UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Campus de Assis

JOSÉ LEONEL LEMOS BUZZO

PLATAFORMA COMPUTACINAL ON-LINE PARA CONFECÇÃO DE MODELOS NUMÉRICOS MULTIVARIADOS SOB DEMANDA

Relatório apresentado ao Conselho do Curso de Graduação em Engenharia Biotecnológica da Faculdade de Ciências e Letras - UNESP, Campus de Assis (SP), como parte das exigências para o cumprimento da disciplina Monografia de Conclusão de Curso II no ano letivo de 2017.

Orientador: Prof. Dr. José Celso Rocha

Assis - SP

21 de junho de 2017

Avenida Dom Antônio 2100 - CEP 19806-900 - Assis - SP - Brasil
Departamento de Ciências Biológicas
Tel.: 55 (018) 3302 5848 - http://www.assis.unesp.br/

UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Campus de Assis

JOSÉ LEONEL LEMOS BUZZO

PLATAFORMA COMPUTACINAL ON-LINE PARA CONFECÇÃO DE MODELOS NUMÉRICOS MULTIVARIADOS SOB DEMANDA

Relatório apresentado ao Conselho do Curso de Graduação em Engenharia Biotecnológica da Faculdade de Ciências e Letras - UNESP, Campus de Assis (SP), como parte das exigências para o cumprimento da disciplina Monografia de Conclusão de Curso II no ano letivo de 2017.

Orientador: Prof. Dr. José Celso Rocha

Assis - SP
21 de junho de 2017

BUZZO, JOSÉ LEONEL LEMOS. 1986 –
PLATAFORMA COMPUTACINAL ON-LINE PARA CONFECÇÃO DE MODELOS NUMÉRICOS MULTIVARIADOS SOB DEMANDA/ JOSÉ LEONEL LEMOS BUZZO. -Assis - SP, 2017 22 p. : il.(preto.); 30 cm.
Orientador:José Celso Rocha.

monografia(Engenharia Biotecnológica) – UNESP, 2017

1. Plataforma Computacional. 2. Modelos Numéricos. I. José Celso Rocha. II. UNESP. III. Faculdade de Ciências e Letras de Assis. IV. PLATAFORMA COMPUTACINAL ON-LINE PARA CONFECÇÃO DE MODELOS NUMÉRICOS MULTIVARIADOS SOB DEMANDA

CDU 00:000:000.7

JOSÉ LEONEL LEMOS BUZZO

PLATAFORMA COMPUTACINAL ON-LINE PARA CONFECÇÃO DE MODELOS NUMÉRICOS MULTIVARIADOS SOB DEMANDA

Relatório apresentado ao Conselho do Curso de Graduação em Engenharia Biotecnológica da Faculdade de Ciências e Letras - UNESP, Campus de Assis (SP), como parte das exigências para o cumprimento da disciplina Monografia de Conclusão de Curso II no ano letivo de 2017.

Trabalho aprovado:

Profº. Drº. José Celso Rocha

Profª. Drª. Juliana de Oliveira

Profº. Drº. Fernando Frei

Assis - SP
21 de junho de 2017

Este trabalho é dedicado às minhas avós:
Izaura Ziroldo Buzzo e **Rosa Peres Martins**
Mulhres fortes e, sobretudo, amorosas.

Agradecimentos

Agradeço a Deus,
por ter cruzado meu caminho com o de **Priscila Soares Navarro Buzzo**,
a quem tive o privilégio de amar.

Tudo está claro agora!
— Flausino

Resumo

Implementou-se uma plataforma computacional, passível de acesso, tanto *on-line*, quanto local (instalável), que viabiliza a submissão de dados experimentais e a automatização dos principais processos de pesquisa em modelos numéricos e de simulação adequados àqueles dados.
Intensionou-se assim, tornar disponível, sob licença livre, uma biblioteca de algoritmos modulares que dê suporte às rotinas de pesquisa supracitadas e que, adicionalmente, permita sua integração a outras plataformas, provendo modelos padronizados, programáveis e sob demanda.
Testando-se sua robustez, duas aplicações diversas e bem sucedidas de modelos numéricos envolvendo Redes Neurais Artificiais foram tomadas da literatura, afim de se comparar os resultados publicados em cada caso com os resultados apresentados pelo modelo correspondente gerado automaticamente pela plataforma.
Eis que a plataforma obteve êxito ao gerar modelos com desempenhos sensivelmente inferiores aos tomados da literatura. Demonstrou-se também a praticidade na confecção dos modelos em detrimento de seus menores desempenhos, a despeito da pluralidade conceitual dos problemas abordados nos dois casos.
Em suma, esta plataforma vêm sugerir novas perspectivas no lavoro técnico computacional, especialmente na possibilidade de se integrar, de forma intuitiva, dados de pesquisadores oriundos de áreas bastante distintas num mesmo aparato sistêmico e tecnológico, por meio de modelos e arquivos padronizados e de livre acesso.

Palavras-chaves: *Plataforma Computacional*, *Modelos Numéricos Multivariados*, *Simulação*.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

ABNT	(Associação Brasileira de Normas Técnicas) Portal para consulta de normas técnicas em infromação e documentação, mantido pelo governo brasileiro. Endereço eletrônico: <https://www.abnt.br>
FSF	(*Free Software Foundation*) Fundação em prol de *Softwares* Livres, fundação internacional responsável pelo sistema GNU (abaixo) e liceças de *software* livre. Endereço eletrônico: <http://www.fsf.org/>
GDC	(*Genomic Data Commons*) Comunalidades de Dados Genômicos, repositório público de dados genômicos e oncológicos mantido pelo instituto nacional do câncer norte americano. Endereço eletrônico: <https://gdc.cancer.gov/>
Git	Programa de controle de versões de arquivos eletrônicos mantido por iniciativa *open source*. Endereço eletrônico: <https://git-scm.com/>
GNU	Sistema operacional e um grande número de programas livres mantido pela FSF. Endereço eletrônico: <https://www.gnu.org/>
GPLv3	(*GNU General Public License version 3*) Versão terceira da Licença Publica Geral da GNU que regulamenta o uso e a distribuição de *softwares* livres. Endereço eletrônico: <http://www.gnu.org/licenses/>
http	(*Hiper Text Transfer Protocol*) Protocolo de Transferência de Hipertexto, proposto e mantido pelo consórcio internacional w3c (*World Wide Web Consortium*) para regulamentação da internet. Endereço eletrônico: <https://www.w3.org/Protocols>
ISO	(*International Organization for Standardization*) Organização Internacional para Padronização. Endereço eletrônico: <https://www.iso.org/>
LaTeX	*LaTeX - A document preparation system*. LaTeX - Um sistema de prepação de documentos, mantido por iniciativa *open source*. Endereço eletrônico: <https://www.latex-project.org/>
Matlab	*The language of technical computing*. A linguagem de computação técnica, plataforma de computação e linguagem de programação desenvolvida pela

empresa MathWorks®.
Endereço eletrônico: <https://www.mathworks.com/>

OSI — (*Open Source Initiative*) Iniciativa *Open Source*, organização não governamental criada em 1998 para redação e regulamentação dos termos e dos direitos de uso e distribuição de *softwares* livres, ou de código aberto.
Endereço eletrônico: <https//www.osi.org/>

POSIX — (*Portable Operating System Interface eXtended*) Interface de Sistema Operacional eXtendida, padrão de interface mínima que um sistema operacinal deve ter.
Endereço eletrônico: <https://standards.ieee.org/findstds/standard/1003.1-2008.html>

R — *The R Project for Statistical Computing*. O Projeto R para Computação Estatística, linguagem de progamação mantida por iniciativa *Open Source*,
Endereço eletrônico: <https://www.r-project.org/>

SSH — (*Security Shell*) Terminal *Shell* Seguro, programa de conexão remota criptografada desenvolvido e mantido por iniciativa *open source*.
Endereço eletrônico: <https://www.openssh.org/>

UCI — *University of California–Irvine Machine Learning Repository*. Repositório público de dados para aprendizagem de máquinas mantido pela universidade norte americana da Califórinia.
Endereço eletrônico: <http://archive.ics.uci.edu/ml/>

XML — (*eXtensible Markup Language*) Linguagem de marcação extensível, o formato de arquivo eletrônico mais comum empregado para descrição textual de dados estruturados hoje.
Endereço eletrônico: <https://www.w3.org/XML/>

SUMÁRIO

1 INTRODUÇÃO

1.1 O acesso à tecnologia e a demanda por plataformas computacionais

A conjuntura científico-industrial impõe importantes desafios ao poderio computacional da atualidade. E, a superação destes, a saber, a obtenção de crescente acurácia na confecção de modelos matemáticos multivariados e não lineares aplicados às tecnologias estratégicas, decisivamente coloca seus detentores enquanto potências multinacionais da técnica, concomitantemente às implicações econômico-sociais advindas.

Vêem-se, nas literaturas das mais variadas áreas do conhecimento, cada vez mais os termos: “Inteligência Computacional”, “Aprendizagem de Maquinas” e “Biologia Matemática”. Estes têm por finalidade fazer alusão a um paradigma emergente nas ciências — a previsibilidade sobre os sistemas dinâmicos que representam os diversos fenômenos e objetos de estudo (LANTZ, 2013; SHALEV-SHWARTZ; BEN-DAVID, 2014).

Previsibilidade em problemas com alto grau de complexidade e que requerem modelos diferenciais não lineares ou mesmo estocásticos é, em todas as áreas do conhecimento, um poder tido como o graal da computação na atualidade. Isto advém do fato de que os métodos tradicionais de modelagem computacional e prediçao estatística, tais como as analises de regressão multivariada, mostram-se bastante limitados quanto à sua capacidade de generalização frente novos dados que extrapolem os originais, além de ter sua implementação rapidamente inviabilizada quando o número de variáveis cresce demasiadamente. Analogamente, tomando-se variáveis que se correlacionam de forma acentuadamente não linear ou imbuídas de ruído, mesmo poucas variáveis podem facilmente saturar sistemas robustos de processamento, exigindo custosas arquiteturas computacionais distribuídas. Eis que a demanda por técnicas integradas capazes de resolver os problemas supracitados passou a ser posta em primeiro plano nas pesquisas de ponta contemporâneas (PERRY; MONTGOMERY; FOWLER, 2007; SAMARASINGHE, 2007).

Evidência disso é a proliferação de diversos portais para disponibilização e armazenamento de dados experimentais na internet, tais como o repositório público UCI, para aprendizagem de máquinas (UNIVERSITY OF CALIFORNIA–IRVINE, 2017), e o repositório GDC, para curação e disponibilização de informações genômicas e oncológicas (NATIONAL CANCER INSTITUTE, 2017). Contrastando com a escassez de pĺataformas computacionais de livre acesso capazes processar estes grandes volumes de dados, justamente pelo custo computacional e diversidade conceitual.

Some-se ainda, a carência por convenções ou protocolos tecnologicamente neutros (universalistas e não regidos por plataformas privadas específicas) que venham a facilitar a tramitação de dados gerados e demandados pelas diferentes áreas da indústria e das ciências, nos seus diversos formatos e de forma integrada, como propõem os padrões ISO e POSIX (INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, 2017; INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, 2017).

1.2 Uma proposta de automatização e padronização de procedimentos acadêmicos e industriais

1.2.1 *Proposta*

Propõe-se aqui implementar uma plataforma computacional, passível de acesso, tanto *online*, quanto local (instalável), que viabilize a submissão de dados experimentais e a automatização dos principais processos de pesquisa em modelos numéricos e de simulação adequados àqueles dados.

Intensionou-se, assim, tornar disponível, sob licença livre, uma biblioteca de algoritmos modulares que dê suporte à estas rotinas de pesquisa e que, adicionalmente, permita sua integração a outras plataformas, provendo modelos padronizados, programáveis e sob demanda.

Referir-se-á à plataforma de forma genérica, simplesmente como "plataforma" ou como "software de suporte à plataforma", para fins de alusão. Porém, tornou-se necessário a escolha de um nome conveniente para o registro do domínio de internet referente ao site que a hospedar. Fora escolhido o nome *neurocelle*, resultando no domínio <http://www.neurocelle.org>, registrado pelo autor.

1.2.2 *Modularização dos procedimentos*

A organização dos algoritmos constituintes do software que suporta a plataforma em questão se dá modularmente, de tal forma que os processos em trâmite possam transcorrer independentemente. Optou-se por esta implementação por recomendação de boas práticas de programação já consagradas em Engenharia de Software. Tal modularização promove a abstração conceitual dos códigos envolvidos, favorecendo sua escrita e manutênção, o isolamento correto dos dados e a produção de um formato comum de saída para aqueles a serem exportados ou trocados entre os módulos. Escrevendo-se sinergicamente os módulos, introduz-se um padrão a ser seguido no delineamento de novos módulos por parte dos usuários.

Dentre as atividades mais corriqueiras desempenadas por pesquisadores, estão: o delineamento dos experimentos, a escrita dos respectivos projetos, o tratamento estatístico dos dados experimentais, criação e simulação de modelos que se adequem aos dados, a escrita de relatórios e artigos e, frequentemente, a disponibilização destes dados e modelos em um formato conveniente à divulgação eletrônica ou redistribuição em outros softwares direcionados a sua aplicação e a usuários comerciais.

Logo, foram propostos sete módulos com nomes sugestivos, voltados à automatização das práticas acadêmicas supracitadas. No entanto, apenas três deles foram de fato implementados. Os outros são mensionados por serem referenciados internamente na estrutura de dados do objeto principal e portanto resguardam o mecanismo que os associarão a suas respectivas funcionalidades em implementações futuras. São eles:

- ***neurocelle*:** Objeto principal de autoinstalação e instalação dos módulos.

- ***sinergio* (não implementado):** Módulo de controle de processos e memória do sistema operacional.
- ***progetto*:** Meta-módulo de manipulação dos outros módulos em um projeto.
- ***astratto*:** Módulo de abstração de modelos.
- ***vero* (não implementado):** Módulo de simulação de modelos.
- ***parlo* (não implementado):** Módulo de confecção de documentos (relatórios, gráficos, entre outros).
- ***viso* (não implementado):** Módulo de criação de interfaces gráficas e instaláveis.

Não obstante o número de módulos implementados e sugeridos, outros podem ser facilmente programados pelo usuário, dada a generalização empregada em sua arquitetura. Sendo assim, qualquer função ou rotina de programação capaz de, tanto aceitar o formato de arquivo convencionado (neste caso XML), quanto de gerá-lo a partir de seus resultados, pode ser referenciada para uso como um módulo pelo objeto principal.

1.2.3 *Confecção automática e resultados padronizados*

Visto que alguns procedimentos de pesquisa cotidianos, como escrita de projetos, relatórios, artigos e até mesmo interfaces gráficas, requerem normas de formatação estabelecidas para seu feitio, *templates* padronizados podem ser aplicados para este fim. Logo, utilizando arquivos XML para descrever as estruturas de dados que o software faz uso, sua transcrição para outros formatos torna-se programável e automática.

Consequentemente, projetos, relatórios e artigos são transcritos, a partir de arquivos XML oriundos da saída do software, em templates ABNT escritos segundo a linguagem de formatação de documentos LaTeX que, por sua vez, são compilados em arquivos no formato PDF. A documentação dos resultados obtidos torna-se, desta forma, padronizada e com melhor qualidade de *layout*. Analogamente, interfaces gráficas específicas aos dados são compiladas a partir do mesmo arquivo XML, podendo ser adicionalmente customizadas pela sua edição.

