are cases where observed data are not required and the characteristics of the forests can be obtained by simulation. Therefore, this study aimed to develop tools to simulate forest monocultures at single tree level for management studies. We sought to create and test, in a GIS environment, a toolkit for the simulation purposes. The development was done in the ArcGIS program, through programming in Python, to create a *Python toolbox*: a group of tools that execute a piece of code to perform tasks in geographic data. The effectiveness of the tools was validated by simulating two known plantations. The first one was composed of *Eucalyptus urophylla* x *Eucalyptus grandis* hybrids in 3.0 x 3.0 m spacing. The second one was composed of eight clones of eucalypt, in an agroforestry system, in 10.0 x 4.0 m spacing. Lastly, an economic study was carried out using data generated by the tools. Two scenarios involving the eucalyptus planting in the 3.0 x 3.0 m spacing simulated in the previous item were compared: a simulated thinning (by using Python scripts in ArcGIS) of 30% of the basal area at the age of three years, with cutting of the remaining wood at six years; and only the clear-cut at the age of six years. The indicators used for the comparison were NPV, EAV and IRR. The developed toolbox was able to perform the desired simulations, with consistent results regarding growth in mean squared diameter, survival, mean height, volume per hectare and diameter distribution. It was also possible to correctly reproduce the two eucalyptus plantations described in the literature. Lastly, thinning and post-thinning growth could be correctly simulated through programming, with this scenario presenting better economic result in relation to cutting and sale of the entire forest at six years.

1. INTRODUÇÃO

A ciência das tomadas de decisão considerando a organização, uso e conservação das florestas e de seus recursos relacionados é chamada de manejo florestal (Boungiorno e Guilles, 2003). Ela prevê, entre outras atividades, a especificação de intervenções silviculturais que atendam as demandas e restrições definidas para um determinado período de tempo. São levados em consideração aspectos econômicos, operacionais, ambientais e sociais, buscando sempre uma sequência de plantios e cortes de maneira que confiram sustentabilidade ao empreendimento florestal (ARAÚJO JÚNIOR, 2012; MACHADO e LOPES, 2008; NOBRE, 1999).

Os avanços frequentes em informática, como o aumento do poder de processamento dos computadores, o fácil acesso ao aprendizado da programação computacional e a sofisticação dos sistemas de informações geográficas (SIG) possibilitaram o desenvolvimento e a aplicação de técnicas de manejo florestal complexas e detalhadas. Como exemplos há Binoti (2012), que gerou quatro sistemas computacionais gratuitos para auxiliar profissionais do setor florestal na resolução de alguns problemas; Marcatti (2013), que estudou o caminhamento ótimo para acesso a parcelas de inventário florestal; Santos (2014), que estudou o crescimento radial mensal de povoamentos de eucalipto na idade de corte, visando propor um modelo de ordenamento da colheita e avaliou os ganhos da inclusão do crescimento mensal como critério de decisão em seu agendamento; Araújo Júnior (2016), que desenvolveu e implementou, por meio de sistemas multiagente, um modelo conceitual de um sistema inteligente para determinação do dimensionamento de frotas de transporte florestal; e Oliveira Neto (2016), que analisou por meio de programação linear a sensibilidade do planejamento estratégico em relação à área de reforma de plantios de eucalipto.

Estudos desta natureza normalmente dependem de um banco de dados contendo as características das árvores de uma floresta ao longo dos anos, o que pode vir a ser um obstáculo à sua execução. Isto porque a disponibilidade destes dados

muitas vezes é restrita, uma vez que se depende da existência de plantios que são dispendiosos, ocupam uma grande área e levam muito tempo para chegarem à idade de corte, limitando a viabilidade de povoamentos experimentais. Como solução, dados oriundos de empresas do setor florestal são amplamente utilizados em trabalhos acadêmicos, como os de Demolinari (2006), Castro (2007), Binoti (2010), Alcântara (2012) e Leite (2012).

Contudo, há casos onde não são necessários dados observados e as características das florestas podem ser obtidas por simulação. Esta situação é vista nos estudos de Martins et al. (2009), que realizaram uma avaliação técnica e econômica de um "harvester" trabalhando em diferentes condições de espaçamento e arranjo de plantios de eucalipto e Thompson et al. (2009), que estudaram estratégias de seleção e penalidades para algoritmos genéticos destinados a resolver problemas espaciais de planejamento florestal. Também há Rodrigues (2103), que avaliou a produção de água em função de alterações de uso do solo e da implantação de florestas de eucalipto em larga escala e Li et al. (2010), que realizaram comparações de três diferentes métodos de geração de cenários destinados a estudos de agendamento espacial de colheita com restrições de adjacência.

Usar florestas hipotéticas não é apenas uma segunda opção para quando a disponibilidade de dados do mundo real for limitada. Na verdade, é uma opção preferida em muitas situações, uma vez que cenários podem ser gerados repetidamente, sendo possível controlar fatores a um nível pré-definido e executar vários testes de hipóteses sobre as soluções finais de planejamento (LI et al., 2010). Esta situação pode ser vista nos estudos de Bettinger et al. (1997; 2002), em que técnicas de planejamento heurístico foram aplicadas a problemas de planejamento florestal e em Smith e Burkhart (1984), em que foram investigados os efeitos da amostragem nos valores das variáveis preditoras em estimativas de produção

A maneira mais comum para gerar dados simulados de florestas é criar grades ou mosaicos, onde cada célula representa um conjunto de árvores. Este tipo de dado apresenta os valores médios das características dos povoamentos, como volume por hectare, área basal, número de árvores por hectare, etc. A simulação em um nível maior de resolução (nível de árvore) permite lidar com a estrutura e padrão das árvores em toda idade, fornecendo maior flexibilidade para responder a diferentes propostas de manejo. Além disso, os valores médios em nível de povoamento podem

ser obtidos a partir das informações em nível de árvore (LI et al., 2010; PRETZSCH et al., 2002).

Em muitos casos pode ser desejável, ou mesmo imprescindível, que os dados simulados de uma floresta estejam dentro de um sistema de informações geográficas (SIG). Os SIG são sistemas que permitem a captura, modelagem, manipulação, recuperação, análise e apresentação de dados referenciados geograficamente (WORBOYS, 1995). A aplicação de SIG pode estar relacionada a estudos envolvendo, por exemplo, conservação do solo, avaliação de impactos ambientais, silvicultura de precisão, levantamento e mapeamento vegetal, estudos de balanço hídrico, inventário florestal e detecção de zonas de aptidão para espécies florestais (BRANDELEIRO et al., 2007; CASTRO, 2004; DIAS et al., 2001; DIAS, et al., 2008; FACCO et al., 2012; FERNANDES et al., 2002; ZAMPIN, 2010).

No Brasil, para fins de simulação, existem os softwares da família SIS da Embrapa, como o SIS Eucalipto, SIS Cedro, SIS Teca e SIS Ptaeda, que são simuladores para manejo, análise econômica, modelagem e de crescimento e produção de florestas equiâneas (EMBRAPA, 2016). O US-EVEN, que simula o crescimento de algumas espécies dos Estados Unidos (BARRETO, 1999) e o MOTTI que produz informações de florestas para a Finlândia (HYNYNEN et al., 2005), são exemplos semelhantes desenvolvidos em outros países. Estes softwares, apesar de eficazes naquilo que se propõem realizar, entregam os resultados apenas em nível de povoamento e sem vínculo com um SIG, limitando aplicações que demandam considerações de ordem espacial por indivíduo. No presente estudo estas limitações foram resolvidas. O objetivo portanto foi construir, implementar e validar um sistema para simular florestas equiâneas em nível de árvore, no tempo e no espaço.

Objetivos específicos foram:

- Criar e testar, em um ambiente SIG, um conjunto de ferramentas para simulação de monoculturas florestais em nível de árvore.
- Validar a eficácia das ferramentas, simulando dois plantios descritos na literatura.
- Demonstrar o emprego do sistema construído, em análises econômicas, para diferentes cenários de manejo.

2. REFERENCIAL TEÓRICO

2.1. Crescimento, produção e mortalidade

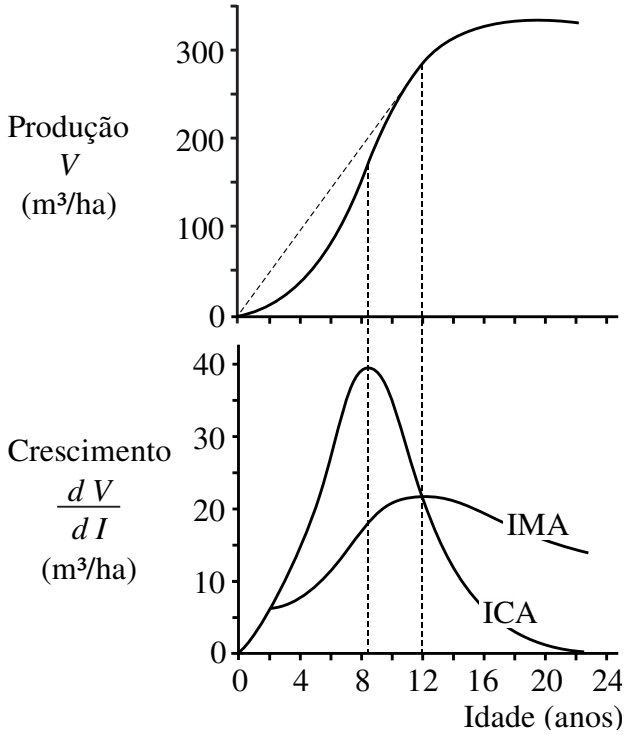
Os componentes mais importantes do desenvolvimento de florestas são o crescimento (incremento) e produção, ingresso e mortalidade. O crescimento de florestas pode ser definido como a mudança em um atributo do povoamento ao longo do tempo. Sendo assim, o crescimento é taxa de produção e a produção em si é o total acumulado em um intervalo de tempo. Desta forma, ao se expressar a produção com uma equação matemática, o crescimento pode ser obtido ao se derivar esta equação (DAVIS e JOHNSON, 1987).

Segundo Vanclay (1994), o incremento é o processo natural de crescimento das árvores vivas; o ingresso refere-se às novas árvores que atingiram e/ou ultrapassaram o tamanho mínimo predeterminado para que fossem medidas; e a mortalidade seria o número de árvores que morreu por causa de fatores como senescência, competição, pragas doenças, etc.

Daniel et al. (1979) generalizam dizendo que o crescimento de qualquer organismo vivo segue a forma sigmoide. Inicialmente, em uma curva de produção de volume de madeira versus idade em povoamentos equiâneos, há um crescimento moderado. Posteriormente, o crescimento acelera e a curva passa a apresentar uma forma convexa em relação ao eixo do tempo até um ponto de inflexão, onde o crecimento desacelera e a curva passa a ser côncava em relação a este eixo. Por fim a curva atinge um ponto de máximo, passando a ser levemente assintótica ou podendo até mesmo decrescer. Este decréscimo se dá em função da senescência do povoamento (ASSMANN, 1970; FINGER, 1992). Para o crescimento de árvores, os momentos em que os fenômenos mencionados ocorrem variam para cada situação de genótipo e ambiente, estando ligada à capacidade produtiva, espécie e densidade (espaçamento).

O comportamento descrito acima é ilustrado na Figura 2.1. O IMA é a taxa média de aumento da produção até uma determinada idade (CAMPOS e LEITE, 2013), podendo ser calculado para volume, peso ou outra variável. O ponto em que o

IMA atinge seu máximo, alcançado quando sua curva encontra a curva de incremento corrente anual (ICA), é por vezes usado como guia para decisões de colheita. O ICA corresponde à taxa de aumento da produção para um determinado período de tempo (DAVIS e JOHNSON, 1987). Assmann (1970) salienta três fases importantes do crescimento: a fase juvenil, que acompanha o aumento no incremento corrente até seu ponto de máximo; a fase de maturidade (maior vigor), que se inicia no ponto de máximo incremento corrente (também ponto de inflexão da curva de produção) e termina no ponto de máximo incremento médio; e a fase de senescência, que ocorre a partir do ponto de máximo incremento médio.

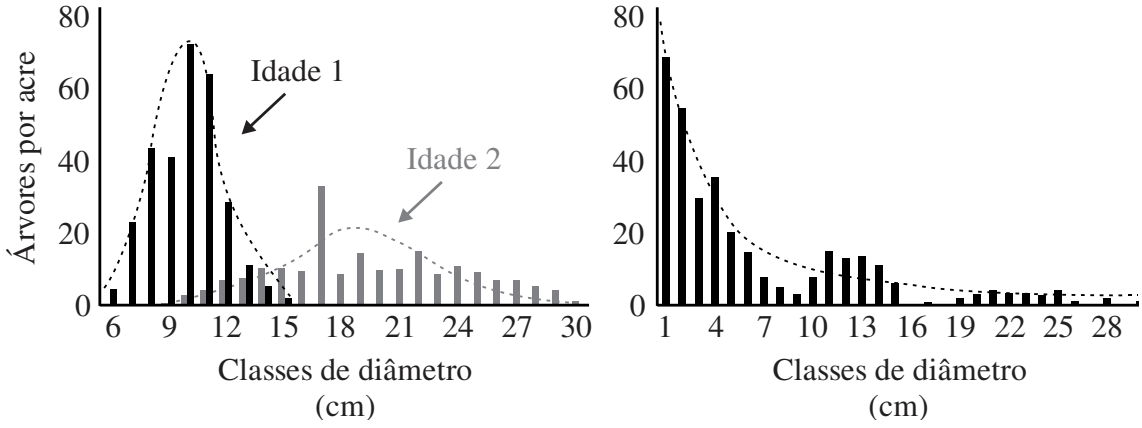


Fonte: adaptado de Campos e Leite (2013).

Figura 2.1 – Relações entre o crescimento e produção em floresta equiânea.

Outra forma de visualizar o desenvolvimento de um povoamento ou floresta é através de distribuições de diâmetro. Como o nome sugere, estas distribuições quantificam o número de árvores por unidade de área e classe de diâmetro. Para florestas equiâneas, a distribuição deve ser aproximadamente normal para uma determinada idade (Figura 2.2 – A). Povoamentos e florestas inequiâneas possuem mais de uma classe de idade e normalmente são compostas por uma grande

quantidade de árvores pequenas. Como resultado, uma distribuição de diâmetros de uma floresta inequiânea deve ter forma exponencial (Figura 2.2 – B) (BETTINGER et al., 2009).



Fonte: adaptado de Bettinger et al. (2009).

Figura 2.2 – Distribuições de diâmetros de florestas equiâneas (A) e de florestas inequiâneas (B).

Em relação à mortalidade, esta pode ser regular, devido à competição entre árvores, ou irregular, quando causada por catástrofes, pragas, incêndios e danos mecânicos (CAMPOS e LEITE, 2013).

2.2. Modelagem

A compreensão da dinâmica de crescimento e produção das florestas é necessária ao seu gerenciamento (CAMPOS e LEITE, 2013). As decisões de manejo são baseadas em dados de seus recursos correntes e futuros (AVERY e BURKHART, 1994).

O modelo de crescimento é uma abstração da dinâmica natural do povoamento e pode abranger, além do crescimento, a mortalidade e outras mudanças na composição e estrutura do povoamento. Compreender a modelagem do crescimento é importante para pesquisadores que estejam interessados em um melhor entendimento das relações existentes na produção durante a vida de uma floresta. (VANCLAY, 1994). Na Engenharia Florestal, os modelos são imprescindíveis na prognose dos recursos florestais, na escolha de metodologias silviculturais e no apoio

às tomadas de decisão no manejo e na política florestal (SPATHELF e NUTTO, 2000).

Existem diversos tipos de modelo capazes de prever o crescimento e a produção, normalmente divididos em duas categorias: os modelos biométricos (ou descritivos ou empíricos), e os modelos de processos (ou ecofisiológicos ou mecanísticos). Dentre modelos biométricos, estão os de crescimento e produção em nível de povoamento, os de distribuição de diâmetros e os de árvores individuais (DAVIS et al., 2005).

Os modelos de processo tentam modelar o crescimento, tomando como entrada variáveis como luz, temperatura e níveis de nutrientes do solo. O desafio é fornecer bases fisiológicas e ecológicas suficientes para assegurar previsões realistas sob uma variedade de locais e condições de povoamento, mesmo quando dados empíricos para calibração são limitados (VANCLAY, 1994). Apesar de ter maior capacidade de extrapolação em relação a outros modelos, seus resultados podem apresentar menor exatidão, como visto em Alexandre (2009). São importantes na análise de diferentes cenários com determinados conjuntos de condições de clima, solo, material genético e tipo de manejo. Permitem assim que sejam feitas expectativas de alguns dos impactos ambientais do processo produtivo (BORGES, 2009). Em estudos no Brasil, o modelo ecofisiológico mais difundido é o 3-PG. Os estudos de Landsberg et al. (1997), Almeida et al. (2004) e Stape et al. (2004) fazem uso deste modelo.

Modelos em nível de povoamento baseiam-se em parâmetros em nível de povoamento, como estoque (número de árvores por hectare), idade, área basal e volume para prever o crescimento ou a produção. Características das árovres podem ser inferidas, mas poucos detalhes de árvores individuais estão disponíveis nos resultados deste tipo de modelo (VANCLAY, 1994). O modelo em nível de povoamento mais utilizado em estudos de crescimento e produção no Brasil é o modelo de Clutter (1963). Este e outros modelos deste tipo são vistos nos trabalhos de Lundgren (1981), Zarnoch et al. (1991), Rodriguez et al. (1997), Nunifu et al. (1999), Bravo-Oviedo et al. (2004), Zonete et al. (2010), Schikowski et al. (2013), Miranda et al. (2015).

Os modelos de árvore individual (MAI) estimam os valores das variáveis dentrométricas de cada árvore do povoamento, bem como a mortalidade. Seu nível

de resolução, maior em relação a outros modelos, os torna uma boa ferramenta para simular diferentes prescrições de manejo (DAVIS et al., 2005). O volume por hectare nos MAI é obtido pela soma do volume de cada árvore (CAMPOS e LEITE, 2013). Este tipo de modelo resulta em descrição mais detalhada da estrutura e dinâmica dos povoamentos do que os modelos em nível de povoamento (MABVURIRA e MIINA, 2002). Eles podem ser do tipo independente da distância, utilizando apenas os atributos dendrométricos das árvores, ou dependente da distância, em que usam, além de dados dendrométricos, dados espaciais indicando a posição da árvore no povoamento. A exigência de dados espaciais pode restringir seu uso a regiões relativamente pequenas (VANCLAY, 1994). Modelos de árvores individual são encontrados em Qin e Cao (2006), Condés e Sterba (2008) e Martins (2011).

Os modelos de distribuição de diâmetros estimam, por meio de uma função densidade de probabilidade (fdp), o número de árvores por hectare por classe de diâmetro (CAMPOS e LEITE, 2013). Eles são adequados para analisar desbaste, uma vez que possibilitam a avaliação econômica de multiprodutos (BURKHART et al., 1981). Gadow (1984) diz que as fdp de Weibull (WEIBULL, 1951) e S_B de Johnson (JOHNSON, 1949) parecem ser as mais apropriadas para modelar distribuição de diâmetros em florestas equiâneas. Estes e outros modelos deste tipo estão presentes em Arce (2004), Bartoszeck et al. (2004) e Machado et al. (2009).

Funções de crescimento também são utilizadas para estimar a evolução das características das árvores, sendo exemplos os modelos Logístico (COX, 1958), Gompertz (GOMPERTZ, 1825), Richards (RICHARDS, 1959) e Lundqvist-Korf (KORF, 1939; LUNDQVIST, 1957).

2.3. Sistemas de informações geográficas

O ramo da ciência que utiliza técnicas matemáticas e computacionais para a análise e processamento de informações georreferenciadas é chamado de geoprocessamento, ou geomática. Os sistemas de informações geográficas (SIG) permitem a manipulação dos dados do geoprocessamento (PIROLI, 2010).

Um SIG é constituído por um conjunto de ferramentas especializadas em adquirir, armazenar, recuperar, transformar e emitir informações espaciais. Estas ferramentas trabalham com as características alfanuméricas de dados geográficos e

também com sua localização espacial (BURROUGH, 1986; DAVIS e CÂMARA, 2001).

Segundo Aronoff (1989), um SIG pode ainda ser definido como um sistema provido de quatro grupos de aptidões para manusear dados georreferenciados: entrada, gerenciamento, manipulação e análise, e saída. O autor comenta que a utilização de sistemas de informações geográficas traz as seguintes vantagens: os dados armazenados digitalmente estão em uma forma mais compacta do que se eles estivessem em mapas de papel; uma grande quantidade de dados pode ser mantida e recuperada com velocidade e a um baixo custo; é possível gerenciar dados espaciais e seus dados de atributos correspondentes, integrando diferentes tipos de dados de atributos em uma única análise, a alta velocidade; e é possível realizar rapidamente análises espaciais complexas e sucessivas, como simulações de diferentes cenários de planejamento.

São exemplos de softwares de SIG o QGIS, o Spring e o ArcGIS.

O QGIS (www.qgis.org) é um Sistema de Informação Geográfica livre, licenciado sob a *GNU General Public License*. É um projeto multiplataforma da Open Source Geospatial Foundation (OSGeo), rodando em Linux, Unix, Mac OSX ,Windows e Android.

O Spring (http://www.dpi.inpe.br/spring) é um SIG livre, desenvolvido pelo INPE em parceria com outras instituições. Possui funções de processamento de imagens, análise espacial, modelagem numérica de terreno (MNT) e consulta a bancos de dados espaciais. Dentre seus objetivos encontra-se o de fornecer um ambiente unificado de geoprocessamento e sensoriamento remoto para aplicações urbanas e ambientais, tornando-se amplamente acessível para a comunidade brasileira.

O ArcGIS (www.arcgis.com) é um conjunto de softwares para SIG (o mais popular em todo mundo) composto de alguns aplicativos com funcionalidades específicas: ArcView é o programa principal, utilizado para visualização e análise de dados, bem como criação e edição elementos geográficos simples; o ArcCatalog oferece uma janela de catálogo que é usada para organizar e gerenciar vários tipos de arquivos e informações geográficas utilizados no ArcGIS; o ArcScene é um visualizador 3D destinado a gerar cenas em perspectiva que permitem navegar e

interagir com os dados geográficos; o ArcGlobe é usado para visualização 3D de conjuntos de dados muito grandes. Ele é baseado em uma exibição dos dados projetados em uma projeção cúbica global, exibidos em diferentes níveis de detalhe e organizados em mosaicos.

2.3.1. Desenvolvimento de ferramentas no ArcGIS

As ferramentas do ArcGIS costumam estar agrupadas em caixas de ferramentas (*toolbox*). Elas executam um código para realizar tarefas em dados geográficos. Junto com a instalação do programa, é incluído um conjunto padrão de ferramentas conhecidas como ferramentas do sistema.

Muitas vezes, profissionais que fazem uso de sistemas de informações geográficas (SIG) necessitam personalizar o comportamento de uma ferramenta do sistema, ampliar suas funções, ou são obrigados a realizar uma tarefa repetidas vezes usando uma ou mais das ferramentas do sistema. Para facilitar isso, o ArcGIS fornece uma plataforma para estender seus recursos através de linguagens de programação / script que possibilitam o acesso a suas funcionalidades. Script é um conjunto de instruções (código de programação) para que uma função seja executada em determinado aplicativo.

O programa usa Python, uma linguagem de programação de código aberto apoiada por uma crescente comunidade de usuários por sua extensa coleção de bibliotecas padrão e de bibliotecas de terceiros. A comunicação entre ArcGIS e Python é feita através de uma biblioteca chamada ArcPy. Esta biblioteca engloba as classes, e funções e módulos necessários para acessar as funcionalidades de geoprocessamento do programa.

A personalização de recursos no ArcGIS pode ser feita a partir da criação de um dos três elementos: *desktop add-in*, *standard toolbox*, e *Python toolbox*. Para tal, um script é escrito usando a linguagem Python e a biblioteca ArcPy. Este script pode ser salvo em um arquivo separado e anexado a uma nova caixa de ferramentas do tipo *standard toolbox*, ou pode ser escrito como parte integral de uma caixa de ferramentas do tipo *Python toolbox*. A *Python toolbox*, objeto do presente estudo, é um arquivo baseado em ASCII e criado inteiramente em Python (MYLEVAGANAM e RAY, 2016; RIGOL-SANCHEZ et al., 2015).

O esqueleto da *Python toolbox* é basicamente uma classe em Python, como pode ser visto na Figura 2.3. Uma *toolbox* pode ter mais de uma ferramenta. Cada ferramenta é definida também por uma classe Python. As ferramentas são associadas à *toolbox* definindo a propriedade *tools* na inicialização de classe dentro do código. Por exemplo, no código mostrado na Figura 2.3 - A, uma ferramenta denominada *CustomBuffer* está associada à caixa de ferramentas *Toolbox*. *label* e *alias* são as propriedades da *Python toolbox*, que são usadas para chama-la em sua execução.

O método de inicialização de classe da ferramenta *CustomBuffer* (Figura 2.3 - B) é usado para definir suas propriedades. A propriedade de *label* será o nome exibido para a ferramenta na interface do ArcGIS e *description* será sua descrição. A propriedade *canRunInBackground* é utilizada para permitir que a ferramenta execute o código em segundo plano.

Em seguida, o método *getParameterInfo()* é utilizado para criar os parâmetros de entrada. Na Figura 2.3 – C, a variável *buffer_distance* é utilizada para definir um valor numérico associado à distância até a qual uma função de *buffer* (a ser definida no código de execução da ferramenta) será executada. O objeto *parameter()* permite definir as propriedades da variável, como o nome exibido (*displayName*), tipo (*datatype*) e se a variável é de entrada ou de saída (*direction*). Este trecho de código contendo o objeto *parameter()* deve ser repetido para outros parâmetros necessários à ferramenta. Ao fim do método, os parâmetros devem ser listados em *params*.

O método *isLicensed()* (Figura 2.3 – D) é utilizado para verificar a existência de licenças necessárias. Um código pode ser adicionado a essa seção para que, caso o usuário não possua alguma licença requerida, uma mensagem seja exibida e a ferramenta não seja executada.

Em *updateParameters* pode-se alterar as propriedades dos parâmetros antes da execução da ferramenta. Ao fornecer dados na interface gráfica da ferramenta, sempre que um parâmetro for alterado, o método *updateParameters* é executado. Ele permite que as propriedades e valores dos parâmetros sejam modificados nesta fase. Na Figura 2.3 – E, o método faz com que o valor de *buffer_distance* seja mudado para zero caso o usuário insira um número negativo.

Em *updateMessages* podem ser definidas mensagens de erro ou advertência para alertar o usuário ou restringir o funcionamento da ferramenta de acordo com os

valores dos parâmetros inseridos na interface. Na Figura 2.3 – F, o método faz com que a mensagem "Valor deve ser menor do que 50" seja exibida caso o usuário preencha o parâmetro *buffer_distance* com valor maior do que 50. A ferramenta não pode ser executada enquanto a mensagem continuar a ser exibida.

Por fim, em *execute()* é definido o código de execução da ferramenta propriamente dito. O exemplo da Figura 2.3 – G define a execução da função *arcpy.Buffer_analysis()*, que corresponde à ferramenta *Buffer* do ArcGIS, tomando *in_feature* e *buffer_distance* como parâmetros de entrada e *out_feature* como saída. A interface da ferramenta pode ser vista na Figura 2.4.

```
import arcpy

class Toolbox(object):                                                    (A)
    def __init__(self):
        self.label = "Toolbox"
        self.alias = "toolbox"

        # List of tool classes associated with this toolbox
        self.tools = [CustomBuffer]

class CustomBuffer(object):                                               (B)
    def __init__(self):
        self.label = "Custom_Buffer"
        self.description = ""
        self.canRunInBackground = False

    def getParameterInfo(self):                                           (C)
        in_feature = arcpy.Parameter(
            displayName="Entrada",
            name="in_feature",
            datatype="Feature Layer",
            parameterType="Required",
            direction="Input")

        buffer_distance = arcpy.Parameter(
            displayName="Dist",
            name="buffer_distance",
            datatype="Double",
            parameterType="Required",
            direction="Input")

        out_feature = arcpy.Parameter(
            displayName="Saida",
            name="out_feature",
            datatype="Feature Class",
            parameterType="Required",
            direction="Output")

        params = [in_feature, buffer_distance, out_feature]
        return params

    def isLicensed(self):                                                 (D)
        return True

    def updateParameters(self, parameters):                               (E)
        buffer_distance = parameters[1]

        if buffer_distance.value < 0:
            buffer_distance.value = 0.0
        return

    def updateMessages(self, parameters):                                 (F)
        buffer_distance = parameters[1]

        if buffer_distance.value > 50:
            buffer_distance.setErrorMessage('Valor deve ser menor do que 50')
        return

    def execute(self, parameters, messages):                              (G)
        in_feature = parameters[0].valueAsText
        buffer_distance = parameters[1].value
        out_feature = parameters[2].valueAsText

        arcpy.Buffer_analysis(in_feature, out_feature, buffer_distance)
        return
```

Figura 2.3 – Exemplo de código de uma *Python toolbox* com: inicialização da *toolbox* e de uma ferramenta (A e B); seus parâmetros (C); licenças (D); validação (E); mensagens (F); e código de execução (G).

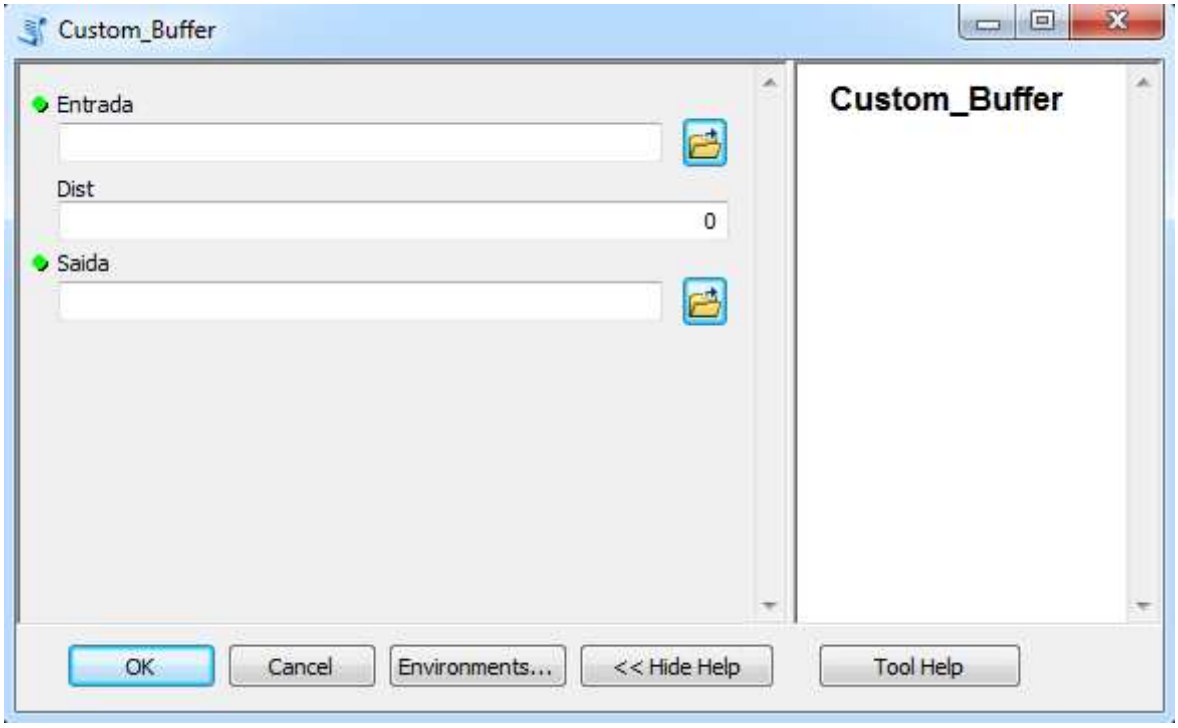


Figura 2.4 – Interface gráfica da ferramenta *Custom_Buffer*, mostrando os campos de preenchimento de *Entrada*, *Dist* e *Saida*.

3. MATERIAL E MÉTODOS

A primeira parte do estudo envolveu a criação e teste de um conjunto de ferramentas com o objetivo de simular povoamentos florestais em nível de árvore individual. Na segunda parte, as ferramentas foram validadas buscando reproduzir dois tipos de plantios de eucalipto encontrados na literatura. Por fim, um estudo econômico utilizando dados gerados pelas ferramentas foi apresentado.

3.1. Ferramentas para simulação em nível de árvore

O ambiente de sistema de informações geográficas escolhido para execução do estudo foi o ArcGIS 10.3, no Windows 7. Por meio da linguagem Python, foi criada no software ArcMap (presente no ArcGIS) uma caixa de ferramentas (*Python toolbox*) com cinco ferramentas voltadas para a simulação. A versão do Python utilizada foi a 2.7.8. O código de programação foi escrito e editado no ambiente de desenvolvimento integrado (IDE) Spyder 2.3.7. As bibliotecas utilizadas no Python foram ArcPy, Random, NumPy e SciPy.

A simulação em nível de árvore individual partiu do princípio visto em Smith e Burkhart (1984), onde os autores, para gerar dados de altura de árvores dominantes e codominantes, aplicaram ao índice de local conhecido uma perturbação seguindo uma distribuição Normal aleatória com média zero e desvio padrão de 0,762. Para o presente estudo, o ponto de partida da caixa de ferramentas para o ArcGIS foi uma ferramenta que permitisse a criação de pontos, cada um representando uma árvore, que teriam valores iniciais de diâmetro a 1,3 m do solo (*dap*) gerados a partir de valores aleatórios dentro de uma distribuição Normal, com média e desvio padrão preestabelecidos. A opção de uso das distribuições Weibull (WEIBULL, 1951) e Gamma, no lugar da distribuição Normal, também foi fornecida.

