Uma Aplicação Desktop Interativa para Desenvolvedores e Usuários de Modelos de Sistemas Dinâmicos em R

Bruno Henrique Pereira[1], *Cristina Freitas Bazzano*[1],
Luís Gustavo Barionni[1], *Adauto Luiz Mancini*[1], *Márcio Nicolau*[2]

[1] Embrapa, Av. André Tosello, nº 209 Campus da Unicamp,
Barão Geraldo, Campinas, São Paulo, Brasil
hbrunopereira@gmail.com, crisfbazz@gmail.com,
luis.barioni@embrapa.br, adauto.mancini@embrapa.br

[2]Embrapa, BR 285 Km 294, Passo Fundo, Rio Grande do Sul, Brasil
marcio.nicolau@embrapa.br

RESUMO

A aplicação apresentada neste artigo propõe facilitar o desenvolvimento e uso de simuladores de modelos de sistemas dinâmicos contínuos. Seu ambiente, intuitivo e didático, permite a definição e simulação dos modelos de forma rápida e organizada. Os modelos desenvolvidos podem ser salvos em arquivos, possibilitando a criação de repositórios e o compartilhamento de simuladores. Assim, a aplicação permite que usuários de diversas áreas do conhecimento, tal como a agropecuária, colaborem no processo de desenvolvimento e simulação de modelos de sistemas dinâmicos.

PALAVRAS-CHAVE: Modelagem e simulação, Compartilhamento de Código, RShiny.

ABSTRACT

The application presented herein proposes to facilitate the development and use of simulators of continuous dynamic system models. Its environment, intuitive and didactic, allows the definition and simulation of the models in a fast and organized way. The developed models can be saved in files, making it possible to create repositories and share simulators. Thus, the application allows users of different areas of knowledge, such as agriculture and livestock, to collaborate in the process of development and simulation of dynamic system models.

KEYWORDS: Modeling and simulation, Code-sharing, RShiny.

INTRODUÇÃO

R é uma linguagem para computação estatística e gráficos (R Core Team, 2017). O ambiente colaborativo do R permite extensão de sua funcionalidade por meio de pacotes desenvolvidos por uma ampla comunidade de pesquisadores e desenvolvedores, incluindo novos algoritmos e técnicas gráficas. Os pacotes deSolve (SOETAERT; PETZOLDT; SETZER, 2010), Simecol (PETZOLDT; RINKE, 2007), Rodeo (KNEIS, 2017) e sdsim dão suporte à implementação de modelos de sistemas dinâmicos (MSDs) contínuos, baseados em sistemas de equações diferenciais ordinárias (EDOs) de primeira ordem.

O desenvolvimento e uso de simuladores de MSDs requer equipes de profissionais com diferentes perfis. Desde profissionais ligados à computação até especialistas de domínio, que utilizam modelos apenas para compreensão do comportamento de um sistema, desinteressados pelos detalhes de sua implementação (FREUA et al., 2014).

Na agropecuária a modelagem de sistemas tornou-se essencial ao longo das últimas décadas. A partir de experimentos baseados em simulações de MSDs é possível entender e predizer o comportamento e performance do sistema real, facilitando a tomada de decisões e a adequação a diversos cenários. Essa abordagem pode ser usada, por exemplo, para prever o crescimento bovino a partir de dados sobre sua dieta. (JONES et al., 2016).

O pacote R sdsim, desenvolvido pelos autores deste artigo, utiliza uma moderna abordagem de implementação orientada a objetos para sistemas dinâmicos (SD) com rigorosa separação do código e algoritmos de solução do modelo e é o único, atualmente, que dá suporte explícito ao desenvolvimento de modelos acoplados. O pacote sdsim também suporta o armazenamento e posterior e recuperação de modelos, o que viabiliza e facilita a distribuição de simuladores, possibilitando que pesquisadores usem o pacote como um meio de comunicação na implementação de MSDs.

A especificação e o uso de modelos criados com o pacote sdsim em script R é intuitivo para usuários com conhecimento intermediário na linguagem. Porém a prototipação de modelos com auxílio de uma aplicação com interface gráfica de usuário pode reduzir drasticamente a curva de aprendizado e a ocorrência de erros de especificação.

O objetivo deste trabalho foi desenvolver uma aplicação, por meio do framework Shiny e do pacote sdsim, para a prototipagem, execução e compartilhamento de simuladores de MSDs atômicos de forma simples e acessível para usuários de diferentes áreas.

