
# sdsim: Um Pacote para Modelagem e Simulação de Sistemas Dinâmicos Contínuos em R

*Cristina Freitas Bazzano*[1], *Bruno Henrique Pereira*[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**

Este artigo apresenta o pacote sdsim, um framework livre para definição, prototipação e simulação de modelos de sistemas dinâmicos contínuos em R, e sua aplicação. Por meio de uma abordagem orientada a objetos, o pacote define uma arquitetura genérica e estruturada para facilitar a implementação de simuladores e o compartilhamento de modelos. A integração com interface gráfica simplifica e torna amigável o uso do pacote por usuários com diferentes níveis. Seu suporte inovador ao acoplamento de modelos permite a definição de sistemas complexos que interagem entre si. Como resultado, o sdsim conta com um repositório de modelos que podem auxiliar no desenvolvimento de novos modelos e fomentar o uso do pacote na comunidade de modelagem e simulação em diversas áreas do conhecimento.

**PALAVRAS-CHAVE:** Equação Diferencial, Programação orientada a objetos, Compartilhamento de código.

**ABSTRACT**

This paper presents the sdsim package, a free framework for defining, prototyping and simulating continuous system dynamics models in R, and it's application. Through an object-oriented approach, the package defines a generic and structured architecture to facilitate the implementation of simulators and the sharing of models. The integration with a graphical interface simplifies and makes a user friendly environment for different users levels of the package. It's innovative support for coupling models allows the definition of complex systems that interact with each other. As a result, sdsim has a repository of models that can assist in the development

of new models and foster the use of the package in the modeling and simulation community.
**KEYWORDS:** Differential Equation, Object-oriented programming, Code-sharing.

## INTRODUÇÃO

Dinâmica de sistemas ou "System Dynamics" (SD) é um método formal de análise e modelagem de sistemas introduzido por Forrester (1961). Esse método alia rigor e generalidade para representação e simulação de sistemas contínuos complexos (STERMAN, 2000) de forma a permitir melhor avaliação de opções e a engenharia de sistemas mais eficientes. O paradigma de SD assume que a dinâmica é determinada por seu estado presente, i.e. o estado atual reflete toda a informação necessária sobre a trajetória pregressa (passado) do sistema que possa influenciar sua trajetória futura.

Em geral, aplicam-se regras iterativas que definem as mudanças do estado do sistema (JOST, 2005). Em modelos de sistemas dinâmicos (MSDs) contínuos, essa regra matemática é definida por um sistema de equações diferenciais ordinárias (EDOs) de primeira ordem, onde cada equação define a taxa de variação de uma variável de estado. Modelos atômicos são implementados como componentes elementares de um software de simulação. Em contrapartida, modelos acoplados são especificados como estruturas contendo um conjunto de sub-modelos (atômicos ou acoplados) com conexões entre si e com entradas e saídas do modelo acoplado (ZEIGLER; KIM; PRAEHOFER, 2000).

Esse método abriu possibilidades para o surgimento de novos modelos que possuem funções que descrevem as mudanças nos estados do sistema, em resposta a ações externas e outros componentes do sistema. Nos sistemas agropecuários iniciou-se uma nova era de nutrição dinâmica, levando a melhores tipos de modelos, incluindo de culturas, pecuária e agricultura (JONES et al., 2016). A simulação desses modelos predizem, por exemplo, o crescimento diário de uma planta baseado em dados meteorológicos diários e informações do solo, da gestão e da genética da planta (GRAEFF et al., 2012). Algumas dessas informações podem ser obtidas via acoplamento, por exemplo, de modelos de balanço hídrico e de nutrientes no solo. Na pecuária, a simulação também pode predizer o crescimento diário do animal baseado, por exemplo, em uma dieta contínua e na interação com um modelo de crescimento do pasto que pode ser estático, contínuo ou acoplado.

