



UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE CIÊNCIAS EXATAS E NATURAIS
FACULDADE DE COMPUTAÇÃO

Jairo Nascimento de Sousa Filho

**Geração de dados sintéticos utilizando a
aplicação Blocks: simulando dados discrepantes
e faltantes.**

Belém

2019

Jairo Nascimento de Sousa Filho

**Geração de dados sintéticos utilizando a aplicação
Blocks: simulando dados discrepantes e faltantes.**

Monografia apresentada na Faculdade de Computação do Instituto de Ciências Exatas e Naturais como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Universidade Federal do Pará

Orientador: Prof. Dr. Carlos Gustavo Resque dos Santos

Belém
2019

Insira a ficha catalográfica aqui.

Jairo Nascimento de Sousa Filho

Geração de dados sintéticos utilizando a aplicação Blocks: simulando dados discrepantes e faltantes.

Monografia apresentada na Faculdade de Computação do Instituto de Ciências Exatas e Naturais como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Conceito: _____

Belém, 1 de janeiro de 2019.

BANCA EXAMINADORA

Prof. Dr. Carlos Gustavo Resque dos Santos - Orientador
UFPA

Prof. Dr. Bianchi Serique Meiguins - Membro Interno
UFPA

MSc. Diego Hortêncio dos Santos - Membro Externo
UFPA

Às pessoas extraordinárias que me assistiram nos melhores e piores momentos e me proporcionaram chegar aqui.

Agradecimentos

Agradeço a Deus por permitir a oportunidade de desenvolver meu trabalho. Também agradeço a minha família por me dar suporte em todos os aspectos da minha vida. Também agradeço ao meu orientador e todos meus colegas do laboratório que me ajudaram incansavelmente a sanar minhas dúvidas. Aos meus amigos da faculdade estiveram comigo nos melhores e piores momentos, trabalhos e descontrações. Agradeço aos meus amigos fora da faculdade também que, por mais que estivessem distantes, também significam muito para mim.

“Se o conhecimento pode criar problemas, não é através da ignorância que podemos solucioná-los.”
(John F. Kennedy)

Resumo

Este trabalho foi desenvolvido tendo como foco a aplicação de geração e visualização de dados sintéticos denominada Blocks, a qual tem a função de produzir dados faltantes e discrepântes de forma sintética. A ferramenta em questão é composta por geradores chamados de Blocos, que por sua vez, podem ser utilizados de forma encadeada permitindo que sejam gerados comportamentos mais complexos. Os Blocos são utilizados de acordo com a categoria de geradores a serem utilizados, produzindo sequências de dados numéricas ou temporais; com distribuição probabilística aleatória; funcionais, que utilizam outra dimensão como parâmetro; acessórios, os quais adicionam formatação ou valor para os dados a partir de formas geométricas. Adicionalmente, o Blocks permite que os dados gerados sejam visualizados a partir de um gráfico do tipo Coordenadas Paralelas que é apresentado na tela inicial da aplicação. Como objetivos norteadores desse estudo são apontados a modelagem, visualização e avaliação da geração de dados sintéticos faltantes e discrepantes. Por fim, a validação desta pesquisa foi realizada com a utilização de técnicas de visualizações tendo como foco o comportamento das dimensões analisadas.

Palavras-chave: Dados Sintéticos, Dados Faltantes, Dados Discrepantes, Visualização de Dados.

Abstract

This work was developed focusing on the application of synthetic data generation and visualization called Blocks, which has the function of producing missing data and outliers synthetically. The tool is composed of generators called Blocks, that can be used in a chained way allowing more complex behaviors to be generated. Blocks are used according to the category of generators to be used, producing numeric or temporal data sequences; with random probability distribution; Function, using another dimension as a parameter; accessories, which add formatting, or value to the data from geometric shapes. Additionally, Blocks allows the generated data to be viewed from a Parallel Coordinates graph that is displayed on the application's home screen. The guiding objectives of this study are the modelling, visualization and evaluation of missing synthetic data and synthetic outliers generation. Finally, the validation of this research was performed using visualization techniques focusing on the behavior of the analyzed dimensions.

Keywords: Synthetic Data, Missing Data, Outliers, Data Visualization.

Listas de ilustrações

Figura 1.	Exemplo de escala de discrepância	22
Figura 2.	Exemplo de ruído e de anomalia. O item A é um exemplo de dado ruidoso, pois desvia-se levemente do padrão dos dados - uma reta. Quanto ao Item B este descaracteriza significativamente o padrão dos dados - este é um exemplo de dado anômalo.	23
Figura 3.	Exemplo de anormalidades; dimensões irrelevantes e instâncias ruídas.	24
Figura 4.	Visão Geral de geração de dados.	24
Figura 5.	Visão Geral de geração de dados do Sketchpad.	25
Figura 6.	Exemplo de árvore de decisão para jogar tennis criado a partir de regras encontradas em um conjunto de dados.	26
Figura 7.	Exemplo da interface do usuário para configuração do gerador de dados.	27
Figura 8.	Fluxo de passos para geração dos dados sintéticos.	27
Figura 9.	Comparação da média dos dados reais e sintéticos na simulação paramétrica e não paramétrica.	28
Figura 10.	Fluxo de passos para geração dos dados sintéticos.	29
Figura 11.	Fluxo de passos para geração dos dados sintéticos.	30
Figura 12.	Usando o DTM Data Generator.	32
Figura 13.	Usando o Redgate SQL Data Generator.	33
Figura 14.	Usando o Microsoft Visual Studio.	34
Figura 15.	Usando o dbForge <i>Test Data Generator</i>	35
Figura 16.	Usando o Mockaroo.	36
Figura 17.	Diagrama de Caso de uso do Blocks	39
Figura 18.	Diagrama de Classes dos geradores do Blocks.	40
Figura 19.	Diagrama de Classes (2 ^a Geração) dos geradores do Blocks.	40
Figura 20.	Diagrama de Classes (3 ^a Geração) dos geradores do Blocks.	41
Figura 21.	Diagrama de Sequência para geração de dados em arquivos no Blocks.	42
Figura 22.	Diagrama de Sequência para visualização de dados no Blocks.	43
Figura 23.	Pseudocódigo sobre a classe Generator do Blocks.	43
Figura 24.	Pseudocódigo sobre a classe MCAR do Blocks.	44
Figura 25.	Pseudocódigo sobre os atributos da classe MNAR do Blocks.	45
Figura 26.	Pseudocódigo sobre a classe MNAR do Blocks para dados numéricos.	45
Figura 27.	Pseudocódigo sobre a classe MNAR do Blocks para dados categóricos.	46
Figura 28.	Pseudocódigo sobre a classe MNAR do Blocks para dados temporais.	46
Figura 29.	Pseudocódigo sobre a classe do Blocks.	47
Figura 30.	Pseudocódigo sobre a classe do Blocks.	47
Figura 31.	Fluxograma de utilização do Blocks.	49

Figura 32. Ilustrando a leitura dos marcos dos quartis. O tamanho do espaço entre os quartis ou entre 0 ou o 100 é o valor da probabilidade de um número ser desse espaço.	53
Figura 33. Conhecendo os elementos da tela principal do Blocks, na sua versão para Windows.	58
Figura 34. Conhecendo os elementos da tela de configurações para geração de dados.	59
Figura 35. Conhecendo os elementos da <i>context menu</i> na aba do modelo.	59
Figura 36. Conhecendo os elementos da tela de configurações para geração de dados.	60
Figura 37. Conhecendo os elementos da tela de configurações para geração de dados.	60
Figura 38. Conhecendo os elementos da tela de configurações para geração de dados.	61
Figura 39. Base sintética de avaliação de Carros.	62
Figura 40. Base sintética de avaliação de Redes Sociais.	63
Figura 41. Base sintética de avaliação de Convênios Médicos.	64
Figura 42. Base sintética de avaliação de Estrutura de Conta Bancária.	64
Figura 43. Base sintética genérica para geração de dados faltantes numéricos. . . .	64
Figura 44. Visualização Coordenadas Paralelas da base de carros.	66
Figura 45. Visualização Scatterplot da base de carros.	66
Figura 46. Visualização Coordenadas Paralelas da base de Redes Sociais.	67
Figura 47. Visualização Histograma da base de Redes Sociais. A unidade de Curtida é bilhões; Seguidores está em milhões e Postagens está em milhares. . .	67
Figura 48. Visualização Dendrograma da base sobre estrutura de conta bancária. .	68
Figura 49. Visualização Gráfico de Colunas da base Convênios Médicos. Eixo X: Planos de Saúde, Eixo Y: Especialidades, Barra: Preço por Consulta .	69
Figura 50. Visualização Gráfico de Colunas da base genérica para visualização de dados faltantes.	70

Lista de tabelas

Tabela 1.	Exemplo de dados ausentes MCAR	21
Tabela 2.	Exemplo de dados ausentes MAR	21
Tabela 3.	Tabela de comparação das funcionalidades de cada aplicação. B.D. = Banco de Dados; S.O. = Sistema Operacional; D.F. = Dados Faltantes; D.D. = Dados Discrepantes; W.S. = <i>Web Service</i>	37
Tabela 4.	Propriedade dos geradores do Blocks. N=Número; C=Categoria; T=Tempo.	50
Tabela 5.	Tabela de atalhos do teclado do Blocks.	79

Lista de abreviaturas e siglas

JSON	JavaScript Object Notation
CSV	Comma Separated Values
DSV	Delimiter Separated Values
TSV	Tabular Separated Values
TSG	Threat Streaming Generator
XML	Extensible Markup Language
SOAP	Simple Object Access Protocol
REST	Representational State Transfer
HTTP	HiperText Transfer Protocol
MCAR	Missing Completely At Random
MAR	Missing At Random
MNAR	Missing not At Random
SVM	Support Vector Machine
CARS	Context-Aware Recommender Systems
UC	Use Case
SMV	Selectable Mode Vocoder
WSDL	Web Services Description Language

Sumário

1	INTRODUÇÃO	15
1.1	Motivação e Justificativa	16
1.2	Objetivos	16
1.2.1	Objetivo Geral	16
1.2.2	Objetivos Específicos	17
1.3	Estrutura do Trabalho	17
2	FUNDAMENTAÇÃO TEÓRICA	18
2.1	Dados Sintéticos	18
2.2	Arquivo	18
2.3	Web Service	19
2.4	Dados Ausentes	20
2.4.1	MCAR	20
2.4.2	MAR	21
2.4.3	MNAR	21
2.5	Dados Discrepantes	22
2.5.1	Dados Ruidosos e Anômalos	23
2.6	Trabalhos Acadêmicos Relacionados	24
2.7	Aplicações Relacionadas	30
3	ARQUITETURA DO PROJETO	38
3.1	Casos de uso do sistema	38
3.2	Diagrama de Classes	39
3.3	Diagrama de Sequência de atividades no Blocks	41
3.4	Pseudocódigos dos geradores desenvolvidos	42
4	SISTEMA BLOCKS	48
4.1	Tipos de Geradores de Dados	49
4.1.1	Sequencial	49
4.1.2	Aleatório	51
4.1.3	Função	51
4.1.4	Acessório	51
4.1.5	Geométrico	52
4.1.6	Baseado em dados reais	53
4.2	Modos de Geração de Dados	53
4.2.1	<i>Streaming Data</i>	54

4.2.2	Web Service	54
4.3	Modos para Visualização de Dados	54
4.3.1	Preview	54
4.3.2	Módulo de Visualização Externo e Integrado	55
4.4	Estrutura da Interface Gráfica do Blocks	55
4.4.1	Mensagens para o usuário	57
5	VALIDAÇÃO VISUAL	62
5.1	Modelagem dos Dados	62
5.2	Apresentação das Visualizações	64
6	CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS	71
	REFERÊNCIAS	73
	Apêndice A – Atalhos do Teclado da aplicação Blocks	79

1 Introdução

Dados sintéticos são dados gerados por uma máquina a partir de outros dados, geradores, fórmulas matemáticas, funções, entre outros. Esses dados, apesar de não possuirem um comportamento comparável aos da realidade, podem ser aplicados de variadas formas. Quer seja substituindo/simplificando dados reais, a exemplo de dados previsíveis que podem ser substituídos por fórmulas ou funções, ou adicionando novos dados a partir de uma instância fazendo algumas alterações. (MORAES, 2019) (MORAES, 2019)

Um problema muito complicado aumenta a demanda por dados sintéticos: confidencialidade dos dados. Ninguém quer ter seus dados sendo usados de forma arbitrária por empresas, mas todos querem melhores serviços. Logo, para melhorar a situação dos dois lados, os dados sintéticos podem atingir um grau de realismo - dependendo da ferramenta e da habilidade do operador - sem ter grandes problemas de privacidade. (MORAES, 2019)

Também, outro problema é a escassez dos dados. Este cenário é encontrado principalmente em lançamento de novas tecnologias ou tratamento de exceções. Então, a partir da amostra existente, gera-se novas instâncias, determinando novos padrões, logo, novos casos de teste ou de uso.

Os dados podem ser variados, como um número, uma categoria, uma marcação de tempo, uma imagem, um áudio ou um vídeo. Portanto, para geração de dados sintéticos são exigidos vários tipos de métodos. Entre os mais conhecidos podem ser citados fórmulas matemáticas, redes neurais, interpolação de frames, embaralhamentos, aleatoriedade etc.

Alguns métodos de geração podem gerar dados sintéticos muito simples. Sendo assim, uma forma de dar realismo aos dados é permitir que algumas instâncias sejam faltantes. Por exemplo, alguém pode se esquecer ou recusar responder uma informação, pode haver uma falha no disco rígido, entre outros. Logo, dados faltantes são considerados comuns em bases de dados reais.

Dados faltantes podem assumir diferentes formas dependendo do seu contexto de análise. Na literatura são encontradas 3 formas: *Missing Completely At Random* (MCAR), *Missing At Random* (MAR) e *Missing Not At Random* (MNAR). Basicamente o MCAR determina que a falta do dado é completamente aleatória. No MAR ainda é aleatório, mas é possível predizer o valor faltante a partir de uma dimensão correlacionada. O MNAR indica que o dado está faltante e o seu motivo não está dito na base de dados, ou seja, pode haver influência de contexto externo.

Outra característica de base de dados reais é a existência de dados discrepantes.

Estes são definidos como dados que saem do padrão, quer seja no valor ou na formatação. (AGGARWAL, 2012) Como exemplo de dados discrepantes podem ser citados uma falha no sensor ou um evento atípico.

Dados discrepantes podem assumir dois tipos: dados ruidosos e dados anômalos. Um dado ruidoso ou simplesmente ruído, possui um baixo grau de discrepância. Já o dado anômalo - ou anomalia - possui um alto grau de discrepancia, podendo até atrapalhar o entendimento do fenômeno e a visualização dos dados.

Pensando em tornar acessíveis os dados sintéticos foi desenvolvido o software de código aberto e gratuito chamado de Blocks. O Blocks oferece vários geradores de dados sintéticos de diferentes categorias. Também permite encadeá-los para que os dados tenham cada vez mais complexidade. É possível gerar sequências, dados randômicos, correlacionar dimensões e adicionar acessórios como ruídos e dados faltantes.

Por conseguinte, o objetivo é apresentar o Blocks e como gerar dados faltantes e discrepantes nessa aplicação. Também serão apresentados os geradores de dados faltantes desenvolvidos no âmbito deste trabalho. Feito isso, o Blocks permite que os dados sejam visualizados, então é interessante que essa funcionalidade seja analisada. Em suma, este trabalho vai apresentar o funcionamento do Blocks modelando e visualizando dados faltantes e discrepantes.

1.1 Motivação e Justificativa

O Blocks está inserido no nicho de geração de dados sintéticos. Há várias aplicações disponíveis para essa função de geração de dados. Contudo, o preço pode ser um obstáculo, bem como algumas funcionalidades que podem faltar como um *Web Service* ou geradores como para dados discrepantes ou faltantes. Outros diferenciais podem ser citados como a performance na obtenção de grande volume de dados ou acréscimo de dados sintéticos em bases reais.

1.2 Objetivos

Nessa seção são apresentados os objetivos do trabalho e as metas para atingí-los.

1.2.1 Objetivo Geral

O principal objetivo deste trabalho é implementar e validar geradores de dados sintéticos faltantes e discrepantes utilizando a aplicação Blocks. Também, a aplicação Blocks é demonstrada desde sua estrutura, a interface gráfica e o uso dos geradores implementados.

1.2.2 Objetivos Específicos

- Implementar geradores de dados faltantes no sistema Blocks.
- Implementar geradores de dados discrepantes no sistema Blocks.
- Validar geradores de dados faltantes e discrepantes através de visualizações.
- Contribuir para o desenvolvimento geral do sistema Blocks.

1.3 Estrutura do Trabalho

Este trabalho está organizado em 6 capítulos, o qual o presente capítulo trouxe uma contextualização do estudo dos dados sintéticos, faltantes, discrepantes, e também os objetivos do trabalho. O segundo capítulo trata da fundamentação teórica do trabalho, a qual busca embasar os conceitos utilizados durante o trabalho, bem como mostrar outros trabalhos e aplicações relacionadas.

O terceiro capítulo visa dar mais detalhes sobre a arquitetura de *software* do Blocks, com a apresentação de diagramas de classe, de sequência e casos de uso. O quarto capítulo apresenta o sistema blocks em seu funcionamento, interface gráfica, modos de geração de dados entre outros aspectos.

No quinto capítulo estão as visualizações a partir dos modelos de dados, que por sua vez, possuem os métodos implementados de dados faltantes e discrepântes. No sexto capítulo, estão o resumo do trabalho apresentado, algumas considerações sobre os resultados e trabalhos futuros para a continuação da pesquisa. Por fim, há um apêndice para mostrar os atalhos de teclados disponíveis no Blocks até o desenvolvimento desse trabalho.

2 Fundamentação Teórica

Neste capítulo é apresentada uma pesquisa sobre a literatura dos dados sintéticos, discrepantes, faltantes, bem como de arquivos, e serviços como *Web Service* e base de dados.

2.1 Dados Sintéticos

Dados sintéticos foi definido como "qualquer dado produzido o qual possa ser aplicado a uma dada situação que não foi obtido por mensuração direta." (EDUCATION, 2016). Como exemplo, Rubin (RUBIN, 1993) a introduziu um conjunto de dados completamente sintético. Com o objetivo de tornar anônimo os domicílios que participaram do censo daquela época. A questão da confidencialidade sempre foi uma característica necessária para dados divulgados, principalmente para dados pessoais. Os dados sintéticos possuem a possibilidade de serem alterados mantendo a mesma ideia, logo, representa os dados reais originais. Essa característica que ajudou na popularização dos dados sintéticos.

A necessidade de dados sintéticos podem ser de várias formas, desde a escassez de dados reais ou indisponibilidade; para teste de dados não usuais; para evitar lidar com questões de privacidade dos dados; teste de aplicação sem precisar modificar dados da aplicação de produção; criar teste de estresse da aplicação com *Big Data* antes de criar versão para produção; bem como não ter a necessidade de adicionar os dados de teste manualmente. (KUMAR, 2019)

A aplicabilidade dos dados sintéticos é ilimitada e é bastante explorada por setores cujos dados são pessoais como o financeiro (LOPEZ-ROJAS; AXELSSON, 2012) e da saúde (BERGEAT et al., 2014). Também são muito bem aplicáveis para exaustivos testes de segurança, os quais são necessários vários casos de teste pesquisador/analista de teste tem controle suficiente das características (fórmulas matemáticas ou regras de geração) e pode usar em um sistema de detecção de fraudes, por exemplo (BARSE; KVARNSTROM; JONSSON, 2003).

2.2 Arquivo

Gerar os dados não é o suficiente, para isso, é necessário oferecer uma forma pronta de uso para o usuário. Para isso, pode ser utilizado os arquivos. Segundo Tanenbaum arquivos são unidades lógicas de informação criadas por processos e gerenciados por sistemas operacionais. Também é um mecanismo de abstração ao usuário para leitura e

escrita em disco. Para que isso funcione, são adotados algumas convenções. A primeira são os sistemas de arquivos. Basicamente, um sistema operacional adota um sistema de arquivos para personalizar a questão da leitura e escrita. Também, um arquivo possui uma extensão (nome.extensão) cuja esta dá mais informações a respeito do conteúdo do arquivo (TANENBAUM; FILHO, 1995).

Os arquivos são amplamente utilizados na computação, e por conta da larga escala de uso, foram criadas várias extensões de arquivos. Toda linguagem de programação possui seu arquivo como .java para Java, .js para JavaScript e .py para Python.

Também existem os arquivos cujos dados são acessados por várias plataformas. O XML é muito utilizado na Web e pela linguagem Java, o JSON (*Javascript Object Notation*, ou em português Notação de Objeto Javascript) tomou uma parcela da web com o uso, principalmente, de APIs, e o CSV (*Comma Separated Values*) é largamente utilizado em aplicações de ciência de dados, por exemplo.

E quanto ao JSON, (BRAY, 2017) (CROCKFORD, 2003) este lançado em 2002, é uma formatação leve para troca de dados. O uso é facilitado tanto para seres humano quanto para máquina. O JSON é um formato de texto que é independente de linguagem, mas foi baseado na notação de objeto do Javascript (ECMA-262, 1999).

Quanto aos tipos de dados suportados, o JSON (BRAY, 2017) é uma sequência de tokens. Os tipos de tokens aceitos é do tipo *object*, *array*, *string*, *number* e nomes literais como *false*, *true* e *null*.

Outra extensão de arquivo é o CSV (SHAFRANOVICH, 2005) (comma-separated values, ou em português Valores Separados por Vírgula) o qual é um arquivo do tipo de texto MIME (Internet Media) (FREED J. KLENSIN, 1996) que utiliza a codificação de caracteres US-ASCII (HAUSENBLAS E. WILDE, 2014). Ao longo dos anos, seu uso foi consolidado para exportar dados entre vários softwares de tabelas (Microsoft suíte para Apple Suíte, por exemplo). A padronização do CSV demorou a ocorrer e por isso, vários outros estilos surgiram, a exemplo, o uso do CSV com ponto-e-vírgula (;). Outros estilos foram criados a ponto de ser chamado de arquivo DSV (RAYMOND, 2003). Por conseguinte, outro estilo que teve notoriedade na troca de dados entre bancos de dados ou tabelas de dados foi o TSV (KORPELA, 2000). A ideia é similar ao CSV, porém é utilizado uma tabulação em vez de vírgula.

2.3 Web Service

Um *Web Service* (GROUP, 2004) é definido como um software criado para suportar interoperabilidade entre máquinas através da rede computadores. Na história do *Web Service* surgiram o Protocolo SOAP (Simple Object Access Protocol) que utiliza o WSDL

(Web Services Description Language) como uma interface descrita em um formato processável por máquinas (GROUP, 2004) e o modelo de arquitetura REST (Representational State Transfer) proposto por Fielding (FIELDING; TAYLOR, 2000) que não utiliza o WSDL. Atualmente é predominante o uso de REST que em vez de exportar serviços como o SOAP, exporta os dados em si e não necessita do WSDL. (STACKIFY, 2017) O REST foi criado em conjunto ao HTTP 1.1 e a sua principal característica é a utilização simplificada dos verbos do protocolo HTTP (GET, POST, PUT, HEAD, OPTIONS e DELETE) (ATTORRE, 2015)

2.4 Dados Ausentes

O termo dados ausentes ou dados faltantes significa que está faltando dados suficientes para se formar uma informação e compreender o fenômeno de interesse ao observar o conjunto de dados. (MCKNIGHT, 2007) Esses dados podem ser perdidos ou não coletados em todas as etapas de obtenção de dados em uma situação real como um participante desistindo ou não respondendo parte da pesquisa, o pesquisador perdendo seu dispositivo de anotação, má operação ao salvar em dispositivos eletrônicos etc. (MCKNIGHT, 2007)

O grande impacto dos dados ausentes encontra-se quando faltam dados a ponto da pesquisa se tornar tendenciosa, inconclusiva ou inconsistente (MCKNIGHT, 2007). Um exemplo seria uma pesquisa de avaliação de motos. Características como modelo, ano, preço, número de vendas e média de notas seriam coletadas. Contudo, ao salvar a pesquisa no mesmo disco rígido, a parte referida a um dos integrantes foi corrompida e, portanto, há uma ausência de dados.

Para compreender e lidar melhor com os dados ausentes foram definidos os mecanismos de dados ausentes. Esses mecanismos são conceituados como a probabilidade de uma resposta ser observada ou estar faltando. Existem 3 mecanismos conhecidos como faltando de forma completamente aleatória - *MCAR* (*Missing completely at random*); faltando de forma aleatória - *MAR* (*Missing At Random*); faltando de forma não aleatória - *MNAR* (*Missing Not At random*) (MOLENBERGHS et al., 2014).

2.4.1 MCAR

Um dado faltante é classificado como MCAR quando a probabilidade da resposta está faltando não é relacionada com outros valores do conjunto de dados nem com os dados que deveriam ser coletados. Vale ressaltar que é muito difícil relacionar este mecanismo nos conjuntos de dados reais (MOLENBERGHS et al., 2014) (LITTLE et al., 2016). Na tabela 1 os dados ausentes MCAR não apresentam correlação com outras propriedades para justificar o dado faltante. Portanto, não há como prever qual o valor do dado faltante.

Tabela 1. Exemplo de dados ausentes MCAR

ID	Estação do ano	Fruta	Receita
1	Verão	Laranja	Alta
2	Inverno	Laranja	Baixa
3	Verão	Morango	Baixa
4	Inverno	Morango	Baixa
5	Outono		Baixa

Fonte: O autor do trabalho.

2.4.2 MAR

Quanto ao MAR, este é definido como a probabilidade da resposta está faltando depende dos dados obtidos, mas não está relacionado com dados fora da amostra coletada. Este é o mecanismo mais fácil de utilizar técnicas de imputação de dados, pois permite a predição de resultados. (MOLENBERGHS et al., 2014) (LITTLE et al., 2016) Na tabela 2 é possível visualizar um exemplo de dados ausentes do mecanismo MAR. Neste caso, assume-se que há correlação entre os valores da tabela e por isso, por predição, assume-se que o valor faltante seja "Alta".

