sid.inpe.br/mtc-m21c/2021/04.07.20.54-TDI

UMA FERRAMENTA PARA AUTOMATIZAR A REPRODUTIBILIDADE DE MÉTODOS COMPUTACIONAIS PARA CLASSIFICAÇÃO DA COBERTURA DA TERRA

Rafael Monteiro Mariano

Dissertação de Mestrado do Curso de Pós-Graduação em Computação Aplicada, orientada pelos Drs. Gilberto Ribeiro de Queiroz, e Pedro Ribeiro de Andrade Neto, aprovada em 01 de abril de 2021.

URL do documento original:
<http://urlib.net/8JMKD3MGP3W34R/44F9DDL>

INPE
São José dos Campos
2021

PUBLICADO POR:

Instituto Nacional de Pesquisas Espaciais - INPE
Coordenação de Ensino, Pesquisa e Extensão (COEPE)
Divisão de Biblioteca (DIBIB)
CEP 12.227-010
São José dos Campos - SP - Brasil
Tel.:(012) 3208-6923/7348
E-mail: pubtc@inpe.br

sid.inpe.br/mtc-m21c/2021/04.07.20.54-TDI

UMA FERRAMENTA PARA AUTOMATIZAR A REPRODUTIBILIDADE DE MÉTODOS COMPUTACIONAIS PARA CLASSIFICAÇÃO DA COBERTURA DA TERRA

Rafael Monteiro Mariano

Dissertação de Mestrado do Curso de Pós-Graduação em Computação Aplicada, orientada pelos Drs. Gilberto Ribeiro de Queiroz, e Pedro Ribeiro de Andrade Neto, aprovada em 01 de abril de 2021.

URL do documento original:
<http://urlib.net/8JMKD3MGP3W34R/44F9DDL>

INPE
São José dos Campos
2021

Dados Internacionais de Catalogação na Publicação (CIP)

Mariano, Rafael Monteiro.

M337f Uma ferramenta para automatizar a reprodutibilidade de métodos computacionais para classificação da cobertura da terra / Rafael Monteiro Mariano. – São José dos Campos : INPE, 2021.

xx + 63 p. ; (sid.inpe.br/mtc-m21c/2021/04.07.20.54-TDI)

Dissertação (Mestrado em Computação Aplicada) – Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2021.

Orientadores : Drs. Gilberto Ribeiro de Queiroz, e Pedro Ribeiro de Andrade Neto.

1. Reprodutibilidade. 2. Uso e cobertura da terra. 3. Séries temporais de sensoriamento remoto. I.Título.

CDU 004.4:528.8

MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÕES

INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS
Serviço de Pós-Graduação - SEPGR

DEFESA FINAL DE DISSERTAÇÃO DE RAFAEL MONTEIRO MARIANO
BANCA Nº 052/2021, REG 141690/2018

No dia 01 de abril de 2021, as 14h, por teleconferência, o(a) aluno(a) mencionado(a) acima defendeu seu trabalho final (apresentação oral seguida de arguição) perante uma Banca Examinadora, cujos membros estão listados abaixo. O(A) aluno(a) foi APROVADO(A) pela Banca Examinadora, por unanimidade, em cumprimento ao requisito exigido para obtenção do Título de Mestre em Computação Aplicada. O trabalho precisa da incorporação das correções sugeridas pela Banca Examinadora e revisão final pelo(s) orientador(es).

Novo Título: " Uma Ferramenta para Automatizar a Reprodutibilidade de Métodos Computacionais para Classificação da Cobertura da Terra. "

Eu, Rafael Duarte Coelho dos Santos, como Presidente da Banca Examinadora, assino esta ATA em nome de todos os membros, com o consentimento dos mesmos.

Dr. Rafael Duarte Coelho dos Santos - Presidente - INPE
Dr. Gilberto Ribeiro de Queiroz - Orientador - INPE
Dr. Pedro Ribeiro de Andrade Neto - Membro Interno - INPE
Dr. Nandamudi Lankalapalli Vijaykumar - Membro Interno - INPE
Dra. Karla Donato Fook - Membro Externo - ITA

Documento assinado eletronicamente por **Rafael Duarte Coelho dos Santos**, **Tecnologista**, em 06/04/2021, às 13:40 (horário oficial de Brasília), com fundamento no art. 6º do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.mctic.gov.br/verifica.html, informando o código verificador **6857959** e o código CRC **7DF0C1EF**.

Referência: Processo nº 01340.001865/2021-38 SEI nº 6857959

A meus pais ***Edvaldo*** *e* ***Romilda***

AGRADECIMENTOS

Agradeço aos meus pais, Edvaldo Santos Mariano e Romilda da Silva Monteiro Mariano, e meus irmãos, Rodrigo e Matheus Monteiro Mariano, pelo apoio, incentivo e paciência durante todo o desenvolvimento deste trabalho.

Agradeço a minha falecida vó Anita pelo exemplo de amor e carinho em vida. Foi uma guerreira e acredito que estaria feliz por está minha conquista.

Agradeço ao meu primo, José Wagner dos Santos Júnior, e meu amigo Ramon dos Santos Oliveira pelas amizades e conversas durante todo período que estive presente no INPE.

Agradeço a minha namorada, Maria Luiza Diofani de Melo, pelo carinho, confiança e incentivo durante essa jornada.

Agradeço aos meus colegas da CAP pelas risadas e momentos de aprendizagem.

Agradeço ao Dr. Alisson Dal Lago e Dr. Luis Eduardo Antunes Vieira pelo incentivo e conhecimento durante a minha entrada no INPE.

Agradeço aos meus orientadores, Dr. Gilberto Ribeiro de Queiroz e Dr. Pedro Andrade, pelo tempo e confiança depositado nesse trabalho.

Agradeço ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pelo apoio financeiro.

RESUMO

A reprodutibilidade das pesquisas é um tópico de grande discussão na comunidade científica. Existe um movimento com o propósito de incentivar os pesquisadores a adotarem medidas para facilitar o acesso aos conjuntos de dados e métodos computacionais usados durante o desenvolvimento de uma pesquisa, permitindo que outros pesquisadores possam reproduzir esse trabalho, não apenas para verificação, mas também para aprendizado. Esta questão tem motivado os principais periódicos a elaborarem documentos de boas práticas e diretrizes que auxiliem os pesquisadores a organizarem os dados, códigos e artefatos de suas publicações para assegurar a reprodução dos trabalhos. Além disso, várias ferramentas computacionais têm sido desenvolvidas com o objetivo de lidar com as questões de reprodutibilidade científica. No entanto, não existe uma solução que facilite a organização dos experimentos durante o desenvolvimento de uma pesquisa e permita a sua reprodutibilidade no contexto de criação de mapas de uso e cobertura da terra baseadas em técnicas de aprendizado de máquina. Este trabalho trata das questões de reprodutibilidade dos experimentos científicos realizados na criação de mapas de uso e cobertura da terra, através da apresentação de uma ferramenta computacional denominada `sits.rep`. Esta ferramenta permite a organização e reprodução de experimentos de classificação de uso e cobertura da terra com foco no pacote R denominado `sits` (*Satellite Image Time Series*).

Palavras-chave: Reprodutibilidade. Uso e Cobertura da Terra. Séries Temporais de Sensoriamento Remoto.

A TOOL FOR AUTOMATING THE REPRODUCIBILITY OF COMPUTATIONAL METHODS FOR LAND COVER CLASSIFICATION

ABSTRACT

Reproducibility research is a topic of great discussion in the scientific community. There are efforts to encourage researchers to share the datasets and computational methods used during a research, allowing other researchers to reproduce their work, not only for verification but also for a better understanding. This topic has motivated international journals to develop good practices documents and guidelines that help researchers to organize their data, codes, and artifacts of the publications to ensure the reproduction of their work. However, there is no solution that facilitates the organization of experiments during the research development nor a solution that allows reproducibility in the context of land use and land cover classifications based on machine learning techniques. This work deals with reproducibility issues of scientific experiments to create land use and land cover maps, through the presentation of a computational tool called *sits.rep*. This tool allows the organization and reproduction of land use and land cover classification experiments, focusing on the R package called *sits* (Satellite Image Time Series).

Keywords: Reproducibility. land use and land cover. Remote Sensing Time Series.

LISTA DE FIGURAS

Pág.

LISTA DE TABELAS

Pág.

LISTA DE ABREVIATURAS E SIGLAS

DOI	–	Digital Object Identifier
ERC	–	Executable Research Compendium
FAPESP	–	Fundação de Amparo à Pesquisa do Estado de São Paulo
FOSTER	–	Facilitate Open Science Training for European Research
IDSIA	–	Istituto Dalle Molle di Studi sull'Intelligenza Artificiale
IFGI	–	Institute for Geoinformatics - University of Munster
INPE	–	Instituto Nacional de Pesquisas Espaciais
JSON	–	JavaScript Object Notation
LXC	–	Linux container
MLP	–	Multilayer Perceptron
o2r	–	Opening Reproducible Research
PRNG	–	Pseudo-Random Number Generators
RERC	–	Reviewed Executable Research Compendium
SATVeg	–	Sistema de Análise Temporal da Vegetação
sits	–	Satellite Image Time Series
SOM	–	self-organizing maps
SVM	–	Support Vector Machine
URC	–	Unvalidated Research Compendium
URI	–	Uniform Resource Identifier
VM	–	Máquinas Virtuais
WTSS	–	Web Time Series Service

SUMÁRIO

Pág.

1 INTRODUÇÃO

A ciência proporcionou uma grande evolução no desenvolvimento das nações nos últimos séculos, o que gerou grande interesse por parte dos governos para investir em conhecimento como forma de solucionar problemas das sociedades atuais. Para Dahlman et al. (1998), a habilidade de inovação e criação de novos conhecimentos, sejam de origens técnicas ou teóricas derivadas das análises de dados, é o novo fator para assegurar o avanço das nações e o crescimento da economia. Essa questão tem se mostrado realidade nos dias de hoje, já que segundo Ranis et al. (2011) as comparações entre as nações estão cada vez menores entre recursos naturais e cada vez maior entre inovações, tecnologias e uso competitivo dos dados.

Um dos pilares da ciência para o desenvolvimento de pesquisas é a reprodutibilidade, tendo como propósito garantir que os resultados observados nos experimentos originais possam ser reproduzidos de forma independente (PENG, 2011; VASILEVSKY et al., 2013; MCNUTT, 2014). Este princípio permite que outros pesquisadores possam avaliar as premissas de uma metodologia proposta, podendo-se descobrir limitações ou até falhas, a partir das quais novos trabalhos podem ser desenvolvidos, avançando assim o conhecimento científico. Entretanto, alguns estudos apontam a existência de pesquisas científicas, publicadas após revisão por pares, com informações incompletas para a reprodutibilidade das mesmas (IOANNIDIS, 2005; PRINZ et al., 2011).

No trabalho de Baker (2016), foi realizado um questionário para descobrir a opinião de 1.576 pesquisadores sobre duas questões: se existe crise na reprodutibilidade das pesquisas científicas e se já tiveram problemas ao reproduzir algum experimento. Para a primeira questão, 52% relataram existir uma crise significativa no meio científico, enquanto somente 3% relataram não existir crise. Para a segunda questão, as respostas foram divididas por áreas de estudos em que os pesquisadores atuam. A área de Química foi a que obteve a maior porcentagem de declarações afirmativas por parte dos pesquisadores, com mais de 80% relatando que já falharam em reproduzir experimentos de outros trabalhos. Enquanto que mais de 60% disseram que já falharam em reproduzir os próprios experimentos. Especificamente sobre a área de Terra e Meio Ambiente, área foco deste trabalho, mais de 60% relataram que já falharam em reproduzir experimentos de outros trabalhos e mais de 40% falharam em reproduzir os próprios experimentos.

No estudo de Gertler et al. (2018), foi realizada a tentativa de replicação de 203 artigos científicos da área de economia publicados no mês de maio de 2016. O objetivo era analisar se os dados e códigos fornecidos em conjunto com as pesquisas seriam

suficientes para recriar os resultados demonstrados nos artigos, como as tabelas e figuras. Desse total, somente 37% foram possíveis de replicação. Em relação aos dados, somente 14% tinham todos os dados e códigos disponíveis para recriar as tabelas e figuras, enquanto que em 59% faltavam algum tipo de dado ou código. Em 3% dos artigos todos os dados estavam disponíveis, porém os códigos usados para gerar os experimentos não funcionaram. Em 24% dos artigos publicados não tinham dados ou códigos disponíveis.

Begley e Ellis (2012) relatam um caso ocorrido na empresa de biotecnologia Amgen, no qual os pesquisadores tentaram reproduzir 53 estudos considerados como históricos relacionados à hematologia e oncologia com o propósito de atestar as novas descobertas. Parte dos trabalhos selecionados descrevem uma nova metodologia, o que poderia alguns dados não se sustentarem. Contudo, somente 6 dos estudos foram possíveis de reprodução, deixando os próprios pesquisadores preocupados. Para os pesquisadores, os resultados não foram reproduzidos porque os dados não foram devidamente analisados por estudos cegos.

Estes estudos alegam falta de rigor técnico e transparência sobre os dados e métodos usados durante o desenvolvimento dos experimentos. Para lidar com esses problemas, a comunidade científica tem desenvolvido documentos de boas práticas[1], *frameworks* e ferramentas computacionais para auxiliar os pesquisadores a organizarem e compartilharem dados, códigos e demais artefatos de suas publicações (GENTLEMAN; LANG, 2007; NOSEK et al., 2015; GIL et al., 2016; BEAULIEU-JONES; GREENE, 2017; PIERRO, 2019; HEALTH et al., 2015). Alguns periódicos já obrigam os autores a depositarem dados e códigos em algum repositório público, de forma a reforçar a importância destas informações para possibilitar a reprodutibilidade das pesquisas.

1.1 Problema

A questão de se garantir a reprodutibilidade é inerente ao processo de desenvolvimento da ciência. Em busca de obter contribuições significativas, pesquisadores costumam explorar diferentes possibilidades em diferentes experimentos para que sejam encontrados resultados satisfatórios. Registrar todos os passos executados até se obter o resultado final requer um esforço de coleta de informações, onde quase todas serão descartadas, ficando apenas os experimentos que obtiveram os melhores resultados. Em experimentos *in silico*, até atitudes simples, como copiar o código

[1] Por exemplo, `https://www.nature.com/sdata/policies/data-policies` e `https://www.sciencemag.org/authors/science-journals-editorial-policies`.

usado por um experimento para o diretório onde os resultados são armazenados, normalmente são ignoradas.

Quando existem perdas de informações relativas ao processo empregado para produzir um determinado resultado, seja perder códigos usados para gerar o resultado de um experimento ou valor usado para inicializar de maneira semi-estocástica um cálculo ou modelo, dificilmente o experimento poderá ser reproduzido com exatidão. Isto pode inviabilizar todo e qualquer esforço executado na cadeia de experimentos posterior a ela. Esses problemas podem ter um impacto grande no processo de publicação das pesquisas científicas que possuem resultados gerados a partir desses dados que foram perdidos.

