Avaliação de Decisões Estratégicas sob Incerteza: Análise das Contribuições da Modelagem Exploratória (EMA) e Robust Decision Making (RDM)

Pedro Nascimento de Lima

Versão Preliminar - R

Abstract

Este documento contém os resultados da dissertação, e é escrito diretamente no R. Os capítulos do projeto da dissertação (1,2 e 3) não estão reproduzidos por completo neste documento, mas sua estrutura e argumentos utilizados está representado para alinharmos o trabalho como um todo, e decidirmos o local de cada argumento. Por hora, vou gerar e escrever o documento no R para agilizar a geração dos gráficos e fórmulas. Após a estabilização do documento, vou levá-lo para o word e formatar.

Contents

1 Introdução:

Problematização sobre os desafios que a incerteza impõe à tomada de decisão estratégica.

1.1 Objeto e Questão de Pesquisa:

- Objeto: Avaliação de Decisões Estratégicas sob incerteza profunda. Uso o framework de processo de decisão estratégica do mintzberg para localizar o objeto da pesquisa.
- Questão de Pesquisa: "Quais são as contribuições da Modelagem Exploratória (EMA) e do Robust Decision Making (RDM) para a avaliação de decisões estratégicas organizacionais em situações de incerteza profunda?"

1.2 Objetivos

1.2.1 Objetivo Geral

"Analisar as contribuições da EMA e do RDM para a avaliação das decisões estratégicas em situações de incerteza profunda."

1.2.2 Objetivos Específicos

a) identificar abordagens para avaliação de decisão estratégica sob incerteza profunda;
b) instanciar o RDM no contexto empresarial;
c) avaliar a instanciação do RDM no contexto empresarial;
d) identificar heurísticas contingenciais na aplicação do RDM no ambiente empresarial.

1.3 Justificativa

Argumentação sobre as limitações das abordagens para tomada de decisão de incerteza. Linha Geral de Argumentação:

- Abordagens Atuais apresentam limitações sob incerteza profunda;
- Existe o RDM (e outros métodos);
- Não existe menção ao RDM na literatura de estratégia em negócio;
- O trabalho contribui realizando uma "exaptação" da abordagem.

2 Fundamentação Teórica

2.1 Avaliação de Decisões Estratégicas Sob Incerteza Profunda

2.1.1 Avaliação de Decisões Estratégicas

2.1.2 Níveis de Incerteza e Incerteza Profunda

2.2 Abordagens para Avaliação de Decisão sob Incerteza Profunda

2.2.1 Identificação de Artefatos

2.2.2 Contextos de Aplicação do RDM

2.3 RDM - Robust Decision Making (...)

3 Método de Pesquisa (...)

4 Contexto de Aplicação - Indústria da Manufatura Aditiva

As discussões desta seção provavelmente irão para o final do capítulo 2.

4.1 Comportamento da Demanda de Impressora 3D

4.2 Principais Players do Mercado

4.3 Comportamento de Variáveis Relevantes

4.4 Sub-divisão dos mercados potenciais da Impressão 3D

4.5 Delimitações do Trabalho

4.6 Questões relevantes levantadas para a simulação.

Questões não respondidas que o meu trabalho pode responder:

- Como pode se comportar a demanda por impressoras 3D?
- Que Estratégia de Capacidade um Player deve adotar para este ramo: Estratégia Agressiva de penetração no mercado ou estratégia "Conservadora".

Trabalho	Bass (1969)	Mahajan Muller (1996)	Dattée, Birdseye (2007)	Maier (1998) - Modelo de Competição	Maier (1998) - Modelo de Substituição	Cui, Zhao, Ravichandran (2011)	Sterman (2007)
Objeto original		Timing de Substituição de gerações de novos produtos com inovação tecnológica. (new product launch strategy)	Substituições Tecnológicas (technological substitutions)	Modelos de Difusão de Novos produtos (new product difgusion models).	Dinâmica de substituição de produtos novos por modelos antigos, assumindo que há monopólio de mercado.	Dynamic New Product Launch Strategies	
Principal Crítica aos demais modelos.		O modelo original de Bass não captura a sucessão de diferentes gerações de produtos.	Simplificam em demasia a heterogeneidade do mercado.	Não consideram a entrada de outros concorrentes no mercado.	Não consideram a entrada de novos modelos no mercado, e o tradeoff entre introduzir um produto cedo ou tarde.	Na maioria das vezes, não consideram estratégias dinâmicas.	
Modelos de Referência Citados.		Bass (1969), Wilson e Norton	Bass (1969) (modelo de difusão), Fischer e Pry (modelo de competição entre tecnologias).	Bass (1969), Milling (1986; 1987; 1989); Maier (1995) e Millinr e Maier (1996)	Fisher e Pry (1971), Norton e Bass (1987)		
X - Incertezas		Tamanho relativo dos mercados potenciais, margem do produto, parâmetros de difusão e substituição.	Heterogeneidade da população de possíveis clientes das substituições. Diferentes classes de clientes podem valorizar aspectos do produto de modo diferente, levando a dinâmicas de adoção diversas.	Tempo de Entrada de outros concorrentes para a divisão do mercado. Market share dos concorrentes em função de seu "coeficiente de inovação".	Tamanho potencial do mercado, Market Share, Multiplicador de Substituição, Tempo de obsolescência, Entrade de novos clientes potenciais, Capacidade Técnica dos Produtos e Preços.		
L - Estratégias / Decisões		Timing entre introdução de novos modelos de produtos com inovação tecnológica.	Obtenção de primeiros usuários que são formadores de opinião para amplificar o efeito da comunicação dentro de uma rede.	Estratégias de Precificação, orçamentação para pesquisa e desenvolvimento, tempo de entrada no mercado, e estratégias de divulgação.			
R - Relações		Mesmas relações contidas no modelo de bass, acrescentadas da relação de substituição de máquinas.	Relações entre fatores sociais (credibilidade, disponibilidade de informação) e a adoção de uma nova tecnologia. Adoção da tecnologia é moderada por um índice de performance da tecnologia e o seu custo.	Precificação, Esforços de Marketing e Delays na Entrega influenciam a probabilidade de compra. A competição (novos entrantes no mercado) também é considerada.	Relações entre incertezas adotadas, e vendas de novos modelos de produtos. O multiplicador de substituição é calculado em função da "capacidade técnica" do novo modelo e de seu preço.		
M - Métricas		Número Total de Produtos Vendidos, por geração.	Vendas totais por tecnologia, Número total de consumidores usuários.	Vendas, número de clientes.			

Figure 1: Temporario - Completar Quadro e Analisar Aqui

- Esperar o cenário melhor se configurar para agir ou agir para conquistar market share de modo preemptivo?
- Quais são as incertezas mais importantes para a determinação da estratégia de capacidade mais adequada?

As questões acima devem levar à escolha da simulação de dinâmica de sistemas como abordagem ideal. Não devem ser colocadas questões acima que a análise não irá ajudar a responder.

Quais players simular.

Que aspectos simular ou não

5 Revisão de Modelos

Em resposta às necessidades do item anterior, os modelos de difusão de novos produtos devem ser avaliados, culminando no modelo do Sterman (XXX). As características dos modelos podem ser brevemente descritas para ajudar nesta delimitação.

[Quadro de Comparação dos Modelos]

(Escrever no Word para facilitar as citações.)

Falar sobre cada modelo e mostrar o Quadro da análise dos modelos. Considerar que cada um dos modelos considera e suas contribuições e limitações para o trabalho atual. Ressaltar o que o Sterman considera e que os demais não consideram para justificar a escolha do Sterman como ponto de partida.

6 Ferramenta Computacional para a Análise RDM

O objetivo desta seção é descrever a ferramenta computacional desenvolvida no âmbito desta dissertação para viabilizar a operacionalização da análise RDM. A decisão por desenvolver a análise nesta dissertação por meio deste ambiente aberto, ainda que em princípio mais custosa, teve por objetivo realizar a análise RDM com a máxima independência possível, sem recorrer à ferramentas terceiras ou privadas. Além disto, o desenvolvimento desta ferramenta computacional permitirá que os resultados desta dissertação sejam reproduzidos. Recomenda-se ao leitor interessado que acesse a ferramenta disponível no link (http://bit.ly/pnldissert) Deste modo, procura-se atender aos requisitos de reprodutibilidade em trabalhos baseados em simulação computacional preconizados por Rahmandad e Sterman (2012).

A primeira barreira para a realização da Análise RDM é a disponibilidade de ferramentas computacionais amigáveis para a operacionalização da análise exploratória. Embora existam frameworks de desenvolvimento úteis para a modelagem exploratória (como o EmaWorkbench (KWAKKEL, 2013) o OpenMORDM (HADKA et al., 2015) e o Rhodium(XXX)), tais ferramentas implicam em empecilhos para a utilização no contexto deste trabalho. Em primeiro lugar, estas ferramentas requerem que seu usuário final programe o modelo computacional e insira os parâmetros diretamente no código fonte. Embora propiciem um ambiente de desenvolvimento adequado para programadores proeficientes nas suas respectivas linguages de programação, estas bibliotecas carecem de interfaces para que os usuários finais interajam com os inputs da simulação (ex.: alterem os parâmetros de entrada e estratégias a serem simuladas), e avaliem imediatamente o resultado das simulações.

A ferramenta EmaWorkbench, desenvolvida na linguagem python não possui interface gráfica, não suporta integração com o software de dinâmica de sistemas iThink, ou com modelos desenvolvidos na linguagem R. Neste sentido, a ferramenta requer que o modelo seja desenvolvido em uma ferramenta como o Vensim, Excel ou um modelo utilizando a linguagem Python.

Considerando a necessidade de flexibilidade durante a execução deste trabalho, o pesquisador optou por desenvolver rotinas computacionais próprias utilizando a linguagem R e bibliotecas de código aberto disponíveis no repositório CRAN. A linguagem R possui bibliotecas para a integração numérica do modelo computacional (biblioteca deSolve), para a calibração do modelo (FME), para a disponibilização dos resultados em um aplicativo web (shiny), e para a visualização interativa dos resultados (ggplot2, plotly). Utilizando tais bibliotecas em conjunto, foi possível implementar as rotinas computacionais para a operacionalização do RDM, cuja estrutura é ilustrada na Figura (XXX).

A ferramenta computacional foi projetada com o objetivo de receber uma planilha de inputs de dados (contendo a definição de estratégias a serem simuladas e incertezas a serem consideradas), e a partir do modelo computacional desenvolvido, rodar os passos da análise RDM com a maior grau de automação possível. A seguir são descritos os quatro principais componentes da ferramenta. O primeiro componente necessário para a análise RDM é um modelo de simulação computacional, e neste caso específico um modelo de dinâmica de sistemas. Este componente possui todas as equações necessárias para a simulação computacional e deve ser definido de modo compatível com a biblioteca deSolve, que é utilizada para a integração numérica do modelo.

O segundo componente (b) trata-se de uma planilha com formato padronizado, contendo as estratégias a serem simuladas e incertezas, incluindo valores máximos e mínimos para cada parâmetro.

Destaca-se que os componentes (a) e (b) podem ser modificados conforme o caso a ser analisado, sem a necessidade de reprogramar todas as funções do Simulador (c), nem do aplicativo web desenvolvido. Esta seção não detalhará cada um dos componentes e análises propiciadas pela ferramenta computacional, as quais serão analisadas nas seções seguintes.

[Parágrafo sobre a análise de perda de oportunidade para a definição da estratégia mais robusta segundo um determinado critério.]