Existe uma miríade de aplicações voltadas à implementação de MSDs com foco em diferentes públicos. Entretanto, diversos critérios definem a ferramenta mais apropriada para o tipo de desenvolvimento e equipe (FREUA et al., 2014). O R, uma linguagem e ambiente de computação científica, destaca-se nesse contexto, por: (a) ser de domínio público; (b) ser compatível com as principais ferramentas do mercado; (c) ser suportado nos principais sistemas operacionais; (d) possuir suporte à entrada, manipulação, análise e visualização de dados; (e) ser uma linguagem de programação funcional com suporte a programação orientada a objetos (OO); (f) possuir elegante suporte à diversos algoritmos para solução de EDOs, por meio do

pacote deSolve (SOETAERT; PETZOLDT; SETZER, 2010), e; (g) ser open-software de natureza, o que facilita o entendimento e a troca de informação entre membros da comunidade (MATLOFF, 2011).

Embora alguns pacotes já se proponham a facilitar a implementação de MSDs em R, tais como o Simecol (PETZOLDT; RINKE, 2007) e o Rodeo (KNEIS, 2017), a implementação de modelos mais complexos, continua sendo dificultado pela rigidez das representações utilizadas atualmente. O pacote sdsim, objetivando suprir as dificuldades encontradas na prototipação e simulação de MSD, utiliza uma abordagem diferenciada, particularmente por ampliar a generalidade, definição e armazenamento dos modelos, suportar acoplamento de modelos e possibilitar a integração com interface gráfica intuitiva construída automaticamente para prototipação e execução de modelos atômicos.

O uso crescente do pacote sdsim pode fomentar a criação de repositórios de modelos na padronização proposta, o que contribuirá para o estudo e aplicação de MSDs às mais diferentes áreas das ciências e da engenharia, inclusive aos sistemas agropecuários.

## MATERIAL E MÉTODOS

Com o objetivo de oferecer uma arquitetura genérica para implementação de MSDs contínuos, dentro do paradigma de EDOs de primeira ordem, um design orientado a objetos (OO) foi aplicado. A estrutura dessa arquitetura ajuda na abstração dos simuladores ao encapsular em uma única estrutura de dados toda a especificação de um MSDs (suas equações, variáveis, parâmetros, etc) e possuir métodos que possibilitam a simulação do modelo instanciado.

Utilizou-se a estrutura de classe do pacote R6 (CHANG, 2017) do R, que é, dentre as possibilidades de escrever código OO no R, aquela que melhor atende às necessidades da arquitetura proposta. Suas capacidades comuns a outras linguagens OO, tais como, atributos e métodos públicos e privados (possibilitando encapsulamento de funcionalidades) e herança (superclasses que possibilitam suporte futuro a mais regras matemáticas), apresentam um ótimo desempenho e justificam sua escolha.

As classe criadas com o R6, e descritas abaixo, padronizam a implementação de simuladores ao encapsular toda a informação necessária para simular um MSD em um único objeto, e dão suporte ao acoplamento de modelos. O compartilhamento desses objetos promove a distribuição e reuso de simuladores de forma simples e compacta, permitindo que modeladores usem o sdsim como um meio de comunicação na implementação de MSDs.

Para oferecer usabilidade na prototipação, simulação e reuso de modelos de sistemas dinâmicos, uma interface gráfica de usuário (GUI) foi construída para o pacote. Nessa aplicação todas as funcionalidades disponíveis para modelos atômicos são apresentadas de maneira visual e clara, o que facilita a interação com todos os tipos de usuários, acelerando a curva de aprendizagem e contribuindo para a quebra da barreira de acesso às novas tecnologias sobre modelagem e simulação.

Além da estrutura para armazenamento dos MSDs também são necessárias rotinas matemáticas para integração temporal das EDOs que modulam o sistema e possibilitam sua simulação. Tais rotinas são implementadas por diversos algoritmos do pacote R deSolve, usado neste projeto para resolver os problemas de valor inicial de um sistemas de ODEs de primeira ordem. As funções do deSolve também oferecem suporte a eventos (mudanças temporais nas variáveis do sistema quando desejado) e permitem o uso de código compilado para um melhor desempenho na rotina de integração.