As ferramentas computacionais lidam com essas questões ao desenvolverem técnicas automatizadas para facilitar a reprodução dos experimentos desenvolvidos em uma pesquisa científica. Existem soluções genéricas, que consideram as questões de reprodutibilidade de maneira posterior à realização de todas as fases da pesquisa (CHIRIGATI et al., 2013; LANDAU, 2018; NÜST et al., 2017; GOVONI et al., 2019). Contudo, os processos envolvidos em uma pesquisa científica não são lineares, envolvendo diversas fases de experimentação que podem explorar diferentes métodos e parâmetros até que seja encontrado algum resultado interessante, que então será usado como resultado final para uma publicação científica. Se questões relativas à reprodutibilidade não forem consideradas *durante* a realização da pesquisa, dificilmente o resultado final publicado poderá ser reproduzido devido às dificuldades para preparação do ambiente, execução dos experimentos, ou até para encontrar a própria versão do código usada para produzir os resultados originais. Assim, é de suma importância tratar a reprodutibilidade durante o desenvolvimento da pesquisa, e não somente ao final do processo.

Existem ferramentas computacionais que buscam tratar das questões de reprodutibilidade durante a fase de experimentação (TOMMASO et al., 2017; GREFF et al., 2017). Estas ferramentas atendem às necessidades específicas de cada área de pesquisa, tais como diferentes formatos de dados, linguagens de programação específicas e o uso ou não de algoritmos estocásticos. Entretanto, essas ferramentas acabam impondo ao pesquisador um trabalho adicional significativo por não serem integradas ao ambiente de experimentação. Com base nisso, uma ferramenta para fins específicos e bem definidos, integrada ao ambiente de experimentação, pode facilitar o processo de reprodutibilidade, coletando todas as informações necessárias durante a execução das etapas de uma pesquisa.

A hipótese deste trabalho é que ferramentas específicas e integradas ao ambiente de experimentação facilitam o processo de reprodutibilidade. Desta forma, este trabalho apresenta uma ferramenta que esteja presente desde o início do desenvolvimento da pesquisa e que seja capaz de gerenciar o armazenamento dos códigos e dados relativos ao contexto do ambiente de desenvolvimento de maneira automatizada. A partir desses dados armazenados, a ferramenta é capaz de reproduzir os experimentos.

1.2 Objetivos

Na área de Observação da Terra são desenvolvidos diversos trabalhos com o intuito de monitorar e avaliar o ambiente natural e as mudanças causadas pelo homem[2]. A disponibilização aberta das imagens de sensoriamento remoto tem crescido significativamente, principalmente através da adoção de políticas de acesso aberto por instituições de pesquisa (SOILLE et al., 2018). Diante desse grande volume de dados, os pesquisadores enfrentam o desafio de projetar tecnologias que possam aproveitar todo o potencial desse conjunto de dados para realizar análises e auxiliar nas políticas públicas (CAMARA et al., 2016).

A partir dessa necessidade, um grupo de pesquisadores do Instituto Nacional de Pesquisas Espaciais (INPE) desenvolveu o pacote *Satellite Image Time Series* (`sits`) (CAMARA et al., 2018) que disponibiliza um conjunto de funcionalidades para manipular, analisar, visualizar e classificar séries temporais derivadas de imagens de satélites com o objetivo de produzir mapas de uso e cobertura da terra. Esse pacote promove o desenvolvimento iterativo e experimental, permitindo que pesquisadores apliquem diferentes métodos de classificação e pós-processamento dos resultados.

O pacote `sits` segue os princípios de `Open Science` ao disponibilizar todo código fonte[3] e os dados de amostras usados em suas pesquisas abertamente[4]. Entretanto, existe um interesse em aprimorar e aplicar conceitos de reprodutibilidade no pacote, facilitando a reprodução dos experimentos de classificação e análises de pós-processamentos sobre esses dados classificados.

Em geral, as pesquisas que usam o pacote `sits` envolvem grandes volumes de dados e atividades com grande complexidade, não existindo uma ferramenta que auxilie na reprodutibilidade desses experimentos. Portanto, esse pacote será usado como base para avaliar a hipótese proposta. O objetivo deste trabalho é desenvolver uma

[2] https://www.earthobservations.org/g_faq.html
[3] https://github.com/e-sensing/sits
[4] https://github.com/e-sensing/inSitu

ferramenta que atenda às questões de reprodutibilidade em pesquisas relacionadas à classificação de séries temporais de imagens de satélite focando no pacote `sits`.

1.3 Estrutura da dissertação

Esta dissertação está organizada da seguinte forma. O Capítulo 2 apresenta uma revisão da literatura com os conceitos de Ciência Aberta, definição da reprodutibilidade, ferramentas que atendem as necessidades reprodutíveis em áreas específicas e genéricas da ciência e apresentação do pacote `sits`. O Capítulo 3 apresenta a ferramenta desenvolvida neste trabalho, sob o nome de `sits.rep` e como um pacote para linguagem R. Além disso, é demonstrado a arquitetura geral do pacote e seus principais conceitos para diferenciar experimentos de classificação e pós-processamento.

O Capítulo 4 descreve o estudo de caso realizado para aplicar o pacote `sits.rep` em um caso real. Foi escolhida uma pesquisa publicada, a qual usa o pacote `sits` para classificar um conjunto de imagens de satélite para criação de mapas de uso e cobertura da terra. A proposta é recriar parte dos experimentos usando o `sits.rep` para verificar se o pacote facilita a reprodução dos experimentos. O Capítulo 5 conclui o trabalho e apresenta possíveis ideias para serem exploradas em trabalhos futuros.

2 REVISÃO DA LITERATURA

A reprodutibilidade científica é um requisito da ciência moderna. Seu principal objetivo é possibilitar que os resultados descritos em uma pesquisa possam ser reproduzidos por outros pesquisadores, aumentando a confiabilidade dos procedimentos usados e dos resultados obtidos (FREIRE et al., 2012; BEGLEY; IOANNIDIS, 2015; FEHR et al., 2016). As questões relativas à reprodutibilidade têm sido discutidas nas mais diversas áreas da ciência, como na psicologia (COLLABORATION, 2012; COLLABORATION et al., 2015), biologia (VAUX et al., 2012; VASILEVSKY et al., 2013), geociências (GIL et al., 2016; OSTERMANN; GRANELL, 2017; NÜST et al., 2018) e computação (GENTLEMAN; LANG, 2007; PENG, 2011; STODDEN; MIGUEZ, 2013; BOETTIGER, 2015). Existem algumas iniciativas que buscam facilitar o processo de reprodução das pesquisas, cada um atendendo demandas específicas de suas áreas. Este Capítulo apresenta os principais conceitos relativos à reprodutibilidade científica e as soluções mais recentes para lidar com essa questão.

2.1 Ciência aberta

O século XX mudou drasticamente a maneira como as informações são obtidas e compartilhadas entre as pessoas. No período pré internet, as informações poderiam demorar dias, semanas e até meses para serem difundidas pela sociedade, enquanto após a sua criação e disponibilização, permitiu compartilhar informações, com qualquer pessoa e de qualquer lugar, em questão de minutos. As pessoas no meio científico viram uma oportunidade de oferecer uma maneira de acesso aberto e irrestrito às informações produzidas pela acadêmia, criando um movimento posteriormente conhecido como *Open Science* (RESEARCH INFORMATION NETWORK NESTA, 2010).

O termo *Open Science* é relativamente novo e, por tanto, não há um consenso na literatura sobre a sua definição exata. Para Fecher e Friesike (2014), *Open Science* é uma palavra genérica que abrange cinco áreas de estudo para a criação e disseminação de conhecimento. Essas áreas possuem propósitos de: tornar os dados científicos abertos, compartilhar de maneira mais eficiente as informações, criar ferramentas e serviços livres para auxiliar os cientistas, tornar a ciência acessível em todos os níveis da sociedade e desenvolver medidas de impacto alternativas para contribuições científicas. Para Salmi (2015), *Open Science* é um movimento para tornar acessível a pesquisa científica e seus dados produzidos, disseminando as informações a todos os níveis da sociedade. Esse movimento representa uma ciência mais aberta e orientada por dados. As principais características são: a transparência na metodologia, observação e coleta dos dados, disponibilidade pública e reutilização dos dados cien-

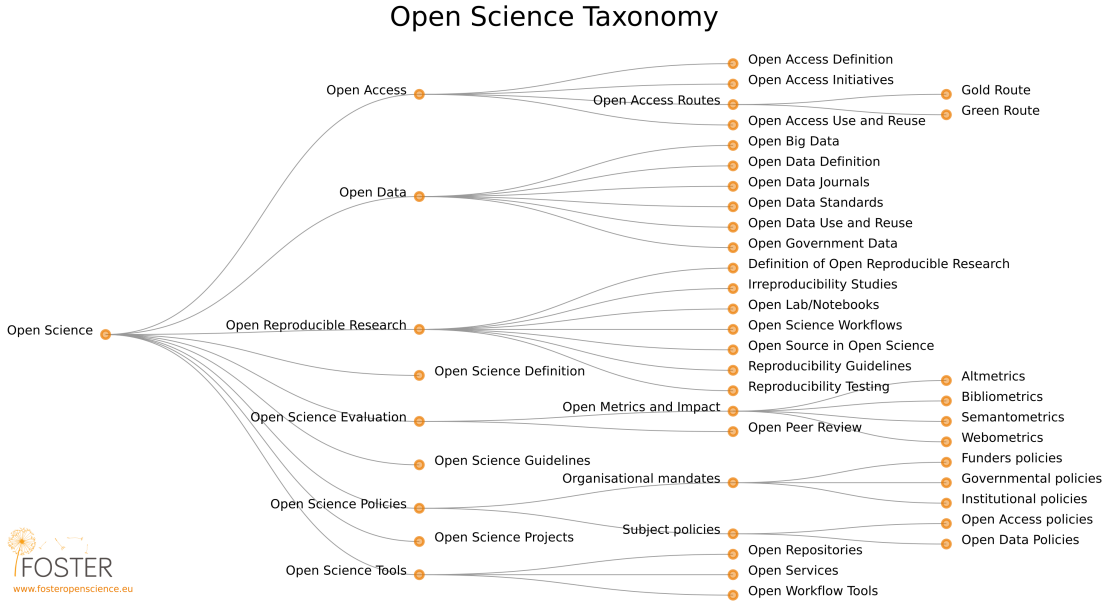
tíficos, acessibilidade pública e transparência da comunicação científica, e o uso de ferramentas baseadas em serviços para facilitar a colaboração científica. Para O'Neill et al. (2015), *Open Science* é a prática da ciência para permitir a colaboração e a contribuição no desenvolvimento das pesquisas. Todos os dados, notas de laboratório e processos das pesquisas estariam disponíveis para acesso gratuito. Devem ser possível reutilizar, redistribuir e reproduzir os métodos e dados das pesquisas. Para Vicente-Sáez e Martínez-Fuentes (2018), *Open Science* é um conhecimento transparente e acessível que é compartilhado e desenvolvido através de redes colaborativas.

Esse tema é tão importante que a Comissão Europeia financiou o desenvolvimento do Projeto FOSTER - *Facilitate Open Science Training for European Research* (FOSTER) com o propósito de capacitar cientistas com os princípios da ciência aberta (PONTIKA et al., 2015). Esse projeto desenvolveu uma taxonomia que define as áreas de domínio de *Open Science*, de forma a representar detalhadamente os seus conceitos e reduzir conflitos entre as diversas definições existentes. A taxonomia define nove grandes áreas: *Open Access*, *Open Data*, *Open Reproducible Research*, *Open Science Definition*, *Open Science Evaluation*, *Open Science Guidelines*, *Open Science Policies*, *Open Science Projects* e *Open Science Tools*. A Figura 2.1 demonstra essas áreas e seus subcampos.

Enquanto que existem similaridades em algumas definições, como o acesso aberto e livre aos dados científicos, outras possuem diferenças destacando as revisões por pares abertas e pesquisas científicas reprodutíveis. De qualquer modo, *Open Science* discute formas para tornar o acesso livre e gratuito a todos os conteúdos abordados nas pesquisas científicas. Além disso, estudam práticas para permitir a reprodutibilidade dos experimentos realizados durante e após o desenvolvimento das pesquisas (PONTIKA et al., 2015). Existem diversas plataformas que seguem essa filosofia para lidar com acesso aberto aos dados e metodologias produzidas na pesquisa, como Zenodo[1], Pangaea[2] e Dataverse[3], e que estão se consolidando como prioridades pelos periódicos e revistas para publicação de artigos científicos. Para lidar com a reprodutibilidade, atualmente existem ferramentas para reproduzir experimentos específicos de áreas de interesse e verificar de maneira automatizada as análises finais das pesquisas. Os estudos relacionadas a reprodutibilidade, seja na criação de ferramentas ou desenvolvimento de boas práticas, são agrupados pela taxonomia na área de *Open Reproducible Research*, demonstrado na Figura 2.1.

[1] https://zenodo.org/record/3188334#.XOswCibQ85k
[2] https://doi.pangaea.de/10.1594/PANGAEA.901178
[3] https://dataverse.org/

Figura 2.1 - Taxonomia definida pelo projeto FOSTER que define nove campos da área de *Open Science*.



Fonte: Pontika et al. (2015).

2.2 Definições de reprodutibilidade e replicabilidade

Na literatura, não existe um consenso para uma definição exata dos termos de reprodutibilidade e replicabilidade e seus usos nas pesquisas científicas. Em muitos casos, os autores usam esses termos para descrever o mesmo fenômeno, não apresentando uma consistência entre as terminologias (GOODMAN et al., 2016; PLESSER, 2018; BOUTHILLIER et al., 2019). Inclusive, o uso desses termos varia historicamente para cada área da pesquisa, com os autores usando um termo para descrever o outro (NATIONAL ACADEMIES OF SCIENCES ENGINEERING MEDICINE, 2019).

No trabalho de Barba (2018), foi realizada uma análise em diversas publicações, de diferentes campos que utilizam esses termos, e conseguiu classificar os diferentes usos dessas terminologias em três grupos: A, B1 e B2. As definições desses grupos são mais claras e expandidas no trabalho de NATIONAL ACADEMIES OF SCIENCES ENGINEERING MEDICINE (2019), e serão apresentadas a seguir:

A: As palavras Reprodutibilidade e Replicabilidade não possuem distinção, sendo usados com o mesmo significado.

B1: A palavra Reprodutibilidade é usada para descrever os casos em que são usados os mesmos dados e métodos da pesquisa para obter os mesmos resultados. Já a Replicabilidade, é para casos em que são aplicados novos dados nos mesmos métodos para obter os mesmos resultados.

B2: A palavra Reprodutibilidade é usada nos casos em que são obtidos os mesmos resultados através dos dados e métodos diferentes do original. Já a Replicabilidade, é para casos em que são obtidos os mesmos resultados através dos métodos e dados originais da pesquisa.