1.2.4 *Controle de versões e repositório central*

Dado que o software será livremente distribuído também para instalação local e que ususráios serão exortados a desenvolverem seus próprios módulos, surge a eventual necessidade de se restaurar seu estado inicial de instalação ou de se registrar um estado estável de desenvolvimetno mais atual. Neste contexto, utiliza-se, e está incluso no software original da plataforma, um software auxiliar para controle de versões — o Git. Postas as funcionalidades deste, quaisquer alterações na árvore de diretórios podem ser revertidas definitivamente, ou podem coexistir como uma versão alternativa em desenvolvimento, ao bel prazer do usuário.

1.3 **Aplicabilidade, robustez e segurança**

1.3.1 *Modelos numéricos e simulações*

Posta a natureza multivariada dos fenômenos envolvidos nas atividades de pesquisa ou de interesse comercial, modelos numéricos condizentes precisam ser implementados. Aqui, é de suma importância a confiança em sua robustez, dadas suas aplicações em áeras estratégicas e de tomada de decisões, as quais se extendem, por exemplo, da previsão de impactos antropogênicos em cenários ecológicos, à acurácea de um diagnóstico oncológico — aplicações cujas falhas acarretariam consequencias catastróficas (GOETHALS et al., 2007; BELLE et al., 2004).

Neste interím, há de se reiterar que modelos numéricos mais robustos e indicados para estas situações, frequentemente, dependem de uma quantidade elevada de parâmetros, os quais, escolhidos erroneamente, podem inviabilizar as análises e modelagens em questão (MONTGOMERY, 2013).

Por isso, é conveniente introduzir brevemente os modelos implementados e seu funcionamento, bem como o contexto dos quais emergiram. Subescrevem-se, pois, as técnicas de modelagem numérica sabidamente mais promissoras e mais amplamente utilizadas hoje, segundo a literatura.

1.3.1.1 Redes Neurais Artificiais

Concebidas justamente no contexto da crise do pensamento determinístico do início do século XX, as *Redes Neurais Artificiais* — RNAs — (ou *Artificial Neural Networks* — ANN) imbuem-se de capacidade de generalização e predição consideráveis, mostrando grande aplicabilidade a problemas o mais plurais possível, estendendo-a ao comportamento do mercado financeiro, à neurociência, à ecologia e tantas outras áreas (KOVACS, 2002).

O termo RNA encerra um vasto conjunto de técnicas mateméticas as quais perfazem, de forma abstrata, o funcionamento do neurônio biológico em processamento e aprendizagem, tendo como peça elementar o seu análogo artificial. E, como no tecido nervoso biológico, conjuntos de neurônios artificiais interligados possuem a peculiaridade de generalizar adaptativamente o reconhecimento de padrões previamente vistos, a outrem desconhecidos.

Não obstante, uma RNA é definida pelo paradigma de aprendizagem em que se insere e, em estreita relação com este, pela sua arquitetura.

Assim, entendendo-se por paradigma de aprendizagem, a forma na qual uma RNA assimila as informações de treinamento, descrevem-se duas categorias de algoritmos de aprendizado – os supervisionados e os não supervisionados. Ambos diferem quanto à presença, ou não, de um gabarito ou alvo, previamente medido, para correção da classificação de cada padrão apresentado à rede durante seu treinamento (HAYKIN, 2008).

Por outro lado, define-se também uma arquitetura de rede pelo arranjo de elementos básicos ou parâmetros arquitetônicos (PAs) que a constituem. Pormenorizadamente, neurônios artificiais, quando dispostos em camadas interligadas subsequentemente, cunham o significado do termo *Perceptron* Multi-Camadas (*Multi-Layer Perceptron* – MLP), atuando como unidades

básicas reconhecedoras de padrões e havendo a possibilidade de se empregar qualquer número destas camadas com diferentes quantidades de neurônios em cada uma. Tais camadas são progressivas, ou seja, cada uma sofre estímulos apenas dos neurônios da camada anterior e seus próprios neurônios somente irão estimular os da próxima, jamais incorrendo em recursão, como é mostrado na figura 1.

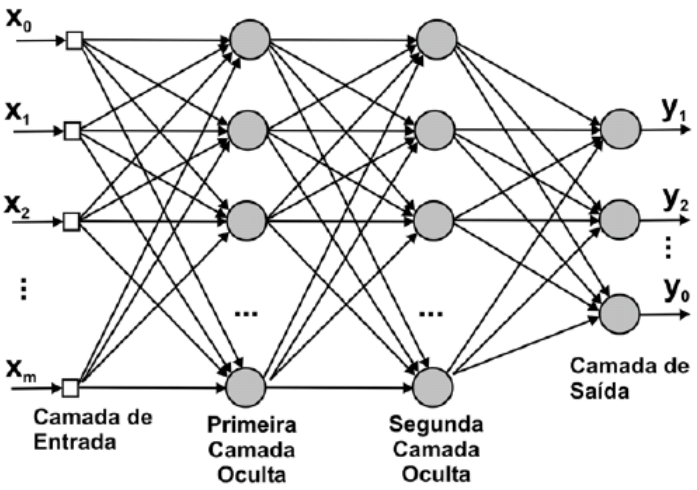

Figura 1 – Ilustração esquematica de uma Rede Neural Artificial.

Adicionalmente, leva-se em consideração a forma com que cada camada transfere informação à adjacente, isto é, pode-se ter diferentes Funções de Transferência (FTF) entre as camadas. E, por fim, há de se pensar no algoritmo utilizado para o ajuste dos pesos sinápticos de cada neurônio, que se aplica à rede como um todo e é dito Função de Treinamento (FTR) (HAGAN; DEMUTH; BEALE, 1996).

Logo, esta pluralidade de arranjos de PAs é tão demasiada, que a reflexão sobre o desempenho que redes de uma arquitetura em particular apresentarão em função destes, não é assunto trivial. Ainda hoje, são pouco claras as relações entre o número e os tipos de variáveis de um dado problema e os fatores que determinam o conjunto das melhores arquiteturas candidatas a que se lhe impõe, fato que confere um caráter heurístico à sua confecção, baseando-a, frequentemente, em "tentativa e erro" (WALZACK; CERPA, 1999).

Não raro, lê-se que, dado um problema de predição, utiliza-se confeccionar três ou quatro arquiteturas de rede, por vezes bastante discrepantes entre si, mas que apresentam desempenhos ou taxas de acerto muito similares. Contraditoriamente, redes arquitetônicamente parecidas, que diferem apenas com a presença de um único neurônio extra em uma das camadas, podem apresentar desempenhos extremamente adversos.

É, portanto, grande o interesse na implentação de um ferramental capaz de automatizar o processo pesquisa e confecção de RNAs ótimizadas à modelagem dos dados de um problema em questão. Eis o objetivo principal da plataforma aqui discutida.

1.3.1.2 Algoritmos Genéticos

Algoritmos Genéticos (AGs) perfazem um conjunto de algoritmos iteravivos que, diante de um dado problema, convergem para uma solução mais adequada, simulando desta forma um processo evolutivo de seleção natural darwiniana destas soluções. A figura 2 ilustra as estapas desempenhadas por um AG.

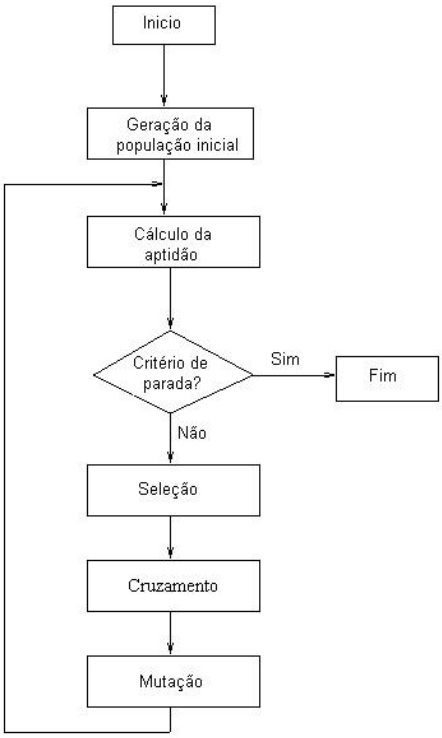

Figura 2 – Fluxograma ilustrativo de um Algoritmo Genético.

Sua aplicação extende-se amplamente a problemas tão plurais quanto as RNAs, porém têm um custo computacional bem menor, fato que os tornam propícios para o calculo imediato de modelos mais simples ou para serem combinados com as RNAs na pesquisa de arquiteturas de redes otimizadas, por exemplo. Combinações deste tipo podem convergir para resultados bastante profícuos, como mostram Filho e Poppi (1998).

1.3.1.3 Modelos estatísticos multivariados

A confecção de modelos estatísticos multivariados é, apesar de complexa, bastante rápida e pouco custosa computacionalmente se comparada à de RNAs (RENCHER; CHRISTENSEN, 2012). Ela se torna ainda mais prática quando utilizados os softwares adequados disponíveis para tal fim, como a linguagem estatística R proposta pelo movimento The R Project (2017b) e a plataforma computacional Matlab® desenvolvida pela empresa Mathworks (2017), ambas ferramentas conssagradas no exercício da computação técnica e científica.

Ademais, diferentemente das RNAs e dos AGs, bons modelos estatísticos tradicionais dificilmente se adequam a dados adversos sem a necessidade de os tratar previamente ou, quando

o fazem, não apresentam boa capacidade de generalização frente a outros dados que não os utilizados no ato da confecção do modelo. Ou seja, a robustez de modelos estatísticos que não envolvem aprendizagem de máquina é mais restrita (SAMARASINGHE, 2007).

A despeito de limitações, modelos estatísticos multivariados encontram, no escopo da plataforma, o papel instrumental de validação dos modelos obtidos por aprendizagem, estabelecendo uma base de comparação para o ajuste ou adequação daqueles.

1.3.2 *Problemas abordados*

Duas aplicações distintas e bem sucedidas de modelos envolvendo RNAs foram escolhidas dentre diversos problemas pesquisados no Laboratório de Matemática Aplicada da UNESP do campus de Assis entre 2012 e 1016, afim de se comparar os resultados publicados em cada caso com os resultados apresentados pelo modelo correspondente gerado automaticamente pela plataforma.

A primeira aplicação fora proposta por Rocha et al. (2017) e discorre sobre a importância de se antever impactos antropogênicos em ambientes lóticos, como rios e riachos. Nela, vinte fatores ambientais (temperatura da água, turbidez, tipo de leito, concentração de oxigênio na água, entre outros) foram considerados preditores relacionados à Riqueza e à Abundância de Macroalgas do local, ambas variáveis de resposta indicativas da qualidade daquele ecossistema. Para tanto, mensurou-se essas variáveis por coleta e análise tradicionais, resultando em vinte e duas variáveis contínuas de oitenta e nove pontos de um rio característico da região Sul. Estes fatores preditores e os respectivos valores Riqueza Abundância de Macroalgas permitiram ajustar um modelo de RNA supervisionada de boa acurácia. E, de posse deste modelo, simulações de impactos ambientais decorrentes da alteração dos fatores preditores tornam-se mais práticas.

Assim, no teste da plataforma, pôs-se-lhe a criar um modelo correspondente, sobre aqueles mesmos dados coletados. O desempenho e a arquitetura, tanto da RNA sugerida pela plataforma, quanto da RNA descrita na literatura, são comparados na seção 4.2. O desempenho sensivelmente inferior do modelo sugerido e sua arquitetura discrepante em relação aos da literatura são funções do menor número de modelos distintos criados pela plataforma durante a pesquisa devido à imposição de uma limitação no tempo de execusão em cada teste.

A segunda aplicação é descrita por Alves (2015) e propõe uma forma prática e de baixo custo para análise da qualidade de água em domínios urbanos, baseada em medidas espectrofotometricas associadas a uma RNA. O levantamento dos dados consistiu em colher, no ano de 2015, amostras de água em cem pontos igualmente distribuídos entre dois rios que atravessam a área urbana da cidade de Assis. Objetivou-se inferir quantitativamente a influência da urbanização sobre a qualidade da água e, deste modo, evidenciar eventuais pontos críticos ou de contaminação em seus percursos.

Semelhante à aplicação anterior, ambos os modelos são comparados na seção 4.3, sendo que, novamente, a plataforma apresentou maior rapidez na confecção de modelos em detrimento de seus desempenhos, ressaltando o caráter estocástico do processo de pesquisa de arquiteturas de RNAs.

1.3.3 *Interveniências gerais e segurança dos dados*

Certamente o fator de maior preocupação presente na implementação técnica da plataforma diz respeito à sua infraestrutura e a forma como manipula os dados de outrém. Dado que sua proposta de implementação e distribuição são ambas livres, segundo a licença de uso descrita na seção 1.5, faz-se uma ressalva quanto às garantias do seu uso. Ocorre que o usuário tem liberdade para alterar o código fonte e reconfigurá-lo, reclamando para si a responsabilidade pela manipulação dos dados por ele processados em servidores ou computadores pessoais.

Relativamente aos dados processados pela interface *on-line* no site da plataforma, apenas o desenvolvedor tem acesso ao código fonte e aos processos em trâmite por conexões seguras de linha de comando via SSH.

E, quanto à submissão ordinária de dados de projetos, usuários podem acessar a *home page* da plataforma[1] livremente via protocolo http, com a garantia do sigilo no retorno de seus resultados no e-mail especificado.

1.4 **Integração e expansão *on-line***

1.4.1 *Desenvolvimento colaborativo e suporte*

O modo como o sofware está disponível e é distribuído, encoraja usuários a compartilharem sua experiência em um ambiente integrado de desenvolvimento, o que permitirá o crescimento do número de módulos, bem como de suas aplicações. A título de exemplo, notou-se, ainda durante a escrita deste trabalho, a auxência de um módulo voltado à confecção automática de desenhos técnicos. Sendo que sua disponibilidade encontra aplicação imediata na indústria, justamente após ocasiões que demandam longas e sucessivas bateladas de simulações envolvidas numa análise de ampliação de escala em bioprocessos industriais, ao desenhar o equipamento envolvido a partir das dimensões calculadas pelo modelo.

Adicionalmente, ambientes colaborativos como o proposto, contribuem para a manutenção e atualização do código fonte do *software* a medida que mais colaboradores reportam erros, sugerem soluções e criam novos módulos. Exemplos são os trabalhos das comunidades de suporte à linguagens de programação LaTeX e R, entre outras (TEX USERS GROUP, 2017; THE R PROJECT, 2017a).

Tem-se em mente, ainda, que a expansão do uso da plataforma, eventualmente exigirá a implementação de bancos de dados relacionais maiores para o manejo do volume crescente de dados gerados nos resultados.

1.4.2 *Movimento* Open Source *e visibilidade*

A proposta segue a filosofia do movimento *Open Source* proposto a partir 1998 pelo grupo de desenvolvedores de *software* que compõem a Fundação para *Softwares* Livres GNU (Free Software Foundation (2017a)), em prol da disseminação de softwares de código aberto para

[1] O endereço eletrônico desta página encontra-se na seção 3.1.2.

universalização e livre distribuição da tecnologia e do conhecimento. Segundo ela, a produção eficiente de um *software* ocorre quando do livre reuso deste, não importando a área de aplicação, plataforma de instalação ou cunho comercial.

1.4.3 *Integração a outras plataformas*

O fato de se padronizar o formato dos resultados de processos programáticos, permite uma fácil exportação dos dados para outros programas ou plataformas, podendo gerar cadeias de comandos sucessivos conhecidas como *pipe lines*. Estas cadeias de comandos perfazem o cerne da reprodutibilidade técnica atual e, por conseguinte, da universalização do acesso aos resultados, de forma que diferentes profissionais podem se beneficiar de informações de outras áreas, outrora inacessíveis, devido ao *design* comum empregado nas várias estapas de uma *pipe line*.