### *Classe SDModel*

A classe *SDModel* pode armazenar em seus membros toda a especificação de um MSD atômico, tais como as variáveis de entrada (e.g. parâmetros, constantes, etc.), as variáveis de estado, as equações diferenciais e auxiliares, funções opcionais de inicialização e pós processamento, e funções opcionais associadas ao disparo e execução de eventos. A função de inicialização pode ser utilizada, por exemplo, para uma inicialização customizada de variáveis de estado e parâmetros do sistema. As funções associadas a eventos podem ser utilizadas para determinar em que condições um evento ocorre e qual a resposta a esse evento, como, por exemplo, alterações abruptas no valor das variáveis de estado do sistema. A rotina de pós-processamento pode ser utilizada para processar estatisticamente os dados de trajetória das variáveis simuladas ou calcular novas variáveis e indicadores.

Um objeto *SDModel* também contém métodos para simulação de um MSD e para visualização do resultado. Na integração do sistema a regra matemática utilizada é a função que calcula as equações diferenciais e o estado atual do sistema é informado nas variáveis de estado. As equações auxiliares são pré avaliadas a cada passo da simulação para serem utilizadas, juntamente com as variáveis de entrada e o estado atual, no cálculo das EDOs do sistema.

No resultado da simulação também são armazenadas, caso existam, as trajetórias das variáveis auxiliares e das séries temporais do modelo, possibilitando a visualização e checagem de resultados intermediários da simulação.

### *Criar um objeto SDModel*

No pacote sdsim, foi implementada a seguinte função construtora para instanciar um objeto *SDModel*: *CreateSDModel(Nome do Modelo Atômico, Função que Calcula as EDOs do Modelo, Função de Inicialização, Função de Pós Processamento, Função que Dispara Eventos, Função que Executa Eventos)*. Esta função cria um objeto *SDModel* com identificação *Nome do Modelo Atômico* que deve especificar pelo menos a *Função que Calcula as EDOs do Modelo*. A especificação das equações do sistema é feita via funções R, o que requer um conhecimento básico da linguagem de script. No script, as variáveis do modelo são inicializadas pelo método *InitiModel* do objeto *SDModel* criado e são informados via dados em memória ou em arquivos. As variáveis podem estar em arquivos texto dentro de um diretório ou em um único arquivo

EXCEL, onde cada arquivo ou aba representa um tipo de variável do sistema.

Caso existam variáveis de entrada que sejam definidas por séries temporais, essas podem ser convertidas automaticamente em funções unárias do tempo. A interpolação das variáveis temporais foi implementada utilizando algoritmos do pacote *stats*, os quais recebem como argumentos o método de interpolação (degrau, linear ou spline), o tempo e os valores da variável para cada instante de tempo, e retornam a função unária correspondente.

A criação e simulação de um objeto *SDModel* pode ser feita, de maneira mais interativa, por meio da GUI implementada utilizando o R Shiny (CHANG et al., 2017) e exportada com o pacote sdsim. Essa interface gráfica, mostra de forma visual os campos necessários para especificar, inicializar e simular um MSD. Existem campos para editar os parâmetros do modelo e campos para inserir o script das funções R necessárias na definição do modelo.

### *Funcionalidades de um objeto SDModel*

Com um objeto *SDModel* inicializado é possível executar a simulação. Para tanto, foi implementado o seguinte método: *Run(Método de Integração Temporal das EDOs, Sequência de Tempo, Variáveis de Estado, Variáveis de Entrada, Booleanas para Detecção de Eventos, Inicialização e Checagem de Erros)*. Este método irá utilizar as variáveis padrão inicializadas no objeto, mescladas com as variáveis passadas como argumento que se queira adicionar ou alterar na lista padrão, para integrar a função que calcula as EDOs do sistema ao longo da *Sequência de Tempo*. O *Método de Integração Temporal das EDOs* define o algoritmo do pacote deSolve que será utilizado na simulação. Os demais parâmetros desse método auxiliam no controle e execução de eventos, inicialização de variáveis e checagem de erros adicional.

Caso o objeto *SDModel* possua a função para detecção de eventos, quando a condição do evento é satisfeita, a função de disparo de eventos é executada. Caso seja especificada a função para a detecção mas não a função de disparo, a simulação é interrompida no primeiro evento encontrado.