O segundo ponto foi a criação de uma ferramenta para a projeção dos valores de diâmetro, a partir de modelos em função do *dap* presente, idade presente e idade futura. Foi incluída na projeção do *dap* a opção de se projetar a sobrevivência das

árvores. Por último, foram geradas ferramentas para estimar a altura e o volume das árvores e classificar seu dap.

Além da programação dos cálculos executados pelas ferramentas, o estudo também contemplou o desenvolvimento da interação da interface gráfica com o usuário, ou seja, os campos que recebem os valores de entrada das ferramentas tiveram limitações ou foram programados para sofrerem mudanças de acordo com o tipo de entrada desejada. Por exemplo: um parâmetro que deve ser do tipo *número* não deveria aceitar como entrada um dado do tipo *texto*. No caso de campos referentes a porcentagens, seus valores foram limitados entre 0 e 100. Os campos opcionais não necessários a determinadas operações foram desabilitados e mensagens de erro foram programadas para serem exibidas em todos os casos onde poderiam existir problemas com os dados de entrada, impossibilitando que a ferramenta fosse executada até que o usuário os resolva.

A caixa de ferramentas (*Python toolbox)* foi composta das seguintes ferramentas:

- Criar árvores com dap
- Projetar *dap* e sobrevivência
- Estimar altura
- Estimar volume
- Classificar dap

Na sequência, é detalhada cada uma das funcionalidades da caixa de ferramentas e os meios que utilizam para chegar a seus resultados. Após o desenvolvimento, cada ferramenta foi testada com um exemplo simples para verificação da qualidade de uso da interface gráfica e da qualidade dos resultados gerados.

Depois de geradas as árvores dos testes, a presença de correlação espacial entre seus diâmetros foi testada. Para isso, foi ajustado um modelo de tendência a um semivariograma dos pontos representado as árvores com seus *dap*, por meio da ferramenta *Geostatistical Wizard* do programa ArcMap. Um semivariograma quantifica a suposição de que coisas próximas umas das outras tendem a ser mais semelhantes do que coisas que são mais distantes umas das outras. Ele mede a força da correlação estatística em função da distância (BALAGUER-BESER et al., 2013).

Maiores explicações a respeito do cálculo de semivariogramas e exemplos de sua aplicação na ciência florestal são encontrados nos estudos de Mello (2004), Mello et al. (2016), Kanegae Junior et al. (2007) e Scolforo (2014).

3.1.1. Criar árvores com dap

Esta ferramenta utiliza como entrada uma feição espacial (arquivo que representa dados geográficos no ArcMAP) do tipo polígono, representando a área em que se deseja criar o povoamento, e gera sobre ela uma malha de pontos orientados, de acordo com um valor de espaçamento fornecido. A função utilizada para isso foi a *CreateFishnet*, da biblioteca ArcPy:

```
arcpy.CreateFishnet_management (out_feature_class, origin_coord,
    y_axis_coord, cell_width, cell_height, number_rows,
    number_columns, labels, template)
```

em que *out_feature_class* é a feição de saída do tipo linha, que contém uma malha de células retangulares; *origin_coord* é o ponto de início da malha; *y_axis_coord* é a coordenada usada para orientar a malha, fazendo com que seja girada pelo mesmo ângulo definido pela linha que liga a origem e a coordenada do eixo y; *cell_width* é a largura de cada célula, definida pelo espaçamento; *cell_height* é a altura de cada célula, definida pelo espaçamento; *number_rows* e *number_columns* são, respectivamente, os números de linhas e colunas da malha, calculados a partir da altura e largura das células; *labels* especifica se uma feição espacial do tipo ponto (posicionados no centro de cada célula) será criada; e *template* especifica a extensão que a malha vai ocupar, sendo definida pela extensão do polígono sobre o qual se deseja simular a floresta.

A função cria uma feição do tipo linha (malha de células) e uma do tipo ponto (*labels*). A feição do tipo linha é apagada. Em seguida é realizada uma interseção da feição de pontos com a feição do tipo polígono fornecida como entrada, através da função:

```
arcpy.Intersect_analysis(in_features, out_feature_class,
    {join_attributes})
```

em que *in_features* é a lista das feições para execução da interseção; *out_feature_class* é a feição de saída com o resultado da interseção; e *join_attributes*

determina quais atributos das feições de entrada serão transferidos para a feição de saída.

Como resultado da interseção, os pontos que representam árvores ficam presentes apenas na área de interesse.

Para que os valores de *dap* sejam gerados para cada ponto, o usuário deve selecionar um tipo de distribuição de diâmetros: Normal, Gamma ou Weibull (WEIBULL, 1951). O usuário também deve especificar para qual idade os parâmetros fornecidos irão gerar os diâmetros. Para a distribuição Normal, os parâmetros de média e desvio padrão devem ser fornecidos. Estes parâmetros devem corresponder às características da distribuição de diâmetros do povoamento que se deseja simular, na idade de interesse. Da mesma forma, para a distribuição Gamma, devem ser fornecidos os parâmetros de forma e escala. Para a distribuição Weibull, os parâmetros são locação, escala e forma.

A criação de *dap* foi programada usando o inverso da função de densidade cumulativa das funções *norm*, *gamma* e *weibull_min* da biblioteca ScyPy:

```
def Normal(mean, standard_deviation, p):
    return norm.ppf(p, loc=mean, scale=standard_deviation)

def Weibull(location, scl, form, p):
    return weibull_min.ppf(p, form, loc=location, scale=scl)

def Gamma(scl, form, p):
    return gamma.ppf(p, a=form, scale=scl)
```

em que *mean* e *standard_deviation* são, respectivamente, média e desvio padrão; *location*, *scl* e *form* são, respectivamente, parâmetros de locação, escala e forma; e *p* é a probabilidade aleatória uniforme ($0 \leq p < 1$).

A alocação dos valores de *dap* criados a cada ponto da malha é orientada pelos valores das células de um *raster* de referência. *Raster* é uma imagem contendo uma matriz de células retangulares. Nesta ferramenta, os valores das células desta imagem serviram como orientação para que exista uma correlação espacial entre os *dap* de árvores que se encontram mais próximas umas das outras. O *raster* é criado com base em um conjunto de pontos contendo valores aleatórios de dap. Para a criação deste conjunto, são gerados pontos em locais aleatórios sobre a superfície do polígono que inicialmente orientou a criação da malha de pontos. O número de pontos por hectare e a distância mínima entre eles são definidos pelo usuário. Quanto

maior o número de pontos e menor a distância mínima, maior a aleatoriedade na alocação dos diâmetros. A função utilizada é a *CreateRandomPoints_management*, da biblioteca ArcPy:

```
arcpy.CreateRandomPoints_management(out_path, out_name,
    {constraining_feature_class}, {number_of_points_or_field},
    {minimum_allowed_distance})
```

em que *out_path* é o diretório onde a feição de pontos será salva; *out_name* é o nome dado para a feição de pontos; *constraining_feature_class* é a feição do tipo polígono sobre a qual os pontos aleatórios serão criados; *number_of_points_or_field* é o número de pontos aleatórios criados; e *minimum_allowed_distance* é a distância mínima a ser mantida entre dois pontos. Caso a restrição de distância mínima não possa ser cumprida, pontos adicionais não são criados.

Os pontos aleatórios da função *CreateRandomPoints_management* são unidos a quatro outros, criados nas extremidades da extensão do polígono de superfície da floresta. A eles são atribuídos, ao acaso, valores gerados pela distribuição de diâmetro anteriormente selecionada. Os valores dos pontos são então interpolados para gerar o *raster* através da função *NaturalNeighbor*, da biblioteca ArcPy:

```
arcpy.sa.NaturalNeighbor(in_point_features, z_field, {cell_size})
```

em que *in_point_features* é a feição de pontos de entrada contendo os valores a serem interpolados para um *raster*; *z_field* é o campo que contém os valores de magnitude para serem interpolados; e *cell_size* é o tamanho da célula do *raster*.

Para dar heterogeneidade à alocação das árvores, em um último passo o usuário fornece um índice de 0 a 1 (índice de heterogeneidade) que causa uma perturbação nos valores das células do *raster*. O novo valor de cada célula é um número aleatório, vindo de uma distribuição triangular, com moda igual ao valor original da célula e limites iguais a (1 ± índice) * valor original. O objetivo deste processo é dar heterogeneidade à alocação das árvores.

A partir daí, cada valor de *dap*, gerado inicialmente pela distribuição de diâmetros, é distribuído aos pontos da malha com base no valor da célula do *raster* sobre a qual o ponto se encontra. Quanto menor o valor da célula, menor o *dap* alocado. O *raster* é apagado após o processo de alocação.

Por fim, é determinada a porcentagem de árvores sobreviventes na idade que os *dap* forem gerados. De acordo com o valor fornecido pelo usuário, um número de pontos é selecionado aleatoriamente e seus campos de diâmetro são deixados como vazio (*null*). Caso o arquivo a ser salvo não permita que campos tenham valor nulo, o *dap* é definido como zero.

Além destes resultados, a ferramenta ainda calcula a área de cada parte individual do polígono de entrada, a área total do polígono e o número de árvores por hectare nas situações sem e com mortalidade. Estes dois últimos itens são calculados, respectivamente, pela divisão da contagem do total de pontos pela área total e pela divisão da contagem de pontos com *dap* diferente de zero ou vazio pela área total.

Após sua criação, esta ferramenta foi testada visando gerar um conjunto de árvores sobre uma feição espacial do tipo polígono, com área total de 3,84 ha. O espaçamento para o teste foi de 3,0 x 2,0 m, a idade considerada foi de dois anos, com 2% de mortalidade e diâmetros a 1,3 m (*dap*) seguindo uma distribuição Normal de média 9,8 e desvio padrão igual a 1,4. A distribuição espacial dos *dap* foi orientada com a geração de 130 pontos aleatórios por ha, com 25 m de distância mínima entre pontos e índice de heterogeneidade igual a 0,3.

3.1.2. Projetar *dap* e sobrevivência

Esta ferramenta projeta os valores de diâmetro presentes em um campo de uma feição espacial do tipo ponto (representando árvores), de uma idade inicial para uma idade futura. Ela faz uso de equações às diferenças, obtidas a partir dos modelos de Schumacher (SCHUMACHER, 1939) (eq. 3.1), Lundqvist-Korf (KORF, 1939; LUNDQVIST, 1957) (eq. 3.2) e Richards (RICHARDS, 1959) (eq. 3.3). O usuário deve selecionar uma das três opções para projeção dos diâmetros:

$$dap_2 = dap_1 exp\left(\beta_1 / I_1 - \beta_1 / I_2\right) + \varepsilon \qquad (3.1)$$

$$dap_2 = dap_1 exp\left[\beta_1 / \left(I_1^{\beta_2}\right) - \beta_1 / \left(I_2^{\beta_2}\right)\right] + \varepsilon \qquad (3.2)$$

$$dap_2 = dap_1 \left[1 - \beta_1 exp\left(-\beta_2 I_2\right)\right]^{1/\left(1-\beta_3\right)} / \left[1 - \beta_1 exp\left(-\beta_2 I_1\right)\right]^{1/\left(1-\beta_3\right)} + \varepsilon \qquad (3.3)$$

em que dap_1 é o diâmetro a 1,3 m na idade atual, em cm; dap_2 é o diâmetro a 1,3 m na idade futura, em cm; I_1 é a idade atual, em anos; I_2 é a idade futura, em anos; β_i são os parâmetros dos modelos; e ε é o erro aleatório, sendo ε ~ N (0, σ²).

Após a seleção do modelo, o usuário deve fornecer seus parâmetros para os cálculos. Em sequência, os campos da feição de pontos que possuem os valores iniciais de idade e de *dap* devem ser selecionados. O usuário também deve especificar o tempo (anos) até o fim das projeções e o intervalo de tempo entre cada projeção. Por exemplo, se a idade inicial é de dois anos, o tempo até o fim das projeções é de seis anos e o intervalo entre elas é de dois anos, as projeções são feitas para as idades de quatro, seis e oito anos.

Para que a taxa que projeta o *dap* (calculada pelo modelo) não seja exatamente a mesma para todas as árvores, um índice de heterogeneidade entre 0 e 1 foi inserido na ferramenta. Com base no índice, a taxa de projeção é recalculada para um valor aleatório de uma distribuição Normal com a taxa original como média. Nesta distribuição, o desvio padrão é calculado e os valores da probabilidade aleatória são limitados de modo que o resultado da taxa de projeção seja sempre maior que 1. O índice de heterogeneidade regula o quão distante da média os valores recalculados podem ser. Quanto mais próximo de 1 for o índice, maior a amplitude da variação.

A segunda funcionalidade desta ferramenta é a projeção da sobrevivência. Caso o usuário deseje, ela é mantida em 100%, bastando apenas selecionar o campo contendo o número de árvores por hectare na idade inicial.

A projeção da sobrevivência pode ser feita com base em dois critérios: uso do modelo de Pienaar e Shiver (PIENAAR e SHIVER, 1981) (eq. 3.4) ou porcentagem. No primeiro caso, é aplicado o modelo:

$$N_2 = N_1 exp\left[-\beta_1\left(I_2^{\beta_2} - I_1^{\beta_2}\right)\right] + \varepsilon \qquad (3.4)$$

em que N_1 é o número de árvores por hectare na idade inicial, N_2 é o número de árvores por hectare na idade final, I_1 é a idade atual, em anos; I_2 é a idade futura, em anos; β_i são os parâmetros do modelo; e ε é o erro aleatório, sendo ε ~ N (0, σ²).

No caso de projeção da sobrevivência por porcentagem, é ajustado um modelo logarítmico (eq. 3.5) cujo resultado de N_2 corresponde exatamente à redução pretendida do número de árvores vivas em relação a N_1.

$$N_2 = \beta_0 + \beta_1 ln(I_2) + \varepsilon \quad (3.5)$$

em que N_2 é o número de árvores por hectare na idade final, I_2 é a idade futura, em anos; β_i são os parâmetros do modelo; e ε é o erro aleatório, sendo $\varepsilon \sim N(0, \sigma^2)$.

Caso existam diferenças de idade entre as árvores na idade inicial, são utilizados valores médios para I_1 e I_2.

As árvores que serão assinaladas como mortas são selecionadas ao acaso. Os campos de diâmetro destas árvores são deixados como vazio (*null*). Caso o arquivo a ser salvo não permita que campos tenham valor nulo, o *dap* é definido como zero.

Por fim, a área total ocupada pelas árvores deve ser fornecida para que a ferramenta também calcule o número de árvores vivas por hectare após as projeções.

Para testar esta ferramenta, os diâmetros das árvores geradas pelo teste da ferramenta *Criar árvores com dap* foram projetados para dois anos no futuro, fazendo uso do modelo de Schumacher, com β_1 igual a 0,8854. Em relação à sobrevivência, foi definido que na idade final ocorreria mortalidade de 1% em relação às árvores vivas da idade inicial. O índice de heterogeneidade foi igual a 0,5.

3.1.3. Estimar altura

Esta ferramenta estima a altura das árvores vivas representadas em uma feição espacial do tipo ponto. Ela cria um campo na tabela de atributos para abrigar os valores e faz uso de equações em função do *dap* ou do *dap*Idade*. O usuário deve escolher entre os modelos de Schumacher (SCHUMACHER, 1939) (eq. 3.6), Lundqvist-Korf (KORF, 1939; LUNDQVIST, 1957) (eq. 3.7), Gompertz (GOMPERTZ, 1825) (eq. 3.8) e Weibull (WEIBULL, 1951) (eq. 3.9):

$$Ht = \beta_0 exp\left(-\beta_1 / X\right) + \varepsilon \quad (3.6)$$

$$Ht = \beta_0 exp\left(-\beta_1 / X^{\beta_2}\right) + \varepsilon \quad (3.7)$$

$$Ht = \beta_0 exp\left[-\beta_1 exp\left(-\beta_2 X\right)\right] + \varepsilon \quad (3.8)$$

$$Ht = \beta_0 - \beta_1 exp\left(-\beta_2 X^{\beta_3}\right) + \varepsilon \quad (3.9)$$

em que *Ht* é a altura total da árvore, em m; *X* é a variável independente do modelo, que pode ser *dap* ou *dap*Idade*, em cm e em m; β_i são os parâmetros dos modelos; e ε é o erro aleatório, sendo ε ~ N (0, σ²).

Após a seleção do modelo, o usuário deve fornecer os parâmetros para os cálculos. Em sequência, o usuário deve selecionar os campos de *dap* e idade (ou apenas *dap*, dependendo da variável independente escolhida) para os quais deseja estimar altura. É possível estimar a altura para mais de um campo de *dap* e idade presentes na tabela de atributos.

Assim como na ferramenta anterior, foi inserido um índice de heterogeneidade que varia de 0 a 1 na ferramenta *Estimar altura*. Ele permite que alturas diferentes sejam calculadas para árvores com o mesmo *dap* e idade, mas preservando a média geral das alturas. A função que utiliza o índice recalcula a altura da árvore de modo que ela esteja dentro de uma distribuição Normal com a altura original como a média. A altura mínima dentro da distribuição é limitada a (1 – índice) * altura original. O valor de probabilidade a partir do qual o valor de altura das árvores é retirado de dentro das distribuições, é gerado ao acaso. Todavia, para uma determinada árvore, este valor de probabilidade será sempre o mesmo, para que haja consistência no cálculo das alturas ao longo do tempo. O cálculo deste tipo de número, chamado pseudoaleatório, é feito pela função *random* da biblioteca *Random* para Python. Ela pode tomar como entrada um valor de *seed*. O número pseudoaleatório gerado para uma determinada *seed* é sempre o mesmo. Sendo assim, a ferramenta calcula um valor de *seed* para cada árvore, dado pela soma de suas coordenadas geográficas. Esta *seed* é utilizada para gerar um número pseudoaleatório entre 0 e 1, que será a probabilidade. Na distribuição Normal, o desvio padrão é calculado e os valores da probabilidade pseudoaleatória são restringidos para que os resultados estejam limitados pelo valor de altura mínima.

Esta ferramenta foi testada estimando a altura das árvores resultantes dos testes das ferramentas *Criar árvores com dap* e *Projetar dap e sobrevivência*. Foi utilizado o modelo de Gompertz, com β_0 igual a 29,1179, β_1 igual a 0,8791 e β_2 igual a 0,0182. Foi tomada como variável independente o *dap* multiplicado pela idade. O índice de heterogeneidade foi de 0,2.

3.1.4. Estimar volume

Esta ferramenta estima o volume das árvores vivas representadas em uma feição espacial do tipo ponto. Para tal, o usuário pode escolher entre utilizar um fator de forma ou o modelo de Schumacher e Hall (SCHUMACHER e HALL, 1933) (eq. 3.10):

$$V = exp\left[\beta_0 + \beta_1 ln(dap) + \beta_2 ln(Ht)\right] + \varepsilon \qquad (3.10)$$

em que V é o volume da árvore, em m^3; *dap* é o diâmetro da árvore a 1,3 m, em cm; *Ht* é a altura total da árvore, em m; β_i são os parâmetros dos modelos; e ε é o erro aleatório, sendo $\varepsilon \sim N(0, \sigma^2)$.

Após a seleção de qual método usar para as estimativas, o usuário deve fornecer o fator de forma ou os parâmetros do modelo para os cálculos. Em sequência, caso o usuário tenha optado por utilizar o modelo de Schumacher e Hall, ele deve selecionar os campos de *dap* e *Ht* para os quais deseja estimar o volume. É possível estimar volume para mais de um campo de *dap* e *Ht* presentes na tabela de atributos dos pontos.

Esta ferramenta foi testada visando estimar o volume das árvores resultantes dos testes das ferramentas *Criar árvores com dap*, *Projetar dap e sobrevivência* e *Estimar altura*. Foi utilizado o modelo de Schumacher e Hall (1933) com β_0 igual a -12,4880, β_1 igual a 2,1073 e β_2 igual a 1,6144.

3.1.5. Classificar dap

Esta ferramenta calcula o centro da classe de diâmetro a que os valores de *dap* de um campo da tabela de atributos pertencem. Seus resultados ajudam a quantificar a frequência de árvores por classe de *dap* para estudos de distribuição de diâmetros.

Ela usa como parâmetro de entrada uma feição espacial do tipo ponto, representando árvores. O usuário deve então definir o tamanho do intervalo de classe desejado e, opcionalmente, o valor do *dap* mínimo a ser considerado na criação das

classes. Caso o valor mínimo não seja fornecido, este será igual ao menor *dap* encontrado na tabela de atributos da feição de pontos.

Selecionam-se então os campos contendo os valores de *dap* que se deseja classificar. É possível classificar mais de um campo de *dap* (referentes a diferentes idades) na tabela de atributos.

Esta ferramenta foi testada buscando classificar o diâmetro das árvores resultantes dos testes das ferramentas *Criar árvores com dap* e *Projetar dap e sobrevivência*. O intervalo de classe foi igual a 2,0 cm e o *dap* mínimo igual a 4,0 cm.

3.2. Simulação de dois plantios de eucalipto em dois arranjos espaciais

Os dois plantios simulados foram os presentes em Demolinari (2006) e Lopes (2007). O primeiro, situado no estado do Pará, foi composto de clones não-desbastados de híbridos *Eucalyptus urophylla* x *Eucalyptus grandis* no espaçamento 3,0 x 3,0 m. O segundo, situado na região Noroeste do estado de Minas Gerais, foi composto de clones híbridos não-desbastados de eucalipto em sistema agroflorestal no espaçamento 10,0 x 4,0 m.

Através da caixa de ferramentas desenvolvida no item anterior, buscou-se gerar florestas cujas características de sobrevivência, altura, *dap*, volume e distribuição de diâmetros se aproximassem o máximo possível das características encontradas nos sítios de índice de local médio presentes nos dois estudos. Fez-se uso das ferramentas: *Criar árvores com dap*, *Projetar dap e sobrevivência*, *Estimar altura*, *Estimar volume* e *Classificar dap*.

A área simulada nos dois casos compreendeu uma região de 71,47 ha. A região com arranjo 3,0 x 3,0 m foi simulada dos dois aos seis anos e a com arranjo 10,0 x 4,0 m, dos dois aos 12 anos.

A distribuição para geração dos diâmetros na idade de dois anos foi a de Weibull. Para projeção do *dap*, foram avaliados os modelos de Schumacher (eq. 3.1), Lundqvist-Korf (eq. 3.2) e Richards (eq. 3.3). A sobrevivência foi projetada pelo modelo de Pienaar e Shiver (eq. 3.4). Para estimar a altura foram avaliados os modelos de Schumacher (eq. 3.6), Lundqvist-Korf (eq. 3.7), Gompertz (eq. 3.8) e

Weibull (eq. 3.9). Os volumes foram estimados pelo modelo de Schumacher e Hall (eq. 3.10). Para o espaçamento 3,0 x 3,0 m, o índice de heterogeneidade na projeção de *dap* foi 0,4 e na estimativa de altura foi 0,2. Para o espaçamento 10,0 x 4,0 m, o índice de heterogeneidade na projeção de *dap* foi 0,3 e na estimativa de altura foi 0,1.

Para obtenção dos parâmetros dos modelos utilizados nas ferramentas, um banco de dados foi organizado no programa Microsoft Excel com as tendências ao longo do tempo de diâmetro quadrático médio (q), sobrevivência e média das alturas, a partir dos valores observados nos estudos de Demolinari (2006) e Lopes (2007). Estas tendências representaram o que era esperado como resultado para as duas simulações e são apresentadas nas seções 3.2.1 e 3.2.2. Os modelos foram ajustados a estes dados pela função Solver do Microsoft Excel, buscando minimizar a soma do quadrado dos erros (eq 3.11):

$$SQE = \sum_{i=1}^{n}\left(\hat{Y}_i - Y_i\right)^2 \qquad (3.11)$$

em que SQE é a soma do quadrado dos erros; $\hat{Y}$ é o valor estimado; e Y é o valor observado.

Foram calculados o erro padrão residual das estimativas (S_{yx}) e o coeficiente de determinação ajustado para a média geral (R^2) para todos os ajustes:

$$S_{yx} = \sqrt{SQE/\left(n-k\right)} \qquad (3.12)$$

$$R^2 = 1 - SQE / \sum_{i=1}^{n}\left(Y_i - \bar{Y}_i\right)^2 \qquad (3.13)$$

em que S_{yx} é o erro padrão residual das estimativas, SQE é a soma do quadrado dos erros; n é o número de observações, k é o número de parâmetros do modelo, Y é o valor observado; e $\bar{Y}$ é a média dos valores observados.

O S_{yx} e o R^2 foram utilizados para selecionar quais dos modelos avaliados seriam utilizados nos casos de projeção de *dap* e estimativa de alturas.

Por fim, para avaliar a eficiência das simulações, foram gerados gráficos de distribuição de diâmetros e de tendência ao longo do tempo para *dap*, altura, volume e sobrevivência dos plantios simulados. Para isso, as tabelas de atributos contendo os dados das feições no ArcGIS foram exportadas para o Microsoft Excel. Estes gráficos foram comparados com as tendências esperadas para os dois espaçamentos.

A seguir são apresentados os dados utilizados para estimar os parâmetros dos modelos presentes nas ferramentas empregadas nas simulações.

3.2.1. Espaçamento 3,0 x 3,0 m

Os parâmetros para o modelo de Weibull, utilizado na ferramenta *Criar árvores com dap*, foram obtidos a partir do seu ajuste aos dados da Tabela 3.1. Os valores da tabela corresponderam à distribuição de diâmetros aos dois anos de idade (24 meses), vista para o plantio de eucalipto na classe de produtividade média (S = 26, para idade-índice de 60 meses) em Demolinari (2006).

Para uso das ferramentas *Projetar dap e sobrevivência* e *Estimar altura*, ajustaram-se os modelos disponíveis (eq. 3.1, eq. 3.2, eq. 3.3, eq. 3.4, eq. 3.6, eq. 3.7, eq. 3.8 e eq. 3.9) aos dados da Tabela 3.2. As tendências de sobrevivência, crescimento em diâmetro, altura total e volume com casca foram obtidas a partir dos dados observados de suas médias, na classe de produtividade média. Os modelos para projetar o *dap* foram ajustados aos valores observados de diâmetro quadrático médio (q). Os ajustes foram feitos conforme as relações: $q = f(I)$, $N = f(I)$ e $\overline{Ht} = f(\mathrm{I}, q)$.

Tabela 3.1 – Frequências para ajuste de distribuição de diâmetros para eucalipto plantado no espaçamento de 3,0 x 3,0 m na idade de dois anos.

	Distribuição diamétrica aos dois anos							
Centro de classe de *dap* (cm)	3	5	7	9	11	13	15	17
Frequência	15	34	111	231	308	231	87	33

Fonte: adaptado de Demolinari (2006).

Não houve ajuste do modelo de volume. Os valores de V da Tabela 3.2 foram utilizados apenas como referência do que era esperado em relação ao resultado da simulação. Para uso da ferramenta *Estimar volume*, os parâmetros usados no modelo de Schumacher e Hall corresponderam a uma média dos parâmetros estimados por estrato em Demolinari (2006):

β_0 = -10,44771, β_1 = 1,86852 e β_2 = 1,17904

Tabela 3.2 – Tendência ao longo do tempo do diâmetro quadrático médio (q), sobrevivência (N), média da altura total ($\overline{Ht}$) e volume por hectare (V) das árvores plantadas no espaçamento 3,0 x 3,0 m.

Idade (anos)	q (cm)	N	$\overline{Ht}$ (m)	V (m^3ha^{-1})
2	10,9	1049,5	14,3	64,66
3	12,5	1040,0	17,5	109,55
4	13,7	1021,0	20,1	150,44
5	14,8	997,1	22,1	191,39
6	15,8	981,9	23,7	230,50

Fonte: adaptado de Demolinari (2006).

3.2.2. Espaçamento 10,0 x 4,0 m

No caso do espaçamento 10,0 x 4,0 m, os parâmetros para o modelo de Weibull, utilizado na ferramenta *Criar árvores com dap*, foram obtidos a partir do seu ajuste aos dados da Tabela 3.1. Os valores da tabela corresponderam à distribuição de diâmetros aos dois anos de idade (24 meses), vista para o plantio de eucalipto em sistema agroflorestal na classe de produtividade média (S = 30, para idade-índice de 72 meses) em Lopes (2007).

Tabela 3.3 – Frequências para ajuste de distribuição de diâmetros para eucalipto plantado no espaçamento de 10,0 x 4,0 m na idade de dois anos.

	Distribuição diamétrica aos dois anos								
Centro de classe de *dap* (cm)	7	9	11	13	15	17	19	21	23
Frequência	4	13	27	52	67	50	25	10	2

Fonte: adaptado de Lopes (2007).

Os dados da Tabela 3.4 foram empregados no ajuste dos modelos presentes nas ferramentas *Projetar dap e sobrevivência* e *Estimar altura* (eq. 3.1, eq. 3.2, eq. 3.3, eq. 3.4, eq. 3.6, eq. 3.7, eq. 3.8 e eq. 3.9). As tendências de sobrevivência, crescimento em diâmetro e altura total foram obtidas a partir dos dados observados em Lopes (2007), na classe de produtividade média. Os modelos para projetar o *dap* foram ajustados aos valores de diâmetro quadrático médio (q). Os ajustes foram feitos conforme as relações: $q = f(I)$, $N = f(I)$ e $\overline{Ht} = f(I, q)$.

O diâmetro quadrático médio (q) em cada idade foi obtido a partir das formas e dimensões das curvas de distribuição de diâmetros presentes no estudo.

Os dados de sobrevivência (N) foram criados utilizando os parâmetros para o modelo de Pienaar e Shiver encontrados em Lopes (2007).

Para a média em altura ($\overline{Ht}$), foram calculadas as médias anuais dos valores de uma amostra gerada pelo modelo de altura encontrado no estudo, com S igual a 30, idade variando entre 24 e 144 meses e *dap* variando de 5 a 35 cm. O modelo foi:

$$\overline{Ht} = -48{,}34692+0{,}03119\ (I)+0{,}32832(S)+8{,}81049\ ln(I*dap) \qquad (3.14)$$

em que $\overline{Ht}$ é a altura total, em m; I é a idade, em meses; S é o índice de local, em m; e *dap* é o é o diâmetro a 1,3 m de altura, em cm.

Uma vez que os dados de volume por hectare (V) não estavam disponíveis com o nível de exatidão desejado, seus valores foram obtidos pela média dos valores de produção para índice de sítio médio vistos em Salles (2010), estudo realizado com dados dos mesmos plantios de Lopes (2007).

Tabela 3.4 – Tendência ao longo do tempo do diâmetro quadrático médio (q), sobrevivência (N), média da altura total ($\overline{Ht}$) e volume (V) de eucalipto plantado no espaçamento 10,0 x 4,0 m.

Idade (anos)	q (cm)	N	$\overline{Ht}$ (m)	V (m^3ha^{-1})
2	15,00	250,00	13,92	9,28
3	17,60	249,47	19,08	35,64
4	18,70	249,08	22,66	64,51
5	19,40	248,77	25,43	89,33
6	20,10	248,51	27,73	109,81
7	20,60	248,28	29,70	126,72
8	21,10	248,09	31,43	140,84
9	21,50	247,91	32,98	152,77
10	21,90	247,75	34,42	162,97
11	22,20	247,60	35,74	171,78
12	22,50	247,46	36,96	179,46

Fonte: adaptado de Lopes (2007) e Salles (2010).

Não houve ajuste de modelo de volume. Os valores de *V* da Tabela 3.4 foram utilizados apenas como referência do que era esperado em relação ao resultado da simulação. Para uso da ferramenta *Estimar volume*, os parâmetros usados no modelo de Schumacher e Hall corresponderam a uma média dos parâmetros estimados por estrato em Salles (2010):

β_0 = -10,59472, β_1 = 1,72151 e β_2 = 1,35542

3.3. Análise econômica de um plantio simulado de eucalipto

Foi proposta uma análise econômica comparando dois cenários baseados no plantio de eucalipto simulado no item 3.2.1 (arranjo 3,0 x 3,0 m). A intenção foi demonstrar como os dados gerados pelas ferramentas podem ser usados em estudos de manejo florestal.

Os cenários propostos foram:

1. Corte de toda a plantação aos seis anos.
2. Desbaste de 30% da área basal aos três anos e corte das árvores remanescentes aos seis anos.

Enquanto para o primeiro caso bastou utilizar os dados simulados no item 3.2.1, o cenário incluindo o desbaste envolveu duas tarefas extras: a partir das árvores simuladas até os três anos, definir quais árvores seriam selecionadas para corte nesta idade, correspondendo à intensidade de desbaste desejada; e quais parâmetros utilizar na projeção do *dap* após o desbaste, uma vez que a taxa de crescimento em diâmetro tende a aumentar com a abertura de espaço para as árvores.

Estas duas tarefas foram executadas através de scripts escritos na janela de Python do programa ArcGIS (Figura 3.1).