1.5 **Perspectivas**

É de intuito maior que a paltaforma aqui apresentada se faça útil a comunidade em geral, seja com finalidade acadêmica no tratamentos de dados, seja em qualquer outra vertente, ainda que comercial, de qualquer uma de suas funções (como a confecção automática de interfaces ou relatórios). E, para tanto, sua licença de uso é aberta, abrangendo a definição GPLv3 proposta pela fundação internacional GNU (FREE SOFTWARE FOUNDATION, 2017b).

Acredita-se que a divulgação gradativa do uso da plataforma, bem como seu fácil acesso, possam incorrer numa maior visibilidade das atividades desenvolvidas pelos pesquisadores que a utilizam, bem como numa maior interação entre estes, como idealizado pelo uso da licença GPLv3 (FREE SOFTWARE FOUNDATION, 2017c).

2 OBJETIVOS

- Implementar uma plataforma computacional, tanto *on-line* quanto localmente instalável, para submissão de dados visando a automação dos processos de pesquisa e simulação de modelos numéricos adequados àqueles dados.
- Criar uma biblioteca de algoritmos e funções que dê suporte às rotinas de pesquisa supracitadas e que, adicionalmente, permita sua integração com outros algoritmos eventualmente fornecidos pelo usuário, provendo modelos programaveis sob demanda.
- Retornar ao usuario os arquivos intermediários gerados, bem como relatórios estatísticos automaticos descritivos dos modelos submetidos e de suas simulações.

3 MATERIAIS E MÉTODOS

3.1 Implementação técnica da plataforma

Utilizou-se a plataforma computacional Matlab®, por dispor de uma linguagem de programação própria sobre a qual se tem desenvolvida uma vasta biblioteca de funções voltadas à criação, treinamento, validação, teste, simulação, análise de desempenhos e muitas outras rotinas, específicas para o implemento de RNAs, reunidas em um pacote de algoritmos denominado *Machine Learning Toolbox*.

A partir do Matlab®, a plataforma *neurocelle* fora implementada como um programa autômomo e passível de instalação local, disponibilizada livremente em repositórios *on-line*, para diversos sistemas operacionais. Os repositórios registram todo o histórico de desenvolvimento das versões da plataforma através do mecânismo de controle de versões Git. sendo que a versão corrente da plataforma é marcada com a *tag* a v0.1.

3.1.1 Download *e instalação*

O usuário interessado pode fazer o *download* do *software* da plataforma, bem como obter informações pertinentes (como o código fonte e todo o histórico de desenvolvimento do projeto) segundo três formas principais, a saber:

- Acessando o repositório principal do projeto *neurocelle* disponível no endereço eletrônico <http://www.neurocelle.org>, na guia “Downloads”.
- Acessando o portal *GitHub* para projetos *Open Source* no endereço <https://github.com/estrogonelda/neurocelle>.
- Copiando qualquer repositório *neurocelle*, ou através do terminal de linha de comando do próprio sistema operacional, desde que já tenha instalado o Git, usando o comando:
  `$ git clone https://github.com/estrogonelda/neurocelle.git`

Depois de feito o *download*, o usuário tem em seu computador um diretório que é um repositório *neurocelle* completo, ou seja, capaz de se auto instalar, auto reparar e auto copiar, entre outras funcionalidades. Então, para terminar a instalação, pelo terminal de linha de comando deve-se digitar:

```
$ cd neurocelle
$ .configure
$ make install
```

Por fim, a instalação está completa! Evetuais erros de instalação podem ser evitados ou solucionados pelo usuário consultando-se o manual de instalação disponível em <http://www.neurocelle.org/documentation.html>, ou pelo terminal com o comando:

```
$ man ncl
```

3.1.2 *Interfaces e meios de acesso ao servidor central*

Foram propostas duas categorias de interfaces para a plataforma *neurocelle*: interfaces livres locais e interfaces restritas remotas. Sendo que as interfaces passíveis de instalação local pelo usuário subdividem-se em três tipos e a remota, relativa a um servidor central, é única. Assim como os módulos, não foram implementadas todas as interfaces subescritas, porém são mostradas por estarem referenciadas internamente por diversas funções, tendo assim influenciado o *design* da arquitetura do projeto. Eis as interfaces livres:

- **Interface de Linha de Comando (*Command Line Interface* - CLI):** Uitilizada através do terminal, ela é simples porém poderosa, permitindo total controle dos argumentos de entrada e saída a cada comando. Ela, no entanto, requer que se saiba uma sintaxe básica de comandos, os quais permitem a criação de *scripts* a serem executados por outros programas ou até mesmo agendados para execução automática sem que seja necessária a presença do usuário. Sua sintaxe segue o padrão:

```
$ ncl [comando] [arquivos_de_entrada] [opções]
```

- **Interface Gráfica**[1] **(*Graphic User Interface* - GUI):** Utilizada por meio de uma janela gráfica do sistema operacional do usuário, ela não requer que se saiba os comandos, porém exige a interação com o usuário. Nela, cada interação provoca uma chamada à CLI no terminal, para que seja executado o comando correspondente. A figura 3 mostra uma possível configuração da janela gerada. Inicia-se uma sessão gráfica com o comando:

```
$ ncl [arquivos_de_entrada] -g
```

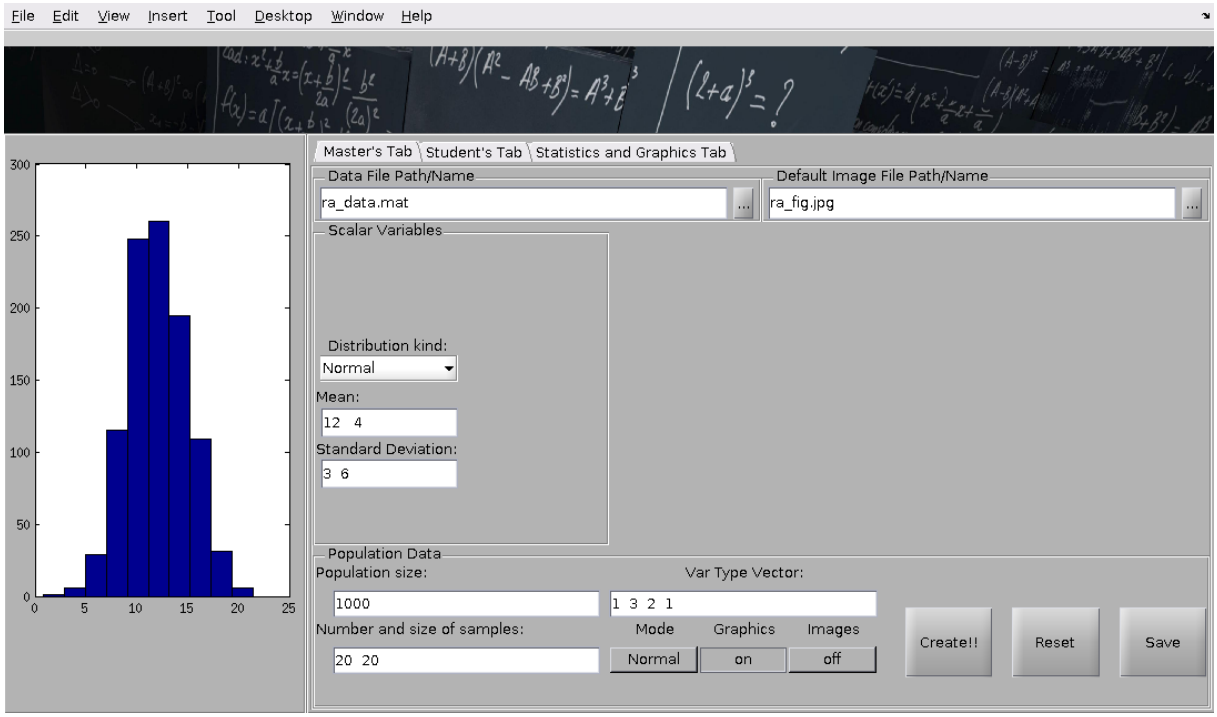

Figura 3 – Interface Gráfica da plataforma *neurocelle*.

[1] Não implementada.

- **Interface de Planilha de Dados**[2] **(*Spreadsheet Interface* - SI):** Semelhante a GUI mas com os dados dispostos em uma planilha em formato `.xls`, `.xlsx` ou `.ods`, onde sua execução ocorre por meio de macros criadas para invocar o comando correspondente na CLI. É uma interface simples e ao mesmo tempo visual já integrada a outros *softwares* como o *Excel* da Microsoft® ou o *Calc* da iniciativa LibreOffice®. A figura 4 também mostra uma possível configuração de planilha gerada.

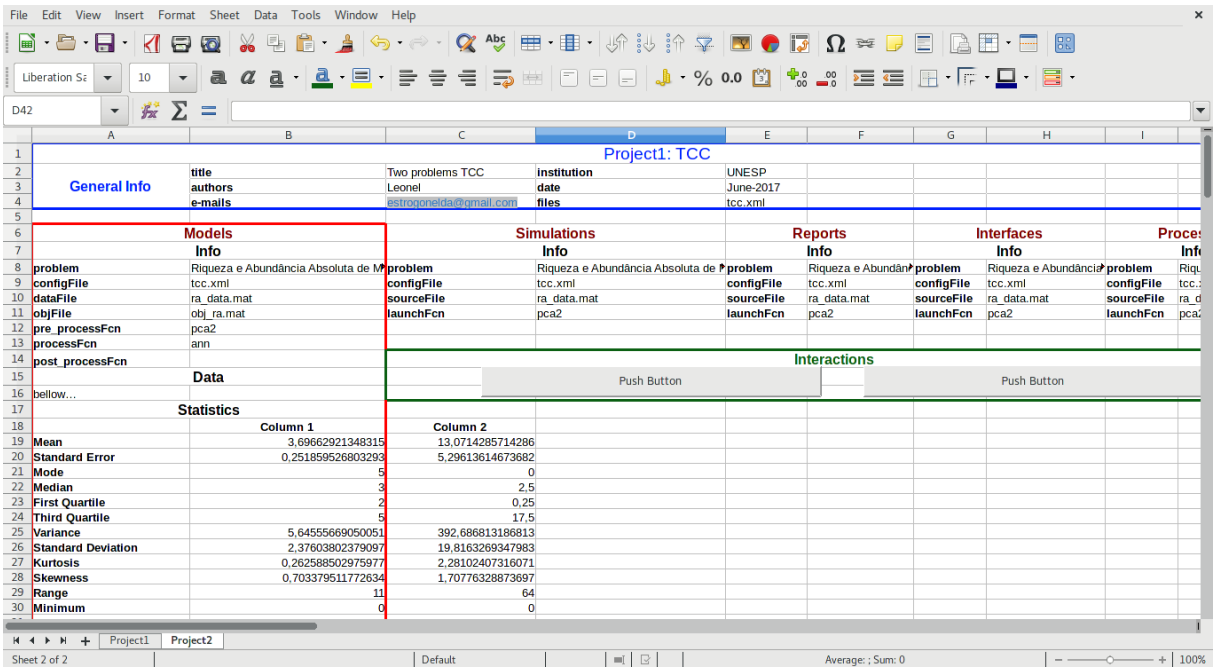

Figura 4 – Interface de Planilha de Dados da plataforma *neurocelle*.

[2] Não implementada.

Já a interface restrita, por outro lado, correponde ao próprio site do projeto, por meio do qual se pode ter acesso ao servidor central por conexões SSH. Ela é dita restrita porque o *upload* de arquivos necessita de um *login* de identificação do usuário, que precisa estar cadastrado. A figura 5 ilustra a *home page* do projeto *neurocelle* e seu mecanismo de *upload* de arquivos.

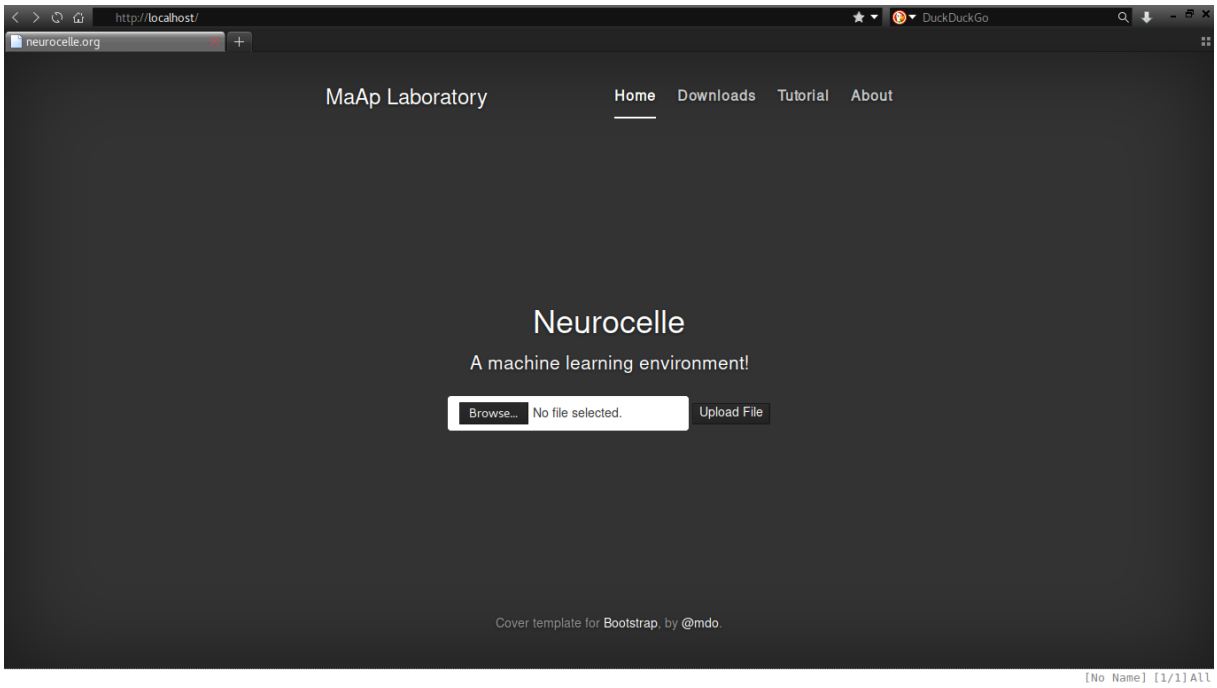

Figura 5 – *Home page* da plataforma *neurocelle* disponível em <http://www.neurocelle.org/>.

Sem embargo à segurança exigida pelo servidor central, a instalação local da plataforma permite a um usuário a configuração de um servidor pessoal para acesso remoto, sob sua responsabilidade, semelhante ao referido servidor central e que pode ser igualmente acessado por uma conexão SSH.

3.1.3 *Objeto principal e Módulos*

O *software* foi implementado segundo o paradigma de programação orientada a objeto que define blocos de código como abstrações de objetos em uso, ou seja, definem um número finito de propriedades (ou variáveis) e métodos (ou funções) para uma finalidade específica e internamente consistente (como criar um modelo de regressão ou treinar uma RNA). Desta forma, introduz-se o termo "encapsulamento", no qual objetos desempenham suas funções isoladamente e, quando em conjunto, perfazem uma hierarquia sistêmica. Isto garante que o funcionamento geral do *software* não venha a ser afetado pelo eventual mal funcionamento de um módulo específico, além de estabelecer uma convenção da organização dos dados e dos argumentos de entrada e saída das funções.