MATERIAL E MÉTODOS

A aplicação aborda todo o processo de definição de MSDs atômicos, da especificação até a execução de simulações. Esses modelos são especificados por variáveis de entrada (e.g. parâmetros, constantes, *inputs* e *switches*), séries temporais, variáveis auxiliares e variáveis de estado, EDOs de primeira ordem, funções opcionais de inicialização e pós processamento, e funções associadas ao disparo e execução de eventos.

Shiny

A aplicação foi desenvolvida por meio do framework Shiny (CHANG et al., 2017), um *framework* para criação de interfaces web interativas associadas a códigos desenvolvidos em R. O Pacote Shiny utiliza programação funcional reativa em seu desenvolvimento, idéia esta obtida de outro *framework* web chamada Meteor (Meteor Development Group, 2016). Utilizou-se pacotes adicionais, específicos para o Shiny, como o shinydashboard (CHANG; RIBEIRO, 2017) para o layout da aplicação, o shinyAce (ALLEN, 2016) para gerar campos de edição de código e o rhandsontable (OWEN, 2016) para gerar planilhas editáveis.

sdsim

O pacote sdsim provê a funcionalidade necessária para execução de simulações. Esse pacote possui suporte a eventos, diversos métodos de integração numérica, processamento dos dados de entrada (e.g. séries temporais, variáveis auxiliares, função de inicialização de parâmetros), e exibição de resultados em tabela e gráficos.

As funções e variáveis dos modelos informadas na aplicação são encapsuladas em objetos da classe SDModel do pacote sdsim. A aplicação usa os métodos *Run* e *Plot* dos objetos instanciados. O método *Run* usa o método de integração numérica, o tempo e o passo da simulação informados na aplicação para a execução da simulação do modelo, e retorna os dados resultantes. O método *Plot*, gera os gráficos do resultado da simulação a partir das configurações de plotagem definidas na aplicação.

Especificação de Modelos

A representação de um modelo atômico (M) utilizada neste trabalho é definida por meio da tupla 1. Os elementos obrigatórios da tupla foram representados por meio do símbolo * (asterisco).

$$M = \{S*, C, P, W, I, A, T, \delta d*, \delta i, \delta R, \delta E, \delta G\} \quad (1)$$

Onde,

- S (*state*) é o vetor de variáveis de estado, contendo o estado inicial do modelo;
- C (*const*) é o vetor de constantes do modelo (imutáveis entre simulações e inacessíveis para calibração estatística);
- P (*parms*) é o vetor de parâmetros do modelo (constantes somente ao longo de uma simulação);
- W (*switches*) é o vetor de variáveis discretas que podem ser utilizadas como seletores no modelo;
- I (*inputs*) é o vetor contendo variáveis de entrada;
- T (*timeSeries*) é o vetor contendo as variáveis de séries temporais, cujos valores dependem do tempo da simulação;

- A (*aux*) é o vetor de variáveis auxiliares, cujos valores são equações dependentes de outras variáveis do sistema avaliadas a cada passo de simulação;
- δd (*Differential Equations*) é uma função que define o vetor das diferenciais das variáveis de estado do sistema;
- δi (*Parameter Initialization*) é a uma função, executada antes do início da simulação, que pode ser usada para estabelecer valores iniciais para os parâmetros e para as variáveis de estado;
- δR (*Root*) é uma função que verifica as condições para a ocorrência de um evento;
- δE (*Events*) é uma função que define a resposta a um determinado evento;
- δG (*Global Functions*) é um vetor de funções adicionais que podem ser executadas dentro do escopo de quaisquer funções definidas no modelo.

Variáveis

Os MSDs representados contêm um conjunto de variáveis de entrada (parâmetros, constantes, *inputs*, *switches*), de séries temporais, de saída, de estado e auxiliares relacionadas entre si por meio das equações diferenciais de primeira ordem. Essas variáveis são definidas por meio de planilhas editáveis.

Essas variáveis são definidas por meio de planilhas editáveis, que possuem as seguintes colunas: (a) *Variable*, armazena o nome das variáveis; (b) *Value*, armazena o valor das variáveis. (c) *Unit* (opcional)), armazena a unidade de medida das variáveis e; (d) *Description* (opcional), armazena breves descrições das variáveis.