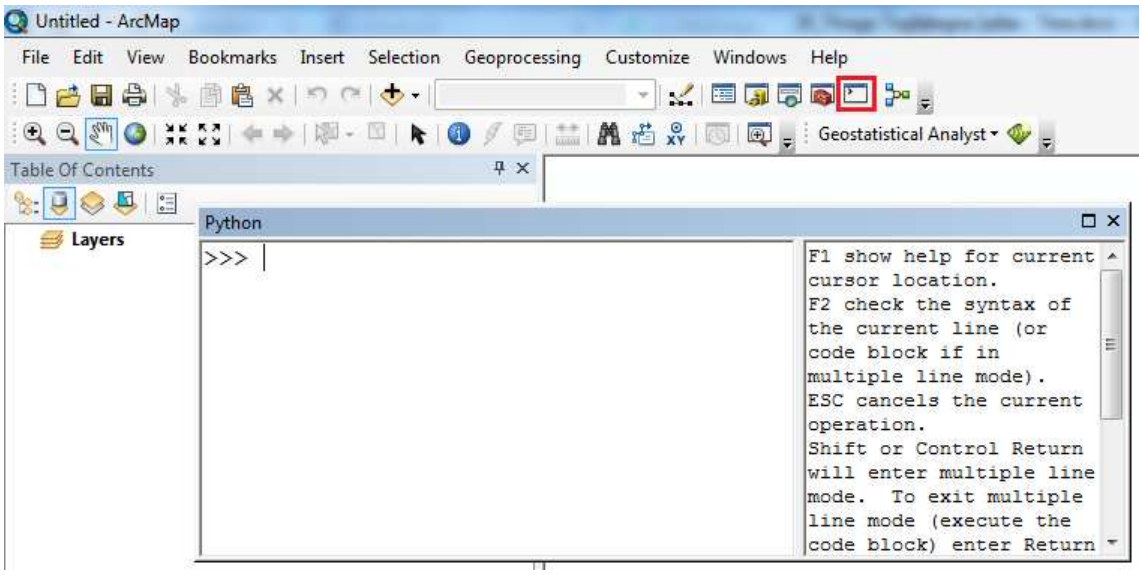


Figura 3.1 – Janela Python do programa ArcGIS, com botão de acesso destacado em vermelho.

3.3.1. Desbaste na simulação

O primeiro passo para simular o desbaste foi duplicar a coluna que contém os valores de *dap* na idade de três anos (*dap_t2*). Para isso, criou-se uma nova coluna na tabela de atributos com o nome *dap_t2_d*, com os valores de suas células iguais aos valores de *dap_t2*. A seleção de árvores para desbaste foi feita sobre a coluna com os dados duplicados.

O script para seleção das árvores iniciou com a importação das três bibliotecas necessárias ao código e a criação de uma variável *arvores* contendo o endereço da feição de pontos que representa as árvores simuladas até a idade de três anos:

```
import arcpy
import numpy as np
import pyperclip

arvores = r'C:\GIS\My_geodata.gdb\my_features\tres_por_tres'
```

O próximo passo foi o cálculo da soma da área seccional a ser removida pelo desbaste de 30%. Primeiramente foi definida a função *area_seccional*, que calcula a área seccional para um determinado valor de *dap*:

```
def area_seccional(dap):
    return (np.pi * dap ** 2) / 40000
```

Depois foi criada uma variável com nome *cursor*, que usa a função *arcpy.da.SearchCursor* para estabelecer um cursor de leitura sobre o campo *dap_t2_d* da feição de pontos armazenada em *arvores*. Este cursor pode ser usado para realizar iterações com os valores das linhas de um campo. A partir dele e da função anteriormente definida, foi criada uma lista (*areas_seccionais*) com os valores das áreas seccionais de todas as árvores vivas em *dap_t2_d*. A redução em área basal a ser feita pelo desbaste (*reducao*) foi igual a 30% da soma dos valores desta lista. O trecho de script correspondente foi:

```
cursor = arcpy.da.SearchCursor(arvores, ['dap_t2_d'])
areas_seccionais = [area_seccional(item[0]) for item in cursor
                    if item[0] is not None]

reducao = sum(areas_seccionais) * 0.3
```

Em sequência, a variável cursor foi redefinida, incluindo também o campo *OID@* que contém o número de identificação de cada árvore (ponto) na tabela de atributos. A partir do cursor, foi criada uma lista (*areas_seccionais_oid*) com os pares de valor de área seccional e número de identificação de cada árvore viva. Esta lista foi organizada em ordem crescente de valor de área seccional, conforme mostra o código:

```
cursor = arcpy.da.SearchCursor(arvores, ['dap_t2_d', 'OID@'])
areas_seccionais_oid = [(area_seccional(item [0]), item [1])
                        for item in cursor if item[0] is not None]

areas_seccionais_oid.sort()
```

O passo seguinte foi criar uma lista com a identificação das árvores cuja soma de áreas seccionais deveria corresponder ao valor de *reducao*. Seu nome foi *lista_desbaste* e, inicialmente, não continha nenhum valor. Uma variável contendo a soma da área seccional das árvores desta lista (*soma_as*) também foi criada. Feito isso, foi definido um laço de repetição sobre os itens de *areas_seccionais_oid* da seguinte forma: para cada item (par de valor de área seccional e número de identificação da árvore) em *areas_seccionais_oid*, caso *soma_as* seja menor que *reducao*, o segundo valor dentro do item (número de identificação da árvore) é adicionado a *lista_desbaste* e o primeiro valor dentro do item (área seccional) é somado a *soma_as*. Quando o valor de *soma_as* alcança o valor de *reducao*, o laço é interrompido, conforme o código:

```
lista_desbaste = []
soma_as = 0.0

for item in areas_seccionais_oid:
    if soma_as < reducao:
        lista_desbaste.append(item[1])
        soma_as += item [0]
    else:
        break
```

O último passo foi transformar a *lista_desbaste* em um texto que fosse copiado para a área de transferência do Windows. Para tal, foi utilizada a função *str*, removendo os caracteres da extremidade da lista (colchetes) e transformando-a em texto corrido. O texto foi copiado através da função *pyperclip.copy*, de acordo com o último trecho do script:

```
arvores_desbastadas = str(lista_desbaste)[1:-1]
pyperclip.copy(arvores_desbastadas)
```

Os números de identificação das árvores a serem cortadas no desbaste (texto copiado) foram usados na função *Select By Attributes...* do ArcGIS para selecioná-las na feição de pontos. Os valores do campo *dap_t2_d* destas árvores foram definidos como nulos e as árvores não-selecionadas foram as remanescentes pós-desbaste.

3.3.2. Tendência de crescimento pós-desbaste

A definição do crescimento após a aplicação do desbaste seguiu o princípio visto em Gorgens et al. (2007), onde os autores observaram que a produção colhida no desbaste, somada à produção do corte final, foi aproximadamente igual à produção de um único corte ao final caso não houvesse desbaste. Logo, o crescimento em *dap* pós-desbaste foi calculado para que o volume total sem desbaste fosse igual ao volume total com desbaste.

O script escrito para se chegar aos valores ideais dos parâmetros de projeção do *dap* pós-desbaste foi iniciado pela importação das bibliotecas necessárias ao código e pela criação de uma variável *arvores* contendo o endereço da feição de pontos que representa as árvores simuladas até a idade de três anos:

```
import arcpy
import numpy as np
from scipy.optimize import fmin

arvores = r'C:\GIS\My_geodata.gdb\my_features\tres_por_tres'
```

O próximo passo foi a definição dos modelos de projeção de *dap* (Richards), estimativa de altura (Weibull) e estimativa de volume (Schumacher e Hall). O código foi o mesmo presente dentro das ferramentas desenvolvidas no item 3.1:

```
def Richards(b1, b2, b3, dap1, i1, i2):
    return (dap1 * (1 - b1 * np.exp(-b2 * i2)) ** (1 / (1 - b3)) /
            (1 - b1 * np.exp(-b2 * i1)) ** (1 / (1 - b3)))

def Weibull(b0, b1, b2, b3, dap1, i1):
    return b0 - b1 * np.exp(-b2 * (dap1 * i1) ** b3)

def S_Hall(b0, b1, b2, dap1, ht):
    return np.exp(b0 + b1 * np.log(dap1) + b2 * np.log(ht))
```

Em sequência, foi criada uma função chamada *dif_volume* que calcula a diferença entre o volume de madeira estimado com os parâmetros fornecidos a ela e um volume de madeira que se deseja atingir, de valor definido pelo usuário. Ela toma como argumento uma lista (*b_R*) contendo três parâmetros para a função de projeção de *dap* (Richards). O objetivo posterior é calcular quais parâmetros esta lista deve conter para minimizar o resultado desta função, ou seja, quais parâmetros fazem com que a diferença entre o volume estimado e o volume desejado seja mais próxima de zero.

O código da função *dif_volume* começa com a listagem dos parâmetros dos modelos de Weibull (*b_W*) e Schumacher e Hall (*b_S_H*). Posteriormente é criado um cursor de leitura sobre o campo *dap_t2_d* (árvores remanescentes após o desbaste) de *arvores*. As listas *dap_t2*, *dap_tf*, *ht_tf* e *v_tf*, irão conter, respectivamente, os valores de *dap* das árvores remanescentes na idade de três anos, e os valores de *dap*, altura e volume das árvores remanescentes na idade de seis anos. O volume por hectare aos seis anos (*v_ha*) é obtido dividindo-se a soma de *v_tf* pela área total da floresta simulada. Por fim, é calculado o módulo da diferença entre o volume de madeira retirado no desbaste aos três anos (*v_desbaste*) somado ao volume colhido aos seis anos (*v_ha*) e o volume total desejado aos seis anos (*v_desejado*).

```
def dif_volume(b_R):
    b_W = [27.79485, 20.78744, 0.02690, 0.90071]
    b_S_H = [-10.44771, 1.86852, 1.17904]
    cursor = arcpy.da.SearchCursor(arvores, ["dap_t2_d"])
    dap_t2 = [item[0] for item in cursor if item is not None]
```

```
dap_tf = [Richards(b_R[0], b_R[1], b_R[2], item, 3.0, 6.0)
          for item in dap_t2]

ht_tf = [Weibull(b_W[0], b_W[1], b_W[2], b_W[3], item, 6.0)
         for item in dap_tf]

v_tf = [S_Hall(b_S_H[0], b_S_H[1], b_S_H[2], item1, item2)
        for item1, item2 in zip(dap_tf, ht_tf)]

v_ha = sum(v_tf) / 71.47
v_desbaste = 29.11
v_desejado = 218.28
dif = np.abs(v_ha + v_desbaste - v_desejado)
return dif
```

Os valores de *v_desbaste* e *v_desejado*, necessários à escrita do script, foram um adiantamento dos resultados da simulação do plantio de eucalipto no espaçamento 3,0 x 3,0 (item 3.2.1) e do desbaste simulado no item 3.3.1.

Para encontrar os parâmetros que minimizam o resultado da função *dif_volume*, foi utilizada a função de otimização *fmin*. Nela, *xtol* é o erro mínimo aceitável para convergência e a lista fornecida contém as suposições iniciais dos parâmetros:

```
print fmin(dif_volume, [0.63327, 0.17115, 0.08032], xtol=0.0000001)
```

Ao se executar a função da forma apresentada acima, precedida de *print*, seu resultado é exibido na tela.

3.3.3. Quantificação do volume de madeira

Com o desbaste simulado e os parâmetros para crescimento em diâmetro pós-desbaste calculados, o volume de madeira obtido em cada colheita pôde ser quantificado nos dois cenários propostos. As tabelas de atributos das feições de pontos geradas para os dois cenários foram exportadas para o Microsoft Excel. Foram então calculados os volumes disponíveis para dois tipos de sortimento:

- Madeira com diâmetro abaixo de 15 cm.
- Madeira com diâmetro acima de 15,0 cm e mínimo 2,20 m de comprimento.

Para isso, foi utilizado o modelo de afilamento do fuste (*taper*) de Garay (1979):

$$d = \beta_0 dap\{1+\beta_1 ln[1-\beta_2(h/Ht)^{\beta_3}]\} + \varepsilon \quad (3.15)$$

$$h = Ht\{exp[1/\beta_1 - d/(\beta_0\beta_1 dap)]/c - 1/c\}^{1/d} + \varepsilon \quad (3.16)$$

em que d é o diâmetro, em cm; dap é diâmetro medido a 1,3 m de altura, em cm; h: é altura, em m; Ht é altura total da árvore, em m; β_i são os parâmetros dos modelos; e ε é o erro aleatório, sendo $\varepsilon \sim N(0, \sigma^2)$.

Os parâmetros utilizados no modelo, vindos de um ajuste prévio a um plantio de eucalipto no estado de Minas Gerais, foram:

$\beta_0 = 2{,}0081$, $\beta_1 = 0{,}1394$, $\beta_2 = 0{,}9996$ e $\beta_3 = 0{,}0096$

As funções de afilamento fornecem a forma da estrutura vertical da árvore. A eq. 3.15 permite estimar qual é o diâmetro (d) em determinada altura da árvore. A eq. 3.16 permite estimar a qual altura (h) ocorre determinado diâmetro na árvore. Dessa forma, é possível estimar o volume de madeira a partir de diferentes alturas e diâmetros intermediários, utilizando a fórmula de Smalian:

$$V = \pi/80.000 * (d_1^2 + d_2^2) * L \quad (3.17)$$

em que V é o volume da tora, em m^3; d_1 e d_2 são os diâmetros nas extremidades da tora, em cm; e L é o comprimento da tora, em m.

Após o cálculo dos volumes por sortimento para cada árvore, foi aplicado um fator de correção a cada uma delas para que os resultados da fórmula de Smalian fossem consistentes com o volume total estimado pelo modelo de Schumacher e Hall (1933).

3.3.4. Receitas, custos e indicadores econômicos

Os custos para as análises foram compostos pelas despesas de implantação e manutenção de florestas de eucalipto (Tabela 1). Os dados são uma adaptação dos custos em Paulino (2012), corrigidos pelo IPC-A para o ano de 2016.

Tabela 3.5 - Custos por hectare com implantação e manutenção de plantações de eucalipto no espaçamento 3,0 x 3,0 m, para o ano de 2016.

Ano	Atividade	Custos (R$ha^{-1})
0	Combate à formiga	106,81
	Roçada	131,26
	Capina química	215,05
	Estradas e aceiros	72,61
	Subsolagem / Fosfatagem	801,53
	Plantio e replantio	795,07
	Adubação de cobertura (NPK)	346,31
	Ferramentas	69,82
	EPI	41,89
	Administração e suporte técnico	139,64
1	Capina	97,72
	Capina química	72,60
	Adubação de cobertura	888,38
	Manutenção de aceiros	16,76
	Combate à formiga	23,74
	Administração e suporte técnico	111,71
2	Adubação de cobertura (KCI)	394,06
	Manutenção de aceiros	16,76
	Combate à formiga	23,74
	Administração, suporte técnico e inventário	20,95
3 em diante	Manutenção de aceiros	83,78
	Combate à formiga	118,68
	Administração, suporte técnico e inventário	104,73

Fonte: adaptado de Paulino (2012).

Para as receitas, em todas as situações foi considerada a venda em pé da madeira. Os preços, obtidos do valor médio em Seapa-MG (2014) e corrigidos também pelo IPCA-A para 2016, foram:

- Madeira com diâmetro abaixo de 15,0 cm: R$ 45,90 m^{-3}.
- Madeira com diâmetro acima de 15,0 cm e mínimo 2,20 m de comprimento: R$ 80,80 m^{-3}.

De posse dos valores da produção de madeira, das receitas e dos custos, foi feito o fluxo de caixa dos dois cenários. A partir do fluxo de caixa, foram calculados os indicadores econômicos valor presente líquido (VPL), valor anual equivalente (VAE) e taxa interna de retorno (TIR), conforme as equações:

$$VPL = \sum R_j(1+i)^{-j} - \sum C_j(1+i)^{-j} \quad (3.18)$$

$$VAE = VPL * i \Big/ \left[1-(1+i)^{-n}\right] \quad (3.19)$$

$$\sum R_j(1+TIR)^{-j} - \sum C_j(1+TIR)^{-j} = 0 \quad (3.20)$$

em que R_j são as receitas no período j, em R$; C_j são os custos no período j, em R$; i é a taxa de desconto em % ao ano; j é o ano de ocorrência; e n é a duração do projeto, em anos.

O VPL representa o lucro atualizado do projeto. O VAE transforma o valor atual do projeto em um fluxo de receitas ou custos periódicos e contínuos, correspondendo no caso da eq. 3.19 ao pagamento anual do VPL. A TIR é a taxa de desconto que iguala o valor atual das receitas ao valor atual dos custos do projeto (REZENDE e OLIVEIRA, 2008).

Souza et al. (2015) mencionam que o investimento deve ser remunerado a uma taxa, no mínimo, igual às cobradas pelos principais financiamentos florestais. As taxas cobradas pelos agentes financeiros, vistas nas linhas de financiamento do Guia de Financiamento Florestal 2016 (SFB, 2016) para produtores rurais, variaram entre 8,0 e 9,5%. Sendo assim, foi adotada uma taxa de desconto igual a 9,0% neste estudo.

4. RESULTADOS

4.1. Ferramentas para simulação em nível de árvore

A caixa de ferramentas (*Python toolbox*) desenvolvida, contendo cinco ferramentas, recebeu o nome *Simular Floresta* (Figura 4.1).

Figura 4.1 – Caixa de ferramentas criada no estudo, como vista no programa ArcMap.

A interface desenvolvida para a ferramenta *Criar árvores com dap* é mostrada na Figura 4.2. Ela apresenta os campos preenchidos de acordo com o exemplo proposto, utilizando como dado de entrada a feição espacial do tipo polígono de nome *talhao* e a feição de saída de nome *tres_por_dois*.

No caso de uso da distribuição Normal, o parâmetro de locação corresponde à média e o parâmetro de escala ao desvio padrão.

A função Weibull pode ser utilizada com dois parâmetros, bastando o usuário deixar o parâmetro de locação igual a zero ou em branco.

Os dizeres *(optional)*, vistos após os nomes dos campos desta e das demais ferramentas desenvolvidas, nem sempre indicam que o referido parâmetro é opcional. Esta inconsistência ocorreu porque foi necessária uma solução alternativa para contornar a limitação de que, na versão 10.3 do ArcGIS, ao se criar uma ferramenta por programação em Python, a definição de seus parâmetros permite que estes sejam apenas do tipo obrigatório ou opcional. Isto cria um problema em

situações em que esta característica é condicional, como por exemplo, para o uso das distribuições de diâmetro, onde o parâmetro de forma era obrigatório para a distribuição Weibull, mas não o era para a distribuição normal. Como solução, os parâmetros que se enquadraram nesta condição foram definidos sempre como *optional* e, dentro da definição das mensagens de erro das ferramentas, foi estabelecido um critério de validação para que uma mensagem de erro apareça quando estes parâmetros passam a ser obrigatórios e são deixados em branco pelo usuário.

A interface contendo os parâmetros de distribuição espacial foi concebida na forma de menu suspenso.

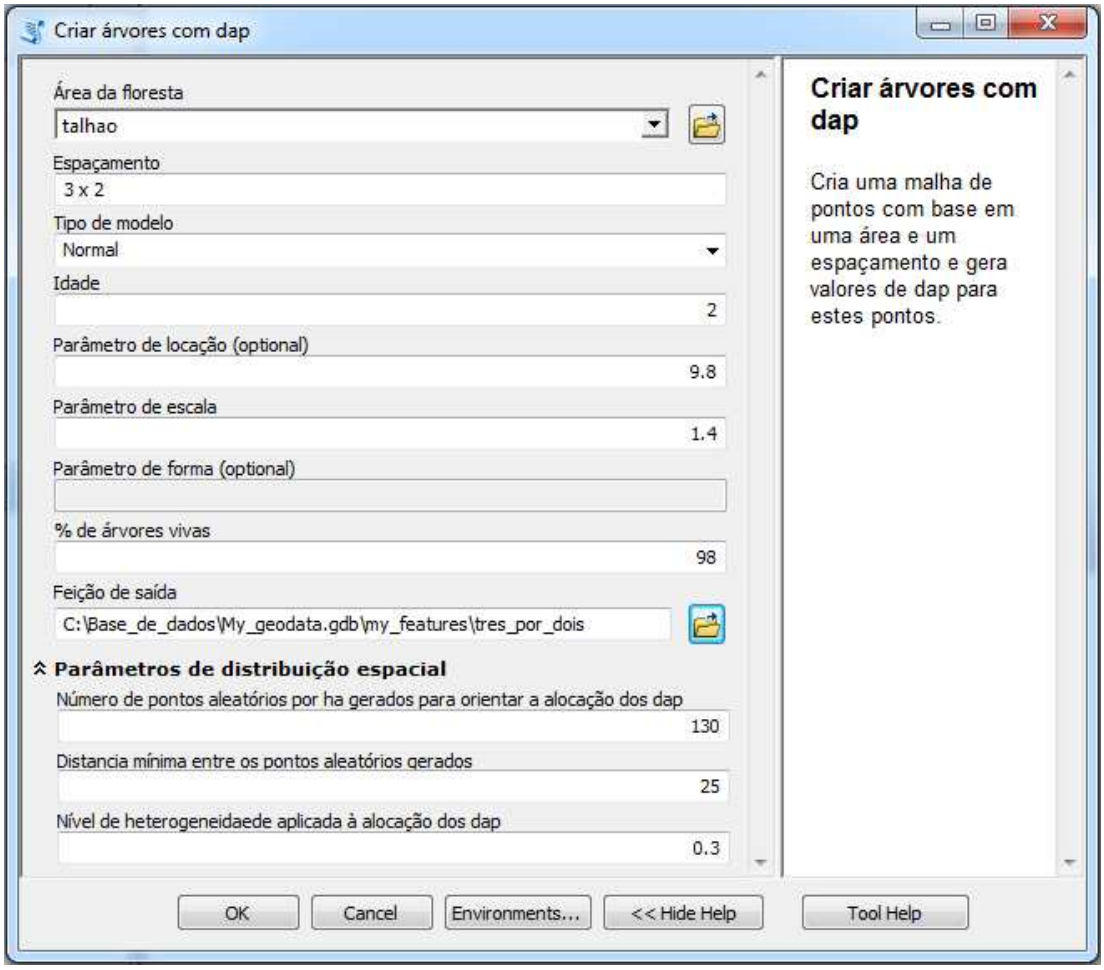


Figura 4.2 – Interface da ferramenta *Criar árvores com dap* com exemplo de preenchimento dos campos.

O campo de espaçamento deve ser preenchido no formato “número x número”. Caso contrário, o usuário recebe a mensagem de erro “Espaçamento deve estar no formato ‘número x número’” e a ferramenta não pode ser executada até que o problema seja resolvido.

Outras validações necessárias para os campos foram:

- Campo *Parâmetro de locação* é obrigatório para a distribuição normal.
- Campo *Parâmetro de escala* é obrigatório para todas as distribuições.
- Campo *Parâmetro de forma* é obrigatório para as distribuições Gamma e Weibull.
- A porcentagem de árvores vivas deve ser um valor entre 0 e 100.

Ao se executar a ferramenta com os campos preenchidos, como na Figura 4.2, foi obtida uma feição espacial do tipo ponto representando as árvores dispostas no espaçamento determinado (Figura 4.3). Atrelada à feição há a tabela de atributos (Figura 4.4) onde podem ser vistos os números de árvores por hectare nas situações sem (*N_t0*) e com mortalidade (*N_t1*), além do *dap* na idade de dois anos (*dap_t1*). Ao se buscar as estatísticas do campo *dap_t1*, obtém-se o histograma com a distribuição dos diâmetros. A expressão *_tn* que segue o nome dos campos é uma referência para situá-los no tempo, um em relação ao outro.

A tabela de atributos da feição de saída ainda contém a descrição do espaçamento das árvores (*Espacam*), a área total da feição de entrada (*Area_total*) e a área individual de cada parte da feição de entrada (*AreaTalhao*), ambas em hectare.

É visto nas estatísticas da Figura 4.4 que, conforme desejado, o número de árvores mortas (*dap = Null*) foi igual a 2% da contagem total (*Count*). A contagem de árvores por hectare também foi correta, uma vez que *N_t0 = Count / Area_total* e *N_dap_t1 =* (*Count - Nulls*) */ Area_total*.

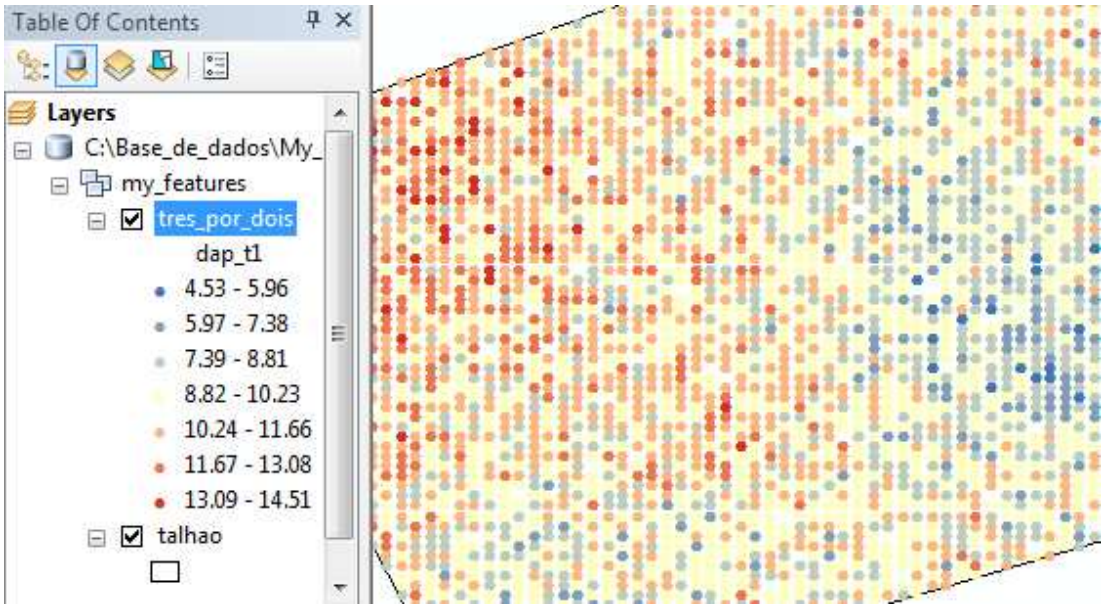


Figura 4.3 – Detalhe de resultado obtido com a ferramenta *Criar árvores com dap*, com os pontos da feição espacial "tres_por_dois" no espaçamento 3 x 2 m.

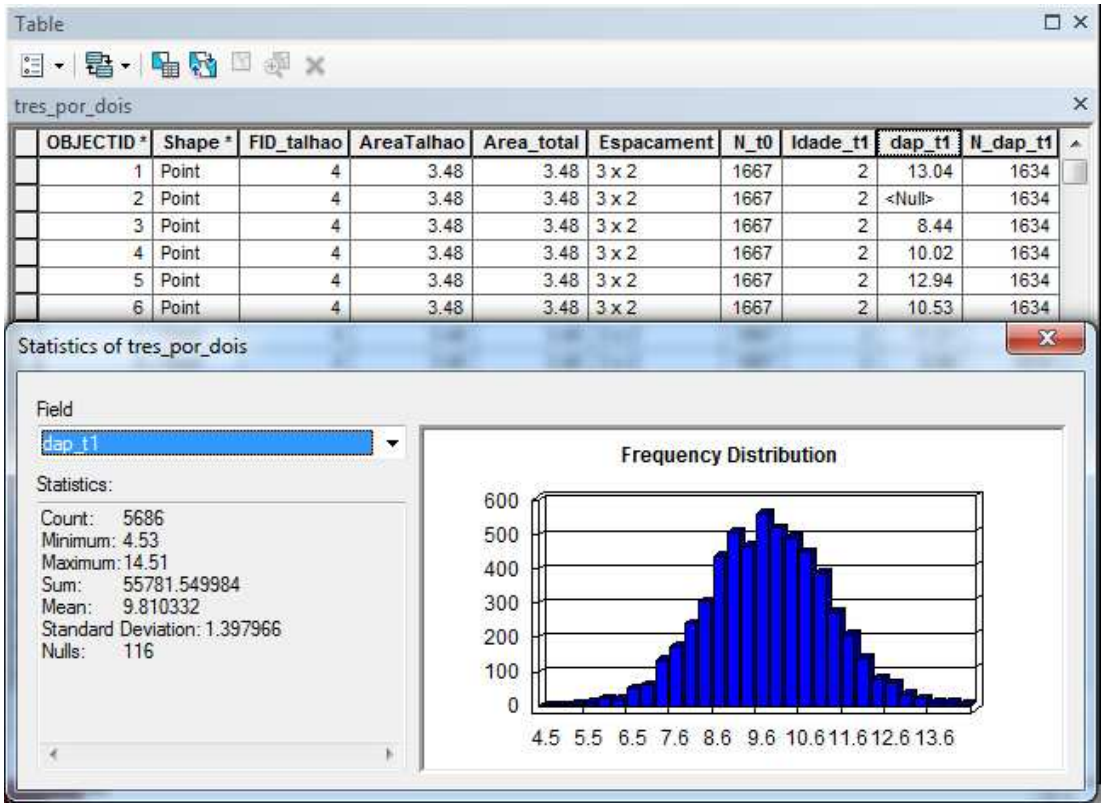


Figura 4.4 – Tabela de atributos da feição espacial *tres_por_dois* e estatísticas do campo *dap_t1*.

A interface da ferramenta *Projetar dap e sobrevivência,* preenchida com os dados de teste, visando projetar os diâmetros do item anterior (feição *tres_por_dois*) é vista na Figura 4.5. Vale mencionar que os valores iniciais de diâmetro e idade vêm dos campos da feição de entrada, enquanto os tempos para a projeção são fornecidos diretamente pelo usuário. O campo *Tempo na primeira projeção* serve para situar as projeções em relação aos valores iniciais de *dap* e idade.

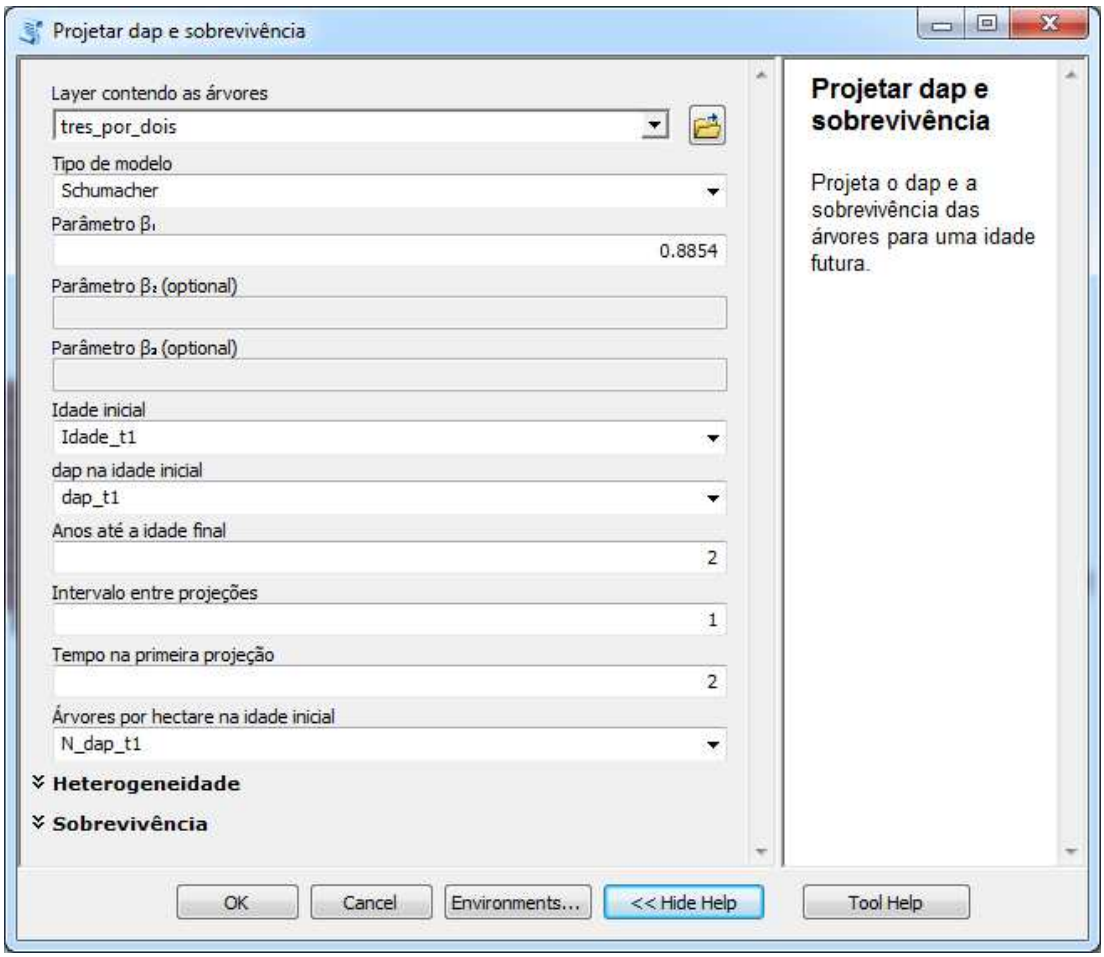


Figura 4.5 – Interface da ferramenta *Projetar dap e sobrevivência* com exemplo de preenchimento dos campos.

Na seção de projeção do *dap*, foi necessária a seguinte validação para os campos:

- Campo *Parâmetro*β_2 é obrigatório para os modelos de Lundqvist_Korf, e Richards.

- Campo *Parâmetro*β_3 é obrigatório para o modelo de Richards.
- Os demais campos são todos obrigatórios em qualquer caso.

As seções de heterogeneidade e de projeção da sobrevivência (Figura 4.6) foram concebidas na forma de menu suspenso. Se no lugar da mortalidade de 1% (correspondente ao teste da ferramenta) fosse desejado manter a sobrevivência em 100%, bastaria ignorar esta seção, uma vez que os valores padrão para *Tipo de sobrevivência* e *Porcentagem de sobrevivência na idade final* são iguais a *Porcentagem* e *100*, respectivamente.

O campo de *Área total* torna-se obrigatório caso seja definido algum nível de mortalidade. Marcando *Field*, ele pode ser preenchido com um campo da tabela de atributos da feição de entrada contendo a área total na primeira célula. Marcando *Double*, ele pode ser preenchido com um valor de área em hectare inserido diretamente no campo.