Tabela 2. Exemplo de dados ausentes MAR

ID	Estação do ano	Fruta	Receita
1	Verão	Laranja	Alta
2	P <small>rimavera</small>	Laranja	Alta
3	Verão	Limão	Alta
4	Inverno	Limão	Baixa
5	Verão	Laranja	

Fonte: O autor do trabalho.

2.4.3 MNAR

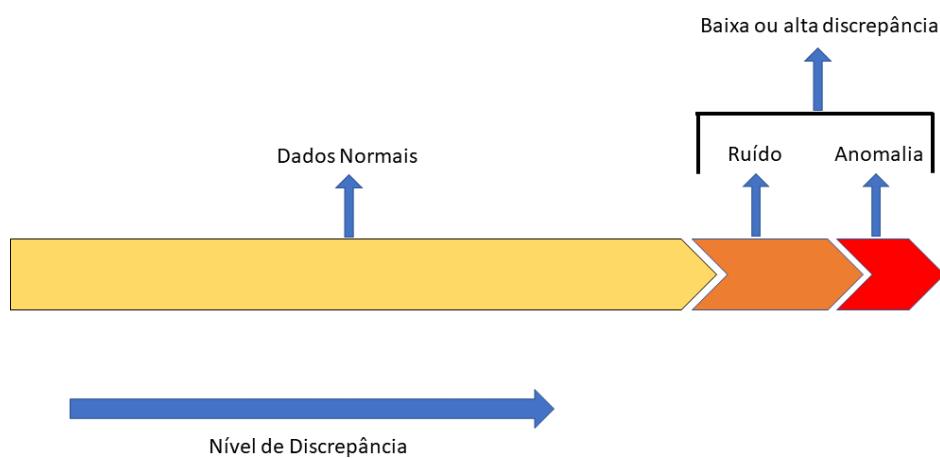
Quanto ao MNAR, este é definido como a probabilidade da resposta está relacionada com os dados que não estão presentes na amostra. (MOLENBERGHS et al., 2014) (LITTLE et al., 2016) Isto é, por algum motivo que não está no conjunto de dados, há dados ausentes. Este mecanismo permite a geração de hipóteses para justificar a ausência desses dados. Ainda na tabela 2 visualiza-se um exemplo de dados ausentes do mecanismo MNAR. O fato da receita de laranja não ter sido divulgada neste registro pode indicar que o produtor não queira preocupar os possíveis investidores (ou partes interessadas no agronegócio) devido uma possível baixa nos rendimentos.

2.5 Dados Discrepantes

Dados discrepantes ou *outliers* são dados que são significativamente diferentes dos outros dados do conjunto de dados. Também conhecidos como anomalias ou dados desviantes esses dados podem ser gerados, em geral, quando o sistema se comporta de forma não usual. Por isso, a presença e a frequência de dados discrepantes também são informações relevantes para com o conjunto de dados. Exemplos desta relevância são para sistemas de detecção de invasão, fraudes de cartão de crédito, diagnósticos médicos e estudos geológicos (AGGARWAL, 2012).

Para identificar os dados discrepantes é um pouco mais subjetivo, isto é, mais dependente de critérios feitos por quem está avaliando, assim como de qual aplicação está sendo extraído o conjunto de dados. Contudo, existe um espectro de dados normais para discrepantes como pode ser visto na figura 1. Nesta figura, justamente o limiar entre os normais para os ruídos e anomalias não são precisamente definidos, mas algoritmos de detecção de discrepância podem dar pontuação de discrepância para cada dado e utilizar este nível (AGGARWAL, 2012).

Figura 1. Exemplo de escala de discrepância



Fonte: Adaptado de (AGGARWAL, 2012).

E se os dados discrepantes não forem tratados, eles podem gerar problemas como redução da precisão do modelo de dados, aumentar a complexidade do modelo e dificultar a legibilidade dos dados (AGGARWAL, 2012) (RATHI, 2019). E para tratá-los, as formas convencionais são remoção de instâncias, filtro de dimensões, combinar essas formas convencionais com algoritmos de validação (como o k-fold); ou detecção de anomalias (como baseado em clusterização, *Selectable Mode Vocoder* (SMV) ou densidade.

2.5.1 Dados Ruidosos e Anômalos

Dados ruidosos são dados indesejaveis, dimensões ou instâncias que não estão relacionadas com o fenômeno estudado. Em geral, dados ruidosos fazem com que algoritmos de aprendizado de máquina encontrem padrões incoerentes (RATHI, 2019). Dados ruidosos e dados anômalos (ver figura 2) diferenciam-se, basicamente, na sua facilidade de percepção em uma visualização e no seu grau de impacto ao inferir sobre os dados.

Figura 2. Exemplo de ruído e de anomalia. O item A é um exemplo de dado ruidoso, pois desvia-se levemente do padrão dos dados - uma reta. Quanto ao Item B este descharacteriza significativamente o padrão dos dados - este é um exemplo de dado anômalo.



Fonte: O autor do trabalho.

Segundo Rathi (2019), dados discrepantes quando em formato tabular podem ser classificados de três formas:

- anormalidades;
- dimensões irrelevantes;
- instâncias ruídas.

As anormalidades são irregularidades tanto nas dimensões dos dados ou no evento estudado. As características irrelevantes são aquelas que não ajudam a explicar o fenômeno. E as instâncias ruídas são aquelas que desviam a forma dos outros dados. (RATHI, 2019)

Na figura 3 podem ser observados exemplos de anormalidades; dimensões irrelevantes e instâncias ruídas.

Figura 3. Exemplo de anormalidades; dimensões irrelevantes e instâncias ruídas.

Índice	Dimensão1	Dimensão2	Dimensão3	Dimensão4	Dimensão..	DimensãoN-1	DimensãoN	Evento	
Registro1									
Registro2									
Registro3									Ruído1
Registro4									Ruído2
Registro5									Ruído3
Registro6									
Registro..									
RegistroN-1									
RegistroN									

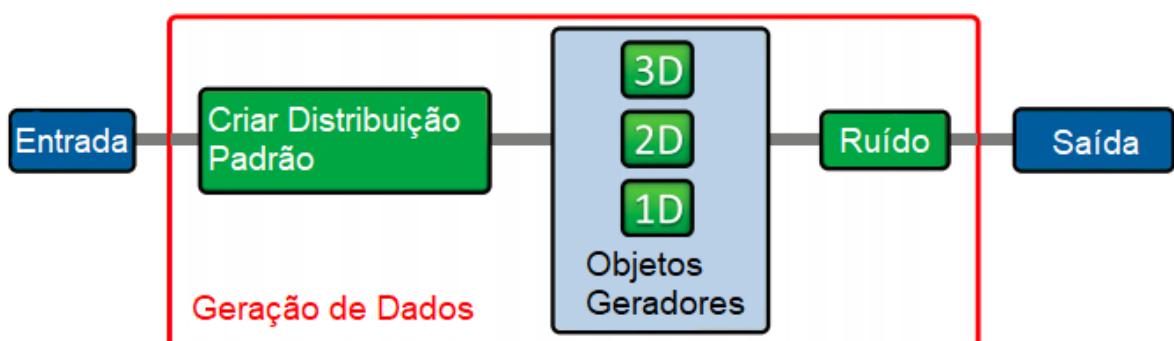
Fonte: Adaptado de (RATHI, 2019).

2.6 Trabalhos Acadêmicos Relacionados

(ALBUQUERQUE; LOWE; MAGNOR, 2011) descreveu um *framework* capaz de gerar dados sintéticos multidimensionais. O sistema como mostra a figura 4 recebe um *input* que representa algumas propriedades do conjunto de dados como número de dimensões, uma distribuição de dados padrão, tipo de dado de cada dimensão entre outros. A partir disso, é criada uma função densidade de probabilidade, com o fim de gerar um conjunto de dados padrão. Essas funções podem ser ajustadas e modeladas através de objetos. Também, essas funções podem ser de 1, 2 ou 3 dimensões. Adicionalmente, pode-se haver ruídos, para simular as irregularidades encontradas em conjunto de dados reais.

O framework apresentado também possui uma interface gráfica para auxiliar o usuário a configurar o conjunto de dados, bem como gerá-lo. Contudo, não foi encontrado uma interface para pré-visualização dos futuros dados gerados. Quanto aos tipos de dados, estes são restritos aos numéricos, quer sejam inteiros ou de ponto flutuante. No texto, nada foi citado sobre dados faltantes ou discrepantes.

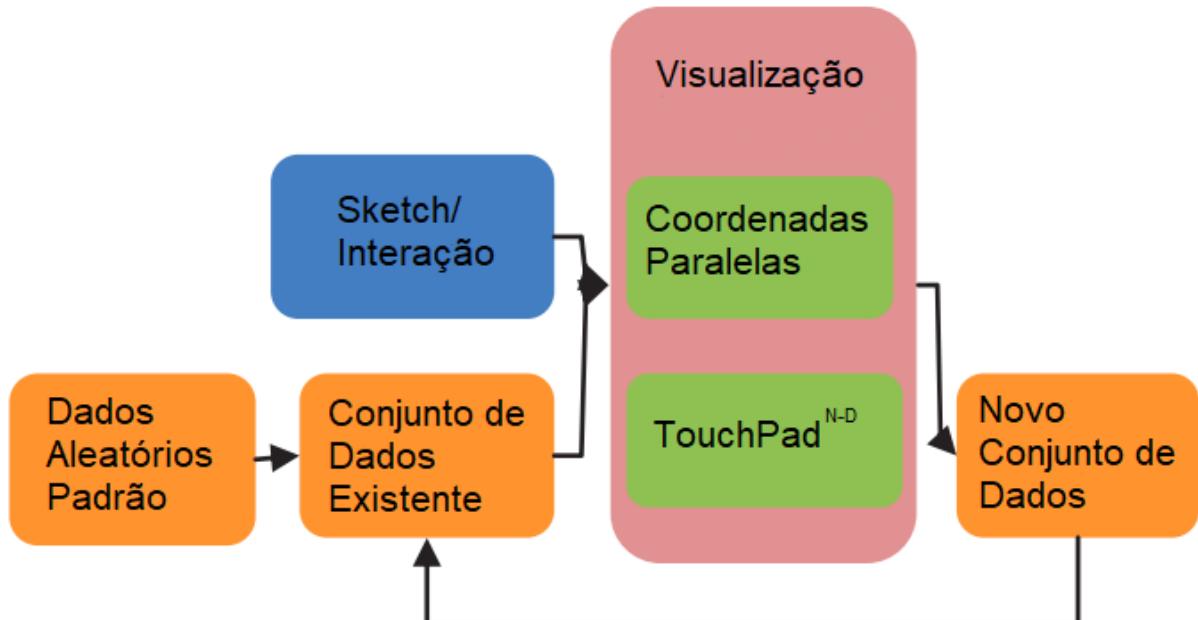
Figura 4. Visão Geral de geração de dados.



Fonte: Adaptado de (ALBUQUERQUE; LOWE; MAGNOR, 2011).

(WANG; RUCHIKACHORN; MUELLER, 2013) apresentou uma aplicação chamada SketchPad cujo principal diferencial é a capacidade de modelar, através de desenho, o comportamento das dimensões do conjunto de dados sintéticos. A priori, o usuário pode iniciar o processo de geração através do zero, de um conjunto de dados já existente, ou um conjunto de dados aleatório. A partir disso, o usuário visualiza os dados no gráfico - que pode ser as coordenadas paralelas ou o *scatterplot* - e pode modificá-lo através de cliques e arrastos. Por conseguinte, os dados podem ser gerados e isto também serve como retroalimentação do sistema. Na figura 5 é possível visualizar a visão geral do funcionamento do SketchPad como a introdução de dados, visualização e modelação utilizando o gráfico Coordenadas Paralelas e o espaço para desenho chamado de *Touchpad n-d* e a criação ou modificação de um conjunto de dados.

Figura 5. Visão Geral de geração de dados do Sketchpad.

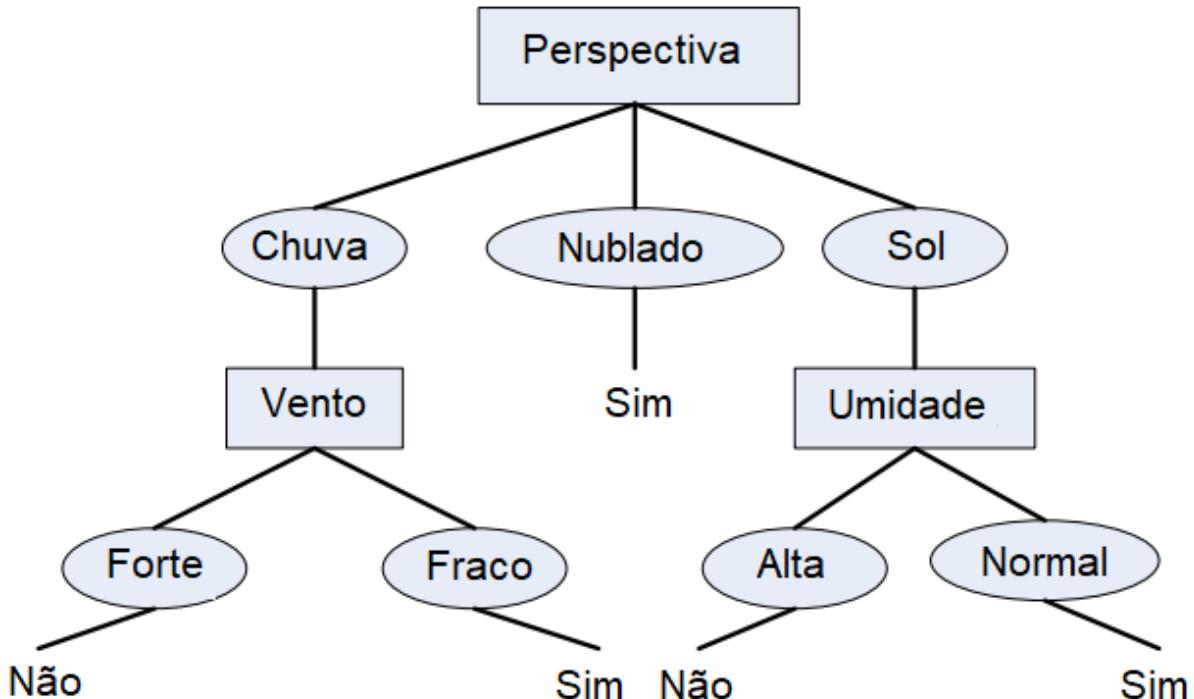


Fonte: Adaptado de (WANG; RUCHIKACHORN; MUELLER, 2013).

(LIU et al., 2016) criou um gerador de dados sintéticos a partir de avaliação de regras de aprendizagem. O sistema funciona criando regras de aprendizagem - usando algoritmos de árvore de decisão como o ID3 - baseado em dados de entrada construindo correlações entre os dados. Na figura 6 é possível visualizar uma árvore de decisão. Durante a leitura do conjunto de dados de entrada é feita a árvore de decisão e, concomitantemente, são geradas as regras de aprendizagem. Essas regras são utilizadas para gerar amostras de dados sintéticos. As regras são interessantes tanto para reutilizar os dados quanto para facilitar em determinar o comportamento dos dados.

(GARCIA; MILLAN, 2011) criaram um sistema de geração de dados sintéticos pensado para desenvolvedores que buscam testar de forma eficiente e exaustiva a sua aplicação. Esses dados podem ser configurados (ver figura 7) de acordo com as preferências

Figura 6. Exemplo de árvore de decisão para jogar tennis criado a partir de regras encontradas em um conjunto de dados.



Fonte: Adaptado de (LIU et al., 2016).

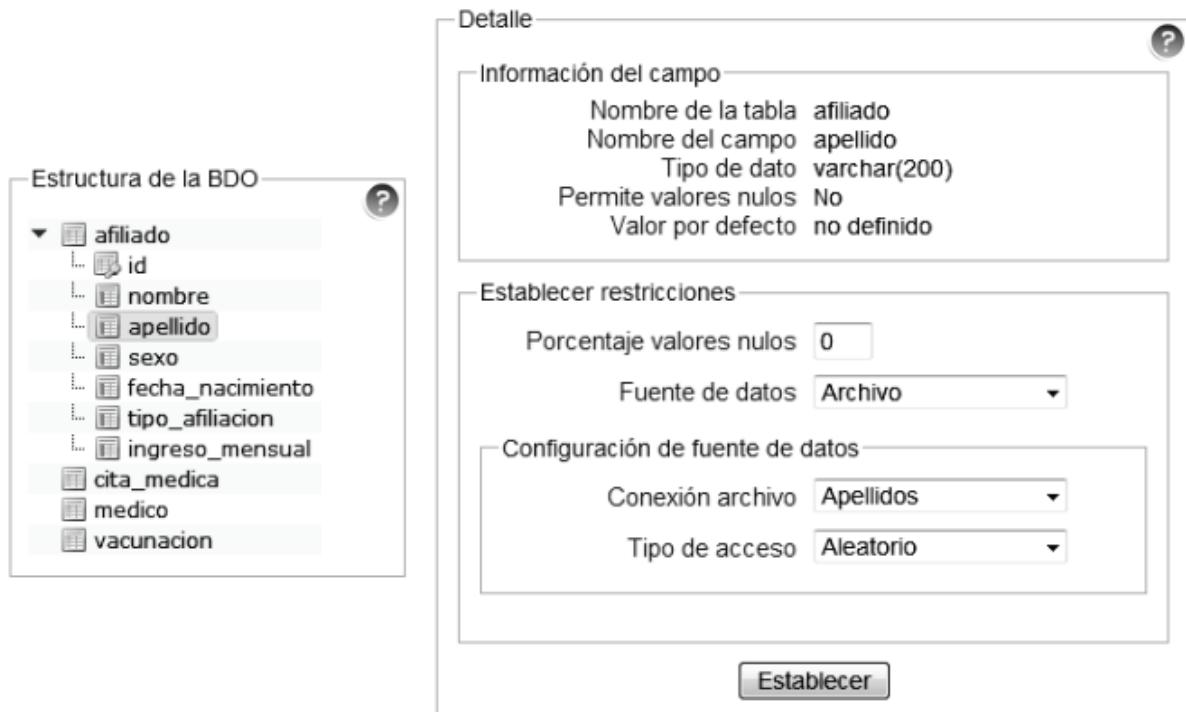
do usuário. As dimensões de dados seguem alguns padrões como a partir de fontes externas (Arquivos, Bibliotecas, Base de dados) Sequencial, Constante, Funcional, Intervalo ou Lista de valores. Contudo, os geradores, para alterar o comportamento dos dados, depende apenas dos parâmetros e não é possível misturar os geradores.

(KOFINAS; SPYROPOULOU; LASPIDOU, 2018) criou uma metodologia para gerar dados sintéticos para simular consumo de água. A metodologia é avaliada por meio de algoritmos de validação - como a visualização dos resultados e fórmulas.

Como pode ser visto na figura 8, a geração dos dados é feita a partir de 2 fases. A fase 1 investiga a distribuição dos dados. Esta fase, primeiramente transforma dados numéricos em séries temporais de 30 segundos. Em alguns casos, não há registro, para isso, é criada uma tabela de incidentes e posteriormente uma probabilidade de existência de registro para que sejam encontradas as classes usadas para construção do histograma de Pearson (DEAN; ILLOWSKY, 2009). Por fim, são comparadas funções de distribuição com a atual com o objetivo de encontrar a que mais se aproxima.

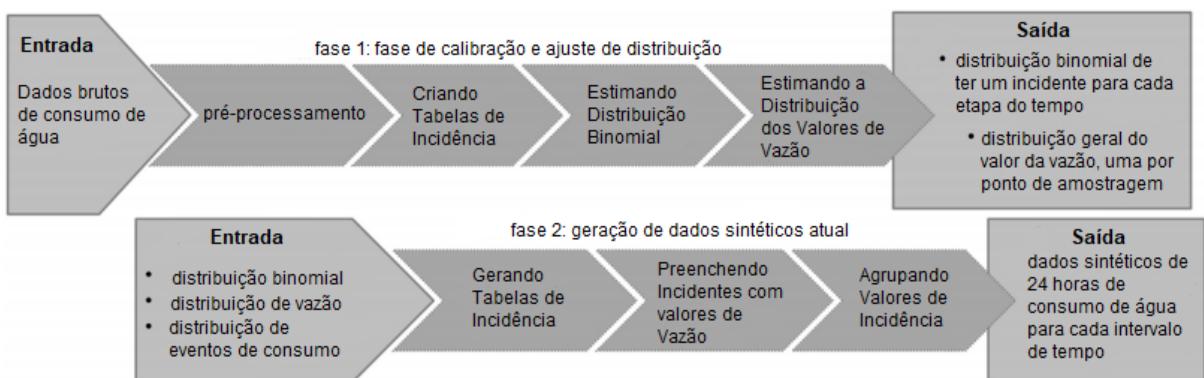
A fase 2 trata da geração de dados sintéticos utilizando a distribuição criada na fase 1 para gerar os dados para vinte e quatro horas, respeitando as características diferenciadas para dias de semana e finais de semana. Apesar de o trabalho ter como principal foco a calibração, modelagem e geração dos dados, não foi disponibilizada uma interface gráfica ao usuário para utilização das visualizações e geração dos dados faltantes e discrepantes.

Figura 7. Exemplo da interface do usuário para configuração do gerador de dados.



Fonte: (GARCIA; MILLAN, 2011).

Figura 8. Fluxo de passos para geração dos dados sintéticos.

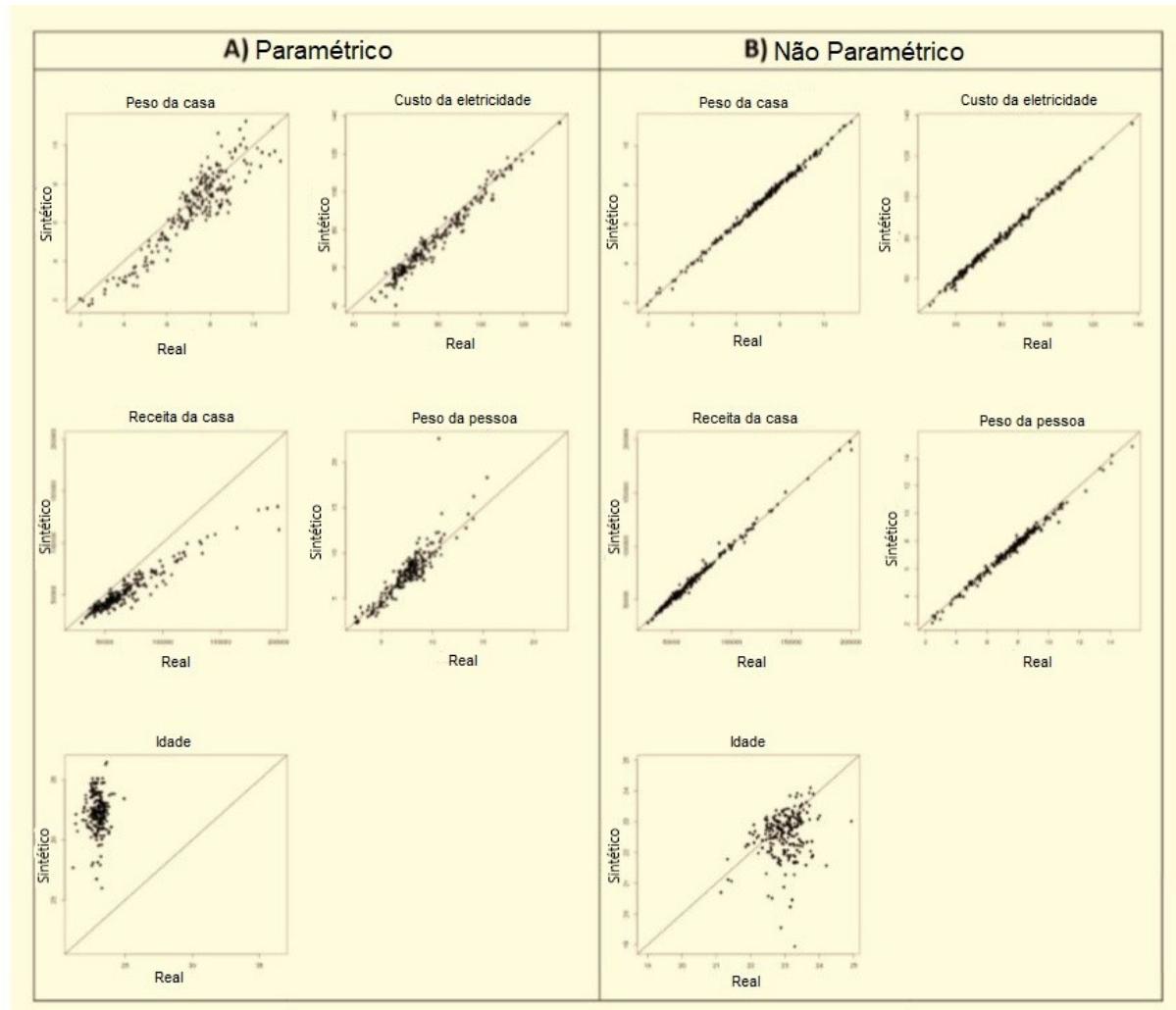


Fonte: Adaptado de (KOFINAS; SPYROPOULOU; LASPIDOU, 2018).

No estudo realizado por (SAKSHAUG; RAGHUNATHAN, 2014) foi aplicado um procedimento de simulação não paramétrica para geração de dados sintéticos de variáveis contínuas com o foco em pequenas áreas geográfica. A diferença entre a geração paramétrica e a não paramétrica é que a primeira utiliza valores sintéticos de uma distribuição normal e o não paramétrico não é baseado em distribuição. Segundo a avaliação do autor os dados sintéticos tiveram validade moderadamente alta em seus testes, mas ressalta a limitação do método não paramétrico. Em geral, o método não paramétrico se mostrou promissor para geração de dados sintéticos para pequenas áreas geográficas, mas faltam testes mais aprofundados como dados de pesquisa em larga escala para substituir os dados reais por dados sintéticos em centros de dados de pesquisa. Na figura 9 é possível observar a

comparação dos resultados da média da simulação paramétrica e da não paramétrica para cada atributo. Na simulação não paramétrica as médias dos dados sintéticos e reais ficam bem próximas, com exceção da Idade, apresentando um bom resultado para a troca de dados reais para dados sintéticos.