A coluna *Value* é preenchida de maneira diferente em algumas planilhas. Nas planilhas *State*, *Const*, *Parms*, *Inputs* e *Switches* cada variável possui um valor estático. Na planilha das variáveis auxiliares a coluna *Value* deve ser preenchida com equações escritas na linguagem R. Essas fórmulas serão computadas em cada passo da simulação. A planilha que define as séries temporais se diferencia por conter duas colunas adicionais: (a) *Times*, armazena um vetor com os tempos das séries temporais; (b) *Interpolation*, armazena o método de interpolação (funções degrau, linear ou spline). Além disso, para essas variáveis a coluna *Value* deve ser preenchida com um vetor de valores. Esses valores correspondem a cada tempo do vetor definido na coluna *Times*, formando uma matriz de valor por tempo que descreve a série temporal. Se o campo *Times* não for preenchido, *Value* deve conter apenas um valor e será considerado constante para todos os tempos de simulação.

Funções

As funções necessárias para a especificação do simulador são escritas em linguagem R, e recebem como argumento as variáveis definidas nas planilhas. Esses argumentos podem ser: (a) *t*, que contém o tempo atual da simulação; (b) *state*, que contém as variáveis de estado; (c) *vars*, que contém as listas *const*, *parms*, *inputs* e *switches*; (d) *timeSeries*, que contém as variáveis de

séries temporais, cujo valor é automaticamente calculado para o tempo *t* da simulação. (e) *aux*, que contém as variáveis auxiliares, cujo valor é automaticamente calculado para o tempo *t* da simulação.

Armazenamento de modelos

O compartilhamento de um modelo especificado com a aplicação é suportado por meio do formato nativo RData, que permite encapsular todos os dados de um modelo em um único arquivo. O arquivo RData armazenado contém um objeto de simulação da classe SDModel, e por isso pode ser carregado e executado via comandos em ambiente R utilizando o pacote sdsim.

A aplicação também permite armazenar o modelo em um projeto R. Isso é útil por fornecer a flexibilidade da linguagem R e toda a funcionalidade do pacote sdsim. O projeto R armazenado possui: (a) arquivos com os parâmetros de entrada; (b) um arquivo de configurações para gráficos; (c) um arquivo com configurações de diretórios e formatos de dados e; (d) um script que descreve as funções do modelo e instancia o objeto de simulação usando o pacote sdsim. O projeto gerado pode então ser executado em ambiente R a partir da recuperação de um objeto de simulação.

Detecção de erros

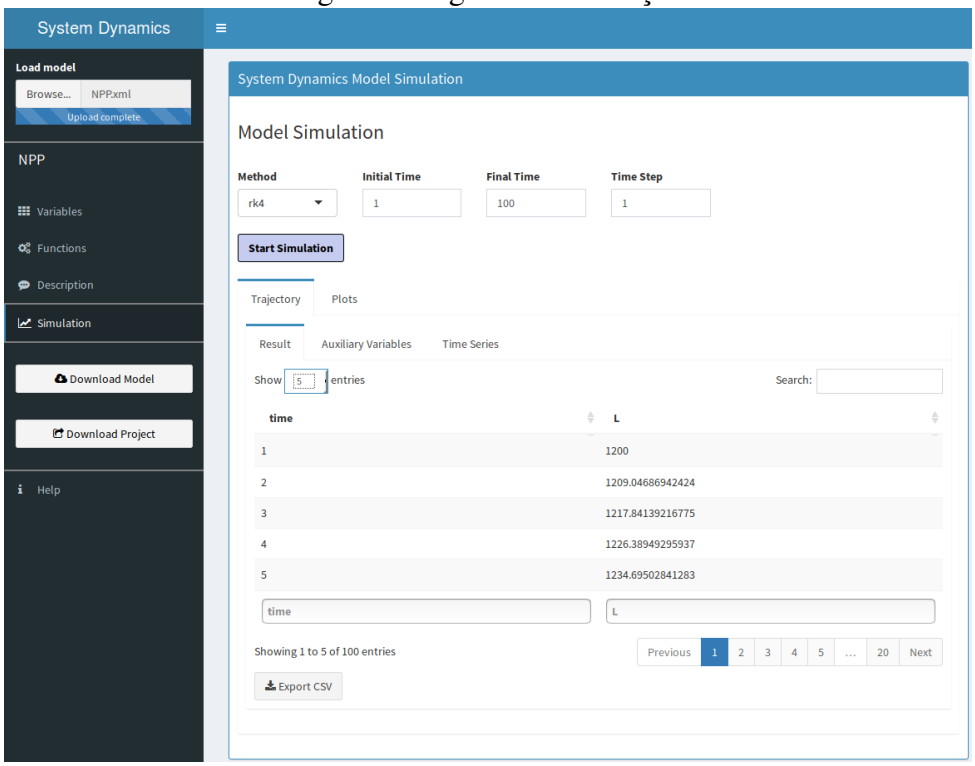
A aplicação é beneficiada pela funcionalidade de deteção de erros inclusa no pacote sdsim. Esta deteção de erros consiste em: (a) checagem de tipo de parâmetros, que notifica o usuário caso algum dos parâmetros tenha sido fornecido de forma incorreta; (b) detecção de valores nulos, ausentes ou infinitos, que podem indicar a presença de erros nos cálculos ou parâmetros do modelo; (c) visualização de dados intermediários para verificar o funcionamento do simulador.