As definições descritas nos grupos B1 e B2 são opostas entre si, e seus usos irão depender de cada área da ciência. Segundo Barba (2018) e NATIONAL ACADEMIES OF SCIENCES ENGINEERING MEDICINE (2019), as terminologias do grupo B1 são frequentemente aplicadas nas áreas de Processamento de Sinal e Computação Científica, enquanto do grupo B2 é utilizadas nos campos de Microbiologia, Imunologia e Ciência da Computação. Para este trabalho, a terminologia usada para a definição de reprodutibilidade será a B1.

2.3 Ferramentas para reprodutibilidade

Esta seção apresenta as ferramentas computacionais encontradas tanto na academia quanto na indústria, que buscam gerenciar a organização dos dados e metadados produzidos pelos experimentos durante ou depois do desenvolvimento da pesquisa. A seguir, serão apresentadas as seguintes ferramentas: *Executable Research Compendium* (ERC), *QResp*, *GenePattern*, *Sacred* e *Neptune*.

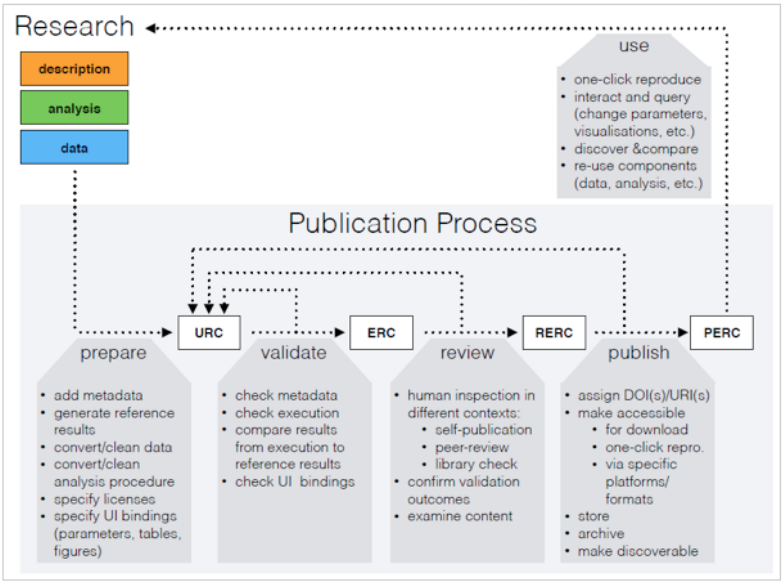
2.3.1 Executable Research Compendium

Nüst et al. (2017) propõem o *Executable Research Compendium* (ERC), uma especificação para empacotar pesquisas computacionais a fim de reproduzir os resultados de pesquisas. O ERC combina dados, *software* (código fonte e ferramentas auxiliares), documentos (metadados e publicações científicas) e uma descrição da interface do usuário. Estas informações são materializadas através de um ambiente de virtualização. Os autores também especificam um processo de publicação que pode ser usado por revistas científicas para disponibilizar tanto a publicação quanto uma interface para reproduzir pesquisas usando o ERC.

O processo de publicação foi desenvolvido para atender os interesses de seus *stakeholders*, como: autores das pesquisas, usuários da plataforma, revisores, periódicos acadêmicos, bibliotecários, curadores e preservacionistas. Esse processo foi imple-

mentado através do projeto o2r, originando em uma plataforma de mesmo nome. As cinco etapas para publicação de um ERC são apresentadas na Figura 2.2.

Figura 2.2 - Fluxograma do Processo de Publicação de um ERC.



Fonte: Nüst et al. (2017).

A primeira etapa, chamada de *prepare*, inicia com o pesquisador organizando seu ambiente de desenvolvimento. Não há nenhuma restrição de como serão organizados os resultados, códigos fonte e dados de entrada precisando somente que contenha um *script* principal, com extensão R ou RMarkdown, e um arquivo de visualização que é a saída desse *script*. O *script* principal é usado para o pesquisador demonstrar o seu trabalho de pesquisa, interligando textos com trechos de código que retornam os resultados das análises, como um gráfico ou tabela, seguindo o conceito de *Literate programming* proposto por Knuth (1984). As licenças de uso para códigos e dados podem ser descritas em um arquivo de configuração.

O pesquisador deve criar um arquivo contendo a especificação do ambiente computacional. Para isso, é usado a plataforma de virtualização Docker[4], para que o pesquisador descreva as ferramentas, sistema operacional e pacotes através de um arquivo texto no formato Dockerfile, para permitir que os usuários da plataforma executem as análises e verifiquem se os resultados condizem com aqueles descritos no texto. Finalizado a organização e a criação dos arquivos, o pesquisador compacta

[4] https://www.docker.com

o ambiente de desenvolvimento no formato ZIP resultando em um *Unvalidated Research Compendium* (URC).

Na segunda etapa, o pesquisador irá submeter o URC para um serviço que irá validar os arquivos principais, de configuração, licenças e as partes reprodutíveis da pesquisa criando um contêiner através do Dockerfile e executando as análises. Caso não ocorra nenhum erro e divergência nos dados da análise, é criado um ERC.

Na terceira etapa, é desempenhada uma inspeção humana para garantir que não ocorra nenhuma publicação de *compendium* questionáveis com conteúdo ofensivo, ilegal e fora dos padrões exigidos por uma instituição ou revista. Essa inspeção pode ser realizada por bibliotecários, gerenciadores de repositórios de dados, editores de revistas científicas e revisores por pares. Concluído o processo de revisão, é criado um *Reviewed Executable Research Compendium* (RERC).

Na quarta etapa, o RERC recebe um *Digital Object Identifier* (DOI)/*Uniform Resource Identifier* (URI), permitindo que seja acessível para *download* e reprodução das análises como um *Published Executable Research Compendium* (PERC) e encerrando o processo de publicação.

Na quinta etapa, os usuários do serviço podem acessar a pesquisa e reproduzir os resultados de um PERC e comparar com os resultados da pesquisa original através de um clique no botão na interface. A vantagem no botão é a reprodução dos resultados de maneira automatizada com a mínima intervenção do usuário.

O projeto o2r é de código aberto, disponível na plataforma GitHub[5]. Tanto o ERC quanto o projeto o2r são desenvolvidos pelo *Institute for Geoinformatics* e *University and Regional Library* em Münster, Alemanha.

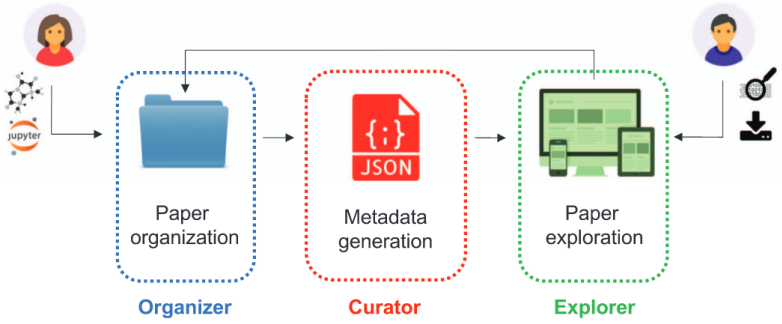
2.3.2 QResp

O QResp (GOVONI et al., 2019) é uma aplicação web de código aberto para cadastros e buscas de dados e processos de pesquisas científicas desenvolvida pela *University of Chicago* e *Argonne National Laboratory*, nos Estados Unidos. Diferentemente do o2r em que a sua reprodutibilidade está na execução das análises e comparação dos resultados através de um ambiente de virtualização, a reprodutibilidade no QResp está na disponibilização de todo o conteúdo desenvolvido na pesquisa, como um texto científico, dados de entrada, resultado e ferramentas, a fim de permitir que os

[5] https://github.com/o2r-project

usuários possam reproduzir com facilidade seus resultados. A aplicação possui três módulos principais: organização, curadoria e exploração. A Figura 2.3 apresenta a arquitetura do QResp, com seus três módulos principais.

Figura 2.3 - Arquitetura da aplicação QResp.



Fonte: Govoni et al. (2019).

No módulo de organização, o pesquisador prepara um servidor próprio com todos os *scripts*, dados, ferramentas auxiliares e documentos da pesquisa. O QResp não armazena os arquivos em seus servidores, por tanto é atribuída ao pesquisador a função de disponibilizar o acesso ao conteúdo através de um servidor via protocolo SSH. Não há uma estrutura de organização padrão para os arquivos. Para permitir que os arquivos possam ser baixados pelos usuários na etapa de exploração, o autor aconselha instalar a aplicação Globus[6] no servidor que irá implementar essa funcionalidade via gridFTP.

No módulo de curadoria, a interface gráfica irá auxiliar o pesquisador a criar um metadado que descreve todo o conteúdo da pesquisa. Inicialmente é informado o endereço do servidor que o pesquisador dispôs no módulo de organização, juntamente com o usuário e senha para o acesso via SSH. Após estabelecida a conexão, é feita a listagem dos arquivos para o pesquisador descrever a origem dos dados (entrada e resultado), os detalhes da publicação, as ferramentas auxiliares e *notebooks*. O QResp oferece a opção de criar um *workflow* para descrever os procedimentos que geraram os resultados analisados na pesquisa científica, a fim de, segundo o autor, explicar de forma detalhada e compacta as estratégias utilizadas. A aplicação cria o metadado em formato JavaScript Object Notation[7] (JSON) e armazena nos servidores do QResp, permitindo a sua consulta na fase de exploração.

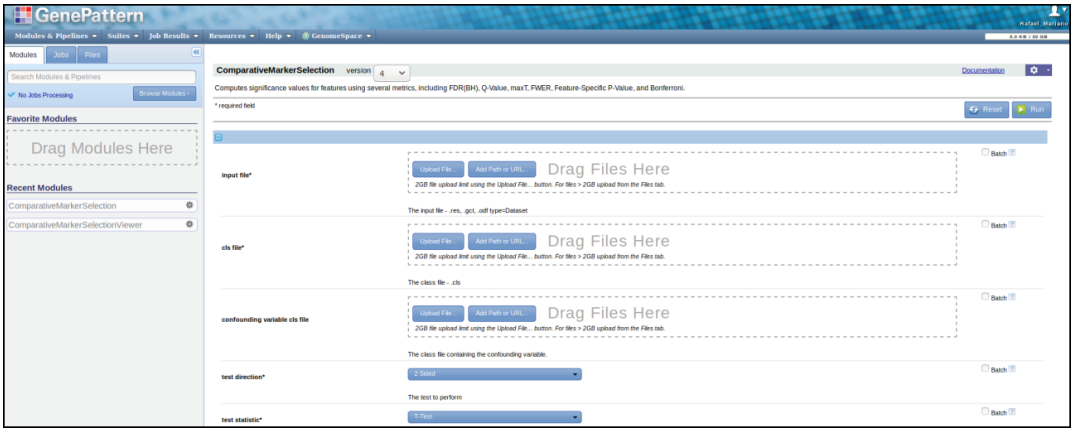
[6] https://www.globus.org/
[7] http://json.org/

No módulo de exploração, os usuários podem consultar as pesquisas científicas cadastradas através do nome do autor principal e do DOI. Os dados como figuras, tabelas, *scripts*, *notebooks*, *workflow* e texto publicados podem ser visualizados e baixados para o computador do usuário.

2.3.3 GenePattern

GenePattern[8] (REICH et al., 2006) é uma plataforma web para reproduzir análises de bioinformática. Ela fornece um conjunto de módulos pré configurados e validados capazes de realizar a formatação dos dados, pré-processamento, análises e visualização dos resultados. Todos os processos de análise e desenvolvimento dos experimentos são realizados através da interface web que permite aos pesquisadores escolher os módulos, alterar seus parâmetros e inserir dados de entrada. O processamento dessas análises é executado no servidor onde o GenePattern encontra-se hospedado. Na Figura 2.4 é demonstrado o uso do módulo *ComparativeMarkerSelection*, que tem o objetivo de encontrar os genes mais correlacionados entre dois conjuntos de dados. O GenePattern exibe os parâmetros para que o pesquisador insira os valores correspondentes e solicite a execução da análise no servidor. É possível visualizar e transferir os resultados para a máquina local, além de permitir a utilização desse resultado como entrada para um outro módulo.

Figura 2.4 - Interface do módulo *ComparativeMarkerSelection*.



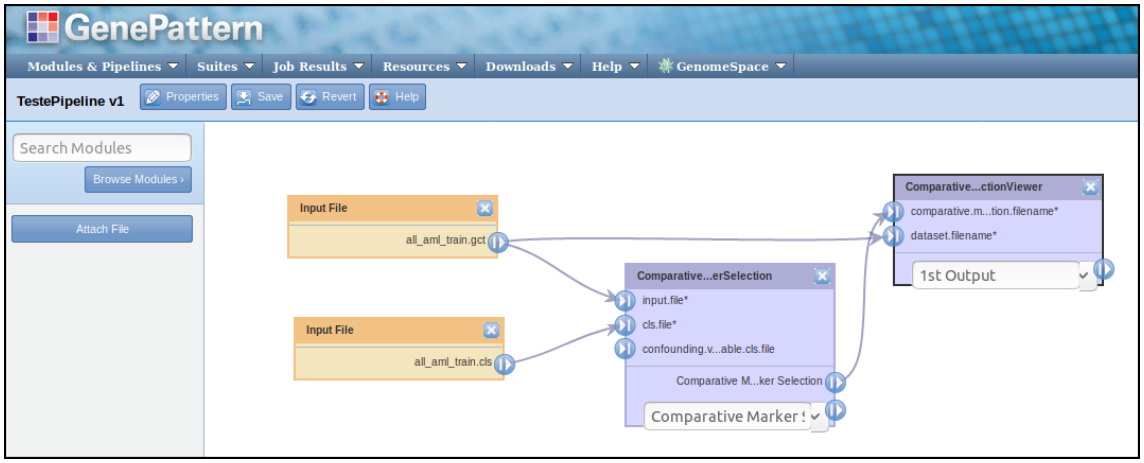
Fonte: Reich et al. (2006).

[8] http://software.broadinstitute.org/cancer/software/genepattern#

A plataforma suporta a reprodutibilidade através do uso de *pipelines* que permitem executar toda a metodologia da pesquisa computacional em um módulo único e executável. Os *pipelines* são desenvolvidos pelos pesquisadores por meio da interface web, encadeando os dados e as saídas dos módulos com as respectivas entradas dos outros módulos. Em cada execução são armazenados metadados referentes aos dados de entrada, módulos, parâmetros e resultado para facilitar a reprodução e a comparação dos experimentos realizados. É possível versionar os *pipelines* alterando os seus parâmetros, dados e módulos. Os pesquisadores podem compartilhar e baixar os *pipelines* através do repositório GParc[9], porém somente o administrador pode instalar novos *pipelines* no GenePattern. A Figura 2.5 apresenta a interface para criação de *pipelines*, com o módulo *ComparativeMarkerSelection* recebendo como entrada dois conjuntos de dados previamente definidos e o seu resultado usado como parte da entrada do módulo *ComparativeMarkerSelectionViewer*. Esse módulo irá criar uma visualização dos genes mais correlacionados. Ao salvar esse *pipeline*, é possível solicitar sua execução ao servidor que irá realizar chamadas internas sem intervenção do usuário, automatizando o processamento da análise.