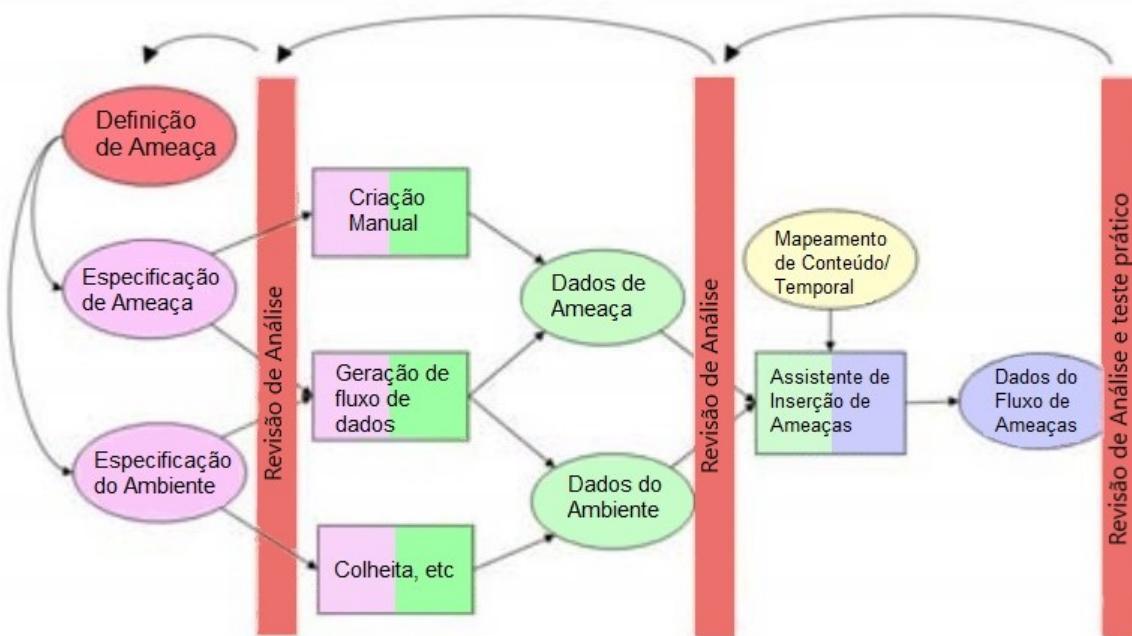
Figura 9. Comparação da média dos dados reais e sintéticos na simulação paramétrica e não paramétrica.



Fonte: Adaptado de (SAKSHAUG; RAGHUNATHAN, 2014).

O projeto *Threat Streaming Generator* (TSG) visa criar um gerador de dados sintéticos realistas com foco em dados para testes. É mostrado o fluxograma dos processos do TSG na figura 10. Primeiramente são definidos qual o tipo de conjunto de dados vai ser gerado. Em seguida são dadas três possibilidades ao usuário de inserir o ambiente e a ameaça: manualmente, através da ferramenta TSG e outras fontes. Por fim, esses dados são analisados por especialistas os quais são responsáveis pela qualidade do conjunto de dados gerado. O projeto é bem competente para desenvolvimento de dados sintéticos, inclusive com a validação pelos especialistas, o qual pode elevar o custo financeiro para geração dos dados. (WHITING; HAACK; VARLEY, 2008)

Figura 10. Fluxo de passos para geração dos dados sintéticos.



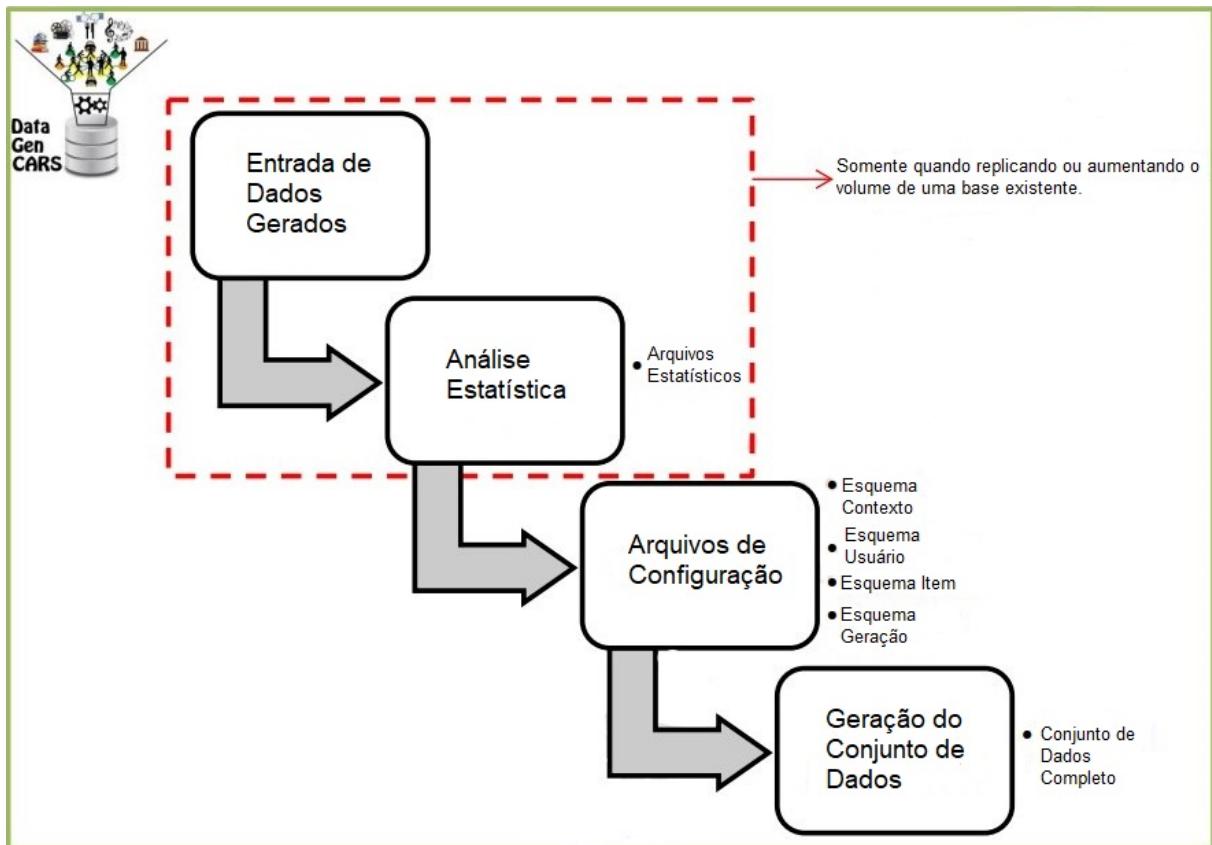
Fonte: Adaptado de (WHITING; HAACK; VARLEY, 2008).

Com o foco em CARS (Context-Aware Recommender Systems - Sistemas de Recomendação sensíveis ao contexto), o DataGenCARS (RODRÍGUEZ-HERNÁNDEZ et al., 2017) é uma ferramenta para gerar dados sintéticos de forma flexível e prática. Permitindo que o usuário possa inserir os tipos de perfis do usuário, tipos de contexto e itens, misturar dados sintéticos e dados reais com o objetivo de aumentar o realismo dos dados gerados.

Na figura 11 mostra-se o funcionamento do DataGenCARS. De início a ferramenta mostra que é possível, de forma opcional, expandir outros conjuntos de dados bem como analisá-los estatisticamente. Mesmo assim, devem ser definidos os esquemas de contexto de usuários, itens e configuração da geração para que se tenha o conjunto de dados. Dentro do tema que foi especificado (CARS) o DataGenCARS permite gerar dados de forma satisfatória bem como validar por meio de visualizações, mas não foi citado um tratamento para geração de dados faltantes e discrepantes.

Pensando em oferecer confidencialidade aos dados governamentais, (LARSEN; HUCKETT, 2012) desenvolveram um gerador de dados sintéticos que alia regressão de quartis com imputação de dados em dados faltantes a partir de dimensões similares (imputação *hot deck* (ANDRIDGE; LITTLE, 2010)) e troca de classificação. A predição de regressão de quantis é feita para proteger dados sensíveis a partir de dados não sensíveis, isto é, o conjunto de dados finais são compostos por dimensões reais - os não sensíveis - e sintéticos - para dimensões sensíveis. Em alguns casos os dados sintéticos são similares aos reais e para isso foi utilizado a imputação *hot deck* junto com a troca de classificação, para

Figura 11. Fluxo de passos para geração dos dados sintéticos.



Fonte: Adaptado de (RODRÍGUEZ-HERNÁNDEZ et al., 2017).

garantir a aleatoriedade e confidencialidade dos dados. Contudo, nada foi citado sobre a geração de dados sintéticos ou faltantes.

Dos trabalhos realizados, nem todos apresentaram uma interface gráfica ao usuário, ou variedade de geradores, com tratamento de dados faltantes ou discrepantes. Também sobre a validação por visualização poucos trabalhos apresentam integração dos dados gerados com uma ferramenta de visualização em tempo real.

2.7 Aplicações Relacionadas

DTM Data Generator (DTM Soft., 2019) é uma plataforma de geração de dados sintéticos que existe desde 1998. Esta possui suporte para geração de dados em arquivos, em banco de dados, também para *Big Data*. Possui suporte multiplataforma, através da versão *multiplatform runtime*, contudo esta é limitada quando comparado à versão Windows, o qual suporta a versão para servidor também. É válido destacar que o DTM Data Generator é um software pago para utilização oferecendo uma versão para testes. Além disso, há categorias de versões pagas, que vão desde limitações de geração (Standart - Professional) à vantagens mais técnicas (Professional - Enterprise).

O DTM Data Generator possui uma vasta coleção de funcionalidades, as quais são acrescentadas de acordo com a versão do software. Adotando a versão mais cara, a lista de funcionalidades é composta por geração de dados em JSON, XML, CSV ou geração por separador customizado. Também permite gerar dados por arquivo DSN (Database Source Name), gerar dados por linha de comando e gerar um arquivo SQL para que não seja necessário conexão com banco de dados.

É possível gerar cerca de nove vírgula dois sextilhões de registros por geração, modos de atualizar dados existentes (adicionar, substituir e ofuscar dados pessoais), e suporte para bibliotecas de dados realistas. A plataforma disponibiliza entrada de dados através de SQL, XML, JSON, pela WEB através de HTTP ou FTP, XLSM, arquivos de texto e *scripts* em Python. Também é possível visualizar e testar os dados gerados, bem como gerá-los nos principais arquivos de texto (TSV, CSV, DSV, JSON, XML) e banco de dados (MS SQL Server, Oracle, DB2, MySQL, PostgreSQL, Informix, Sybase, SQLite e Firebird).

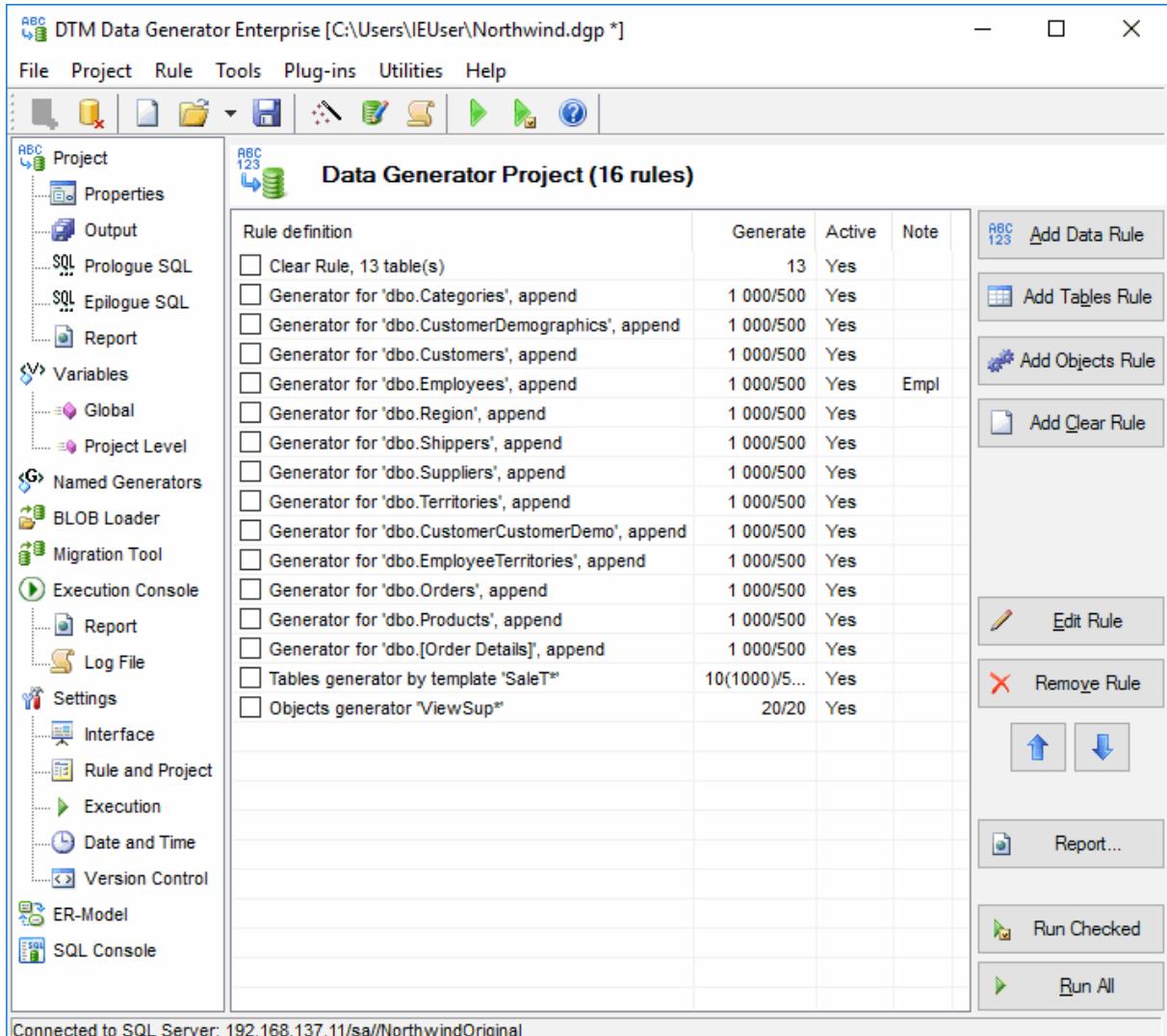
Há uma suíte de produtos relacionados fornecidos pela DTM soft. Além do gerador de dados, há o gerador de dados XML para teste de aplicação (*DTM Test XML Generator*); um gerador de planilhas Excel (*DTM Data Generator for Excel*); testador exaustivo - teste de estresse - de banco de dados (*DTM DB Stress*); bem como editor, visualizador (*DTM Data Editor*), comparador e sincronizador de banco de dados (*DTM Data Comparer*), entre outros. Não foi encontrado uma documentação com mais detalhes dos geradores e se há suporte para geração de dados faltantes, bem como discrepantes.

O SQL Data Generator (Red Gate, 2019) é um software que compõe uma suíte de ferramentas (chamada de SQL Toolbelt) da Red Gate. O software é exclusivo para o ecossistema Windows, com suporte ao Windows 7 ou superior, à versão Windows Server, SQL Server (2008 ao 2017), .NET e Oracle. Este produto é distribuído através de licenças pagas e vitalícias, com atualizações gratuitas e no mínimo 1 ano de suporte gratuito. Vale ressaltar que é possível testar o produto por 14 dias gratuitamente.

O *SQL Toolbelt* tem funcionalidades bem delimitadas e a função do *Data Generator* é popular um banco de dados. A população da base de dados é feita ao escolher, primeiramente, uma tabela do banco. A partir disso, escolhe-se um gerador para cada coluna da tabela. Um gerador tem classificação fortemente baseada na realidade, isto é, possui geradores como palavras relacionadas à compras, pagamentos, pessoas (primeiro e último nome), dado geográficos e afins.

A ferramenta também disponibiliza a geração a partir de expressões regulares *Regex generator* e *scripts* em Python. Por se tratar de banco de dados, também há checagem e tratamento de *constraints*, *Foreign keys* e *Dependencies*. O *SQL Data Generator* também permite lidar com arquivos XML, quer seja para geração de valores XML, como utilizar como dados de entrada, além de mesclá-los com o *Regex generator*.

Figura 12. Usando o DTM Data Generator.



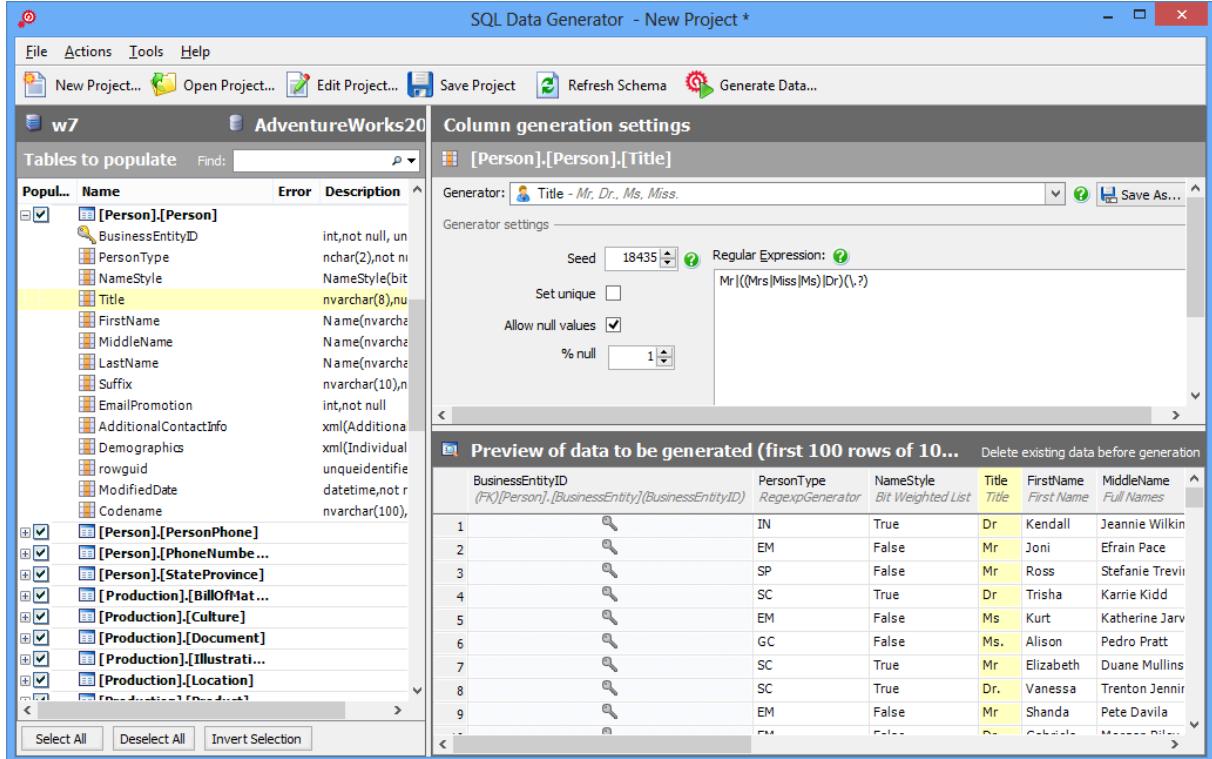
Fonte: (DTM Soft., 2019).

Quanto ao *SQL Toolbelt* oferecido pela *Red Gate*, ele conta com 2 versões, o completo com quartoze programas e o *essentials* com dez. Entre os mais relevantes, pode-se citar o *SQL Data Compare*, *SQL Data Generator*, *SQL Test*, *SQL Backup Pro* e *SQL Scripts Manager*.

Na documentação do SQL (Red Gate Documentation, 2019) é possível encontrar os geradores de dados sintéticos. Entre os exemplos estão gerador por *regex*, importação de arquivo, lista ponderada etc. Não houve uma citação por um gerador de dados discrepantes ou faltantes, contudo é possível criar geradores de dados manualmente utilizando XML em conjunto com classes pré-definidas. Também o uso de embaralhador de texto, bem como a lista ponderada seja possível gerar dados faltantes e discrepantes.

Microsoft Visual Studio (MICROSOFT, 2019) é um pacote multiplataforma de programas da Microsoft para desenvolvimento de *software*. Este é composto por quatro versões (*Express*, *Professional*, *Premium* e *Ultimate*) e a opção de gerar dados para teste

Figura 13. Usando o Redgate SQL Data Generator.



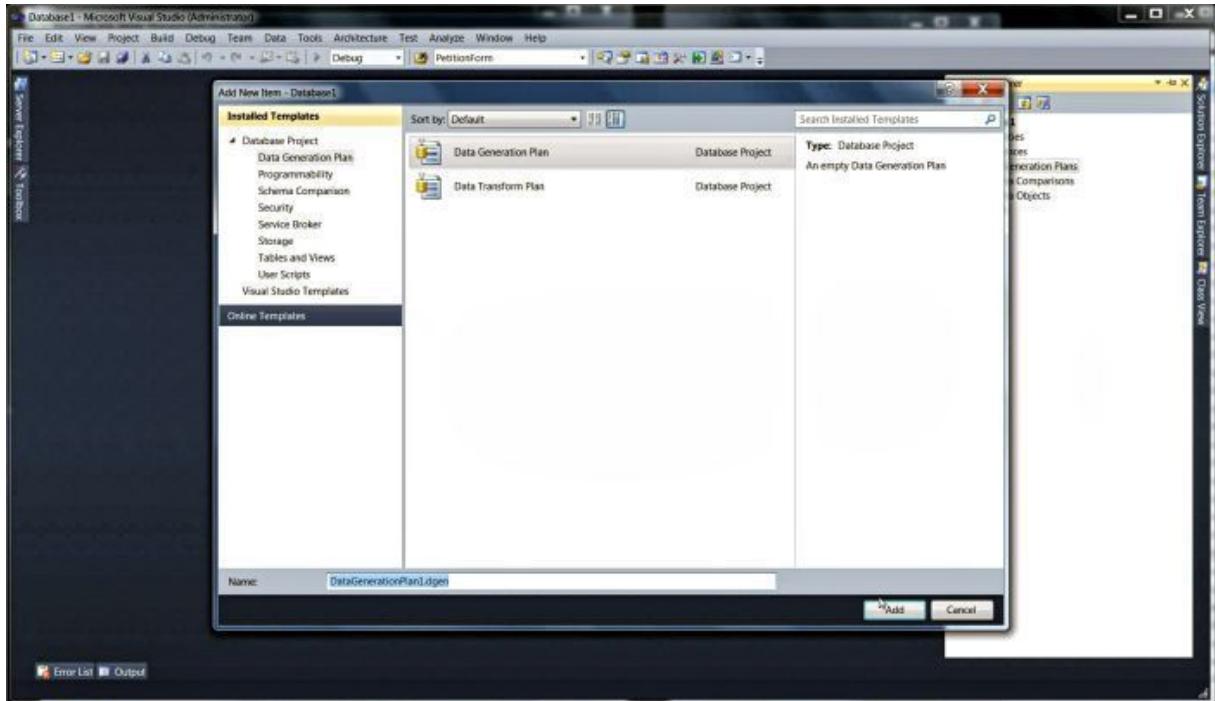
Fonte: (Red Gate Documentation, 2019).

está disponível a partir da versão *Premium*. O foco é permitir que verifique o comportamento do banco de dados, sem relacioná-lo com os dados da aplicação em produção.

Para gerar os dados de teste, deve-se utilizar os geradores de dados (*Data Generators*) que são correlacionados às tabelas do banco de dados. Os geradores podem ser dos mais primitivos (Binários, Inteiros, Data, *Float*), como de Imagem, Dinheiro, Expressão Regular, Categórico, entre outros. Entre os geradores genéricos pré-definidos, não foi encontrado suporte para dados faltantes ou dados discrepantes, mas há a possibilidade de criar geradores personalizados. Também é disponibilizado um Plano de Geração de Dados (*Data Generation Plan*), feito em XML, que contém informações do banco de dados, o tipo de dados de cada gerador e a quantidade de dados para ser gerado. Este plano serve basicamente para reutilização da lógica de teste.

Test Data Generator (DEVART, 2018) é uma ferramenta GUI (*Graphical User Interface*) desenvolvida pela dbForge para gerar dados de teste para banco de dados SQL desde 1997. O software possui mais de duzentos geradores predifinidos e configuráveis, os quais permitem a geração de dados mais inteligentes, isto é, mais próximos da realidade, como nomes, localização, dados de saúde e afins. Quanto à compatibilidade, este é exclusivo do ecossistema Windows, com suporte ao *Windows 7*, *Windows Server 2008*, *SQL Server Azure* 2008 ou versões superiores. Além da GUI, também há o suporte para geração de dados a partir da linha de comando. O produto é distribuído sob licenças pagas e

Figura 14. Usando o Microsoft Visual Studio.



Fonte: (MICROSOFT, 2019).