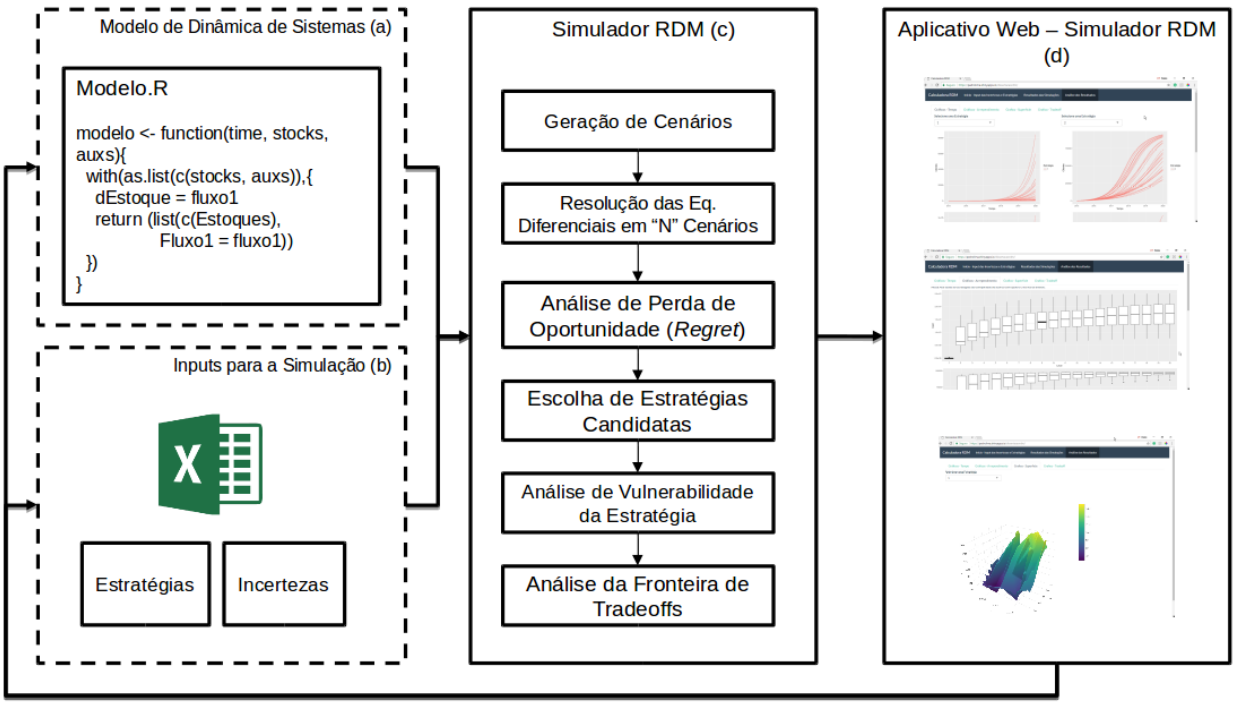


Figure 2: Projeto Modular da Ferramenta Computacional para a Análise RDM

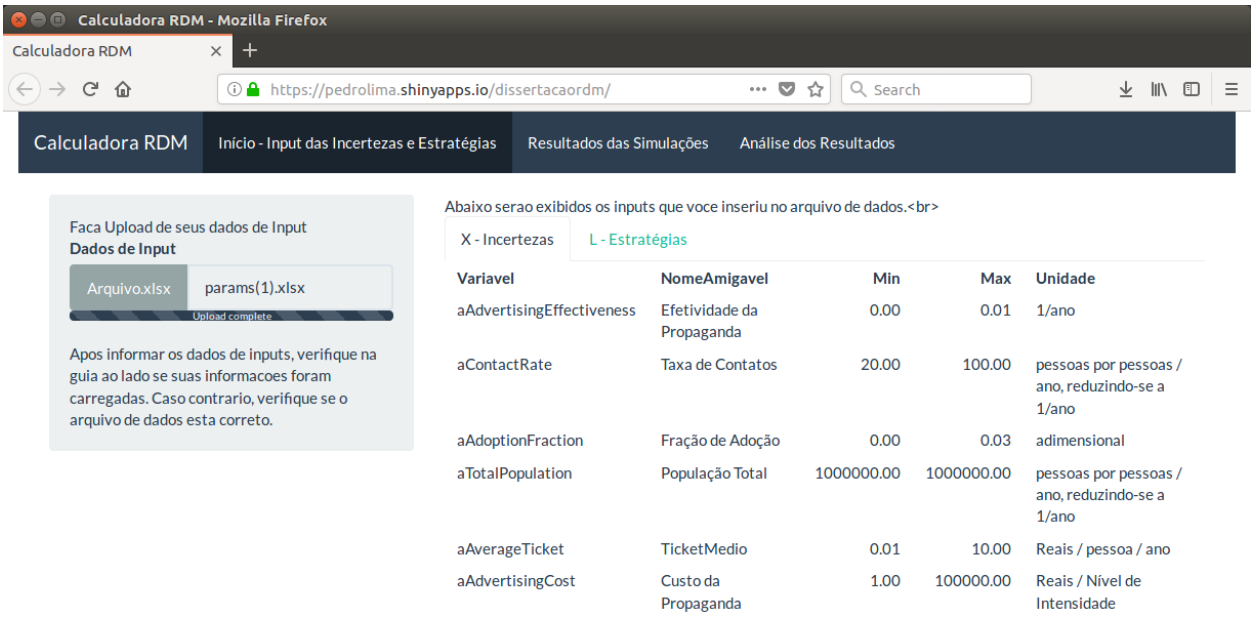


Figure 3: Tela de Inputs da Ferramenta Computacional

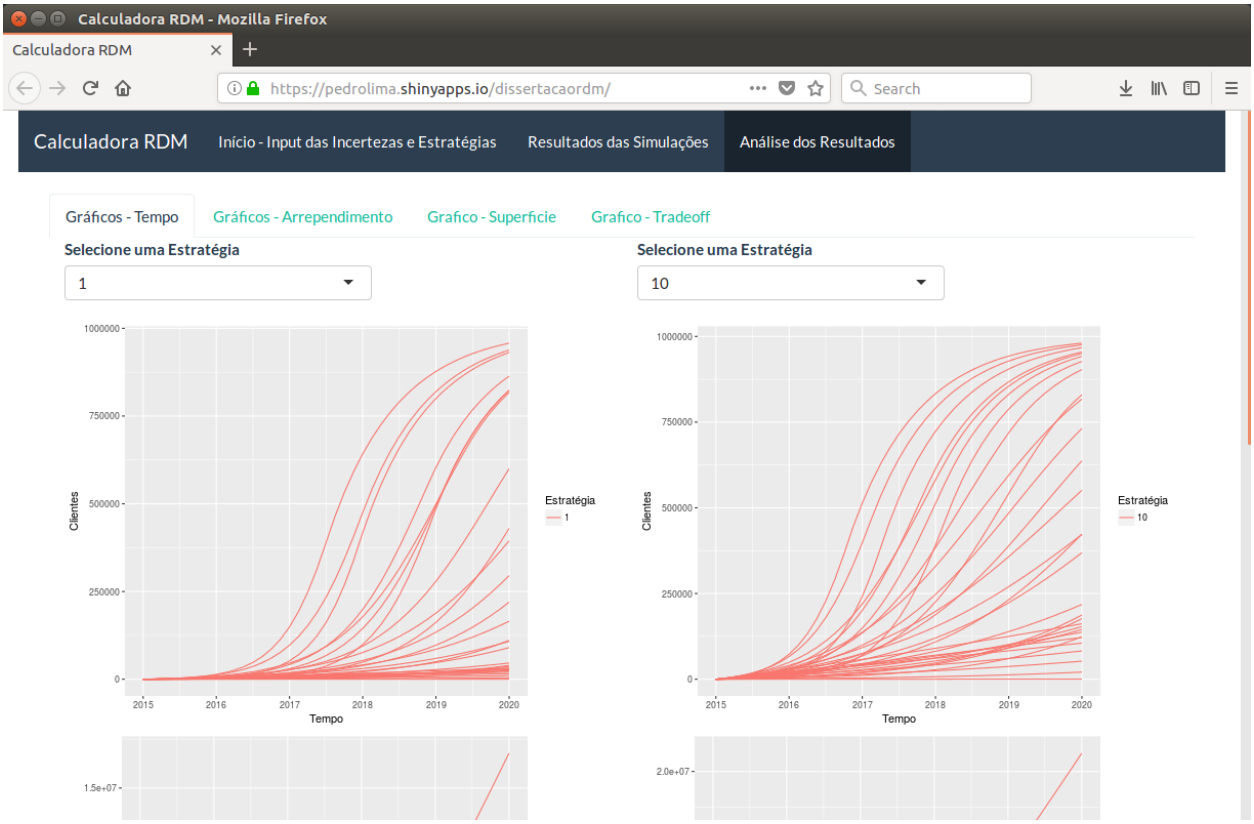


Figure 4: Comparação de Estratégias em "N" cenários utilizando a Ferramenta Computacional

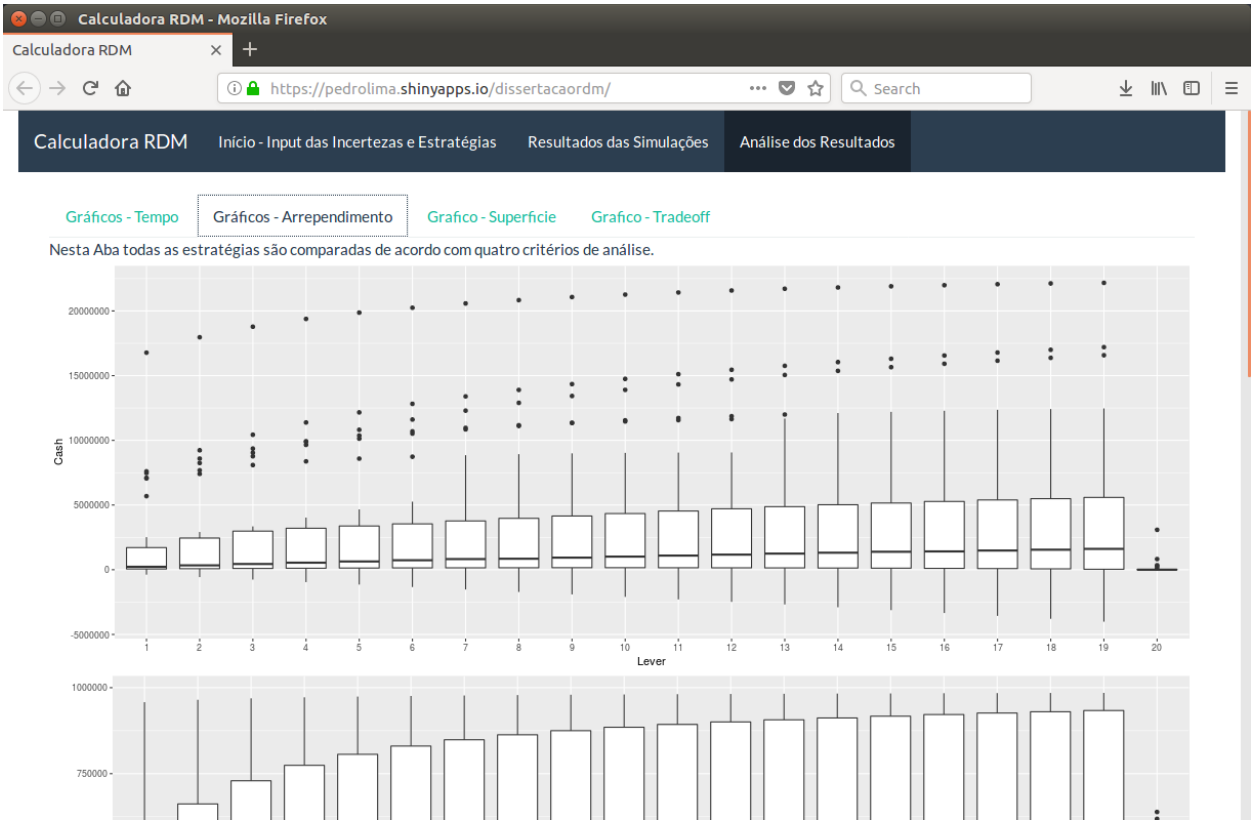


Figure 5: Análise de Perda de Oportunidade das Estratégias

7 Modelo da Competição na Indústria de Impressoras 3D

O modelo proposto inicialmente por Sterman (XX) foi utilizado como ponto de partida deste trabalho, por possuir uma série de características desejáveis para este trabalho. Em primeiro lugar, o modelo não é restrito a monopólios, como o modelo de Bass (XX) e outros modelos deste trabalho (identificar e citar aqui). Além disso, o modelo possui uma estrutura de dinâmica competitiva considerando a interação de diversos fatores presentes na Indústria da Manufatura Aditiva, incluindo curvas de aprendizagens, diferentes players expandindo sua capacidade produtiva em função da demanda prospectada no mercado.