O GenePattern disponibiliza uma API REST para que os pesquisadores possam acessar os seus módulos através das linguagens de programação e combinar o uso da plataforma com outras bibliotecas de bioinformática. A ferramenta está disponível para instalação local ou acesso via web (REICH et al., 2006). O GenePattern foi financiado pelo programa de tecnologia informática para pesquisa do câncer do *National Cancer Institute* e *National Institute of General Medical Sciences*.

Figura 2.5 - Interface para o desenvolvimento de *pipelines* na plataforma GenePattern.



Fonte: Reich et al. (2006).

[9] http://www.gparc.org/

2.3.4 Sacred

Sacred (GREFF et al., 2017) é um *framework* de código aberto[10] desenvolvido na linguagem Python com o objetivo de configurar, organizar e registrar os experimentos computacionais envolvendo algoritmos de aprendizagem de máquina. O *Sacred* é desenvolvido no *Istituto Dalle Molle di Studi sull'Intelligenza Artificiale* (IDSIA), em Manno, Suiça. Segundo os autores, há uma dificuldade em reproduzir os experimentos que utilizam algoritmos de aprendizado de máquina em razão de exigir diversas execuções dos experimentos para diferentes valores dos hiperparâmetros. Para enfrentar esse problema, o *Sacred* permite que o desenvolvedor introduza funções e decoradores no código fonte para que em cada execução dos experimentos, as suas informações relevantes sejam capturadas e armazenadas automaticamente, e posteriormente serem visualizadas através de uma interface web as informações de cada experimento.

Para coletar as informações de cada experimento, inicialmente é preciso instanciar a classe `Experiment` e decorar uma função principal para inicializar a execução. No *script* da Figura 2.6, foi decorado na linha 16 a função principal `run` com `@ex.automain` que irá executar uma classificação dos dados de Íris com a rede neural *Multilayer Perceptron* (MLP). Em cada execução, as informações de dados de entrada, resultado, código fonte, pacotes de dependências, sistema operacional e a semente usada para a geração de números aleatórios são coletadas.

Um dos problemas em reproduzir os resultados de *scripts* que utilizam aprendizagem de máquina é o seu fator semi-estocástico. A sua estocasticidade é adquirida através do algoritmo *Pseudo-Random Number Generators* (PRNG), que gera uma sequência de números aleatórios a partir de um valor inicial conhecido como a semente da execução (*seed*). Como o *seed* precisa ser determinado, é possível reproduzir os resultados dos experimentos anteriores caso o seu valor seja conhecido. Por tanto, *Sacred* permite fixar um valor para o *seed* e armazenar esse dado como parte da configuração de cada experimento.

Os hiperparâmetros de um algoritmo de aprendizagem de máquina não são coletados automaticamente quando estão dentro da função principal, precisando inicializar os hiperparâmetros em uma função separada com o decorador de configuração. Na linha 9 do *script* na Figura 2.6, é demonstrado o seu uso com a função `cfg` decorado com o `@ex.config`, inicializando os hiper parâmetros da MLP.

[10] `https://github.com/IDSIA/sacred`

Figura 2.6 - *Script* para classificação do conjunto de dados Íris utilizando a rede neural *Multilayer Perceptron* com anotações do pacote *Sacred*.

```
1  from sacred import Experiment
2  from sacred.observers import MongoObserver
3  from import_lib import MLPClassifier, numpy as np, datasets, validation_curve
4
5  ex = Experiment()
6  ex.observers.append(MongoObserver.create
7                          (url = 'localhost:27017', db_name = 'sacred'))
8  @ex.config
9  def cfg():
10     solver = 'lbfgs'
11     activation = 'relu'
12     hidden_layer = (10,)
13
14 @ex.automain
15 def run(solver, activation, hidden_layer, _run):
16     clf = MLPClassifier(solver = solver, activation = activation, hidden_layer_sizes = hidden_layer)
17
18     data = datasets.load_iris().data
19     target = datasets.load_iris().target
20
21     train, test = validation_curve(clf, data, target, cv=5, n_jobs = 1,
22         param_name = "epsilon", param_range = np.logspace(-6, -1, 5), scoring = "accuracy")
23
24     train = np.reshape(train, (1, np.size(train)))[0]
25     test = np.reshape(test, (1, np.size(test)))[0]
26     for cont in range(test.size):
27         _run.log_scalar("train_scores", train[cont])
28         _run.log_scalar("test_scores", test[cont])
```

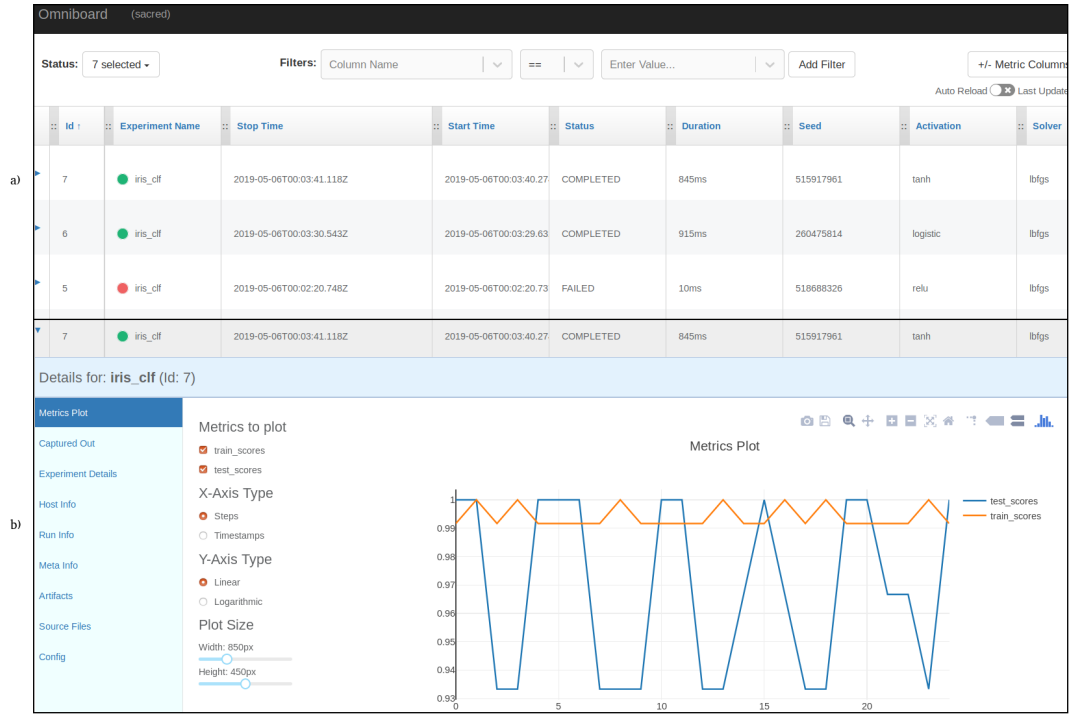
Fonte: Adaptado de Greff et al. (2017).

As informações coletadas podem ser armazenadas em um banco de dados relacional, não relacional ou em arquivos. Caso o desenvolvedor opte por armazenar no banco de dados não relacional MongoDB, é possível visualizar em uma interface web as informações de todas as execuções dos experimentos. Uma dessas interfaces web disponíveis é o *Ominiboard*[11], um ferramenta *front-end* web para facilitar a visualização dos experimentos e métricas coletados pelo *Sacred*.

A Figura 2.7 (a) demonstra a interface principal do Ominiboard com todos os experimentos armazenados no MongoDB. É possível visualizar o momento inicial e final da execução do experimento, se houve falha na execução, a semente utilizada para a geração de números aleatórios, a função de ativação, entre outros, além de permitir a ordenação por atributo. Outros dados como o código fonte, o resultado de saída, sistema operacional e dependências é possível visualizar individualmente, como demonstrado na Figura 2.7 (b). É possível enviar dados para serem plotados em um gráfico através do objeto `_run`, inicializado na linha 17 e usado nas linhas 31 e 32 do *script* da Figura 2.6.

[11] https://github.com/vivekratnavel/omniboard

Figura 2.7 - (a) Interface principal da ferramenta Ominiboard. Os experimentos são disponibilizados para consulta de seus dados. (b) Ao clicar em um experimento, é possível analisar um gráfico de métricas e visualizar os metadados, como: dependências, sistema operacional, código fonte e o resultado de saída da execução.



Fonte: Adaptado de Subramanian e Sako (2018).

2.3.5 Neptune

Neptune (HRYCIUK, 2016)[12] é uma ferramenta de código fechado desenvolvido pela empresa deepsense.ai para o acompanhamento e organização de projetos que envolvem experimentos computacionais de aprendizagem de máquina. Ele é composto por um pacote da linguagem Python, para capturar as informações dos experimentos, e uma aplicação web, para visualizar e analisar as informações coletadas. Os diferenciais do *Neptune* são permitir a criação de equipes para o desenvolvimento dos experimentos em um ambiente colaborativo e armazenamento dos dados na nuvem em um repositório próprio.

[12] https://ui.neptune.ml/

A Figura 2.8 exemplifica a utilização da biblioteca Python do *Neptune* com uma classificação dos dados Íris utilizando uma rede neural MLP. A construção dos experimentos é similar à realizada no *Sacred*, com a diferença de que é necessário um *token* de acesso e identificação para o armazenamento dos dados na nuvem. Esse *token* é solicitado pelo pesquisador na interface web. As informações de sistema, dependências e a semente da execução não são capturados automaticamente pela ferramenta. Para isso, é necessário utilizar funções da própria biblioteca para o envio de dados, como é demonstrado na linha 16 com o envio dos dados de acurácia.

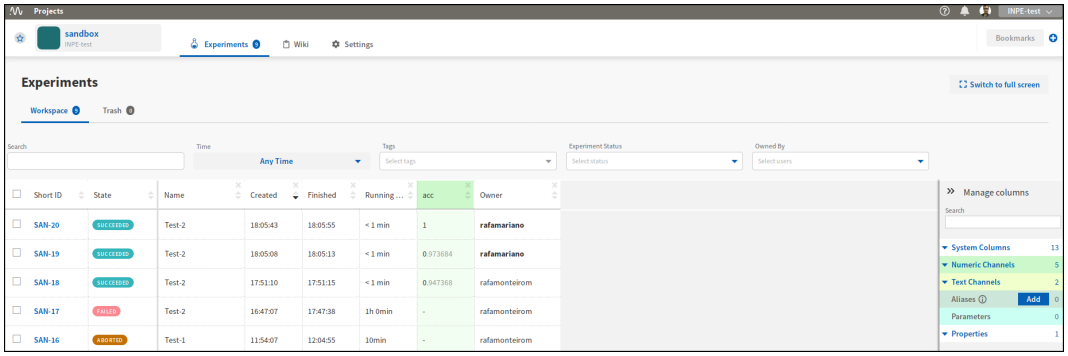
Figura 2.8 - *Script* para classificação do conjunto de dados Íris utilizando a rede neural *Multilayer Perceptron* usando a biblioteca *Neptune*.

```
1  import neptune
2  from import_lib import MLPClassifier, load_iris, \
3      train_test_split, accuracy_score
4
5  neptune.init(api_token='token',
6               project_qualified_name='INPE-test/sandbox')
7
8  neptune.create_experiment('Test-2')
9
10 iris = load_iris()
11 X_train, X_test, y_train, y_test = train_test_split(iris.data, iris.target)
12
13 mlp = MLPClassifier(hidden_layer_sizes=(13,13,13),max_iter=500)
14 mlp.fit(X_train,y_train)
15
16 neptune.send_metric('acc', accuracy_score(y_test, mlp.predict(X_test)))
17 neptune.stop()
```

Fonte: Deepsense.ai (2018).

A interface web permite criar projetos e visualizar seus experimentos. Todas as informações enviadas pelo *script* podem ser visualizadas na interface principal e ao acessar o experimento individual é possível baixar o código fonte, obter acesso aos seus metadados e visualizar gráficos com dados de métricas. A Figura 2.9 demonstra a interface web do *Neptune* em um projeto com dois usuários contribuindo com os experimentos. Os usuários podem estar associados a dois papéis: membro ou administrador. Um usuário associado ao papel membro pode desenvolver e contribuir com os experimentos do projeto e visualizar as informações. O papel administrador permite remover experimentos, adicionar novos membros, alterar os níveis de permissão, além das atividades realizadas por um usuário do nível membro.

Figura 2.9 - Interface web da ferramenta *Neptune*.



Fonte: Deepsense.ai (2018).

O *Neptune* é gratuito para uso com limite de três colaboradores e 5 GB de dados. Para uso por projetos com equipes maiores, que demandem mais recursos computacionais, é necessário adotar um plano comercial através de uma assinatura mensal.

2.4 Satellite Image Time Series (sits)

O `sits` é um pacote desenvolvido na linguagem R e que faz parte de uma iniciativa aberta com o propósito de disponibilizar ferramentas para visualização, suavização, clusterização e classificação de séries temporais (CAMARA et al., 2018). Para as classificações, são utilizadas amostras de dados *in situ*, juntamente com as respectivas séries temporais destas amostras e algoritmos de aprendizagem de máquina, como o *Support Vector Machine* (SVM) e MLP. O grupo de pesquisa responsável pelo desenvolvimento do `sits` armazena todas as amostras de treinamento relevantes no pacote `inSitu`[13]. Da mesma forma, os dados de séries temporais usados para a classificação estão disponíveis através do pacote `EOCubes`[14].

Este pacote suporta recuperação de séries temporais através de serviços web, como *Web Time Series Service* (WTSS) (QUEIROZ et al., 2015), Sistema de Análise Temporal da Vegetação (SATVeg) da EMBRAPA [3], bem como de arquivos matriciais. A Tabela 2.1 exemplifica essa organização com as seis primeiras colunas referentes aos metadados e a última coluna para os dados. As duas primeiras colunas estão relacionadas à localização espacial da observação em coordenadas geográficas de longitude e latitude para o elipsoide WGS84. A terceira e a quarta coluna são referentes

[13] https://github.com/e-sensing/inSitu
[14] https://github.com/e-sensing/EOCubes
[3] https://www.satveg.cnptia.embrapa.br/satveg/login.html

ao intervalo temporal da observação. A coluna *label* atribui a classe da amostra e a *coverage* indica o produto que produziu aquela informação. E, por fim, a coluna *time_series* contém os dados brutos das bandas ou índices espectrais referentes à observação.