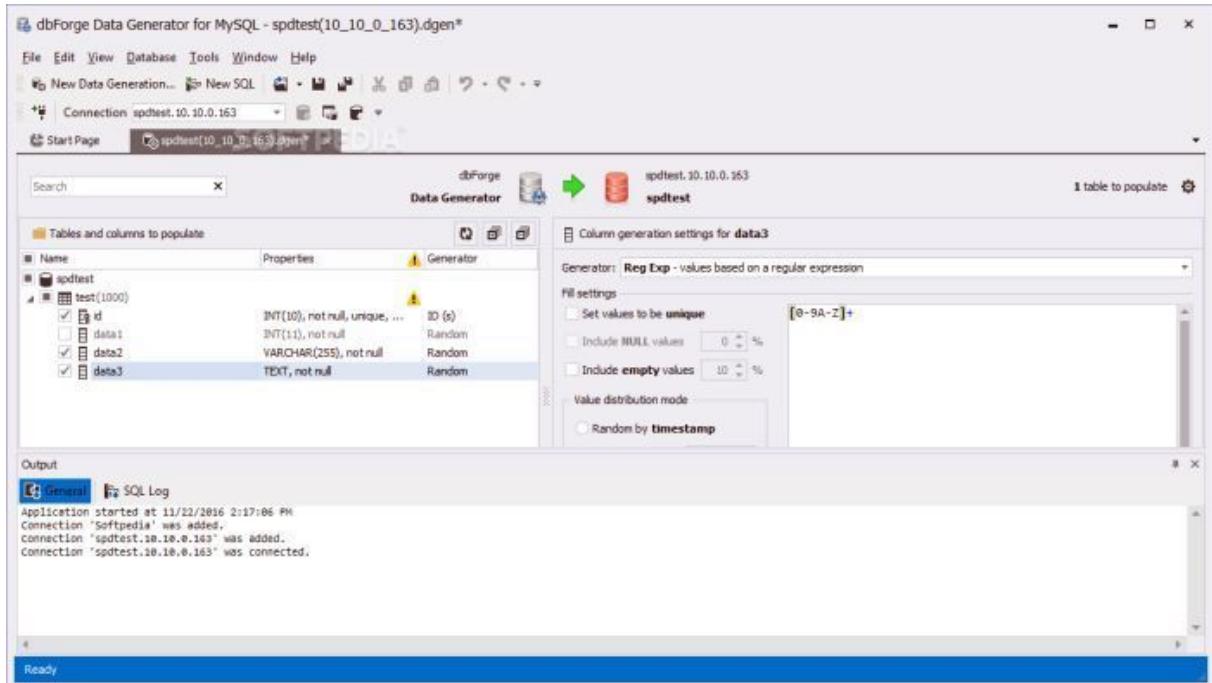
vitalícias, porém, com suporte ao cliente com tempo limitado e com trinta dias gratuitos para avaliação.

Para usar o *dbForge Test Data Generator*, é preciso fazer uma conexão com banco de dados. A partir disso, utiliza-se os *Data Generators* para determinar o comportamento dos dados para determinada coluna da tabela selecionada no Banco de dados. Os Geradores de dados podem ser do tipo *Basics* e *Advanced*. Do primeiro tipo, são formas mais próximas dos dados primitivos, como datas, texto *lorem ipsum*, JSON, *ReGex*. Já o avançado conta com número de cartão de crédito, aniversário, número de conta bancária internacional, IPv4, *hash* de senhas. A geração de dados resume-se à população de banco de dados, não há uma forma de exportar os dados em arquivos como CSV e JSON. Ao acessar a documentação (Devart Documentation, 2019) dos geradores, elas não estavam disponíveis (código HTTP 404), logo não foi encontrado mais informações inclusive sobre dados faltantes ou discrepantes.

Há um suíte exclusivo para SQL Server, contudo também para Oracle, MySQL, PostgreSQL entre outros. Neste suíte, há várias ferramentas que auxiliam na manutenção, mas não, necessariamente, a geração de dados, a exemplo de um pre-visualizador de dados. Destes, pode-se citar um comparador de dados, criador de *querys*, um monitor - para supervisão do banco de dados - e afins.

Mockaroo (MOCKAROO, 2019) é um *web site* e *framework* para desenvolver dados de teste. Há um total de cento e quarenta e três geradores, sendo a maioria considerados

Figura 15. Usando o dbForge Test Data Generator.



Fonte: (DEVART, 2018).

geradores realistas. Por ser um site, é possível acessá-lo por qualquer sistema operacional, dependendo apenas de conexão com a internet. O produto possui versões gratuita e pagas. - *Free, Silver, Gold, Enterprise* as quais variam no *host*, o qual pode ser do Mockaroo ou privado, máximo de registros por download, velocidade de download e preço.

Na tela inicial, é possível escolher o nome da coluna, o tipo de gerador, entre outras opções. O campo *blank* é possível determinar uma porcentagem de dados que ficarão em branco em ordem aleatória. Por conta da característica aleatória, o mecanismo de dados faltantes utilizado é o MCAR.

Ao lado de *blank* há o botão de fórmula, o qual é possível operar uma função com o valor gerado. Essas funções podem ser matemáticas ou outras disponibilizadas de acordo com a sintaxe de fórmula do Mockaroo. A partir dessa funcionalidade, é possível gerar dados discrepantes tanto ruidosos como anômalos, dependendo da função e parâmetros inseridos.

Ainda na tela inicial encontra-se o botão para *download* dos dados, pré-visualização dos mesmos, - sem gráficos, apenas tabular e CSV - algumas configurações como quantidade de linhas, formato dos dados para *download*, botão para clone ou deleção de banco de dados, e importação de dados csv/Excel ou SQL.

Outro serviço interessante da ferramenta é Mockaroo APIs (Mockaroo Documentation, 2019). Este consiste em baixar dados programaticamente através de requisições REST (*Representational State Transfer*). As requisições podem ser feitas de duas formas,

a *Generate API* - gera os dados através de um banco de dados salvo e os envia pelo corpo de uma requisição - e *Mock APIs* que simula um *back-end* como tratamento de parâmetros e simulação de erros. É pensado para desenvolvimento ágil de aplicações *front-end*, isto é, sem perder muito tempo com o *back-end* inicialmente.

Figura 16. Usando o Mockaroo.

Field Name	Type	Options
<input type="text" value="id"/>	Row Number	blank: 0 % <input type="button" value="fx"/> <input type="button" value="x"/>
<input type="text" value="first_name"/>	First Name	blank: 0 % <input type="button" value="fx"/> <input type="button" value="x"/>
<input type="text" value="last_name"/>	Last Name	blank: 0 % <input type="button" value="fx"/> <input type="button" value="x"/>
<input type="text" value="email"/>	Email Address	blank: 0 % <input type="button" value="fx"/> <input type="button" value="x"/>
<input type="text" value="gender"/>	Gender	blank: 0 % <input type="button" value="fx"/> <input type="button" value="x"/>
<input type="text" value="ip_address"/>	IP Address v4	blank: 0 % <input type="button" value="fx"/> <input type="button" value="x"/>
<input type="button" value="Add another field"/>		
# Rows: <input type="text" value="1000"/> Format: <input type="button" value="CSV"/> Line Ending: <input type="button" value="Unix (LF)"/> Include: <input checked="" type="checkbox"/> header <input type="checkbox"/> BOM		
<input style="background-color: #2e7131; color: white; padding: 5px; border-radius: 5px; border: none; font-weight: bold; margin-right: 10px;" type="button" value="Download Data"/> <input type="button" value="Preview"/> <input style="border: none; border-bottom: 1px solid black; padding: 0 5px;" type="button" value="More"/> Want to save this for later? Sign up for free!		

Fonte: (MOCKAROO, 2019).

Na tabela 3 é apresentada uma comparação de funcionalidades entre as aplicações estudadas. A dimensão B.D. diz respeito se a aplicação é capaz de se comunicar com um banco de dados para criar novas instâncias de dados. A dimensão Arquivo é referente à geração de dados e salvá-los em um arquivo. O suporte aos sistemas operacionais como Windows, Linux e MacOS é mostrado na dimensão S.O. O DTM possui suporte multiplataforma por meio da versão *multiplatform runtime*, mas é limitada em relação à versão Windows.

A dimensão D.F. e D.D. apresenta o suporte à geração de dados faltantes e discrepantes, respectivamente. As instâncias que possuem sim* nessas dimensões significam que há suporte a geração de dados sintéticos e discrepantes, mas são feitos através de geradores personalizados, isto é, não há geradores predefinidos para esse fim. E o não* é porque a documentação sobre os geradores estava offline para consulta durante o desenvolvimento desse estudo. Quanto à dimensão Preço, esta refere-se ao acesso gratuito ou não a todas as funcionalidades da aplicação. O Gratuito* significa que há acesso gratuito, mas é limitado em processamento, havendo outras versões pagas.

Tabela 3. Tabela de comparação das funcionalidades de cada aplicação. B.D. = Banco de Dados; S.O. = Sistema Operacional; D.F. = Dados Faltantes; D.D. = Dados Discrepantes; W.S. = *Web Service*.

Nome	B.D.	Arquivo	S.O.	D.F.	D.D.	W.S.	Preço
<i>Blocks</i>	Não	Sim	Vários	Sim	Sim	Sim	Gratuito
<i>DTM D.G.</i>	Sim	Sim	Vários*	Não	Sim*	Sim	Pago
<i>SQL D.G.</i>	Sim	Não	Windows	Sim*	Sim*	Não	Pago
<i>MS Visual Studio</i>	Sim	Não	Vários	Sim*	Sim*	Não	Pago
<i>Test D.G.</i>	Sim	Não	Windows	Não*	Não*	Não	Pago
<i>Mockaroo</i>	Não	Sim	Web	Sim	Sim	Sim	Gratuito*

Fonte: O autor do trabalho.

3 Arquitetura do projeto

Em termos de organização do projeto, foi adotado o padrão de arquitetura de *software* MVC (*Model, View, Controller*), um modelo incremental de desenvolvimento, com reuniões diárias para discussão de problemas e melhorias, e utilização da ferramenta Trello (ATLASSIAN, 2019) para organização e consistência das informações.

Quanto ao desenvolvimento, foi utilizada a linguagem Javascript com foco para *Desktop*, por meio da utilização do *Framework Electron* (OpenJS Foundation, 2019a). Também foi adicionado o *jQuery* para agilizar a codificação do projeto e o Node.js 10 (OpenJS Foundation, 2019b) para acessar recursos do sistema operacional, para o desenvolvimento do *Web Service* e também para dar suporte ao Electron. Do Javascript foi utilizado o Ecmascript 6 e seus novos recursos como o desenvolvimento assíncrono com as *Promises* e *arrow functions*.

3.1 Casos de uso do sistema

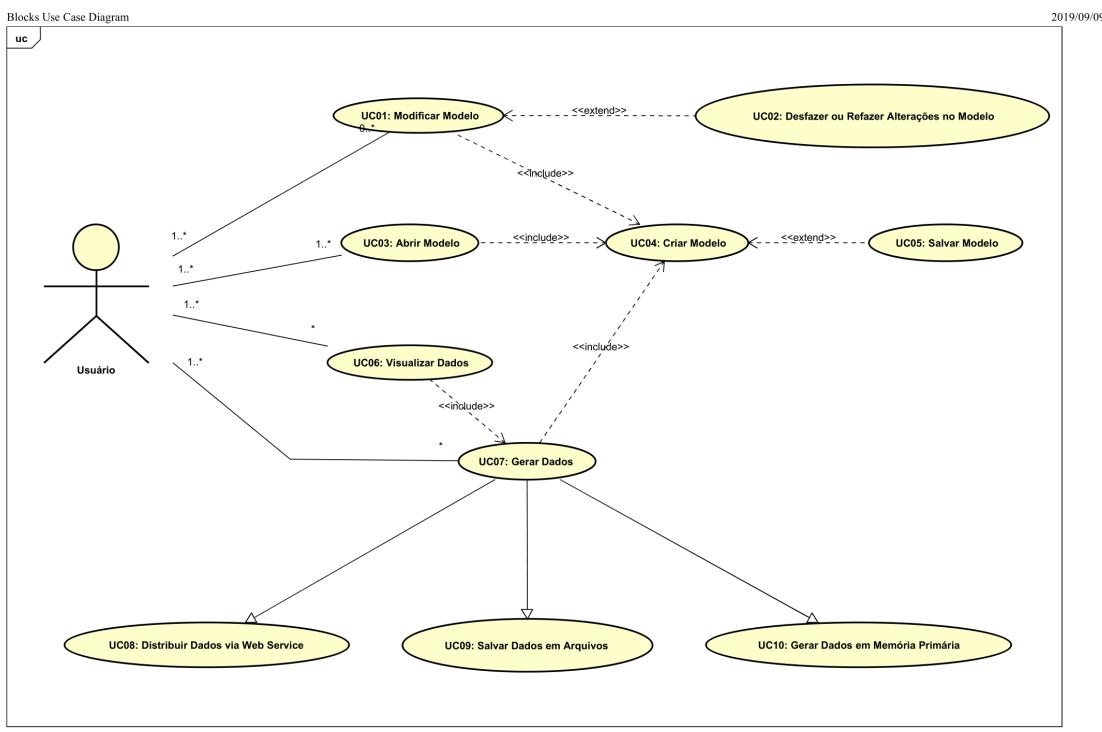
O diagrama de casos de uso visa demonstrar as diferentes formas de um usuário pode utilizar um sistema (Lucid Software Inc., 2019). Por conta disso, é utilizado o Diagrama de Casos de Uso para demonstrar as principais funcionalidades do sistema Blocks. Na figura 17 são encontrados um total de dez casos de uso, nomeados de UC01 a UC10.

No UC01 mostra que é possível modificar um modelo, isto é, adicionar ou remover dimensões, alterar geradores e afins; também há inclusão do UC04 e a possibilidade de desfazer ou refazer as alterações (UC02). O UC04 representa a criação do modelo. Outros casos de uso que possuem relação com UC04 é o UC03 e UC07 como inclusão e UC05 como extensão.

UC03 é uma funcionalidade para reaproveitamento do modelo, também está ligado à validação de teste por outros pesquisadores. UC07 é o principal caso de uso do Blocks, o qual refere-se à geração de dados, progredindo para o *Big Data* ou não. Esta geração pode ser feita, especificamente, de três formas: gerando dados em memória, para ser utilizado dentro do sistema (UC10); salvar em arquivos - TSV, CSV e JSON - (UC09); e distribuir os dados através de requisições HTTP (UC08).

Incluindo a UC07, mais especificamente a UC10, a UC06 permite que o usuário visualize os dados. Assim, a visualização pode ser rápida (*preview*) ou detalhada (*VisApplication*), tendo um panorama do comportamento dos dados. Com isso, pode-se validar os dados, ter *insights* para aprimorar o modelo e afins.

Figura 17. Diagrama de Caso de uso do Blocks

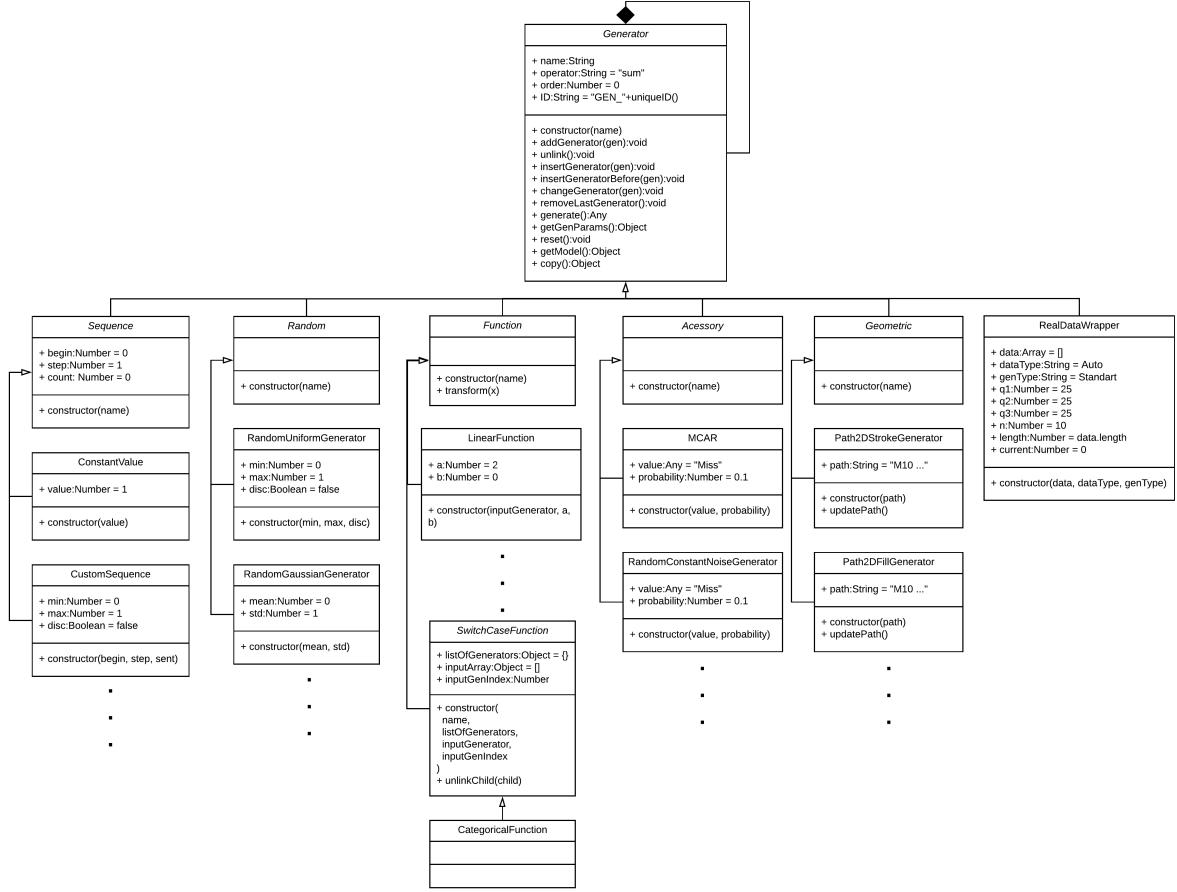


Fonte: O autor do trabalho.

3.2 Diagrama de Classes

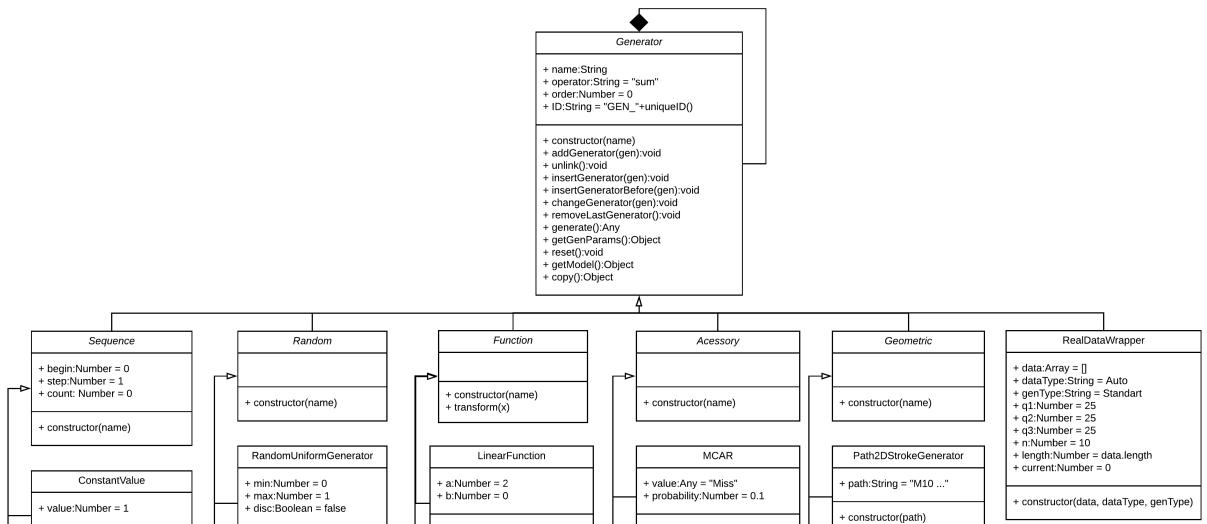
No diagrama de classes visto na figura 18 é possível encontrar a estrutura da classe geradores de forma resumida. Há uma classe abstrata chamada *Generator* que possui a maioria dos métodos e atributos, sendo que alguns deles são sobrescritos nas subclasses. Na segunda geração de classes (ver figura 19), elas existem para identificar as categorias - exceto *Real Data Wrapper* e não adicionam novas propriedades - com exceção da classe *Sequence*, por exemplo. Então a terceira geração de classes (ver figura 20) são os geradores em si, que herdam das suas respectivas categorias, mas uma exceção é a classe *SwitchCaseFunction* a qual possui a propriedade de permitir geradores distintos de acordo com o valor.

Figura 18. Diagrama de Classes dos geradores do Blocks.



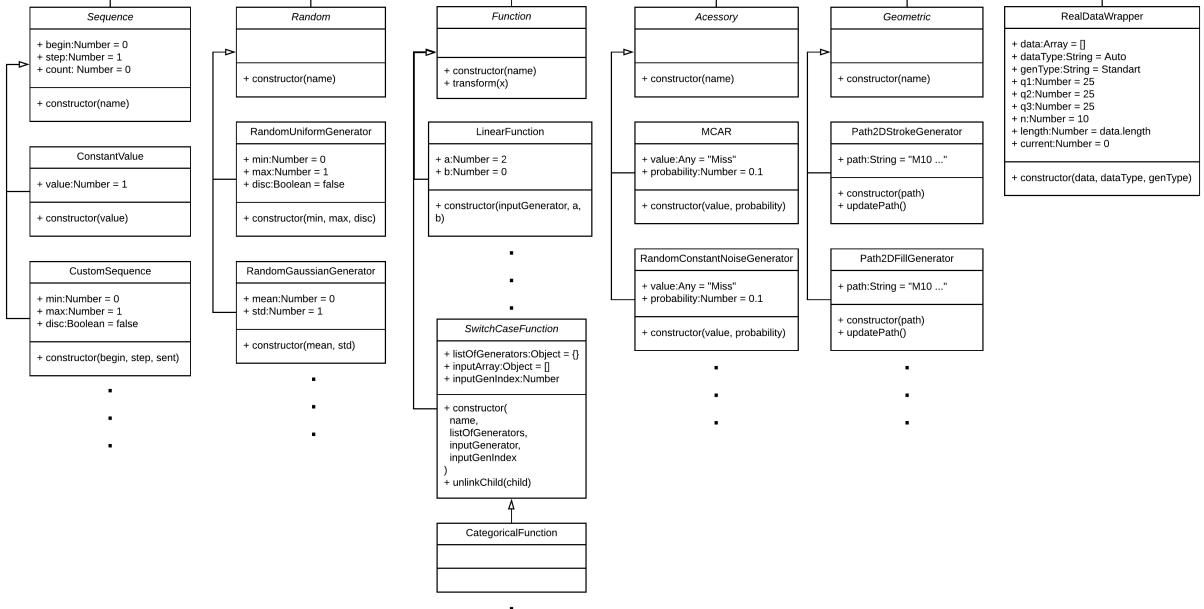
Fonte: O autor do trabalho.

Figura 19. Diagrama de Classes (2^a Geração) dos geradores do Blocks.



Fonte: O autor do trabalho.

Figura 20. Diagrama de Classes (3^a Geração) dos geradores do Blocks.



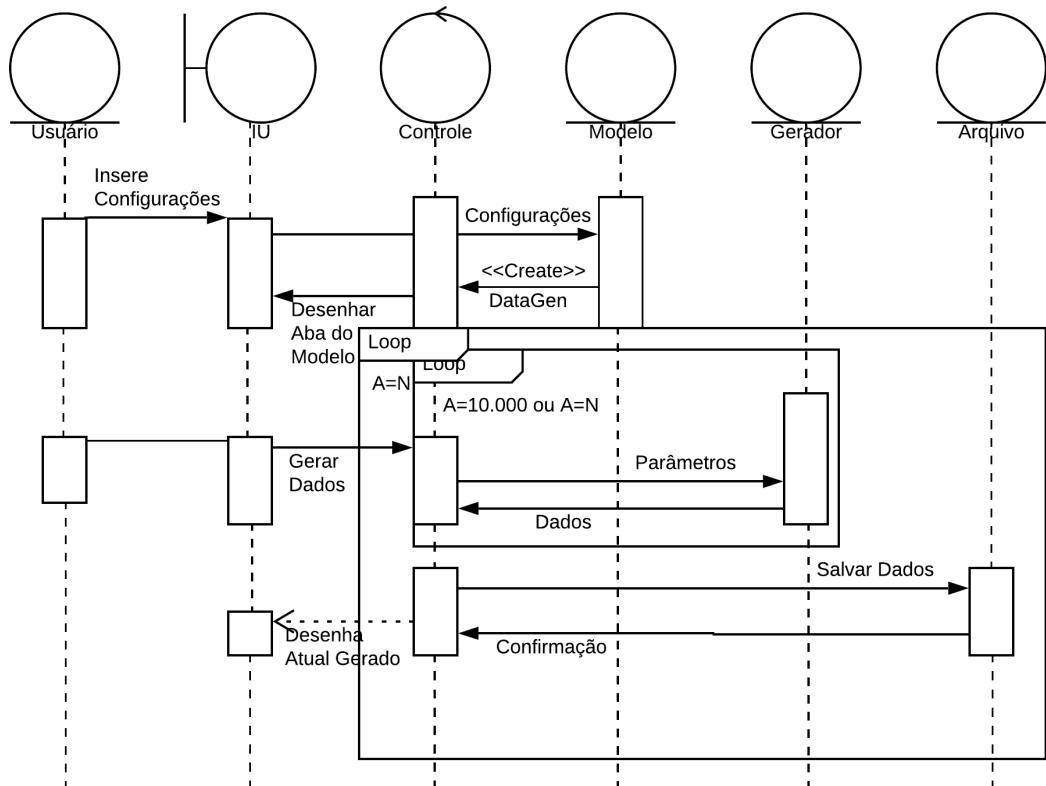
Fonte: O autor do trabalho.

3.3 Diagrama de Sequência de atividades no Blocks

Para gerar os diagramas de sequência do Blocks, foram utilizados os casos de uso UC06 e UC09, vistos na figura 17. No diagrama 21 percebe-se que há a introdução das configurações do modelo pelo usuário, então o sistema cria/modifica o objeto modelo (UC01). Mais abaixo, há 2 loops para geração dos dados, o primeiro é a geração pelos geradores presentes no modelo até um certo limite, então esse conjunto de dados é salvo no arquivo e em seguida há uma resposta ao usuário de que houve progresso.

No diagrama de sequência visto na figura 22 é em relação ao UC06. Primeiramente, assim como na geração do arquivo, há o uso do UC01. Em seguida há geração dos dados em memória referente ao UC10. Com esses dados em memória, o *preview* é desenhado automaticamente. A partir dos dados, há uma comunicação via *WebSocket* com o VisApplication enviando os dados. O VisApplication retorna as opções de visualização, o usuário escolhe uma e então é retornado a visualização para que esta seja renderizada.