7.1 Diagrama de Fronteiras do Modelo

A Figura (XX) ilustra os módulos do modelo e suas principais relações. Esta seção introduzirá as principais característstsicas do modelo, e argumentará sua relação com a indústria da manufatura aditiva. Além disto, a seção definirá as principais relações existentes entre os módulos e justificará a decisão pela inclusão de cada um destes módulos no modelo. Em seguida, a formulação matemática de cada um dos módulos será detalhada. Finalmente, esta seção também sintetizará as modificações empregadas no modelo original de Sterman (xx), justificando tais alterações.

{Porque usar o modelo do Sterman.} O Modelo proposto inicialmente por Sterman (XX) compreende um conjunto de características que o tornam apropriado para servir como ponto de partida deste trabalho. Em primeiro lugar, o modelo apresenta

{Características Básicas do Modelo} Uma primeira característica importante para a compreensão do modelo é a escolha pela vetorização da maioria de seus módulos em diferentes players produtores de impressoras 3D.

Um segundo aspecto importante para a compreensão do modelo é que o mesmo ocupa-se de decisões estratégicas relacionadas à capacidade produtiva da empresa. Em específico, o modelo ocupa-se de analisar estratégias de crescimento de capacidade agressivas versus estratégias conservadoras. (ex.: O player busca maximizar sua receita apropriando-se de um alto nível de market-share por meio de estratégias de crescimento agressivas). Para o tema da indústria de impressão 3D, esta decisão pode ser considerada adequada, considerando o tamanho do investimento necessário para a impressão 3D. Pode parecer, a princípio, que uma estratégia robusta seja criar expectativas modestas para uma indústria ainda em ascenção, e crescer o investimento apenas após uma demonstração clara de crescimento. . . .

{Alterações necessárias no Modelo de Sterman} - Questão: Market Share é apenas dividido por preço e delay na entrega, enquanto a performance do produto não parece ser considerada; - Solução: Criar setor de investimento em P&D influenciando a performance do produto juntamente com a experiência de produção;

- Questão: Estratégia de crescimento é “Conservadora” ou “Agressiva”, e não possui opção adaptativa;
- Solução: Avaliar primeiro as duas estratégias na primeira rodada do modelo e em seguida adicionar uma estratégia adaptativa (provavelmente a agressiva no início e conservadora no final).

No modelo atual, a demanda global pelo produto é determinada em função do preço, e de parâmetros que estimam o tamanho do mercado potencial, e sua reação à acréscimos ou decréscimos no preço por meio de uma curva de preço versus demanda. A demanda global calculada obtida em equilíbrio com o preço é sujeita à um processo de difusão do produto. Considerar o processo de difusão de um novo produto é uma prática presente em diversos modelos similadres (Ex: Bass (XX), citar outros), visto que a difusão de um novo produto não é instantânea. A difusão do produto é dada a partir da demanda global determinada pelo preço, e parâmetros que medem a velocidade de difusão do produto no mercado alvo.

O próximo conjunto de módulos do modelo é vetorizado por produtores de impressora 3D (a partir deste momento denominados como players). Esta característica torna o modelo útil para a avaliação da decisão estratégica de e um player específico, e permite a consideração de decisões estratégicas de outros players sobre o resultado da estratégia de um player em questão.Este aspecto será essencial para simular situações onde players existentes no mercado possuem estratégias de crescimento agressivas ou conservadoras, e o como estas decisões impactam o resultado da estratégia de um dado player. De modo similar, esta característica

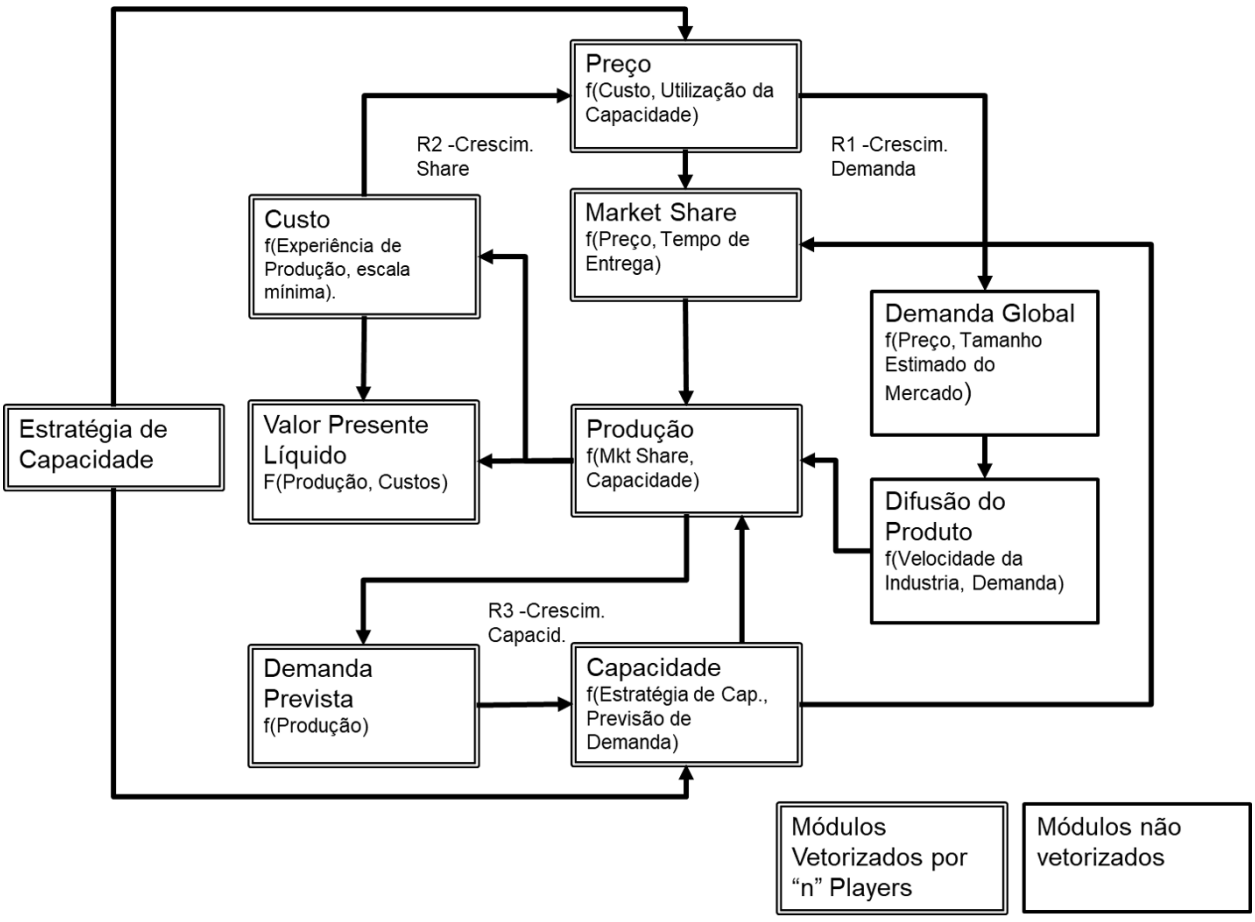


Figure 6: Modelo de Dinâmica Competitiva - Diagrama de Fronteiras

permite simular o impacto positivo que a expansão de outros players pode ter, expandindo o mercado de tal modo que haja mais demanda global para os demais players.

Este aspecto é relevante para a representação da indústria da manufatura aditiva, visto que a adição de capacidade por outros players, e decisões relacionadas à sua precificação tendem à influenciar a decisão da empresa.

Em seguida, a produção de cada um dos players simulados no moeolo é estimada, utilizando as informações de demanda, capacidade dos players e market share estimado. A produção, de modo imediato, gera caixa para os players, atualizando seu valor presente líquido em caixa.

Três macro-enlaces de feedback podem ser visualizados nesta estrutura. O primeiro enlace, R1, tende à estimular o crescimento da demanda por meio da expansão do mercado. Uma vez que parcelas cada vez maiores da

No modelo proposto por Sterman (XX) dois players, inicialmente com a mesma capacidade produtiva, iniciam vendendo produtos a um mercado em expansão.

7.2 Demanda Global

A demanda Total da indústria anual D^T é formada pela soma de dois tipos de demanda. A demanda inicial D^I dos produtos (ou seja, à primeira compra realizada por um usuário da impressora 3D), e à demanda oriúnda de recompras D^R, realizadas em função do fim da vida útil do equipamento.

$$D^T = D^I + D^R$$

A demanda inicial é calculada D^I em função do número médio de unidades vendidas por clientes μ e do número de clientes dA que adotou o produto em um intervalo de tempo dt:

$$D^I = \mu \frac{dA}{dt}$$

7.3 Difusão do Produto

O crescimento do número de clientes A que aderiram às impressoras 3D em um dado instante de tempo t é um estoque modelado por meio do modelo padrão de difusão de Bass (XXXX). Neste modelo o crescimento da população de clientes que aderem à uma ideia é dependente do tamanho total da população POP, do número de clientes que não adotaram N, da fração de inovadores que adotam ao produto ano a ano independentemente de outros usuários α e do parâmetro β que mede a força da difusão do produto por boca-a-boca. A não-negatividade da equação é garantida obtendo-se o máximo entre a equação e zero. Além disto, o valor inicial do número de clientes A_{t_0} é calibrado a partir do númer.....

$$A_t = A_{t_0} + \int_{t_0}^{t} MAX\left(0, N\left(\alpha + \beta \frac{A}{POP}\right)\right); A_{t_0} = \theta A^*$$

O número de consumidores potenciais N é modelado como o máximo entre zero e a diferença entre o número de clientes que irá adotar o produto em algum momento A^* e o número de clientes que adotou o produto A.