Caso o *Tipo de sobrevivência* fosse definido como *Modelo de Pienaar e Shiver*, os campos correspondentes aos seus parâmetros teriam se tornado obrigatórios.

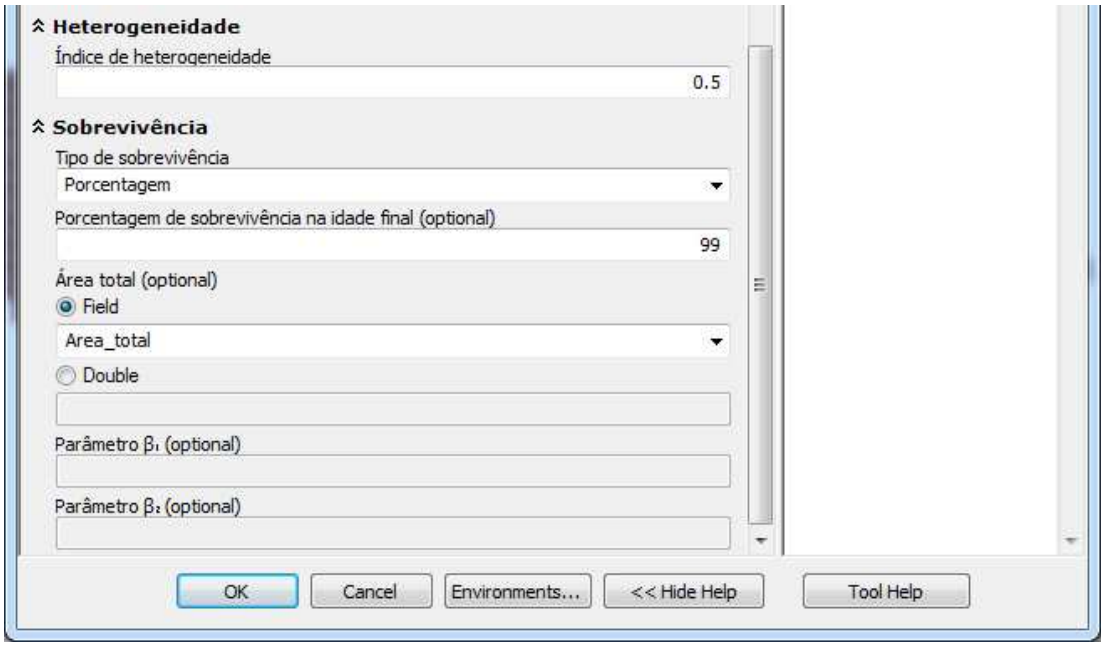


Figura 4.6 – Menus *Heterogenidade* e *Sobrevivência* da ferramenta *Projetar dap e sobrevivência* com exemplo de preenchimento dos campos.

Na Figura 4.7 são vistos os resultados das projeções. O número de árvores por hectare na idade final (*N_dap_t3*) correspondeu à diminuição pretendida em relação a *N_dap_t1*. Consistentemente, ocorreu crescimento gradual dos diâmetros ao longo do tempo e a forma de sino da distribuição dos valores de *dap* na idade final foi preservada.

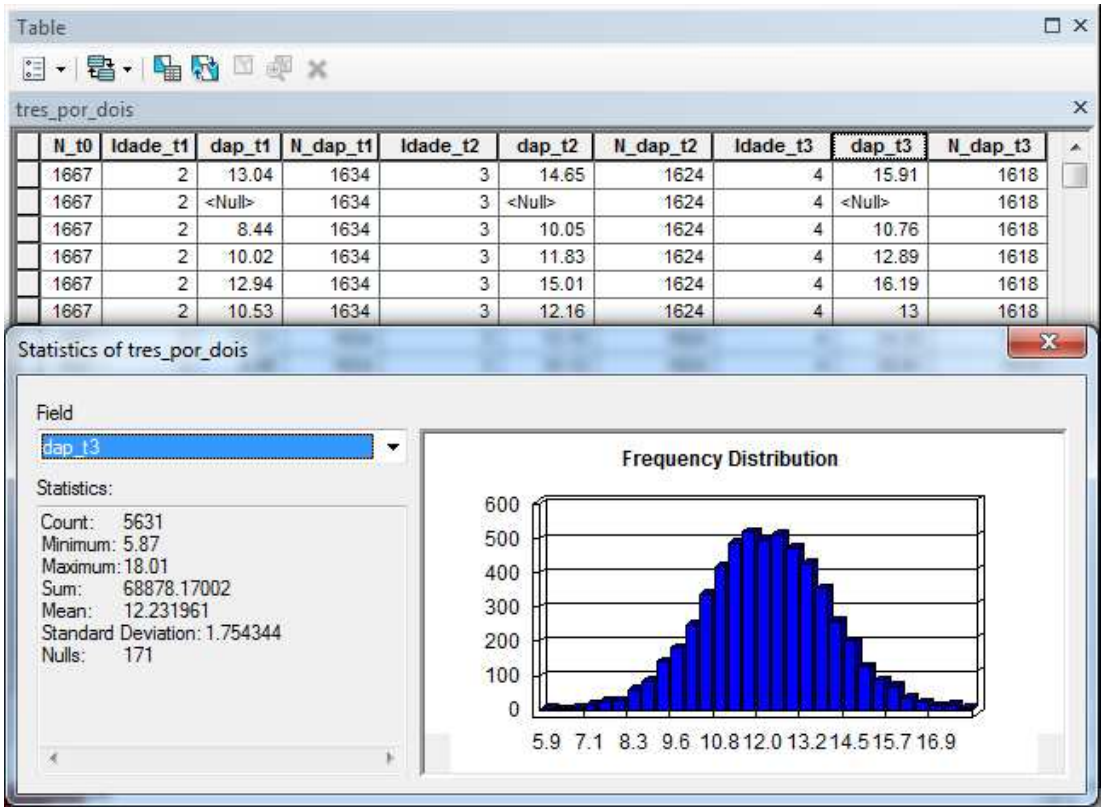


Figura 4.7 – Tabela de atributos da feição espacial *tres_por_dois* com os resultados da projeção de diâmetros e estatísticas do campo *dap_t3*.

Caso existissem diferenças entre as idades das árvores, por exemplo, se a floresta fosse composta por indivíduos de dois e três anos, a ferramenta realizaria as projeções individualmente. As árvores de dois anos teriam quatro e as de três teriam cinco anos na idade final.

A interface desenvolvida para a ferramenta *Estimar altura* é mostrada na Figura 4.8. Ela apresenta o exemplo visando estimar a altura das árvores do item anterior (feição *tres_por_dois*) nas três idades disponíveis. A seção de heterogeneidade foi concebida como menu suspenso.

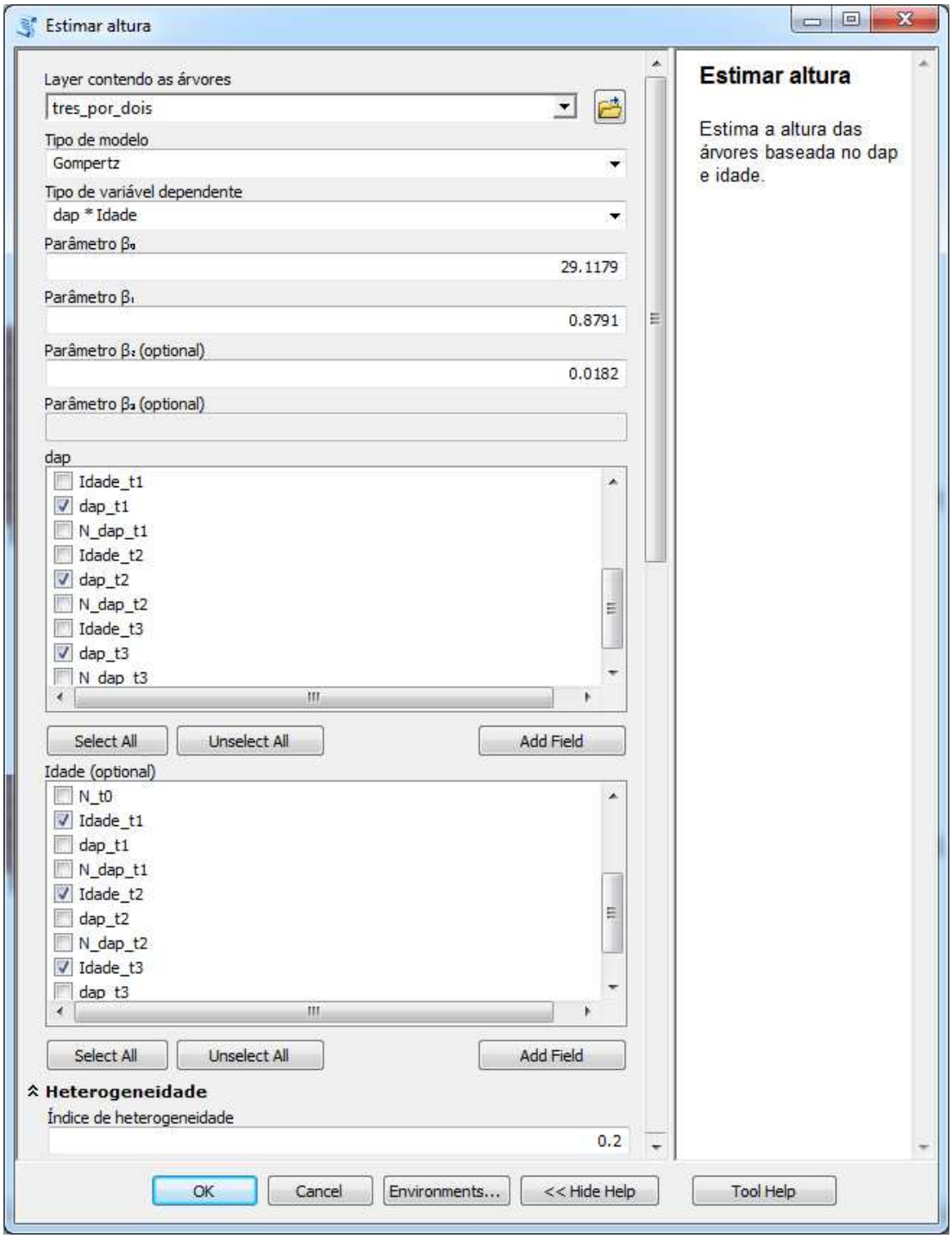


Figura 4.8 – Interface da ferramenta *Estimar altura* com exemplo de preenchimento dos campos.

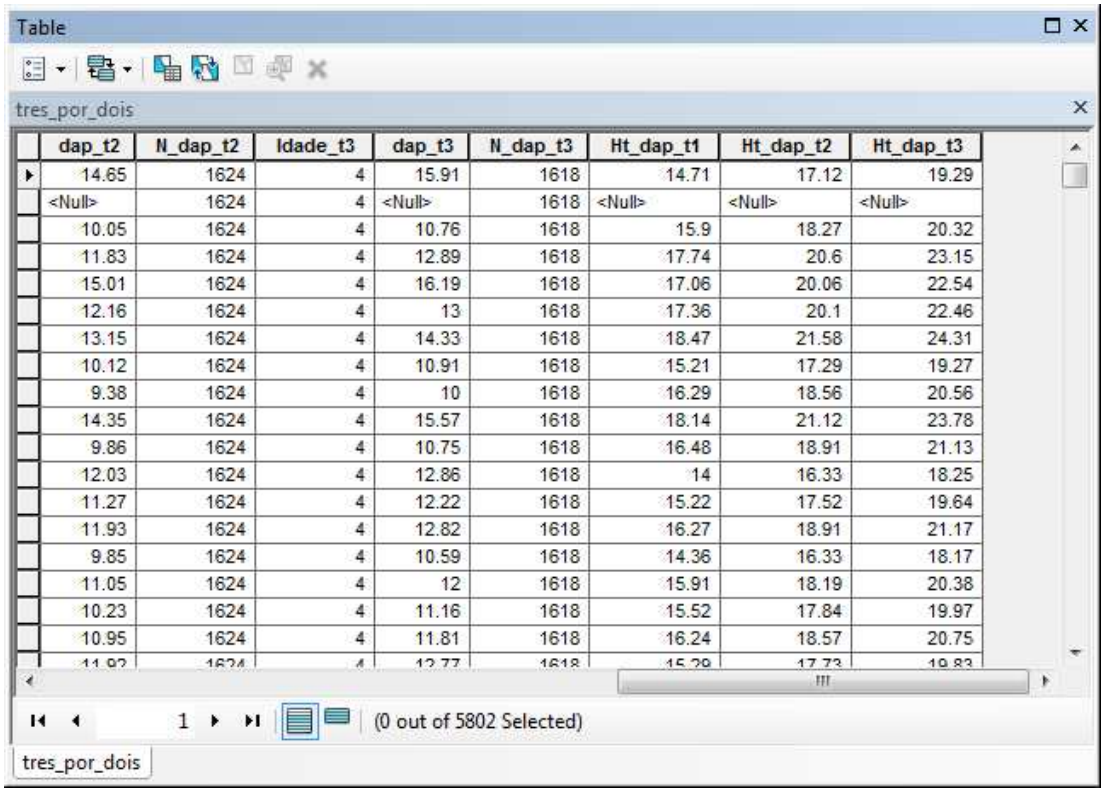
Caso a variável independente tivesse sido apenas o *dap*, a seleção de campos de idade seria desabilitada. No exemplo da Figura 4.8, se o número de campos de *dap* selecionados fosse diferente do número de campos de idade selecionados, a

ferramenta mostraria a mensagem de erro "Número de campos de *dap* e idade deve ser o mesmo", impossibilitando sua execução até que o problema fosse resolvido.

Outras validações necessárias aos campos foram:

- A seleção dos campos de *dap* é obrigatória.
- Campos *Parâmetro* β_0 e *Parâmetro* β_1 são obrigatórios.
- Campo *Parâmetro* β_2 é obrigatório para os modelos de Weibull, Gompertz e Lundqvist_Korf, .
- Campo *Parâmetro* β_3 é obrigatório para o modelo de Weibull.

Os resultados das estimativas de altura (*Ht_dap_t1*, *Ht_dap_t2* e *Ht_dap_t3*) ficaram agrupados ao final da tabela de atributos apresentada na Figura 4.9. Sua tendência de aumento acompanhou a tendência vista para os diâmetros. Para as árvores mortas, os valores de altura também foram deixados como vazios.



dap_t2	N_dap_t2	Idade_t3	dap_t3	N_dap_t3	Ht_dap_t1	Ht_dap_t2	Ht_dap_t3
14.65	1624	4	15.91	1618	14.71	17.12	19.29
<Null>	1624	4	<Null>	1618	<Null>	<Null>	<Null>
10.05	1624	4	10.76	1618	15.9	18.27	20.32
11.83	1624	4	12.89	1618	17.74	20.6	23.15
15.01	1624	4	16.19	1618	17.06	20.06	22.54
12.16	1624	4	13	1618	17.36	20.1	22.46
13.15	1624	4	14.33	1618	18.47	21.58	24.31
10.12	1624	4	10.91	1618	15.21	17.29	19.27
9.38	1624	4	10	1618	16.29	18.56	20.56
14.35	1624	4	15.57	1618	18.14	21.12	23.78
9.86	1624	4	10.75	1618	16.48	18.91	21.13
12.03	1624	4	12.86	1618	14	16.33	18.25
11.27	1624	4	12.22	1618	15.22	17.52	19.64
11.93	1624	4	12.82	1618	16.27	18.91	21.17
9.85	1624	4	10.59	1618	14.36	16.33	18.17
11.05	1624	4	12	1618	15.91	18.19	20.38
10.23	1624	4	11.16	1618	15.52	17.84	19.97
10.95	1624	4	11.81	1618	16.24	18.57	20.75

Figura 4.9 – Tabela de atributos da feição espacial *tres_por_dois* com os resultados das estimativas de altura para três idades.

A interface com o exemplo de teste da ferramenta *Estimar volume* é apresentada na Figura 4.10.

Análogo ao que acontece com a ferramenta *Estimar altura*, caso o número de campos de *dap* selecionados seja diferente do número de campos de altura, a ferramenta mostra a mensagem de erro "Número de campos de *dap* e altura deve ser o mesmo", impossibilitando sua execução até que o problema seja resolvido.

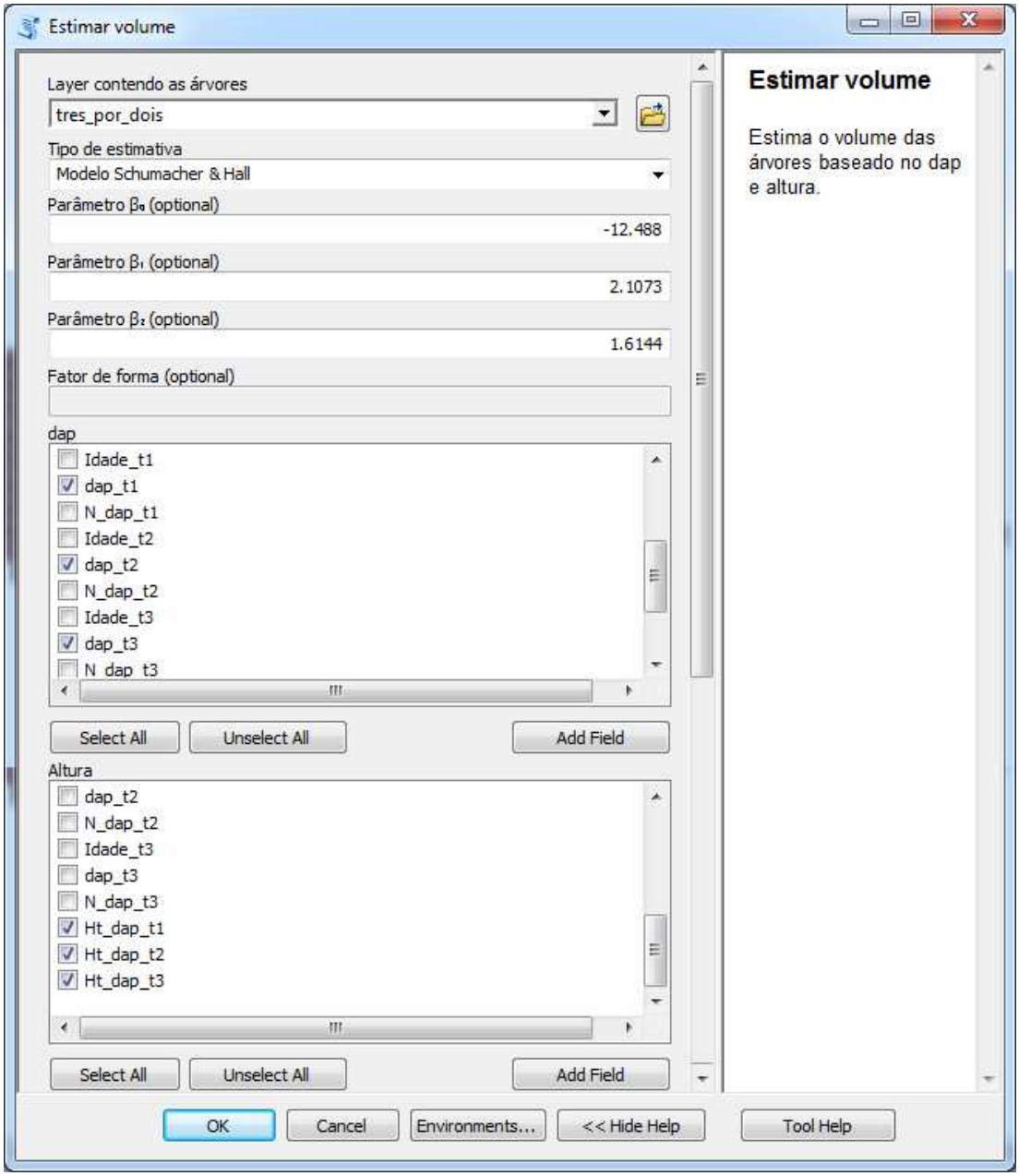
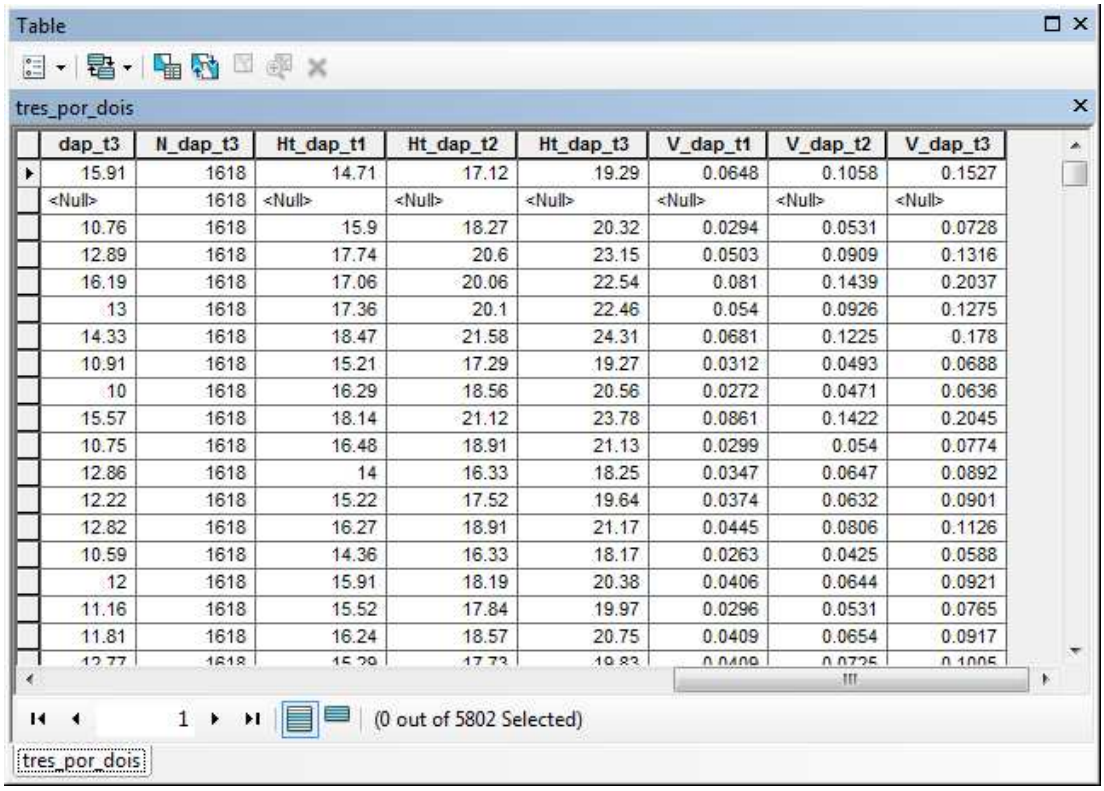


Figura 4.10 – Interface da ferramenta *Estimar volume* com exemplo de preenchimento dos campos.

Em relação à validação dos parâmetros, se *Tipo de estimativa* for igual a *Modelo Schumacher e Hall*, os campos dos parâmetros β_0, β_1 e β_2 tornam-se obrigatórios. Caso *Tipo de estimativa* seja igual a *Fator de forma*, os campos mencionados são desabilitados e o campo *Fator de forma* é habilitado e se torna obrigatório. Os campos de seleção de *dap* e altura são sempre obrigatórios.

Os valores de volume (*V_dap_t1*, *V_dap_t2* e *V_dap_t3*) ficaram agrupados ao final da tabela de atributos (Figura 4.11). Sua tendência de aumento acompanhou a tendência vista para *dap* e *Ht*. Para as árvores mortas os valores de volume também foram deixados como vazios.



dap_t3	N_dap_t3	Ht_dap_t1	Ht_dap_t2	Ht_dap_t3	V_dap_t1	V_dap_t2	V_dap_t3
15.91	1618	14.71	17.12	19.29	0.0648	0.1058	0.1527
<Null>	1618	<Null>	<Null>	<Null>	<Null>	<Null>	<Null>
10.76	1618	15.9	18.27	20.32	0.0294	0.0531	0.0728
12.89	1618	17.74	20.6	23.15	0.0503	0.0909	0.1316
16.19	1618	17.06	20.06	22.54	0.081	0.1439	0.2037
13	1618	17.36	20.1	22.46	0.054	0.0926	0.1275
14.33	1618	18.47	21.58	24.31	0.0681	0.1225	0.178
10.91	1618	15.21	17.29	19.27	0.0312	0.0493	0.0688
10	1618	16.29	18.56	20.56	0.0272	0.0471	0.0636
15.57	1618	18.14	21.12	23.78	0.0861	0.1422	0.2045
10.75	1618	16.48	18.91	21.13	0.0299	0.054	0.0774
12.86	1618	14	16.33	18.25	0.0347	0.0647	0.0892
12.22	1618	15.22	17.52	19.64	0.0374	0.0632	0.0901
12.82	1618	16.27	18.91	21.17	0.0445	0.0806	0.1126
10.59	1618	14.36	16.33	18.17	0.0263	0.0425	0.0588
12	1618	15.91	18.19	20.38	0.0406	0.0644	0.0921
11.16	1618	15.52	17.84	19.97	0.0296	0.0531	0.0765
11.81	1618	16.24	18.57	20.75	0.0409	0.0654	0.0917

Figura 4.11 – Tabela de atributos da feição espacial *tres_por_dois* com os resultados das estimativas de volume para três idades (*V_dap_t1*, *V_dap_t2*, *V_dap_t3*).

A interface desenvolvida para a ferramenta *Classificar dap* é vista na Figura 4.12, com a feição *tres_por_dois* e seus campos de *dap* selecionados para teste. Os

valores utilizados de diâmetro mínimo e intervalo de classe são fornecidos por padrão ao usuário. O campo *Intervalo de classe* e o campo de seleção de *dap* são obrigatórios.

Assim como nos itens anteriores, os campos contendo os centros de classe de diâmetro (*dap_t1_C*, *dap_t2_C* e *dap_t3_C*) ficaram agrupados ao final da tabela de atributos (Figura 4.13). Para as árvores mortas, os valores de classe de *dap* também foram deixados como vazios.

Se comparados os valores de *dap_t1*, *dap_t2* e *dap_t3* da Figura 4.7 com os valores de *dap_t1_C*, *dap_t2_C* e *dap_t3_C* da Figura 4.13, vê-se que a classificação dos diâmetros foi feita corretamente.

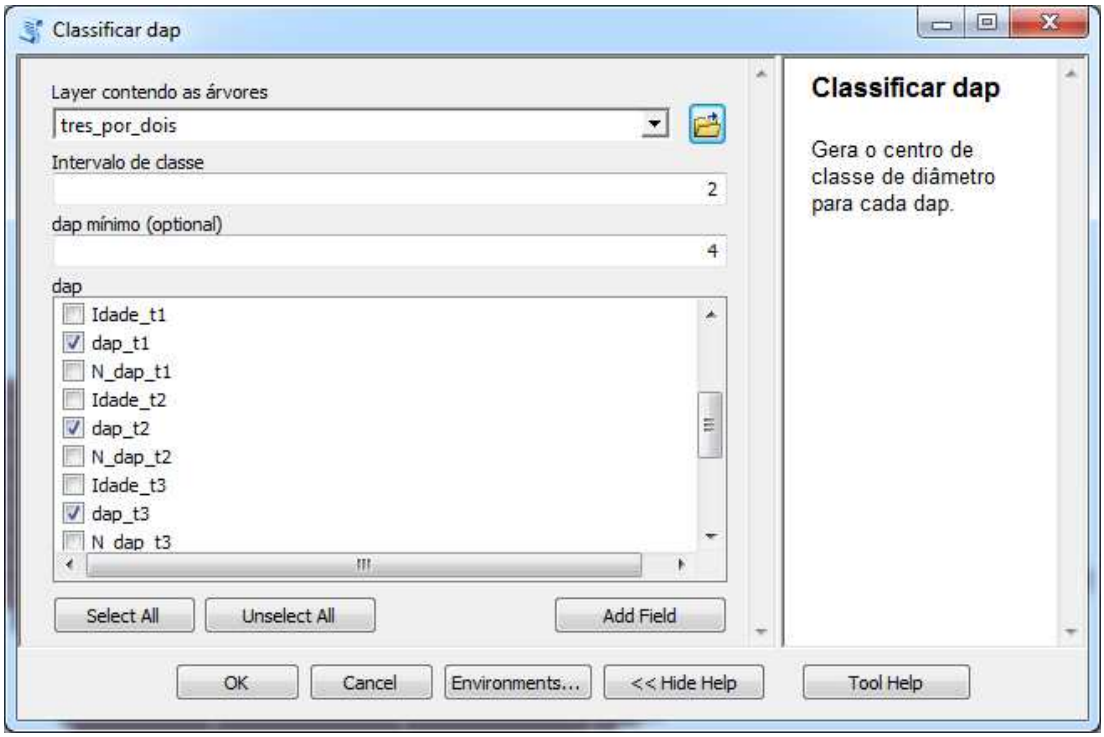


Figura 4.12 – Interface da ferramenta *Classificar dap* com exemplo de preenchimento dos campos.

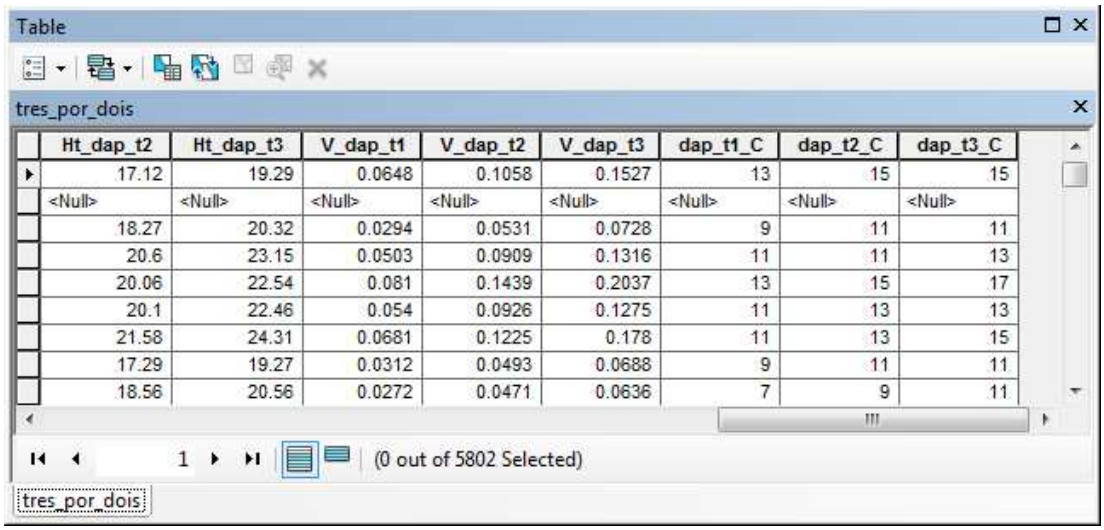


Figura 4.13 – Tabela de atributos da feição espacial *tres_por_dois* com os resultados da classificação de *dap* para três idades (*dap_t1_C*, *dap_t2_C*, *dap_t3_C*).

4.1.1. Correlação espacial entre as árvores

O semivariograma da Figura 4.5 apresenta, para os valores de *dap_t1* (*dap* na idade de dois anos), as semivariâncias (diferenças entre os valores de pontos situados a uma determinada distância) versus as distâncias correspondentes, bem como a curva do modelo ajustado a estas semivariâncias. Uma vez que a média das semivariâncias teve início baixo e tendeu a subir com o aumento da distância, com posterior estabilização, e que o modelo de tendência teve bom ajuste, passando junto aos pontos de média, confirmou-se a existência de correlação entre o diâmetro das árvores mais próximas umas das outras.

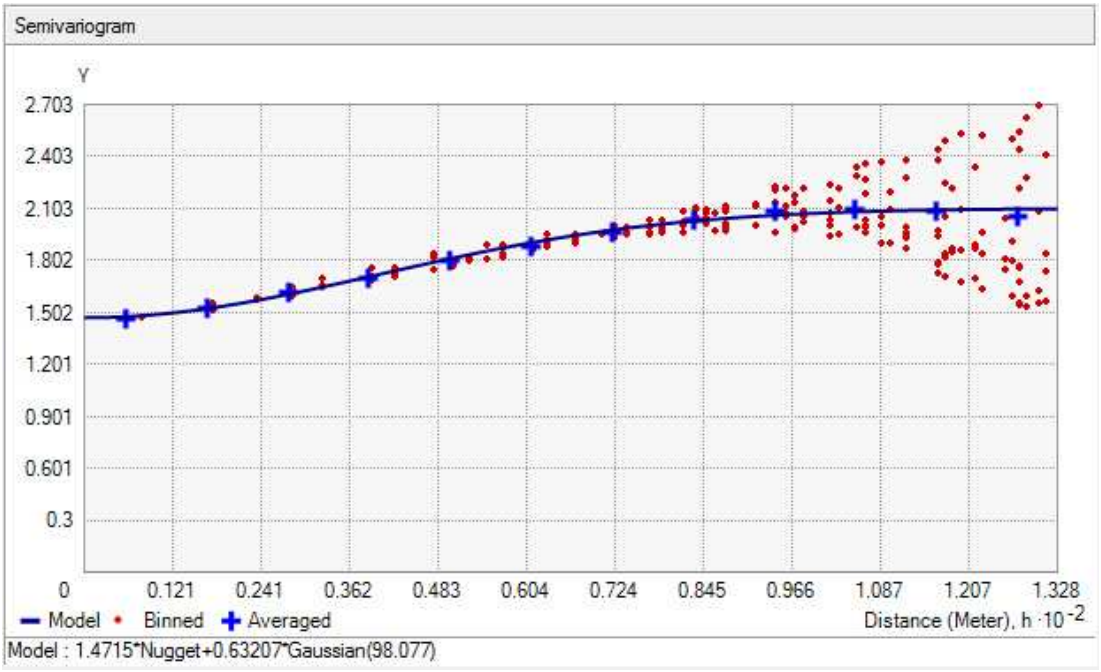


Figura 4.14 - Semivariograma, gerado pelo programa ArcMap, dos *dap* das árvores simuladas na idade de dois anos, contendo os valores das semivariâncias (*Binned*), suas médias (*Averaged*) e o modelo ajustado (*Model*).