Figura 21. Diagrama de Sequência para geração de dados em arquivos no Blocks.



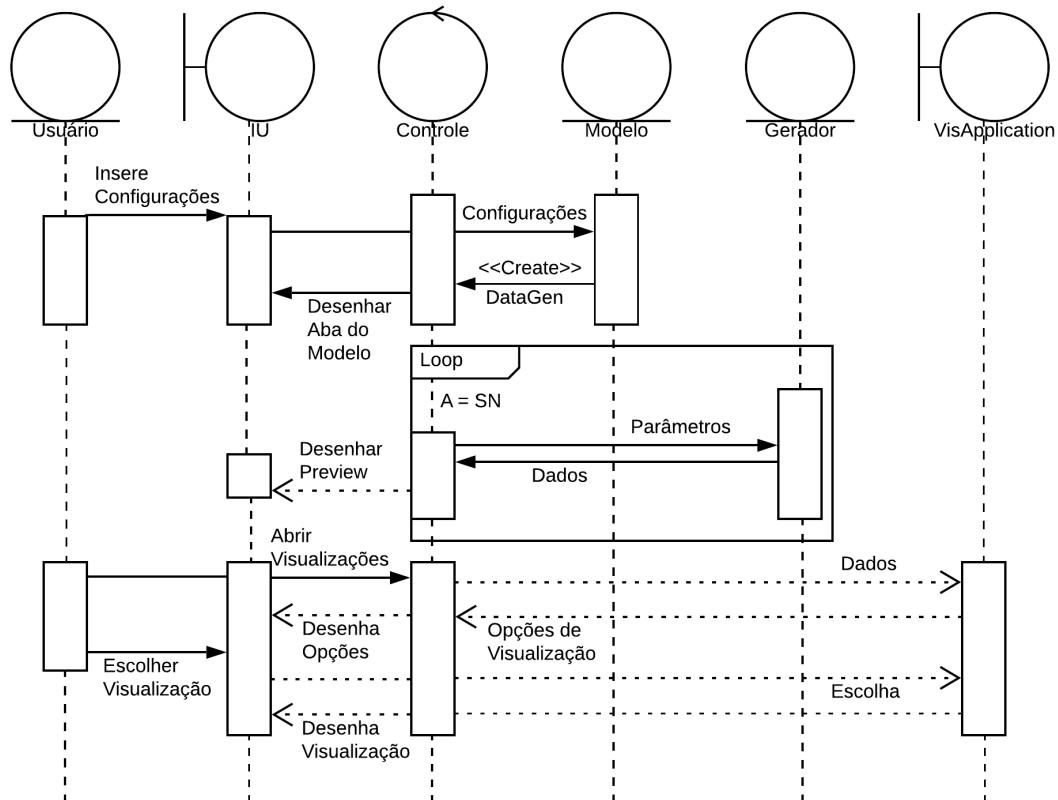
Fonte: O autor do trabalho.

3.4 Pseudocódigos dos geradores desenvolvidos

São apresentados pseudocódigos nessa seção com o objetivo de demonstrar como foram implementados os métodos para gerar dados sintéticos e discrepantes. Primeiramente é mostrada a classe Generator (ver figura 23), que utiliza o padrão de projeto *Decorator* com o fim de encadear os geradores. Nessa classe estão os principais atributos como o nome do gerador, os geradores encadeados (gerador), um operador (soma, subtração, divisão etc) entre outros. Quanto ao método gerar, este verifica se há um gerador encadeado, se não ele retorna o valor recebido. Por ser uma classe abstrata, a classe Generator recebe o valor da subclasse instanciada.

O gerador MCAR (ver classe na figura 24) tem como atributos um valor (o dado faltante recebe esse valor) e uma probabilidade, isto é, uma taxa aproximada de dados faltantes nessa dimensão. Para gerar, ele verifica a probabilidade e então retorna o valor faltante, se não o valor do gerador encadeado.

Figura 22. Diagrama de Sequência para visualização de dados no Blocks.



Fonte: O autor do trabalho.

Figura 23. Pseudocódigo sobre a classe Generator do Blocks.

```

classe Generator{
    construtor(nomeDoGerador) {
        nome = nomeDoGerador;
        operador = operadorDeSoma ou semOperador;
        gerador = naoDefinido;
        ordem = 0;
        ID = IDUnico();
    }
    gerar(valorVindoDaSubclasse){
        se(gerador e operador){
            retorna ultimoValorGerado = operador(valorVindoDaSubclasse, gerador.gerar());
        }
        retorna ultimoValorGerado = valorVindoDaSubClasse;
    }
    (...)

}
    
```

Fonte: O autor do trabalho.

Figura 24. Pseudocódigo sobre a classe MCAR do Blocks.

```

classe MCAR estende Acessorio{
    construtor(valorDado, probabilidadeDada) {
        superclasse("MCAR");
        valor = valorDado ou "Miss";
        probabilidade = probabilidadeDada ou 0,1;
    }
    gerar(){
        se(valorAleatorio() > probabilidade) retorna ultimoValorGerado = superclasse.gerar(0);
        retorna ultimoValorGerado = valor;
    }
    (...)
```

Fonte: O autor do trabalho.

A classe do gerador MNAR (ver figura 25) foi dividida em 4 figuras, por conta do seu tamanho. Nesta figura, é possível ver os atributos no método construtor e 3 condições dentro do método gerar. O MNAR possui valor e probabilidade assim como o MCAR (figura 24), mas possui outros atributos para tratar os 3 tipos de dado. primeiroPadrao e segundoPadrao são úteis para o usuário colocar os valores de intervalo ou uma lista de categorias (somente para o primeiroPadrao). E a máscara é utilizada para determinar o formato do tempo utilizado e funciona somente para dados temporais.

Na figura 26 é apresentado como o gerador MNAR lida com dados numéricos. Primeiramente é verificado se a probabilidade é atendida, se não, é retornado o valor do gerador. Se sim, é verificado se o valor gerado (pelo gerador encadeado) está no intervalo dado pelo usuário, se sim, o dado é faltante. Para dados categóricos (ver figura 27) o gerador MNAR utiliza apenas o primeiroPadrao e, nesse caso, recebe uma lista de categorias. Depois de verificar a probabilidade, é verificado se a categoria gerada está na lista de categorias dada, se sim, o dado é faltante, se não é retornado o valor vindo do gerador encadeado. O gerador MNAR para dados temporais (ver figura 28) funciona de forma similar para com dados numéricos 26. Pois também utiliza do intervalo, no caso temporal, para determinar os dados faltantes. O gerador *Noise* (ver figura 29) é um dos principais para geração de dados discrepantes.

Ele utiliza também a probabilidade para ser a taxa de dados discrepantes e de um outro gerador para adicionar ruídos na dimensão. O nível de discrepância é dado pelo atributo intensidade, o qual multiplica o valor gerado.

Figura 25. Pseudocódigo sobre os atributos da classe MNAR do Blocks.

```

classe MNAR estende Acessorio{
    construtor(valorDado, probabilidadeDada, primeiroPadraoDado, segundoPadraoDado, mascaraDada){
        superclasse("MNAR");
        valor = valorDado ou "Miss";
        probabilidade = probabilidadeDada ou 1;
        tipoColuna = tipoColunaGeradorEncadeado ou tipoColunaGeradorSuperclasse;
        primeiroPadrao = primeiroPadraoDado ou "";
        segundoPadraoDado = segundoPadraoDado ou "";
        mascara = mascaraDada ou "HH:mm:ss";
    }
    gerar(){
        valorGerado = superclasse.gerar(0);
        se(tipoColuna == "numerica") {...}
        se(tipoColuna == "categorica") {...}
        se(tipoColuna == "temporal") {...}
    }
}

```

Fonte: O autor do trabalho.

Figura 26. Pseudocódigo sobre a classe MNAR do Blocks para dados numéricos.

```

classe MNAR estende Acessorio{
    ...
    gerar(){
        se(tipoColuna == "numerica") {
            se(valorAleatorio() > probabilidade)
                se(valorGerado > primeiroPadrao e valorGerado < segundoPadrao)
                    retorna ultimoValorGerado = valor;
                retorna ultimoValorGerado = valorGerado;
        }
    }
}

```

Fonte: O autor do trabalho.

Já o gerador *Constant Noise* (ver figura 30) funciona de forma semelhante ao *Noise*, mas não possui o atributo intensidade. O seu método consiste em adicionar um ruído fixo em uma determinada amostra (definida pela probabilidade) dos dados.

Figura 27. Pseudocódigo sobre a classe MNAR do Blocks para dados categóricos.

```

classe MNAR estende Acessorio{
    (...)

    gerar(){
        se(tipoColuna == "categorica") {
            se(valorAleatorio() > probabilidade)
                se(primeiroPadrao.inclue(valorGerado))
                    retorna ultimoValorGerado = valor;
                retorna ultimoValorGerado = valorGerado;
            }
        }
    }
}

```

Fonte: O autor do trabalho.

Figura 28. Pseudocódigo sobre a classe MNAR do Blocks para dados temporais.

```

classe MNAR estende Acessorio{
    (...)

    gerar(){
        se(tipoColuna == "temporal") {
            se(valorAleatorio() > probabilidade)
                se(valorGerado > primeiroPadrao e valorGerado < segundoPadrao)
                    retorna ultimoValorGerado = valor;
                retorna ultimoValorGerado = valorGerado;
            }
        }
    }
}

```

Fonte: O autor do trabalho.

Figura 29. Pseudocódigo sobre a classe do Blocks.

```

classe RandomNoiseGenerator estende Acessorio{
    construtor(probabilidadeDada, intensidadeDada, geradorDado){
        superclasse("Noise Generator");
        gerador2 = geradorDado ou geradorPadrao;
        probabilidade = probabilidadeDada ou 0,3;
        tipogerador = geradorDado.nome;
        intensidade = intensidadeDada ou 1;
    }
    gerar(){
        se(valorAleatorio() > probabilidade) {
            retorna ultimoValorGerado = superclasse.gerar(gerador2.gerar())*intensidade;
        } se não {
            retorna ultimoValorGerado = superclasse.gerar(0);
        }
    }
}

```

Fonte: O autor do trabalho.

Figura 30. Pseudocódigo sobre a classe do Blocks.

```

classe RandomConstantNoiseGenerator estende Acessorio{
    construtor(probabilidadeDada, intensidadeDada, geradorDado){
        superclasse("MCAR");
        valor = valorDado ou 1;
        probabilidade = probabilidadeDada ou 0,3;
    }
    gerar(){
        se(valorAleatorio() > probabilidade) {
            retorna ultimoValorGerado = superclasse.gerar(0)+valor;
        } se não {
            retorna ultimoValorGerado = superclasse.gerar(0);
        }
    }
}

```

Fonte: O autor do trabalho.

4 Sistema Blocks

Neste capítulo o sistema Blocks será apresentado em detalhes, como seu fluxo de funcionamento, funcionalidades, detalhes sobre a interface do usuário entre outros. De modo geral, o sistema é chamado de Blocks e visa ser um gerador de dados sintéticos baseado na combinação de modelos de dados. Assim, o usuário pode manipular um ou mais modelos e cada modelo pode conter N dimensões, que por sua vez podem conter M geradores de dados encadeados.

Os geradores podem gerar dados numéricos, categóricos, temporais, entre outros. O resultado de um gerador pode servir de entrada para outro através de operadores. Os operadores podem ou não aplicar uma operação matemática (soma, subtração, divisão ou multiplicação) ao resultado do gerador anterior - a leitura de anterior e posterior é da esquerda para a direita, respectivamente. Junto com os operadores, também há outras propriedades que variam de acordo com o gerador.

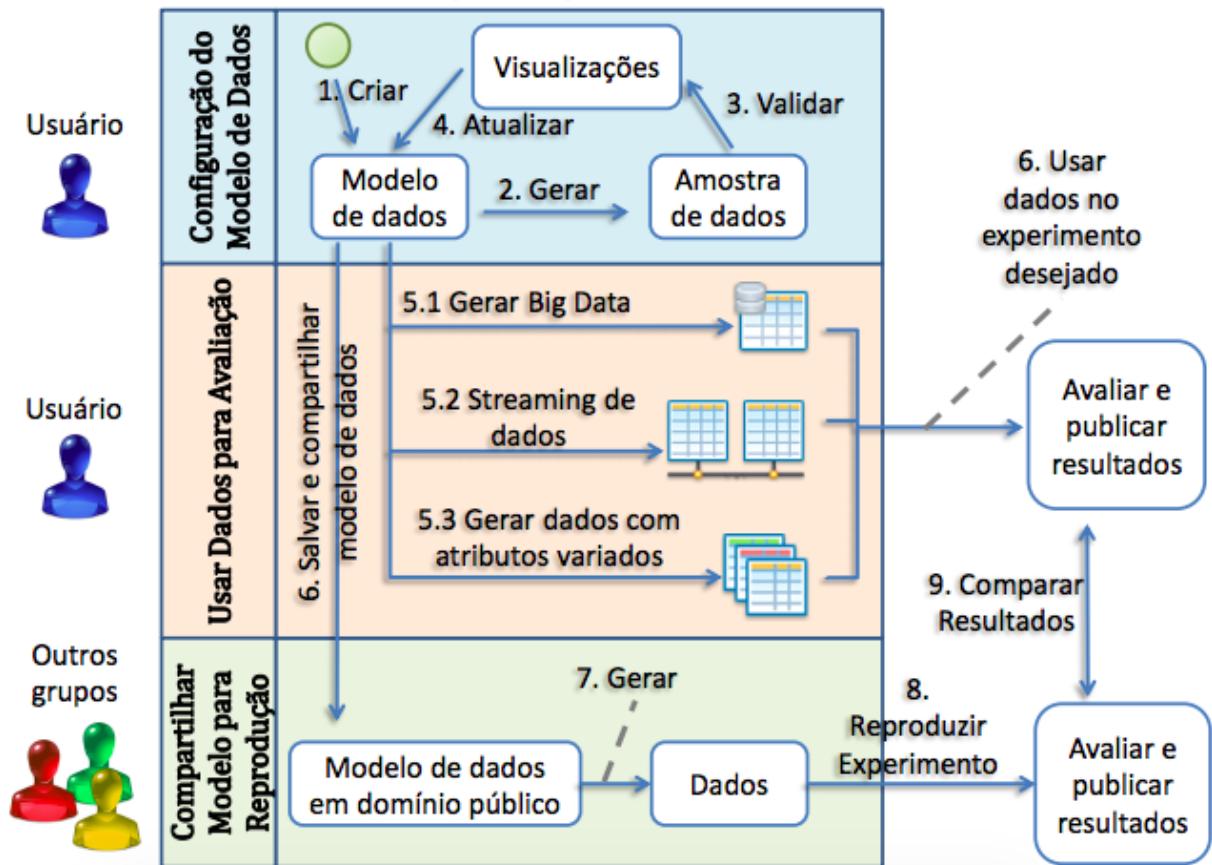
Também é disponibilizado um pré-visualizador de dados com o gráfico coordenadas paralelas, o qual é colorido e interativo, sendo seu volume de dados independente do volume a ser gerado. Além do *preview*, há uma integralização com um visualizador de dados mais elaborado e com mais opções de visualização, o *VisApplication*.

O Blocks permite gerar os dados em arquivos JSON, CSV, TSV ou por de requisições HTTP do tipo GET (*Web Service*). Vale ressaltar que é possível configurar a quantidade de dados gerados, pré-visualizados, formato dos dados gerados, se contém o cabeçalho dos dados no arquivo final e as propriedades do *Web Service*.

Na figura 31 é possível visualizar como foi pensada a utilização da aplicação. Na aba "Configuração do Modelo de Dados"encontra-se como o usuário pode configurar o modelo. O usuário define as dimensões e os seus geradores, assim como o comportamento dos dados pode ser validado pelo *preview*. Caso seja necessário, o modelo pode ser atualizado a qualquer momento.

Na aba "Usar Dados para Avaliação"são demonstrados os caminhos para geração de dados. É possível a geração de dados por *Big Data*, isto é, um grande conjunto de dados multidimensionais, apenas por meio de arquivos. Também é possível a geração de dados por *Streaming*, a qual proporciona mais controle ao usuário sobre o processo de geração e também é disponibilizado o *Web Service*. O caminho 5.3, visto na figura 31 demonstra uma forma iterativa de geração de arquivos de dados, com mudanças programadas de atributos. Esses dados podem ser usados em experimentos, testes e afins, cujos resultados podem ser publicados.

Figura 31. Fluxograma de utilização do Blocks.



Fonte: adaptado de (Brito Y., 2019).

E na aba "Compartilhar Modelo para Reprodução" é apresentado que os processos anteriores podem ser replicados a partir do mesmo modelo de dados e o resultado pode ser comparado, facilitando o consumo de dados por pesquisadores.

4.1 Tipos de Geradores de Dados

Na tabela 4 são apresentados todos os geradores do Blocks até o momento desde trabalho. São apresentados seus nomes, suas categorias, o tipo de valores retornados, os tipos de parâmetros, se possui correlação - isto é, se é baseado no valor de outra dimensão - se são dependentes - ou seja, se precisa de um gerador encadeado para funcionar corretamente.

4.1.1 Sequencial

Os geradores da categoria Sequencial geram valores encadeados dado um padrão. É possível gerar o próprio padrão a partir do gerador *Custom Sequence*, o qual você determina um valor Inicial (*Begin*) e o valor Intervalar (*Step*), isto é, o qual vai ser incrementado ou decrementado dado uma Sentença (*sentence*) customizada.

Tabela 4. Propriedade dos geradores do Blocks. N=Número; C=Categoria; T=Tempo.

Nome	Categoria	T. dos valores	Correlação	Dependente
Constant	Sequencial	N	Não	Não
Counter	Sequencial	N	Não	Não
Fixed Time	Sequencial	T	Não	Não
Sinusoidal Sequence	Sequencial	N	Não	Não
Custom Sequence	Sequencial	N	Não	Não
Poisson Time	Aleatório	T	Não	Não
Uniform	Aleatório	N	Não	Não
Gaussian	Aleatório	N	Não	Não
Poisson	Aleatório	N	Não	Não
Bernoulli	Aleatório	N	Não	Não
Cauchy	Aleatório	N	Não	Não
Weighted Categorical	Aleatório	C	Não	Não
Categorical	Aleatório	C	Não	Não
Categorical Quantity	Aleatório	C	Não	Não
Linear	Função	N	Sim	Não
Quadratic	Função	N	Sim	Não
Polynomial	Função	N	Sim	Não
Exponential	Função	N	Sim	Não
Logarithm	Função	N	Sim	Não
Sinusoidal	Função	N	Sim	Não
Categorical	Função	C	Sim	Não
Piecewise	Função	N	Sim	Não
TimeLaps	Função	T	Sim	Não
MCAR	Acessório	N, C ou T	Não	Sim
MNAR	Acessório	N, C ou T	Não	Sim
Noise	Acessório	N, C ou T	Não	Sim
Constant Noise	Acessório	N, C ou T	Não	Sim
Range Filter	Acessório	N, C ou T	Não	Sim
Linear Scale	Acessório	N, C ou T	Não	Sim
No Repeat	Acessório	N, C ou T	Não	Sim
MinMax	Acessório	N, C ou T	Não	Sim
Low-Pass Filter	Acessório	N, C ou T	Não	Sim
Get Extra Value	Acessório	N, C ou T	Não	Sim
CubicBezier	Geométrico	N	Não	Não
Path2D Stroke	Geométrico	N	Não	Não
Path2D Fill	Geométrico	N	Não	Não

Fonte: O autor do trabalho.

Contudo, já são predefinidos alguns geradores como o *Constant*, o qual define um valor único de geração; o *Counter*, funciona como um contador, no qual é definido o valor Inicial e o Intervalar; o *Fixed Time Generator* gera um intervalo de tempo, no qual pode ser definido o valor inicial (*init*), Intervalar e a máscara (*mask*), isto é, como o tempo deve ser formatado; o *Sinusoidal Sequence* gera de acordo com a função senoidal, que, além

do valor Inicial e Intervalar, há o 'a' de Amplitude, 'b' de frequência ângular e 'c' para representar a fase da onda.

4.1.2 Aleatório

A categoria aleatória de dados contém um grande número de geradores, pois existe uma diversidade de distribuições de probabilidade comparada às outras categorias.

Esta categoria conta com geradores uniformes, isto é, a distribuição dos dados é equalizada; Também há um gerador de dados de tempo, parecido com o *Fixed Time Generator* com a diferença que o comportamento é definido pela fórmula de *Poisson* e que há mais duas configurações: unidade de tempo - a qual pode ser desde milissegundos a anos - e o lambda, advindo da fórmula. Há uma distribuição de *poisson* também, apenas com o lambda; É disponibilizado geradores de fórmulas clássicas com a normal (*Gaussian*), *Bernoulli*, e *Cauchy* com seus devidos parâmetros.

Além de números, também é possível gerar dados categóricos (*Categorical*), dadas as palavras - também chamado de categorias - inicialmente. Similarmente há o *Weighted Categorical* que possui valores de probabilidade para cada palavra, e já o *Categorical Quantity*, em vez de probabilidade, define quantas vezes cada palavra deve aparecer.

4.1.3 Função

A categoria funções (*Function*) serve para gerar dados de acordo com outra dimensão chamado de *input*, isto é, facilita a correlação entre dimensões. Para dados numéricos, disponibiliza-se as função de primeiro grau (*Linear Function*), segundo grau (*Quadratic Function*), exponencial (*Exponential Function*), logarítmica (*Logarithm Function*) e a *Piecewise Function*, cuja função é definida por subfunções, e no caso, é possível definir o gerador desejado até um determinado valor chamado de *Intervals* e depois pode-se escolher outro gerador.

Para dados categóricos, há a função categórica (*Categorical Function*) e a *TimeLaps Function* a qual funciona de forma semelhante ao gerador *Piecewise Function*, só que utiliza uma quantidade de tempo como limiar - imagine uma corrida de fórmula 1 e cada vez que os carros passam pela linha de chegada eles completam uma volta. Esta volta é o limiar também chamado de *Laps* - e para o *input*, somente geradores de tempo.

4.1.4 Acessório

Os geradores da categoria Acessórios (*Acessory*) foram pensados especialmente para serem concatenados com outros geradores, com o fim de incrementá-los. Entre os geradores acessórios, pode-se citar o *Missing Value*, o qual foi subdividido em 2 geradores: o *MCAR* e *MNAR*.

O MCAR usa a probabilidade para definir se um dado será faltante; O MNAR trabalha de forma diferente para cada tipo de dado. Os tipos de dado são os Numéricos, os Categóricos e os Temporais.

O MAR é gerado a partir do encadeamento de geradores. Isto é, por conta do MAR levar em consideração a correlação a uma dimensão, este pode utilizar um gerador da categoria função em conjunto do MCAR ou MNAR.

Para os Numéricos e os Temporais, é definido um intervalo no qual os dados vão ser faltantes. Para os dados Categóricos, é definida uma lista de categorias na qual todas essas categorias faltarão na geração de dados.

Os geradores MNAR e o MCAR foram desenvolvidos ou modificados no sistema ao longo deste trabalho. Entretanto a implementação foi baseada nos artigos (TWALA, 2009) (XIA et al., 2017) para implementação do MNAR, os trabalhos de (RIEGER; HOTHORN; STROBL, 2010) (XIA et al., 2017) são semelhantes ao gerador MCAR, pois utilizam uma probabilidade para definir aleatoriamente um dado faltante. Os artigos de implementação também podem ser encontrados no artigo (SANTOS et al., 2019) e segundo o mesmo, as implementações são consideradas univariadas, isto é, apenas uma dimensão é afetada.

O *Noise Generator* que adiciona dados fora do padrão, conhecido como ruído, com uma determinada probabilidade, intensidade - o que ajuda a criar o nível de discrepância - e a partir de três distribuições: uniforme, normal e de Poisson. O *Constant Noise Generator* também adiciona ruídos, mas só que é um valor específico com determinada probabilidade de ser adicionado; o *Ranger Filter* permite retirar do conjunto de dados os valores que estão entre os valores de início (*Begin*) e fim (*End*); o *Linear Scale* permite que os determinados (selecionados através do *MinIn* e *MaxIn*) dados sejam escalados através do *minOut* e *MaxOut*.

O *No Repeat* retira dados repetidos do conjunto; o *MinMax* define quais valores serão os maiores e menores de acordo com os parâmetros dados; o *Low-Pass Filter* realiza um filtro passa-baixa nos dados. Ou seja, o valor sucessor é uma media ponderada (valor recebido pelo parâmetro *Smooth*) do valor anterior com o valor gerado; o *Get Extra Value* pega os retornos extras dos geradores que retornam mais que um valor.

4.1.5 Geométrico

Os geradores Geométricos (*Geometric*) permitem que seja gerado dados a partir de formas geométricas. Para isso são disponibilizados três geradores. O primeiro deles (*Path2D Stroke*) gera dados bidimensionais presentes nas arestas de um polígono inserido pelo usuário. O segundo (*Path2D Fill*) gera os dados pertencentes do polígono inserido pelo usuário. Quanto ao terceiro (*CubicBezier*), este gera dados para desenho de uma curva bezier cúbica a partir de seus pontos.

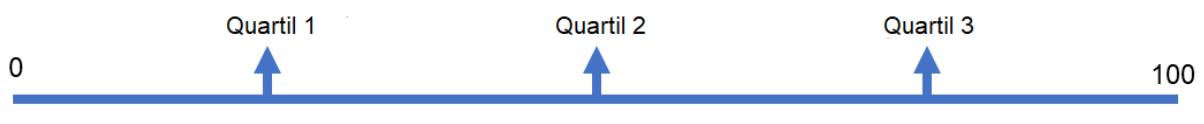
4.1.6 Baseado em dados reais

Para esta categoria, existe um gerador, chamado como *Real Data Wrapper*. Ele é criado automaticamente quando o usuário importa um conjunto de dados reais através de um CSV. Este gerador recebe tantos valores categóricos como numéricos e essa informação pode ser decidida automaticamente pelo gerador ou ser forçada pelo usuário. É possível gerar uma quantidade superior de dados em relação ao conjunto de dados real, para isso é feito um tratamento de inputação de dados. Esse tratamento é feito por meio de funções de geração, chamadas de *GenType*.