A plataforma *neurocelle* em si é um objeto da classe *anima* cujas funções restringem-se à instalação geral do diretório de base no sistema, eventuais reparos a este diretório, controle de processos, mapeamento de erros, validação de usuários e inicialização dos módulos. Adicionalmente, ela foi planejada para ser modular e facilmente customizável, de tal forma que os vá-

rios módulos são todos objetos derivados de um mesmo *template* de uma superclasse chamada *modulo*. Este procedimento de derivação é chamado de "herança" daquela classe e, consequentemente, faz com que todos estes módulos compartilhem diversos atributos e metodos como, por exemplo, o caminho referente aos diretórios de base e do usuário em questão, entre outros dados.

Portanto, no prosseguimento, reservar-se-á exclusivamente à descrição dos algoritmos e processos distintivos de cada módulo.

3.1.3.1 Módulo *progetto*

Este é o módulo de escopo central dentro da superclasse *neurocelle* e, com exceção do módulo *sinergio*, pode comportar todos os outros módulos como atributos seus. É o único módulo cujo acesso aos dados é permitido ao usuário e perfaz todas as rotinas descritas para um projeto por meio de um arquivo de configuração correspondente.

É possivel que um mesmo usuário opere mais de um projeto simultâneamente e, que em cada projeto, haja diversos modelos, simulações, interfaces e saídas de documentos como submódulos. Neste caso, um *array* destes objetos será criado para um processamento sequencial ou paralelo, dependendo das instruções contidas em seu arquivo de configuração.

3.1.3.2 Módulo *sinergio* (não implementado)

Módulo responsável pelo controle dos processos e da memória do sistema operacional e é inacessível pelo usuário. É o único módulo que é não está subordinado a um módulo *progetto*, antes, ele é um atributo da superclasse *neurocelle*, estando no mesmo nível que os módulos *progetto*. Assim, este módulo é que faz o intermédio entre a requisição do usuário e as chamadas dos processos relativos aos projetos, se houver memória suficiente para isto.

3.1.3.3 Módulo *astratto*

Módulo responsável pela confecção de modelos a partir dos dados de entrada fornecidos pelo usuário. A versão v0.1 possui implementação para modelos de regressão multivariada (paramétrica) e de RNAs supervisionadas[3], segundo três *designs* experimentais de pesquisa e otimização: randômico, exaustivo e por algoritmo genético simples (embora outros possam ser referidos pelo usuário).

Todos os modelos são automaticamente criados por meio de uma biblioteca de funções referida internamente como *libml* desenvolvida pelo autor. Esta biblioteca pode ser usada independentemente desta plataforma, no entanto, recomenda-se seu uso concomitante, dado o elevado número de parâmetros utilizados que são facilmente editáveis pelo arquivo de configuração do projeto.

[3] O usuário pode desenvolver *designs* customizados para serem utilizados pelo módulo *astratto*. Estes novos *designs* podem incluir regressões multivariadas não paramétricas, RNAs não supervisionadas e modelos por lógica *fuzzy*.

3.1.3.4 Módulo *vero* (não implementado)

Módulo responsável pela confecção automática de arquivos de simulação para modelos *astratto*. Sua saída consiste em gerar, programaticamente, arquivos `.mdl` a serem simulados pela ferramenta Simulink® que acompanha o *software* Matlab. A descrição da interação entre os diversos modelos de um mesmo projeto deve constar no arquivo de configuração de entrada.

3.1.3.5 Módulo *parlo* (não implementado)

Módulo responsável pela criação de documentos de saída resultantes da análise dos modelos e suas simulações, sejam eles relatórios, gráficos ou tabelas nos formatos `.pdf`, `.tex` ou `html`.

3.1.3.6 Módulo *viso* (não implementado)

Módulo responsável pela confecção de interfaces específicas para instalação dos modelos gerados. Elas podem conter, segundo o arquivo de configuração do projeto e os dados de entrada, abas para simulação, gráficos e exportação de documentos do módulo *parlo*. Os possíveis arquivos de saída são no formato `.m` ou `.exe` para instalação.

3.2 **Testes procedimentais**

O passo inicial para todos os procedimentos de confecção de modelos, simulações, interfaces e documentos, é a edição de um ou mais arquivos de configuração. Este tipo de arquivo (no formato `.xml`) permite a descrição de uma grande quantidade de parâmetros, os quais, em absencia, são supridos por valores padrão. Ou seja, o usuário não precisa necessariamente fornecer todos os parâmetros, apenas os que deseja. E, para obter um *template* deste arquivo de configuração com uma lista exaustiva destes parâmetros e seus respectivos possíveis valores, basta introduzir o seguinte comando no terminal:

```
$ ncl
```

Assim, um novo arquivo surgirá no diretório local, pronto para ser editado. O anexo 1 apresenta sua estrutura básica.

3.2.1 *Carregamento de dados e inicialização dos módulos*

Quando especificado um arquivo com autorização para leitura existente em qualquer local da árvore de diretórios, inicializam-se os módulos e ocorre seu carregamento, seguido da validação dos valores especificados pelo usuário em cada campo. Valores faltantes, inexistentes ou mal escritos são substituidos por valores padrão. O carregamento de um arquivo hipotético chamado `file1.xml` localizado no diretório `dir1/dir2` dá-se pelo comando:

```
$ ncl dir1/dir2/file1.xml
```

Posto desta forma, após a inicialização dos módulos, todos os procedimentos possíveis indicados no arquivo serão executados e eventuais erros serão registrados em um arquivo de log no diretório base.

O primeiro módulo a ser inicializado é o *sinergio*, para que sejam chamados os outros processos. Logo em seguida, pelo menos um módulo *progetto* é carregado com seus submódulos dentro como atributos. Passa-se então a analizar os dados para confecção de pelo menos um modelo, através do módulo *astratto*.

3.2.2 *Confecção procedural de modelos*

O arquivo de configuração deve fazer referência (através do campo `datafile` dentro das *tags* `<info>...</info>`) a outro arquivo onde se encontram os dados experimentais a serem modelados. Eis que, se este arquivo de dados não estiver vazio e for legível, inicia-se o processo de análise sensível multivariada, com pré tratamento, modelagem e pós tratamento dos dados, como segue:

1. Detecção da presença de matriz de alvos (*targets*) para decisão entre confecção de modelo supervisionado ou não supervisionado.
2. Teste do normalidade multivariada, comumente feita apenas sobre a matriz de entradas, mas que pode ser extendida à matriz de alvos.
3. Detecção e remoção da *outliers*.
4. Normalização, padronização e redução de dimensionalidade nos dados.
5. Modelagem em si: regressão multivariada ou RNAs.
6. *Designs* — comparações e seleção entre modelos.
7. Retorno de dados ao usuário.

3.2.2.1 Testes de normalidade multivariada e detecção de *outliers*

Testes de detecção de normalidade em dados multivariados apresentam-se consideravelmente mais complexos do que aqueles comumente aplicados a dados univariados como o teste de Shapiro-Wilk. Ocorre que, com mais de uma variável, não é possível dividir o espaço amostral em setores ordenados (como percentis) para proceder com um teste de aderência. Some-se a isto o fato de que a verificação de normalidade univariada, feito nas variáveis separadamente, não implica em normalidade conjunta.

Assim, a título de ilustração dos conceitos presentes na explanação teórica dos testes implementados, lança-se mão dos dados encontrados da tabela 1 retirada de Elston e Grizzle (1962 apud RENCHER; CHRISTENSEN, 2012, p. 87).

Tabela 1 – Comprimento femural de 20 garotos em 4 idades distintas.

Indivíduo	**Idade (anos)**			
	8 (y1)	8.5 (y2)	9 (y3)	9.5 (y4)
1	47.8	48.8	49.0	49.7
2	46.4	47.3	47.7	48.4
3	46.3	46.8	47.8	48.5
4	45.1	45.3	46.1	47.2
5	47.6	48.5	48.9	49.3
6	52.5	53.2	53.3	53.7
7	51.2	53.0	54.3	54.5
8	49.8	50.0	50.3	52.7
9	48.1	50.8	52.3	54.4
10	45.0	47.0	47.3	48.3
11	51.2	51.4	51.6	51.9
12	48.5	49.2	53.0	55.5
13	52.1	52.8	53.7	55.0
14	48.2	48.9	49.3	49.8
15	49.6	50.4	51.2	51.8
16	50.7	51.7	52.7	53.3
17	47.2	47.7	48.4	49.5
18	53.3	54.6	55.1	55.3
19	46.2	47.5	48.1	48.4
20	46.3	47.6	51.3	51.8

Ademais, dois testes de hipótese sobre normalidade multivariada foram implementados na biblioteca *libml* que a plataforma faz uso.

O primeiro teste, proposto por Gnanadesikan e Kettenring (1972 apud RENCHER; CHRISTENSEN, 2012, p. 106), baseia-se nas distâncias padronizadas entre cada vetor p-variado y_i e a média amostral $\bar{y}$, relativamente à matriz de covariancias amostral S, como na equação 3.1.

$$D_i^2 = (y_i - \bar{y})' S^{-1} (y_i - \bar{y}), i = 1, 2, \ldots, n. \tag{3.1}$$

Segundo ele, se y_i apresentar distribuição normal p-variada, então u_i, na equação 3.2, apresenta uma distribuição Beta com parâmetros a e b, cujos valores podem ser obtido integrando-se sua função de densidade de probabilidade dada pela equação 3.3.

$$u_i = \frac{nD_i^2}{(n-1)^2} \tag{3.2}$$

$$\int_0^{v_i} \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} x^{a-1}(1-x)^{b-1} dx = \frac{i-\alpha}{n-\alpha-\beta+1}, \tag{3.3}$$

sendo que $v_i = (i - \frac{1}{2})/n$ são os n quantis da amostra e a, b, α e β, tais que

$$a = \frac{1}{2}p, \quad b = \frac{1}{2}(n-p-1), \quad \alpha = \frac{p-2}{2p} \text{ e } \quad \beta = \frac{n-p-3}{2(n-p-1)} \tag{3.4}$$

Valores de u_i que vierem a exceder o valor crítico são prováveis *outliers* na amostra e por isso devem ser removidos dela com cautela. Este primeiro método foi escolhido por possibilitar

a constatação visual do padrão de normalidade da amostra por meio de um gráfico característico denominado Q-Qplot. Para tanto, ordenam-se os valores de u_i gerando $u_{(i)}$, tais que $u_{(1)} \leq u_{(2)} \leq \cdots \leq u_{(n)}$ e então plotam-se os pares $(u_{(i)}, v_i)$, resultando no perfil apresentado na figura 6.

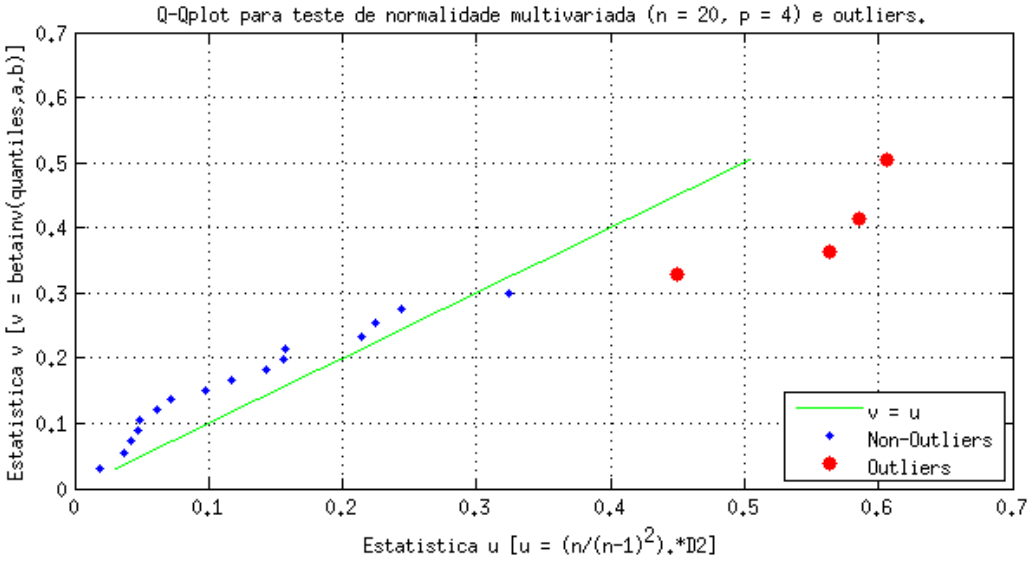

Figura 6 – Gráfico Q-Qplot dos valores de comprimento femural da tabela 1.

Nela são evidenciados quatro *outliers* relativos aos indivíduos das posições 8, 20, 12 e 9 da tabela, em ordem crescente de seus valores de u_i. Estes valores são, respectivamente, 0.4493, 0.5637, 0.5861 e 0.6067, encontrando-se significativamente distantes da reta verde.

O segundo teste não apresenta apelo visual e pode ser conduzido computacionalmente sem a interação com o usuário. Ele é atribuido a Mardia (1970 apud RENCHER; CHRISTENSEN, 2012, p. 107) e reside em mensurações de b_1 e b_2, ambos estimadores dos valores de assimetria e curtose $\beta_{1,p}$ e $\beta_{2,p}$ de uma distribuição p-variada, respectivamente:

$$\beta_{1,p} = E[(y-\mu)'\Sigma(y-\mu)]^3, \tag{3.5}$$

$$\beta_{2,p} = E[(y-\mu)'\Sigma(y-\mu)]^2 \tag{3.6}$$

E argumenta que $\beta_{1,p} = 0$ e $\beta_{2,p} = p(p+2)$ para uma distribuição $N_p(\mu, \Sigma)$, logo, os valores amostrais de b_1 e b_2 devem estar aproximos àqueles, dentro de um limite de tolerância, para que a distribuição amostral possa ser considerada normal p-variada. Consequentemente, o teste desta proximidade perfaz o teste de hipótese de normalidade desejado.

O calculo de b_1 e b_2 exige que inicialmente seja definido g_{ij}, tal que

$$g_{ij} = (y_i - \bar{y})'\hat{\Sigma}(y_j - \bar{y}), \tag{3.7}$$

onde $\hat{\Sigma} = \sum_{i=1}^{n}(y_i - \bar{y})(y_i - \bar{y})'/n$ é um estimador de máxima verossimilhança populacional. Então, os estimadores de $\beta_{1,p}$ e $\beta_{2,p}$ podem ser obtidos por

$$b_1 = \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} g_{ij}^3 \quad \text{e} \tag{3.8}$$

$$b_2 = \frac{1}{n}\sum_{i=1}^{n} g_{ij}^2. \tag{3.9}$$

Mardia, em seu trabalho, apresenta tabelas para valores críticos de b_1 e b_2 para $p = 1, 2, 3$ e 4, porém quando $p > 4$ e $n \geq 50$, um método genérico de aproximações é recomendado. Assim, definem-se z_1 e z_2 respectivamente em função de b_1 e b_2, como segue

$$z_1 = \frac{(p+1)(n+1)(n+3)}{6[(n+1)(p+1)-6]} b_1 \quad \text{e} \tag{3.10}$$

$$z_2 = \frac{b_2 - p(p+2)}{\sqrt{8p(p+2)/n}}. \tag{3.11}$$

Aqui, z_1 apresenta distribuição χ^2 com $\frac{1}{6}p(p+1)(p+2)$ graus de liberdade e z_2, uma distribuição $N(0,1)$. A hipótese de normalidade p-variada deve ser rejeitada para $z_1 > \chi^2_{0.05}$ e para valores extremos de z_2 em um teste bicaudal padronizado. Sugere-se o teste de z_2 como bicaudal pois, um baixo valor indica uma distribuição de platicúrtica (achatada) enquanto um alto valor indica uma distribuição leptocúrtica (esguia).