O modelo de tendência que melhor se ajustou aos dados foi o Gaussiano. O efeito pepita (*nugget*), que é a variação aleatória que pode ser atribuída a erros de medição ou fontes espaciais de variação em distâncias menores do que o intervalo de amostragem, foi igual a 1,47. O alcance (range), que é o limite de distância em que as variações são explicadas pela correlação espacial, foi igual a 98,08 m. A contribuição ou patamar (partial sill), que é a quantidade de variação explicada pelo modelo, foi igual a 0,63.

4.2. Simulação de dois plantios de eucalipto em dois arranjos espaciais

Foram produzidas duas feições do tipo ponto (representado as árvores), uma para cada simulação, com seus respectivos valores de diâmetro, número de árvores por hectare, altura e volume ao longo das idades presentes em suas tabelas de atributos.

As Figuras 4.15 e 4.16 mostram uma visualização do aspecto geral das áreas simuladas nos espaçamentos 3,0 x 3,0 m e 10 x 4,0 m respectivamente. Essa representação dos resultados pode ser obtida no software ArcScene (parte do ArcGIS), personalizando a exibição dos pontos das feições como figuras de árvores, dimensionadas pelo campo de altura.

As distribuições Weibull ajustadas aos dois plantios estimaram bem as tendências desejadas, apresentando alto coeficiente de determinação (R^2) e baixo erro padrão residual (S_{yx}) (Tabela 4.1). Os ajustes dos demais modelos também seguiram esta tendência em relação ao R^2 e ao S_{yx}.

Figura 4.15 – Aspecto geral do resultado da simulação do plantio de eucalipto aos dois anos em espaçamento 3,0 x 3,0 m, visto no software ArcScene.

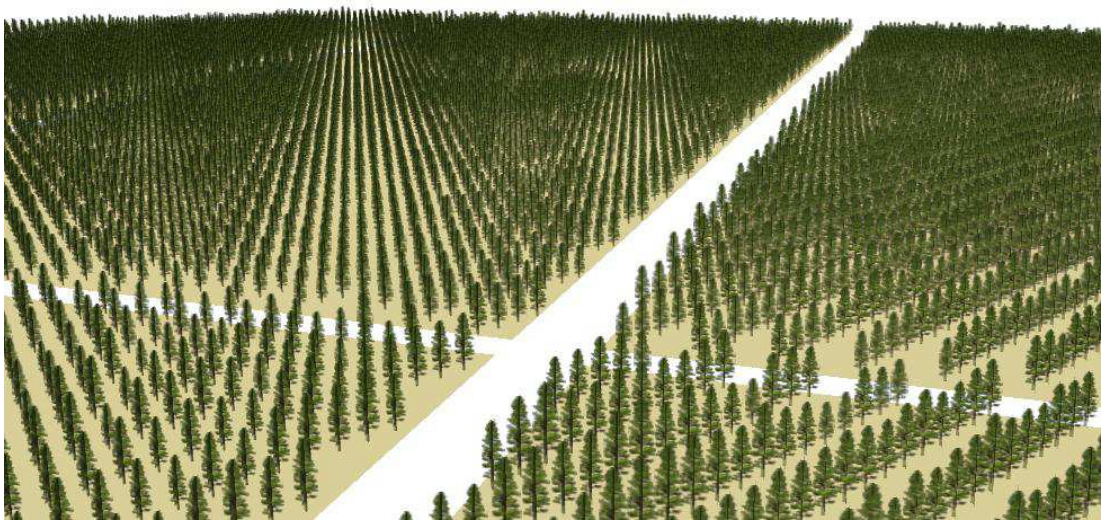


Figura 4.16 – Aspecto geral do resultado da simulação do plantio de eucalipto aos dois anos em sistema agroflorestal no espaçamento 10,0 x 4,0 m, visto no software ArcScene.

Para projeção do *dap*, o modelo de Richards (eq. 3.3) teve melhor ajuste aos dados do espaçamento 3,0 x 3,0 m, enquanto o modelo de Lundqvist-Korf (eq. 3.2) foi melhor para o espaçamento 10,0 x 4,0 m (Tabela 4.2). Também houve êxito nos ajustes do modelo de Pienaar e Shiver (eq. 3.4) para projeção da sobrevivência (Tabela 4.3). Para estimativa de altura, o modelo de Weibull (eq. 3.9) teve melhor ajuste aos dados do espaçamento 3,0 x 3,0 m, enquanto o modelo de Lundqvist-Korf (eq. 3.7) foi melhor para o espaçamento 10,0 x 4,0 m (Tabela 4.4).

Tabela 4.1 – Estimativas dos parâmetros da distribuição Weibull ajustada às distribuições de diâmetros dos dois espaçamentos, coeficiente de determinação (R^2) e erro padrão residual (S_{yx}) dos ajustes.

Plantio	Parâmetro de locação	Parâmetro de escala	Parâmetro de forma	R^2 (%)	S_{yx}
Demolinari (2006) 3,0 x 3,0 m	2,33630	9,53052	3,50449	99,90	3,05
Lopes (2007) 10,0 x 4,0 m	4,36646	11,6598	4,18232	99,72	1,15

Tabela 4.2 – Estimativas dos parâmetros dos modelos de projeção de *dap* ajustados aos dois espaçamentos, coeficiente de determinação (R^2) e erro padrão residual (S_{yx}) dos ajustes.

Plantio	Modelo	$\hat{\beta}_1$	$\hat{\beta}_2$	$\hat{\beta}_3$	R^2 (%)	S_{yx}
Demolinari (2006) 3,0 x 3,0 m	Schumacher	1,02449	-	-	91,00	0,43
	Lundqvist-Korf	1,59886	0,29478	-	98,79	0,19
	Richards	**0,63327**	**0,17115**	**0,08032**	**99,87**	**0,09**
Lopes (2007) 10,0 x 4,0 m	Schumacher	0,92552	-	-	96,88	0,30
	Lundqvist-Korf	**0,90619**	**0,69347**	**-**	**99,54**	**0,12**
	Richards	0,95572	0,24027	-1,11793	98,80	0,20

Tabela 4.3 – Estimativas dos parâmetros do modelo de Pienaar e Shiver para projeção da sobrevivência, ajustado aos dois espaçamentos.

Plantio	$\hat{\beta}_1$	$\hat{\beta}_2$	R^2 (%)	S_{yx}
Demolinari (2006) 3,0 x 3,0 m	0,00382	1,70542	98,26	4,15
Lopes (2007) 10,0 x 4,0 m	0,03691	0,12596	100,00	0,00

Tabela 4.4 – Estimativas dos parâmetros dos modelos para estimativa de altura ajustados aos dois espaçamentos, coeficiente de determinação (R^2) e erro padrão residual (S_{yx}) dos ajustes.

Plantio	Modelo	$\hat{\beta}_0$	$\hat{\beta}_1$	$\hat{\beta}_2$	$\hat{\beta}_3$	R^2 (%)	S_{yx}
Demolinari (2006) 3,0 x 3,0 m	Schumacher	26,94195	14,71541	-	-	97,36	0,70
	L.-Korf	78,92554	3,58624	0,24027	-	99,97	0,09
	Gompertz	25,90906	1,03557	0,02564	-	99,98	0,08
	Weibull	**27,79485**	**20,78744**	**0,02690**	**0,90071**	**99,99**	**0,04**
Lopes (2007) 10,0 x 4,0 m	Schumacher	40,33199	38,95293	-	-	95,73	1,80
	L.-Korf	**104,7813**	**5,583754**	**0,29946**	**-**	**99,99**	**0,07**
	Gompertz	38,26547	1,359135	0,01197	-	99,45	0,65
	Weibull	60,41623	73,52455	0,11071	0,41567	99,99	0,11

Com os modelos e parâmetros definidos, a simulação para o plantio de eucalipto no espaçamento 3,0 x 3,0 m seguiu o seguinte fluxo de ferramentas: *Criar árvores com dap* (distribuição Weibull) → *Projetar dap e sobrevivência* (modelo de Richards; modelo de Pienaar e Shiver) → *Estimar altura* (modelo Weibull) →

Estimar volume (modelo de Schumacher e Hall, com parâmetros definidos em material e métodos) → *Classificar dap*. Para o espaçamento 10,0 x 4,0 m, o fluxo foi: *Criar árvores com dap* (distribuição Weibull) → *Projetar dap e sobrevivência* (modelo de Lundqvist-Korf; modelo de Pienaar e Shiver) → *Estimar altura* (modelo de Lundqvist-Korf) → *Estimar volume* (modelo de Schumacher e Hall, com parâmetros definidos em material e métodos) → *Classificar dap*. Estes passos resultaram nas feições de pontos (exibidos como árvores) das Figuras 4.15 e 4.16. Vinculadas a estas feições há as tabelas de atributos, que contêm os dados de área total, espaçamento e número de árvores por hectare, *dap*, altura, volume e classe de diâmetro das árvores ao longo dos anos.

Exportar as tabelas de atributos das feições para planilhas do programa Microsoft Excel permitiu totalizar o diâmetro quadrático médio (q), número de árvores por hectare (N), média das alturas ($\overline{Ht}$), e volume de madeira com casca por hectare (V) (Figuras 4.17 e 4.18).

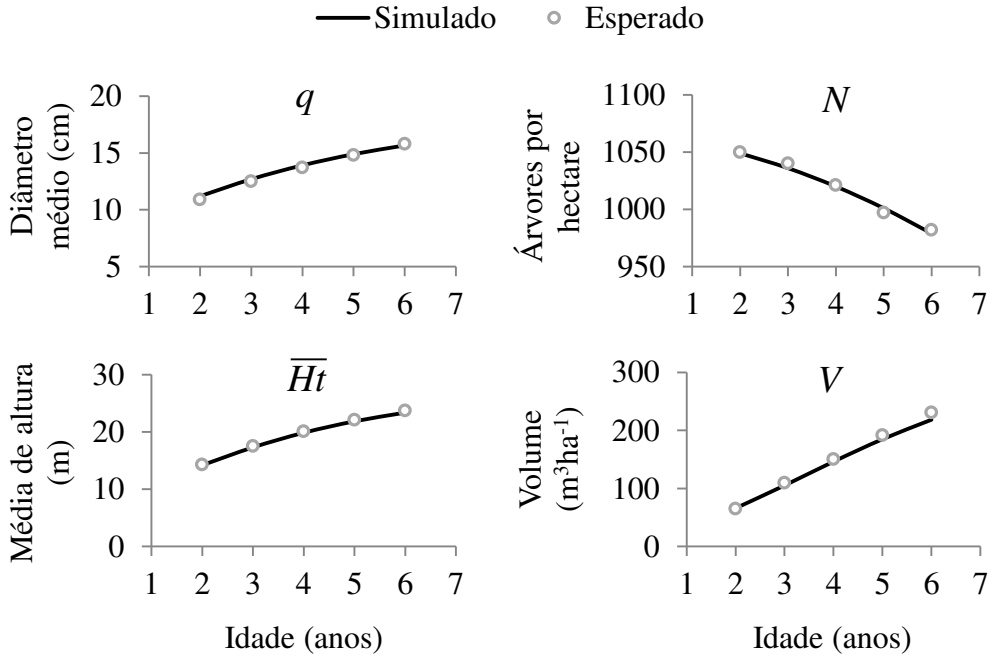


Figura 4.17 – Valores esperados e curvas simuladas de diâmetro quadrático médio (q), árvores por hectare (N), Média das alturas ($\overline{Ht}$) e volume (V) para plantio de eucalipto no espaçamento 3,0 x 3,0 m.

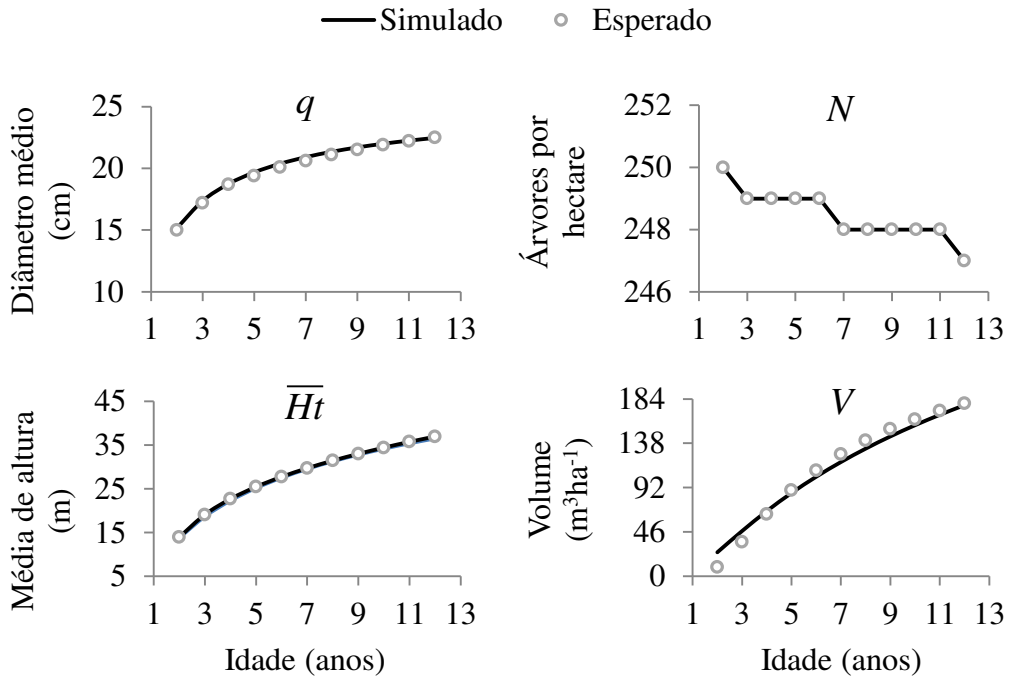


Figura 4.18 – Valores esperados e curvas simuladas de diâmetro quadrático médio (q), árvores por hectare (N), Média das alturas ($\overline{Ht}$) e volume (V) para plantio de eucalipto no espaçamento 10,0 x 4,0 m.

Os pontos com legenda *Esperado* nas figuras Figuras 4.17 e 4.18 correspondem aos valores presentes nas Tabelas 3.2 e 3.4. O fato das curvas de q, N, $\overline{Ht}$, e V terem passado sempre próximas a estes pontos indica que houve boa exatidão na simulação destas variáveis.

As planilhas também permitiram totalizar a frequência de árvores por classe de *dap* ao longo do tempo (Figura 4.19). As distribuições de diâmetros na idade de dois anos corresponderam às árvores criadas a partir da distribuição Weibull, enquanto as demais curvas corresponderam aos diâmetros projetados nas idades seguintes.

Também houve boa exatidão em relação às distribuições de diâmetro. As classes de *dap* vistas nas diferentes idades dos plantios em Demolinari (2006) e Lopes (2007) (Figura 4.20), quando não foram as mesmas, foram muito próximas àquelas dos plantios simulados.

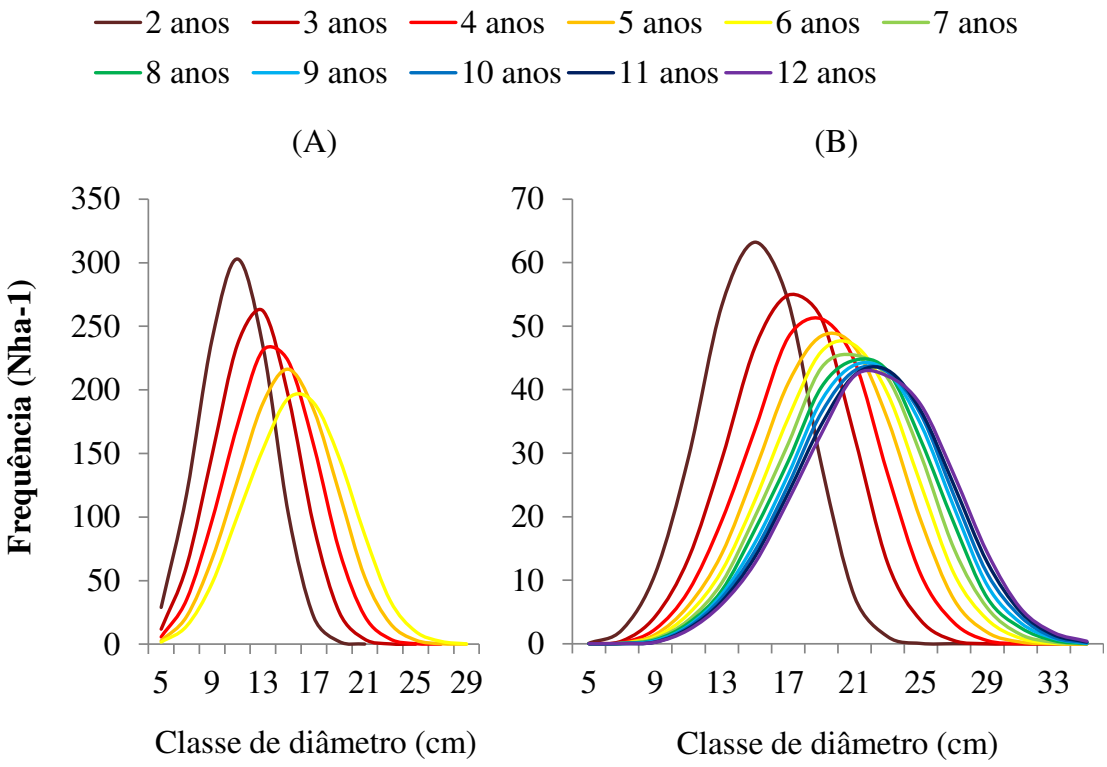
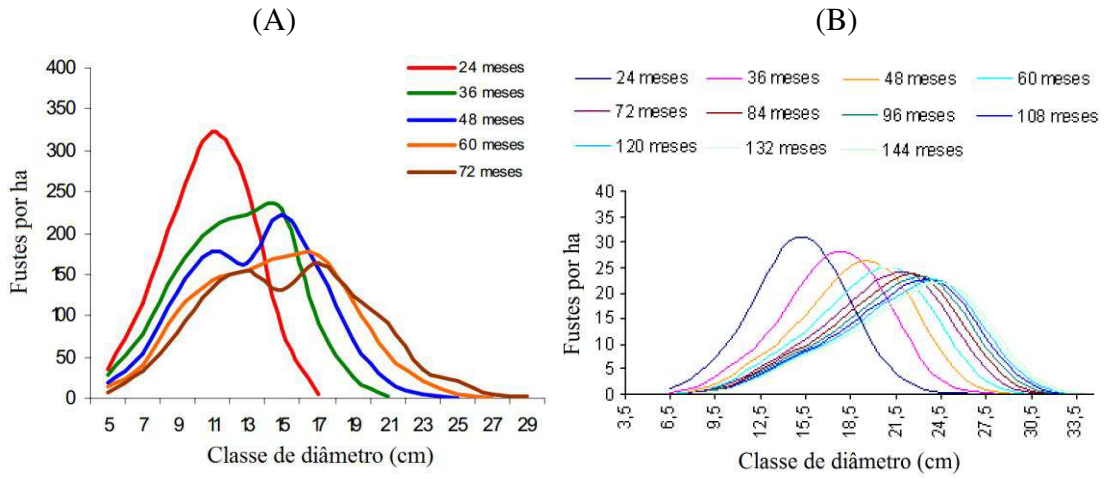


Figura 4.19 – Distribuições de diâmetros para as simulações de eucalipto simulado no espaçamento 3,0 x 3,0 (A) e em sistema agroflorestal no espaçamento 10,0 x 4,0 m (B).



Fonte: adaptado de Demolinari (2006) e Lopes (2007).

Figura 4.20 – Distribuições de diâmetros de eucalipto plantado no espaçamento 3,0 x 3,0 (A) e em sistema agroflorestal no espaçamento 10,0 x 4,0 m (B).

4.3. Análise econômica de um plantio simulado de eucalipto

O script de desbaste selecionou, de um total de 74.105 árvores, 35.623 para corte na idade de três anos. A menor classe de diâmetro encontrada entre as árvores remanescentes foi de 13 cm.

Os coeficientes do modelo de Richards (eq. 3.3) para projeção do *dap* após o desbaste, calculados pelo script de minimização da diferença entre os volumes estimado e desejado, foram: $\beta_1 = 0{,}73977$, $\beta_2 = 0{,}16137$ e $\beta_3 = 0{,}08273$.

As curvas de produção de madeira nos dois cenários são apresentadas na Figura 4.21. O volume a ser colhido no cenário 1, que previa o corte de toda a floresta aos seis anos, foi de 218,28 m^3ha^{-1}. O volume a ser colhido no desbaste do cenário 2 foi de 29,11 m^3ha^{-1} e no corte final de 189,07 m^3ha^{-1}, totalizando 218,18 m^3ha^{-1}. Estes valores de produção corresponderam à soma dos volumes por sortimento, estimados pelo modelo de *taper* e fórmula de Smalian (Tabela 4.5).

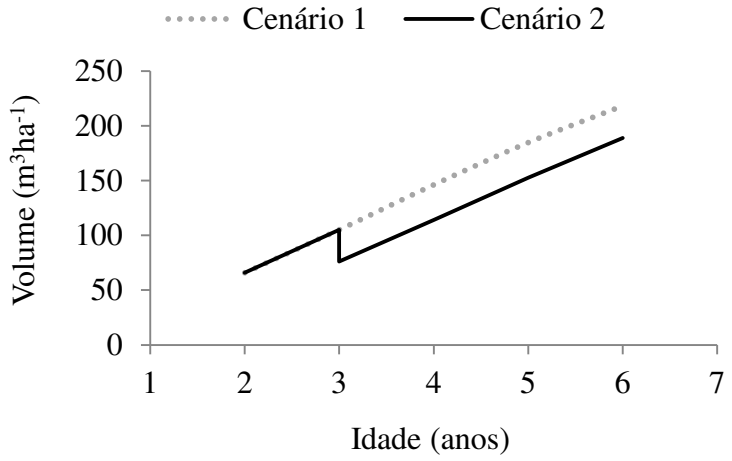


Figura 4.21 – Curvas de produção de madeira para o plantio de eucalipto simulado nos cenários sem desbaste (Cenário 1) e com desbaste (Cenário 2).

A madeira de valor mais alto (toretes acima de 15 cm) foi encontrada em maior quantidade para o corte final no cenário 2, como visto na Tabela 4.5 e na Figura 4.22. Este resultado refletiu no fluxo de caixa (Tabela 4.6), onde a receita do cenário 2 no sexto ano foi aproximadamente igual à receita do cenário 1, mesmo

havendo uma diferença de 15,45% entre os volumes a serem colhidos nesta idade. Com isso, a receita total do cenário 2, que inclui a venda de madeira do desbaste no ano 3, foi 9,88% maior do que a receita total do cenário 1.

Tabela 4.5 – Produção de volume por sortimento, no desbaste e no corte final, para os dois cenários do plantio de eucalipto simulado.

Cenário	Sortimento	Volume (m^3ha^{-1})	
		Desbaste	Final
1	Diâmetro < 15 cm	-	133,14
	Diâmetro > 15 cm	-	85,14
2	Diâmetro < 15 cm	29,11	67,01
	Diâmetro > 15 cm	-	122,06

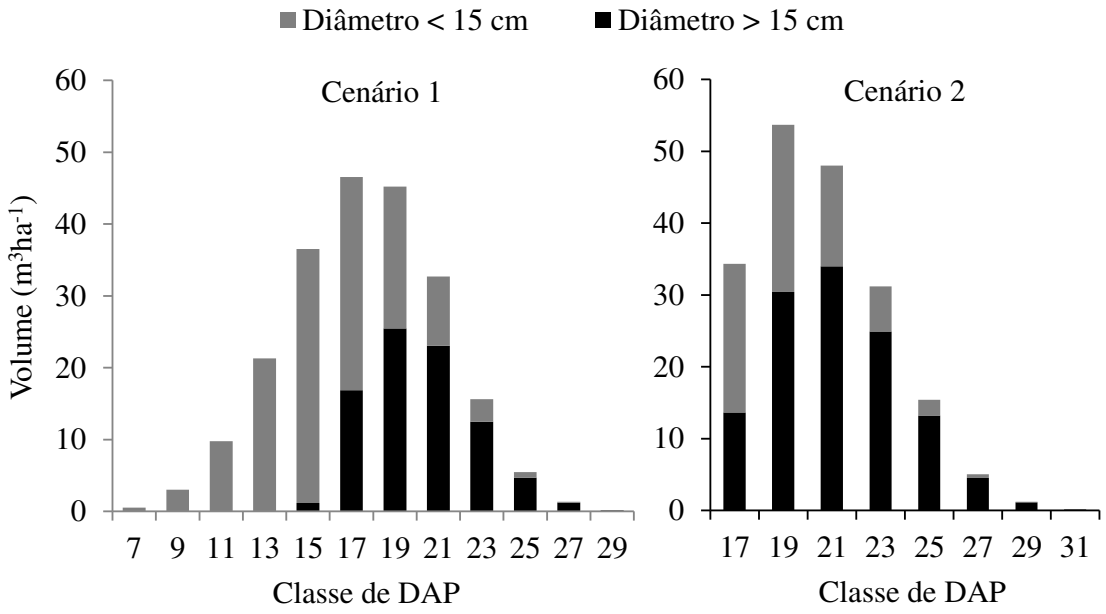


Figura 4.22 – Volumes por classe de diâmetro e sortimento na idade de seis anos para plantio de eucalipto simulado nos cenários sem desbaste (1) e com desbaste (2).

Os dois cenários foram economicamente viáveis, com VPL e VAE maiores que zero (Tabela 4.7). A taxa de desconto utilizada (9,0%) ficou bem abaixo dos valores das TIR, correspondendo a menos de sua metade. Todos os indicadores

econômicos foram melhores para o cenário 2, com uma diferença entre os dois cenários de aproximadamente 37% para os valores de VPL e VAE.

Tabela 4.6 – Fluxo de caixa para o plantio de eucalipto simulado nos cenários sem desbaste (cenário 1) e com desbaste (cenário 2).

Ano	Custo ($R\$ha^{-1}$)	Receita no cenário 1 ($R\$ha^{-1}$)	Receita no cenário 2 ($R\$ha^{-1}$)
0	2.719,99	-	-
1	1.210,90	-	-
2	455,51	-	-
3	307,19	-	1.336,15
4	307,19	-	-
5	307,19	-	-
6	307,19	12.990,44	12.938,21
Total	5.615,18	12.990,44	14.274,36

Tabela 4.7 – Valores dos indicadores econômicos valor presente líquido (VPL), valor anual equivalente (VAE) e taxa interna de retorno (TIR), cenários sem desbaste (cenário 1) e com desbaste (cenário 2).

Cenário	VPL ($R\$ha^{-1}$)	VAE ($R\$ha^{-1}$)	TIR
1	2.693,81	600,50	18,71%
2	3.694,42	823,56	22,72%

5. DISCUSSÃO

É vantajoso o fato das cinco ferramentas deste estudo, cujas interfaces foram apresentadas nas Figuras 4.2, 4.5, 4.6, 4.8, 4.10 e 4.12, terem sido desenvolvidas dentro de um único arquivo chamado *Simular Floresta.pyt* (Figura 4.1). Uma vez que a definição dos parâmetros de entrada, sua a validação e a rotina de execução das ferramentas foram definidos diretamente no texto deste arquivo, o acesso e distribuição destas ferramentas fica simplificado, pois o arquivo pode ser facilmente compartilhado, e a colaboração entre diferentes usuários na sua edição é feita em qualquer editor de texto ou ambiente de desenvolvimento (IDE).

As caixas de ferramentas (*toolbox*) criadas no ambiente do ArcGIS se estabeleceram como recursos interessantes para a comunidade científica (BROWN, 2014; JEBUR et al., 2015; MARCATTI, 2013). Elas são recursos fáceis de operar, pois possuem interface gráfica amigável e, além disso, um usuário com moderado conhecimento de programação pode facilmente editá-las, potencialmente aumentando suas funcionalidades ou adequando sua execução a uma tarefa específica não contemplada previamente. Como visto em Roberts et al. (2010), o programador pode ainda integrar recursos de outras linguagens além do Python, como C++, Matlab e R.

Os resultados obtidos com o uso da caixa de ferramentas *Simular Floresta* foram satisfatórios em relação ao realismo. O formato das distribuições de *dap* nos exemplos de uso das ferramentas (Figuras 4.4 e 4.7) foi coerente com o que é encontrado em estudos de distribuição de diâmetros de florestas equiâneas (CASTRO et al., 2016; GONZÁLEZ et al., 2002; NOGUEIRA et al., 2006). Isto aliado à boa consistência dos resultados das estimativas de sobrevivência, altura e volume (Figuras 4.7, 4.9 e 4.11), bem como da classificação de *dap* (Figura 4.13) indica que as ferramentas podem ser utilizadas de forma confiável para os fins que foram propostas.

Como esperado, os resultados do semivariograma confirmaram que houve correlação entre as dimensões das árvores criadas e suas vizinhas (Figura 4.14). O

efeito pepita (*nugget*) foi um pouco alto devido à heterogeneidade aleatória (não-espacial) inserida durante a alocação dos diâmetros aos pontos. Por conta disso, a contribuição (*partial sill*) não foi tão grande. O alcance (*range*) de 98,08 m foi próximo ao valor de 108,90 m encontrado por Lundgren et al. (2015), também para diâmetro de eucalipto no espaçamento 3,0 x 2,0 m. Lima et al. (2010) encontraram um valor mais baixo (67,50 m), porém para eucalipto no espaçamento 4,0 x 4,0 m. Caso os valores do semivariograma ficassem fora da realidade, as características de distribuição espacial poderiam ser ajustadas, uma vez que a alocação dos diâmetros iniciais pode ser manipulada através dos parâmetros *Número de pontos aleatórios por ha (...)*, *Distância mínima entre os pontos (...)* e *Nível de heterogeneidade (...)* da ferramenta *Criar árvores com dap*.

No desenvolvimento da caixa de ferramentas, optou-se por não incluir o índice de competição. Normalmente encontrado em modelos em nível de árvore individual, o índice de competição costuma ser utilizado para orientar a intensidade do crescimento em altura e *dap*, bem como influenciar na mortalidade das árvores. A falta do índice impede que a interação entre a árvore e as suas competidoras seja contabilizada na modelagem, diminuindo um pouco o realismo biológico da simulação. Todavia, Martins et al. (2011) mencionam que a competição é difícil de ser mensurada, pois não se conhece bem sua ligação com a redução dos recursos disponíveis e com a redução da taxa de crescimento. Sendo assim, uma vez que esta variável possui comportamento particular para diferentes espécies e condições de plantio, não foi viável implementá-la satisfatoriamente de maneira generalizada nas ferramentas. Pelos bons resultados vistos no item 4.2 deste estudo, a ausência do índice, de modo geral, não prejudicou as simulações.

Houve consistência na simulação dos plantios de eucalipto de Demolinari (2006) e Lopes (2007). As características das Figuras 4.15 e 4.16 indicam bom realismo biológico das simulações, deixando visível a diferença da densidade de plantas no espaço em decorrência das diferenças entre os dois espaçamentos. Os pontos onde havia árvores mortas na idade inicial do plantio no espaçamento 3,0 x 3,0 m também puderam ser vistos. Outra característica observada foi a rugosidade do dossel, correspondendo às regiões distintas em que as árvores tendem a ser maiores ou menores em relação às demais.

Em relação aos modelos presentes nas Tabelas 4.1, 4.2, 4.3 e 4.4, vale reforçar que estes foram ajustados para parametrizar as ferramentas de simulação. Os coeficientes de determinação (R^2) e erro padrão residual (S_{yx}) são referentes aos ajustes dos modelos aos dados médios dos plantios (Tabelas 3.1, 3.2, 3.3 e 3.4) e não devem ser tomados como medidas de exatidão da simulação como um todo.

Os valores de $R^2 = 100$ e $S_{yx} = 0{,}00$ vistos na Tabela 4.3, foram consistentes. Isto porque as tendências do espaçamento 10,0 x 4,0 m utilizadas para ajuste do modelo de Pienaar e Shiver (Tabela 3.4) foram obtidas a partir da aplicação deste mesmo modelo encontrado em Lopes (2007), diferindo apenas no fato da variável idade ser originalmente em meses e não em anos.

O resultado das simulações foi mais bem avaliado ao se comparar as tendências calculadas de diâmetro quadrático médio (q), árvores por hectare (N), médias das alturas ($\overline{Ht}$), e volume (V) de todas as árvores criadas pela caixa de ferramentas *Simular Floresta* com os seus valores esperados (Figuras 4.17 e 4.18), bem como as distribuições de diâmetros simuladas (Figura 4.19) com as distribuições esperadas (Figura 4.20).

Nestas comparações, ficou confirmada a capacidade das ferramentas em simular em nível de árvore as condições vistas na literatura para os dois espaçamentos por meio da parametrização proposta. Em termos de realismo, também foram obtidos bons resultados, com as curvas de q, $\overline{Ht}$, e V apresentando forma logarítmica, com tendência de estabilização do crescimento nas idades finais, como visto em Cao (2004) e Crecente-Campo et al. (2010).