Essas funções podem ser do tipo *Standart*, que utiliza os dados do início ao fim de forma cíclica até chegar ao número desejado de registros. Também pode ser do tipo *Reverse* que ao contrário do *Standart*, utiliza os dados do final ao início. É disponibilizado o modo aleatório (*Random*) e outras variações.

A primeira variação do modo aleatório é o *QuartileRandom*, que divide o conjunto de dados em três marcos e a probabilidade de se pegar um dado daquele quartil é proporcional ao tamanho do marco. A leitura dos Marcos pode ser visualizada na figura 32. Então, se o valor do espaço entre 0 e quartil 1 for 100, todos os dados gerados serão do primeiro 1/4 do conjunto de dados. A segunda variação é o *AverageRandom* que utiliza o valor da média e da variancia - [Média - Variancia, Média + Variancia] de um conjunto de dados numérico ou utiliza os N valores categóricos mais frequentes com distribuição uniforme.

Figura 32. Ilustrando a leitura dos marcos dos quartis. O tamanho do espaço entre os quartis ou entre 0 ou o 100 é o valor da probabilidade de um número ser desse espaço.



Fonte: O autor do trabalho.

4.2 Modos de Geração de Dados

Nessa seção é demonstrado como é feita a concretização do comportamento dos geradores, isto é, os dados propriamente ditos. Gerar os dados é tão importante quanto gerar o modelo, visto que os dados gerados podem ser utilizados para testes de aplicações, por exemplo.

Os dados são gerados por *Web Service* e em dois tipos de memória: a primária e a secundária. A memória primária serve de base para escrever na memória secundária e também alimenta as visualizações de dados.

4.2.1 *Streaming Data*

Para otimizar a geração de grande volumes de dados, foi utilizado o conceito de *Streaming Data*, isto é, gerar o volume de dados aos poucos - em blocos - para que não haja estouro de memória primária. Também foi utilizado esse processo assincronamente, para que a interface de usuário não seja bloqueada durante o processo e permitir que o progresso da geração seja acompanhado.

O usuário escolhe a quantidade de dados a ser gerada e a aplicação define automaticamente a quantidade de blocos. Cada bloco é fixado em até 10.000 instâncias. Esse valor foi definido de forma aleatória. Mais testes serão feitos com o intuito otimizar o número de instâncias de cada bloco. O bloco é processado e armazenado temporariamente na memória primária até que ele esteja completo. Então, o bloco é escrito na memória secundária.

4.2.2 *Web Service*

Quanto ao *Web Service*, este foi pensado para facilitar o teste de aplicação. Cada modelo é independente, isto é, podem ser habilitados somente os modelos desejados para distribuição. E além da configuração por dentro do *software*, também é possível criar configurações temporárias para cada requisição, sem alterar as configurações do modelo.

Os parâmetros disponíveis para configuração temporária pela URI são: o nome do modelo, o formato dos dados e a quantidade de registro. É disponibilizado um ícone de aviso ao usuário quando um modelo está distribuindo dados via *Web Service* na aba do modelo. Um exemplo de URI para fazer requisição HTTP do tipo GET é: (<http://localhost:8000/?modelid=MODEL_r6w2ffk3.mva&nsmple=100&format=csv>), nos quais "modelid" é o *ID* do modelo, "nsmple" é a quantidade de instâncias desejada e "format" é o formato do arquivo desejado.

4.3 Modos para Visualização de Dados

O sistema permite modelar os dados, gerá-los e também visualizá-los. Nessa seção são mostradas as duas formas de visualizar dados no Blocks. A primeira fica acessível na tela inicial do gerador chamado de *Preview*. A outra é um programa a parte que é integrado ao Blocks, isto é, há o compartilhamento do modelo de dados.

4.3.1 *Preview*

O pré-visualizador de dados foi criado pensando em oferecer uma visualização rápida e abrangente do modelo de dados. Para isso, foi escolhido o gráfico Coordenadas Paralelas, por conta de sua característica de visualização prática de dados multidimensionais. Também

foram adicionadas algumas características extras como diferenciação por cores (mapa de calor para dados numéricos e cor única para dados categóricos); filtro de dimensão, para que fosse visualizado apenas o que for necessário; escolha de dimensão como referencial, isto é, a partir da dimensão escolhida, verificar como os dados se comportam nas outras dimensões. Isso pode ser ativado tanto clicando sobre o nome da dimensão, quanto através do *ComboBox* acima do *preview*; também é possível recarregá-lo e desativá-lo, para travamentos quando for trabalhar com *Big Data*, por exemplo.

4.3.2 Módulo de Visualização Externo e Integrado

O módulo chamado VisApplication é um conjunto de técnicas de visualização reutilizáveis. Ela pode chamada por dentro do Blocks e já pode utilizar os dados do modelo atual. Dentre as visualizações disponíveis pode-se citar as Coordenadas Paralelas, *Scatter Plot*, *Treemap*, *Sunburst*, *Bar Chart* entre outros. Como diferencial, algumas funcionalidades são adicionadas como detalhe sob demanda, *zoom*, marcação de dados (*Highlight*), múltiplas visualizações simultâneas, entre outras.

4.4 Estrutura da Interface Gráfica do Blocks

O sistema possui uma interface gráfica para *Desktop* e segue um modelo de estrutura que possui uma janela principal, mais algumas janelas auxiliares. Isso significa que há uma tela com as principais informações (ver figura 33), e outras informações mais específicas são visualizadas através de *tabs*, *modals*, *alerts* e correlacionados.

Na figura 33 pode-se encontrar os principais elementos para utilizar o Blocks. Nesta figura são encontradas 22 marcações e serão descritas na lista a seguir.

1. São definidas as propriedades da dimensão, é possível visualizar informações e alterar o título;
2. Nome atual do modelo;
3. Ícone para mostrar que este modelo está servindo dados via (*Web Service*);
4. Ícone para mostrar alterações não salvas no modelo;
5. Botão para fechar o modelo;
6. Espaço para as dimensões do modelo;
7. Botão para criar um novo modelo;
8. Gerador;

9. Botão responsável por criar e aumentar o encadeamento de geradores;
10. Botão para excluir um gerador da cadeia;
11. filtro de dimensões - quando este ícone estiver com uma reta na diagonal sobre o filtro, significa que a dimensão não será incluída na geração e nem visualização dos dados;
12. Botão para excluir a dimensão selecionada;
13. Configurações atuais do gerador - tipo de gerador, que apresenta uma lista de categorias que, por sua vez, cada uma apresenta uma lista de geradores e também as propriedades do gerador selecionado;
14. Botões para excluir ou alterar o posicionamento da dimensão selecionada;
15. Botões para copiar geradores;
16. Botão para esconder o pré-visualizador de dados;
17. Botão para recarregar o pré-visualizador de dados;
18. Ao clicar no título, a dimensão é selecionada como origem para as cores;
19. *ComboBox* para selecionar dimensão para dar origem às cores - útil para saber como os dados da dimensão selecionada se comporta nas outras dimensões.
20. Pré-visualizador de dados utilizando o Coordenadas Paralelas;
21. Botões para gerar os dados a partir do modelo atual e para manipular as algumas configurações do modelo atual;
22. Agrupa botões para abrir, criar, ou salvar um modelo; desfazer ou refazer uma alteração ou abrir o *VisApplication*.

Ao clicar com o botão direito do *mouse* no título do modelo (M2) aparece um menu, como visto na figura 35. Esse *context menu* permite renomear ou deletar o modelo, exportá-lo como arquivo .DOT e também manipular algumas informações para o Web Service Como copiar para a área de transferência o ID do modelo, uma URI padrão (*localhost*); ativar ou desativar o modelo pra *Web Service*, bem como abrir a URI em um software padrão do usuário.

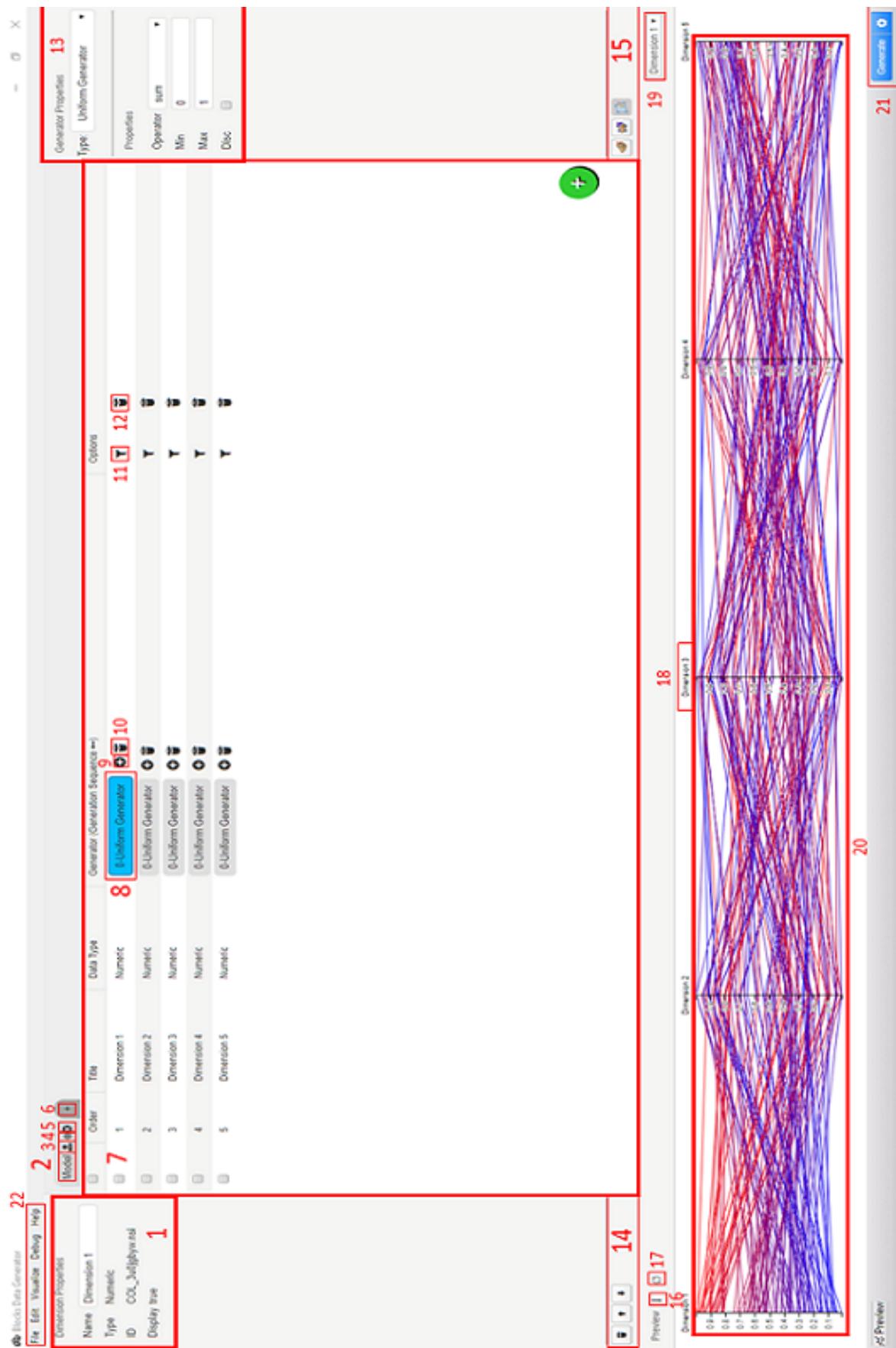
Ao lado do botão (*Generate*) para iniciar a geração é encontrado outro botão com o ícone de engrenagem (M21). Na figura 34, no lado esquerdo, encontra-se 3 seções. A primeira é dedicada para configurações gerais do modelo, como quantidade de dados para ser gerados, mostrado no *preview* ou formato dos dados. A segunda seção (*Parameter*

Iterator) foi desenvolvida para quem precisar criar mais de um arquivo variando apenas alguns parâmetros, de forma iterativa. A terceira seção é especial para o *Web Service* no qual, liga ou desliga o servidor ou troca a porta padrão para servir os dados.

4.4.1 Mensagens para o usuário

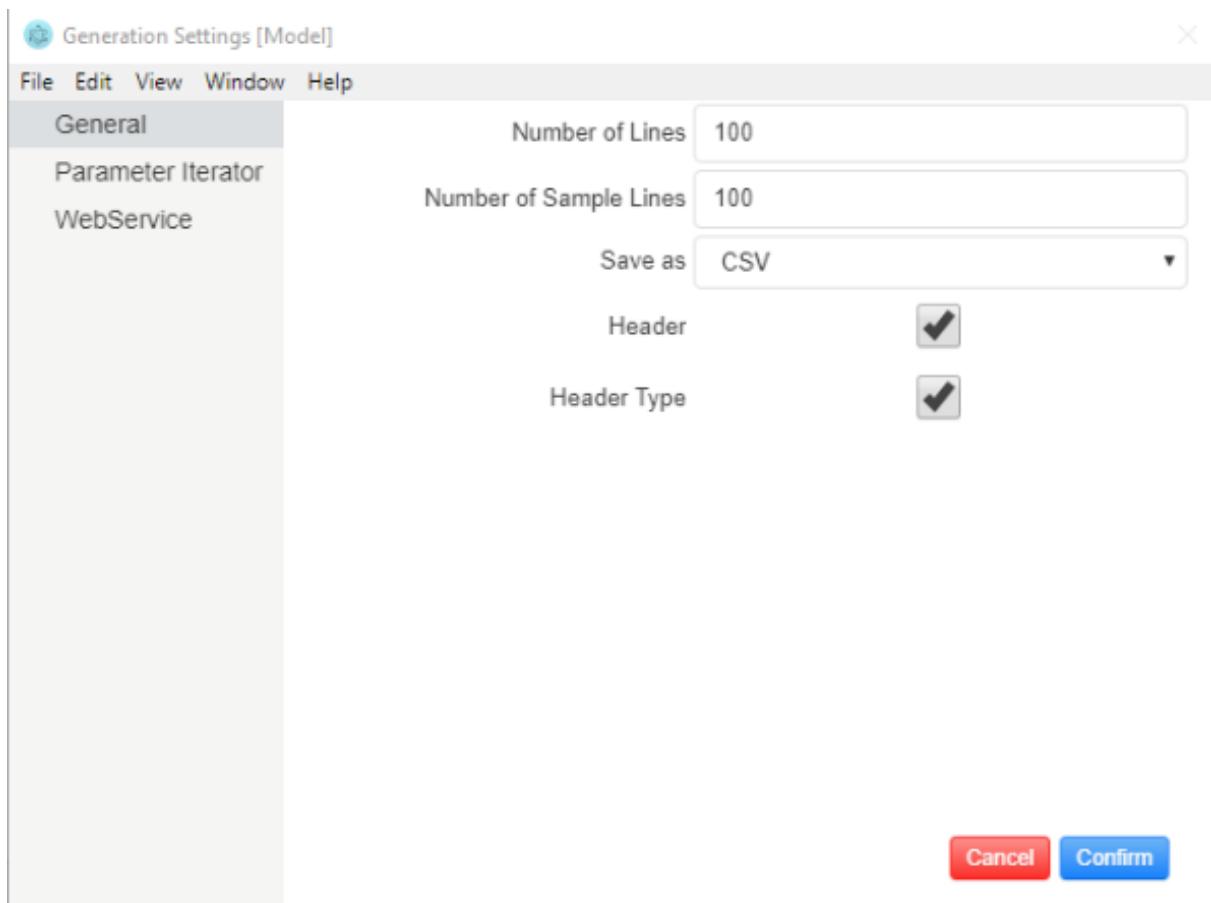
O sistema precisa avisar o usuário de falhas, perguntar sobre preferências e afins. Para isso, o Blocks utiliza-se de *Dialogs* (ver figura 36) para receber um caminho para salvar ou abrir um arquivo. Há um espaço dedicado na tela principal para mensagens advindas de um processo de geração de dados. Mensagem para preparação, progresso, finalização ou falha na geração de dados pode ser acompanhado pelo *Footer Display*. (ver figura 37). Para outros avisos mais genéricos como erros ou tarefas bem sucedidas, bem como avisos mais detalhados há o *Modal* (ver figura 38). Também há Atalhos do Teclado da aplicação Blocks para demonstrar os atalhos do teclado, para melhorar a eficiência de usuários que preferem esse recurso.

Figura 33. Conhecendo os elementos da tela principal do Blocks, na sua versão para Windows.



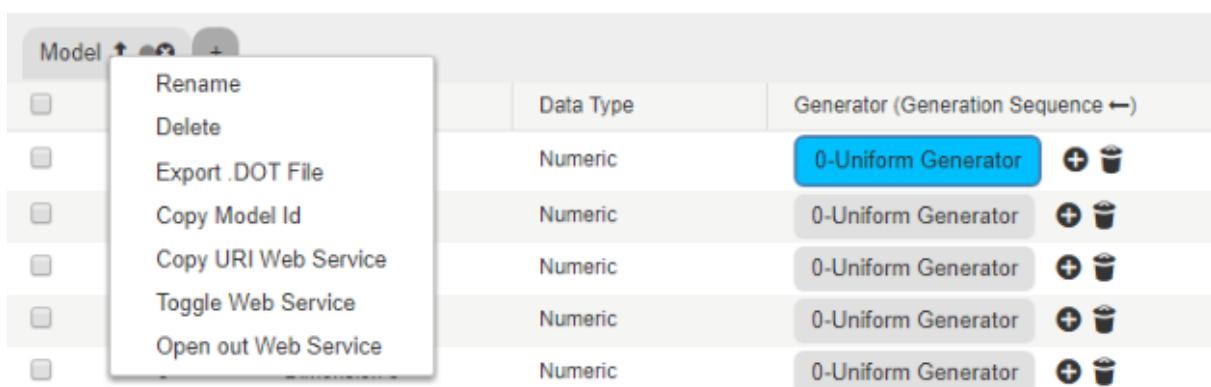
Fonte: O autor do trabalho.

Figura 34. Conhecendo os elementos da tela de configurações para geração de dados.



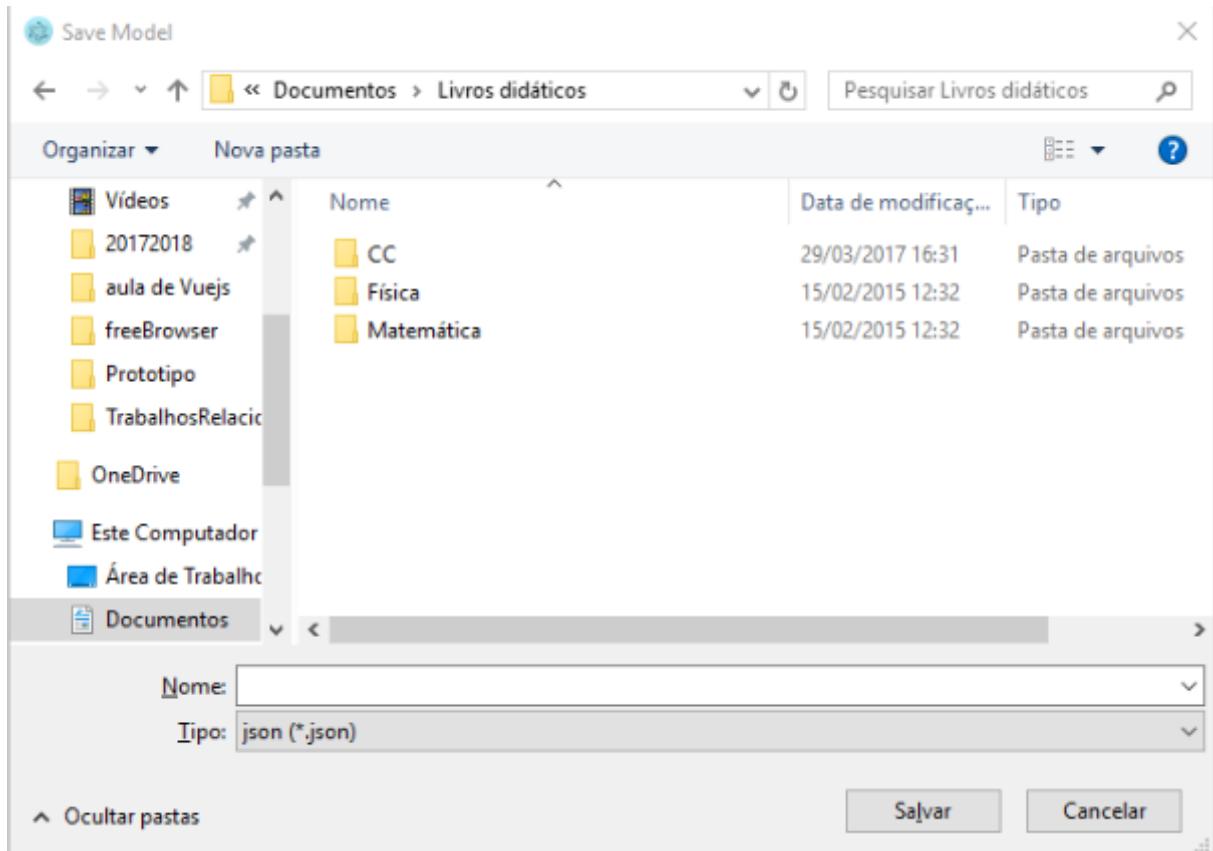
Fonte: O autor do trabalho.

Figura 35. Conhecendo os elementos da *context menu* na aba do modelo.



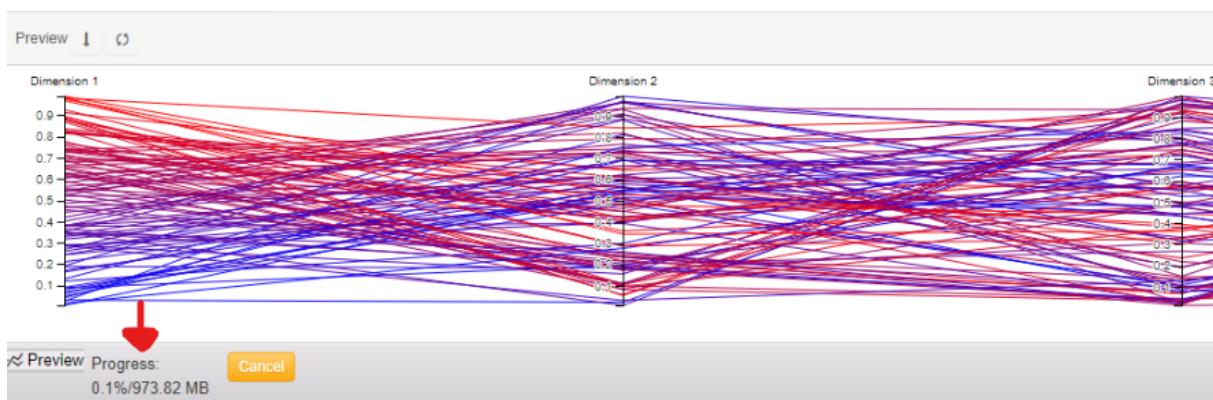
Fonte: O autor do trabalho.

Figura 36. Conhecendo os elementos da tela de configurações para geração de dados.



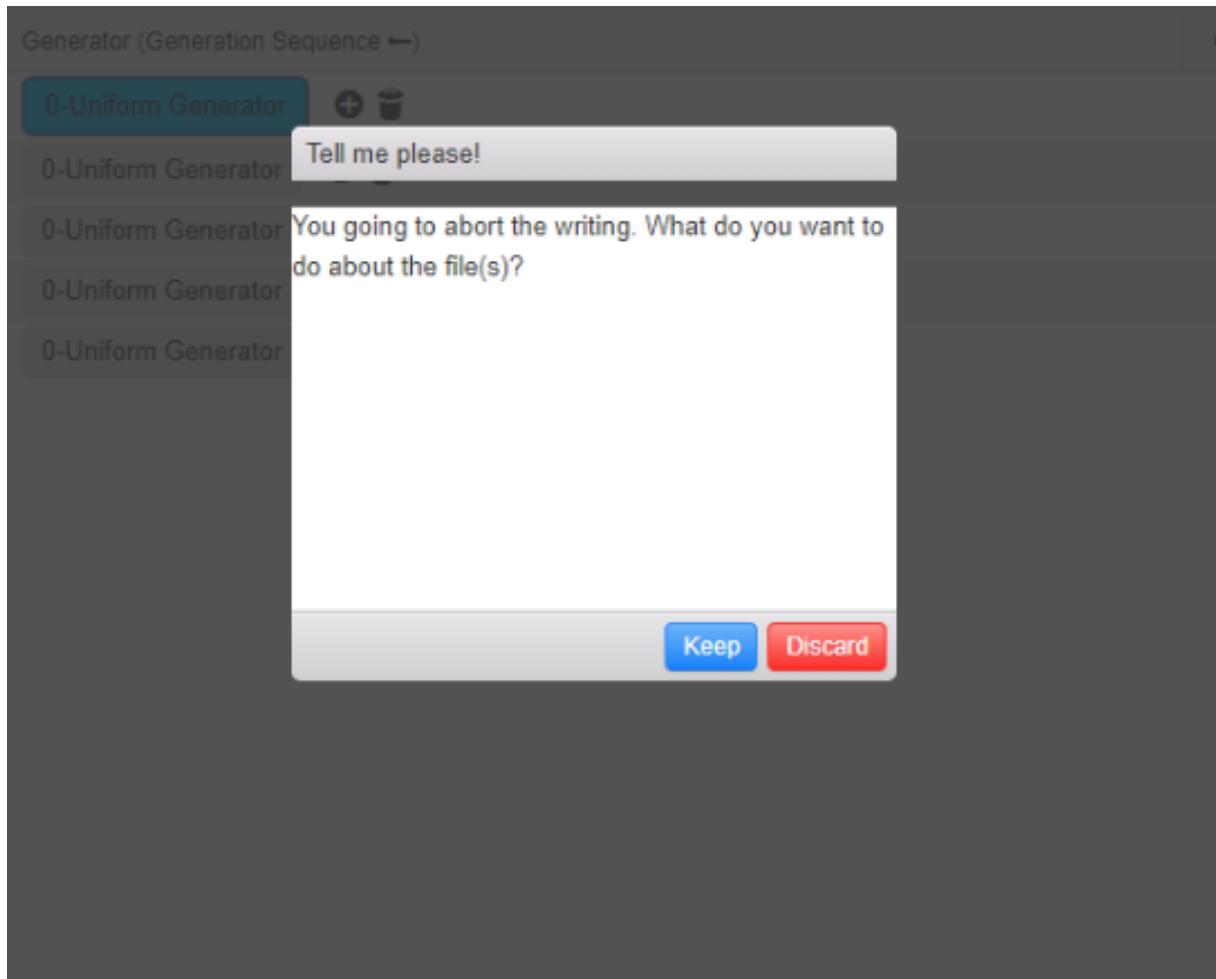
Fonte: O autor do trabalho.