$$N = MAX(0, A^* - A)$$

O número de clientes que irá adotar o produto A^* é calculado segundo uma curva de demanda linear, variando em função do menor preço encontrado no mercado P^{min}, e da inclinação da curva de demanda σ, que corresponde à $(A^* - POP^r)/(P^{min} - P^r)$. Para a calibração da curva de preço e demanda, um preço de

referência P^r e uma demanda de referência POP^r. Além disto, a demanda nunca será maior do que a população total POP, nem menor do que 0.

$$A^* = MIN\left(POP, POP^r * MAX\left(0, 1 + \frac{\sigma(P^{min} - P^r)}{POP^r}\right)\right)$$

A inclinação da curva de demanda σ, por sua vez, é calculada em função da população de referência POP^r, do preço de referência P^r e da elasticidade da curva de demanda ε_d.

$$\sigma = -\varepsilon_d\left(\frac{POP^r}{p^r}\right)$$

A demanda oriúnda da necessidade de substituição dos produtos depende do número de impressoras 3D já vendidos pela empresa I_i, e de uma taxa percentual de descarte de impressoras δ. Esta taxa percentual de descarte de impressoras corresponde ao inverso da vida útil média das impressoras vendidas. O modelo pressupõe que impressoras descartadas pelo fim da sua vida útil são

$$D^r = \sum_i D_i \ ; \ D_i = \delta I_i$$

Installed Base:

$$I_{i,t} = I_{i,t_0} + \int_{t_0}^{t} S_{i,t} - D_{i,t}$$

Market Share

Orders:

$$O_i = S_i D^T$$

Share:

$$S_i = A_i / \sum_i A_i$$

Atratividade: - Aqui deve entrar também a performance do produto. Standard Logit decision model

$$A_i = e^{\varepsilon_p \frac{P_i}{P^r}} * e^{\varepsilon_a (\frac{B_i}{S_i})/\tau^r}$$

7.4 A Firma

O lucro líquido a valor presente π_t da firma i é definido como um estoque calculado em função das receitas e custos da empresa, trazidos a valor presente por um fator ρ. As receita líquida da empresa é calculada a partir do número de produtos entregues s_i pela empresa i e da diferença entre o preço médio dos produtos entregues $\bar{p_i}$, e do seu respectivo custo variável unitário vc_i. Os custos fixos da empresa são calculados a partir da sua capacidade C_i e de um custo fixo unitário fc_i. Desta maneira, o lucro líquido da empresa no tempo t será dado conforme esta equação:

$$\pi_t = \int_{t_0}^{t} [R_i - (C_i^f + C_i^v)] * e^{-\rho * t}$$

Receita:

$$R_i = S_i * (V_i / B_i)$$

Valor da Carteira de Vendas:

$$V_{i,t} = V_{i,t_0} + \int_{t_0}^{t} P_{i,t} * O_{i,t} - R_{i,t}$$

Custos:

$$C_i^f = u_i^f * K_i \; ; \; C_i^v = u_i^v * S_i$$

Custos Variáveis e Fixos decrescem conforme uma curva de experiência Standard learning curve:

$$u_i^f = u_0^f (E/E_0)^\gamma; \; u_i^v = u_0^v (E/E_0)^\gamma$$

Esta formlua pressupõe que não há troca de experiência entre os players, e que não há "perda de experiência".

$$E_{i,t} = E_{i,t0} + \int_{t_0}^{t} S_i$$

7.4.1 Produção

Shipmentso é igual à Produção é igual a shipments, desprezando estoques na cadeia produtiva.

$$Q_i = MIN(Q_i^*, K_i); S_i = Q_i$$

Considera-se um sistema Make to Order, não considera estoques na cadeia. Para eles, o estoque na cadeia introduziria um efeito chicote ainda pior para a estratégia Get big fast, e por isso foi possível desconsidera-lo.

Neste ponto será necessário tomar uma decisão se este aspecto é importante para as estratégias consideradas ou não.

Delivery Delay:

$$\tau_i = B_i * Q_i$$

"Target Ship Rate:"

$$Q_i^* = B/\tau_i^*$$

Backlog de Produção:

$$B_{i,t} = B_{i,t0} + \int_{t_0}^{t} O_i - Q_i$$

Capacidade: Ajusta-se conforme uma função Erlang Lag de terceira ordem.

Este é o operador φ Erlang Lag.

$$K_i = \varphi(K^*, \lambda)$$

Capacidade Alvo e Previsão da Demanda A capacidade Alvo da Empresa K^* market share alvo S^* previsão da demanda D^* taxa de utilização de capacidade u^*

$$K^* = MAX(K^{min}, S^* D^e / u^*)$$

mínima escala de produção eficiente K^{min}.

Demanda Prevista (Demanda Esperada) D^e: Demanda Observada-Reportada D^r Anos de Previsão λ Taxa esperada de crescimento da demanda g^e

$$D^e = D^r * e^{\lambda * g^e}$$

Taxa de crescimento da demanda:

Horizonte Histórico usado para a previsão h Demanda Observada-Reportada D^r

$$g^e = ln(D_t^r / D_t^r - h)/h$$

Demanda Observada-Reportada D^r - Segue um suavização exponencial:

$$dD^r/dt = (D^T - D^r)/\tau^r$$

Market Share Alvo e Estratégia da Firma:

$$S^* = \begin{cases} MAX(S_i^{min}, S_i^u), \ if \ Str_i = Agress. \\ \\ MIN(S_i^{max}, S_i^u), \ if \ Str_i = Conserv. \end{cases}$$

Se a firma busca uma estratégia agressiva, a mesma busca um share dominante do mercado. Uma estratégia conservadora, por outro lado, busca acomodação entre seus rivais, e define um market share modesto.

A empresa agressiva também busca explorar sua vantagem aproveitando-se da demora dos outros players ainda aumentando seu share quando ela identifica que haverá demanda não atendida pelos outros players.

Market Share "Não-Disputado":

$$S_i^u = MAX(0, D_i^u / D^e)$$

Demanda não contestada:

$$D_i^u = D^e - u^* \sum_{j \neq i} K_j^e$$

Capacidade dos competidores esperada:

$$K_j^e = wK_j^{e^*} + (1 - w)K_j$$

Calculo da Capacidade com defasagem - segue uma suavização exponencial:

$$dK_j^{e^*}/dt = (K_j^* - K_j^{e^*})/\tau^c$$

Preço: Preço também ajusta-se a um valor alvo com delay e tempo de ajuste.

$$dP_i/dt = (P_i^* - P_i)/\tau^p$$

Equação do Preço Alvo:

$$P^* = MAX\left[u^v, P\left(1+\alpha^c\left(\frac{P^c}{P}-1\right)\right)\left(1+\alpha^d\left(\frac{Q^*}{u^*K}-1\right)\right)\left(1+\alpha^s\left(S^*-S\right)\right)\right]$$

Parâmetros, Unidades Valores Máximos e Mínimos:

$$P^C = (1+m^*)(u_i^f + u_i^f)$$

Implementação do Modelo Computacional

O modelo matemático descrito na seção anterior foi implementado no software R. O código fonte implementado no software R está disponível no Apêndice (XX). Adicionalmente, o modelo foi implementado no software Ithink 10.0.3, com o propósito de verificar a consistência dos resultados obtidos no software R. Considerando que a integração numérica invariavelmente traz erros ao processo do calculo, (Sterman XXX).

O modelo foi implementado segundo as diretrizes constantes em Dungan (XXXX), e utilizou a biblioteca deSolve (procurar e Citar XXXX) para a resolução das equações diferenciais.

Os resultados deste trabalho podem ser observados no link bit.ly/reproddissertpnl.

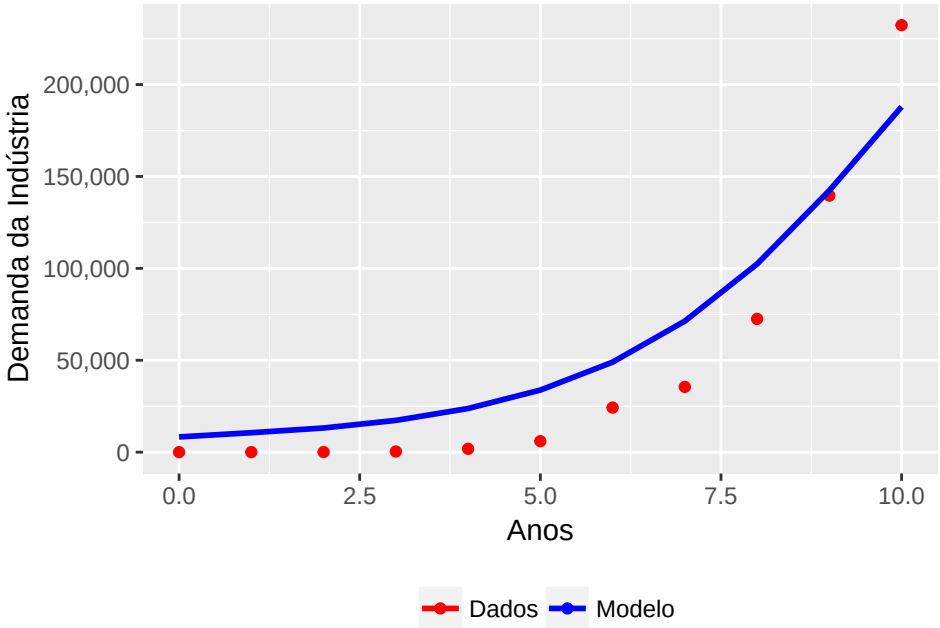
7.4.2 Testes Estruturais / Testes de Valores Extremos

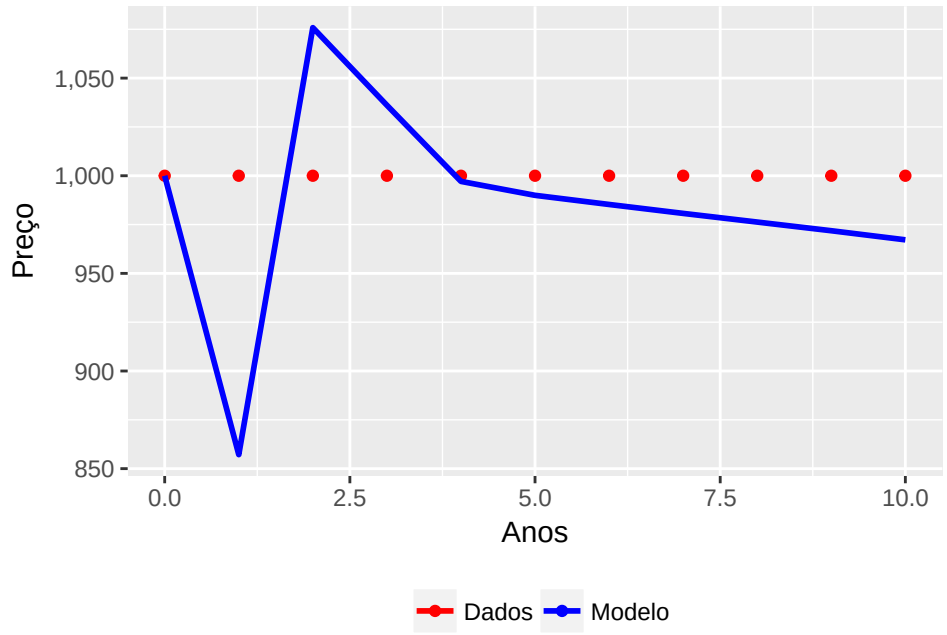
7.5 Calibração do Modelo e Comparação com Dados Históricos

A modelagem exploratória, per si, abandona a premissa de que modelos de simulação computacional apenas serão úteis se validados (Bankes XX). Ainda assim, os modelos podem ser verificados visando avaliar sua consistência interna, bem como os seus resultados podem ser comparados com dados históricos para observar a capacidade do modelo em explicar o comportamento passado. (Sterman XXXX) Considerando estas premissas, esta seção apresenta os testes realizados no modelo.