Em relação às distribuições de diâmetros, para o espaçamento 10,0 x 4,0 m, há uma discrepância em relação aos valores esperados em certas idades porque o modelo de distribuição de diâmetros utilizado em Lopes (2007) foi truncado à direita, gerando uma tendência que não foi tão bem simulada pelas ferramentas desenvolvidas. Segundo Soares et al. (2011), a função Weibull truncada à direita é utilizada quando se deseja considerar que a distribuição estimada deve estar dentro de um intervalo determinado considerando um diâmetro máximo. De toda forma, o comportamento das curvas de distribuição de *dap* das duas simulações ao longo do tempo foi consistente. Em concordância com o que é discutido em Leite et al. (2005), as distribuições de diâmetros se achataram longo do tempo e deslocaram para a

direita, sendo esse seu comportamento natural em povoamentos equiâneos. Com isso, pode-se inferir que a mudança do número de árvores por classe de diâmetro ao longo do tempo, obtida pela simulação, está de acordo com a teoria. Outro aspecto que também esteve conforme a teoria foi o comportamento do ingresso de árvores em sucessivas classes de diâmetros, pois pôde ser visto que as áreas correspondentes aos ingressos (diferenças entre as áreas à direita de curvas sucessivas) diminuíram com o passar do tempo.

Tratando dos cenários simulados para análise econômica (Tabela 4.5 e Figuras 4.21 e 4.22), foi visto que os resultados estiveram de acordo com o que é comentado em Gorgens et al. (2007). Os autores mencionam que o desbaste não influencia a produção líquida final e sim a qualidade final da madeira. Isto porque o desbaste proporciona o aumento do diâmetro mínimo da floresta e a manutenção do diâmetro máximo ao longo do tempo. Estas características justificam o fato da tendência de crescimento após o desbaste se manter igual à tendência de crescimento da floresta não-desbastada e a distribuição de diâmetros da floresta desbastada ter suas árvores concentradas em classes maiores, produzindo maior volume de toras acima de 15 cm de diâmetro.

Os resultados do fluxo de caixa e dos indicadores econômicos (Tabelas 4.6 e 4.7) também foram consistentes com o que é encontrado na literatura. A cultura do eucalipto tem se mostrado viável sob as taxas de desconto normalmente aplicadas ao setor florestal. Assim como nos resultados da Tabela 4.7, o valor presente líquido (VPL), o valor anual equivalente (VAE) e a taxa interna de retorno (TIR) também foram melhores quando se aplicou o desbaste às plantações em Soares et al. (2003), Dias et al. (2005) e Folmann et al. (2014). Esta vantagem econômica obtida ao se manejar plantações de eucalipto sob regime de desbaste já era esperada. Têm maior relevância neste trecho do estudo os métodos aplicados para se definir as árvores selecionadas para desbaste e definir tendência de crescimento pós-desbaste. Eles demonstram o potencial de manipulação dos dados gerados pela caixa de ferramentas *Simular Floresta*. Além da interação com uma grande quantidade de funções disponíveis através da programação em Python, as ferramentas padrão de geoprocessamento do ArcGIS também podem ser usadas para se trabalhar os dados simulados. Aproveitando-se disso, outros estudos econômicos podem ser realizados, como por exemplo, avaliar a perda de receita em consequência de uma mudança na

legislação que aumente as áreas de preservação permanente da propriedade, ou a diferença nos lucros de um empreendimento pela troca da monocultura em uma parte da propriedade por um sistema agroflorestal.

A reprodução de florestas em ambiente computacional não é algo essencialmente novo. Outros simuladores em nível de árvore individual já foram produzidos. O SILVA (PRETZSCH et al., 2002), por exemplo, simula florestas que ocorrem na Europa a partir de modelos dependentes da distância e o PROGNAUS (MONSERUD et al., 1997) simula florestas que ocorrem especificamente na Áustria utilizando modelos independentes da distância. Para simulação de árvores de florestas dos Estados Unidos há o Forest Vegetation Simulator (FVS) (CROOKSTON e DIXON, 2005). Também merecem ser citados o Capsis (Computer-Aided Projection for Strategies In Silviculture) (DUFOUR-KOWALSKI et al., 2012) e o GreenLab (COURNÈDE et al., 2009). Estes programas muitas vezes apresentam um nível de realidade biológica maior quando comparado às ferramentas desenvolvidas no presente estudo. Isto porque eles normalmente envolvem em suas simulações o uso de variáveis ambientais, como a diferença entre a temperatura mais fria e mais quente do ano, capacidade relativa de retenção de água do solo, precipitação, suprimento de nutrientes no solo, bem como o cálculo da competição entre as árvores a partir de modelos tridimensionais de copa. Isto permite que possam ser usados para investigar respostas de árvores e povoamentos a mudanças nas condições ambientais, incluindo as provocadas por manejo silvicultural.

Apesar dos simuladores citados produzirem bons resultados, há algumas diferenças que tornam o uso da caixa de ferramentas *Simular Floresta* mais desejável em muitos casos. Estes simuladores algumas vezes possuem a parametrização de suas ferramentas feita de forma específica em termos de condições ambientais e estruturas de povoamento, o que torna as suas capacidades preditivas limitadas fora das condições das regiões para que foram desenvolvidos. Vê-se em Cournedè et al. (2009) que uma alternativa como a modelagem funcional-estrutural, presente no programa GreenLab, pode contornar este problema, mas demanda capacidade computacional além daquela que o usuário comum possui. A simulação pelas ferramentas desenvolvidas no presente estudo é de execução simples, feita a partir de poucos modelos, não é limitada a uma localidade específica e pode ser executada em pouco tempo em qualquer computador disponível no mercado atual. Além disso, na

caixa de ferramentas *Simular Floresta* os períodos e intervalos de crescimento das árvores podem ser definidos em anos ou frações de ano, enquanto o menor passo de tempo de simulação do programa SILVA, por exemplo, é um período de crescimento de 5 anos. Este intervalo seria inadequado para estudos com espécies de crescimento rápido. Por fim, ter as ferramentas funcionando em um ambiente SIG, deixa abertas opções diversas de análises de geoprocessamento aplicadas aos resultados simulados, o que outros simuladores normalmente não permitem.

Vale ressaltar que os resultados obtidos, apesar de consistentes em relação às variáveis de interesse, são uma representação aproximada do que acontece no mundo real.

As ferramentas apresentam limitações. Por exemplo, há uma grande quantidade de variáveis ambientais locais que influenciam o crescimento de cada árvore e que não são contabilizadas neste tipo de simulação. Como mencionam Encinas et al. (2005), o crescimento de uma árvore poderá apresentar diferentes variações nas suas dimensões em altura, diâmetro, volume, área basal e peso, em função de diversos fatores que nem sempre podem ser controlados ou monitorados, como os fatores genéticos das espécies e suas interações com o ambiente. Outras restrições são: o fato da simulação não contemplar a resposta a tratos silviculturais, como desbastes e fertilização; as ferramentas não englobarem o uso de modelos com entradas diferentes de *dap* e idade; e as ferramentas não permitirem a criação de árvores com uma orientação específica no terreno. Por fim, selecionar aleatoriamente as árvores vítimas de mortalidade, sem contabilizar a competição, não é o ideal.

Como este estudo é um primeiro passo, aprimoramentos futuros podem ser feitos para resolver estas limitações. Por exemplo, dentro código de programação da ferramenta *Criar árvores com dap*, a função utilizada para criação dos pontos permite que a sua orientação seja escolhida. Para tal, deve-se modificar a ferramenta, adicionando como parâmetro de entrada a variável "y_axis_coord", descrita na seção 3.1.1. Alguns artifícios também podem ser utilizados, como classificar as porções do terreno de acordo com as suas características, representando assim sítios ambientalmente diferentes, e executar as ferramentas com parâmetros específicos para as porções semelhantes.

6. CONCLUSÕES

A caixa de ferramentas *Simular Floresta* é capaz de simular, com eficiência, florestas equiâneas em nível de árvore, dentro de um ambiente SIG.

Os resultados das ferramentas desenvolvidas neste estudo são consistentes em relação ao que é esperado das tendências ao longo do tempo de crescimento em diâmetro quadrático médio, sobrevivência, média das alturas, volume por hectare e distribuição de diâmetros.

É possível parametrizar as ferramentas de *Simular Floresta* a partir de dados da literatura para simular, em nível de árvore, plantios que estejam presentes em estudos como os de Demolinari (2006) e Lopes (2007).

É possível, através de programação em Python, selecionar árvores para desbaste com base em critério de redução de área basal nas monoculturas simuladas. Também por programação é possível estimar a tendência de crescimento em *dap* das árvores remanescentes após o desbaste.

A aplicação de desbaste de 30% da área basal aos três anos e venda do restante da madeira aos seis anos a plantios de eucalipto simulados no espaçamento 3,0 x 3,0 m, na situação de venda da madeira em pé distribuída em dois sortimentos, traz melhores resultados econômicos em relação ao corte e venda de toda a floresta aos seis anos.

7. REFERÊNCIAS BIBLIOGRÁFICAS

ALCÂNTARA, A. E. M. de. **Alternativas de modelagem para projeção do crescimento de eucalipto em nível de povoamento**. 2012. 66 f. Dissertação (Mestrado em Ciência Florestal) - Universidade Federal de Viçosa, Viçosa, MG, 2012.

ALEXANDRE, P. M. de M. **Calibração do modelo 3-PG para povoamentos de pinheiro bravo (*Pinus pinaster*) em Portugal**. 2009. 73 f. Dissertação (Mestrado em Engenharia Florestal e dos Recursos Naturais) - Universidade Técnica de Lisboa, Lisboa.

ALMEIDA, A.; TOMÉ, M. Sistema para a Predição do Crescimento da Cortiça. **Silva Lusitana**, v. 16, n. 1, p.83-95, 2008.

ALMEIDA, A. C.; LANDSBERG, J. J.; SANDS, P. J. Parameterisation of 3-PG model for fast-growing *Eucalyptus grandis* plantations. **Forest Ecology and Management**, n. 193, p. 179-195, 2004.

ARAÚJO JÚNIOR, C. A. **Um sistema multiagente para otimização do transporte florestal**. 2016. 82 f. Tese (Doutorado em Ciência Flroestal) - Universidade Federal de Viçosa, Viçosa.

ARAÚJO JÚNIOR, C. A.; LEITE, H. G.; CASTRO, R. V. O.; BINOTI, D. H. B.; ALCÂNTARA, A. E. M. DE; BINOTI, M. L. M. da S. Modelagem da distribuição diamétrica de povoamentos de eucalipto utilizando a função Gama. **Cerne**, v. v. 19, n. 2, p. 307-314, 2013.

ARCE, J. R. Modelagem da estrutura de florestas clonais de *Populus deltoides* Marsh. através de distribuições diamétricas probabilísticas. **Ciência Florestal**, v. 14, n. 1, p. 149-164, 2004.

ARONOFF, S. **Geographic information systems: a management perspective**. Ottawa: WDL Publications, 1989. 295p.

ASSMANN, E. **The principles of Forest yield study.** New York: Pergamon Press, 1970. 506p.

AVERY, T. E., BURKHART, H. E. **Forest measurements**. 4. ed. New York: McGraw-Hill Book Co, 1994. 408 p

BALAGUER-BESER, A.; RUIZ, L. A.; HERMOSILLA, T.; RECIO , J. A. Using semivariogram indices to analyze heterogeneity in spatial patterns in remotely sensed images. **Computers & Geosciences**, v. 50, p. 115-127, 2013.

BARRETO, L. S. US-EVEN. A program to support the forestry of some even-aged north-American stands. **Silva Lusitana**, v. 7, n. 2, p. 233-248, 1999.

BARTOSZECK, A. C. de P. e S.; MACHADO, S. do A.; FIGUEIREDO FILHO, A.; OLIVEIRA, E. B. A distribuição diamétrica para bracatingais em diferentes idades, sítios e densidades na região metropolitana de Curitiba. **Floresta**, v. 34, n. 3, p. 305-323, 2004.

BETTINGER, P.; BOSTON, K.; SIRY, L.P.; GREBNER, R.L. **Forest management and planning**. New York: Elsevier, 2009. 331p.

BETTINGER, P.; GRAETZ, D.; BOSTON, K. SESSIONS, J.; CHUNG, W. Eight heuristic planning techniques applied to three increasingly difficult wildlife planning problems. **Silva Fennica**, v. 36, n. 2, p. 561-584, 2002.

BETTINGER, P.; SESSIONS, J.; BOSTON, K. Using Tabu search to schedule timber harvests subject to spatial wildlife goals for big game. **Ecological Modelling**, n. 94, p. 111-123, 1997.

BINOTI, D. H. B. **Sistemas computacionais aplicados ao manejo florestal**. 2012. 122 f. Tese (Doutorado em Ciência Flroestal) - Universidade Federal de Viçosa, Viçosa.

BINOTI, D. H. B. **Estratégias de regulação de florestas equiâneas com vistas ao manejo da paisagem**. 2010. 159 f. Dissertação (Mestrado em Ciência Florestal) - Universidade Federal de Viçosa, Viçosa, MG.

BORGES, J. S. **Parametrização, calibração e validação do modelo 3-PG para eucalipto na região do Cerrado de Minas Gerais**. 2009. 89 f. Dissertação (Mestrado em Solos e Nutrição de Plantas) - Universidade Federal de Viçosa, Viçosa, MG, 2009.

BRANDELEIRO, C.; ANTUNES, M. U. F.; GIOTTO, E. Silvicultura de precisão: nova tecnologia para o desenvolvimento florestal. **Ambiência**, v. 3, n. 2, p. 269-281, 2007.

BRAVO-OVIEDO, A.; RÍO, M. del; MONTERO, G. Site index curves and growth model for Mediterranean maritime pine (*Pinus pinaster* Ait.) in Spain. **Forest Ecology and Management**, n. 201, p. 187-197, 2004.

BROWN, J. L. SDMtoolbox: a Python-based GIS toolkit for landscape genetic, biogeographic and species distribution model analyses. **Methods in Ecology and Evolution**. v. 5, n. 7 p. 694–700, 2014.

BUONGIORNO, J.; GILLES, J. K. **Decision methods for forest resource management**. San Diego, CA: Academic Press. 2003. 439p.

BURKHART, H. E.; CAO, Q. V.; WARE, K. D. **A comparison of growth and yield prediction models for loblolly pine**. Blacksburg: Virginia Polytechnic Institute and State University, School of Forestry and Wildlife Resources, 1981. 59 p. (Publ., FWS-2).

BURROUGH, P. A. **Principles of geographical information systems for land resources assessment**. Oxford: Clarendon Press, 1986. 193 p.

CAMPOS, J. C. C.; LEITE, H. G. **Mensuração florestal**: perguntas e respostas. 4 Ed. Viçosa: Editora UFV, 2013. 605 p.

CAO, Q. V. Annual tree growth predictions from periodic measurements. In: Connor, K. F. (Ed.), BIENNIAL SOUTHERN SILVICULTURAL RESEARCH CONFERENCE, 12., General Technical Report SRS-71, 2004. Asheville, NC. **Proceedings...** Asheville, NC: USDA Forest Service Southern Research Station, 2004, p. 212-215.

CASTRO, J. P. **Aplicação da detecção remota em inventário florestal**. 2004. 274 f. Tese (Doutorado em Ciências Florestais) - Universidade de Trás-os-Montes e Alto Douro, Vila Real.

CASTRO, R. R. de. **Regulação de florestas equiâneas incluindo restrições de adjacência**. 2007. 74 f. Dissertação (Mestrado em Ciência Florestal) - Universidade Federal de Viçosa, Viçosa, MG.

CASTRO, R. V. O.; ARAÚJO, R. A. A.; LEITE, H. G., CASTRO, A. F. N. M.; SILVA, A.; PEREIRA, R. S.; LEAL, F. A. Modelagem do crescimento e da produção de povoamentos de *Eucalyptus* em nível de distribuição diamétrica utilizando índice de local. **Revista Árvore**, v. 40, n. 1, p. 107-116.

CLUTTER, J. L. Compatible growth and yield models for loblolly pine. **Forest Science**, v. 9, n. 3, p. 354-371, 1963.

CONDÉS, S. e STERBA, U. Comparing an individual tree growth model for Pinus halepensis Mill. in the Spanish region of Murcia with yield tables gained from the same area. **European Journal of Forest Research**. v. 127, n. 3, p. 253-261.

COURNÀDE, P-H.; GUYARD, T.; BAYOL, B.; GRIFFON, S.; COLIGNY, F. de.; BORIANNE, P.; JAEGER, M.; REFFYE, P. de. A forest growth simulator based on functional-structural modelling of individual trees. In: Li, B.; Jaeger, M.; Guo, Y. (Eds.), PLANT GROWTH, MODELING, SIMULATION, VISUALIZATION, AND APPLICATIONS (PMA), 3., 2009. Los Alamitos, CA. **Proceedings...** Washington, DC, USA: IEEE Computer Society, 2009, p. 34-41.

COX, D. R. The regression analysis of binary sequences. **Journal of the Royal Statistical Society**, v. 20, n. 2, p. 215-242, 1958.

CRECENTE-CAMPO, SOARES, P.; TOMÉ, M.; DIÉGUEZ-ARANDA, U. D. Modelling annual individual-tree growth and mortality of Scots pine with data obtained at irregular measurement intervals and containing missing observations. **Forest Ecology and Management**, n. 260, p. 1965-1974, 2010.

CROOKSTON, N. L.; DIXON, G. E. The forest vegetation simulator: A review of its structure, content, and applications. **Computers and Electronics in Agriculture**, v. 49, p. 60-80, 2005.

DANIEL, T. W.; HELMS, J. A.; BAKER, F. **Principles of silviculture**. New York: Mc Graw-Hill, 1979, 500 p.

DAVIS, C.; CÂMARA, G. **Arquitetura de sistemas de informações geográficas**. In: CÂMARA, G.; DAVIS, C.; MONTEIRO, A. M. V. Introdução à Ciência da Geoinformação. São José dos Campos: Inpe, 2001. Disponível em:

< http://mtc-m12.sid.inpe.br/attachment.cgi/sid.inpe.br/sergio/2004/04.19.14.10/doc/cap3-arquitetura.pdf >. Acesso em: 29 dez. 2007.

DAVIS, L. S.; JOHNSON, K. N. **Forest management**, 3 ed. New York: McGraw-Hill, 1987. 790p.

DAVIS, L. S.; JOHNSON, K. N.; BETTINGER, P.; HOWARD, T. E. **Forest management**: to sustain ecological, economic, and social values. 4. ed. Illinois: Waveland, 2005. 804p.

DEMOLINARI, R. de A. **Crescimento de povoamentos de eucalipto não-desbastados**. 2006. 81 f. Tese (Doutorado em Ciência Florestal) - Universidade Federal de Viçosa, Viçosa, MG.

DIAS, A. N.; LEITE, H. G.; SILVA, M. L. da; CARVALHO, A. F. de. Avaliação financeira de plantações de eucalipto submetidas a desbaste. **Revista Árvore**, v. 29, n. 3, p.419-429, 2005.

DIAS, J. E.; GOMES, O. V. de O.; GOES, M. H. de B. Áreas de riscos de erosão do solo: uma aplicação por geoprocessamento. **Floresta e Ambiente**, v. 8, n. 1, p. 01-10, 2001.

DIAS, S. S.; FERREIRA, A. G.; GONÇALVES, A. C. Definição de zonas de aptidão para espécies florestais com base em características edafo-climáticas. **Silva Lusitana**, número especial, p. 17–35, 2008.

DUFOUR-KOWALSKI, S.; COURBAUD, B.; DREYFUS, P.; MEREDIEU, C.; COLIGNY F. de. Capsis: an open software framework and community for forest growth modelling. **Annals of Forest Science** v. 69, p. 221–233, 2012.

EMBRAPA. **Softwares Florestais - Portal Embrapa**. Disponível em: < https://www.embrapa.br/florestas/transferencia-de-tecnologia/softwares-florestais >. Acesso em: 01 ago. 2016.

ENCINAS, J. I.; SILVA, G. F.; PINTO, J. R. R. **Idade e crescimento das árvores**. Brasília: UnB, 2005. 40p.

FACCO, A. G.; RIBEIRO, A.; PRUSKI, F. F.; MONTEIRO, W. C.; LEITE, F. P.; ANDRADE, R. G.; MENEZES, S. J. M. da C de. Técnicas de geoinformação para estimativa do balanço hídrico em eucalipto. **Pesquisa Agropecuária Brasileira**, v. 47, n. 9, p. 1243-1250, 2012.

FERNANDES, E. N.; FERNANDES FILHO, E. I.; SILVA, E.; SILVA, C. A. B. da; RICARDO, J. de F. EROSYS: sistema de apoio ao processo de avaliação de impactos ambientais de atividades agropecuárias. **Revista Brasileira de Agroinformática**, v. 4, n. 1, p. 1-12, 2002.

FINGER, C. A. G. **Fundamentos de Biometria Florestal.** Santa Maria, RS: UFSM/CEPEF/FATEC, 1992. 269p.

FOLMANN, W. T.; MIRANDA, G. de M.; DIAS, A. N.; MORO, F. de C.; FERNANDEZ, M. L. Q. Viabilidade de projetos florestais em três regimes de manejo na mesorregião Centro-Oriental do Paraná. **Floresta**, v. 44, n. 1, p. 153-160, 2014.

GADOW, K. von. Die Erfassung von Durchmesserverteilungen in gleichaltigen Kiefernbeständen, **Forstwissenschafliches Centralblatt**, v. 103, p. 369-374, 1984.

GARAY, L. **Tropical forest utilization system**. VIII. A tree taper model for entire stem profile including buttressing. Seattle: Institute of Forest Products, College of Forest Resources, University of Washington, 1979. 64 p. (contrib. 36).

GOMPERTZ, B. On the nature of the function expressive of the law of human mortality, and on a new mode of determining the value of life contingencies. **Philosophical Transactions of the Royal Society of London**, v. 115, p. 513-585, 1825.

GONZÁLEZ, J. G. A.; SCHRÖDER, J.; SOALLEIRO, R. R.; GONZÁLEZ, A. D. R. Modelling the effect of thinnings on the diameter distribution of even-aged Maritime pine stands. **Forest Ecology and Management**, n. 165, p. 57-65, 2002.

GORGENS, E. B.; LEITE, H. G.; NOGUEIRA, G. S.; DIAS, A. N. Tendência de crescimento de povoamento de eucalipto após aplicação de desbaste. **Revista Árvore**, v. 31, n. 5, p. 879-885, 2007.

HYNYNEN, J.; AHTIKOSKI, A.; SIITONEN, J.; SIEVÄNEN R.; LISKI, J. Applying the MOTTI simulator to analyze the effects of alternative management schedules on timber and non-timber production. **Forest Ecology and Management**, n. 207, p. 5-18, 2005.

JEBUR, M. N. ; PRADHAN, B.; SHAFRI, H. Z. M.; YUSOFF, Z. M.; TEHRANY, M. S. An integrated user-friendly ArcMAP tool for bivariate statistical modelling in geoscience applications. **Geoscientific Model Development**, v. 8, n. 3, p. 881-891, 2015.

JOHNSON, N. L. Systems of frequency curves generated by methods of translation. **Biometrika**, v. 36, n. 1/2, p. 149-176, 1949.

KANEGAE JÚNIOR, H.; MELLO, J. M. de; SCOLFORO, J. R. S.; OLIVEIRA, A. D. de. Avaliação da continuidade espacial de características dendrométricas em diferentes idades de povoamentos clonais de *Eucalyptus sp*. **Revista Árvore**, v. 31, n. 15, p. 859-866, 2007

KORF, V. Příspěvek k matematické definici vzrůstového zákona hmot lesních porostů. [Contribution to the mathematical definition of forest stands volume growth law]. **Lesnicka Prace**, v. 18, p. 339-379, 1939.

LANDSBERG, J. J.; WARING, R. H. A generalised model of forest productivity using simplified concepts of radiation-use efficiency, carbon balance and partitioning. **Forest Ecology and Management**, n. 95, p. 209-228, 1997.

LEITE, E. da S. **Modelagem técnica e econômica de um sistema de colheita florestal mecanizada de toras curtas**. 2012. 130 f. Tese (Doutorado em Ciência Florestal) – Universidade Federal de Viçosa, Viçosa, 2012.

LEITE, H. G.; NOGUEIRA, G. S.; CAMPOS, J. C. C.; SOUZA, A. L. de; CARVALHO, A. Avaliação de um modelo de distribuição diamétrica ajustado para povoamentos de *Eucalyptus sp*. submetidos a desbaste. **Revista Árvore**, v. 29, n. 2, p. 271-280, 2005.

LI, R.; BETTINGER, P.; WEISKITTEL, A. Comparisons of three different methods used to generate forest landscapes for spatial harvest scheduling problems with adjacency restrictions. **Mathematical and Computational Forestry & Natural-Resource Sciences**. v. 2, p. 53–60, 2010.

LIMA, C. G. da R.; CARVALHO, M. de P. e; NARIMATSU, K. C. P.; SILVA, M. G. da; QUEIROZ, H. A. de. Atributos físico-químicos de um latossolo do cerrado brasileiro e sua relação com características dendrométricas do eucalipto. **Revista Brasileira de Ciência do Solo**, v. 34, n. 1, p.163-173, 2010.

LOPES, P. F. **Modelo de distribuição de diâmetros para clones de eucalipto em sistema agroflorestal**. 2007. 42 f. Dissertação (Mestrado em Ciência Florestal) – Universidade Federal de Viçosa, Viçosa.

LUNDGREN, A. L. **The effect of initial number of trees per acre and thinning densities on timber yields from red pine plantations in the Lake States**. USDA Forest Service, Research Paper NC-193, 25 p. 1981.

LUNDGREN, W. J. C.; SILVA, J. A. A. da; FERREIRA, R. L. C. Estimação de volume de madeira de eucalipto por cokrigagem, krigagem e regressão. **Cerne**, v. 21, n. 2, p. 243-250, 2015.

LUNDQVIST, B. On the height growth in cultivated stands of pine and spruce in Northern Sweden. **Meddelanden Fran Statens Skogfoesk-institut**, v. 47, p. 1-64, 1957.

MABVURIRA, B. e MIINA, J. Individual-tree growth and mortality models for *Eucalyptus grandis* (Hill) Maiden plantations in Zimbabwe. **Forest Ecology and Management**. v. 161, n. 1, p 231-245, 2002

MACHADO, C. C.; LOPES, E. S. **Colheita florestal**. 2.ed. Viçosa: Editora UFV, 2008. 501 p.

MACHADO, S. do A.; AUGUSTYNCZICK, A. L. D.; NASCIMENTO, R. G. M.; FIGURA, M. A.; SILVA, L. C. R. da; MIGUEL, E. P.; TÉO, S. J. Distribuição diamétrica de *Araucaria angustifolia* (Bert.) O. Ktze. em um fragmento de floresta ombrófila mista. **Scientia Agraria**, v. 10, n. 2, p. 103-110, 2009.

MARCATTI, G. E. **Caminhamento ótimo para acesso às parcelas de inventário florestal**. 2013. 41 f. Dissertação (Mestrado), Universidade Federal de Viçosa, Viçosa.

MARTINS, F.B. **Modelagem de crescimento em nível de árvore individual para plantios comerciais de eucalipto**. 2011. 143p. Tese (Doutorado) – Universidade Federal de Viçosa, Viçosa.

MARTINS, F. B.; SOARES, C. P. B.; LEITE, H. G.; SOUZA, A. L. de; CASTRO, R. V. O. Índices de competição em árvores individuais de eucalipto. **Pesquisa Agropecuária Brasileira**, v. 46, n. 9, p. 1089-1098, 2011.

MARTINS, R. J.; SEIXAS, F.; STAPE, J. L. Avaliação técnica e econômica de um harvester trabalhando em diferentes condições de espaçamento e arranjo de plantio em povoamento de eucalipto. **Scientia Forestalis**, v. 37, n. 83, p. 253-263, 2009.

MELLO, J. M. de. **Geoestatística aplicada ao inventário florestal**. 2004. 122 f. Tese (Doutorado em Recursos Florestais) - Escola Superior de Agricultura Luiz de Queiroz, Universidade de São Paulo, Piracicaba.

MELLO, J. M. de; OLIVEIRA, M. S. de; BATISTA, J. L. F.; JUSTINIANO JÚNIOR, P. R.; KANEGAE JÚNIOR, H. Uso do estimador geoestatístico para predição volumétrica por talhão. **Floresta**, v. 36, n. 2, p. 251-260, 2006.

MIRANDA, R. O. V. de; DIAS, A. N.; FIGUEIREDO FILHO, A.; SOARES, I. D.; CRUZ, J. P. da. Modelagem do crescimento e produção em classes de precipitação pluviométrica para *Eucalyptus* sp. **Floresta**, v. 45, n. 1, p. 117-128, 2015.

MONSERUD, R. A., STERBA, H., HASENAUER, H. The single-tree stand growth simulator PROGNAUS. In: Teck, R.; Moser, M.; Adams, J. (Eds.), FOREST VEGETATION SIMULATOR CONFERENCE, 1997. Fort Collins, CO. **Proceedings…** Ogden, UT: U.S. Department of Agriculture, Forest Service, Intermountain Research Station, 1997, p. 50-56.

MYLEVAGANAM, S.; RAY, C. A spatial evapotranspiration tool at grid scale. **Open Journal of Applied Sciences**, n. 6, p. 64-77, 2016.

NOBRE, S. R. **A heurística da Razão-R aplicada a problemas de gestão florestal**. 1998. 98 f. Dissertação (Mestrado), Universidade Federal de Lavras, Lavras, MG 1998.

NOGUEIRA, G. S.; LEITE, H. G.; CAMPOS, J. C. C.; TAKIZAWA, F. H.; COUTO, L. Avaliação de um modelo de distribuição diamétrica ajustado para povoamentos de *Tectona grandis* submetidos a desbaste. **Revista Árvore**, v. 30, n. 3, p. 377-387, 2006.

NUNIFU, T. K.; MURCHISON, H. G. Provisional yield models of Teak (*Tectona grandis* Linn F.) plantations in northern Ghana. **Forest Ecology and Management**, n. 120, p. 171–178, 1999.

OLIVEIRA NETO, R. R. de. **Sensibilidade no planejamento estratégico quanto a riscos e incertezas**. 2016. 50 f. Dissertação (Mestrado em Ciência Flroestal) - Universidade Federal de Viçosa, Viçosa.

PAULINO, E. J. **Influência do espaçamento e da idade na produção de biomassa e na rotação econômica em plantios de eucalipto**, 2012. 60f. Dissertação (Mestrado em Ciência Florestal). Universidade Federal dos Vales do Jequitinhonha e Mucuri.

PIENAAR, L. V.; SHIVER, B. D. Survival functions for site prepared slash pine plantations in the flat woods of Georgia Northern Florida. **Southern Journal of Applied Forestry**, v.5, n.2, p.59-62, 1981.

PIROLI, E. L. **Introdução ao geoprocessamento**. Ourinhos: UNESP/Campus Experimental de Ourinhos, 2010. 46 p.

PRETZSCH, H.; BIBER, P.; ĎURSKÝ, J. The single tree-based stand simulator SILVA: construction, application and evaluation. **Forest Ecology and Management**, n. 162, p. 3-21, 2002.

QIN, J. e CAO, Q. V. Using disaggregation to link individual-tree and wholestand growth models. **Canadian Journal of Forest Research**. v. 36, n. 4, p 953-960, 2006.

REZENDE, J. L. P.; OLIVEIRA, A. D. **Análise econômica e social de projetos florestais**. 2. ed. Viçosa: UFV, 2008, 386 p.

RICHARDS, F. J. A flexible growth function for empirical use. **Journal of Experimental Botany**, v. 10, p. 290-300, 1959.

RIGOL-SANCHEZ, J. P.; STUART. N.; PULIDO-BOSCH, A. ArcGeomorphometry: A toolbox for geomorphometric characterisation of DEMs in the ArcGIS environment. **Computers & Geosciences**, n. 85, p. 155–163, 2015.

ROBERTS, J. J.; BEST, B. D.; DUNN, D. C.; TREML, E. A.; HALPIN, P. N. Marine Geospatial Ecology Tools: An integrated framework for ecological geoprocessing with ArcGIS, Python, R, MATLAB, and C++. **Environmental Modelling & Software**, v. 25, n. 10, p. 1197-1207, 2010.

RODRIGUES, E. L. **Avaliação da produção de água (vazão), em função de alterações de uso do solo e da implantação de florestas de eucalipto em larga escala na bacia do Rio Pará - Alto São Francisco/MG**. 2013. 102 f. Dissertação (Mestrado em Análise e Modelagem de Sistemas Ambientais) – Universidade Federal de Belo Horizonte, Belo Horizonte.

RODRIGUEZ, L. C. E.; BUENO, A. R. S.; RODRIGUES, F. Rotações de eucaliptos mais longas: análise volumétrica e econômica. **Scientia Forestalis**, n. 51, p. 15-28, 1997.

SANTOS, A. C. de A. **Crescimento mensal de povoamentos de eucalipto n idade de corte**. 2014. 42 f. Dissertação (Mestrado em Ciência Florestal) - Universidade Federal de Viçosa, Viçosa.

SBF - Serviço Florestal Brasileiro. **Guia de Financiamento Florestal 2016**. Serviço Florestal Brasileiro, Ministério do Meio Ambiente. Brasília: MMA, 2016. 104 p. Disponível em: < http://www.florestal.gov.br/destaques/guia-de-financiamento-florestal-2016 >. Acesso em: 15 nov. 2016.

SCOLFORO, H. F. **Interpoladores espaciais na estimativa da distribuição do estoque de carbono da vegetação arbórea em Minas Gerais, Brasil**. 2014. 75 f. Dissertação (Mestrado em Engenharia Florestal) - Universidade Federal de Lavras, Lavras.