Para plotar o resultado da simulação de um MSD foi implementado o método *Plot* na classe do objeto *SDModel*. Esse método é capaz de plotar a trajetória de todas as variáveis de estado e, caso existam, as variáveis auxiliares e as séries temporais. Os gráficos gerados podem correlacionar as variáveis ou ser em função do tempo da simulação, seguindo as configurações de plot armazenadas no objeto. Se nenhuma configuração é definida, apenas as variáveis de estado são plotadas em função do tempo.

Um objeto *SDModel* também possui métodos para checar a descrição e unidade das variáveis, para recuperar informações sobre a última simulação (e.g. método de integração utilizado, tempo de simulação, etc.), para salvar em arquivos de texto o resultado da última simulação executada com seus respectivos dados de entrada e para limpar o objeto de todos os dados referentes a última simulação executada (e.g. as trajetórias das variáveis).

### *Classe CoupledSDAssembler*

A classe *CoupledSDAssembler* utilizada para estruturar a criação de simuladores de MSDs acoplados (MSDAs) também foi especificada usando a estrutura de classe do pacote R6. Essa classe representa um MSDA composto por objetos *SDModel* inicializados e uma lista de conexões que definem a comunicação entre os modelos, i.e. a atribuição de valores de variáveis de estado de um modelo para outro(s) modelo(s). As conexões possibilitam especificar laços de feedback de informação do sistema modelado. Dessa forma, modelos atômicos existentes podem ser reutilizados na definição de outros sistemas mais complexos.

A solução numérica do modelo acoplado é feita executando-se todas as conexões e em seguida calculando-se as diferenciais de todos os modelos componentes antes de cada transição de estado. Isso evita o efeito colateral de assincronia entre componentes do sistema, o que poderia gerar erros na simulação.

### *Criar um objeto CoupledSDAssembler*

Foi implementada a seguinte função construtora para classe *CoupledSDAssembler*: *CreateCoupledSDModel(Nome do Modelo Acoplado, Lista de Objetos SDModels, Lista de Conexões)*. Esta função cria um objeto *CoupledSDAssembler* com identificação *Nome do Modelo Acoplado* que possui como componentes a *Lista de Objetos SDModels*, inicializados e identificados pelo *Nome do Modelo Atômico* de cada objeto, e a *Lista de Conexões* que definem a comunicação entre os modelos.

Os métodos *AddModels*, *AddConnections*, *RemoveModels* e *RemoveConnections* de um objeto da classe CoupledSDAssembler podem auxiliar na reestruturação de modelos acoplados, possibilitando o acoplamento ou a remoção de modelos do sistema complexo, sempre mantendo a lista de conexões consistente com os modelos componentes.

### *Funcionalidades de um objeto CoupledSDAssembler*

Na classe *CoupledSDAssembler* o método *Run(Método de Integração Temporal das EDOs, Sequência de Tempo, Booleanas para Detecção e Configuração de Eventos)* executa a simulação de um MSDA. Este método irá utilizar as variáveis padrão inicializadas nos objetos *SDModel* acoplados para integrar a função que calcula as EDOs de todos os modelos componentes simultaneamente ao longo da *Sequência de Tempo*. O *Método de Integração Temporal das EDOs* define o algoritmo do pacote deSolve que será utilizado na simulação. Os demais parâmetros desse método auxiliam no controle e execução de eventos.

A classe *CoupledSDAssembler* também possui suporte a eventos e visualização gráfica dos resultados semelhante à classe *SDModel*.

### *Tratamento de erros no pacote sdsim*

A função *tryCatch* do R (MATLOFF, 2011) é utilizada ao longo do pacote sdsim para possibilitar a detecção de avisos e erros, customização das mensagens exibidas e controle do valor de retorno.

Ademais, as classes *SDModel* e *CoupledSDAssembler* possuem uma validação das EDOs que definem o sistema. Essa validação checa por valores inválidos (e.g. resultados nulos, ausentes ou infinitos) gerados na execução do primeiro passo da simulação e que podem decorrer de erros na especificação do modelo. O usuário recebe um aviso para cada variável com valor inválido encontrada na checagem, o que pode indicar que o modelo deva ser refatorado.