[Explicar o Procedimento de Calibração, fonte dos dados e objetivos da calibração.]

Calibração da Demanda Global:





Variáveis utilizadas na calibração e erros:

name	scale	N	SSR.unweighted	SSR.unscaled	SSR
fIndustryOrderRate	1	11	6.658340e+09	6.658340e+09	6.658340e+09
sPrice1	1	11	3.058842e+04	3.058842e+04	3.058842e+04

Tabela de Resíduos da Calibração:

name	x	obs	mod	weight	res.unweighted	res
fIndustryOrderRate	0	11	8292.1095	1	8281.109470	8281.109470
fIndustryOrderRate	1	30	10579.4832	1	10549.483155	10549.483155
fIndustryOrderRate	2	66	13152.3721	1	13086.372106	13086.372106
fIndustryOrderRate	3	355	17290.8739	1	16935.873937	16935.873937
fIndustryOrderRate	4	1816	23758.9904	1	21942.990402	21942.990402
fIndustryOrderRate	5	5978	33779.1580	1	27801.157993	27801.157993
fIndustryOrderRate	6	24265	48984.8198	1	24719.819755	24719.819755
fIndustryOrderRate	7	35508	71298.9731	1	35790.973101	35790.973101
fIndustryOrderRate	8	72503	102381.8904	1	29878.890440	29878.890440
fIndustryOrderRate	9	139584	142354.8982	1	2770.898173	2770.898173
fIndustryOrderRate	10	232336	187911.8368	1	-44424.163235	-44424.163235
sPrice1	0	1000	1000.0000	1	0.000000	0.000000
sPrice1	1	1000	857.1704	1	-142.829635	-142.829635
sPrice1	2	1000	1075.8558	1	75.855833	75.855833
sPrice1	3	1000	1036.0913	1	36.091330	36.091330
sPrice1	4	1000	997.1030	1	-2.896956	-2.896956
sPrice1	5	1000	989.9648	1	-10.035218	-10.035218
sPrice1	6	1000	985.2858	1	-14.714245	-14.714245
sPrice1	7	1000	980.7071	1	-19.292923	-19.292923
sPrice1	8	1000	976.2790	1	-23.721017	-23.721017
sPrice1	9	1000	971.8372	1	-28.162828	-28.162828
sPrice1	10	1000	967.1706	1	-32.829432	-32.829432

Parâmetros calibrados com este procedimento:

aPopulation	1.549793e+06
aWOMStrength	4.804582e-01

7.6 Análise RDM

7.6.1 XLRM

7.6.2 Geração de Casos (Rodada 1)

7.6.3 Análise de Vulnerabilidades (Rodada 1)

7.6.4 Modificações do Modelo para a segunda Rodada

7.6.5 Geração de Casos (Rodada 2)

7.6.6 Análise de Vulnerabilidades (Rodada 2)

7.6.7 Análise de Tradeoffs

7.7 Discussão dos Resultados

8 Conclusões

9 Apêndices

9.1 Códigos da Ferramenta Computacional

9.1.1 Modelo Computacional:

```
## function(time, stocks, auxs, modo = "completo"){
##   with(as.list(c(stocks, auxs)),{
##
##     # Criando uma variavel n_tempo local
##     n_tempo = nrow(list.variaveis.globais$sReportedIndustryVolume)
##
##     ##### VETORIZANDO ESTOQUES #####
##
##     #Estoques Vetorizados = substituindo estoques pela forma vetorizada (pra que seja possivel formul
##     # Esta implementação tem por objetivo não gerar a necessidade de referenciar os estoque spelo seu
##     sNPVProfit = stocks[(N_PLAYERS*0+1):(N_PLAYERS*1)]
##     sValueOfBacklog = stocks[(N_PLAYERS*1+1):(N_PLAYERS*2)]
##     sBacklog = stocks[(N_PLAYERS*2+1):(N_PLAYERS*3)]
##     sInstalledBase = stocks[(N_PLAYERS*3+1):(N_PLAYERS*4)]
##     sPrice = stocks[(N_PLAYERS*4+1):(N_PLAYERS*5)]
##     sCumulativeAdopters = stocks[(N_PLAYERS*5+1)]
##     sReportedIndustryVolume = stocks[(N_PLAYERS*6):(N_PLAYERS*6+1)]
##     sCumulativeProduction = stocks[(N_PLAYERS*7):(N_PLAYERS*7+1)]
##     sPerceivedCompTargetCapacity = stocks[(N_PLAYERS*8):(N_PLAYERS*8+1)]
```

```
##     sSmoothCapacity1 = stocks[(N_PLAYERS*9):(N_PLAYERS*9+1)]
##     sSmoothCapacity2 = stocks[(N_PLAYERS*10):(N_PLAYERS*10+1)]
##     sSmoothCapacity3 = stocks[(N_PLAYERS*11):(N_PLAYERS*11+1)]
##
##     #Obtendo o número da linha no qual estou
##     linha = (time * (n_tempo - 1)) / FINISH + 1
##
##     list.variaveis.globais$sReportedIndustryVolume[linha,] <<- sReportedIndustryVolume
##
##     # Gravando a Variável sReportedIndustryVolume no vetor global
##
##     ##### DIFFUSION SECTOR #####
##     aDemandCurveSlope = (- aReferencePopulation * aReferenceIndustryDemandElasticity )/ ( aReferenceI
##
##     aLowestPrice = min(sPrice)
##
##     aIndustryDemand = min(
##       aPopulation,
##       aReferencePopulation * max(
##         0,
##         1 + aDemandCurveSlope * (aLowestPrice - aReferencePrice) / aReferencePopulation
##       )
##     )
##
##     checkIndustryDemand = aIndustryDemand
##
##     aInitialCumulativeAdopters = aInitialDiffusionFraction * aIndustryDemand
##
##     aNonAdopters = aIndustryDemand - sCumulativeAdopters
##
##     checkNonAdopters = aNonAdopters
##
##     # Ajuste temporário: Colocar o adoption Rate como Fluxo apenas positivo.
##
##     fAdoptionRate = max(0,
##                         aNonAdopters * (aInnovatorAdoptionFraction + aWOMStrength * sCumulativeAdopte
##
##     checkAdoptionRate = fAdoptionRate
##
##     ##### ORDERS SECTOR - PT 1 #####
##
##     fDiscardRate = sInstalledBase * aFractionalDiscardRate
##
##     ##### INDUSTRY DEMAND SECTOR #####
##
##     fReorderRate = sum(fDiscardRate)
##
##     aInitialOrderRate = aUnitsPerHousehold * fAdoptionRate
##
##     fIndustryOrderRate = fReorderRate + aInitialOrderRate
##
##     checkIndustryOrderRate = fIndustryOrderRate
##
##     ##### ORDERS SECTOR - PT 2 #####
```

```
##
##     aDesiredShipments = sBacklog / aNormalDeliveryDelay
##
##     ### CAPACITY SECTOR - PT 1 ####
##
##     aCapacity = aSwitchForPerfectCapacity * (aDesiredShipments / aNormalCapacityUtilization) + (1-aSw
##
##     aNormalProduction = aCapacity * aNormalCapacityUtilization
##
##     aIndustryNormalProduction = sum(aNormalProduction)
##
##     ##### ORDERS SECTOR - PT 3 #####
##
##     fShipments = aSwitchForCapacity * pmin(aDesiredShipments, aCapacity) + (1-aSwitchForCapacity) * a
##
##     aCapacityUtilization = fShipments / aCapacity
##
##     aIndustryShipments = sum(fShipments)
##
##     aMarketShare = fShipments / aIndustryShipments
##
##     aDeliveryDelay = sBacklog / fShipments
##
##     checkIndustryShipments = aIndustryShipments
##
##     ##### MARKET SECTOR #####
##
##     aAttractivenessFromAvailability = exp(aSensOfAttractToAvailability*(aDeliveryDelay/aReferenceDeli
##
##     aAttractivenessFromPrice = exp(aSensOfAttractToPrice*(sPrice/aReferencePrice))
##
##     aAttractiveness = aAttractivenessFromAvailability * aAttractivenessFromPrice
##
##     aTotalAttractiveness = sum(aAttractiveness)
##
##     aOrderShare = aAttractiveness / aTotalAttractiveness
##
##     ##### ORDERS SECTOR - PT 3 #####
##
##     fOrders = fIndustryOrderRate * aOrderShare
##
##     checkOrders = sum(fOrders)
##
##     ##### EXPECTED INDUSTRY DEMAND SECTOR #####
##
##     aInitialDemandForecast = fReorderRate
##
##     aIndustryVolume = pmax(aInitialDemandForecast,
##                            aSwitchForShipmentsInForecast*aIndustryShipments+
##                              (1-aSwitchForShipmentsInForecast)*fIndustryOrderRate)
##
##
##     # Variavel com SMOOTH - Primeira Ordem: - Retirando o DT, o calculo funcionou corretamente!
##     fsmooth_ReportedIndustryVolume = ((aIndustryVolume - sReportedIndustryVolume) / aVolumeReportingD
```

```
##
##     # Variavel com DELAY - A definição das constantes aqui devem ser alteradas se as condicoes inicia
##     # Esta implementacao considera que os delays sempre serao iguais. Se os delays nao forem iguais,
##     if(time > aTimeForHistoricalVolume) {
##       nlinhas_delay = aTimeForHistoricalVolume / STEP
##       aLaggedIndustryVolume = list.variaveis.globais$sReportedIndustryVolume[(linha - nlinhas_delay)
##     } else {
##       aLaggedIndustryVolume = list.variaveis.globais$sReportedIndustryVolume[1,]
##     }
##
##     aExpGrowthInVolume =  log(sReportedIndustryVolume/aLaggedIndustryVolume)/aTimeForHistoricalVolume
##
##     aExpectedIndustryDemand = sReportedIndustryVolume*exp(aForecastHorizon*aCapacityAcquisitionDelay*
##
##     list.variaveis.globais$aExpectedIndustryDemand[linha,] = aExpectedIndustryDemand
##
##     # Mais uma variável com delay
##     if(time > aCapacityAcquisitionDelay) {
##       nlinhas_delay = aCapacityAcquisitionDelay / STEP
##       aLaggedVolumeForecast = list.variaveis.globais$aExpectedIndustryDemand[linha-nlinhas_delay,]
##     } else {
##       aLaggedVolumeForecast = list.variaveis.globais$aExpectedIndustryDemand[1,]
##     }
##
##     aForecastError = (aLaggedVolumeForecast - aIndustryVolume)/(1e-009+aIndustryVolume)
##
##     checkLaggedVolumeForecast = mean(aLaggedVolumeForecast)
##
##     ##### TARGET CAPACITY SECTOR #####
##
##     aIndustryCapacity = sum(aCapacity)
##
##     aCompetitorCapacity = aIndustryCapacity - aCapacity
##
##     aExpectedCompCapacity = aNormalCapacityUtilization*(aWeightOnSupplyLine*sPerceivedCompTargetCapac
##
##     aUncontestedDemand = pmax(0, aExpectedIndustryDemand - aExpectedCompCapacity)
##
##     aUncontestedMarketShare = aUncontestedDemand / aExpectedIndustryDemand
##
##     aSwitchForCapacityStrategy1 = ifelse(aSwitchForCapacityStrategy == 1, 1, 0)
##     aSwitchForCapacityStrategy2 = ifelse(aSwitchForCapacityStrategy == 2, 1, 0)
##     aSwitchForCapacityStrategy3 = ifelse(aSwitchForCapacityStrategy == 3, 1, 0)
##     aSwitchForCapacityStrategy4 = ifelse(aSwitchForCapacityStrategy == 4, 1, 0)
##
##     aTargetMarketShare = {
##         aSwitchForCapacityStrategy1*pmax(aDesiredMarketShare,aUncontestedMarketShare) +
##         aSwitchForCapacityStrategy2*pmin(aDesiredMarketShare,aUncontestedMarketShare) +
##         aSwitchForCapacityStrategy3*aDesiredMarketShare +
##         aSwitchForCapacityStrategy4*aUncontestedMarketShare
##     }
##
##
##     aTargetCapacity = pmax(aMinimumEfficientScale,
```

```
##                             aTargetMarketShare*aExpectedIndustryDemand/aNormalCapacityUtilization)
##
##     aTargetNormalProduction = aTargetCapacity * aNormalCapacityUtilization
##
##     aIndustryTotalTargetCapacity = sum(aTargetCapacity)
##
##     aCompetitorTargetCapacity = aIndustryTotalTargetCapacity - aTargetCapacity
##
##     fChangePerceivedCompTargetCapacity = (aCompetitorTargetCapacity - sPerceivedCompTargetCapacity) /
##
##     checkCompetitorTargetCapacity = mean(aCompetitorTargetCapacity)
##
##     ##### CAPACITY SECTOR  - PT 2 - FLUXOS #####
##     fchangeSmoothCapacity1 = (aTargetCapacity - sSmoothCapacity1) / (aCapacityAcquisitionDelay / 3)
##     fchangeSmoothCapacity2 = (sSmoothCapacity1 - sSmoothCapacity2) / (aCapacityAcquisitionDelay / 3)
##     fchangeSmoothCapacity3 = (sSmoothCapacity2 - sSmoothCapacity3) / (aCapacityAcquisitionDelay / 3)
##
##
##     ##### LEARNING CURVE SECTOR #####
##     fProduction = fShipments
##
##     aLCExponent = log(aLCStrength)/log(2)
##
##     aLearning = (sCumulativeProduction/aInitialProductionExperience)^aLCExponent
##
##     aInitialUnitFixedCost = (aInitialPrice/(1+aNormalProfitMargin))*aRatioOfFixedToVarCost*(1/(1+aRa
##
##     aInitialUnitVariableCost = (aInitialPrice/(1+aNormalProfitMargin))*(1/(1+aRatioOfFixedToVarCost/a
##
##     aUnitFixedCost = aLearning * aInitialUnitFixedCost
##
##     aUnitVariableCost = aLearning * aInitialUnitVariableCost
##
##     checkUnitFixedCost = mean(aUnitFixedCost)
##
##     checkUnitVariableCost = mean(aUnitVariableCost)
##
##     ##### PRICE SECTOR #####
##
##     aBasePrice = (1+aNormalProfitMargin)*(aUnitVariableCost+aUnitFixedCost/aNormalCapacityUtilizatio
##
##     aDemandSupplyBalance = aDesiredShipments/(aNormalCapacityUtilization*aCapacity)
##
##     aTargetPrice =
##       pmax(aUnitVariableCost,
##            sPrice*
##              (1+aSensOfPriceToCosts*((aBasePrice/sPrice)-1))*
##              (1+aSensOfPriceToDSBalance*(aDemandSupplyBalance-1))*
##              (1+aSensOfPriceToShare*((aTargetMarketShare-aMarketShare))))
##
##     checkTargetPrice = mean(aTargetPrice)
##
##     fChangeInPrice = (aTargetPrice - sPrice) / aPriceAdjustmentTime
##
```

```
##     ##### NET INCOME SECTOR #####
##
##     aDiscountFactor = exp(-aDiscountRate*time) #
##
##     fValueOfNewOrders = fOrders * sPrice
##
##     checkValueOfNewOrders1 = fValueOfNewOrders[1] #
##
##     aAveragePriceOfOrderBook = sValueOfBacklog / sBacklog
##
##     fRevenue = fShipments * aAveragePriceOfOrderBook #
##
##     checkRevenue1 = fRevenue[1] #
##
##     aVariableCost = fShipments * aUnitVariableCost #
##
##     aFixedCost = aCapacity * aUnitFixedCost #
##
##     fCost = aFixedCost + aVariableCost #
##
##     fNetIncome = fRevenue - fCost #
##
##     fNPVProfitChange = fNetIncome * aDiscountFactor #
##
##     checkNPVProfitChange = mean(fNPVProfitChange) #
##
##     aNPVIndustryProfits = sum(sNPVProfit) #
##
##
##     ##### ESTOQUES #####
##
##     d_NPVProfit_dt = fNPVProfitChange
##
##     d_ValueOfBacklog_dt = fValueOfNewOrders - fRevenue
##
##     d_Backlog_dt = fOrders - fShipments
##
##     d_InstalledBase_dt = fShipments - fDiscardRate
##
##     d_Price_dt = fChangeInPrice
##
##     d_CumulativeAdopters_dt = fAdoptionRate
##
##     d_sReportedIndustryVolume_dt = fsmooth_ReportedIndustryVolume
##
##     d_CumulativeProduction_dt = fProduction
##
##     d_PerceivedCompTargetCapacity_dt = fChangePerceivedCompTargetCapacity
##
##     d_SmoothCapacity1_dt = fchangeSmoothCapacity1
##
##     d_SmoothCapacity2_dt = fchangeSmoothCapacity2
##
##     d_SmoothCapacity3_dt = fchangeSmoothCapacity3
```

```
##
##
##
##     # Variaveis de Estoques Iniciais
##
##     BacklogIni = (1/length(fNetIncome)) * fIndustryOrderRate * aNormalDeliveryDelay
##     InstalledBaseIni = (1/length(fNetIncome)) * aUnitsPerHousehold * sCumulativeAdopters
##
##     CumulativeAdoptersIni = aInitialCumulativeAdopters
##
##     ValueOfBacklogIni = sPrice * BacklogIni
##
##     ReportedIndustryVolumeIni = aIndustryVolume
##
##     CumulativeProductionIni = aInitialProductionExperience
##
##     PerceivedCompTargetCapacityIni = aCompetitorCapacity
##
##     CapacityIni = (1/length(fNetIncome)) * fIndustryOrderRate / aNormalCapacityUtilization
##
##     ##### ESTOQUES - INICIAIS #####
##
##     stocks_ini = list(
##       BacklogIni = BacklogIni,
##       InstalledBaseIni = InstalledBaseIni,
##       CumulativeAdoptersIni = CumulativeAdoptersIni,
##       ValueOfBacklogIni = ValueOfBacklogIni,
##       ReportedIndustryVolumeIni = ReportedIndustryVolumeIni,
##       CumulativeProductionIni = CumulativeProductionIni,
##       PerceivedCompTargetCapacityIni = PerceivedCompTargetCapacityIni,
##       CapacityIni = CapacityIni
##     )
##
##
##
##
##     ##### COMPARAR RESULTADOS COM O ITHINK #####
##
##     if(VERIFICAR_STOCKS){
##       for (variavel in variaveis_ithink_stocks) {
##         # Definir o tipo de variavel
##         # Variavel é um estoque?
##         variavel_ithink_alterada = gsub(pattern = "\\[", replacement = "", x = variavel, ignore.case
##         variavel_ithink_alterada = gsub(pattern = "\\]", replacement = "", x = variavel_ithink_altera
##
##         # Verificar apenas Estoques:
##         variavel_ithink_alterada = paste("s", variavel_ithink_alterada, sep = "")
##
##         # Valor da Variavel Calculada
##         valor_variavel_R = eval(parse(text = variavel_ithink_alterada))
##
##         valor_variavel_ithink = dados_ithink_stocks[[linha,variavel]]
##
##         diferenca = valor_variavel_R - valor_variavel_ithink
```

```
##
##         if (abs(x = diferenca) > CHECK_PRECISION){
##           message(paste("Estoque Diff:", time, linha, variavel, diferenca, sep = " - "))
##           if(BROWSE_ON_DIFF){
##             browser()
##           }
##         }
##       }
##     }
##
##
##     if(VERIFICAR_CHECKS){
##       for (variavel in variaveis_ithink_checks) {
##         # Definir o tipo de variavel
##         # Variavel é um estoque?
##         variavel_ithink_alterada = gsub(pattern = "\\[", replacement = "", x = variavel, ignore.case
##         variavel_ithink_alterada = gsub(pattern = "\\]", replacement = "", x = variavel_ithink_altera
##
##         # Verificar apenas Estoques:
##         #variavel_ithink_alterada = paste("s", variavel_ithink_alterada, sep = "")
##
##         # Valor da Variavel Calculada
##         valor_variavel_R = eval(parse(text = variavel_ithink_alterada))
##
##         valor_variavel_ithink = dados_ithink_checks[[linha,variavel]]
##
##         diferenca = valor_variavel_R - valor_variavel_ithink
##
##         if(!is.na(diferenca)){
##           if (abs(x = diferenca) > CHECK_PRECISION){
##             message(paste("Check Diff:", time, linha, variavel, diferenca, sep = " - "))
##             if(BROWSE_ON_DIFF){
##               browser()
##             }
##           }
##         }
##
##       }
##     }
##
##     ##### VARIÁVEIS RETORNADAS #####
##
##     ## Parar se o tempo chegou ao fim.
##     if(time == FINISH){
##     # browser()
##     }
##
##     resultado_completo = list(c(
##       d_NPVProfit_dt
##       ,d_ValueOfBacklog_dt
##       ,d_Backlog_dt
##       ,d_InstalledBase_dt
##       ,d_Price_dt
##       ,d_CumulativeAdopters_dt
```

```
##       ,d_sReportedIndustryVolume_dt
##       ,d_CumulativeProduction_dt
##       ,d_PerceivedCompTargetCapacity_dt
##       ,d_SmoothCapacity1_dt
##       ,d_SmoothCapacity2_dt
##       ,d_SmoothCapacity3_dt
##     )
##     ,fIndustryOrderRate = fIndustryOrderRate
##     ,aNonAdopters = aNonAdopters
##     ,fReorderRate = fReorderRate
##     ,aIndustryShipments = aIndustryShipments
##     ,aIndustryVolume = aIndustryVolume
##     ,fDiscardRate = fDiscardRate
##     ,aDiscountFactor = aDiscountFactor
##     ,aDiscountRate = aDiscountRate
##     ,fNPVProfitChange = fNPVProfitChange
##     ,fNetIncome = fNetIncome
##     ,aNPVIndustryProfits = aNPVIndustryProfits
##     ,aInitialDemandForecast = aInitialDemandForecast
##     ,aLaggedVolumeForecast = aLaggedVolumeForecast
##     ,aForecastError = aForecastError
##     ,aTargetCapacity = aTargetCapacity
##     ,aCompetitorTargetCapacity = aCompetitorTargetCapacity)
##
##     return (if(modo == "inicial"){
##       stocks_ini
##     } else {
##       resultado_completo
##     })
##   })
## }
```