Tabela 2.1 - Estrutura de dados trabalhada pelo pacote sits.

lon	**lat**	**start_date**	**end_date**	**label**	**coverage**	**time_series**
<dbl>	<dbl>	<date>	<date>	<chr>	<chr>	<list>
-57.8	-9.76	2006-09-14	2007-08-29	Pasture	MOD13Q1	<tibble [23...
-55.2	-10.8	2013-09-14	2014-08-29	Pasture	MOD13Q1	<tibble [23...
-51.9	-13.4	2014-09-14	2015-08-29	Pasture	MOD13Q1	<tibble [23...

Para trabalhar com classificações, é necessário obter amostras de séries temporais no formato de dados descrito na Tabela 2.1. Um conjunto de amostras é usado para treinar um algoritmo de aprendizado de máquina, cujo resultado é usado para classificar séries temporais para uma determinada área de estudo. O conjunto de classes definido pelas amostras será o mesmo usado para produzir a legenda final da classificação. Para cada `pixel`, o algoritmo de classificação produz uma probabilidade daquele `pixel` pertencer a cada uma das classes em cada intervalo de tempo. Tipicamente, a classe com maior probabilidade em cada `pixel` é escolhida como classificação final. Ao final do processo, mapas de uso e cobertura da terra são produzidos para todos os intervalos de tempo disponíveis. Estes intervalos de tempo geralmente estão associados ao calendário agrícola, possuindo uma periodicidade anual. A Figura 2.10 demonstra um exemplo de código para treinamento e classificação de um conjunto de séries temporais usando o pacote `sits`. Na linha 3 é importado a base de dados de amostras que será usada para treinamento. Esses dados foram produzidos pela EMBRAPA para o Estado do Mato Grosso. Em seguida na linha 5, é definido os parâmetros para construção do modelo de rede neural MLP usando técnicas de `Deep Learning`. Na linha 12 é treinado esse modelo com a base de amostras da EMBRAPA. Na linha 15 é classificado uma série temporal de um ponto usando o modelo treinado anteriormente.

Figura 2.10 - *Script* de classificação do Mato Grosso utilizando *Deep Learning* com a rede neural MLP.

```r
library("sits")

data(samples_mt_ndvi)
# train a machine learning model using deep learning
train_dl <- sits_deeplearning(
  units  = c(512, 512),
  activation = "elu",
  dropout_rates = c(0.40, 0.30),
  optimizer = keras::optimizer_adam(lr = 0.001),
  epochs = 50)

dl_model <- sits_train(samples_mt_ndvi, train_dl)
# get a point to be classified
data(point_ndvi)
class.tb <- sits_classify(point_ndvi, dl_model)
sits_plot(class.tb)
```

Fonte: Produção do Autor.

Atualmente, o grupo de pesquisa responsável pelo desenvolvimento do `sits` considera que todas as amostras de treinamento relevantes estão armazenadas no pacote `inSitu`[1]. Da mesma forma, todos os dados de séries temporais usados para a classificação estão disponíveis através do pacote `EOCubes`[2]. Ambos pacotes são desenvolvidos por esse grupo de pesquisa.

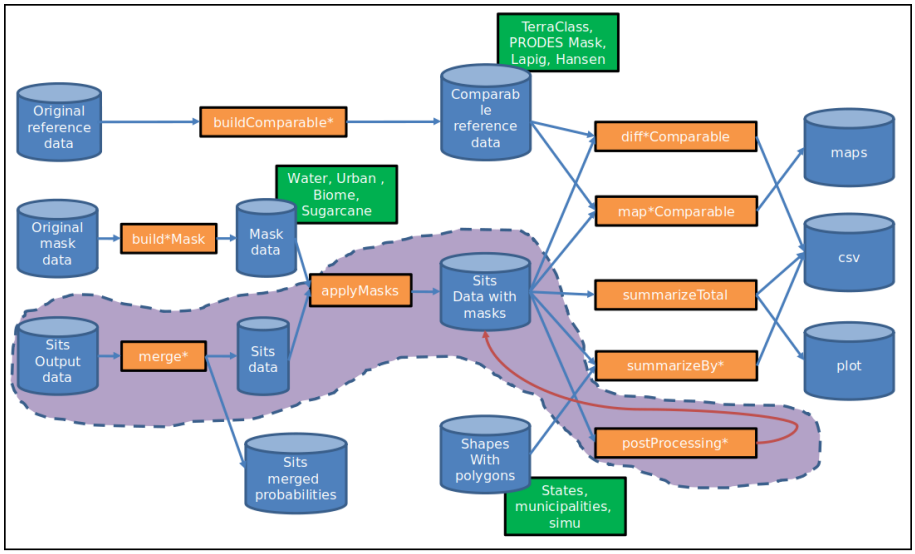
Após as classificações, é realizado um pós-processamento onde são aplicados:

- Máscaras para identificação de áreas específicas tais como água e áreas urbanas. Este procedimento introduz novos elementos à legenda usada na classificação.
- Filtros espaciais com o objetivo de suavizar as classificações. Por exemplo, um `pixel` classificado como pasto, cercado de `pixeis` de soja, sendo que a probabilidade de pasto neste `pixel` ficou muito próxima da classe soja, terá a sua classe alterada para soja ao aplicar um filtro espacial.
- Regras de transição, para remover inconsistências temporais. Por exemplo, a transição de floresta para pasto e novamente para floresta, não é possível. Uma vez que a classe floresta uma vez desmatada nunca mais volta à sua condição original. Neste caso, novas classes podem ser introduzidas na legenda da classificação.

[1] https://github.com/e-sensing/inSitu
[2] https://github.com/e-sensing/EOCubes

O processo de pós-processamento é experimental, podendo envolver a aplicação de diferentes passos sobre um mesmo conjunto de dados, de forma a entender ou melhorar os resultados das classificações. Desta forma, a criação de informações após as classificações segue um fluxo que pode conter bifurcações. Ao final deste processo, apenas um caminho será escolhido como a classificação final. A Figura 2.11 ilustra esse fluxo experimental do pós-processamento utilizando o pacote `sits.validate`, que é usado para pós-processar e validar as classificações, bem como agregar as saídas em diferentes representações. Os dados estão em azul, enquanto os *scripts* estão em laranja. As setas indicam os fluxos de dados, usados como entrada ou disponibilizados como saídas dos *scripts*. Em verde são mostrados os dados primários de diferentes fontes que estão preparados para serem usados como máscaras nas classificações do `sits`. A parte da figura destacada em roxo, representa os pós-processamentos desenvolvidos para tratar diretamente as classificações. Cada *script* cria novos resultados que podem ser usados como entrada para o próximo algoritmo de pós-processamento.

Figura 2.11 - Processo de pós-processamento e validação das classificações do `sits` usando classificações visuais como referência.



Fonte: Andrade (2018).

Esse pacote foi desenvolvido dentro do projeto e-sensing[15], fomentado pela Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) e executado por uma equipe multidisciplinar do INPE.

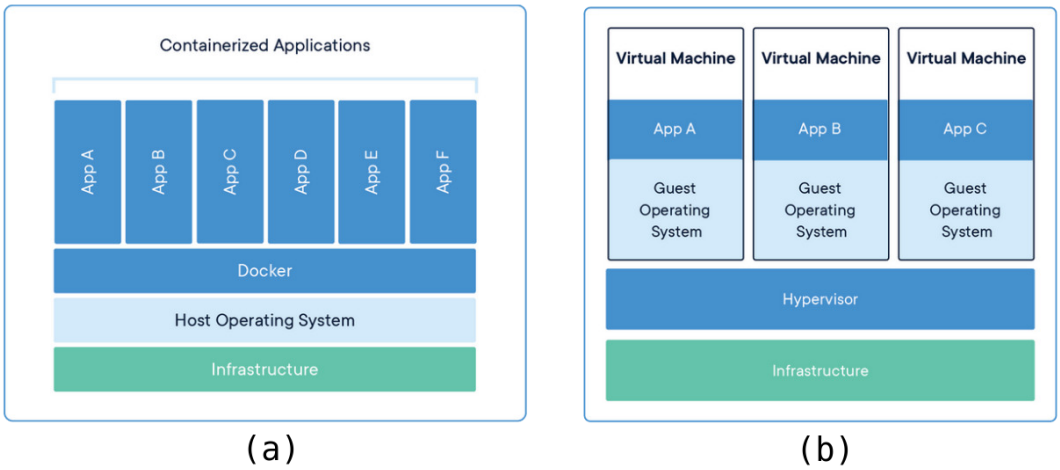
2.5 Docker para pesquisas reprodutíveis

Docker é uma ferramenta de código aberto para virtualização de ambientes computacionais. É usado a tecnologia `Linux container` (LXC) a fim de usar os recursos disponíveis pelo kernel para isolar os processos. Dessa forma, é possível empacotar arquivos, aplicações e até mesmo ambientes completos em um ambiente isolado, chamado de `container`, e executá-los de maneira independente. Além disso, com o Docker é possível flexibilizar o gerenciamento das aplicações de uma infraestrutura computacional, permitindo executar e migrar diversos ambientes separadamente que se comunicam entre si. Caso um ambiente falhe, basta derrubar um `container` e subir um novo criando maior segurança, estabilidade e facilidade para gerenciar as aplicações (BOETTIGER, 2015; GOMES, 2017; DOCKER, 2021c; DOCKER, 2021a).

A Figura 2.12 apresenta a arquitetura do modelo de virtualização adotado pelo Docker e as Máquinas Virtuais (VM) convencionais. Na coluna `(a)` o módulo Docker está intermediando a comunicação entre o kernel e as aplicações diretamente, compartilhando os recursos já disponíveis na máquina hospedeiro com o `container` tornando esse meio de virtualização leve para o sistema. Já na coluna `(b)` a tecnologia `Hypervisor` que está intermediando a comunicação, criando ambientes isolados com sistemas operacionais próprios. Esse tipo de metodologia, apesar da eficácia para virtualização de ambientes, exige uma quantidade de espaço de armazenamento maior pois exige a inclusão de sistemas operacionais inteiros. Além disso, cria um uso maior dos recursos disponíveis na infraestrutura.

[15] `http://www.esensing.org`

Figura 2.12 - Modelos de arquitetura de virtualização adotado pelo Docker (a) e Máquinas Virtuais (b).



Fonte: Adaptado de Docker (2021c).

A plataforma Docker disponibiliza um conjunto de ferramentas que permite os desenvolvedores criarem e compartilharem novos `container` com a comunidade. Os desenvolvedores constroem um documento de texto chamado de `Dockerfile`, no qual descrevem as configurações da infraestrutura do ambiente, como quais aplicações serão usados e as portas disponíveis para serviços. A partir desse documento, é gerada uma `imagem` que abstraí todo ambiente do usuário, encapsulando seus dados e aplicações em arquivos, e que pode ser usado para criação dos contêineres. As `imagens` podem ser compartilhadas com a comunidade permitindo outras pessoas terem acesso ao ambiente construído pronto para uso. Todas as `imagens` compartilhadas com a comunidade são disponibilizadas de maneira livre e aberta no serviço `DockerHub` (DOCKER, 2021b).

A possibilidade de construir ambientes empacotados prontos para uso e compartilhá-los com outras pessoas através de um serviço web pode ser usado pelos pesquisadores para facilitar a reprodutibilidade científica. Qualquer tipo de diferença nos resultados dos experimentos causado pelo ambiente, será solucionado ao compartilhar tanto os códigos e dados que fizeram parte da execução dos experimentos quanto o próprio ambiente. Além disso, priva do pesquisador que deseja reproduzir um trabalho científico a responsabilidade de buscar as dependências e suas versões exatas para que os experimentos funcionem próximo ao original. Neste caso, dependendo de como foi preparado, os experimentos serão executados como se fossem a primeira vez (BOETTIGER, 2015; KULKARNI et al., 2018).

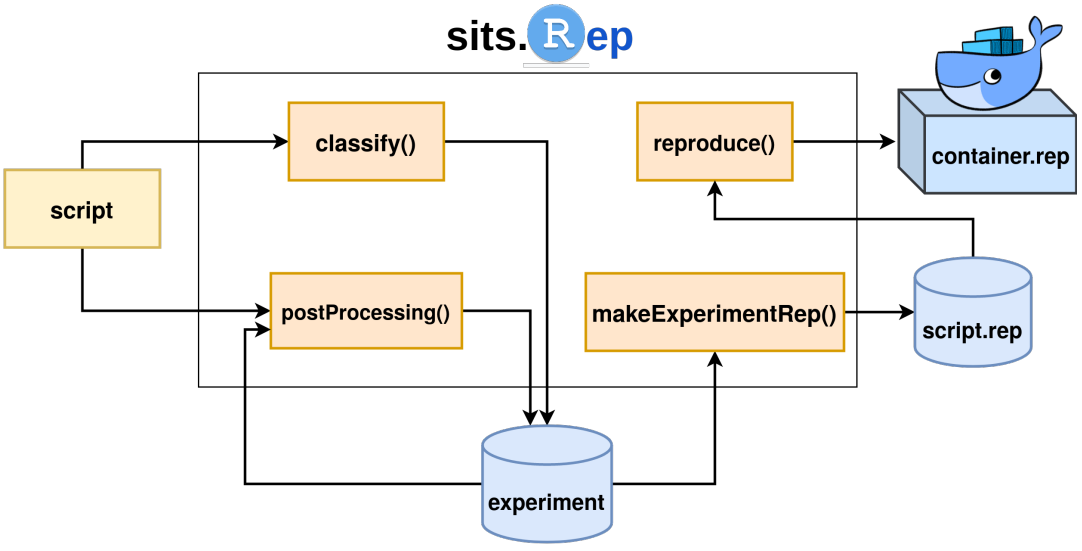
3 SITS.REP

Este capítulo apresenta o *Reproducible Satellite Image Time Series* (`sits.rep`), um pacote desenvolvido na linguagem R com o propósito de automatizar a organização e reprodução dos experimentos desenvolvidos usando o pacote `sits`. Durante a organização dos experimentos, são armazenados dados, códigos e demais artefatos necessários para criação de experimentos que envolvem classificação e pós-processamento de séries temporais extraídas de imagens de satélite. A partir do armazenamento sistematizado desses dados, o pacote `sits.rep` permite reproduzir esses experimentos, criando um ambiente computacional capaz de obter os mesmos resultados dos experimentos executados originalmente.

3.1 Visão geral do pacote

Nesta seção é demonstrada uma visão geral do pacote `sits.rep`, destacando suas funcionalidades e características. A Figura 3.1 mostra o fluxo de execução do pacote com suas principais funções e dados de entrada.

Figura 3.1 - Visão geral do pacote `sits.rep`.



Fonte: Produção do Autor.