## RESULTADOS E DISCUSSÃO

Uma versão preliminar do pacote sdsim pode ser obtida no seguinte endereço eletrônico: `https://goo.gl/SQeUTu`. Essa versão já conta algumas das melhorias previstas para o pacote e não necessariamente corresponde a versão que foi utilizada no exemplo da Figura 1.

A Figura 1 exemplifica o uso deste pacote em script para implementação de um MSDA que representa a interação do modelo simplificado de crescimento foliar de pastagens (CFP) com um modelo simplificado de quantidade de água disponível no solo (AS).

```
library(sdsim)

## MODELO DE CRESCIMENTO FOLIAR DE PASTAGENS (CFP)
# Q Radiação Solar
# L Massa de folhas
# SLA Área especifica da folha
# LLS Vida útil da folha (dias)
# fSW Absorção de água pelo solo
# fWRatio Razão entre estoque e capacidade máxima de água no solo
# fLPartition Partição de Nutrientes
auxCpf <- list(
  fLAI =  "state[['L']] * vars$par$SLA",        ## Índice da Área da Folha
  fLIF =  "1 - exp(-vars$par$k * auxSfLAI)",    ## Índice de Intercepção de Luz
  fRADint =  "timeSeries$Q * auxSfLIF",         ## Radiação Interceptada
  fWRatio = "min( (vars$inputs$W / vars$par$Wmax), 1)",
  fSW =  "1 / (1 + ((1 - auxSfWRatio) / vars$par$SWconst) ^ vars$par$SWpower)",
  fGPP =  "vars$par$epsilon * auxSfSW * auxSfRADint", ## Produção Primária Bruta
  fNPP =  "auxSfGPP * vars$par$y",                  ## Produção Primária liquida
  fLPartition =  "(vars$par$Lmax - state[['L']]) / vars$par$Lmax")

# CFP Model Definition
dCpf <- function(t, state, vars, timeSeries, aux)
{
  dL <- auxSfNPP * auxSfLPartition - state[['L']] * (1/vars$par$LLS)
  return(list(c(dL = dL)))
}

# Variáveis do modelo em planilha excel
cpfInputConfig <- list(defaultInputPath = "CFP.xlsx", timeSeriesDirectory = ".")
# Cria e inicializa o modelo CFP
cfp <- CreateSDModel("CFP", ModelDefinition = dCpf)
cfp$InitModel(inputConfig = cpfInputConfig, aux = auxCpf,
              timeS = seq(from = 1, to = 100, by = 1), method = "rk4")
```

```
## MODELO DE QUANTIDADE DE ÁGUA DISPONÍVEL NO SOLO (AS)
# ETo Evaporação do Tanque Classe A
# SSHC Condutividade hidráulica saturada do solo
# fPP Quantidade de Água Drenada do Solo Saturado
# fCanCond Condução Canopy
# fETR Evapotranspiração Foliar
auxAs <- list(
  fCanCond = "vars$par$MaxCond * min(1, vars$inputs$L / vars$par$Lmax)",
  fETR = "vars$inputs$ETo * auxSfCanCond * state[['W']] / vars$par$Wmax",
  fPP = "max(timeSeries$Rain + state[['W']] - vars$par$Wmax, 0)*vars$par$SSHC")

# AS Model Definition
dAs <- function(t, state, vars, timeSeries, aux)
{
  dW <- timeSeries$Rain + vars$inputs$Irrig - auxSfETR - auxSfPP
  return(list(c(dW)))
}

# Variáveis do modelo em planilha excel
asInputConfig <- list(defaultInputPath = "AS.xlsx", timeSeriesDirectory = ".")
# Cria e inicializa o modelo CFP
as <- CreateSDModel("AS", ModelDefinition = dAs)
as$InitModel(inputConfig = asInputConfig, aux = auxAs,
             timeS = seq(from = 0, to = 100, by = 1), method = "rk4")

## MODELO ACOPLADO DO CFP COM O AS
connectionSimpleGrowth <- list(c("CFP", "W", "AS", "W"),
                               c("AS", "L", "CFP", "L"))
sg <- CreateCoupledSDModel("SimpleGrowth", models = list(cfp, as),
                           connections = connectionSimpleGrowth)
sg$Run()
sg$Plot()
```