Aplicando-se este último teste aos dados da tabela 1, obtem-se $b_1 = 11.338$ e $b_2 = 28.884$, sendo que seus valores críticos são, respectivamente, 9.9 e 27.1, o que é justificado pela presença dos *outliers* evidenciados na figura 6.

Não obstante, ambos os testes apresentados são realizados simultâneamente pela plataforma, segundo a opção do usuário. No entanto, o feitio do gráfico Q-Qplot deve ser requisitado explicitamente no arquivo de configuração por meio do campo `plots` na *tag* `<reports>...</reports>`. E, tendo-se optado pela detecção de *outliers*, sua remoção é automática.

3.2.2.2 Normalização, padronização e redução de dimensionalidade dos dados

Designa-se por normalização, o conjunto de técnicas que promovem uma transformação sobre os dados, colocando todas as variáveis sob numa mesma escala e uma mesma amplitude de dispersão. Porém nem todas estas técnicas preservam a estrutura de correlações presentes nos dados originais. Por isso, nos casos em que há grande contraste entre as escalas de pelo menos duas variáveis na amostra, valores extremos acabam distorcendo a representação das outras variáveis. Por exemplo, em duas variaveis, a saber, concentração de metabólitos em uma dada solução e número de células viáveis nela, encontram-se valores, respectivamante, das ordens de 10^3 mols/L e 10^6 celulas/L, diferindo numa proporção de $10^9 : 1$. Normalizá-las numa faixa entre -1 e 1 acarretaria uma valoração subestimada da variância da menor variável visto que sua aplitude de dispersão é ínfima comparada a da outra, que varia em um escala desproporcionalmente maior. Por isso, faz-se necessário o uso parcimonioso das técnicas em função das diferenças entre as escalas apresentadas por cada variável da amostra. Há de se preferir técnicas que preservem a estrutura de correlações entre as variáveis.

Outrossim, a presença de multicolinearidade entre as variáveis acarreta a necessidade de ajuste do número de parâmetros envolvidos num modelo (os pesos sinápticos nas RNAs, por

exemplo) dentro de um subespaço de dimensionalidade reduzida, ou seja, muitos pontos podem estar dispostos coplanarmente no espaço p-variado. Isso faz com que alguns autovalores das matrizes de covariâncias ou correlações amostrais tendam a zero, indicando as variáveis redundantes que podem aumentar os erros do modelo devido à presença de parâmetros sobresalhentes. Eis a importância da detecção de multicolinearidade nos dados e da consequente redução da dimensionalidade — reduzir a complexidade dos modelo (JOHNSON, 2007).

Neste ínterim, foram implementadas três técnicas que podem ser escolhidas pelo usuário para o pré tratamento de seus dados, segundo Samarasinghe (2007, cap. 6).

- **Normalização por faixa simples:** Normalizam-se separadamente as variáveis de entrada para que se restrinjam a uma faixa de valores, proporcionalmente a suas posições dentro da amplitude da distribuição original. Logo, para se restringir uma distribuição entre os valores H e L, utiliza-se a equação 3.12.

$$x_{i(T)} = L + (H - L)\left(\frac{x_i - x_{min}}{x_{max} - x_{min}}\right), \tag{3.12}$$

  sendo que $x_{i(T)}$ é o novo valor transformado de x_i e x_{max} e x_{min} são os limites da amplitude original. Nesta técnica a estrutura de correlação entre as variáveis não é preservada, além ocultar variáveis de escalas menores quando normalizadas conjuntamente. Além disso, notou-se que, o Matlab realiza este processo automaticamente ao treinar RNAs.

- **Padronização:** As variáveis são redimensionadas de acordo com seus respectivos desvios padrão ao redor de suas médias, como na equação 3.13.

$$z_i = \frac{x_i - \bar{x}}{\sigma}, \tag{3.13}$$

  sendo z_i o valor padronizado de x_i e $\bar{x}$ e σ, média e desvio padrão amostrais, nesta ordem.

- **Análise de Componentes Principais (PCA) padronizados:** Esta técnica permite, tanto a padronização conjunta das variáveis (por meio da matriz de covariâncias ou da matriz de correlações amostral), quanto uma transformação linear que elimine a redundância na multicolinearidade entre elas.

  Assim, seja y uma amostra p-variada com n elementos e A, a matriz dos autovetores relativos à sua matriz de correlações (ou matriz de covariâncias) amostral. Tem-se que $z = A_k' y$ é a matriz de escores dos n y_i, segundo os k primeiros componentes principais da amostra. Neste caso, o novo conjunto dos z_i explicaria uma porcentagem da variância da amostra original dada por

$$P = \frac{\sum_{j=1}^{k} \lambda_j}{\sum_{i=1}^{n} \lambda_i}, \tag{3.14}$$

  com λ_i sendo os autovalores da matriz de correlações (ou de covariâncias) amostral.

  Aplicando a técnica de PCA sobre os dados da tabela 1, redimensionam-se os escores originais das coordenadas de cada indivíduo para que se eliminem as correlações entre

as variáveis, definindo novos eixos cartesianos para elas. Eis a matriz PC com as quatro componentes principais (autovetores) calculadas sobre a matriz de correlações amostral, juntamente com seus autovalores $Avals$ associados:

$$PC = \begin{pmatrix} 0.4937 & 0.5854 & 0.5671 & -0.3035 \\ 0.5066 & 0.3810 & -0.5358 & 0.5578 \\ 0.5089 & -0.3399 & -0.4428 & -0.6553 \\ 0.4906 & -0.6298 & 0.4419 & 0.4090 \end{pmatrix} \quad Avals = \begin{pmatrix} 3.6961 \\ 0.2551 \\ 0.0321 \\ 0.0168 \end{pmatrix}$$

A explicação da variância total original promovida por esta transformação linear dos eixos cresce com o número de componentes principais considerados (sendo aqui, no máximo, 4). O gráfico tipo screeplot na figura 7 apresenta a contribuição de cada componente para a explicação da variância total da tabela 1. Nele, pode-se notar que os dois primeiros componentes principais já são suficientes para explicar aproximadamente 98% da variância original dos dados.

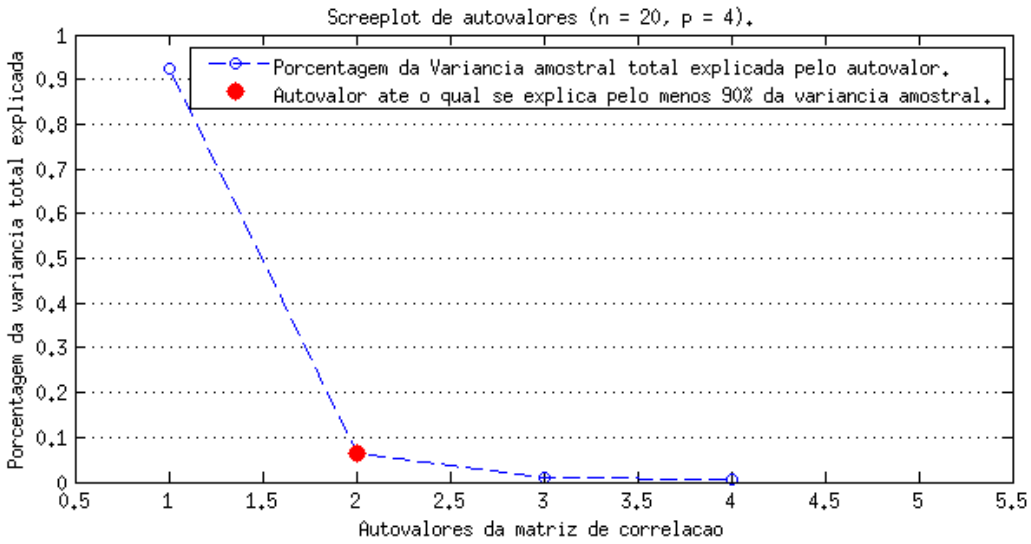

Figura 7 – Gráfico screeplot dos Autovalores da tabela 1.

Este fato acarreta uma diminuição do número de variáveis, e assim, da dimensionalidade do problema, fazendo com que se represente um mesmo indivíduo com um número menor de coordenadas. Por exemplo, o indivíduo 1, de coordenadas $(47.8, 48.8, 49.0, 49.7)$, passsou a ser representado como $(-0.95, 0.28)$. Logo, estas transformações não causam grande comprometimento da informação contida nos dados originais, ao passo que reduzem a complexidade de sua representação.

3.2.2.3 Modelagens

O processo de modelagem em si ocorre segundo a demanda do usuário especificada no arquivo de configuração. A versão v0.1 da plataforma disponibiliza dois tipos de modelos:

- **Regressão Multipla Multivariada:** Consiste em criar, através do algorítmo de mínimos quadrados, um modelo linear com o número de coeficientes (variáveis de entrada) que permita o melhor se ajuste aos dados de entrada, como na equação 3.15:

$$\hat{y_k} = \beta_{0k} + x_1\beta_{1k} + x_2\beta_{2k} + \cdots + x_q\beta_{qk} + \epsilon_k, \quad k = 1, 2, \ldots, p. \tag{3.15}$$

ou matricialmente como $\hat{Y} = XB + \epsilon$.

- **Redes Neurais Artificiais supervisionadas:** Algoritmo iterativo que consiste em ajustar os pesos de uma rede de forma a minimizar o erro médio quadrático entre todos os alvos (*targets*) e suas respectivas saídas por meio de uma retropropagação destes ajustes, segundo um gradiente descendente. Ou seja, dado o erro $E = t - y$ e os pesos sinápticos w_m de uma determinada época m do treinamento, para N padrões de treinamento, tem-se a determinação deste gradiente d_m segundo a equação 3.16.

$$d_m = \sum_{n=1}^{N} \left[\frac{\delta E}{\delta w_m}\right]_n \tag{3.16}$$

3.2.2.4 Calculo dos desempenhos

Sem dúvida, o maior desafio na confecção de modelos é aferir sua adequação aos dados. Visto que ambos os modelos que a plataforma oferece são supervisionados, ou seja, possuem uma matriz de alvos para o ajuste dos erros, técnicas estatísticas para mensuração da associação entre estes alvos e as saídas preditas se fazem necessárias. No entanto, frequentemente se tem modelos com mais de uma saída, arguindo num escopo multivariado.

É posto que o teste t de Student é o mais indicado quando se deseja aferir uma eventual diferença existente entre os escores de duas amostras univariadas que apresentem distribuições normais. Analogamente, para amostras normais multivariadas, tem-se o teste T^2 de Hotelling que sumariza as distâncias entre os vetores amostrais e suas respectivas médias, segundo as matrizes de covariâncias amostrais num espaço p-dimensional. A equação 3.17 mostra sua forma característica:

$$T^2 = (\bar{y} - \mu)' \left(\frac{S}{n}\right)^{-1} (\bar{y} - \mu). \tag{3.17}$$

O teste da hipótese nula de que a amostra foi retirada da mesma população a que se atribui a média considerada, pode ser conduzido por averiguação dos valores críticos tabelados para $T^2_{p,v}$, ou por uma transformação a valores correspondentes numa distribuição $F_{p,v-p+1}$, com $v = n - 1$ para uma única amostra testada, ou $v = n_1 + n_2 - 2$ para o teste entre duas amostras não pareadas. Os valores correspondentes de $F_{p,v-p+1}$ são obtidos pela equação 3.18

$$F_{p,v-p+1} = \frac{v - p + 1}{pv} T^2_{p,v} \tag{3.18}$$

No entanto, quando se têm duas amostras não balanceadas ($n_1 \neq n_2$), deve-se adaptá-la como na equação 3.19:

$$T^2 = (\bar{y_1} - \bar{y_2})' \left[\left(\frac{1}{n_1} + \frac{1}{n_2}\right) S_{pl}\right]^{-1} (\bar{y_1} - \bar{y_2}), \tag{3.19}$$

com $S_{pl} = \frac{1}{(n_1+n_2-2)}[(n_1-1)S_1 + (n_2-1)S_2]$.

E, como no caso dos modelos produzidos pela plataforma, com duas amostras pareadas, vem à equação 3.20:

$$T^2 = \bar{d}' \left(\frac{S_d}{n}\right)^{-1} \bar{d}, \tag{3.20}$$

com $d_i = y_{1i} - y_{2i}$, $\bar{d} = \frac{1}{n}\sum_{i=1}^{n} d_i$ e S_d a matriz de covariancias de d.

Tendo, pois, sido explicado o embasamento do teste T^2 de Hotelling, procede-se ao cálculo dos desempenhos destes modelos por até dois critérios univariados, ou até quatro critérios multivariados, conforme o tipo e o número de saídas, a saber.

- **Erro Médio Quadrático (univariado):** Média da soma dos quadrados dos erros entre todos alvos e suas respectivas saídas preditas, como na equação 3.21.

$$mse = \frac{1}{n}\sum_{i=1}^{n}(t_i - y_i)^2 \tag{3.21}$$

- r^2 **de Pearson,** r^2 **de Spearman ou Coeficiente V de Cramer (univariado):** Coeficientes de correlação entre os alvos e as saídas preditas, segundo cada variável separadamente. A decisão entre qual dos três coeficientes utilizar depende do nível de mensuração da variável em questão, como mostram Levin e Fox (2004, cap. 12) e Triola (2012).

  Ademais, tratando-se as varáveis separadamente, gera-se o problema de se ignorar suas as correlações, além de causar indecisão quando contrastam de forma compensatória, obrigando o usuário a criar uma regra de exclusão ou a conduzir um segundo teste baseado nos valores de r^2 obtidos.

- Λ **de Wilk (multivariado):** O coeficiente Λ de Wilk mede a associação entre duas ou mais amostras p-variadas através da estatística T^2 de Hotelling (equação 3.20). Ele pode ser calculado pelas equações 3.22 e 3.23.

$$T^2 = (n_1 + n_2 - 2)\frac{1-\Lambda}{\Lambda} \tag{3.22}$$

$$\Lambda = \prod_{i=1}^{s}\frac{1}{1+\lambda_i}, \tag{3.23}$$

  com λ_i sendo o i-ésimo autovalor de S_d.

- $U^{(s)}$ **de Lawley–Hotelling (multivariado):** Também baseado no teste T^2, assemelha-se ao coeficiente Λ de Wilk e é dado pelas equações 3.24 e 3.25.

$$T^2 = (n_1 + n_2 - 2)U^{(s)} \tag{3.24}$$

$$U^{(s)} = \sum_{i=1}^{s}\lambda_i \tag{3.25}$$

- $V^{(s)}$ **de Pillai (multivariado):** Semelhante ao coeficiente Λ de Wilk, é dado pelas equações 3.26 e 3.27.

$$T^2 = (n_1 + n_2 - 2)\frac{V^{(s)}}{1 - V^{(s)}} \tag{3.26}$$

$$V^{(s)} = \sum_{i=1}^{s} \frac{\lambda_i}{1 + \lambda_i}, \tag{3.27}$$

- θ **de Roy (multivariado):** Como os anteriores, porém dado por 3.28 e 3.29.

$$T^2 = (n_1 + n_2 - 2)\frac{\theta}{1 - \theta} \tag{3.28}$$

$$\theta = \frac{\lambda_1}{1 + \lambda_1}, \tag{3.29}$$

Sendo que, dentre estes quatro coeficientes multivariados, o Λ de Wilk seguido pelo $V^{(s)}$ de Pillai, são tidos como os mais robustos para amostras que venham a apresentar pequenos desvios de normalidade e homocedasticidade. Além disso, tem-se a restrição de $0 \leq \Lambda \leq 1$, fato que o torna ideal para a comparação entre modelos de problemas diferentes. E, também, há de se notar que o coeficiente Λ de Wilk perfaz um teste inverso, no sentido de que a hipótese nula deve ser rejeitada para valores abaixo de um crítico, ao contrário de T^2.