RESULTADOS E DISCUSSÃO

A aplicação (Figura 1) apresenta toda a estrutura de especificação intuitivamente organizada, sendo necessário apenas preencher os campos correspondentes. Para facilitar o uso, instruções de preenchimento foram organizadas em breves descrições que acompanham cada campo de edição. Além disso, um menu de ajuda oferece um tutorial sobre como utilizar a aplicação.

O exemplo utilizado nas figuras é uma simplificação do modelo de crescimento de massa foliar de uma pastagem. A taxa de crescimento (dL) da massa foliar (L) do modelo é influenciada positivamente pela radiação solar diária, pela partição de fotoassimilados e pela quantidade de água disponível no solo (W). Usando o pacote sdsim um modelo acoplado pode ser criado a partir do modelo de crescimento foliar e um modelo de água disponível no solo, que calcula a variação de W ao decorrer da simulação. Neste exemplo o valor de W é constante de 50mm durante a simulação, pois a aplicação só suporta modelos atômicos, e L possui valor inicial de 1200 kg/ha.

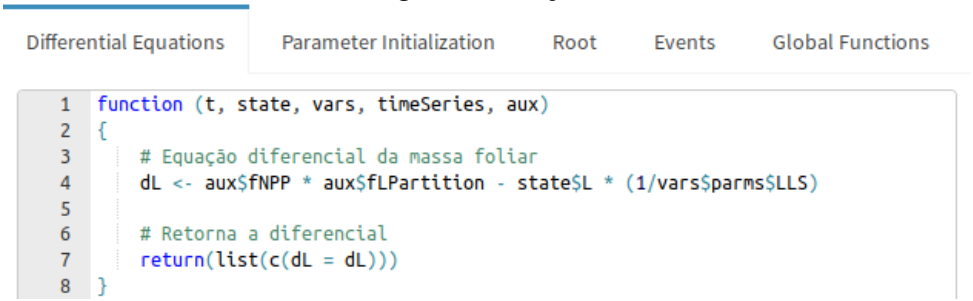
Figura 1: Página de Simulação



Especificação de funções

As funções são preenchidas em campos para edição (Figura 2). Esses campos possuem numeração de linhas, coloração do texto seguindo a formatação de código R, predição de palavras, entre outras funcionalidades que facilitam a definição das funções em linguagem R.

Figura 2: Funções



Especificação de variáveis

As variáveis de um modelo, disponíveis no menu Variables, são definidas usando as planilhas (Figuras 3, 4, 5 e 6) editáveis, onde cada linha representa uma variável.

Figura 3: Variável de Estado

	Variable	Value	Unit	Description
1	L	1200	kg/ha	Massa foliar

Figura 4: Variáveis de Entrada

	Variable	Value	Unit	Description
1	W	50	mm	Água em estoque
2	Q	18	MJ/m^2/day	Radiação global

Figura 5: Parâmetros

	Variable	Value	Unit	Description
1	SLA	0,002857143	m2/kg	Área foliar específica
2	k	0,5		Coeficiente de extinção da radiação
3	SWconst	0,7		As constantes são: 0,7 para areia, 0,6 para silte arenoso, 0,5 para silte argiloso, 0,4 para argila
4	epsilon	8	gDM/MJ	Eficiência de uso da radiação fotossinteticamente ativa
5	y	0,47		Coeficiente de respiração
6	Lmax	3000	kg/ha	Massa foliar máxima
7	LLS	50	dias	Tempo de vida da folha
8	SWpower	9		A expoente na equação para modificadores SW: 9 para areia, 7 para silte arenoso, 5 para silte argiloso e 3 para argila
9	Wmax	80	mm	Capacidade máxima de armazenamento de água do solo