Figura 1: Implementação do Modelo Acoplado de Crescimento Foliar de Pastagens Simplificado

Os modelos componentes foram criados com a classe *SDModel*, inicializados com dados em planilhas e acoplados utilizando a classe *CoupledSDAssembler*. A taxa de crescimento da massa foliar (L) do modelo CFP é influenciada positivamente pela radiação solar diária, pela partição de nutrientes disponíveis e pela quantidade de água disponível no solo (W) dada pelo acoplamento, e é afetado negativamente pelo decaimento das folhas. A taxa de crescimento da água disponível no solo (W) do modelo AS é influenciada positivamente pela chuva diária

e pela irrigação aplicada, e é afetada negativamente pela evapotranspiração foliar da pastagem (em função de L) acoplada e pela drenagem do solo saturado.

O resultado da simulação do modelo acoplado presente na Figura 2 mostra que uma maior quantidade de massa foliar implica em uma maior evapotranspiração e consequente queda da quantidade de água disponível no solo. Essa queda da água implica em uma queda no crescimento foliar e assim segue a interação entre os componentes.

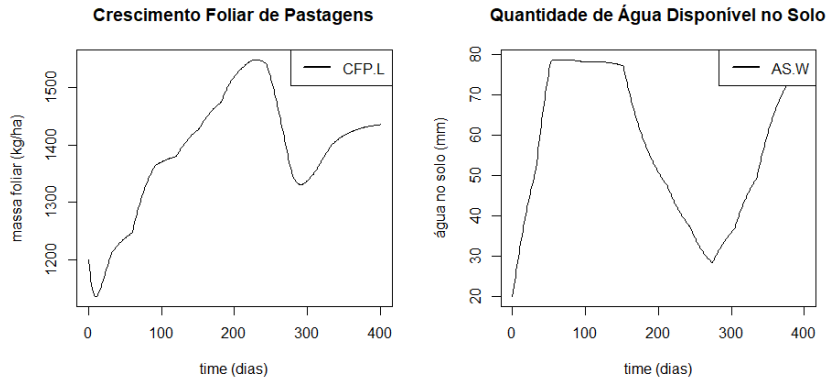


Figura 2: Resultado da Simulação do Modelo Acoplado da Figura 1

O pacote sdsim conta com um repositório de modelos implementados e prontos para serem executados. Estes modelos podem ser carregados com a função *data* e permitem que os usuários do R possam se familiarizar com o uso do pacote.

## CONCLUSÕES

A arquitetura proposta pelo pacote sdsim destina-se a solucionar a estruturação da definição de MSD atômicos no R. Essa arquitetura com estrutura OO incorpora dados e funções em um único objeto *SDModel* capaz de modelar a representação de sistemas dinâmicos baseados em EDOs de primeira ordem, permitindo a padronização generalizada tão desejada. Este padrão acelera o processo de desenvolvimento de simuladores por apresentar a estrutura que o modelista deve seguir para definir e armazenar seus modelos.

Os simuladores criados usando o sdsim podem ser salvos em formato *Rdata* com a função *save* do R, a função padrão para salvar objetos R, para serem armazenados, por exemplo, em repositórios compartilhados. O entendimento e uso da estrutura proposta auxilia na compreensão de simuladores de MSD de terceiros, facilitando a distribuição e reuso dos mesmos. Estes modelos podem então ser executados e modificados de acordo com a necessidade de cada usuário.

A integração do pacote com uma GUI construída automaticamente para prototipação e execução de modelos atômicos contribui para a simplicidade e facilidade de uso do pacote por programadores pouco experientes. Essa interface poderá auxiliar na difusão de conhecimentos sobre modelagem e simulação a todos os níveis de usuários.