3.2.2.5 *Designs* — comparações e seleção entre modelos

Os métodos disponíveis para escolha de um melhor modelo são denominados *designs* e consistem em algoritmos iterativos voltados à pesquisa do impacto da variação dos parâmetros (como o número de coeficientes nas regressões ou o número de neurônios nas RNAs) nos desempenhos dos modelos feitos, visando sua otimização. A versão v0.1 da plataforma oferece quatro possíveis *designs*:

- **Design Randômico:** Inicialmente, os parâmetros do modelo (como ordem e coeficientes para uma regressão linear, ou número de neurônios e funções de treinamento ou ativação para uma RNA) são aleatóriamente escolhidos e o modelo correspondente é confeccionado, tendo seu desempenho aferido. Numa secunda iteração, este processo é repetido, porém o desempenho do modelo resultante é comparado com o do anterior pelo coeficiente escolhido (Λ, $U^{(s)}$ $V^{(s)}$ ou θ), resguardando-se o melhor. Finalmente, após um grande número de sucessivas iterações, o melhor modelo retido e apresentado ao usuário.

  Este *design* é bastante prático e simples, porém pouca informação a respeito do impacto da escolha dos diferentes parâmetros no desempenho do modelo é revelada.

- **Design Exaustivo:** Semelhante ao *design* randômico, porém este realiza sistematicamente todas as permutações de parâmetros possíveis, confeccionando uma amostra de modelos semelhantes a cada combinação e aferindo-lhes os desempenhos. Após a exautão das possibilidades, a melhor configuração de parâmetros é evidencida comparando-se

os desempenhos das amostras por uma MANOVA cujo número de fatores corresponde ao número de diferentes parâmetros permutados, apresentando-se-lhe ao usuário.

Pesquisando RNAs, por exemplo, poder-se-ia eleger como parâmetros, o número de neurônios em cada camada, a função de treinamento e as funções de transferência (ou de ativação) para cada camada, entre outros.

Este *design* é bastante custoso computacionalmente, mas retêm informações sobre as relações entre o parâmetros de um modelo e seu desempenho.

- **Design por Algoritmo Genético:** Simulando um processo evolutivo darwiniano, uma população inicial de modelos é criada de forma aleatória pelo sorteio dos parâmetros a cada indivíduo (modelo). Tem-se aí a primeira geração desta população. Prossegue-se, então, ao cálculo do desempenho de cada indivíduo, ordenando-se-lhes. Ademais, uma porcentagem fixa dos piores modelos é desprezada e, dos remanescentes, perfaz-se uma série de cruzamentos dois a dois, onde características de cada progenitor (modelo sobrevivente) são atribuídas ao acaso a cada indivíduo da prole. O número de cruzamentos deve recompor o número de indivíduos desprezados, criando uma segunda geração. Este processo perdura por sucessivas gerações convergindo, ao final, em uma população homogênea e melhor adaptada à modelagem do problema em questão.

  Este *design*, relativamente aos outros apresentados, encontra-se em situação intermediária no que tange, tanto o custo computacional, quanto o nível de informação retida sobre o impacto da variação dos parâmetros nos desempenhos.

- **Design customizado:** É deixada a possibilidade de que o usuário desenvolva seu próprio *design* e faça com que a plataforma o utilize. Ele deve desenvolver uma função computacional (um arquivo `.m`, `.p` ou `.mex`) passível de ser referenciada no diretório em que estiver, ou seja, o caminho para seu arquivo na árvore de diretórios deve ser acessível e constar no campo `designFcn` da tag `<models>...<\models>` no arquivo de configuração.

3.2.2.6 Retorno de dados ao usuário

Estando a plataforma instalada no computador pessoal do usuário, ele pode requerer diversos tipos de arquivos intermediários de dados gerados pela plataforma durante o processo de modelagem, entre eles, o próprio arquivo binário do modelo (arquivo `.mat`), o script de interface (arquivo `.m`) e o relátório gráfico em pdf.

Porém, se optou por utilizar uma plataforma *on-line*, tanto no servidor central, quanto em um servidor prórpio, ele os pode receber por e-mail em um arquivo compactado (arquivo `.zip` ou `.tgz`).

Caso retenha inicialmente apenas o arquivo binário, pode-se, a partir dele e a qualquer momento, recriar os demais com o seginte comando no terminal:

```
$ ncl [arquivo_binario.mat] -o -all
```

3.3 Obtendo ajuda

A lista completa dos comandos da versão v0.1 encontra-se em seu manual, bem como diretrizes de como operar o *software* em diversas situações. O usuário pode obtê-los no endereço eletrônico <http://www.neurocelle.org/documentation.html> ou em uma instalação local pelo seguinte comando via terminal:

```
$ man ncl
```

Ele pode ainda obter ajuda sobre um comando, estando dentro de uma sessão, com os comandos de ajuda geral e específica, respectivamente, como abaixo:

```
$ ncl -h
```

ou

```
$ ncl -h [comando_em_duvida]
```

4 RESULTADOS E DISCUSSÕES

4.1 Sequencia de resultados e convenções

Prezando por consistência textual, os elementos gráficos relativos a cada procedimento computacional serão apresentados na sequencia em que foram realizados, a saber:

1. Detecção de matriz de alvos, tamanho das amostras e suas dimensionalidades.
2. PCA e redução de dimensionalidade[1].
3. Teste de normalidade sobre a matriz de entradas reduzida por PCA e remoção de *outliers*.
4. Comparação dos modelos criados contra os da literatura.

Convencionar-se-ão as segintes abreviações nas tabelas de resultados comparados:

- NP - Número de padrões de treinamento utilizados (tamanho da amostra).
- NN(in) - Número de neurônios na camada de entrada (número de variáveis preditoras);
- NN(out) - Número de Neurônios na camada de saída (número de variáveis preditas);
- NCO - Número de Camadas Ocultas;
- NN(1) - Número de Neurônios na primeira camada oculta;
- NN(2) Número de Neurônios na segunda camada oculta;
- FTN - Função de Treinamento (ou de Aprendizagem);
- FTF(1-2) - Função de Transferência entre a primeira camada oculta e a segunda;
- FTF(2-out) - Função de Transferência entre a segunda camada oculta e a camada de saída;
- FTF(out) - Função de Transferência da camada de saída;
- *mse* - Erro Médio Quadrático.

Todas as iterações dividem o número de padrões para treinamento, validação e teste, respectivamente, em 60%, 20% e 20%. Tendo sido estabelecido previamente a lista de parâmetros a serem permutados como sendo os mesmos citados na literatura, tem-se *trainrp*, *trainlm* e *traingda* para as funções de treinamento; *tansig*, *logsig*, *purelin* e *hardlim* para funções de transferência; e [1 50] para variação do número de neurônios das camadas. Todas as permutações totalizam 480000 possibilidades de arquiteturas distintas, o que é inexeqível em tempo hábil. Portanto, restringiu-se o tempo de pesquisa em cada *design* a 24 horas, interrompendo-se o processo ao término do tempo.

[1] Decidiu-se realizar a técnica de PCA antes do teste de normalidade, justamente para que a padronização dos escores eliminassem, tanto as diferenças entre as escalas das variáveis de entrada, quanto as correlações existente entre elas, promovendo uma transformação prévia à normalidade.

4.2 Problema da determinação de Riqueza e Abundância Absoluta de macroalgas em ambientes lóticos

4.2.1 *Preâmbulo*

Primeiramente abordado por Rocha et al. (2017), o problema da predição de Riqueza e Abundância Aboluta de macroalgas foi modelado exclusivamentente por RNAs. Nele, utilizaram-se todos os vinte fatores ambientais medidos como preditores. Adversamente, foi pesquisado um modelo para cada saída predita separadamente, um exclusivamente para predição de Abundância Absoluta e outro para Riqueza. O único pré tratamento dos dados foi a normalização por faixa simples (entre -1 e 1), tanto das entradas como das saídas. Nenhuma remoção de *outliers* foi feita e todos os oitenta e nove padrões de aprendizagem foram utilizados. Por fim, a determinação dos melhores modelos se deram pelos *designs* randômico e exaustivo. Os desempenhos de ambos os modelos são comparados com aquele gerado pela plataforma na tabela 2.

4.2.2 *Abordagem da plataforma*

Sumarizando, a matriz de entradas original apresenta oitenta e nove padrões, com vinte variáveis cada (dimensões 89x20), em escalas diversas. A matriz de saídas apresenta duas variáveis para cada padrão (89x2), também com escalas discrepantes. Assim, realizando-se PCA sobre a matriz de entradas, obteve-se o gráfico da figura 8, onde é mostrada a relevância que cada componente principal terá na explicação da variância total da amostra após a transformação dos escores. Vê-se ali, que os quatorze primeiros componentes são suficientes para representar mais de 90% de toda variância original.

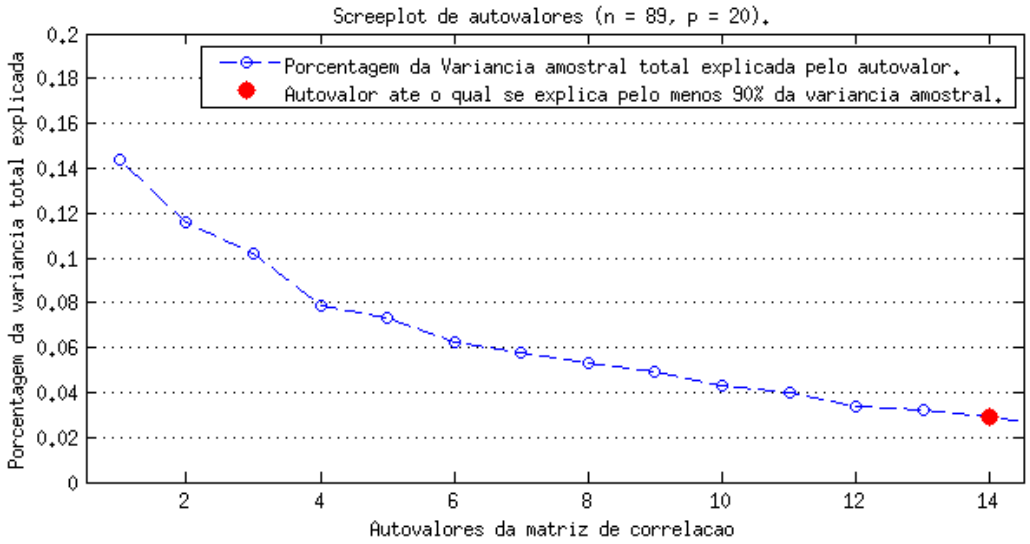

Figura 8 – Gráfico screeplot dos Autovalores do problema de Riqueza e Abundância.

Por conseguinte, a nova representação dos escores transformados por PCA mostra-se desprovida de qualquer padrão hiperelipsóide, como posto no gráfico tipo biplot da figura 9.

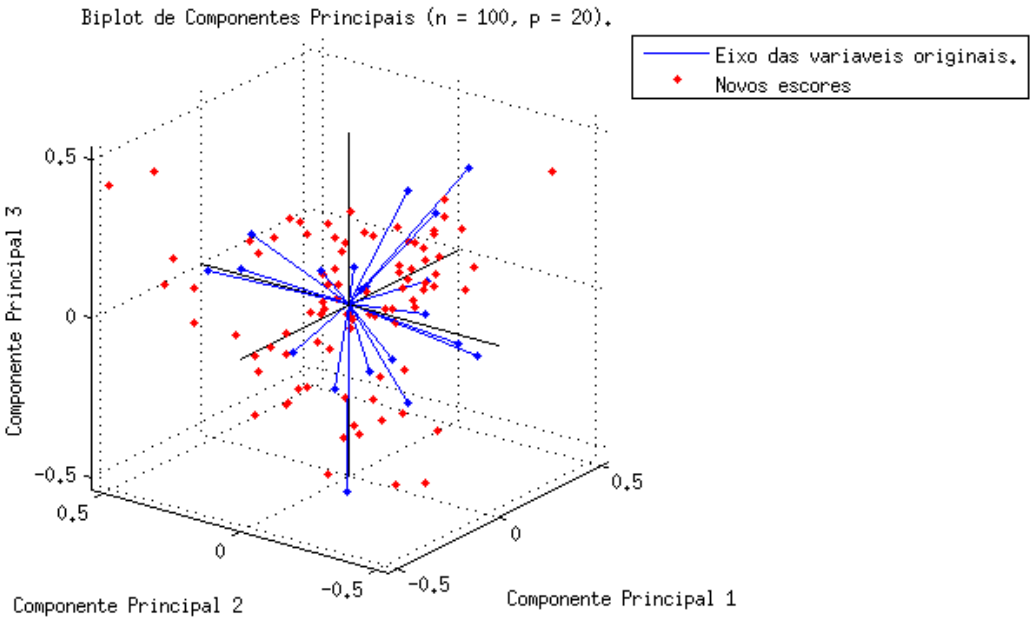

Figura 9 – Gráfico biplot dos Componentes Principais do problema de Riqueza e Abundância.

Eis que, restringindo-se apenas aos quatorze primeiros componentes, a nova matriz de entrada foi submetida ao teste de normalidade e oito *outliers* foram evidenciados. A figura 10 mostra um gráfico Q-Qplot para análize visual deste padrão de normalidade. Infelizmente, devido aos *outliers*, a amostra desvia-se significativame da normalidade exigindo a remoção deles, fato que reduz a matriz de entrada e a de saída para oitenta e um padrões.

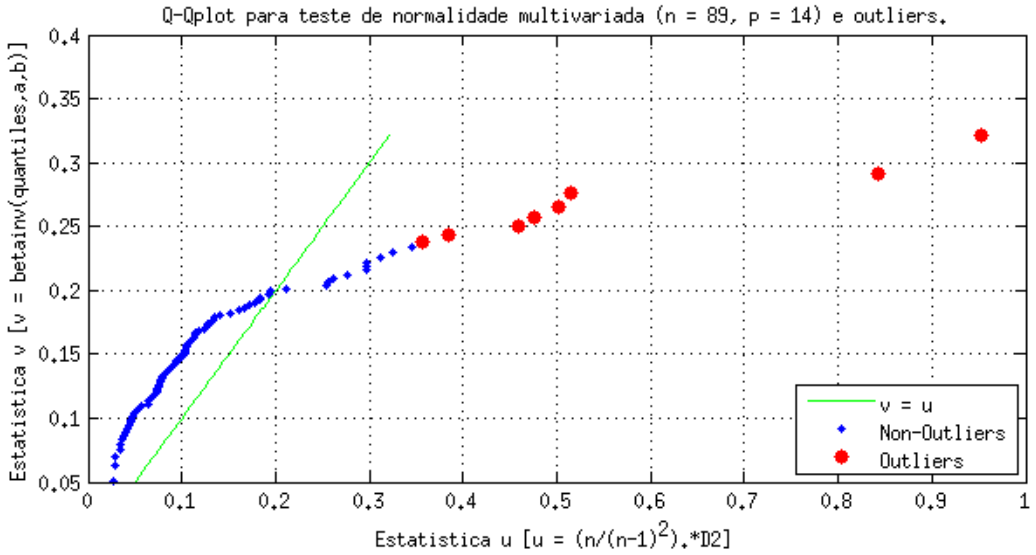

Figura 10 – Gráfico Q-Qplot dos escores de Componentes Principais do problema de Riqueza e Abundância.