Para o melhor entendimento, são definidos os seguintes conceitos:

script: Trechos de código escrito na linguagem R que possui um conjunto de instruções para gerar classificações a partir de imagens de satélite ou pós-

processar classificações. O `sits.rep` assume que cada etapa que gera novos resultados deve ser encapsulada em um *script* individual.

experimento: Conjunto de arquivos que armazena dados, metadados e *script* relativos a uma etapa do processo de classificação ou pós-processamento.

script.rep: Conjunto de arquivos com as informações necessárias para criação do ambiente reprodutível, incluindo o *script* original, *script* auxiliares, metadados e arquivos de especificação do ambiente Docker.

container.rep: Ambiente reprodutível, contém todos os softwares e *scripts* necessários para reproduzir uma classificação de uso e cobertura da terra.

O `sits.rep` pode receber dois tipos diferentes de *scripts*, que obrigatoriamente devem ser escritos sob um protocolo específico para serem interpretados corretamente. O primeiro script é de classificação e deve ser executado primariamente para inicializar o processo de criação dos experimentos, sendo processado dentro do ambiente do pacote. Neste ambiente, serão capturados em tempo de execução metadados específicos relacionados ao contexto e resultados desse processamento. Esses dados serão armazenados e considerados como um experimento para o `sits.rep`. O experimento criado poderá posteriormente ser reproduzido e seus resultados validados por outros pesquisadores que desejem analisar as etapas executadas e os resultados gerados.

Após a criação do experimento de classificação, é possível executar o segundo tipo de *script*, específico para pós-processamento, com o propósito de processar os seus resultados e criar um novo conjunto de dados. Todo processo de coleta e armazenamento dos metadados irá se repetir e será criado um novo experimento, porém com a existência de uma dependência entre o experimento anterior, usado como entrada, com o novo. Essa dependência irá se refletir na reprodução dos experimentos, no qual o `sits.rep` precisará reproduzir primeiro o experimento pai e, a partir do seu resultado, reproduzir o experimento alvo.

Uma vantagem do uso do pacote `sits` é que o grupo de pesquisa que o desenvolve também disponibiliza os dados de amostras e classificação usadas em suas pesquisas científicas de maneira aberta e livre para comunidade. Esses dados de amostras e classificação estão armazenados respectivamente nos pacotes `inSitu`[1] e `EOCubes`[2] para linguagem R e disponíveis na plataforma GitHub. Essa plataforma é um repositório de dados com o objetivo de disponibilizar códigos de programação de maneira

[1] https://github.com/e-sensing/inSitu
[2] https://github.com/e-sensing/EOCubes

aberta para todos e usa o sistema de versionamento Git para gerenciar e manter diferentes versões dos projetos. Essa característica permite capturar versões específicas dos projetos disponibilizados na plataforma. O `sits.rep` usa essa característica em favor da reprodutibilidade e economia de transferência dos dados, já que, não é preciso manter os dados de amostras e classificação em conjunto com os experimentos criados. Esses dados serão obtidos no momento em que forem necessários, quando o pesquisador reproduzir os experimentos. Nas próximas seções serão apresentados detalhes destes conceitos, bem como cada uma das funcionalidades do `sits.rep`.

3.2 Classificação

Para a criação dos mapas de uso e cobertura da terra, inicialmente os pesquisadores desenvolvem e executam *scripts* em R usando o pacote `sits` para classificar um conjunto de séries temporais de imagens de satélite. É por este motivo que o processamento de informações usando o `sits.rep` é iniciado com um *script* de classificação. Através da função `classify()`, o `sits.rep` executa um *script* para gerar uma classificação da mesma forma como é realizada pelo `sits`. Em tempo de execução, o `sits.rep` sobrescreve as funções presentes no sits para coletar os metadados referentes ao contexto do experimento, bem como o resultado final do processamento. Todos estes metadados são armazenados no experimento produzido ao final deste processo, que será chamado de experimento de classificação.

Durante a execução do experimento de classificação, são capturados metadados de contexto que contém informações sobre os pacotes importados, juntamente com as suas versões, dependências e seus respectivos repositórios principais. É importante manter esses dados, já que reproduzir experimentos com versões dos pacotes diferentes daqueles originalmente usados pode ocasionar no uso de funções com algoritmos e assinaturas modificadas, gerando resultados distintos ou até erros de execução. Também é armazenada a semente aleatória, o nome do experimento dado pelo pesquisador, um arquivo cópia do *script* , dados de cobertura referente ao serviço que irá requisitar as imagens a serem classificadas, os tipos de resultados que o experimento gerou e se um arquivo *hash* foi criado. A Figura 3.2 mostra a estrutura dos metadados criados ao executar um experimento de classificação. Esses metadados são armazenados em um arquivo no formato `JSON`.

Figura 3.2 - Metadado criado pelo `sits.rep` após a execução do `Script de Classificação`.

```
{
  "tree": "deep_learning",
  "parent": {},
  "process": "classification",
  "seed": 29576,
  "script": "classification.R",
  "import": [
    {
      "library": "sits",
      "version": "1.12.6",
      "dependencies": [
        {
          "library": "magrittr",
          "version": "1.5"
        }, ...
      ],
      "git_repository": "e-sensing",
      "git_commit": "d92bdc2f9b0f6158a8ae7ae8b7e4544fcca8390a"
    }, ...
  ],
  "train": {
    "directory": "train",
    "file": "model.rds"
  },
  "coverage": {
    "directory": "coverage",
    "service": "EOCUBES",
    "name": "MOD13Q1/006",
    "timeline": "48 month",
    "geom": "geom/geom.shp"
  },
  "result": {
    "file": true,
    "rds": true
  },
  "hash": true
}
```

Fonte: Produção do Autor.

Sempre após finalizada a execução de um experimento de classificação, o `sits.rep` verifica se o mesmo produziu como resultado um conjunto de imagens de satélite. Caso seja positivo, cada imagem é processada em um algoritmo de criptografia chamado **sha1sum**, do sistema Unix, e é gerada uma sequência única de caracteres que será armazenada em um arquivo de metadados.

Essa sequência é gerada a partir de atributos internos da imagem. Portanto, se duas imagens criadas em momentos distintos possuírem os mesmos atributos internos, os caracteres obtidos pelo algoritmo de criptografia serão os mesmos. Entretanto, se houver qualquer alteração na imagem produzida, seja tamanho ou variação no valor de qualquer `pixel`, o conjunto de caracteres serão diferentes e consequentemente as classificações serão facilmente identificadas como diferentes, sem precisar comparar as imagens. A Figura 3.3 mostra um exemplo de como os caracteres são armazenados. O primeiro conjunto de caracteres representa o valor **hash** das imagens. O segundo descreve o arquivo que a imagem está armazenada.

Figura 3.3 - Arquivo **hash** criado pelo `sits.rep` após executar um experimento, contendo um identificador único para cada imagem gerada.

```
9cc2599931fe6e4677bcf4efcf19d967d592f1fa ./MT_h12v10_2000_9_2001_8.tif
07b59e8fdd2e39c73a3a6420a05b206bf7193e97 ./MT_h12v10_2001_9_2002_8.tif
6ca425f6d12ec560f9bf4c2c6f7329f771266643 ./MT_h12v10_2002_9_2003_8.tif
3bcf1d436eb6cd72b87872f8163717fbf0641d13 ./MT_h12v10_2003_9_2004_8.tif
448f2aefd7f5bed9aa3f292bcded279ce6b57348 ./MT_h12v10_2004_9_2005_8.tif
be784533503a4a196e8db3a3d65f98e428e5a7d7 ./MT_h12v10_2005_9_2006_8.tif
5f6868dd19665e844c99b122794a0e5427c89a23 ./MT_h12v10_2006_9_2007_8.tif
442414c25ea5b0d377277d7a5522f9a46d3041fa ./MT_h12v10_2007_9_2008_8.tif
f4222513411d84921ecd36cb27f334c22f06a662 ./MT_h12v10_2008_9_2009_8.tif
727e01d483901c8d37722f93fbce56e8eb4399cf ./MT_h12v10_2009_9_2010_8.tif
ebbcdca86921dfa63b8ce6c8126b97776193a444 ./MT_h12v10_2010_9_2011_8.tif
83cf11bee041f6f78d544865c778f5d346244d3c ./MT_h12v10_2011_9_2012_8.tif
```

Fonte: Produção do Autor.

Para o `sits.rep`, todo experimento é imutável. Para alterar um experimento, é necessário alterar o *script* usado para criar o mesmo, disponível no próprio experimento, e então executá-lo novamente, produzindo um novo experimento. Isso garante que os resultados sejam sempre os mesmos, estando prontos para serem reproduzidos a qualquer momento.

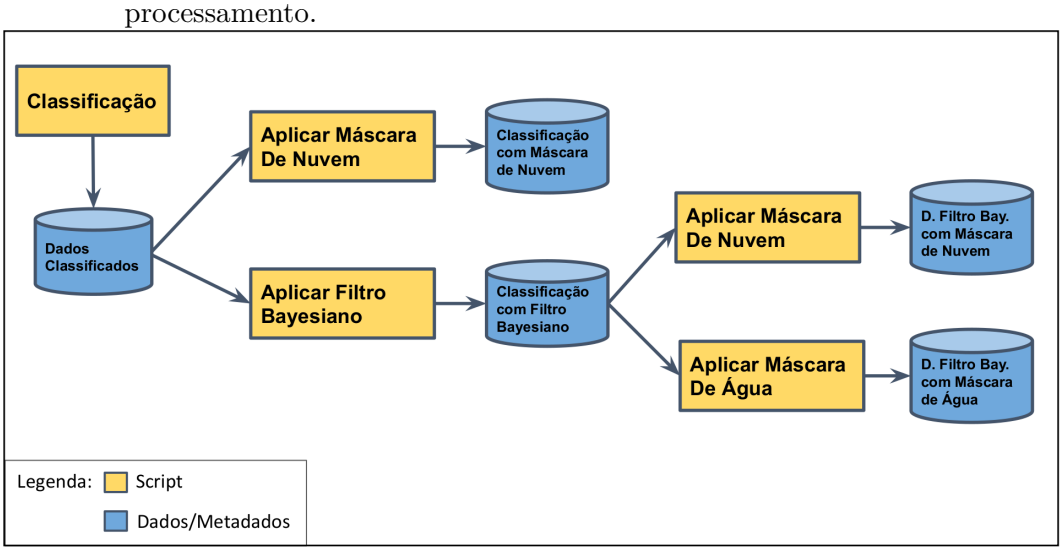
3.3 Pós-processamento

Após gerar experimentos de classificação, podem ser realizadas ações de pós-processamento sobre os resultados. Exemplos de etapas de pós-processamento incluem aplicar filtros, máscaras ou regras de transição. Estas ações têm o propósito de garantir certas propriedades aos resultados e também melhorar a acurácia dos mesmos.

No `sits.rep`, é possível usar um experimento de classificação como entrada para um *script* de pós-processamento. A função `postProcessing()` executa um *script* de pós-processamento de forma similar a um experimento de classificação. Durante a execução desta função, o `sits.rep` coleta e armazena os metadados necessários para garantir a reprodutibilidade do mesmo. Toda etapa de pós-processamento recebe como entrada (1) um *script* de pós-processamento e (2) o resultado de uma classificação ou o resultado de um pós-processamento. Esse *script* é executado de maneira similar aos *scripts* de classificação, de tal modo que os metadados também são coletados e armazenados em tempo de execução. Como o *script* de pós-processamento recebe os resultados de um experimento como entrada, existe uma relação de dependência entre estes experimentos. Esta informação é importante no processo de reprodução, uma vez que este experimento usa como entrada os resultados finais da execução prévia de um ou mais experimentos.

Para manter as relações entre os experimentos, o `sits.rep` organiza os mesmos em uma estrutura de dados do tipo árvore. O nó raiz se refere a um experimento de classificação e todos os demais representam experimentos de pós-processamento. A Figura 3.4 exemplifica esse conceito com um *script* de classificação sendo executado e inicializando uma árvore de experimentos, produzindo um experimento chamado de *Dados Classificados*. A partir dos *Dados Classificados*, são executados dois *scripts* de pós-processamento independentes, cujos resultados não estarão relacionados, chamados de *Classificação com Máscara de Nuvem* e *Classificação com Filtro Bayesiano*. Sobre este último nó, são executados dois outros *scripts*, produzindo resultados também independentes.

Figura 3.4 - Relação de dependência entre os processos de classificação e pós-processamento.



Fonte: Produção do Autor.

O objetivo de usar uma estrutura de dados em árvore é permitir o isolamento dos dados de cada experimento de maneira simples e organizada. Intuitivamente, é fácil de entender qual experimento depende de qual e, de maneira macro, quais foram os passos para chegar a um resultado de um experimento de pós-processamento. Além disso, ela permite rapidamente adicionar um nó filho dado um nó pai, ou até buscar todos os nós ancestrais de um experimento. Sobre a busca, como árvores não permitem que um mesmo nó filho possua dois pais, não é preciso percorrer todos os experimentos de uma árvore para descobrir as dependências de um determinado experimento filho.

Os metadados criados durante a execução de um experimento de pós-processamento são similares àqueles criados ao executar experimentos de classificação, como pode ser visto na Figura 3.5. O atributo `args` é usado para indicar ao pacote `sits.rep` quais resultados do experimento pai estarão disponíveis para o experimento de pós-processamento usar. O atributo `parent` indica o nome do experimento pai. Note que um experimento de classificação não possui um `parent` por ser um nó raiz da árvore.

Figura 3.5 - Metadado criado pelo `sits.rep` após a execução do `Script de Pós-Processamento`.

```
{
  "tree": "verificar_eocubes_funcionando",
  "parent": "classification",
  "process": "smooth",
  "import": [...],
  "seed": 29552,
  "args": {
    "input_file": true,
    "input_rds": true
  },
  "result": {
    "file": true,
    "rds": true
  },
  "hash": true,
  "script": "smooth.R"
}
```

Fonte: Produção do Autor.

Para organizar a dependência entre experimentos, ou seja, quais herdam os resultados de outros experimentos, é usado um arquivo de texto em que cada linha irá conter dois conjuntos de caracteres separados por um espaço que representa o nome de cada experimento. A exceção é dada sempre ao primeiro item desse metadado, no qual o primeiro conjunto de caracteres irá informar o nome da árvore, que irá armazenar todos os experimentos descritos neste arquivo, e o segundo, o nome do experimento de classificação definida por padrão como `classification`. Esse arquivo será escrito sempre após a execução do experimento, mantendo a ordem do primeiro conjunto de caracteres informando o experimento pai e segundo o experimento filho dependente. A Figura 3.6 mostra um metadado de uma árvore chamada `tree_classification_deep_learning` que possui dois experimentos: `classification` e `smooth`. Como pode ser visto na linha dois, o experimento `smooth` depende dos resultados do experimento pai `classification`, pois está inserida no segundo conjunto de caracteres.

Figura 3.6 - Arquivo texto de metadado que armazena dados sobre quais árvores os experimentos pertencem e suas dependências.

```
"tree_classification_deep_learning" "classification"
"classification" "smooth"
```

Fonte: Produção do Autor.