9.1.2 Rotinas para a Simulação RDM

- **Função Simular RDM e Escolher Estrategia**: Simula cenários do RDM, realiza a análise de perda de oportunidade e define a estratégia candidata utilizando um critério pré-determinado:

```
## function(inputs = "params.xlsx", sdmodel = sdmodel, opcoes = opcoes) {
##
##
##   output_simulacao = simular_RDM(arquivo_de_inputs=inputs ,sdmodel = sdmodel, n = opcoes$N)
##
##   ## Simular
##   dados_simulacao = output_simulacao$DadosSimulacao
##
##   # Selecionando dados do último ano:
##   dados = selecionar_ultimo_periodo(dados_simulacao = dados_simulacao, var_tempo = opcoes$VarTempo)
##
##   # Analisar Regret
##   analise_regret = calcular_e_resumir_regret(dados = dados, var_resposta = opcoes$VarResposta, var_c
##
##   # Escolher a Estratégia Candidata, com base no critério de robustez dos percentis
##   estrategia_candidata = escolher_estrategia_candidata(dados = analise_regret$Dados, resumo_estrateg
##
```

```
##   message(paste("A Estrategia candidata é a ", estrategia_candidata$Lever))
##
##   output = list(
##     DadosSimulados = dados_simulacao,
##     DadosUltimoPeriodo = dados,
##     AnaliseRegret = analise_regret,
##     Inputs = output_simulacao$Inputs,
##     Ensemble = output_simulacao$Ensemble,
##     EstrategiaCandidata =  as.numeric(estrategia_candidata[opcoes$VarEstrategias]),
##     Opcoes = opcoes,
##     SdModel = sdmodel
##   )
##
##   output
##
## }
```

- **Carregar Inputs**:

```
## function (arquivo_de_inputs="params.xlsx", abas_a_ler = c("params", "levers"), nomes_inputs = c("Para
##
##   # Criando uma list para os inputs
##   message(
##     paste("01. funcoes.R/carregar_inputs: Iniciando Carregamento de Inputs (funcao carregar_inputs()
##           "arquivo_de_inputs = ", arquivo_de_inputs)
##   )
##   inputs = vector(mode = "list", length = length(nomes_inputs))
##   names(inputs) = nomes_inputs
##
##   # Preenchendo os Dados dos Inputs
##   for (aba in abas_a_ler) {
##     n_aba = which(aba == abas_a_ler)
##     inputs[[n_aba]] = readxl::read_excel(arquivo_de_inputs,sheet = aba)
##   }
##
##   message("01. funcoes.R/carregar_inputs: Finalizando Carregamento de Inputs.")
##   return(inputs)
##
## }
```

- **Obter LHS Ensemble**:

```
## function (params, n=100) {
##   message("01. funcoes.R/obter_lhs_ensemble: Iniciando Obtenção do Ensemble.")
##   #Obtendo DataFrame de Parâmetros
##
##   nvar = length(params$Variavel)
##   pontos = n
##
##   # Obtendo um Hypercubo com as Variáveis que eu quero
##   randomLHS <- randomLHS(pontos, nvar)
##
##   p = as.data.frame(randomLHS)
##   min = as.vector(params$Min)
##   max = as.vector(params$Max)
##   variaveis = as.vector(params$Variavel)
```

```
##
##   # Transformando o Hypercubo em variáveis
##   # var <- matrix(nrow=pontos, ncol=variaveis)
##   ensemble = matrix(nrow = pontos, ncol = nvar+1)
##
##   # Montando o Ensemble
##   for (var in variaveis) {
##     i = which(x = variaveis == var)
##
##     # Aqui o i é +1 porque a primeira coluna será o cenário.
##     ensemble[,i+1] = qunif(p = randomLHS[,i], min = min[i], max = max[i])
##   }
##
##   # Adicionando A variável "Scenario"
##   variaveis = c(c(VAR_SCENARIO),variaveis)
##
##   colnames(ensemble) = variaveis
##
##   ensemble[,VAR_SCENARIO] = 1:nrow(ensemble)
##
##   ensemble
## }
```

- **Ampliar Ensemble como Levers**:

```
## function(ensemble, levers) {
##
##   variaveis_adicionais = names(dplyr::select(levers, -LeverCode))
##
##   linhas_ensemble_incial = nrow(ensemble)
##   novo_ensemble = matrix(0, nrow = nrow(ensemble)*length(levers$Lever), ncol = ncol(ensemble) + leng
##
##   names_old_ensemble = colnames(ensemble)
##   names_novo_ensemble = c(names_old_ensemble, variaveis_adicionais)
##
##   colnames(novo_ensemble) = names_novo_ensemble
##
##   j = 1
##   for (l in seq_along(levers$Lever)) {
##     lini = j
##     lfim = j + linhas_ensemble_incial-1
##     matriz_var_adicionais = as.matrix(levers[l,variaveis_adicionais])
##     novo_ensemble[lini:lfim,names_old_ensemble] = ensemble
##     novo_ensemble[lini:lfim,variaveis_adicionais] = matrix(matriz_var_adicionais, nrow = linhas_ense
##     j = j + linhas_ensemble_incial
##   }
##
##   novo_ensemble
##
## }
```

- **Simular**:

```
## function(stocks, simtime, modelo, ensemble, nomes_variaveis_final) {
##   message("01. funcoes.R/simular: Iniciando Simulação.")
##   # Rodando a Simulação (uma vez), com a primeira linha do ensemble - Ajuda a saber se funciona.
```

```
##   # Esta função apenas funciona com o estoque inicial fixo, será necessário implementar de outra forr
##   o<-data.frame(ode(y=stocks, times=simtime, func = modelo,
##                     parms=ensemble[1,], method="euler"))
##   pontos = nrow(ensemble)
##
##   nlinhas = nrow(o)
##
##   ncolunas = ncol(o)+1
##
##   # Montando uma matriz com todos os dados para a simulação
##   dados_simulacao = matrix(nrow = pontos*nlinhas, ncol = ncolunas)
##
##   # J é o índice dos dados simulados
##   j = 1
##   # Rodando a Simulacao Em todo o Ensemble
##   for (i in 1:nrow(ensemble)) {
##     resultados_simulacao = ode(y=stocks, times=simtime, func = modelo,
##                                parms=ensemble[i,], method="euler")
##     linhas = nrow(resultados_simulacao)
##
##
##     # Avançando a linha inicial e Final da Simulação
##     l_inicial = j
##     l_final = j + linhas-1
##
##     # Adicionando o resultado ao ensemble
##     dados_simulacao[l_inicial:l_final,1:ncolunas-1] = resultados_simulacao
##
##     # Adicionando o Número do Cenário
##     dados_simulacao[l_inicial:l_final,ncolunas] = ensemble[i,VAR_SCENARIO]
##
##     # Exibindo uma Mensagem de Status
##     if (i %% 100 == 0) {
##       message(paste(i, "simulações finalizadas."))
##     }
##
##     # Avançando o índice dos dados simulados
##     j = j + linhas
##   }
##
##   colnames(dados_simulacao) = nomes_variaveis_final
##
##   dados_simulacao = as.data.frame(dados_simulacao)
##   names(dados_simulacao) = nomes_variaveis_final
##
##   message("01. funcoes.R/simular: Finalizando Simulacao.")
##
##   dados_simulacao
## }
```

- **Simular RDM**:

```
## function(arquivo_de_inputs="params.xlsx", sdmodel, n = 10){
##   t_inicio = Sys.time()
##   message("Bem vindo ao SIMULADOR RDM! Pedro Lima.")
```

```
##   message(paste("Iniciando Simulacao RDM: ", t_inicio))
##
##   # Carregando Inputs
##   inputs = carregar_inputs(arquivo_de_inputs = arquivo_de_inputs)
##
##   # Obter Ensemble LHS (Sem Variáveis das Estratégias)
##   ensemble = obter_lhs_ensemble(params = inputs$Parametros, n = n)
##
##   # Ampliar Ensemble com as variáveis das Estratégias
##   novo_ensemble = ampliar_ensemble_com_levers(ensemble = ensemble, levers = inputs$Levers)
##
##   # Rodando a Simulação
##   nestrategias = length(inputs$Levers$Lever)
##   nfuturos = nrow(ensemble)
##   ntempo = ((sdmodel$Finish - sdmodel$Start)/sdmodel$Step)
##
##   message(paste("Esta rotina realizará", nestrategias * nfuturos * ntempo, "Simulacoes.\n (", nestra
##
##   dados_simulacao = simular(stocks = sdmodel$Stocks, simtime = sdmodel$SimTime, modelo = sdmodel$Mod
##
##   t_fim = Sys.time()
##
##   message("Finalizando Simulacao. Tempo de Simulacao: ", t_fim - t_inicio)
##
##   output = list(
##     Inputs = inputs,
##     Ensemble = ensemble,
##     NovoEnsemble = novo_ensemble,
##     DadosSimulacao = dados_simulacao
##   )
##
##   output
##
## }
```

- **Calcular Regret**:

```
## function(dados, var_resposta, var_group) {
##   var_maximo = paste("MaximoPor", var_group, sep = "")
##   var_minimo = paste("MinimoPor", var_group, sep = "")
##   var_regret = paste(var_resposta, "Regret", sep = "")
##   var_regret_perc = paste(var_regret, "Perc", sep = "")
##
##   dados[var_maximo] = calcular_maximo_por_variavel(var_resposta = var_resposta, var_group = var_grou
##
##   dados[var_minimo] = calcular_minimo_por_variavel(var_resposta = var_resposta, var_group = var_grou
##
##   dados[var_regret] = dados[var_maximo] - dados[var_resposta]
##
##   dados[var_regret_perc] = dados[var_regret] / (dados[var_maximo] - dados[var_minimo])
##
##   dados
## }
```

- **Resumir Variável Resposta**:

```
## function(dados = dados_ano_final, var_resposta = "Cash", var_group = "Lever") {
##   var_regret = paste(var_resposta, "Regret", sep = "")
##   var_regret_perc = paste(var_regret, "Perc", sep = "")
##
##   call = substitute(
##     expr =
##       dplyr::group_by(dados, VarGroup)
##     %>% select(VarGroup, VarResposta, VarRegret, VarRegretPerc)
##     %>% summarise(VarMedio = mean(VarResposta),
##                   VarDev = sd(VarResposta),
##                   Percentil25Var = quantile(VarResposta, probs = c(0.25)),
##                   Percentil75Var = quantile(VarResposta, probs = c(0.75)),
##                   RegretMedio = mean(VarRegret),
##                   DesvioRegret = sd(VarRegret),
##                   Percentil25Regret = quantile(VarRegret, probs = c(0.25)),
##                   Percentil75Regret = quantile(VarRegret, probs = c(0.75)),
##                   RegretMedioPerc = mean(VarRegretPerc),
##                   DesvioRegretPerc = sd(VarRegretPerc),
##                   Percentil25RegretPerc = quantile(VarRegretPerc, probs = c(0.25)),
##                   Percentil75RegretPerc = quantile(VarRegretPerc, probs = c(0.75))
##     )
##     ,
##     env = list(VarGroup = as.name(var_group),
##                VarResposta = as.name(var_resposta),
##                VarRegret = as.name(var_regret),
##                VarRegretPerc = as.name(var_regret_perc)
##     )
##   )
##
##   resumo = eval(call)
##
##   colnames(resumo) = c(
##     var_group,
##     paste(var_resposta, "Medio", sep = ""),
##     paste(var_resposta, "Desvio", sep = ""),
##     paste(var_resposta, "Percentil25", sep = ""),
##     paste(var_resposta, "Percentil75", sep = ""),
##     paste(var_regret, "Medio", sep = ""),
##     paste(var_regret, "Desvio", sep = ""),
##     paste(var_regret, "Percentil25", sep = ""),
##     paste(var_regret, "Percentil75", sep = ""),
##     paste(var_regret_perc, "Medio", sep = ""),
##     paste(var_regret_perc, "Desvio", sep = ""),
##     paste(var_regret_perc, "Percentil25", sep = ""),
##     paste(var_regret_perc, "Percentil75", sep = "")
##   )
##
##   resumo
## }
```