Figura 6: Variáveis Auxiliares

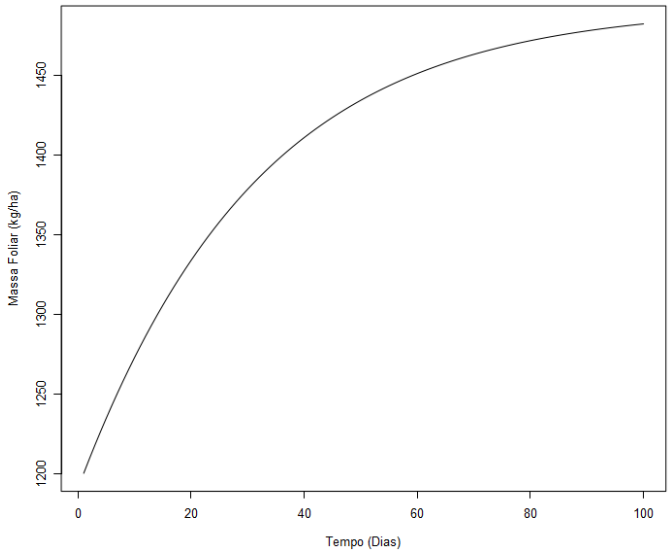
	Variable	Value	Unit	Description
1	fLAI	stateSL * varsSparmsSSLA		Índice da Área da Folha
2	fLIF	1 - exp(-varsSparmsSk * auxSfLAI)		Índice de Intercepção de Luz
3	fRADint	varsSinputsSQ * auxSfLIF		Radiação Interceptada
4	fWRatio	min((varsSinputsSW/varsSparmsSWmax), 1)		Razão entre estoque e capacidade máxima de água no solo
5	fSW	1/(1 + ((1 – auxSfWRatio)/varsSparmsSSWconst)^varsSparmsSSWpower)		Absorção de água pelo solo
6	fGPP	varsSparmsSepsilon * auxSfSW * auxSfRADint		Produção Primária Bruta
7	fNPP	auxSfGPP * varsSparmsSy		Produção Primária líquida
8	fLPartition	(varsSparmsSLmax – stateSL)/varsSparmsSLmax		Partição de Nutrientes

Execução de simulações

A execução do simulador é feita por meio do menu '*Simulation*'. O usuário escolhe o método de integração numérica e informa o tempo e o passo da simulação. Os métodos de integração disponíveis são os mesmos disponibilizados pelo pacote deSolve (e.g. euler, radau, runge-kutta, adams). Quando o botão '*Start Simulation*' é pressionado a simulação do modelo é executada e a trajetória das variáveis de estado, das variáveis auxiliares e das séries temporais são exibidas em uma tabela (Figura 1). Todas as trajetórias podem ser exportadas para arquivos em formato *csv* (valores separados por vírgula), que podem facilmente ser visualizados em planilhas de cálculo como o Calc e o Excel, por exemplo.

Os resultados podem ser exibidos graficamente (Figura 7) correlacionando as variáveis contidas nas trajetórias ou em função do tempo de simulação. Esse comportamento pode ser definido através da planilha de configuração de gráficos. Caso nenhuma configuração gráfica seja definida, por padrão, apenas as variáveis de estado serão projetadas em relação ao tempo.

Figura 7: Trajetória da Massa Foliar



Documentação de modelos

O menu *Description* contém um campo de texto para documentação de detalhes sobre o modelo e seu funcionamento. Esse mecanismo facilita o entendimento de modelos compartilhados por terceiros e permite seu armazenamento em repositórios.

Salvar e carregar modelos

Um modelo criado pela aplicação pode ser armazenado em formato de arquivo RData usando o botão '*Download Model*'. Este botão abre uma janela em que o usuário pode informar o nome do arquivo e baixar o modelo. Após armazenado, o modelo pode ser carregado de volta à aplicação por meio do menu '*Load Model*'.

Exportação de projetos

Um modelo pode ser exportado para formato de um projeto R, que pode ser armazenado por meio de um arquivo compactado (*zip*) usando o botão '*Download Project*'. Este botão abre uma janela em que o usuário pode informar o nome do projeto, configurar formatos de arquivo para os parâmetros do modelo (planilhas em formato excel ou formato *csv*), e baixar o projeto.