Por fim, com as matrizes de entradas e saídas reduzidas e padronizadas foram pesquisados modelos de RNAs para predição simultânea de Riqueza e de Abundância, segundo dois *designs* — randômico e exaustivo.

4.2.3 *Comparações*

O melhor modelo obtido pelo critério do coeficiente Λ de Wilk é comparado com o da literatura na tabela 2.

Tabela 2 – Comparações de desempenhos de RNAs para Riqueza e Abundância de macroalgas.

Parâmetros	**Modelos**		
	Riqueza	**Abundância**	**Simultânea**
NP	89	89	81
NN(in)	20	20	14
NN(out)	1	1	2
NCO	2	2	2
NN(1)	28	19	5
NN(2)	13	9	10
FTN	*trainrp*	*trainlm*	*trainrp*
FTF(1-2)	*tansig*	*tansig*	*tansig*
FTF(2-out)	*logsig*	*tansig*	*tansig*
FTF(out)	v.n.c.*	v.n.c.	*tansig*
R^2(tr)	0.92	0.96	[0.88, 0.91]
R^2(v)	0.86	0.86	[0.81, 0.87]
R^2(t)	0.91	0.87	[0.83, 0.84]
mse	0.0180	$0.1189\text{x}10^{-10}$	0.3602
Coef. Λ	v.n.c.	v.n.c	[0.9832 0.9764 0.9694]

*: Valor não conhecido.

Ademais, para que se possa analizar a acurácea das saídas preditas em relação aos seus alvos, apresentam-se nas figuras 11 e 12, os gráficos comparativos dos escores padronizados para cada saída separadamente, tendo sido incluídos os *outliers* que não foram utilizados durante o treinamento da RNA.

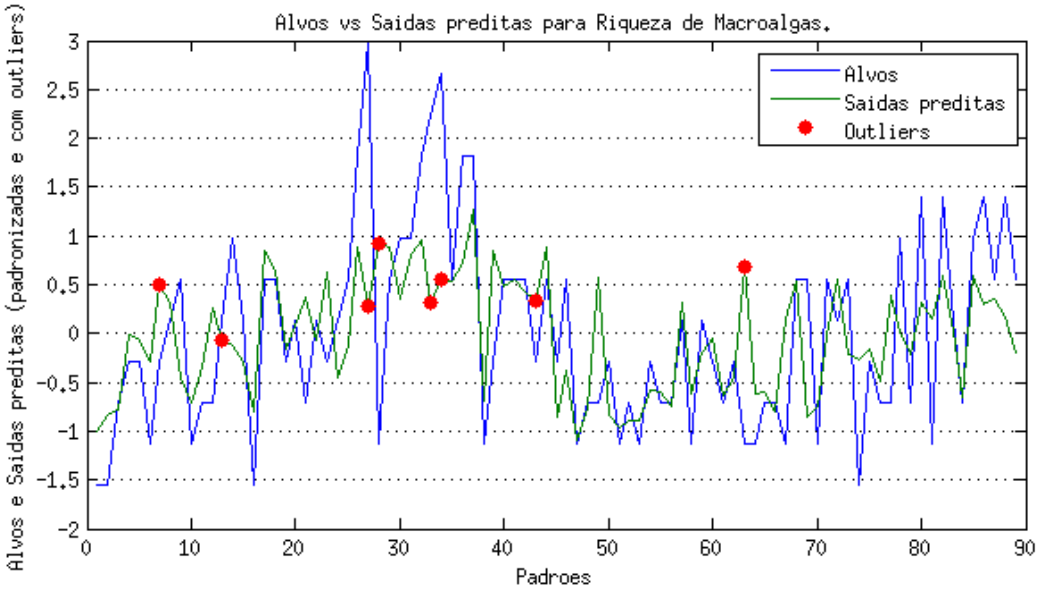

Figura 11 – Gráfico comparativo dos escores padronizados de Riqueza de Macroalgas, contra os respectivos alvos.

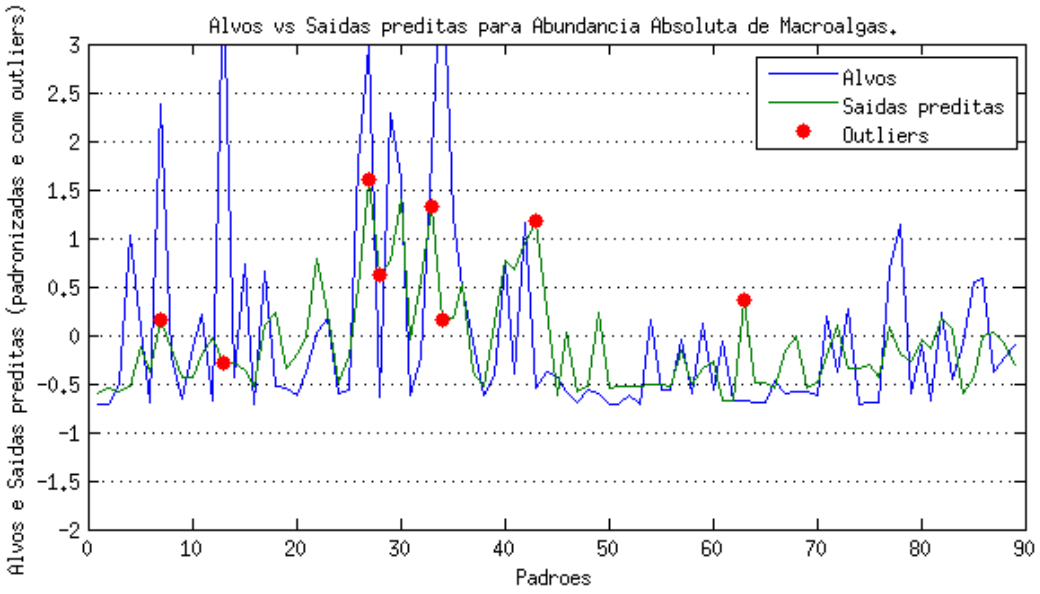

Figura 12 – Gráfico comparativo dos escores padronizados de Abundância Absoluta de Macroalgas, contra os respectivos alvos.

Nota-se que a plataforma obteve um modelo aparentemente inferior ao da literatura. No entanto, argui-se pelo fato de ter havido predição simultânea no modelo obtido pela plataforma, o que aumenta sua complexidade. Outrossim, o critério para seleção do melhor modelo na literatura foi o coeficiente r^2 de Pearson para correlações simples entre saídas preditas e alvos, separadamente, o que despreza eventuais correlações entre as saídas distintas. Assim, dado que não houve acesso aos vetores de erros daqueles modelos, não foi possível calcular a matriz de covariâncias para ambas as saídas conjuntamente e, consequentemente, não se pôde obter Λ para aquele caso. Porém, o modelo da plataforma obteve, por exemplo, para o conjunto de teste, $\Lambda = 0.9694$, equivalendo a $T^2 = 2.5253$ (eq. 3.22) e a $F_{14,67} = 0.1511$ (eq. 3.18), enquanto $F_{0.05,14,67} = 1.84$ é o valor crítico. Portanto, não se rejeitaria a hipótese nula de que as médias amostrais das matrizes de alvos e a da matriz de saídas preditas são iguais, fato que coloca o modelo como viável a estas predições.

Porém, fica evidente que, apesar de todos os coeficientes de correlação serem superiores a 0.8 em ambos os modelos, remanesce uma dúvida no contexto multivariado: em que grau, tratar separadamente as variáveis preditas atenua os efeitos de covariância, causando a discrepância observada entre os desempenhos dos modelos? Aqui, mesmo a comparação via *mse* se torna inviável pois, dadas as tranformações prévias feitas pela plataforma sobre as matrizes de alvos, seu impacto nas escalas modificam os valores de *mse* por consequencia.

4.3 Problema da determinação do Índice de Qualidade da Água

4.3.1 *Preâmbulo*

Alves (2015) associou os escores da escala de Índice de Qualidade da Água em domínios urbanos a medições espectrofotométricas que são práticas e de baixo custo, por meio de RNAs em seu trabalho. Esta modelagem apresenta somentente uma variável a ser predita e é medida

em nível intervalar. Não obstante, o número de variáveis preditoras é demasiadamente alto, chegando seiscentas e onze. Este grande número de entradas corresponde às absorbâncias do espectro de comprimentos de onda vão de 190 a 800 nanometros. A matriz de entrada foi pré tratada por diferênciais de segunda ordem sobre as entradas, seguida de normalização por faixa simples, entre -1 e 1. Não houve teste de normalidade, nem detecção de *outliers*, nem redução de dimensionalidade. Mesmo assim, um modelo satisfatório foi obtidos por *design* randômico e seu desempenho é comparado com o do modelo gerado pela plataforma no tabela 3.

4.3.2 *Abordagem da plataforma*

A grande matriz de entrada original apresenta cem padrões, com seiscentas e onze variáveis preditoras para cada um (dimensões 100x611), numa mesma escala. Já a matriz de saídas apresenta apenas uma variável (100x1), em escala discrepante das entradas. Logo se notou que o grande número de variáveis preditoras reflete a multicolinearidade presente nos dados de absorbância, pois um mesmo composto químico absorve radiação em diversos comprimentos de onda, além de haver grupos de compostos semelhantes que podem absorver diferentes quantidades de luz para um mesmo comprimento. Por isso, iniciou-se por realizar uma PCA, resultando no gráfico da figura 13. Ali, apenas três componentes principais conseguem, incrivelmente, explicar mais de 90% da variância total da amostra, tamanha a redundância de informação presente nos dados originais.

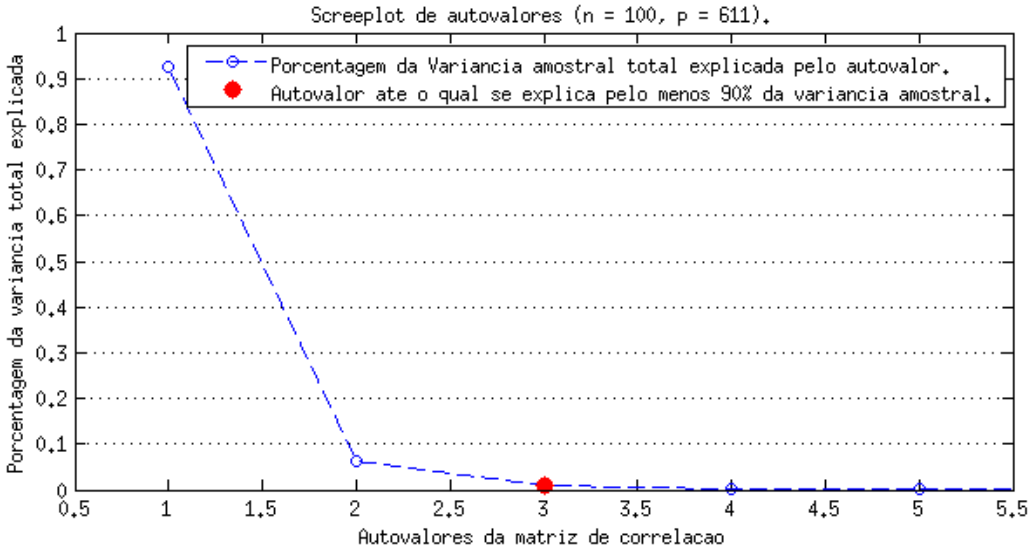

Figura 13 – Gráfico screeplot dos Autovalores do problema de Índice de Qualidade da Água.

Após a PCA, os dados de entrada foram simultaneamente padronizados pela matriz de covariâncias amostras e "descorrelacionados" pela rotação dos eixos cartesianos originais. A figura 14 mostra um gráfico biplot em que os novos escores não apresentam padrão hiperelipsóide aparente. No entanto, os novos escores escores difundem-se quase que sobre uma reta ao longo da primeira componente principal devido à sua relevância sobre a explicação da variância amostral.

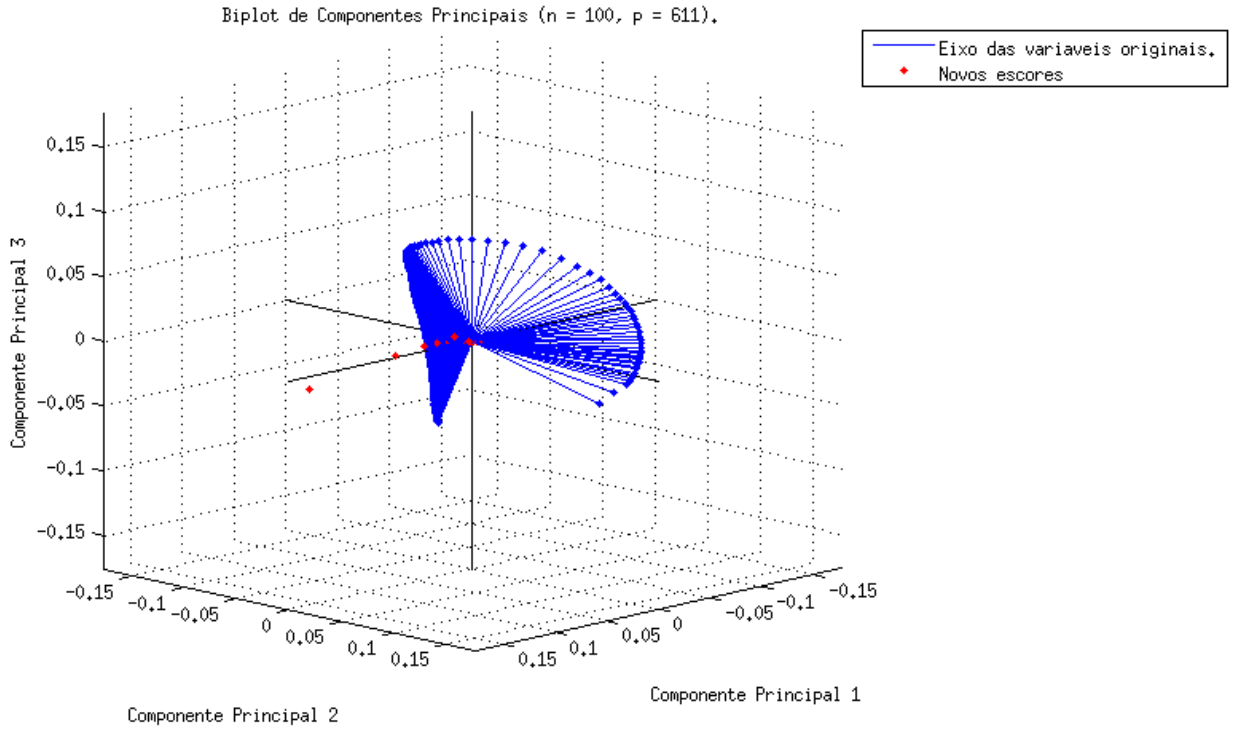

Figura 14 – Gráfico biplot dos Componentes Principais do problema de Índice de Qualidade da Água.