- **Escolher Estratégia Candidata**:

```
## function(dados, resumo_estrategias, var_resposta, var_criterio = "RegretPercPercentil75", sentido =
##
##   var_respota_criterio = paste(var_resposta, var_criterio, sep = "")
```

```
##
##
##   # Esta lista de criterios deve ser mantida igual à lista que a funcao resumir_variavel_resposta()
##   possiveis_var_criterios = c("Percentil25", "Percentil75", "Medio", "Desvio", "RegretMedio", "Regre
##
##   # Conferindo alguns pressupostos basicos:
##   possiveis_var_respota_e_criterios = paste(var_resposta, possiveis_var_criterios, sep = "")
##
##   # Conferindo se a variável de resposta e variável de critério combinam corretamente:
##   if (!all(possiveis_var_respota_e_criterios %in% names(resumo_estrategias))){
##     stop("Existe algo errado com a sua variavel de resposta ou variavel de criterio (a combinacao das
##   }
##
##   # Conferindo se a Variavel de criterio está correta.
##   if(!var_criterio %in% possiveis_var_criterios){
##     stop(paste("Esta variavel de criterio esta incorreta. escolha entre:",possiveis_var_criterios))
##   }
##
##
##   # Agora sim, posso escolhenr a estratégia que tem o menor percentil percentual 75 (assim como Lempe
##   estrategias_candidatas = switch(sentido,
##                                   "min" = escolher_estrategia_min(resumo_estrategias, var_respota_cri
##                                   "max" = escolher_estrategia_max(resumo_estrategias, var_respota_cri
##
##   estrategias_candidatas
## }
```

- **Calcular e Resumir Regret**:

```
## function(dados, var_resposta, var_cenarios, var_estrategias) {
##   dados = calcular_regret(dados = dados, var_resposta = var_resposta, var_group = var_cenarios)
##
##   # Resumindo Variável de Resposta Cash:
##   resumo_estrategias = resumir_variavel_resposta(dados = dados, var_resposta = var_resposta, var_grou
##
##   # Formar lista de outputs dessta análise
##   output = list(
##     Dados = dados,
##     ResumoEstrategias = resumo_estrategias
##   )
##
##   output
## }
```

- **Analisar Ensemble com Melhor Estratégia**:

```
## function(ensemble, dados_regret, var_cenarios, var_estrategias, var_resposta, estrategia_candidata) {
##
##
##   ensemble = as.data.frame(ensemble)
##   dados_regret = as.data.frame(dados_regret)
##
##
##   dados_regret["MelhorEstrategia"] = dados_regret[var_resposta] == dados_regret$MaximoPorScenario
##
##   linhas_melhores_estrategias = which(dados_regret[var_resposta] == dados_regret$MaximoPorScenario)
```

```
##
##   variaveis = c(var_cenarios, var_estrategias, var_resposta)
##
##   melhores_estrategias = as.data.frame(dados_regret[linhas_melhores_estrategias, variaveis])
##
##   ensemble_com_melhor_estrategia = dplyr::inner_join(ensemble, melhores_estrategias)
##
##   ensemble_com_melhor_estrategia["EstrategiaCandidata"] = ensemble_com_melhor_estrategia[var_estrate
##
##   #ensemble_com_melhor_estrategia = as.factor(ensemble_com_melhor_estrategia[var_estrategias])
##
##   ensemble_com_melhor_estrategia
##
## }
```

- **Funções do Aplicativo Web** :

```
## function(input, output, session) {
##
##   # Esta função apenas retorna o arquivo de Dados
##   CarregaDados <- reactive({
##     validate(
##       need(input$DadosInput != "", "Escolha o arquivo de simulacao de dados corretamente!")
##     )
##     arquivodados <- input$DadosInput
##     if (is.null(arquivodados))
##       return(NULL)
##     file.copy(arquivodados$datapath,
##               paste(arquivodados$datapath, ".xlsx", sep=""))
##     return(arquivodados)
##   })
##
##   # Esta função retorna a lista inputs
##   inputs = reactive({
##     inputs = CarregaDados()
##     if (is.null(inputs))
##       return(NULL)
##     withProgress(message = 'Carregando...', value = 0.3, {
##       #dados = simular_cba(paste(inputs$datapath, ".xlsx", sep=""), modo = "completo")
##       objeto_inputs = carregar_inputs(paste(inputs$datapath, ".xlsx", sep=""))
##       incProgress(1, detail = "Finalizando")
##     })
##
##     # if (is.null(arquivoinputs))
##     #   return(NULL)
##     # inputs = carregar_inputs(arquivoinputs)
##     return(objeto_inputs)
##   })
##
##   # Inputs - Lista de Levers
##   inputs_vetor_levers = reactive({
##     inputs()$Lever$Levers
##   })
##
##
```

```
##
##
##   # Tentativa de deixar a escolha de estratégias dinâmica.
##   # observe({
##   #   # Can also set the label and select items
##   #   updateSelectInput("gr1_estrategia_selecionada",
##   #                     label = "Estratégia",
##   #                     choices = output$inputs_vetor_levers(),
##   #                     selected = tail(inputs_vetor_levers, 1)
##   #   )
##   # })
##
##
##   # Dados de Absenteismo simulados
##   output_rdm = reactive({
##       inputs = CarregaDados()
##       if (is.null(inputs))
##         return(NULL)
##       withProgress(message = 'Calculando...', value = 0.3, {
##         #dados = simular_cba(paste(inputs$datapath, ".xlsx", sep=""), modo = "completo")
##         dados = simularRDM_e_escolher_estrategia(inputs = paste(inputs$datapath, ".xlsx", sep=""), s
##         incProgress(1, detail = "Finalizando")
##       })
##       return(dados)
##     })
##
##
##   # Parametros
##   resultados_dados_simulados = reactive({
##     output_rdm()$DadosSimulados
##     })
##
##   # Resultados Último Períodos
##   resultados_dados_ultimo_periodo = reactive({
##     output_rdm()$DadosUltimoPeriodo
##   })
##
##   # Estratégia Candidata
##   resultados_estrategia_candidata = reactive({
##     output_rdm()$EstrategiaCandidata
##   })
##
##   # Resumo das Estrategias
##   resultados_resumo_estrategias = reactive({
##     output_rdm()$AnaliseRegret$ResumoEstrategias
##   })
##
##   resultados_analise_regret_dados = reactive({
##     output_rdm()$AnaliseRegret$Dados
##   })
##
##
##   ensemble_analisado = reactive({
##     message("Ensemble_analisado")
```

```
##     analisar_ensemble_com_melhor_estrategia(ensemble = output_rdm()$Ensemble,
##                                             dados_regret = output_rdm()$AnaliseRegret$Dados,
##                                             var_cenarios = opcoes$VarCenarios,
##                                             var_estrategias = opcoes$VarEstrategias,
##                                             var_resposta = opcoes$VarResposta,
##                                             estrategia_candidata = output_rdm()$EstrategiaCandidata)
##   })
##
##   ###### OUTPUTS ######
##   output$dados_simulados_table <- renderTable({
##     head(resultados_dados_simulados(),n = 100)
##   })
##
##   output$analise_regret_table <- renderTable({
##     resultados_resumo_estrategias()
##   })
##
##   output$plot_clientes1 = renderPlot({
##     dados = resultados_dados_simulados()
##     plot_clientes_uma_estrategia(dados = dados,estrategia = input$gr1_estrategia_selecionada)
##   })
##
##   output$plot_cash1 = renderPlot({
##     dados = resultados_dados_simulados()
##     plot_cash_uma_estrategia(dados = dados,estrategia = input$gr1_estrategia_selecionada)
##   })
##
##   output$plot_taxa1 = renderPlot({
##     dados = resultados_dados_simulados()
##     plot_taxa_adocao_uma_estrategia(dados = dados,estrategia = input$gr1_estrategia_selecionada)
##   })
##
##   output$plot_clientes2 = renderPlot({
##     dados = resultados_dados_simulados()
##     plot_clientes_uma_estrategia(dados = dados,estrategia = input$gr2_estrategia_selecionada)
##   })
##
##   output$plot_cash2 = renderPlot({
##     dados = resultados_dados_simulados()
##     plot_cash_uma_estrategia(dados = dados,estrategia = input$gr2_estrategia_selecionada)
##   })
##
##   output$plot_taxa2 = renderPlot({
##     dados = resultados_dados_simulados()
##     plot_taxa_adocao_uma_estrategia(dados = dados,estrategia = input$gr2_estrategia_selecionada)
##   })
##
##   output$plot_whisker_lever_cash = renderPlot({
##     grafico_whisker_por_lever(dados_regret = resultados_analise_regret_dados(), variavel = "Cash")
##   })
##
##   output$plot_whisker_lever_adopters = renderPlot({
##     grafico_whisker_por_lever(dados_regret = resultados_analise_regret_dados(), variavel = "Adopters
##   })
```

```
##
##   output$plot_whisker_lever_regretperc = renderPlot({
##     grafico_whisker_por_lever(dados_regret = resultados_analise_regret_dados(), variavel = "CashRegr
##   })
##
##   output$plot_whisker_lever_regret = renderPlot({
##     grafico_whisker_por_lever(dados_regret = resultados_analise_regret_dados(), variavel = "CashRegr
##   })
##
##   output$plot_tradeoff = renderPlotly({
##     plot_fronteira_tradeoff_estrategia(results = output_rdm(), opcoes = opcoes) %>%
##       layout(autosize = F, width = 800, height = 800, margin = 50)
##   })
##
##   output$plot_estrategias_versus_incertezas = renderPlot({
##
##     incertezas = c("aAdvertisingEffectiveness", "aContactRate", "aAdoptionFraction", "aAdvertisingCo
##
##     ensemble_analisado = ensemble_analisado()
##     message("plot_estrategias_versus_incertezas")
##     plot_estrategias_versus_incertezas(ensemble_analisado, incertezas)
##
##   })
##
##
##
##
##   output$plot_superficie = renderPlotly({
##     dados_ultimo_ano = resultados_dados_ultimo_periodo()
##     variaveis = c("AdoptionFraction", "AdvertisingCost", "Cash")
##     estrategia = input$estrategia_superficie
##     gerar_grafico_superficie(dados_ultimo_ano, variaveis, estrategia = estrategia)  %>%
##       layout(autosize = F, width = 800, height = 800, margin = 50)
##   })
##
##   output$leverstable <- renderTable({
##     tabela = as.data.frame(inputs()$Levers)
##     tabela
##   })
##
##   output$parametrostable <- renderTable({
##     inputs()$Parametros
##   })
##
##   output$downloadData <- downloadHandler(
##     filename = function() { paste("output_simulacao", '.csv', sep='') },
##     content = function(file) {
##       write.table(resultados_cbr(),file,sep=";",dec=",",row.names = FALSE)
##     }
##   )
##
## }
```