SEAPA-MG - Secretaria de Estado de Agricultura, Pecuária e Abastecimento de Minas Gerais. **Preços Correntes - 14/01/2014**. Subsecretaria do Agronegócio. 2014. Disponível em: < http://www.agricultura.mg.gov.br/images/files/precos_correntes/cotacao_14_01_2014.pdf >. Acesso em: 11 nov. 2016.

SOARES, T. S.; LEITE, H. G.; SOARES, C. P. B.; VALE, A. B. do. Projeção da distribuição diamétrica e produção de povoamentos de eucalipto empregando diferentes formas da função weibull. **Revista Árvore**, v. 35, n. 5, p. 1027-1032, 2011.

SOARES, T. S.; SILVA, M. L. da; GAMA, J. R. V.; CARVALHO, R. M. M. A.; VALE, R. S. do. Avaliação econômica de plantações de eucalipto submetidas a desbaste. **Revista Árvore**, v. 27, n. 4, p.481-486, 2003.

SOUZA, V.S.; TIMOFEICZYK JUNIOR, R.; BERGER, R.; SILVA, J. C. G. L. da; DELACOTE, P. Rentabilidade econômica do arrendamento de terra para cultivo de eucalipto em São Paulo. **Floresta e Ambiente**, v. 22, n. 3, p. 345-354, 2015.

SCHIKOWSKI, A. B.; CORTE, A. P. D.; SANQUETTA, C. R. Modelagem do crescimento e de biomassa individual de Pinus. **Pesquisa Florestal Brasileira**, v. 33, n. 75, p. 269-278, 2013.

SCHUMACHER, F. X. A new growth curve and its application to timber yield studies. **Journal of Forestry**, v. 37, p. 819–820, 1939.

SCHUMACHER, F. X.; HALL, F. S. Logarithmic expression of timber-tree volume. **Journal of Agricultural Research**, v.47, n.9, p.719-734, 1933.

SMITH, J. L.; BURKHART, H. E. A simulation study assessing the effect of sampling for predictor variable values on estimates of yield. **Canadian Journal of Forest Research**, v. 14, p. 326-330, 1984.

SPATHELF, P.; NUTTO, L. **Modelagem aplicada ao crescimento florestal**. Santa Maria: UFSM, 2000. 70p.

STAPE, J. L.; RYAN, M. G.; BINKLEY, D. Testing the utility of the 3-PG model for growth of *Eucalyptus grandis* x *urophylla* with natural and manipulated supplies of water and nutrients. **Forest Ecology and Management**, v. 193, p. 219-234, 2004.

THOMPSON, M. P.; HAMANN, J. D.; SESSIONS, J. Selection and penalty strategies for genetic algorithms designed to solve spatial forest planning problems. **International Journal of Forestry Research**, v. 2009, 14 p., 2009.

VANCLAY, J. K. **Modelling forest growth and yield**: apllications to mixed tropical forests. Wallingford: CAB International, 1994. 312 p.

WEIBULL, E. H. W. A statistical distribution function of wide applicability. **Journal of Applied Mechanics**, v. 18, p. 293–297, 1951.

WORBOYS, M. F. **GIS: A Computing Perspective**. London: Taylor and Francis, 1995. 376 p.

ZAMPIN, I. C. **Sistemas de informação geográfica aplicados no levantamento e mapeamento de *Orchidaceae* na bacia hidrográfica do rio Corumbataí - SP**. 2010. 149 f. Tese (Doutorado em Geografia) - Universidade Estadual Paulista, Rio Claro, SP.

ZARNOCH S. J.; FEDUCCIA D. P.; BALDWIN V. C.; DELL T. R. **Growth and yield predictions for thinned and unthinned slash pine plantations on cutover sites in the West Gulf region**, New Orleans, LA: U.S. Department of Agriculture, Forest Service, Res. Pap. SO-264, 1991. 32 p.

ZONETE, M. F.; RODRIGUEZ, L. C. E.; PACKALÉN P. Estimação de parâmetros biométricos de plantios clonais de eucalipto no sul da Bahia: uma aplicação da tecnologia laser aerotransportada. **Scientia Forestalis**, v. 38, n. 86, p. 225-235, 2010.