CONCLUSÕES

A aplicação desenvolvida apresenta vários benefícios para a modelagem e simulação de sistemas agropecuários, pois ela proporciona um ambiente amigável, organizado e intuitivo para o desenvolvimento de simuladores de modelos de sistemas dinâmicos contínuos. Seu uso facilita o aprendizado de modelagem e não requer conhecimento avançado de programação; além de

promover documentação, a expressividade e a manutenção de modelos de maior porte sem se apoiar na construção de diagramas complexos e extensos.

Além disso, o uso da aplicação acelera o processo de desenvolvimento de simuladores. Para criar um simulador usando a linguagem R é necessária uma estrutura por meio da qual as variáveis serão declaradas, as funções especificadas e o simulador inicializado e executado. A aplicação cria essa estrutura automaticamente, sendo necessário apenas o preenchimento de cada campo que compõe o modelo, simplificando o desenvolvimento e favorecendo a padronização, a checagem e a auditoria do modelo.

Os modelos criados pela aplicação possuem uma estrutura bem definida e padronizada. Esses modelos podem ser documentados, encapsulados em objetos e armazenados em arquivos no formato RData. Esse método facilita a portabilidade, o entendimento, o reuso e a manutenção de modelos.

A partir dos modelos armazenados em RData e projetos R também é possível criar repositórios de compartilhamento de modelos. Esses modelos podem então ser executados, calibrados e modificados de acordo com a necessidade de cada usuário. Isso também possibilita que esses modelos sejam carregados e manipulados em scripts R, fornecendo aos modelos maior funcionalidade do sdsim, como o acoplamento de modelos, assim como todos os outros recursos disponíveis para a linguagem R.

Trabalhos futuros incluem adição de funcionalidades na aplicação, e.g. definir séries temporais por meio de arquivos e permitir calibração de modelos. Pretende-se hospedar a aplicação em um servidor web, permitindo seu uso em ambiente online, sem necessitar da instalação local do R e do pacote sdsim, e adicionar suporte ao acoplamento de modelos.

AGRADECIMENTOS

Agradecimentos aos membros envolvidos no desenvolvimento deste trabalho e à Embrapa Informática Agropecuária pela oportunidade concedida.

REFERÊNCIAS

ALLEN, J. *Ace Editor Bindings for Shiny*. [S.l.], 2016. Disponível em: <https://CRAN.R-project.org/package=shinyAce>.

CHANG, W. et al. *Web Application Framework for R*. [S.l.], 2017. Disponível em: <https://CRAN.R-project.org/package=shiny>.

CHANG, W.; RIBEIRO, B. B. *Create Dashboards with 'Shiny'*. [S.l.], 2017. Disponível em: <https://CRAN.R-project.org/package=shinydashboard>.

FREUA, M. C. et al. A comparison between three different approaches to omplement a system dynamic model: an assessment by a multidisciplinary team. *Embrapa Informática*

Agropecuária. Boletim de pesquisa e desenvolvimento, 36, 2014. ISSN 1677-9266. Disponível em: <https://ainfo.cnptia.embrapa.br/digital/bitstream/item/117682/1/Livro-BolPesq36.pdf>.

JONES, J. W. et al. Brief history of agricultural systems modeling. 2016. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0308521X16301585>.

KNEIS, D. *A Code Generator for ODE-Based Models*. [S.l.], 2017. Disponível em: <https://CRAN.R-project.org/package=rodeo>.

Meteor Development Group. *Meteor: Build Apps with JavaScript*. [S.l.], 2016. Disponível em: <https://www.meteor.com>.

OWEN, J. *Interface to the 'Handsontable.js' Library*. [S.l.], 2016. Disponível em: <https://CRAN.R-project.org/package=rhandsontable>.

PETZOLDT, T.; RINKE, K. simecol: An object-oriented framework for ecological modeling in r. *Journal of Statistical Software*, v. 22, n. 9, p. 1–31, 2007. ISSN 1548-7660. Disponível em: <http://www.jstatsoft.org/v22/i09>.

R Core Team. *R: A Language and Environment for Statistical Computing*. Vienna, Austria, 2017. Disponível em: <https://www.R-project.org>.

SOETAERT, K.; PETZOLDT, T.; SETZER, R. Solving differential equations in r: Package desolve. *Journal of Statistical Software, Articles*, v. 33, n. 9, p. 1–25, 2010. ISSN 1548-7660. Disponível em: <https://www.jstatsoft.org/v033/i09>.