Figura 37. Conhecendo os elementos da tela de configurações para geração de dados.



Fonte: O autor do trabalho.

Figura 38. Conhecendo os elementos da tela de configurações para geração de dados.



Fonte: O autor do trabalho.

5 Validação Visual

Este capítulo é dedicado a apresentar como podem ser modelados os dados discrepantes e faltantes no Blocks. Também, serão apresentados os dados gerados e técnicas de visualização a partir das possibilidades do sistema.

5.1 Modelagem dos Dados

Foram criadas cinco bases de dados de contextos genéricos, simulando abstratamente contextos reais. Para criação dessas bases com 500 instâncias foram utilizados os geradores *Noise Generator* e o *Constant Noise Generator* para gerar dados discrepantes. E para criação de dados faltantes foram utilizados os geradores *MCAR*, *MAR* e *MNAR*. Também foram utilizados outros geradores para auxiliar no comportamento das dimensões como os *Linear Function*, *Linear Scale*, *MinMax* e *Piecewise Function*.

A primeira base é sobre uma avaliação de carros, onde há os seguintes atributos: categoria do carro, ano, marca, preço, notas dos críticos e do público. Como pode ser visto na figura 39, Categoria e Marca são um conjunto de palavras genéricas e uniformemente distribuídas, assim como o Ano do carro, mas com números inteiros.

O preço já tem um tratamento de discrepância - os carros podem ficar 30% mais caros ou mais baratos - e uma correlação com o ano - pois carros mais velhos tendem a ficar mais baratos. A nota dos críticos é gerada de forma uniforme e imparcial, para ser um referencial técnico.

Contudo, houve uma configuração de MNAR na modelagem, na qual as notas abaixo de 2 estão faltando, propositalmente. A nota do público tem dados faltantes com relação ao simples fato de uma pessoa preferir não opinar sobre o assunto e correlação com o preço - inversamente proporcional - e nota dos críticos - diretamente proporcional. Quando a nota dos críticos é faltante, o público leva em consideração apenas o preço.

Figura 39. Base sintética de avaliação de Carros.

Title	Data Type	Generator (Generation Sequence ←)
Categoria	Categorical	0-Categorical  
Ano	Numeric	0-Uniform Generator  
Marca	Categorical	0-Categorical  
Preço	Numeric	0-Noise Generator  1-Noise Generator  2-Uniform Generator  3-Linear Scale  4-Linear Function  
Nota dos críticos	Numeric	0-MNAR  1-Uniform Generator  
Nota do público	Numeric	0-Piecewise Function   <= Miss  1-MCAR  2-Linear Scale  3-Linear Function   > Miss  1-MCAR  2-Linear Function  3-Linear Scale  4-Linear Function  

Fonte: O autor do trabalho.

A segunda base (ver figura 40) diz respeito a uma avaliação de Redes Sociais. As dimensões presentes são o nome que é uma categoria genérica; idade que varia de 18 a 65 anos - mas possui dados faltantes acima de quarenta anos; postagens que possui uma discrepância fraca - um ruído - para valores altos, visando simular pessoas que postam vários vídeos diariamente. Também há uma discrepância para valores baixos, simulando aquelas pessoas que postam esporadicamente; as dimensões Postagens, Seguidores e Curtidas possuem um limite de valores, para manter os dados aproximados da realidade.

Figura 40. Base sintética de avaliação de Redes Sociais.

Title	Data Type	Generator (Generation Sequence →)							
Nome	Categorical	0-Categorical	+ ↪						
Idade	Numeric	0-MNAR	1-Uniform Generator	+ ↪					
Postagens	Numeric	0-MinMax	1-Noise Generator	2-Noise Generator	3-Uniform Generator	4-Gaussian Generator	+ ↪		
Seguidores	Numeric	0-MinMax	1-Noise Generator	2-Noise Generator	3-MinMax	4-Uniform Generator	5-Gaussian Generator	+ ↪	
Curtidas	Numeric	0-MinMax	1-Noise Generator	2-Noise Generator	3-Linear Function	4-Linear Function	+ ↪		

Fonte: O autor do trabalho.

Mais duas dimensões são disponibilizadas e também complexas. Entre elas estão a dimensão Seguidores que possui uma forte discrepância, para simular grandes famosos, mas são muito raros. E ainda a dimensão Curtidas, que possui correlação de Seguidores - cerca de 60% das curtidas são de seguidores - e postagens - quanto mais postagens, mais curtidas acumuladas. Além da correlação, possui a discrepância fraca e muito forte, para simular boas postagens e postagens virais.

A terceira base como pode ser visto na figura 41 mostra um esquema de convênios médicos. Primeiramente há o profissional e sua especialidade - ambos dados categóricos uniformes; Quanto ao Plano de saúde - é referente aos que atende - e esta dimensão possui um peso, devido à popularidade e/ou acessibilidade dos planos; E o Preço da Consulta varia de acordo com o plano de saúde. Para essa variação foram escolhidos vários geradores da categoria aleatória com parâmetros diferentes, com o intuito de gerar dados discrepantes sem utilizar o gerador *Noise Generator*.

A base que simula uma estrutura de conta bancária (ver figura 42) foi feita com a intenção de mostrar dados hierárquicos. A identificação de uma conta bancária é composta três campos sendo eles: um banco composto por 3 dígitos, uma agência de 4 dígitos e uma conta composta por 8 dígitos. Foram acrescentados umas opções de dados faltantes do tipo MCAR e MNAR.

Há também uma base genérica como vista na figura 43 com o intuito de mostrar os dados faltantes de uma forma mais simples. A primeira dimensão é chamada de dimensão observada, que é a dimensão utilizada para predizer dados faltantes no mecanismo MAR. A segunda dimensão possui dados faltantes a partir do mecanismo MAR. A terceira dimensão possui dados faltantes a partir do mecanismo MCAR. E a quarta dimensão possui dados

Figura 41. Base sintética de avaliação de Convênios Médicos.

	Order	Title	Data Type	Generator (Generation Sequence ←)
	1	Profissional	Categorical	0-MCAR 1-Categorical
	2	Especialidades	Categorical	0-MCAR 1-Categorical
	3	Planos de Saúde	Categorical	0-Weighted Categorical
	4	Preço por Consulta	Numeric	0-Categorical Function A1 0-Gaussian Generator U1 0-Uniform Generator H1 0-Uniform Generator BS1 0-MinMax 1-Cauchy Generator SS2 0-Poisson Generator SYS1 0-Uniform Generator

Fonte: O autor do trabalho.

Figura 42. Base sintética de avaliação de Estrutura de Conta Bancária.

	Order	Title	Data Type	Generator (Generation Sequence ←)
	1	Tipo da conta	Categorical	0-Categorical
	2	Agência	Numeric	0-MCAR 2-Uniform Generator
	3	Banco	Numeric	0-Uniform Generator
	4	Conta	Numeric	0-MNAR 1-No Repeat 2-Uniform Generator

Fonte: O autor do trabalho.

faltantes a partir do mecanismo MNAR.

Figura 43. Base sintética genérica para geração de dados faltantes numéricos.

	Order	Title	Data Type	Generator (Generation Sequence ←)
	1	Observada	Numeric	0-Uniform Generator
	2	MAR	Numeric	0-Piecewise Function <= 50 0-Uniform Generator > 50 0-MCAR 1-Uniform Generator
	3	MCAR	Numeric	0-MCAR 1-Uniform Generator
	4	MNAR	Numeric	0-MNAR 1-Uniform Generator

Fonte: O autor do trabalho.

5.2 Apresentação das Visualizações

Uma vez que o modelo está pronto é preciso visualizar os dados e analisar os seus resultados. O Blocks possui um sistema de visualização integrado (*VisApplication*) e suas visualizações são utilizadas como exemplos. Entretanto, outras aplicações foram utilizadas para visualização de dados como o RAWGraphs (DensityDesign Research Lab, 2019) e o Excel (Microsoft,).

Para a base de Carros foram utilizadas duas técnicas de visualizações de dados, Coordenadas Paralelas e Scatterplot. Na figura 44, a qual possui três gráficos de coordenadas paralelas agrupadas. Os gráficos se diferenciam na origem da coloração dos dados. O gráfico mais em cima tem como origem a dimensão Preço, o do meio tem Nota dos críticos e o mais embaixo tem a Nota do público.

Nessa visualização é possível perceber a grande quantidade de carros baratos, por conta da predominância de linhas azuis (marcação alaranjada) no primeiro gráfico e há uma forte correlação com o ano (marcação cinza) e possuem, em geral, uma alta nota dos críticos e do público (Marcação rosa). Pode-se perceber um vácuo nas notas 1 e 2 (marcação verde) na dimensão das Notas dos críticos, o que é resultado do gerador MNAR. Também é observável a correlação (marcação rosa) entre as notas dos críticos e nota do público.

Na figura 45 mais detalhes podem ser percebidos por conta da dispersão. Primeiramente, os valores abaixo de 0 (marcação verde) são os dados faltantes. Esse padrão é adotado não só nesse modelo de dados.

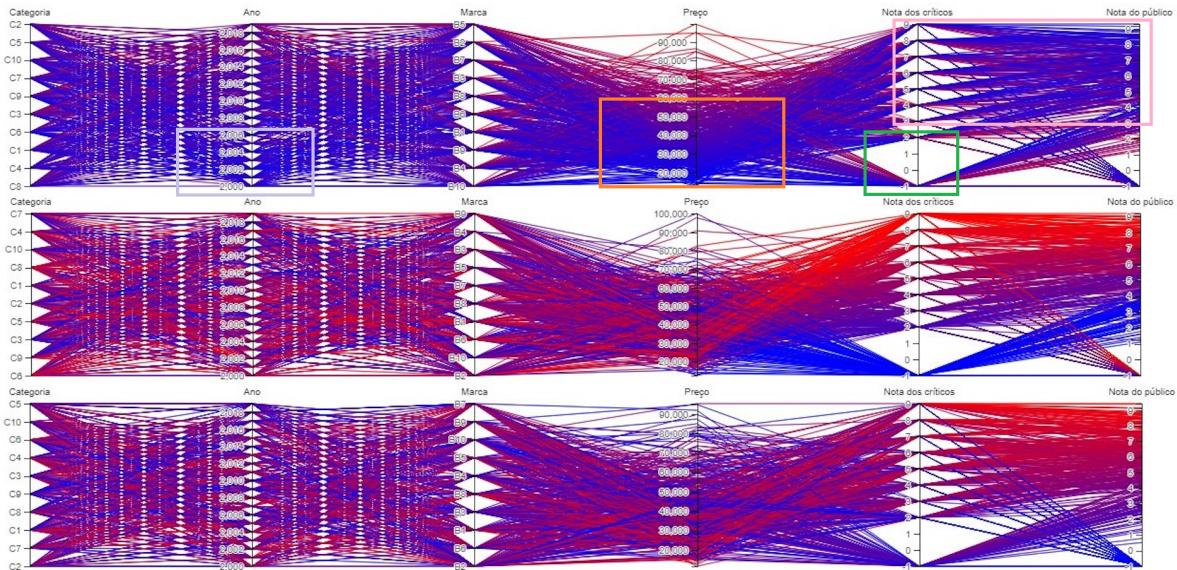
Ainda na figura 45, encontra-se a correlação entre Nota do público e dos Críticos (Marcação vermelha). No preço, há um vácuo na faixa de 80.000 (marcação cinza), apesar de não ter sido intencional, caracteriza-se um MNAR, pois a resposta não está na base de dados. Sobre a correlação, é possível perceber com mais clareza a correlação diretamente proporcional entre nota dos críticos e do público (marcação vermelha), e uma leve correlação inversamente proporcional entre preço e nota do público (marcação alaranjada) e também entre preço e ano (marcação em roxo).

A figura 46 apresenta as principais características do modelo de dados sobre Redes Sociais. Na dimensão Idade é possível encontrar dados faltantes a partir dos 40 anos (marcação alaranjada), logo, caracteriza-se um MNAR, pois não está na base de dados a explicação. E os dados discrepantes nas dimensões de seguidores, curtidas e postagens. É possível identificar exemplos de dados ruidosos (marcação verde) e anômalos (marcação vermelha). O padrão encontrado é que as pessoas possuam uma faixa de duas mil postagens, menos de cinco milhões de seguidores e menos de quinhentas mil curtidas no total. E por conta dos dados discrepantes, a própria escala é prejudicada.

No histograma (ver figura 47) é possível visualizar a questão da escala dos dados. A frequência dos dados numa mesma coluna é muito pouco ou nada nas outras ratifica o problema do alto grau de discrepância da base. Este problema pode ser visto nas dimensões de Curtida, Postagens e Seguidores (marcação alaranjada). Em Idade, pode ser visto uma alta frequência de dados faltantes, mas uma frequência melhor distribuída (marcação em verde).

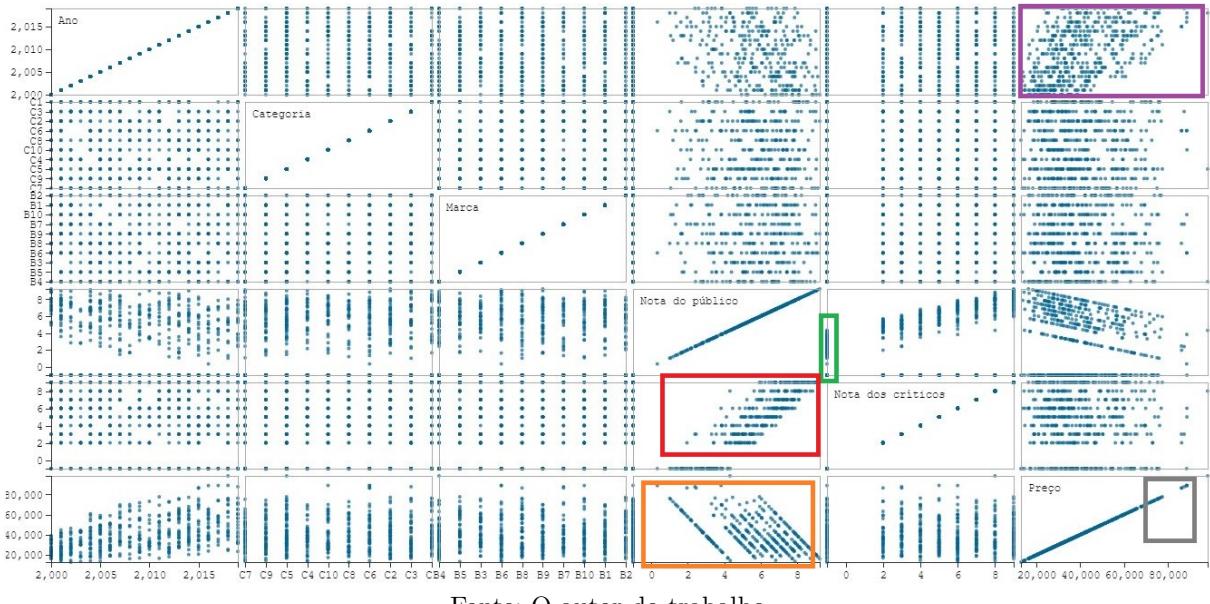
As visualizações das bases anteriores foram geradas pelo *VisApplication* e o *Preview*

Figura 44. Visualização Coordenadas Paralelas da base de carros.



Fonte: O autor do trabalho.

Figura 45. Visualização Scatterplot da base de carros.

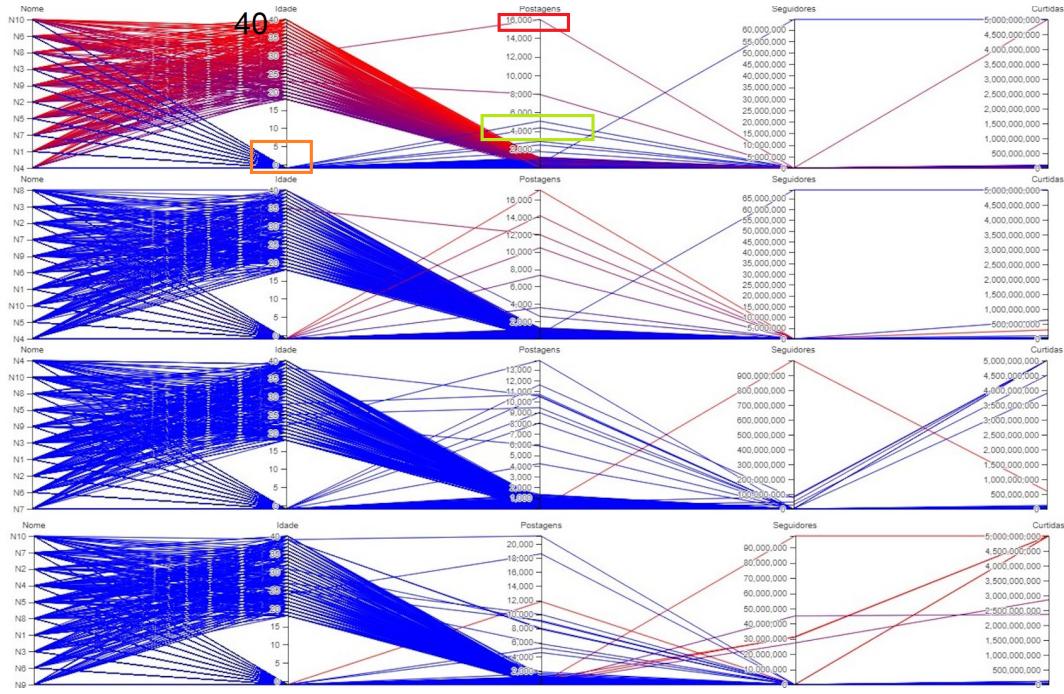


Fonte: O autor do trabalho.

do Blocks. As visualizações do modelo de base genérica foram geradas no Excel, utilizando os gráficos de linha, pontos e colunas agrupadas.

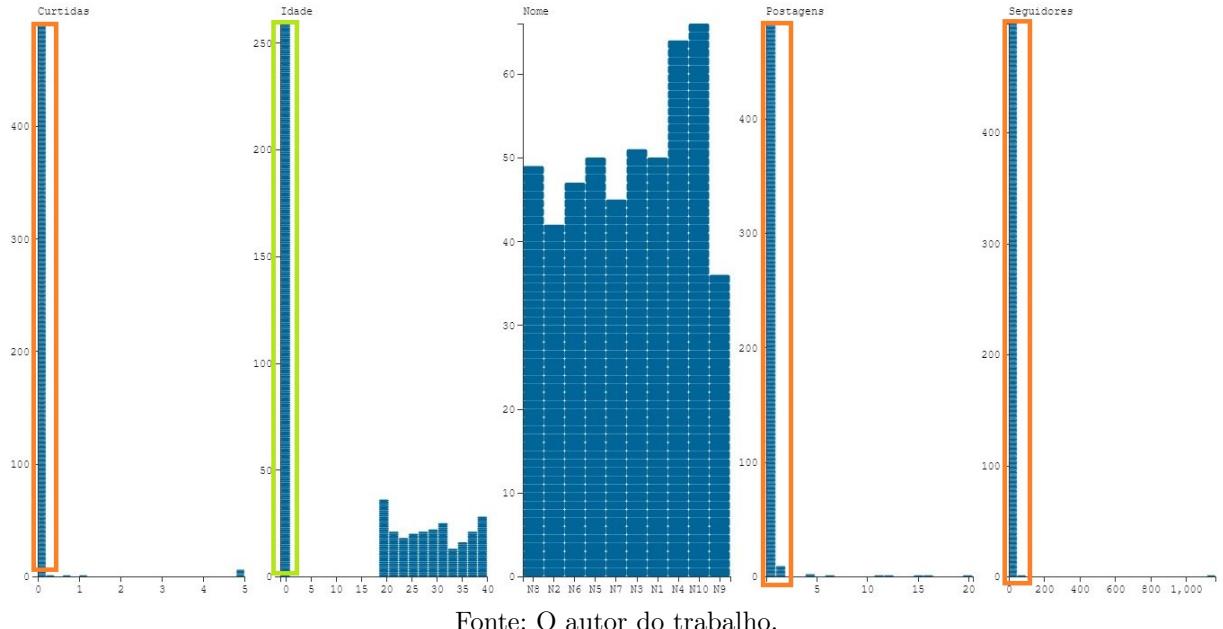
O modelo de dados sobre estruturas de conta bancária possui uma arquitetura hierárquica e pode ser vista por meio do uso de um Dendrograma (ver figura 48). A priori, para esta visualização foi necessário reduzir a 10% - 50 instâncias - o volume de dados, para que as propriedades hierárquicas possam ser percebidas. Na visualização é possível identificar dados faltantes do tipo MCAR, visto que, foram perdidas de forma aleatória,

Figura 46. Visualização Coordenadas Paralelas da base de Redes Sociais.



Fonte: O autor do trabalho.

Figura 47. Visualização Histograma da base de Redes Sociais. A unidade de Curtida é bilhões; Seguidores está em milhões e Postagens está em milhares.

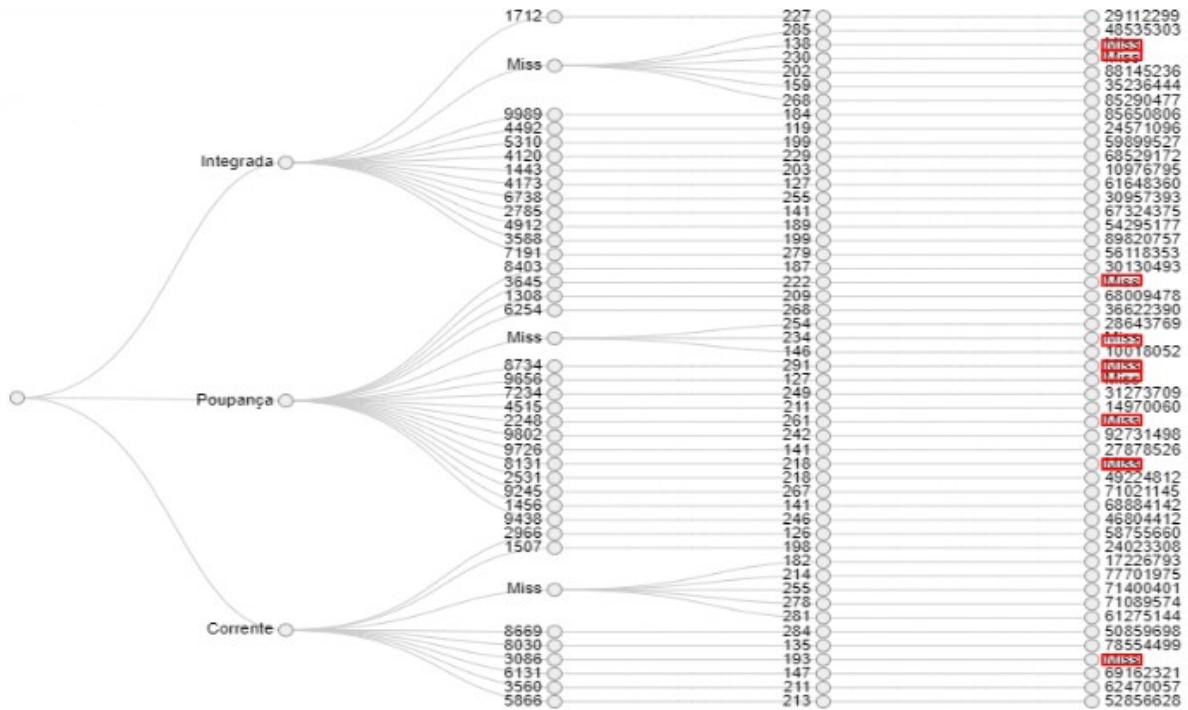


Fonte: O autor do trabalho.

mas não foi imaginada uma forma de gerar dados discrepantes. Também percebe-se que os valores faltantes foram repetidos para que não fosse afetada a propriedade dos dados hierárquicos.

Na figura 49 é possível ver um gráfico de barras subagrupado por especialidades médicas. Neste gráfico pode-se comparar os valores de cada especialidade por plano de saúde. Primeiramente, o SYS1 possui uma discrepância anômala, pois os dados são

Figura 48. Visualização Dendrograma da base sobre estrutura de conta bancária.