O encadeamento dos experimentos é essencial para garantir a reprodutibilidade. No `sits.rep`, para reproduzir os resultados de um determinado experimento, são necessárias as informações de todo o caminho a partir do experimento de classificação até o experimento escolhido. A reprodução de um experimento corresponde à execução em sequência de todos os experimentos na mesma ordem definida pelo caminho.

3.4 Script reprodutível (`script.rep`)

Como apresentado anteriormente, o processo de criação de mapas de uso e cobertura da terra envolve a realização de diferentes experimentos que serão armazenados em caminhos diferentes na árvore de experimentos. Um destes caminhos terá o resultado mais satisfatório para o pesquisador e, portanto, será usado como resultado final do estudo. O experimento escolhido será usado para criar um *script* reprodutível, ou `script.rep`.

Para criar um *script* reprodutível, o pacote `sits.rep` possui uma função denominada `makeExperimentRep()`. Esta função recebe como argumento o nome de um experimento presente na árvore e unifica todos os metadados dos experimentos dependentes ao longo do caminho até o experimento selecionado. Esse novo metadado contém a lista de dependências de todos pacotes usados nos *scripts* dos experimentos, a ordem de execução dos experimentos e os nomes dos diretórios para controle de armazenamento dos resultados.

A Figura 3.7 exemplifica um arquivo de metadado unificado. A chave `process` contem um vetor usado para controlar a ordem de execução dos experimentos. Neste exemplo, o primeiro experimento a ser executado é o `classification` e em seguida, o experimento `smooth`. Cada experimento tem um nome, um diretório onde os resultados serão armazenados, metadados que irão informar dados de ambiente como a semente da execução, e o *script* que será executado.

Figura 3.7 - Metadado unificado para orquestrar a ordem de execução dos experimentos, agrupar as dependências e indicar os diretórios de controle.

```
{
  "import": [...],
  "process": [
    {
      "name": "classification",
      "dir": "./classification",
      "metadata": "./classification/metadata.JSON",
      "script": "./classification/classification.R"
    },
    {
      "name": "smooth",
      "dir": "./smooth",
      "metadata": "./smooth/metadata.JSON",
      "script": "./smooth/smooth.R"
    }
  ],
  "dir_principal": "~/sits.rep",
  "dir_name": "smooth_rep"
}
```

Fonte: Produção do Autor.

O último arquivo gerado é um *Dockerfile*, que contém a especificação para construir o ambiente capaz de reproduzir o experimento selecionado. Com isto, o `script.rep` poderá ser reproduzido e seus resultados validados por outros pesquisadores que desejam analisar as etapas executadas e os resultados gerados. A Figura 3.8 mostra um *Dockerfile* gerado pelo `sits.rep`.

Figura 3.8 - Dockerfile contendo todas as dependências necessárias para configurar um ambiente capaz de reproduzir os experimentos executados com o `sits.rep`.

```
FROM rafaelmariano/sits:1.12.6
RUN mkdir -p /usr/bin
RUN mkdir -p sits-rep
ADD ./ sits-rep
RUN rm sits-rep/Dockerfile
RUN cd sits-rep && Rscript install_dependencies.R
RUN cd sits-rep && Rscript script-rep.R
```

Fonte: Produção do Autor.

Os arquivos gerados no final da execução são relacionados ao metadado para construção do ambiente reprodutível e orquestração da ordem de execução dos experimentos. Os dados de séries temporais de imagens de satélite usados como entradas para treinamento e classificação dos experimentos serão obtidos somente quando os experimentos forem reproduzidos. Dessa forma, o espaço de armazenamento usado pelo *script* reprodutível é irrelevante. Essa característica permite facilmente compartilhar o *script* reprodutível com outros pesquisadores, seja por meio de e-mail ou através de plataformas web de repositórios de dados abertos. O baixo volume de armazenamento permite um pesquisador compartilhar diversos experimentos reprodutíveis em plataformas abertas ao publicar sua pesquisa, já que diversas delas permitem um limite gratuito e considerável de espaço para disponibilização de arquivos via nuvem.

3.5 Ambiente reprodutível (`container.rep`)

A partir de um `script.rep`, é possível criar o ambiente computacional necessário para reproduzir os experimentos. A função `buildContainer()` é usada para esta finalidade. A partir do arquivo *Dockerfile*, contido no repositório `script.rep`, é construída uma imagem *Docker* contendo a versão correta do `sits` e suas bibliotecas de dependência, além de eventuais dependências adicionais extraídas dos *scripts*. Os *scripts* de classificação e pós-processamento, juntamente com um *script* especial para orquestração da execução dos experimentos na ordem determinada nos metadados, são incluídos nesta imagem. Ao final da geração da imagem, também é criado um contêiner preparado para execução.

Ao acessar o ambiente do contêiner, poderá ser iniciado o processo de reprodução dos experimentos. Dessa forma, os *scripts* auxiliares do `sits.rep` irão executar cada experimento informado nos metadados. No final da execução, os resultados serão armazenados e, para cada um deles, será criado um identificador *hash*. Esse valor será armazenado em um arquivo específico para este experimento e será iniciado o processo de verificação. Através dos metadados, será lido o arquivo com os identificadores *hash* gerados originalmente, durante a execução das funções `classify()` e `postProcessing()`. Eles serão comparados com os identificadores *hash* dos resultados obtidos na execução dos experimentos dentro do contêiner. Caso os valores sejam iguais, então o experimento reproduzido é considerado válido. Caso os valores sejam diferentes, tem-se a indicação de que o resultado obtido não foi capaz de reproduzir o experimento original. Nessa situação, possivelmente, o `sits.rep` não coletou alguma informação essencial para garantia da reprodutibilidade. Independente do

resultado, o processo de reprodução irá continuar até que o último experimento seja executado e verificado.

3.6 Demonstração do pacote `sits.rep`

Esta seção apresenta uma demonstração sobre a capacidade do `sits.rep` de ajudar os pesquisadores a gerar resultados reprodutíveis. O código-fonte da criação e organização dos experimentos é mostrado na Figura 3.9. O uso do `sits.rep` inicia com a definição de uma árvore, que neste caso é definido como `deep` como é mostrado na linha 2, onde será armazenado todos os experimentos criados e seus resultados. Em seguida, o *script* de classificação é executado, como mostrado na linha 3. Este *script* `classification.R` utiliza os dados de amostras do pacote `inSitu` para treinar um modelo de aprendizagem de máquina com o algoritmo de *Deep Learning*[3]. Após o treinamento, são solicitadas ao pacote `EOCubes` as imagens de satélite do produto `MOD13Q1` de uma região de estudo localizada no Estado do Mato Grosso. Estas imagens são classificadas usando o modelo treinado. O `sits.rep` utiliza uma estratégia de introspecção no código do pacote `sits`, o que possibilita coletar automaticamente diversos metadados do *script* de classificação, armazenando os resultados em um diretório denominado `classification`, sob a árvore de dependência.

Figura 3.9 - Código do pacote `sits.rep` para a criação dos experimentos de classificação e pós processamento.

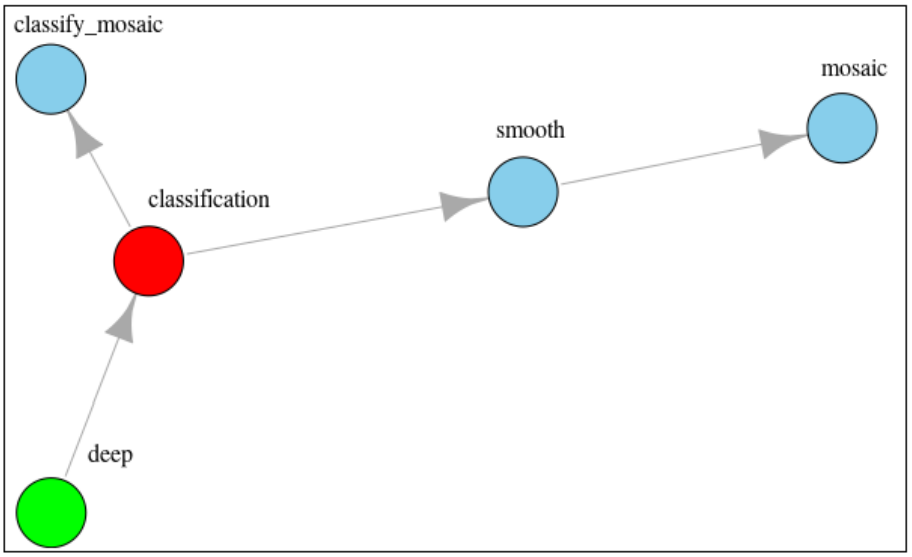
```
1  library(sits.rep)
2  sits.rep::useTree("deep")
3  sits.rep::classify(script = "classification.R")
4
5  sits.rep::pos_processing(parent = "classification",
6      process = "smooth", func = sits.rep::smooth)
7
8  sits.rep::pos_processing(parent = "smooth",
9      process = "mosaic", func = sits.rep::merge)
10
11 sits.rep::pos_processing(parent = "classification",
12     process = "classify_mosaic", func = sits.rep::merge)
```

Fonte: Produção do Autor.

[3]O código fonte do *script* de classificação pode ser encontrado no seguinte repositório: https://github.com/RafaMariano/sits_rep_estudo_de_caso

Após executada a classificação, os resultados serão usados como entrada para os experimentos de pós-processamento. O código da linha 5 executa uma função de segunda ordem para aplicar um filtro Bayesiano sobre todas as imagens classificadas. Os dois próximos comandos são executados para criar mosaicos sobre as imagens do resultado do experimento inicial, bem como do experimento `'smooth'`. O pacote `EOCubes` divide a grade original do MODIS em grades de tamanho 10x10, sendo interessante produzir apenas uma imagem para posteriormente aplicar máscaras. Ao criar um novo experimento a partir de um nó que já possua um filho (por exemplo, na linha 11), o `sits.rep` automaticamente cria uma bifurcação na árvore. A Figura 3.10 mostra a organização lógica das dependências entre os experimentos criados anteriormente na árvore `deep`.

Figura 3.10 - Experimentos criados na árvore `deep` e suas dependências. A esfera verde indica a árvore. A esfera vermelha é o experimento de classificação. As esferas azuis são experimentos de pós-processamento.

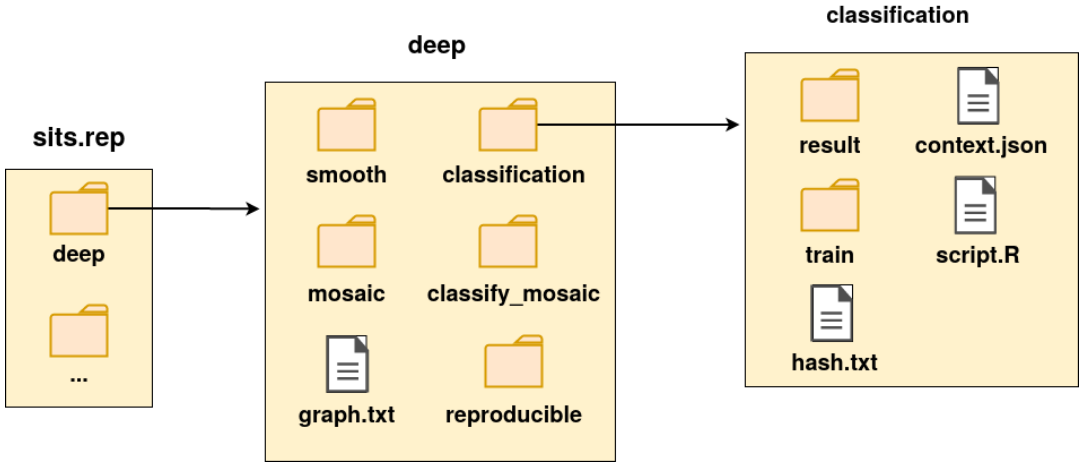


Fonte: Produção do Autor.

O `sits.rep` irá armazenar e organizar os experimentos em uma estrutura de diretório similar ao mostrado no exemplo da Figura 3.11. Será criado um diretório principal chamado de `sits.rep`, no qual irá conter outros diretórios que representam as árvores de dependência criada pelo pesquisador. Nos diretórios das árvores serão armazenados os experimentos de classificação e pós-processamento. A ordem de execução desses experimentos e suas relações de dependências serão armazenados no arquivo `graph.txt`.

Cada diretório de experimento irá conter arquivos de metadados, que incluem os resultados da execução, dados de contexto, `hash` dos resultados, *script*e outros dados específicos. No caso da classificação, o metadado específico armazenado é um modelo de aprendizado de máquina usado para criar as classificações das imagens de satélite. Caso o pesquisador tenha interesse em aproveitar algum modelo que obteve um resultado satisfatório, poderá fazer uso acessando o diretório do experimento de classificação. Também serão armazenados no diretório principal da árvore, especificamente no diretório `reproducible`, todos os *scripts* reprodutíveis criados pelos pesquisadores.

Figura 3.11 - Estrutura de diretório criada e organizada pelo `sits.rep` para armazenar os experimentos.



Fonte: Produção do Autor.

Para reproduzir os experimentos, o pesquisador deverá escolher um dos nós da árvore para gerar o `script` reprodutível. O código-fonte para construção do *script* reprodutível e reprodução dos experimentos é mostrado na Figura 3.12. Este `script` envolverá todos os nós a partir da raiz até o nó selecionado. Para isso, basta informar para a função `makeExperimentRep` o nome do experimento e do diretório onde serão armazenados os arquivos, como demonstrado na linha 3 da Figura 3.12. Esse *script* reprodutível conterá todos os passos aplicados até a reprodução dos mesmos resultados do nó selecionado para reprodução. Ele é armazenado em um diretório que poderá ser carregado em plataformas abertas para que qualquer usuário possa reproduzir os experimentos executados.

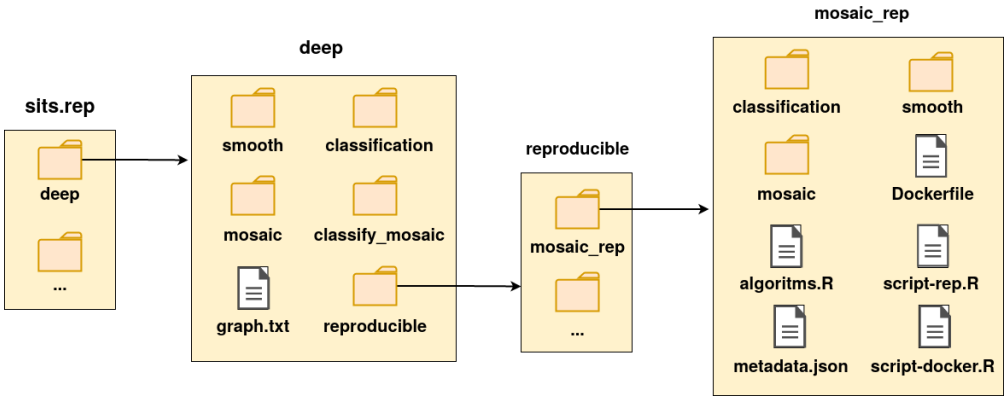
Figura 3.12 - Exemplo de código do pacote `sits.rep` para reprodução dos experimentos.

```
library(sits.rep)
sits.rep::useTree("deep")
sits.rep::makeExperimentRep("mosaic", "mosaic_rep")
sits.rep::reproduce("mosaic_rep")
```

Fonte: Produção do Autor.