Em seguida, passou-se ao teste de normalidade e à detecção de *outliers* sobre a matriz de entradas reduzida pela PCA (dimensões 100x3). E eis que, dada a trânsformação promovida pela PCA, emergiu-se um padrão muito próximo da normalidade, com apenas seis *outliers* num total de cem. A figura 15 apresenta o gráfico tipo Q-Qplot com tal padrão de normalidade.

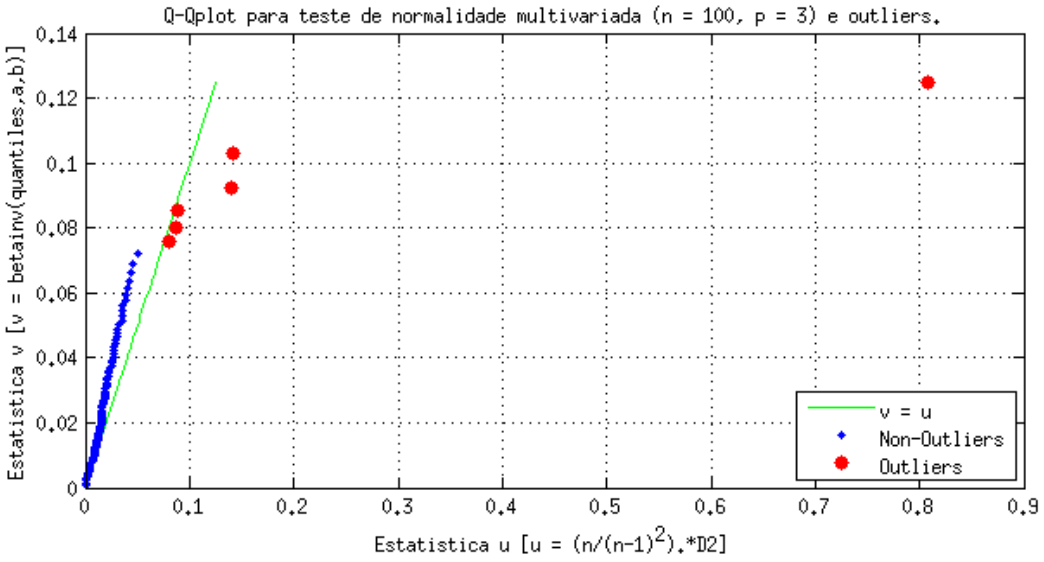

Figura 15 – Gráfico Q-Qplot dos escores de Componentes Principais do problema de Índice de Qualidade da Água.

Eis que a retirada dos pontos aberrantes reduziu o número de padrões para noventa e quatro, possibilitando o inicio do processo de pesquisa de RNAs segundo os *designs* randômico e

exaustivo, pelo critério de seleção Λ de Wilk.

4.3.3 *Comparações*

Por fim, os modelos são comparados na tabela 3, onde mais uma vez o modelo obtido mostrou-se inferior ao referido por Alves (2015), mas em um problema univariado. Neste caso, a comparação entre os coeficientes r^2 de Pearson pode ser feita diretamente, ignorando-se o coeficiente Λ.

Tabela 3 – Comparações de desempenhos de RNAs para o Índice de Qualidade de Água.

Parâmetros	**Modelos**	
	IQA Literatura	**IQA Plataforma**
NP	100	94
NN(in)	611	3
NN(out)	1	1
NCO	2	2
NN(1)	17	39
NN(2)	10	12
FTN	*trainrp*	*trainlm*
FTF(1-2)	*logsig*	*tansig*
FTF(2-out)	*logsig*	*tansig*
FTF(out)	v.n.c.*	*tansig*
R^2(tr)	0.95	0.92
R^2(v)	0.95	0.89
R^2(t)	0.95	0.86
mse	$0.8410\text{x}10^{-5}$	0.3993
Coef. Λ	v.n.c.	[0.9998 0.9991 0.9995]

*: Valor não conhecido.

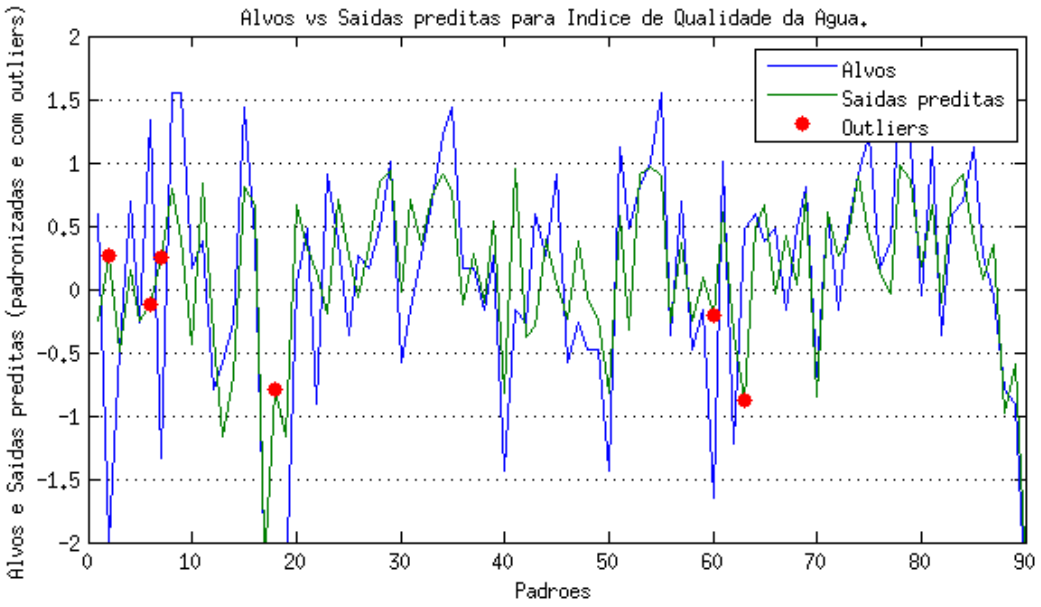

Figura 16 – Gráfico comparativo dos escores padronizados de Índice de Qualidade da Água, contra os respectivos alvos.

Acredita-se que os piores resultados sejam um reflexo da restrição no tempo de processamento na pesquisa, o que leva à indagação sobre a frequência de surgimento de redes propensas a um melhor desempenho e o eventual emprego de um *design* customizado baseado em algoritmos de Monte Carlo.

5 CONCLUSÕES

Eis que fora implementada uma plataforma computacional *on-line* e passível de instalação local para confecção automática de modelos matemáticos sob demanda. Escolheu-se-lhe o nome *neurocelle*, hospedada em dois repositórios nos endereços eletrônicos <http://www.neurocelle.org> e <https://github.com/estrogonelda/neurocelle>.

Testando-se sua robustez, duas aplicações diversas e bem sucedidas de modelos numéricos envolvendo Redes Neurais Artificiais foram tomadas da literatura, afim de se comparar os resultados publicados em cada caso com os resultados apresentados pelo modelo correspondente gerado automaticamente pela plataforma.

A plataforma obteve êxito ao gerar modelos de acurácea significativa, mas com desempenhos sensívelmente inferiores aos dos modelos tomados da literatura. Eis que esta inferioridade aludida é questionada em um contexto multivariado. Além de se demonstrar a praticidade na confecção dos modelos em detrimento dos menores desempenhos, bem como à pluralidade dos problemas abordado nos dois casos.

Em suma, esta plataforma vêm sugerir novas perspectivas no lavoro técnico computacional, especialmente na possibilidade de se integrar, de forma intuitiva, dados de pesquisadores oriundos de áreas bastante distintas num mesmo aparato sistêmico e tecnológico, por meio de modelos e arquivos padronizados e de livre acesso.

REFERÊNCIAS

ALVES, E. M. *Uso de espectrofotometria UV–vis associada a redes neurais artificiais como alternativa para determinação do índice de qualidade da água*. Dissertação (Monografia) — Universidade Estadual Paulista “Julio de Mesquita Filho”, 2015.

BELLE, G. V. et al. *Biostatistics: a methodology for the health sciences*. 2. ed. [S.l.]: Wiley, 2004.

ELSTON, R. C.; GRIZZLE, J. E. Estimation of time-response curves and their confidence bands. *Biometrics*, v. 18, p. 148–159, 1962.

FILHO, P. A. C.; POPPI, R. J. Genetic algorithm in chemistry. *Instituto de Química - Universidade Estadual de Campinas*, 1998.

FREE SOFTWARE FOUNDATION. *The GNU Operating System and the Free Software Movement*. 2017. Disponível em: <https://www.gnu.org/>. Acesso em: 27 jan 2017.

FREE SOFTWARE FOUNDATION. *Licenses - GNU Project - Free Software Foundation*. 2017. Disponível em: <https://www.gnu.org/licenses/#GPL>. Acesso em: 27 jan 2017.

FREE SOFTWARE FOUNDATION. *Why you shouldn’t use the Lesser GPL for your next library - GNU Project - Free Software Foundation*. 2017. Disponível em: <https://www.gnu.org/philosophy/why-not-lgpl.htm>. Acesso em: 27 jan 2017.

GNANADESIKAN, R.; KETTENRING, J. R. Robust estimates, residuals, and outliers detection with multiresponse data. *Biometrics*, v. 28, p. 81–124, 1972.

GOETHALS, P. L. M. et al. Applications of artificial neural networks predicting macroinvertebrates in freshwaters. *Aquat. Ecol.*, p. 491–508, 2007.

HAGAN, M. T.; DEMUTH, H. B.; BEALE, M. *Neural Network Design*. 1. ed. [S.l.]: Critic Publishing House, 1996.

HAYKIN, S. *Redes Neurais: Princípios e Pratica*. 2. ed. [S.l.]: Bookman, 2008.

INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS. *IEEE SA - 1003.1-2008 - IEEE Standard for Information Technology - Portab*. 2017. Disponível em: <https://www.ieee.org/>. Acesso em: 27 jan 2017.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. *ISO - International Organization for Standardization*. 2017. Disponível em: <http://www.iso.org/iso/home.html>. Acesso em: 27 jan 2017.

JOHNSON, D. W. W. R. A. *Applied Multivariate Statistical Analysis (6th Edition)*. 6. ed. [S.l.]: Prentice Hall, 2007. ISBN 0131877151,9780131877153.

KOVACS, Z. L. *Redes neurais artificiais: Fundamentos e Aplicações*. 3. ed. [S.l.]: Livraria da Física, 2002.

LANTZ, B. *Machine Learning with R*. 1. ed. [S.l.]: Packt, 2013.

LEVIN, J.; FOX, J. A. *Estatística para Ciências Humanas*. 9. ed. [S.l.]: Pearson Prentice Hall, 2004.

MARDIA, K. V. Measures of multivariate skewness and kurtosis with applications. *Biometrika*, v. 57, p. 519–530, 1970.

MATHWORKS. *Matlab - The Language of Technical Computing*. 2017. Disponível em: <https://www.mathworks.com/products/matlab/>. Acesso em: 27 jan 2017.

MONTGOMERY, D. C. *Design and analisys of experiments*. 8. ed. [S.l.]: Wiley, 2013.

NATIONAL CANCER INSTITUTE. *NCI Genomic Data Commons*. 2017. Disponível em: <http://gdc.cancer.gov/>. Acesso em: 27 jan 2017.

PERRY, L. A.; MONTGOMERY, D. C.; FOWLER, J. W. A partition experimental design for a sequential process with a large number of variables. *Quality and Reliability Engineering International*, v. 23, p. 555–564, 2007.

RENCHER, A. C.; CHRISTENSEN, W. F. *Methods of Multivariate Analysis*. 3. ed. [S.l.]: Wiley, 2012.

ROCHA, J. C. et al. Modeling the species richness and abundance of lotic macroalgae based on habitat characteristics by artificial neural networks: a potencially useful tool for stream biomonitoring programs. *Journal of Applied Phycology*, 2017.

SAMARASINGHE, S. *Neural Networks for Applied Sciences and Engineering: from Fundamentals to Complex Pattern Recognition*. 1. ed. [S.l.]: Auerbach, 2007.

SHALEV-SHWARTZ, S.; BEN-DAVID, S. *Understanding Machine Learning from Theory to Algorithms*. 1. ed. [S.l.]: Cambridge University Press, 2014.

TEX USERS GROUP. *Comprehensive TEX Archive Network*. 2017. Disponível em: <https://www.ctan.org/>. Acesso em: 27 jan 2017.

THE R PROJECT. *The Comprehensive R Archive Network*. 2017. Disponível em: <https://cran.r-project.org/>. Acesso em: 27 jan 2017.

THE R PROJECT. *R: The R Project for Statistical Computing*. 2017. Disponível em: <https://www.r-project.org/>. Acesso em: 27 jan 2017.

TRIOLA, M. F. *Introdução à Estatística*. 10. ed. [S.l.]: LTC, 2012.

UNIVERSITY OF CALIFORNIA–IRVINE. *UCI Machine Learnig Repository*. 2017. Disponível em: <http://archive.ics.uci.edu/ml/>. Acesso em: 27 jan 2017.

WALZACK, S.; CERPA, N. Heuristic principles for the design of artificial neural networks. *Information and Software Technology*, v. 41, p. 107–117, 1999.

Anexo 1

***Template* de arquivo de configuração geral**

```
<!DOCTYPE nclml>
<projects>
  <info>
    title = Project1;
    authors = Author1, Author2;
    emails = author1@gmail.com, author2@hotmail.com;
    institution = Institution1, Institution2;
    date = 2017-mai;
    files = % File1, File2, File3;
    signature = % gpg_key1, gpg_key2;
  </info>

  <models>
    % --- Information data.
    <info>
      problem = Problem1;
      configfile = problem1.xml;
      datafile = problem1.mat;
      objfile = obj_problem1.mat;
      pre_processfcn
      processfcn
      post_processfcn
    </info>
    % --- Statistics data.
    <statistics>
    </statistics>
    % --- Parameters data.
    <parameters>
      % --- General params.
      model_type = regression, ann, fuzzy;
      model_sub_type
      model_fcn = ra, ann, fzz;
      design_type = random, exaustive, monte_carlo, genetic;
      design_fcn = designer
      performance_crit = intervalar, ordinal, nominal, multivariate;
      ordenation_crit = scalar, unit, arch, model;
      selection_crit = Wilk-Lambda;
      force_iterations = y;
```

```
        num_iterations = [10 10000];
        num_saves = 5;
        % --- Model specific params (Artificial Neural Networks).
        ann_type = supervised, non-supervised;
        % Deterministic params.
        num_hidden_layers = [1 5];
        num_neurons_layers = [1 100];
        trainFcn = trainlm, trainrp, trainscg, traingdx, traingda;
        transferFcn_layers = logsig, purelin, tansig, hardlim;
        extra_deterministic_param = none;
        % Stochastic params.
        divide_crit = random, persistent_random;
        raffling_weights_crit = random, persistent_random;
        raffling_bias_crit = random, persistent_random;
        num_mc_vals = [.01 .125 .25 .5 1];
        num_lr_vals = [.001 .01 .1 1];
        extra_stochastic_param = none;
        % Other params.
        design_vct
        design_mask = [0 0; 1 1; 1 1; 1 1; 1 1;...
            1 1; 1 1; 1 1; 1 1; 1 1];
        design_archs
        num_design_archs
        num_archs_units
        num_units_replicates = [1 100];
        num_total
        % --- Design specific params (Genetic Algorithm).
        population_size = 100;
        survive_ratio = 0.3;
        mutation_rate = 0.05;
        genes
        imigration
        num_generations = 200;
      </parameters>
    <models>

    <models> % Placeholder for a second model. </models>
    <simulations></simulations>
    <reports></reports>
    <interfaces></interfaces>
</projects>
```