Ao nosso conhecimento o sdsim é o primeiro pacote R a suportar formalmente o acoplamento de modelos por meio de uma abordagem orientada a objetos. Modelos acoplados são objetos *CoupleSDAssembler* contendo um conjunto de modelos atômicos componentes (objetos *SDModel's*) com conexões entre si. Dessa forma o entendimento do padrão adotado para modelos atômicos é suficiente para criação de modelos acoplados e, portanto, estes também usufruem da facilidade de definição e compartilhamento que a estrutura OO confere.

O acoplamento de modelos permite a definição de modelos mais complexos que interagem entre si. Dessa forma se torna viável, por exemplo, a construção de um modelo de crescimento de cultura acoplado com modelos de doenças que atacam essa plantação. A simulação deste modelo reproduz a interação entre seus componentes e auxilia na obtenção de informações para estimar os impactos no crescimento e no rendimento da cultura.

Na nova fase de desenvolvimento do sdsim os conceitos de simulador, modelo e cenário (valor das variáveis do sistema para um ambiente específico) serão desagregados para dar lugar a uma nova estrutura de classes. Essa estrutura possuirá novas classes para definir um modelo atômico, um modelo acoplado e um cenário, e possuirá uma função para executar a simulação de um modelo em um dado cenário. O resultado da simulação será armazenado em um novo objeto com funcionalidades para analise e visualização das trajetórias obtidas.

Futuramente, pretende-se estender a funcionalidade do pacote de forma a incluir métodos para calibração, intercomparação, otimização, filtragem e outras análises para MSDs atômicos e acoplados, nas classes *SDModel* e *CoupledSDAssembler*, respectivamente. Além disso, uma GUI com suporte a modelos acoplados e a todas as futuras funcionalidades pode diferenciar a usabilidade do pacote.

O fato da classe R6 possuir herança, possibilita que outros paradigmas sejam utilizados na regra matemática que integra o sistema. Dessa forma, em versões futuras do pacote, além de EDOs, pode-se incluir outras famílias de modelos (e.g. estatísticas, eventos discretos, baseado em indivíduos, espacialmente explícitos) aplicadas individualmente ou acopladas.

Seguindo a filosofia open-software do R, o pacote sdsim pretende oferecer um framework livre para definição e simulação de MSDs atômicos e acoplados. O código completo do pacote será publicado sob uma licença que garantirá uso irrestrito nas diversas áreas do conhecimento, e possivelmente a inclusão de modelos feitos por terceiros.

## AGRADECIMENTOS

Agradecemos aos membros da equipe da Embrapa Informática Agropecuária e a todos os outros colaboradores que empenharam esforços no desenvolvimento do pacote sdsim, tornando este artigo possível.

## REFERÊNCIAS

CHANG, W. R6: Classes with reference semantics. 2017. Disponível em: <https://cran.r-

project.org/package=R6>.

CHANG, W. et al. Web application framework for r. 2017. Disponível em: <https://CRAN.R-project.org/package=shiny>.

FORRESTER, J. W. *Industrial Dynamics*. 1st. ed. [S.l.]: Pegasus Communications, 1961. ISBN 1614275335.

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>.

GRAEFF, S. et al. *Crop Models as Decision Support Systems in Crop Production*. 1st. ed. Crop Production Technologies, 2012. ISBN 978953307781. Disponível em: <http://www.intechopen.com/books/crop-production-technologies/crop-modelsas-decision-support-systems-in-crop-production>.

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

JOST, J. *Dynamical systems*: Examples of complex behaviour. 1st. ed. [S.l.]: Springer, 2005. ISBN 9783540229087.

KNEIS, D. A code generator for ode-based models. 2017. Disponível em: <https://CRAN.R-project.org/package=rodeo>.

MATLOFF, N. *The art of R programming*: A tour of statistical software design. 1st. ed. [S.l.]: No Starch Press, 2011. ISBN 9781593273842.

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>.

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>.

STERMAN, J. D. *Business dynamics*: systems thinking and modeling for a complex world. 1st. ed. [S.l.]: McGraw-Hill Higher Education, 2000. ISBN 0072311355.

ZEIGLER, B.; KIM, T.; PRAEHOFER, H. *Theory of Modeling and Simulation*. 2nd. ed. [S.l.]: Academic Press, 2000. ISBN 9780127784557.