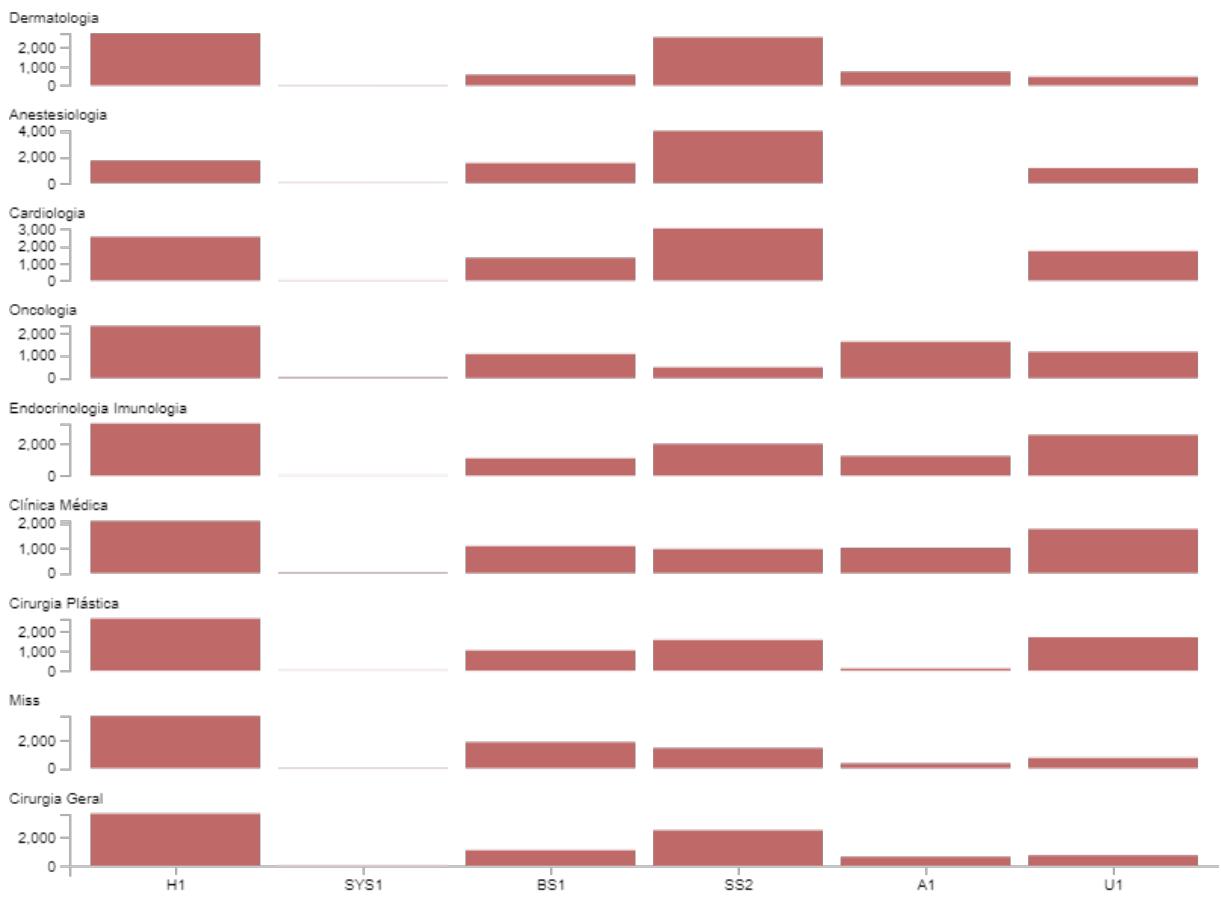


Fonte: O autor do trabalho.

próximos de 0 - fora adicionados valores ínfimos para que não seja confundido com um dado faltante. Os dados faltantes são representados por uma barra sem tamanho. É possível identificar que há discrepância nos dados, mas não foi utilizado um gerador específico, apenas uma função categórica e diferentes geradores de dados, os quais foram escolhidos com o intuito de gerar um comportamento discrepante.

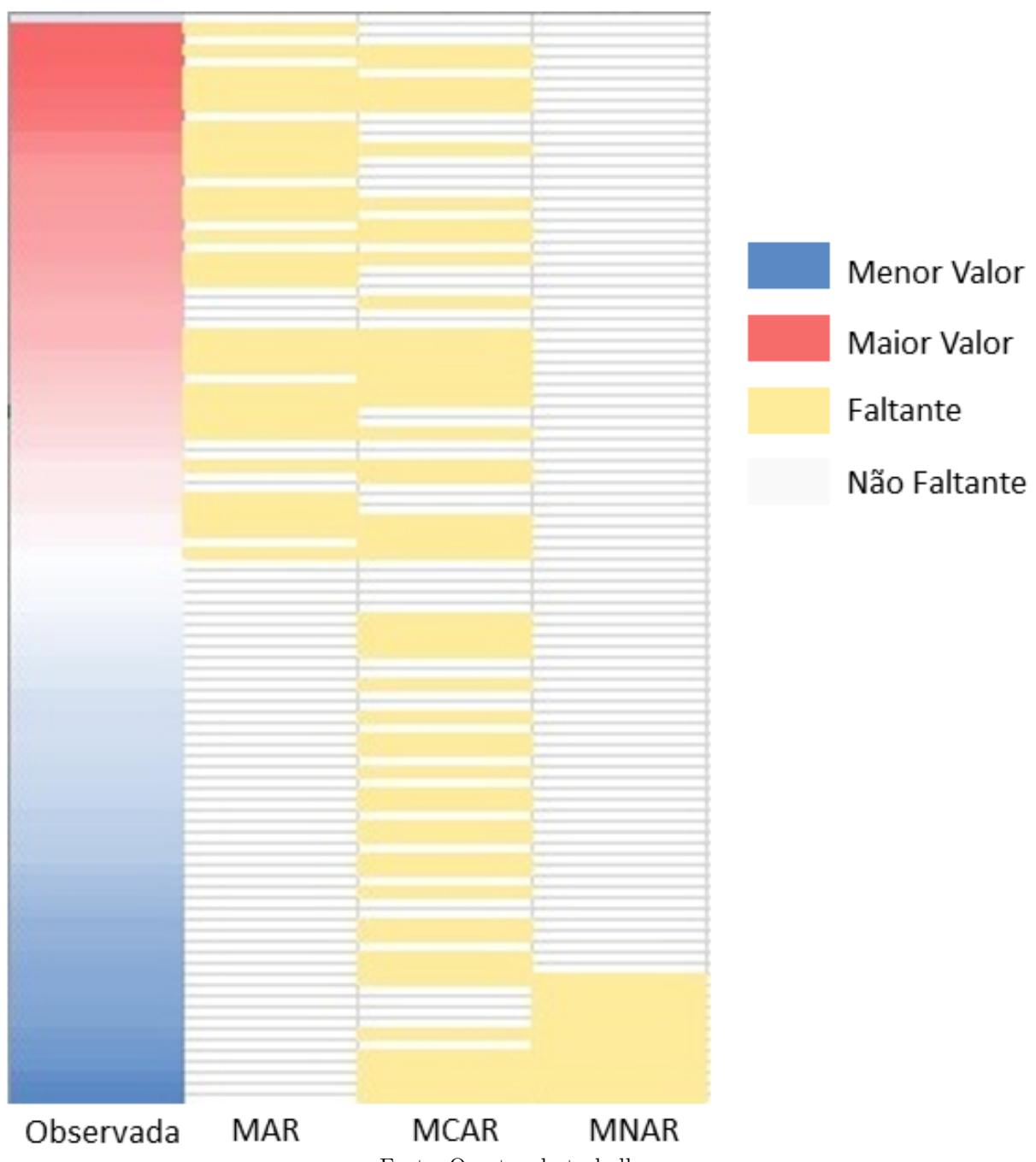
Também foi gerada uma base genérica para visualização dos mecanismo de dados faltantes e essas podem ser vistas na figura 50. As duas dimensões à esquerda foram ordenadas juntas, as outras foram dissociadas para que fossem mantidas as propriedades dos dados faltantes. Na figura percebe-se que os valores da dimensão MAR podem ser previstos a partir da dimensão Observada. A dimensão MCAR possui dados faltantes espalhados, ilustrando um comportamento completamente aleatório. E a coluna MNAR possui somente seus valores mais baixos como dados faltantes, o que pode ter uma explicação fora da base de dados.

Figura 49. Visualização Gráfico de Colunas da base Convênios Médicos. Eixo X: Planos de Saúde, Eixo Y: Especialidades, Barra: Preço por Consulta



Fonte: O autor do trabalho.

Figura 50. Visualização Gráfico de Colunas da base genérica para visualização de dados faltantes.



Fonte: O autor do trabalho.

6 Considerações Finais e Trabalhos Futuros

O desenvolvimento do presente trabalho proporcionou a análise da geração de dados sintéticos discrepantes e faltantes na aplicação Blocks. O sistema é *Open Source* e permite a geração e visualização de dados sintéticos e implementação de novos recursos ao sistema.

Os dados sintéticos são importantes para mais variadas aplicações, por conta da seu desvinculo com a confidencialidade dos dados. O fato de geração basear-se em modelos permite que o compartilhamento dos dados seja mais leve, mas exigindo do processamento em contrapeso. A utilização (modelagem, geração ou visualização) gratuita de dados sintéticos permite que a pesquisa e teste de aplicações seja fomentado e democratizado, pois é acessível para todos.

O Blocks é um sistema competente no que diz respeito à modelagem e geração de dados sintéticos. Sua geração de dados é baseada em modelos, os quais por sua vez possuem blocos de geradores que podem ser encadeados. Esse encadeamento permite que o comportamento dos dados sejam mais complexos, o que distancia de dados mais previsíveis e comuns. Além do encadeamento, o sistema possui uma vasta gama de geradores, desde sequenciais, aleatórios, funcionais, acessórios e geométricos, os quais permitem que o usuário crie variados cenários e níveis de abstração.

O Blocks também permite que esses dados sejam distribuídos via *Web Service*; salvos em arquivos - CSV, JSON, TSV; O sistema permite visualizar os modelos de dados de forma rápida e prática ou de forma mais detalhada e diversa com a integração com o *VisApplication*.

Nos resultados apresentados na seção Validação Visual, os dados faltantes do mecanismo MCAR apresentaram a característica aleatória dada uma taxa de ausência. No MAR, os dados também foram faltantes, mas foi possível enxergar correlação entre as dimensões que possui a ausência e a dimensão observada. O gerador do mecanismo MNAR permite que os dados faltem dada uma lógica - para simular um motivo fora da base de dados. Portanto, de forma geral, a aplicação permitiu que as características - dados faltantes e discrepantes - fossem percebidas no conjunto de dados de forma satisfatória, como vista na seção de Validação Visual, pois além da discrepância e falta de dados, também foi possível fazer correlações.

Os mecanismos de dados faltantes são utilizados para entender porquê aquele dado está faltando. São três mecanismos: *Missing Completely At Random* (MCAR), *Missing At Random* (MAR) e *Missing Not At Random* (MNAR). O mecanismo MCAR é utilizado quando o motivo do dado está faltando não pode ser previsto pela base de dados nem

por fatores externos, por isso é dito como completamente aleatório. O MAR é utilizado quando o dado está faltando por um motivo qualquer, mas seu valor pode ser previsto por meio de alguma dimensão dentro da base dados. Quanto ao MNAR, este mecanismo é utilizado quando o dado está faltando por algum fator externo à base de dados.

Quanto aos dados discrepantes, os quais podem ser subclassificados em ruidosos e anômalos, tem como principal característica um limiar entre eles. Isso porque dados discrepantes podem existir por um valor muito distante dos outros, ou por um caractere alfabético em uma dimensão numérica, ou uma palavra embaralhada, entre outros. Ou seja, o que determina um dado ser ruidoso ou anômalo é o seu impacto na análise dos dados, como a referência distorcida de escala dos dados em gráficos, média dos valores da base de dados muito mais alta ou baixa do que sem os dados discrepantes etc.

Para propor trabalhos futuros para os problemas vistos, é interessante que o Blocks possua mais geradores categóricos, como gerar a partir de uma expressão regular, a partir de um arquivo, geração de anagramas datas as letras ou uma categoria, permitir embaralhar categorias etc. Um gerador temporal sazonal, que possui a estrutura de encadeamento similar ao *TimeLaps Function*. Alguns acessórios também são interessantes como formatar números de diferentes geradores - como ter números discretos ou com casas decimais fixas. Também testes de exaustão com o objetivo de otimizar o número de instâncias de cada bloco para geração de dados.

Referências

- AGGARWAL, C. C. An introduction to outlier analysis. In: *Outlier Analysis*. Springer New York, 2012. p. 1–40. Disponível em: <https://doi.org/10.1007/978-1-4614-6396-2_1>. Citado 2 vezes nas páginas 16 e 22.
- ALBUQUERQUE, G.; LOWE, T.; MAGNOR, M. Synthetic generation of high-dimensional datasets. *IEEE transactions on visualization and computer graphics*, IEEE, v. 17, n. 12, p. 2317–2324, 2011. Citado na página 24.
- ANDRIDGE, R. R.; LITTLE, R. J. A review of hot deck imputation for survey non-response. *International statistical review*, Wiley Online Library, v. 78, n. 1, p. 40–64, 2010. Citado na página 29.
- ATLASSIAN. *Trello*. 2019. Disponível em: <<https://trello.com>>. Acesso em: 22 dec 2019. Citado na página 38.
- ATTORRE, B. F. M. *Web Services REST versus SOAP*. 2015. DevMedia. Disponível em: <<https://www.devmedia.com.br/web-services-rest-versus-soap/32451>>. Acesso em: 09 dec 2019. Citado na página 20.
- BARSE, E. L.; KVARNSTROM, H.; JONSSON, E. Synthesizing test data for fraud detection systems. In: IEEE. *19th Annual Computer Security Applications Conference, 2003. Proceedings*. [S.I.], 2003. p. 384–394. Citado na página 18.
- BERGEAT, M. et al. A french anonymization experiment with health data. In: . [S.I.: s.n.], 2014. Citado na página 18.
- BRAY, T. *The JavaScript Object Notation (JSON) Data Interchange Format*. 2017. Internet Engineering Task Force (IETF). Disponível em: <<https://tools.ietf.org/html/rfc8259>>. Acesso em: 31 jul 2019. Citado na página 19.
- CROCKFORD, D. *ECMA-404 The JSON Data Interchange Standard*. 2003. Json.org. Disponível em: <<https://json.org/json-pt.html>>. Acesso em: 31 jul 2019. Citado na página 19.
- DEAN, S.; ILLOWSKY, B. Descriptive statistics: Histogram. Retrieved from the Connexions Web site: <http://cnx.org/content/m16298/1.11>, 2009. Citado na página 26.
- DensityDesign Research Lab. *RAWGraphs*. 2019. Disponível em: <<https://app.rawgraphs.io>>. Acesso em: 22 dec 2019. Citado na página 64.
- DEVART. *Data Generator for SQL Server*. 2018. [Https://www.devart.com](https://www.devart.com). Disponível em: <<https://docs.devart.com/data-generator-for-sql-server/>>. Acesso em: 21 ago 2019. Citado 2 vezes nas páginas 33 e 35.
- Devart Documentation. *Data Generator for SQL Server Documentation*. 2019. Disponível em: <<https://docs.devart.com/data-generator-for-sql-server/basic-generators/check-constraint-generator>>. Acesso em: 10 dec 2019. Citado na página 34.

DTM Soft. *DTM Database Tools*. 2019. [Http://www.sqledit.com/](http://www.sqledit.com/). Disponível em: <<http://www.sqledit.com/dg/index.html>>. Acesso em: 17 ago 2019. Citado 2 vezes nas páginas 30 e 32.

EDUCATION, M.-H. *The McGraw-Hill Dictionary of Scientific and Technical Terms, Seventh Edition (McGraw-Hill Dictionary of Scientific & Technical Terms)*. McGraw-Hill Professional, 2016. ISBN 0071608990. Disponível em: <<https://www.amazon.com/McGraw-Hill-Dictionary-Scientific-Technical-Seventh/dp/0071608990?SubscriptionId=AKIAIOBINVYZQZ2U3A&ttag=chimbori05-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=0071608990>>. Citado na página 18.

FIELDING, R. T.; TAYLOR, R. N. *Architectural styles and the design of network-based software architectures*. [S.l.]: University of California, Irvine Doctoral dissertation, 2000. v. 7. Citado na página 20.

FREED J. KLENSIN, J. P. N. *Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures*. 1996. Internet Engineering Task Force (IETF). Disponível em: <<https://tools.ietf.org/html/rfc2048>>. Acesso em: 31 jul 2019. Citado na página 19.

GARCIA, D.; MILLAN, M. A prototype of synthetic data generator. In: IEEE. *2011 6th Colombian Computing Congress (CCC)*. [S.l.], 2011. p. 1–6. Citado 2 vezes nas páginas 25 e 27.

GROUP, W. W. *Web Services Architecture*. 2004. [Www.w3.org](http://www.w3.org). Disponível em: <<https://www.w3.org/TR/ws-arch/>>. Acesso em: 02 ago 2019. Citado 2 vezes nas páginas 19 e 20.

HAUSENBLAS E. WILDE, J. T. M. *ECMA-404 The JSON Data Interchange Standard*. 2014. Internet Engineering Task Force (IETF). Disponível em: <<https://tools.ietf.org/html/rfc7111#page-3>>. Acesso em: 31 jul 2019. Citado na página 19.

KOFINAS, D. T.; SPYROPOULOU, A.; LASPIDOU, C. S. A methodology for synthetic household water consumption data generation. *Environmental modelling & software*, Elsevier, v. 100, p. 48–66, 2018. Citado 2 vezes nas páginas 26 e 27.

KORPELA, J. *Tab Separated Values (TSV): a format for tabular data exchange*. 2000. [Http://jkorpela.fi](http://jkorpela.fi). Disponível em: <<http://jkorpela.fi/TSV.html>>. Acesso em: 31 jul 2019. Citado na página 19.

KUMAR, V. *15 Best Test Data Generation Tools In 2019*. 2019. [Https://www.rankred.com](https://www.rankred.com). Disponível em: <<https://www.rankred.com/test-data-generation-tools/>>. Acesso em: 17 ago 2019. Citado na página 18.

LARSEN, M. D.; HUCKETT, J. C. Multimethod synthetic data generation for confidentiality and measurement of disclosure risk. *International Journal of Information Privacy, Security and Integrity*, Inderscience Publishers, v. 1, n. 2/3, p. 184, 2012. Disponível em: <<https://doi.org/10.1504/ijipsi.2012.046132>>. Citado na página 29.

LITTLE, T. D. et al. Missing data. *Developmental psychopathology*, Wiley Online Library, p. 1–37, 2016. Citado 2 vezes nas páginas 20 e 21.

LIU, R. et al. Synthetic data generator for classification rules learning. In: IEEE. *2016 7th International Conference on Cloud Computing and Big Data (CCBD)*. [S.l.], 2016. p. 357–361. Citado 2 vezes nas páginas 25 e 26.

LOPEZ-ROJAS, E. A.; AXELSSON, S. Money laundering detection using synthetic data. In: LINKÖPING UNIVERSITY ELECTRONIC PRESS. *The 27th annual workshop of the Swedish Artificial Intelligence Society (SAIS); 14-15 May 2012; Örebro; Sweden*. [S.l.], 2012. p. 33–40. Citado na página 18.

Lucid Software Inc. *UML Use Case Diagram Tutorial*. 2019. [Https://www.lucidchart.com/](https://www.lucidchart.com/). Disponível em: <<https://www.lucidchart.com/pages/uml-use-case-diagram>>. Acesso em: 09 sep 2019. Citado na página 38.

MCKNIGHT, P. E. *Missing Data: A Gentle Introduction (Methodology in the Social Sciences)*. The Guilford Press, 2007. ISBN 9781593853938. Disponível em: <<https://www.xarg.org/ref/a/1593853939/>>. Citado na página 20.

Microsoft. *Microsoft Excel*. Disponível em: <<https://products.office.com/pt-br/excel>>. Citado na página 64.

MICROSOFT. *Generating Test Data for Databases by Using Data Generators*. 2019. [Https://www.microsoft.com/](https://www.microsoft.com/). Disponível em: <[https://docs.microsoft.com/en-us/previous-versions/visualstudio/visual-studio-2010/dd193262\(v=vs.100\)](https://docs.microsoft.com/en-us/previous-versions/visualstudio/visual-studio-2010/dd193262(v=vs.100))>. Acesso em: 19 ago 2019. Citado 2 vezes nas páginas 32 e 34.

MOCKAROO. *Mockaroo, realistic data generator*. 2019. [Https://www.mockaroo.com/](https://www.mockaroo.com/). Disponível em: <<https://www.mockaroo.com/>>. Acesso em: 23 ago 2019. Citado 2 vezes nas páginas 34 e 36.

Mockaroo Documentation. *Mockaroo APIs*. 2019. [Https://www.mockaroo.com/](https://www.mockaroo.com/). Disponível em: <<https://www.mockaroo.com/api/docs>>. Acesso em: 23 ago 2019. Citado na página 35.

MOLENBERGHS, G. et al. *Handbook of missing data methodology*. [S.l.]: Chapman and Hall/CRC, 2014. Citado 2 vezes nas páginas 20 e 21.

MORAES, D. *Um pensamento sobre os Dados Sintéticos*. 2019. Update or Die. Disponível em: <<https://www.updateordie.com/2019/09/25/um-pensamento-sobre-os-dados-sinteticos>>. Acesso em: 09 dec 2019. Citado na página 15.

OpenJS Foundation. *Electron*. 2019. Disponível em: <<https://electronjs.org>>. Acesso em: 22 dec 2019. Citado na página 38.

OpenJS Foundation. *Node.js*. 2019. Disponível em: <<https://nodejs.org/pt-br/>>. Acesso em: 22 dec 2019. Citado na página 38.

RATHI, A. *Dealing with Noisy Data in Data Science*. Analytics Vidhya, 2019. Disponível em: <<https://medium.com/analytics-vidhya/dealing-with-noisy-data-in-data-science-e177a4e32621>>. Citado 3 vezes nas páginas 22, 23 e 24.

- RAYMOND, E. S. *Data File Metaformats. Chapter 5. Textuality.* 2003. <Http://www.catb.org>. Disponível em: <<http://www.catb.org/~esr/writings/taoup/html/ch05s02.html>>. Acesso em: 31 jul 2019. Citado na página 19.
- Red Gate. *SQL Data Generator*. 2019. <Https://www.red-gate.com/>. Disponível em: <<https://www.red-gate.com/products/sql-development/sql-data-generator/>>. Acesso em: 18 ago 2019. Citado na página 31.
- Red Gate Documentation. *SQL Data Generator Documentation*. 2019. Disponível em: <https://documentation.red-gate.com/xx/files/7471185/15925341/1/1370009573853/SQL+Data+Generator+1_2.pdf>. Acesso em: 10 dec 2019. Citado 2 vezes nas páginas 32 e 33.
- RIEGER, A.; HOTHORN, T.; STROBL, C. Random forests with missing values in the covariates. 2010. Citado na página 52.
- RODRÍGUEZ-HERNÁNDEZ, M. del C. et al. Datagencars: A generator of synthetic data for the evaluation of context-aware recommendation systems. *Pervasive and Mobile Computing*, Elsevier BV, v. 38, p. 516–541, jul. 2017. Disponível em: <<https://doi.org/10.1016/j.pmcj.2016.09.020>>. Citado 2 vezes nas páginas 29 e 30.
- RUBIN, D. B. Statistical disclosure limitation. *Journal of official Statistics*, v. 9, n. 2, p. 461–468, 1993. Citado na página 18.
- SAKSHAUG, J. W.; RAGHUNATHAN, T. E. Nonparametric generation of synthetic data for small geographic areas. In: SPRINGER. *International Conference on Privacy in Statistical Databases*. [S.l.], 2014. p. 213–231. Citado 2 vezes nas páginas 27 e 28.
- SANTOS, M. S. et al. Generating synthetic missing data: A review by missing mechanism. *IEEE Access*, IEEE, v. 7, p. 11651–11667, 2019. Citado na página 52.
- SHAFRANOVICH, Y. *Common Format and MIME Type for Comma-Separated Values (CSV) Files*. 2005. Internet Engineering Task Force (IETF). Disponível em: <<https://tools.ietf.org/html/rfc4180#page-2>>. Acesso em: 31 jul 2019. Citado na página 19.
- STACKIFY. *SOAP vs. REST: The Differences and Benefits Between the Two Widely-Used Web Service Communication Protocols*. 2017. Stackify.com. Disponível em: <<https://stackify.com/soap-vs-rest/>>. Acesso em: 02 ago 2019. Citado na página 20.
- TANENBAUM, A. S.; FILHO, N. M. *Sistemas operacionais modernos*. [S.l.]: Prentice-Hall, 1995. v. 3. Citado na página 19.
- TWALA, B. An empirical comparison of techniques for handling incomplete data using decision trees. *Applied Artificial Intelligence*, Taylor & Francis, v. 23, n. 5, p. 373–405, 2009. Citado na página 52.
- WANG, B.; RUCHIKACHORN, P.; MUELLER, K. Sketchpadn-d: Wydiwyg sculpting and editing in high-dimensional space. *IEEE Transactions on Visualization and Computer Graphics*, IEEE, v. 19, n. 12, p. 2060–2069, 2013. Citado na página 25.

WHITING, M. A.; HAACK, J.; VARLEY, C. Creating realistic, scenario-based synthetic data for test and evaluation of information analytics software. In: ACM. *Proceedings of the 2008 Workshop on BEyond time and errors: novel evaLuation methods for Information Visualization*. [S.l.], 2008. p. 8. Citado 2 vezes nas páginas 28 e 29.

XIA, J. et al. Adjusted weight voting algorithm for random forests in handling missing values. *Pattern Recognition*, Elsevier, v. 69, p. 52–60, 2017. Citado na página 52.

Apêndices

Apêndice A – Atalhos do Teclado da aplicação Blocks

Modelar um conjunto de dados pode ser exaustivo, portanto, alguns atalhos podem facilitar na prevenção e correção de erros, eficiência e economia de esforço e afins. Visando dar praticidade ao usuário, o Block Data Generator possui atalhos de teclados que podem ser vistos na tabela 5.

Tabela 5. Tabela de atalhos do teclado do Blocks.

Nome	Comando	Descrição
<i>New Model</i>	Ctrl/Cmd+M	Cria novo modelo.
<i>Delete Model</i>	Ctrl/Cmd+W	Deleta modelo atual.
<i>New Dimension</i>	Ctrl/Cmd+D	Cria nova dimensão.
<i>Open Model</i>	Ctrl/Cmd+O	Insere um modelo no sistema.
<i>Save Model</i>	Ctrl/Cmd+S	Salva as alterações do modelo atual.
<i>Undo</i>	Ctrl/Cmd+Z	Desfaz uma alteração.
<i>Redo</i>	Ctrl/Cmd+Shift+Z	Refaz uma alteração.

Fonte: O autor do trabalho.

Além de acesso pelo teclado, os atalhos podem ser acessados pela barra superior da tela inicial 33 (22). É válido ressaltar sobre a questão de edição da modelagem, que o Blocks salva as mudanças no automaticamente em arquivo separado, o qual pode ser recuperado se não forem salvas/descartadas adequadamente.