APÊNDICE

Abaixo segue o código de programação contido na caixa de ferramentas *Simular Floresta*.

```
import arcpy
import random
import numpy as np
from scipy.stats import norm, gamma, weibull_min
from arcpy import env
# Allows to overwrite files
env.overwriteOutput = True


class Toolbox(object):
    def __init__(self):
        self.label = "Simular floresta"
        self.alias = "simularfloresta"
        self.tools = [CreateTreesandDiameters,
                      ProjectDiameters,
                      EstimateHeight,
                      EstimateVolume,
                      dbhClasses]


class CreateTreesandDiameters(object):
    def __init__(self):
```

```
        self.label = u"Criar \xe1rvores com dap"
        self.description = (u"Cria uma malha de pontos com base em uma \xe1rea"
                            u" e um espa\xe7amento e gera valores de dap para "
                            u"estes pontos.")
        self.canRunInBackground = False

    def getParameterInfo(self):
        forest_surface = arcpy.Parameter(
            displayName=u"\xc1rea da floresta",
            name="forest_surface",
            datatype="Feature Layer",
            parameterType="Required",
            direction="Input")
        forest_surface.filter.list = ["Polygon"]

        spacing = arcpy.Parameter(
            displayName=u"Espa\xe7amento",
            name="spacing",
            datatype="String",
            parameterType="Required",
            direction="Input")
        spacing.value = "3 x 3"

        eq_type = arcpy.Parameter(
            displayName="Tipo de modelo",
            name="eq_type",
            datatype="String",
            parameterType="Required",
            direction="Input")
        eq_type.filter.list = ["Weibull", "Normal", "Gamma"]
        eq_type.value = "Weibull"
```

```
age = arcpy.Parameter(
    displayName="Idade",
    name="age",
    datatype="Double",
    parameterType="Required",
    direction="Input")
age.value = 2.0

location_prm = arcpy.Parameter(
    displayName=u"Par\xe2metro de loca\xe7\xe3o",
    name="location_prm",
    datatype="Double",
    parameterType="Optional",
    direction="Input")
location_prm.value = 2.33630

scale_prm = arcpy.Parameter(
    displayName=u"Par\xe2metro de escala",
    name="scale_prm",
    datatype="Double",
    parameterType="Required",
    direction="Input")
scale_prm.value = 9.53052

form_prm = arcpy.Parameter(
    displayName=u"Par\xe2metro de forma",
    name="form_prm",
    datatype="Double",
    parameterType="Optional",
    direction="Input")
form_prm.value = 3.50449
```

```
alive_trees = arcpy.Parameter(
    displayName=u"% de \xe1rvores vivas",
    name="alive_trees",
    datatype="Double",
    parameterType="Required",
    direction="Input")
alive_trees.filter.type = "Range"
alive_trees.filter.list = [1.0, 100.0]
alive_trees.value = 100.0

out_feature = arcpy.Parameter(
    displayName="Fei\xe7\xe3o de sa\xedda",
    name="out_feature",
    datatype="Feature Class",
    parameterType="Required",
    direction="Output")

points_ha = arcpy.Parameter(
    displayName=(u"N\xfamero de pontos aleat\xf3rios por ha gerados "
                 u"para orientar a aloca\xe7\xe3o dos dap"),
    name="points_ha",
    datatype="Double",
    parameterType="Required",
    direction="Input",
    category=u"Par\xe2metros de distribui\xe7\xe3o espacial")
points_ha.value = 130

min_dist = arcpy.Parameter(
    displayName=(u"Distancia m\xednima entre os pontos "
                 u"aleat\xf3rios gerados"),
    name="min_dist",
    datatype="Double",
```

```
        parameterType="Required",
        direction="Input",
        category=u"Par\xe2metros de distribui\xe7\xe3o espacial")
    min_dist.value = 25.0

    heterogeneity = arcpy.Parameter(
        displayName=(u"N\xedvel de heterogeneidaede aplicada \xe0"
                     u" aloca\xe7\xe3o dos dap"),
        name="heterogeneity",
        datatype="Double",
        parameterType="Required",
        direction="Input",
        category=u"Par\xe2metros de distribui\xe7\xe3o espacial")
    heterogeneity.filter.type = "Range"
    heterogeneity.filter.list = [0.0, 1.0]
    heterogeneity.value = 0.30

    params = [forest_surface,
              spacing,
              eq_type,
              age,
              location_prm,
              scale_prm,
              form_prm,
              alive_trees,
              out_feature,
              points_ha,
              min_dist,
              heterogeneity]
    return params

def isLicensed(self):
```

```
        return True

    def updateParameters(self, parameters):
        eq_type = parameters[2]
        location_prm = parameters[4]
        scale_prm = parameters[5]
        form_prm = parameters[6]

        # Field validation according to what is inserted as parameters.
        if eq_type.value == "Normal":
            location_prm.enabled = True
            scale_prm.enabled = True
            form_prm.value = None
            form_prm.enabled = False
        elif eq_type.value == "Weibull":
            location_prm.enabled = True
            scale_prm.enabled = True
            form_prm.enabled = True
        elif eq_type.value == "Gamma":
            location_prm.value = None
            location_prm.enabled = False
            scale_prm.enabled = True
            form_prm.enabled = True
        else:
            location_prm.value = None
            location_prm.enabled = False
            scale_prm.value = None
            scale_prm.enabled = False
            form_prm.value = None
            form_prm.enabled = False
        return
```

```
def updateMessages(self, parameters):
    spacing = parameters[1]
    eq_type = parameters[2]
    location_prm = parameters[4]
    form_prm = parameters[6]

    # Field validation for conditional required values
    if eq_type.value == "Weibull":
        if not form_prm.value and (form_prm.value != 0):
            form_prm.setErrorMessage(
                u"Par\xe2metro de forma \xe9 necess\xe1rio para uso "
                u"do modelo selecionado.")
    elif eq_type.value == "Normal":
        if not location_prm.value and (location_prm.value != 0):
            location_prm.setErrorMessage(
                u"Par\xe2metro de loca\xe7\xe3o \xe9 necess\xe1rio "
                u"para uso do modelo selecionado.")
    elif eq_type.value == "Gamma":
        if not form_prm.value and (form_prm.value != 0):
            form_prm.setErrorMessage(
                u"Par\xe2metro de forma \xe9 necess\xe1rio para uso "
                u"do modelo selecionado.")

    # Checks if spacing is in the right format. It should be a string with
    # three items: a float, an 'x' and another float. Spaces make no
    # difference.
    def convert_to_float(x):
        try:
            return float(x)
        except ValueError:
            return x
```

```
    spacing_txt = spacing.valueAsText
    if spacing.value:
        if hasattr(spacing_txt, "split"):
            # If the value of spacing can be splitted, it creates a list
            # with each character of the given spacing. The character is
            # converted to float if it's possible.
            spacings = [convert_to_float(k) for k in spacing_txt.split()]
        else:
            spacing.setErrorMessage(
                (u'Espa\xe7amento deve estar no formato: "n\xfamero x '
                 u'n\xfamero"'))

        condition_1 = len(spacings) != 3
        condition_2 = str(spacings[1]).lower() != "x"
        condition_3 = type(spacings[0]) is not float
        condition_4 = type(spacings[2]) is not float
        if condition_1 or condition_2 or condition_3 or condition_4:
            spacing.setErrorMessage(
                (u'Espa\xe7amento deve estar no formato: "n\xfamero x '
                 u'n\xfamero"'))
    return

def execute(self, parameters, messages):
    forest_surface = parameters[0].valueAsText
    spacing = parameters[1].valueAsText
    eq_type = parameters[2].valueAsText
    age = parameters[3].value
    location_prm = parameters[4].value
    scale_prm = parameters[5].value
    form_prm = parameters[6].value
    alive_trees = parameters[7].value
    out_feature = parameters[8].valueAsText
```

```
points_ha = parameters[9].value
min_dist = parameters[10].value
heterogeneity = parameters[11].value

# Values for extent, origin point and y axis.
desc_surface = arcpy.Describe(forest_surface)
xtent_list = str(desc_surface.extent).split()[:4]
xtent = " ".join(k for k in xtent_list)
origin_point = " ".join(xtent_list[:2])
y_axis = xtent_list[0] + " " + str(float(xtent_list[1]) + 10)

# Sets the coordinate system of the main layer as the one for the
# outputs.
env.outputCoordinateSystem = desc_surface.SpatialReference

# Creates a field called Stand_area and calculates the area for each
# feature of the forest_surface layer.
arcpy.AddField_management(forest_surface, "AreaTalhao", "DOUBLE")

with arcpy.da.UpdateCursor(forest_surface,
                           ["SHAPE@AREA", "AreaTalhao"]) as cursor:
    for row in cursor:
        row[1] = round(row[0] / 10000.0, 2)
        cursor.updateRow(row)

# Creates and calculates the field for Total_area of the stands.
stands_area_list = arcpy.da.TableToNumPyArray(
    forest_surface, "AreaTalhao", "", True).astype(np.float32).tolist()

total_area = round(sum(stands_area_list), 2)
arcpy.AddField_management(forest_surface, "Area_total", "DOUBLE")
```

```
with arcpy.da.UpdateCursor(forest_surface, ["Area_total"]) as cursor:
    for row in cursor:
        row[0] = total_area
        cursor.updateRow(row)

# Assigns the values of spacing to separate variables.
spacings = [float(x) for x in spacing.split() if x.lower() != 'x']
spacing_1 = spacings[0]
spacing_2 = spacings[1]

# Creates a fishnet with labels (points) for the given spacing and
# extent. The points are the representation of the trees.
arcpy.CreateFishnet_management(out_feature, origin_point, y_axis,
                               spacing_1, spacing_2, "", "", "",
                               "LABELS", xtent)

# Erases fishnet, leaving only the feature containing the points
# (labels), wich ends with _label.
desc_out = arcpy.Describe(out_feature)

# Creates an intersection feature based on the forest_extension and the
# points of the trees (_label).
path = desc_out.Path
catalog_path = desc_out.catalogPath
basename = desc_out.baseName
data_type = arcpy.Describe(catalog_path).dataType
if data_type not in ("FeatureLayer", "FeatureClass"):
    labels = path + "\\" + basename + "_label.shp"
else:
    labels = path + "\\" + basename + "_label"

intersect = "in_memory" + "\\" + 'intersect'
```

```
arcpy.Intersect_analysis([forest_surface, labels], intersect, "ALL",
                         "", "POINT")

# Saves the result of the intersection, that was in the memory, as a
# file. The file has its fields mapped so that it contains only the
# area and ID fields.
forest_surface_id = ('FID_' + desc_surface.baseName)
fmps1 = arcpy.FieldMappings()
fmps1.addTable(intersect)
field_list = [str(forest_surface_id), 'AreaTalhao', 'Area_total']
for field in fmps1.fields:
    if field.name not in field_list:
        fmps1.removeFieldMap(
            fmps1.findFieldMapIndex(field.name))

out_name = arcpy.Describe(out_feature).file
arcpy.FeatureClassToFeatureClass_conversion(intersect, path, out_name,
                                            field_mapping=fmps1)

# Erases the _label feature and the intersect in memory, leaving only
# the feature containing the intersect (path + out_name).
arcpy.Delete_management(labels)
arcpy.Delete_management(intersect)

# Creates and calculates the Spacing field.
arcpy.AddField_management(out_feature, "Espacament", "TEXT")
with arcpy.da.UpdateCursor(out_feature, ["Espacament"]) as cursor:
    for row in cursor:
        row[0] = spacing
        cursor.updateRow(row)

# Creates and alculates the field containing the number of trees per
```

```
# hectare at time zero (creation of the forest, without any mortality).
n_trees_t0 = int(arcpy.GetCount_management(out_feature).getOutput(0))
N_ha = round(n_trees_t0 / total_area, 0)
arcpy.AddField_management(out_feature, "N_t0", "FLOAT")

with arcpy.da.UpdateCursor(out_feature, ["N_t0"]) as cursor:
    for row in cursor:
        row[0] = N_ha
        cursor.updateRow(row)

# Adds and calculates a field for age.
arcpy.AddField_management(out_feature, "Idade_t1", "FLOAT")

with arcpy.da.UpdateCursor(out_feature, ["Idade_t1"]) as cursor:
    for row in cursor:
        row[0] = age
        cursor.updateRow(row)

# Adds a field for dbh.
arcpy.AddField_management(out_feature, "dap_t1", "FLOAT")

# Definition of eq parameters.
if location_prm == "#" or not location_prm:
    location_prm = 0.0

location = location_prm
scale = scale_prm
form = form_prm

# Definition of the models.
def Weibull(location, scl, form, p):
    return weibull_min.ppf(p, form, loc=location, scale=scl)
```

```
def Normal(mean, standard_deviation, p):
    return norm.ppf(p, loc=mean, scale=standard_deviation)

def Gamma(scl, form, p):
    return gamma.ppf(p, a=form, scale=scl)

# Calculation of dbh.
oid_list = [k[0] for k in
            sorted(arcpy.da.SearchCursor(out_feature, ['OID@']))]

if eq_type == 'Weibull':
    dbh_list = [Weibull(location, scale, form, random.random())
                for k in oid_list]
elif eq_type == 'Normal':
    dbh_list = [Normal(location, scale, random.random())
                for k in oid_list]
elif eq_type == 'Gamma':
    dbh_list = [Gamma(scale, form, random.random())
                for k in oid_list]

# Base folder adress.
folder = arcpy.Describe(out_feature).path
desc = arcpy.Describe(folder)
while desc.dataType != 'Folder':
    folder = desc.path
    desc = arcpy.Describe(folder)

# Creation of a base surface with spatial correlation for allocating
# all dbh.
# Step 1: Create a point on each vertice of the forest surface extent.
out_cell_size = (spacing_1 * spacing_2)**(0.5)
```

```
vertices_names = ['1', '2', '3', '4']
xtent_list_float = [float(k) for k in xtent_list]
vertices_coords = [(xtent_list_float[0] - 3*out_cell_size,
                    xtent_list_float[1] - 3*out_cell_size),
                   (xtent_list_float[0] - 3*out_cell_size,
                    xtent_list_float[3] + 3*out_cell_size),
                   (xtent_list_float[2] + 3*out_cell_size,
                    xtent_list_float[1] - 3*out_cell_size),
                   (xtent_list_float[2] + 3*out_cell_size,
                    xtent_list_float[3] + 3*out_cell_size)]

vertices_list = [(k, i[0], i[1]) for k, i in
                 zip(vertices_names, vertices_coords)]

fields_and_datatype = {'names': ('point',
                                 'coord_x',
                                 'coord_y'),
                       'formats': ('a1',
                                   np.dtype(float),
                                   np.dtype(float))}

vertices_array = np.rec.fromrecords(vertices_list,
                                    dtype=fields_and_datatype)

out_table = "in_memory" + "\\" + 'out_table'
arcpy.da.NumPyArrayToTable(vertices_array, out_table)
arcpy.MakeXYEventLayer_management(out_table, 'coord_x',
                                  'coord_y', 'xtent_vertices')

# Step 2: Creates random points for each 1 ha of the forest surface,
# maintaining a minimum distance between points.
out_dissolve = folder + "\\" + 'out_dissolve.shp'
```

```
arcpy.Dissolve_management(forest_surface, out_dissolve)
n_rand_points = int(total_area*points_ha)
arcpy.CreateRandomPoints_management(
                            folder, 'rand_points', out_dissolve,
                            number_of_points_or_field=n_rand_points,
                            minimum_allowed_distance=min_dist)

# Step 3: Merge the extent points and random points into a feature
# called surface_points.
surface_points = folder + "\\" + 'surface_points.shp'
rand_points = folder + "\\" + 'rand_points.shp'
fmps2 = arcpy.FieldMappings()
fmps2.addTable('xtent_vertices')
fmps2.addTable(rand_points)
for field in fmps2.fields:
    if field.name not in ['dap_t1']:
        fmps2.removeFieldMap(
            fmps2.findFieldMapIndex(field.name))

arcpy.Merge_management(['xtent_vertices', rand_points],
                       surface_points, fmps2)

# Step 4: Adds a field for values in the surface_points and erases
# unecessary features.
arcpy.AddField_management(surface_points, "value", "DOUBLE")
arcpy.Delete_management(rand_points)
arcpy.Delete_management(out_dissolve)
arcpy.Delete_management(out_table)

# Step 5: Creates values for the points in surface_points, according
# to the diameter distribution, and generate the raster surface using
# Natural Neighbour interpolation. Erases the surface_points.
```

```
if eq_type == 'Weibull':
    srf_points_dbh = {k[0]: Weibull(location, scale, form,
                                    random.random())
                      for k in arcpy.da.SearchCursor(surface_points,
                                                     ['OID@'])}
elif eq_type == 'Normal':
    srf_points_dbh = {k[0]: Normal(location, scale,
                                   random.random())
                      for k in arcpy.da.SearchCursor(surface_points,
                                                     ['OID@'])}
elif eq_type == 'Gamma':
    srf_points_dbh = {k[0]: Gamma(scale, form,
                                  random.random())
                      for k in arcpy.da.SearchCursor(surface_points,
                                                     ['OID@'])}

with arcpy.da.UpdateCursor(surface_points,
                           ['OID@', 'value']) as cursor:
    for row in cursor:
        row[1] = srf_points_dbh[row[0]]
        cursor.updateRow(row)

nn_raster = folder + "\\" + 'NN_raster'
outNN = arcpy.sa.NaturalNeighbor(surface_points, 'value',
                                 out_cell_size)

outNN.save(nn_raster)
arcpy.Delete_management(surface_points)

# Extracts the values from the raster to a table, together with the
# corresponding OIDs from the point feature of the trees. Erases the
# raster.
```

```
raster_table = nn_raster + "_table"
arcpy.ExtractValuesToTable_ga(out_feature, nn_raster, raster_table)
arcpy.Delete_management(nn_raster)

# Gives some heterogeneity to the values of the table from the raster.
if heterogeneity > 0:
    with arcpy.da.UpdateCursor(raster_table, ['VALUE']) as cursor:
        for row in cursor:
            value = abs(row[0])
            row[0] = np.random.triangular(value * (1 - heterogeneity),
                                          value,
                                          value * (1 + heterogeneity))
            cursor.updateRow(row)

# Makes a list of the values and OIDs from the raster_table. OIDs are
# then sorted based on the values. This will serve as base for
# allocating the dbh.
table_list = [(k[1], k[0]) for k in
              arcpy.da.SearchCursor(raster_table,
                                    ['SrcID_Feat', 'Value'])]

table_list.sort()
arcpy.Delete_management(raster_table)

# Dbh_list is also sorted.
dbh_list.sort()

# Now that the list from the raster (table_list) and the list of dbh
# (dbh_list) are sorted, assigns OIDs from table_list to the values of
# dbh. This assignment corresponds to the allocation of the dbh
# according to the previous raster.
final_dbh_oids = {k[1]: round(v, 2) for k, v in
```

```
                zip(table_list, dbh_list)}

# Fills the dbh field.
with arcpy.da.UpdateCursor(out_feature, ['OID@', 'dap_t1']) as cursor:
    for row in cursor:
        if row[1] in (None, 0):
            row[1] = final_dbh_oids[row[0]]
            cursor.updateRow(row)
        else:
            pass
            cursor.updateRow(row)

# Dead trees at the creation of dbh.
arcpy.AddMessage(u"Removendo \xe1rvores mortas")

# Takes a random sample of trees, based on the percentage of living
# trees.
desc = arcpy.Describe(out_feature)
data_type = arcpy.Describe(desc.catalogPath).dataType
n_trees_t1 = int(arcpy.GetCount_management(out_feature).getOutput(0))
ndead_trees = int(round((n_trees_t1 * (100 - alive_trees)) / 100.0, 0))
random_sample = random.sample(oid_list[1:], ndead_trees)
random_sample_str = ", ".join(str(k) for k in random_sample)

# Selects rows in the tree layer, based on the random sample that was
# previously defined, than sets the dbh of the selected rows to Null or
# zero depending on the data type.
if alive_trees < 100:
    selection_expression = ("" + desc.OIDFieldName + ' IN (' +
                            random_sample_str + ')' + "")

    arcpy.MakeFeatureLayer_management(out_feature, "temp_layer")
```

```
    arcpy.SelectLayerByAttribute_management("temp_layer",
                                            "NEW_SELECTION",
                                            selection_expression)

    if data_type not in ("FeatureLayer", "FeatureClass"):
        with arcpy.da.UpdateCursor("temp_layer",
                                   ['dap_t1']) as cursor:
                                        for row in cursor:
                                            row[0] = 0.00
                                            cursor.updateRow(row)
    else:
        with arcpy.da.UpdateCursor("temp_layer",
                                   ['dap_t1']) as cursor:
                                        for row in cursor:
                                            row[0] = None
                                            cursor.updateRow(row)

# Calculates the number of living trees / Calculates the total area /
# Calculates the number of trees per hectare / Adds the field for N/ha
# and calculates it.
N = len([k[0] for k in
        arcpy.da.SearchCursor(out_feature, ["dap_t1"])
        if k[0] not in (None, 0)])

N_ha = round(N / total_area, 0)
arcpy.AddField_management(out_feature, "N_" + "dap_t1", "FLOAT")

with arcpy.da.UpdateCursor(out_feature, ["N_" + "dap_t1"]) as cursor:
    for row in cursor:
        row[0] = N_ha
        cursor.updateRow(row)
return
```

```
class ProjectDiameters(object):
    def __init__(self):
        self.label = u"Projetar dap e sobreviv\xeancia"
        self.description = (u"Projeta o dap e a sobreviv\xeancia das "
                            u"\xe1rvores para uma idade futura.")
        self.canRunInBackground = False

    def getParameterInfo(self):
        trees = arcpy.Parameter(
            displayName=u"Layer contendo as \xe1rvores",
            name="trees",
            datatype="Feature Layer",
            parameterType="Required",
            direction="Input")
        trees.filter.list = ["Point"]

        eq_type = arcpy.Parameter(
            displayName="Tipo de modelo",
            name="eq_type",
            datatype="String",
            parameterType="Required",
            direction="Input")
        eq_type.filter.type = "ValueList"
        eq_type.filter.list = ["Lundqvist_Korf", "Schumacher", "Richards"]
        eq_type.value = "Richards"

        b1_dbh = arcpy.Parameter(
            displayName=u"Par\xe2metro \u03b2\u2081",
            name="b1_dbh",
            datatype="Double",
```

```
    parameterType="Required",
    direction="Input")
b1_dbh.value = 0.63327

b2_dbh = arcpy.Parameter(
    displayName=u"Par\xe2metro \u03b2\u2082",
    name="b2_dbh",
    datatype="Double",
    parameterType="Optional",
    direction="Input")
b2_dbh.value = 0.17115

b3_dbh = arcpy.Parameter(
    displayName=u"Par\xe2metro \u03b2\u2083",
    name="b3_dbh",
    datatype="Double",
    parameterType="Optional",
    direction="Input")
b3_dbh.value = 0.08032

initial_age_field = arcpy.Parameter(
    displayName="Idade inicial",
    name="initial_age_field",
    datatype="Field",
    parameterType="Required",
    direction="Input")
initial_age_field.parameterDependencies = [trees.name]

dbh_init_field = arcpy.Parameter(
    displayName="dap na idade inicial",
    name="dbh_init_field",
    datatype="Field",
```

```
    parameterType="Required",
    direction="Input",)
dbh_init_field.parameterDependencies = [trees.name]

years = arcpy.Parameter(
    displayName=u"Anos at\xe9 a idade final",
    name="years",
    datatype="Double",
    parameterType="Required",
    direction="Input")

interval = arcpy.Parameter(
    displayName=u"Intervalo entre proje\xe7\xf5es",
    name="interval",
    datatype="Double",
    parameterType="Required",
    direction="Input")
interval.value = 1

time_first_proj = arcpy.Parameter(
    displayName=u"Tempo na primeira proje\xe7\xe3o",
    name="time_first_proj",
    datatype="Double",
    parameterType="Required",
    direction="Input")
time_first_proj.filter.type = "Range"
time_first_proj.filter.list = [2.00, float('inf')]

ntrees_init_field = arcpy.Parameter(
    displayName=u"\xc1rvores por hectare na idade inicial",
    name="ntrees_init_field",
    datatype="Field",
```

```
    parameterType="Required",
    direction="Input")
ntrees_init_field.parameterDependencies = [trees.name]

surv_type = arcpy.Parameter(
    displayName=u"Tipo de sobreviv\xeancia",
    name="surv_type",
    datatype="String",
    parameterType="Required",
    direction="Input",
    category=u"Sobreviv\xeancia")
surv_type.type = "ValueList"
surv_type.filter.list = ["Porcentagem", "Modelo de Pienaar e Shiver"]
surv_type.value = "Modelo de Pienaar e Shiver"

surv_percent = arcpy.Parameter(
    displayName=u"Porcentagem de sobreviv\xeancia na idade final",
    name="surv_percent",
    datatype="Double",
    parameterType="Optional",
    direction="Input",
    category=u"Sobreviv\xeancia")
surv_percent.value = 100
surv_percent.filter.type = "Range"
surv_percent.filter.list = [1.00, 100.00]

total_area = arcpy.Parameter(
    displayName=u"\xc1rea total",
    name="total_area",
    datatype=["Field", "Double"],
    parameterType="Optional",
    direction="Input",
```

```
    category=u"Sobreviv\xeancia")
total_area.parameterDependencies = [trees.name]

b1_n = arcpy.Parameter(
    displayName=u"Par\xe2metro \u03b2\u2081",
    name="b1_n",
    datatype="Double",
    parameterType="Optional",
    direction="Input",
    category=u"Sobreviv\xeancia")
b1_n.value = 0.00382

b2_n = arcpy.Parameter(
    displayName=u"Par\xe2metro \u03b2\u2082",
    name="b2_n",
    datatype="Double",
    parameterType="Optional",
    direction="Input",
    category=u"Sobreviv\xeancia")
b2_n.value = 1.70542

het_index = arcpy.Parameter(
    displayName=u"\xcdndice de heterogeneidade",
    name="var_index",
    datatype="Double",
    parameterType="Required",
    direction="Input",
    category=u"Heterogeneidade")
het_index.value = 0.5
het_index.filter.type = "Range"
het_index.filter.list = [0.00, 1.00]
```

```
    params = [trees,
              eq_type,
              b1_dbh,
              b2_dbh,
              b3_dbh,
              initial_age_field,
              dbh_init_field,
              years,
              interval,
              time_first_proj,
              ntrees_init_field,
              surv_type,
              surv_percent,
              total_area,
              b1_n,
              b2_n,
              het_index]
    return params

def isLicensed(self):
    return True

def updateParameters(self, parameters):
    eq_type = parameters[1]
    b1_dbh = parameters[2]
    b2_dbh = parameters[3]
    b3_dbh = parameters[4]
    dbh_init_field = parameters[6]
    time_first_proj = parameters[9]
    surv_type = parameters[11]
    surv_percent = parameters[12]
    total_area = parameters[13]
```

```
b1_n = parameters[14]
b2_n = parameters[15]

# Converts time_first_proj to int.
if time_first_proj.value:
    time_first_proj.value = int(time_first_proj.value)

# Field validation according to what is inserted as eq_type.
if eq_type.value == "Schumacher":
    b1_dbh.enabled = True
    b2_dbh.value = None
    b2_dbh.enabled = False
    b3_dbh.value = None
    b3_dbh.enabled = False
elif eq_type.value == "Lundqvist_Korf":
    b1_dbh.enabled = True
    b2_dbh.enabled = True
    b3_dbh.value = None
    b3_dbh.enabled = False

elif eq_type.value == "Richards":
    b1_dbh.enabled = True
    b2_dbh.enabled = True
    b3_dbh.enabled = True
else:
    b1_dbh.value = None
    b1_dbh.enabled = False
    b2_dbh.value = None
    b2_dbh.enabled = False
    b3_dbh.value = None
    b3_dbh.enabled = False
```

```
# Checks if the two last characters of dbh_init_field name are int,
# sums it + 1 and sets them as default value for time_final_age
my_list = []
if dbh_init_field.value:
    for x in list(dbh_init_field.valueAsText[-2:]):
        try:
            my_list.append(int(x))
        except:
            pass

cond_1 = dbh_init_field.value
cond_2 = not time_first_proj.value
cond_3 = len(my_list) > 0
if cond_3:
    cond_4 = int("".join(str(x) for x in my_list))
cond_5 = time_first_proj.value
if cond_1 and cond_2 and cond_3:
    time_first_proj.value = cond_4 + 1

# If time_first_proj <= time at dbh_init_field, changes time_first_proj
# to a greater value than time at dbh_init_field.
if cond_1 and cond_3 and cond_5:
    if time_first_proj.value <= cond_4:
        time_first_proj.value = cond_4 + 1

# Field validation for survival
if surv_type.value == "Porcentagem":
    surv_percent.enabled = True
    b1_n.value = None
    b1_n.enabled = False
    b2_n.value = None
    b2_n.enabled = False
```

```
            if surv_percent.value == 100:
                total_area.enabled = False
                total_area.value = None
            else:
                total_area.enabled = True
        elif surv_type.value == "Modelo de Pienaar e Shiver":
            surv_percent.value = None
            surv_percent.enabled = False
            b1_n.enabled = True
            b2_n.enabled = True
        else:
            surv_percent.value = None
            surv_percent.enabled = False
            b1_n.value = None
            b1_n.enabled = False
            b2_n.value = None
            b2_n.enabled = False
        return

    def updateMessages(self, parameters):
        eq_type = parameters[1]
        b2_dbh = parameters[3]
        b3_dbh = parameters[4]
        surv_type = parameters[11]
        surv_percent = parameters[12]
        total_area = parameters[13]
        b1_n = parameters[14]
        b2_n = parameters[15]

        if eq_type.value:
            if eq_type.value in ("Lundqvist_Korf", "Richards"):
                if not b2_dbh.value and (b2_dbh.value != 0):
```

```
            b2_dbh.setErrorMessage(
                u"Par\xe2metro \xe9 necess\xe1rio para uso do modelo "
                u"selecionado.")
    if eq_type.value == "Richards":
        if not b3_dbh.value and (b3_dbh.value != 0):
            b3_dbh.setErrorMessage(
                u"Par\xe2metro \xe9 necess\xe1rio para uso do modelo "
                u"selecionado.")

if surv_type.value != "Porcentagem":
    if not total_area.value:
        total_area.setErrorMessage(
            u"\xc1rea total \xe9 necess\xe1ria para se calcular o "
            u"n\xfamero de \xe1rvores por hectare.")
    if not b1_n.value and (b1_n.value != 0):
        b1_n.setErrorMessage(
            u"Par\xe2metro \xe9 necess\xe1rio para uso do modelo "
            u"selecionado.")
    if not b2_n.value and (b2_n.value != 0):
        b2_n.setErrorMessage(
            u"Par\xe2metro \xe9 necess\xe1rio para uso do modelo "
            u"selecionado.")
else:
    if not surv_percent.value:
        surv_percent.setErrorMessage(
            u"\xc1rea total \xe9 necess\xe1ria para se calcular o "
            u"n\xfamero de \xe1rvores por hectare.")
    if surv_percent.value < 100:
        if not total_area.value:
            total_area.setErrorMessage(
                u"\xc1rea total \xe9 necess\xe1ria para se "
                u"calcular o n\xfamero de \xe1rvores por hectare.")
```

```
    return

def execute(self, parameters, messages):

    trees = parameters[0].valueAsText
    eq_type = parameters[1].valueAsText
    b1_dbh = parameters[2].value
    b2_dbh = parameters[3].value
    b3_dbh = parameters[4].value
    initial_age_field = parameters[5].valueAsText
    dbh_init_field = parameters[6].valueAsText
    years = parameters[7].value
    interval = parameters[8].value
    time_first_proj = parameters[9].value
    ntrees_init_field = parameters[10].valueAsText
    surv_type = parameters[11].valueAsText
    surv_percent = parameters[12].value
    total_area = parameters[13].valueAsText
    b1_n = parameters[14].value
    b2_n = parameters[15].value
    het_index = parameters[16].value

    arcpy.AddMessage("Preparando dados")

    # Definition of the dbh models.
    def Lundqvist_Korf(dbh1, a1, a2, b1, b2):
        return dbh1 * np.exp(b1/a1**b2 - b1/a2**b2)

    def Schumacher(dbh1, a1, a2, b1):
        return dbh1 * np.exp(b1/a1 - b1/a2)

    def Richards(dbh1, a1, a2, b1, b2, b3):
```

```
    return ((dbh1*(1-b1*np.exp(-b2*a2))**(1/(1-b3))) /
            (1-b1*np.exp(-b2*a1))**(1/(1-b3)))

# Definition of survival model
def Pienaar_Shiver(n1, a1, a2, b1, b2):
    return n1*np.exp(-b1*(a2**b2-a1**b2))

# At this point, some lists ans dictionaries have to be created to
# guide the calculations in the 'for' loop that comes in sequence.
# The loop is used because the projections may be performed more than
# once.

# The type of the data.
desc = arcpy.Describe(trees)
data_type = arcpy.Describe(desc.catalogPath).dataType

# The dictionary of the OID's and ages at time zero of the projections.
init_ages = {k[0]: k[1] for k in
             arcpy.da.SearchCursor(trees,
                                   ["OID@", initial_age_field])}

# The dictionary containing the age for each OID at each time in the
# projections (age_dic).
increments_list = [0]
increment = interval
while increment <= years:
    increments_list.append(increment)
    increment += interval
increments_array = np.array(increments_list)
age_dic = {k: increments_array + init_ages[k] for k in init_ages}

# The list with each time of the projections.
```

```
time_list = [k + int(time_first_proj) for k in
             range(len(increments_list) - 1)]

# The list with the names of the dbh fields for each time in the
# projections.
dbh_list = ["dap_t" + str(k) for k in time_list]
dbh_list.insert(0, dbh_init_field)

# The list with the names of the 'trees per hectare' fields for each
# time in the projections.
n_list = ["N_" + k for k in dbh_list[1:]]
n_list.insert(0, ntrees_init_field)

# Gets the value for the number os trees at time zero (n0).
with arcpy.da.SearchCursor(trees, [ntrees_init_field]) as cursor:
    for row in cursor:
        n0 = row[0]
        break

# Gets the value for the age time zero (a0).
a0_list = [k[0] for k in
           arcpy.da.SearchCursor(trees, [initial_age_field])
           if k[0] not in (None, 0)]

a0 = sum(a0_list) / len(a0_list)

# Gets the value for the age at final time (a_final).
a_final_list = [k[0] + increments_list[-1] for k in
                arcpy.da.SearchCursor(trees, [initial_age_field])
                if k[0] not in (None, 0)]

a_final = sum(a_final_list) / len(a_final_list)
```

```
# Fits a logarithmic curve based on the number of trees at initial
# age an the survival rate at final age.
cond1 = surv_type == 'Porcentagem'
cond2 = surv_percent < 100
if cond1 and cond2:
    n_final = surv_percent / 100 * n0
    xy_list = [(np.log(a0), n0), (np.log(a_final), n_final)]

    def Beta1(xy_list):
        sum_xy = sum([k[0]*k[1] for k in xy_list])
        sum_x = sum([k[0] for k in xy_list])
        sum_y = sum([k[1] for k in xy_list])
        sum_x2 = sum([k[0]**2 for k in xy_list])
        n = len(xy_list)
        b1 = (sum_xy - sum_x * sum_y / n) / (sum_x2 - sum_x**2 / n)
        return b1

    def Beta0(xy_list, b1):
        average_x = sum([k[0] for k in xy_list]) / len(xy_list)
        average_y = sum([k[1] for k in xy_list]) / len(xy_list)
        b0 = average_y - b1 * average_x
        return b0

    b1_n = Beta1(xy_list)
    b0_n = Beta0(xy_list, b1_n)

# Gets the value of the total area.
if total_area:
    try:
        total_area = float(total_area)
    except ValueError:
```

```
        with arcpy.da.SearchCursor(trees, [total_area]) as cursor:
            for row in cursor:
                total_area = row[0]
                break
else:
    total_area = 0.0

# The 'for' loop that executes the projections. It runs once for each
# time given in the time_list.
for i in range(len(time_list)):
    # Initial dbh field
    i_dbh_f = dbh_list[i]
    # Final dbh field
    dbh_field = dbh_list[i + 1]
    time = time_list[i]
    age_field = "Idade_t" + str(time)
    # Trees per hectare field
    n1_field = n_list[i]
    n2_field = n_list[i + 1]

    # Calculations of projected dbh. It involves the creation of a
    # dictionary with values of OID, initial age, final age and initial
    # dbh and uses it combined with the models to estimate the final
    # (projected) dbh.
    arcpy.AddMessage(u"Calculando di\xe2metros para " + dbh_field)

    # Dictionary containing the values of OID, initial age, final age
    # and initial dbh.
    age_dbh = {k[0]: (age_dic[k[0]][i], age_dic[k[0]][i + 1], k[1]) for
               k in arcpy.da.SearchCursor(trees, ["OID@", i_dbh_f])}

    # Calculation of the growth rate for each tree (dbh2 / dbh1).
```

```
if eq_type == "Lundqvist_Korf":
    growth = {k: (age_dbh[k][2],
                  Lundqvist_Korf(age_dbh[k][2], age_dbh[k][0],
                                 age_dbh[k][1], b1_dbh, b2_dbh) /
                  age_dbh[k][2])
              if age_dbh[k][2] not in (None, 0)
              else (age_dbh[k][2], age_dbh[k][2])
              for k in age_dbh}
elif eq_type == "Schumacher":
    growth = {k: (age_dbh[k][2],
                  Schumacher(age_dbh[k][2], age_dbh[k][0],
                             age_dbh[k][1], b1_dbh) /
                  age_dbh[k][2])
              if age_dbh[k][2] not in (None, 0)
              else (age_dbh[k][2], age_dbh[k][2])
              for k in age_dbh}
elif eq_type == "Richards":
    growth = {k: (age_dbh[k][2],
                  Richards(age_dbh[k][2], age_dbh[k][0],
                           age_dbh[k][1], b1_dbh, b2_dbh, b3_dbh) /
                  age_dbh[k][2])
              if age_dbh[k][2] not in (None, 0)
              else (age_dbh[k][2], age_dbh[k][2])
              for k in age_dbh}

# Definition of a formula to apply of a disturbance in the growth
# rate of each tree.
def Disturbance(growth_rate, het_index):
    # Recalculates the growth rate. The new growth rate will be
    # a random value from a normal distribution with the original
    # rate as the mean. The standard deviation is defined so that
    # 99.7% of the possible values of the distribution are between
```

```
    # 1.0 and 2 * growt_rate - 1. The values of the randomized
    # probability inside the normal distribution are limited so
    # that 100% of the results are within the condition above.
    deviation = (growth_rate - 1) / 3.0
    p_min = norm.cdf(1.0, loc=growth_rate, scale=deviation)
    p_min = (0.5 - p_min) * (1 - het_index) + p_min
    p_max = 1 - p_min
    new_rate = norm.ppf(random.uniform(p_min, p_max),
                        loc=growth_rate, scale=deviation)
    return new_rate

# Calculation of the projected dbh (prj_dbh). The 'if' statement is
# there to maintain the values of dbh of the dead trees as zero or
# None.
prj_dbh = {k: round(Disturbance(growth[k][1], het_index) *
                    growth[k][0], 2)
           if growth[k][0] not in (None, 0)
           else growth[k][0] for k in growth}

# Adds and calculates a field for age.
arcpy.AddField_management(trees, age_field, "FLOAT")

with arcpy.da.UpdateCursor(trees, ["OID@", age_field]) as cursor:
    for row in cursor:
        row[1] = age_dic[row[0]][i + 1]
        cursor.updateRow(row)

# Adds and fills the field for dbh.
arcpy.AddField_management(trees, dbh_field, "FLOAT")
with arcpy.da.UpdateCursor(trees, ["OID@", dbh_field]) as cursor:
    for row in cursor:
        row[1] = prj_dbh[row[0]]
```

```
        cursor.updateRow(row)

# Projection survival. When using the Pienaar and Shiver model, it
# takes the average age of the trees (a1 and a2) as input.
arcpy.AddMessage(u"Projetando a sobreviv\xeancia")

# Values for average initial age (a1) and average final age (a2).
initial_age_list = [age_dbh[k][0] for k in age_dbh]
final_age_list = [age_dbh[k][1] for k in age_dbh]
a1 = sum(initial_age_list) / len(initial_age_list)
a2 = sum(final_age_list) / len(final_age_list)

# Calculation of the number of trees per hectare at initial age
# (n1), trees per hectare at final age (n2) and the total number of
# dead trees at final age (ndead_trees).
if surv_type == "Porcentagem":
    with arcpy.da.SearchCursor(trees, [n1_field]) as cursor:
        for row in cursor:
            n1 = row[0]
            break

    if surv_percent < 100:
        n2 = b0_n + b1_n * np.log(a2)
    else:
        n2 = n1
elif surv_type == "Modelo de Pienaar e Shiver":
    n1 = Pienaar_Shiver(n0, a0, a1, b1_n, b2_n)
    n2 = Pienaar_Shiver(n1, a1, a2, b1_n, b2_n)

ndead_trees = int(round((n1 - n2) * total_area, 2))

# Takes a random sample of OIDs from living trees, based on the
```

```
# number of dead trees (ndead_trees).
living_trees = [k[0] for k in
                sorted(arcpy.da.SearchCursor(trees,
                                             ['OID@', dbh_field]))
                if k[1] not in (None, 0)]
random_sample = random.sample(living_trees[1:], ndead_trees)
random_sample_str = ", ".join(str(k) for k in random_sample)

# Selects rows in the tree layer, based on the random sample that
# was previously defined, than sets the dbh of the selected rows to
# Null orzero depending on the data type.
if random_sample_str is not "":
    selection_expression = ("" + desc.OIDFieldName + ' IN (' +
                            random_sample_str + ')' + "")

    arcpy.MakeFeatureLayer_management(trees, "temp_layer")
    arcpy.SelectLayerByAttribute_management("temp_layer",
                                            "NEW_SELECTION",
                                            selection_expression)

    if data_type not in ("FeatureLayer", "FeatureClass"):
        with arcpy.da.UpdateCursor("temp_layer",
                                   [dbh_field]) as cursor:
            for row in cursor:
                row[0] = 0.00
                cursor.updateRow(row)
    else:
        with arcpy.da.UpdateCursor("temp_layer",
                                   [dbh_field]) as cursor:
            for row in cursor:
                row[0] = None
                cursor.updateRow(row)
```

```
            arcpy.AddField_management(trees, n2_field, "FLOAT")
            with arcpy.da.UpdateCursor(trees, [n2_field]) as cursor:
                for row in cursor:
                    row[0] = round(n2, 0)
                    cursor.updateRow(row)
        return


class EstimateHeight(object):
    def __init__(self):
        self.label = "Estimar altura"
        self.description = (u"Estima a altura das \xe1rvores baseada no dap e "
                            u"idade.")
        self.canRunInBackground = False

    def getParameterInfo(self):
        trees = arcpy.Parameter(
            displayName=u"Layer contendo as \xe1rvores",
            name="trees",
            datatype="Feature Layer",
            parameterType="Required",
            direction="Input")
        trees.filter.list = ["Point"]

        eq_type = arcpy.Parameter(
            displayName="Tipo de modelo",
            name="eq_type",
            datatype="String",
            parameterType="Required",
            direction="Input")
        eq_type.filter.type = "ValueList"
```

```
eq_type.filter.list = ["Lundqvist_Korf", "Schumacher",
                       "Gompertz", "Weibull"]
eq_type.value = "Weibull"

dependent_variable = arcpy.Parameter(
    displayName=u"Tipo de vari\xe1vel dependente",
    name="dependent_variable",
    datatype="String",
    parameterType="Required",
    direction="Input")
dependent_variable.filter.type = "ValueList"
dependent_variable.filter.list = ["dap", "dap * Idade"]
dependent_variable.value = "dap * Idade"

b0_prm = arcpy.Parameter(
    displayName=u"Par\xe2metro \u03b2\u2080",
    name="b0_prm",
    datatype="Double",
    parameterType="Required",
    direction="Input")
b0_prm.value = 27.79485

b1_prm = arcpy.Parameter(
    displayName=u"Par\xe2metro \u03b2\u2081",
    name="b1_prm",
    datatype="Double",
    parameterType="Required",
    direction="Input")
b1_prm.value = 20.78744

b2_prm = arcpy.Parameter(
    displayName=u"Par\xe2metro \u03b2\u2082",
```

```
    name="b2_prm",
    datatype="Double",
    parameterType="Optional",
    direction="Input")
b2_prm.value = 0.02690

b3_prm = arcpy.Parameter(
    displayName=u"Par\xe2metro \u03b2\u2083",
    name="b3_prm",
    datatype="Double",
    parameterType="Optional",
    direction="Input")
b3_prm.value = 0.90071

dbh_fields = arcpy.Parameter(
    displayName="dap",
    name="dbh_fields",
    datatype="Field",
    parameterType="Required",
    direction="Input",
    multiValue=True)
dbh_fields.parameterDependencies = [trees.name]

age_fields = arcpy.Parameter(
    displayName="Idade",
    name="age_fields",
    datatype="Field",
    parameterType="Optional",
    direction="Input",
    multiValue=True)
age_fields.parameterDependencies = [trees.name]
```

```
    het_index = arcpy.Parameter(
        displayName=u"\xcdndice de heterogeneidade",
        name="var_index",
        datatype="Double",
        parameterType="Required",
        direction="Input",
        category=u"Heterogeneidade")
    het_index.value = 0.20
    het_index.filter.type = "Range"
    het_index.filter.list = [0.00, 0.99]

    params = [trees,
              eq_type,
              dependent_variable,
              b0_prm,
              b1_prm,
              b2_prm,
              b3_prm,
              dbh_fields,
              age_fields,
              het_index]
    return params

def isLicensed(self):
    return True

def updateParameters(self, parameters):
    eq_type = parameters[1]
    dependent_variable = parameters[2]
    b2_prm = parameters[5]
    b3_prm = parameters[6]
    age_fields = parameters[8]
```

```
    # Field validation
    if dependent_variable.value == "dap * Idade":
        age_fields.enabled = True
    elif dependent_variable.value == "dap":
        age_fields.enabled = False

    if eq_type.value == "Weibull":
        b2_prm.enabled = True
        b3_prm.enabled = True
    elif eq_type.value in ("Lundqvist_Korf", "Gompertz"):
        b2_prm.enabled = True
        b3_prm.value = None
        b3_prm.enabled = False
    elif eq_type.value == "Schumacher":
        b2_prm.value = None
        b2_prm.enabled = False
        b3_prm.value = None
        b3_prm.enabled = False

def updateMessages(self, parameters):
    eq_type = parameters[1]
    dependent_variable = parameters[2]
    b2_prm = parameters[5]
    b3_prm = parameters[6]
    dbh_fields = parameters[7]
    age_fields = parameters[8]

    if dbh_fields.value:
        dbh_fields_list = dbh_fields.valueAsText.split(";")
    if age_fields.value:
        age_fields_list = age_fields.valueAsText.split(";")
```

```
    if dependent_variable.value == "dap * Idade":
        if not age_fields.value:
            age_fields.setErrorMessage(
                u"Campo(s) de idade s\xe3o necess\xe1rios quando a "
                u"vari\xe1vel dependente \xea 'dap * Idade'")
        if dbh_fields.value and age_fields.value:
            if len(dbh_fields_list) != len(age_fields_list):
                dbh_fields.setErrorMessage(
                    u"N\xfamero de campos de dap e Idade deve ser o mesmo")
                age_fields.setErrorMessage(
                    u"N\xfamero de campos de dap e Idade deve ser o mesmo")

    if eq_type.value:
        if eq_type.value is "Weibull" or "Lundqvist_Korf" or "Gompertz":
            if not b2_prm.value and (b2_prm.value != 0):
                b2_prm.setErrorMessage(
                    u"Par\xe2metro \xe9 necess\xe1rio para uso do modelo "
                    "selecionado.")
        if eq_type.value == "Weibull":
            if not b3_prm.value and (b3_prm.value != 0):
                b3_prm.setErrorMessage(
                    u"Par\xe2metro \xe9 necess\xe1rio para uso do modelo "
                    "selecionado.")
    return

def execute(self, parameters, messages):
    trees = parameters[0].valueAsText
    eq_type = parameters[1].valueAsText
    dependent_variable = parameters[2].valueAsText
    b0_prm = parameters[3].value
    b1_prm = parameters[4].value
```

```
b2_prm = parameters[5].value
b3_prm = parameters[6].value
dbh_fields = parameters[7].valueAsText
age_fields = parameters[8].valueAsText
het_index = parameters[9].value

dbh_fields_list = dbh_fields.split(";")

# Creates a dummy value for age_fields, so the calculations of height
# can work when dependent variable is only dbh.
if age_fields == "#" or not age_fields:
    age_fields = len(dbh_fields_list) * "OID@;"
    age_fields = age_fields[:-1]
age_fields_list = age_fields.split(";")

# Calculation of a 'seed' for each tree. The seed is an input value to
# the calculation of pseudo-random numbers, used when establishing a
# disturbance on the estimates of height. It could be any number, as
# long as it is a unique value for each tree.
seeds = {k[0]: k[1][0] + k[1][1] for k in
         arcpy.da.SearchCursor(trees, ['OID@', 'SHAPE@XY'])}

# Definition of the models. The models are conditioned by the dependent
# variable being dbh or dbh*Age.
def Lundqvist_Korf(b0, b1, b2, dbh, age):
    if dependent_variable == "dap * Idade":
        return b0*np.exp(-b1/(dbh*age)**b2)
    else:
        return b0*np.exp(-b1/dbh**b2)

def Schumacher(b0, b1, dbh, age):
    if dependent_variable == "dap * Idade":
```

```
        return b0*np.exp(-b1/(dbh*age))
    else:
        return b0*np.exp(-b1/dbh)

def Gompertz(b0, b1, b2, dbh, age):
    if dependent_variable == "dap * Idade":
        return b0*np.exp(-b1*np.exp(-b2*(dbh*age)))
    else:
        return b0*np.exp(-b1*np.exp(-b2*dbh))

def Weibull(b0, b1, b2, b3, dbh, age):
    if dependent_variable == "dap * Idade":
        return b0-b1*np.exp(-b2*(dbh*age)**b3)
    else:
        return b0-b1*np.exp(-b2*dbh**b3)

# Definition of a formula to apply of a disturbance on the height of
# each tree.
def Disturbance(seed, height, het_index):
    # Recalculates the height of the tree (new_height). The new_height
    # will be a value from a normal distribution with the original
    # height as the mean. The value of std is defined so that 99.7% of
    # the possible values of the distribution are between the minimum
    # and maximum desired new_height. The seed can be any number and it
    # is used as an input to generate a pseudo-random number 'p'. The
    # pseudo-random value generated for a particular seed is always the
    # same. According to the 'seeds' calculation performed previously,
    # each value of seed used here is unique for each tree. Therefore,
    # the pseudo-random number generated for each seed is always the
    # same. 'p' is the probability used to get the new value of height
    # (new_height) in the normal distribution and is limited so the
    # minimum possible new_height is height * (1 - het_index).
```

```
    ht_min = height * (1 - het_index)
    std = (height - ht_min) / 3.0
    p_min = norm.cdf(ht_min, loc=height, scale=std)
    p_max = 1 - p_min
    random.seed(seed)
    p = random.uniform(p_min, p_max)
    new_height = norm.ppf(p, loc=height, scale=std)
    return new_height

# The 'for' loop that estimates the heights. It runs once for each
# item in dbh_fields_list.
for dbhfield, agefield in zip(dbh_fields_list, age_fields_list):
    # Dictionary of OIDs, age and dbh.
    dbh_dic = {k[0]: (k[1], k[2]) for k in
               arcpy.da.SearchCursor(trees,
                                     ["OID@", agefield, dbhfield])}

    arcpy.AddMessage("Estimando alturas de " + dbhfield)

    # Dictionary of OIDs and estimated heights.
    if eq_type == "Lundqvist_Korf":
        heights = {k: Lundqvist_Korf(b0_prm, b1_prm, b2_prm,
                                     dbh_dic[k][1], dbh_dic[k][0])
                   if dbh_dic[k][1] not in (None, 0)
                   else dbh_dic[k][1] for k in dbh_dic}
    elif eq_type == "Schumacher":
        heights = {k: Schumacher(b0_prm, b1_prm,
                                 dbh_dic[k][1], dbh_dic[k][0])
                   if dbh_dic[k][1] not in (None, 0)
                   else dbh_dic[k][1] for k in dbh_dic}
    elif eq_type == "Gompertz":
        heights = {k: Gompertz(b0_prm, b1_prm, b2_prm,
```

```
                                    dbh_dic[k][1], dbh_dic[k][0])
                        if dbh_dic[k][1] not in (None, 0)
                        else dbh_dic[k][1] for k in dbh_dic}
            elif eq_type == "Weibull":
                heights = {k: Weibull(b0_prm, b1_prm, b2_prm, b3_prm,
                                    dbh_dic[k][1], dbh_dic[k][0])
                        if dbh_dic[k][1] not in (None, 0)
                        else dbh_dic[k][1] for k in dbh_dic}

            # Estimation of heights. The 'if' statement is there to maintain
            # the values of height of the deadtrees as zero or None.
            new_heights = {k: round(Disturbance(seeds[k], heights[k],
                                                het_index), 2)
                        if heights[k] not in (None, 0)
                        else heights[k] for k in heights}

            # Adds field for height
            height_field = "Ht_" + dbhfield
            arcpy.AddField_management(trees, height_field, "FLOAT")

            # Fills field of height
            with arcpy.da.UpdateCursor(trees, ['OID@',
                                               height_field]) as cursor:
                for row in cursor:
                    row[1] = new_heights[row[0]]
                    cursor.updateRow(row)
        return


class EstimateVolume(object):
    def __init__(self):
        self.label = "Estimar volume"
```

```
    self.description = (u"Estima o volume das \xe1rvores baseado no dap e "
                        "altura.")
    self.canRunInBackground = False

def getParameterInfo(self):
    trees = arcpy.Parameter(
        displayName=u"Layer contendo as \xe1rvores",
        name="trees",
        datatype="Feature Layer",
        parameterType="Required",
        direction="Input")
    trees.filter.list = ["Point"]

    eq_type = arcpy.Parameter(
        displayName="Tipo de estimativa",
        name="eq_type",
        datatype="String",
        parameterType="Required",
        direction="Input")
    eq_type.filter.type = "ValueList"
    eq_type.filter.list = ["Modelo Schumacher & Hall", "Fator de forma"]
    eq_type.value = "Modelo Schumacher & Hall"

    b0_prm = arcpy.Parameter(
        displayName=u"Par\xe2metro \u03b2\u2080",
        name="b0_prm",
        datatype="Double",
        parameterType="Optional",
        direction="Input")
    b0_prm.value = -10.44771

    b1_prm = arcpy.Parameter(
```

```
    displayName=u"Par\xe2metro \u03b2\u2081",
    name="b1_prm",
    datatype="Double",
    parameterType="Optional",
    direction="Input")
b1_prm.value = 1.86852

b2_prm = arcpy.Parameter(
    displayName=u"Par\xe2metro \u03b2\u2082",
    name="b2_prm",
    datatype="Double",
    parameterType="Optional",
    direction="Input")
b2_prm.value = 1.17904

ff = arcpy.Parameter(
    displayName="Fator de forma",
    name="ff",
    datatype="Double",
    parameterType="Optional",
    direction="Input")

dbh_fields = arcpy.Parameter(
    displayName="dap",
    name="dbh_fields",
    datatype="Field",
    parameterType="Required",
    direction="Input",
    multiValue=True)
dbh_fields.parameterDependencies = [trees.name]

h_fields = arcpy.Parameter(
```

```
        displayName="Altura",
        name="h_fields",
        datatype="Field",
        parameterType="Required",
        direction="Input",
        multiValue=True)
    h_fields.parameterDependencies = [trees.name]

    params = [trees,
              eq_type,
              b0_prm,
              b1_prm,
              b2_prm,
              ff,
              dbh_fields,
              h_fields]
    return params

def isLicensed(self):
    return True

def updateParameters(self, parameters):
    eq_type = parameters[1]
    b0_prm = parameters[2]
    b1_prm = parameters[3]
    b2_prm = parameters[4]
    ff = parameters[5]

    # Form validation
    if eq_type.value == "Modelo Schumacher & Hall":
        ff.value = None
        ff.enabled = False
```

```
            b0_prm.enabled = True
            b1_prm.enabled = True
            b2_prm.enabled = True
        else:
            ff.enabled = True
            b0_prm.value = None
            b0_prm.enabled = False
            b1_prm.value = None
            b1_prm.enabled = False
            b2_prm.value = None
            b2_prm.enabled = False
        return

    def updateMessages(self, parameters):
        eq_type = parameters[1]
        b0_prm = parameters[2]
        b1_prm = parameters[3]
        b2_prm = parameters[4]
        ff = parameters[5]
        dbh_fields = parameters[6]
        h_fields = parameters[7]

        if dbh_fields.value:
            dbh_fields_list = dbh_fields.valueAsText.split(";")
        if h_fields.value:
            h_fields_list = h_fields.valueAsText.split(";")

        if dbh_fields.value and h_fields.value:
            if len(dbh_fields_list) != len(h_fields_list):
                dbh_fields.setErrorMessage(
                    u"N\xfamero de campos de dap e altura deve ser o mesmo")
                h_fields.setErrorMessage(
```

```
                u"N\xfamero de campos de dap e altura deve ser o mesmo")

        if eq_type.value == "Modelo Schumacher & Hall":
            if not b0_prm.value and (b0_prm.value != 0):
                b0_prm.setErrorMessage(
                    u"Par\xe2metro \xe9 necess\xe1rio para uso do modelo "
                    "selecionado.")
            if not b1_prm.value and (b1_prm.value != 0):
                b1_prm.setErrorMessage(
                    u"Par\xe2metro \xe9 necess\xe1rio para uso do modelo "
                    "selecionado.")
            if not b2_prm.value and (b2_prm.value != 0):
                b2_prm.setErrorMessage(
                    u"Par\xe2metro \xe9 necess\xe1rio para uso do modelo "
                    "selecionado.")
        else:
            if not ff.value:
                ff.setErrorMessage(
                    u"Fator de forma \xe9 necess\xe1rio para c\xe1lculo do "
                    "volume.")
        return

    def execute(self, parameters, messages):
        trees = parameters[0].valueAsText
        eq_type = parameters[1].valueAsText
        b0_prm = parameters[2].value
        b1_prm = parameters[3].value
        b2_prm = parameters[4].value
        ff = parameters[5].value
        dbh_fields = parameters[6].valueAsText
        h_fields = parameters[7].valueAsText
```

```
dbh_fields_list = dbh_fields.split(";")
h_fields_list = h_fields.split(";")

# Definition of the models.
def Schumacher_Hall(b0, b1, b2, dbh, height):
    return np.exp(b0 + b1 * np.log(dbh) + b2 * np.log(height))

def Form_factor(ff, dbh, height):
    return (np.pi * dbh ** 2 / 40000) * height * ff

# Adds fields for volume and calculates it.
for dbhfield, hfield in zip(dbh_fields_list, h_fields_list):
    volume_field = "V_" + dbhfield
    arcpy.AddField_management(trees, volume_field, "FLOAT")
    if eq_type == "Modelo Schumacher & Hall":
        volumes = {k[0]: round(Schumacher_Hall(b0_prm, b1_prm, b2_prm,
                                               k[1], k[2]), 4)
                   if k[1] not in (None, 0) else k[1] for k in
                   arcpy.da.SearchCursor(trees,
                                         ['OID@', dbhfield, hfield])}
    elif eq_type == "Fator de forma":
        volumes = {k[0]: round(Form_factor(ff, k[1], k[2]), 4)
                   if k[1] not in (None, 0) else k[1] for k in
                   arcpy.da.SearchCursor(trees,
                                         ['OID@', dbhfield, hfield])}

    with arcpy.da.UpdateCursor(trees,
                               ['OID@', volume_field]) as cursor:
        for row in cursor:
            row[1] = volumes[row[0]]
            cursor.updateRow(row)
return
```

```
class dbhClasses(object):
    def __init__(self):
        self.label = "Classificar dap"
        self.description = (u"Gera o centro de classe de di\xe2metro para "
                            "cada dap.")
        self.canRunInBackground = False

    def getParameterInfo(self):
        trees = arcpy.Parameter(
            displayName=u"Layer contendo as \xe1rvores",
            name="trees",
            datatype="Feature Layer",
            parameterType="Required",
            direction="Input")
        trees.filter.list = ["Point"]

        class_range = arcpy.Parameter(
            displayName="Intervalo de classe",
            name="class_range",
            datatype="Double",
            parameterType="Required",
            direction="Input")
        class_range.value = 2

        minimum_dbh = arcpy.Parameter(
            displayName=u"dap m\xednimo",
            name="minimum_dbh",
            datatype="Double",
            parameterType="Optional",
            direction="Input")
```

```
    minimum_dbh.value = 4

    dbh_fields = arcpy.Parameter(
        displayName="dap",
        name="dbh_fields",
        datatype="Field",
        parameterType="Required",
        direction="Input",
        multiValue=True)
    dbh_fields.parameterDependencies = [trees.name]

    params = [trees,
              class_range,
              minimum_dbh,
              dbh_fields]
    return params

def isLicensed(self):
    return True

def updateParameters(self, parameters):
    return

def updateMessages(self, parameters):
    return

def execute(self, parameters, messages):
    trees = parameters[0].valueAsText
    class_range = parameters[1].value
    minimum_dbh = parameters[2].value
    dbh_fields = parameters[3].valueAsText
```

```
dbh_fields_list = dbh_fields.split(";")

# List containing all smallest values of dbh.
small_dbh_list = []
big_dbh_list = []
for field in dbh_fields_list:
    dbh_list = [k[0] for k in arcpy.da.SearchCursor(trees, [field])
                if k[0] not in (None, 0)]

    small_dbh_list.append(min(dbh_list))
    big_dbh_list.append(max(dbh_list))

# Value for minimum_dbh if it is left empty and value for smallest_dbh.
if minimum_dbh == "#" or not minimum_dbh:
    minimum_dbh = 1
    smallest_dbh = min(small_dbh_list)
else:
    smallest_dbh = minimum_dbh

# Creates a list with the upper limits of the diameter classes.
first_class = int(smallest_dbh + class_range)
last_class = int(max(big_dbh_list) + class_range + 1)
c_range = int(class_range)
range_of_limits = range(first_class, last_class, c_range)
class_limits = [k for k in range_of_limits]

# Definition of the function that classifies dbh.
def dbh_class(dbh):
    if dbh < minimum_dbh:
        return minimum_dbh + class_range / 2.0
    else:
        for k in class_limits:
```

```
            if dbh < k and (dbh >= k - class_range):
                return k - class_range / 2.0

for field in dbh_fields_list:
    arcpy.AddField_management(trees, field + "_C", "FLOAT")
    with arcpy.da.UpdateCursor(trees, [field, field + "_C"]) as cursor:
        for row in cursor:
            if row[0] in (None, 0):
                row[1] = row[0]
                cursor.updateRow(row)
            else:
                row[1] = dbh_class(row[0])
                cursor.updateRow(row)
return
```