Adicionalmente, o `sits.rep` possui uma função chamada `reproduce`, que recebe o diretório de um `script` reprodutível, e constrói uma imagem Docker contendo todas as dependências, nas mesmas versões usadas para gerar a classificação, juntamente com o `script` reprodutível. Com isto, todo o processo de reprodutibilidade é garantido, desde a coleta das informações enquanto o usuário executa `scripts` de classificação e pós-processamento, até a geração de um contêiner reprodutível, que poderá ser disponibilizado em um repositório na Web. A Figura 3.13 mostra a estrutura de diretório e os arquivos criados pelo `sits.rep` após executar a função `makeExperimentRep` e `reproduce`. Os arquivos dos *scripts* reprodutíveis são armazenados no diretório `reproducible`. Além dos metadados do experimento, o *script* reprodutível irá conter alguns algoritmos que irão orquestrar a execução dos experimento dentro do `container`. O *script* reprodutível `mosaic_rep` poderá ser compartilhado e reproduzido por outros pesquisadores usando a função `reproduce`.

Figura 3.13 - Organização dos arquivos de um experimento.



Fonte: Produção do Autor.

4 ESTUDO DE CASO

Neste capítulo é apresentado um estudo de caso para demonstrar a capacidade do pacote `sits.rep` de automatizar a criação e reprodução de experimentos criados a partir do pacote `sits`. O *script* executado neste estudo de caso é baseado no artigo de Simoes et al. (2020), cujo *script* disponível no repositório de dados Zenodo (SIMOES et al., 2019).

4.1 Mapas de uso e cobertura do Mato Grosso de 2001 a 2017

No artigo de Simoes et al. (2020), é apresentada uma metodologia para construir um conjunto de mapas de uso e cobertura da terra de todo Estado do Mato Grosso, entre os anos de 2001 a 2017. Esses dados são importantes para auxiliar nas análises sobre mudanças ocorridas na cobertura terrestre ao longo do tempo, identificando áreas de perda de vegetação natural e expansão agrícola. A partir destes dados, é possível diferenciar as possíveis causas do desmatamento naquela região, como, por exemplo, pela expansão de culturas ou pastagens.

Para identificar quais foram os usos e a cobertura terrestre no Estado, foram usadas técnicas de aprendizagem de máquina para classificar séries temporais de imagens de satélite. Uma base de dados de 1.892 amostras, entre os anos de 2000 a 2015, foi treinada usando um modelo SVM. Antes do treinamento, as amostras foram pré-validadas usando o algoritmo de `self-organizing maps` (SOM), com propósito de eliminar dados com rótulos imprecisos ou com baixa separabilidade, melhorando a precisão modelo e, consequentemente, o resultado da classificação. A estrutura de organização dos dados de amostras é semelhante à apresentada na Tabela 2.1, possuindo as seguintes classes: floresta, cerrado, pasto, soja (monocultura), algodão (monocultura), soja-algodão (safra dupla), soja-milho (safra dupla), soja-milheto (safra dupla) e soja-girassol (safra dupla). A base de dados dessas amostras está disponível abertamente no pacote `inSitu`.

A partir do modelo treinado, são classificados seis conjuntos de imagens de satélite do produto MOD13Q1 do sensor MODIS. Esse produto possui uma resolução espacial de 250 metros e frequência temporal de 16 dias. Foram usadas as bandas do infravermelho próximo (NIR) e infravermelho médio (MID), em conjunto com os índices de vegetação `Normalized Diference Vegetation Index` (NDVI) e `Enhanced Vegetation Index` (EVI). Os dados foram organizados em uma estrutura de cubo de dados, aproveitando a dimensão temporal para classificar os `pixeis` ao longo do tempo.

Após a classificação, é realizada uma série de pós-processamento para melhorar a qualidade da mesma. Para tal, é aplicado um filtro Bayesiano e máscaras de áreas urbanas, cana-de-açúcar, água, floresta e cerrado. As duas últimas máscaras são obtidas dos projetos PRODES Amazon e PRODES Cerrado, ambos produzidos pelo INPE. Os mapas finais foram validados usando o algoritmo de `5-fold cross-validation`, que estimou uma acurácia de 96%. Esses dados foram comparados com dados disponibilizados pelo IBGE e PRODES para os anos de 2001 a 2017. Em suas avaliações, os autores concluem que os mapas produzidos são confiáveis e consistentes com a realidade.

4.2 `Script` de classificação e pós-processamento

Como dito anteriormente, o *script* executado no estudo de caso de Simoes et al. (2020) está disponível de maneira aberta no repositório de dados Zenodo (SIMOES et al., 2019). Este *script* contém quatro processamentos principais separados em funções. O resultado de um determinado processamento é usado como entrada para o próximo. O resultado produzido a partir de cada processamento é um diretório contendo um conjunto mapas anuais de uso e cobertura da terra entre os anos de 2001 a 2017.

No primeiro processamento, são realizados todos os passos para treinar e classificar as séries temporais, além de ser aplicado um pós-processamento para a junção das imagens em um único mosaico para cada ano. As amostras são obtidas diretamente do pacote `inSitu` e foram avaliadas previamente. As bandas e os índices de vegetação são selecionados e entregues para uma função do pacote `sits` que irá realizar o treinamento. É definido o modelo SVM com alguns parâmetros usando valores padrões da própria função, porém modificando o valor do custo para 1 e a fórmula para uma função linear. Em seguida, é definida a cobertura que irá ser classificada e de qual serviço serão obtidas as imagens. O serviço usado é o `EOCubes`, compatível com o pacote `sits` que fornece imagens de satélites armazenados em repositórios gerenciados pelos próprios pesquisadores. Para cobertura, também é preciso fornecer uma grade que corresponde toda área de interesse que será classificada, no caso, o Estado do Mato Grosso. A partir do modelo treinado e da cobertura definida, é executada a classificação das imagens com os resultados sendo armazenados em um diretório local.

Na Figura 4.1, é mostrado o trecho de código adaptado do *script* com as fases de obtenção das amostras, definição do modelo de aprendizagem de máquina e classificação das imagens. Essas três fases usam primariamente o pacote `sits` em suas

execuções. Para mostrar especificamente o processo de classificação, foi necessário omitir partes do código. O parâmetro `outputDir` corresponde ao diretório `MT`. Já os parâmetros `mem_size` e `processors` possuem os valores 2 e 1, respectivamente.

Figura 4.1 - Código para classificação séries temporais de imagens de satélite usando o pacote `sits`.

```
data(br_mt_1_8K_9classes_6bands)

output_classify <- paste(outputDir, "1.Classification", sep = "/")

bands <- c("evi", "ndvi", "nir", "mir")

samples.tb <- br_mt_1_8K_9classes_6bands %>%
    sits_select_bands_(bands = bands)

model.svm <- samples.tb %>%
    sits_train(ml_method = sits_svm(cost=1,
            formula=sits_formula_linear()))

cov.tb <- sits_coverage(
        service = "EOCUBES",
        name = "MOD13Q1/006",
        bands = bands,
        geom = sf::read_sf(system.file("extdata/MT/shape/MT.shp",
                                package = "inSitu")))

rasters.tb <- sits_classify_cubes(
        file = paste(output_classify, "MT", sep = "/"),
        coverage = cov.tb,
        ml_model = model.svm,
        memsize = mem_size,
        multicores = processors)
```

Fonte: Adaptado de Simoes et al. (2019).

Após a classificação das imagens, o processamento é seguido com aplicação de um filtro Bayesiano para suavizar os ruídos e melhorar os resultados. É aplicada uma janela 3x3 sobre os `pixeis` de cada imagem com um valor de ruído igual a 10, e o resultado é armazenado em um segundo diretório local. Por fim, os conjuntos de imagens para cada ano são agrupados e transformados em mosaicos que representam toda área de cobertura do Estado do Mato Grosso. Esses mosaicos são armazenados em um terceiro diretório e é o resultado final do primeiro processamento. A Figura 4.2 mostra o trecho de código adaptado do *script* que apresenta as funções usadas para aplicação do filtro Bayesiano sobre as imagens classificadas e a criação do mosaico.

Figura 4.2 - Código de pós-processamento para aplicação do filtro Bayesiano nas imagens classificadas e criação do mosaico.

```
output_smooth <- paste(outputDir, "2.Classification_Smooth", sep = "/")
output_mosaic <- paste(outputDir, "3.Classification_Mosaic", sep = "/")

sits_bayes_postprocess(
        rasters.tb,
        window = matrix(1, nrow = 3, ncol = 3, byrow = TRUE),
        noise = 10,
        file = paste(outputDir_smooth, "smooth", sep = "/"))

files_input <- list.files(output_smooth, pattern = ".*\\.tif",
    full.names = TRUE)

files_years <- gsub("^.*smooth_MT_[^_]{6}_[0-9]+_[0-9]+_[0-9]+_[0-9]+_
    ([0-9]+)_.*\\.tif", "\\1", files_input)

for (year in unique(files_years)) {
    year_list <- files_input[files_years == year]
    res <- lapply(year_list, raster::raster)
    res$filename <- paste(output_mosaic,
        sprintf("MT_%s.tif", year), sep = "/")
    do.call(raster::merge, res)
}
```

Fonte: Adaptado de Simoes et al. (2019).

No segundo processamento são adicionadas três classes não incluídas na classificação original: área urbana, cana-de-açúcar e água. Para tal, o processamento é realizado alterando diretamente os valores do mosaico de resultado com os dados fornecidos pelas máscaras dessas classes em seus respectivos `pixeis` para cada ano. A ordem de aplicação das máscaras sobre os resultados é importante, uma vez que, caso um `pixel` esteja marcado em duas máscaras diferentes, a última máscara a ser aplicada indicará o valor final do mesmo. Finalizada a inclusão dessas três classes, são aplicadas máscaras de floresta (PRODES Amazon) e cerrado (PRODES Cerrado), mas somente para o resultado do mosaico do ano de 2001. Essa decisão foi tomada para que os dados possam ser mais consistentes com os anos anteriores a 2001, já que o método de classificação não possui essas informações. Na Tabela 4.1 são mostradas as regras usadas para aplicação das duas máscaras PRODES.

Tabela 4.1 - Regras aplicadas nos mosaicos de resultados (M_i) usando as máscaras PRODES Amazônia (A_i) e Cerrado (C_i).

Condição	**Classe de resultado para M_i**
A_i == (Floresta)	Floresta
A_i == (Não Floresta) e M_i == (Floresta)	Cerrado
A_i == (Desmatamento) e M_i == (Floresta)	Vegetação Secundária
C_i == (Não Antropisados) e M_i == (Floresta)	Cerrado
C_i == (Antropisados) e M_i == (Floresta)	Vegetação Secundária

Fonte: Adaptado de Simoes et al. (2020).

Ainda no segundo processamento, são aplicadas regras de cálculo de mudanças de uso da terra (cálculo LUC) sobre todos os mapas de resultado para eliminar inconsistências temporais existentes nos dados. Um exemplo apresentado pelo Simoes et al. (2020), é o caso de uma área ser classificada como floresta em um ano específico, e nos anos seguintes como cerrado. Essa é uma transição impossível de acontecer. O cálculo é realizado ao aplicar as regras de transição sobre um conjunto de anos para definir as classes dos anos inconsistentes. As regras aplicadas são mostradas na Tabela 4.2.

Tabela 4.2 - Regras de transição. As classes são identificadas como: floresta (F), cerrado (C), área de pasto (P), cultura de soja (S) e vegetação secundária (SV). O símbolo '*' indica as classes que serão atualizadas. A transição das classes é realizada da esquerda do símbolo '→' para direita.

Situação	**Resultado**
C → F*	C → C*
C → C → P* → C	C → C → C* → C
C → C → S* → C	C → C → C* → C
P → P → C* → C* → P	P → P → P* → P* → P
F → C* → F → F	F → F* → F → F
F → F → C* → F	F → F → F* → F
F → C* → F	F → F* → F
F → C*	F → F*
F → F → P → F*	F → F → P → SV*
P → P → F* → P	P → P → SV* → P

Fonte: Adaptado de Simoes et al. (2020).

O terceiro processamento aplica o cálculo LUC com o foco em identificar áreas de vegetação secundária, aquelas que houveram intervenção humana. Podem ocorrer casos de uma área de floresta natural ser desmatada em um ano e após o abandono voltar a crescer. É interessante diferenciar as florestas primárias das vegetações secundárias, porém a classificação original rotula somente como floresta as áreas abandonadas após o desmatamento.

O quarto e último processamento agrupa todos os resultados em uma imagem única para cada ano. Devido a dimensão dos dados, o mosaico foi dividido em grupos menores para serem processados em lote. Dessa forma, a memória não é sobrecarregada, além de permitir o processamento paralelo. Após os resultados serem agrupados, o processamento é finalizado.

4.3 Ambiente de Desenvolvimento

Os experimentos desenvolvidos no *script* reprodutível foram executados em um ambiente computacional com o Sistema Operacional Ubuntu 18.04 usando o software R na versão 3.6.2. Algumas dependências de terceiros são necessárias para permitir o uso do `sits`. São elas:

- Linguagem Python na versão 3.6.8[1].
- Pacote Tensorflow na versão 1.9 para Python[2].
- Pacote h5py na versão 2.10.0 para Python[3].
- Pacote GEOS na versão 3.7.1 para S.O Ubuntu 18.04[4].
- Pacote Proj na versão 5.2 para S.O Ubuntu 18.04[5].
- Pacote HDF5 na versão 1.10.4 para S.O Ubuntu 18.04[6].
- Pacote GDAL na versão 2.4.2 para S.O Ubuntu 18.04[7].
- Pacote sf para Ubuntu[8].

[1] `https://www.python.org/ftp/python/3.6.8/Python-3.6.8.tgz`

[2] Use o seguinte comando para instalar: pip3.6 install –ignore-installed –upgrade tensorflow==1.9.0

[3] Use o seguinte comando para instalar: pip3.6 install –ignore-installed –upgrade h5py==2.10.0

[4] `download.osgeo.org/geos/geos-3.7.1.tar.bz2`

[5] `download.osgeo.org/proj/proj-5.2.0.tar.gz`

[6] `https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.10/hdf5-1.10.4/src/hdf5-1.10.4.tar.bz2`

[7] `download.osgeo.org/gdal/2.4.2/gdal-2.4.2.tar.gz`

[8] `https://github.com/r-spatial/sf`

Além disso, é preciso instalar outros pacotes de dependências na linguagem R e, por fim, o pacote `sits` como mostra a Figura 4.3. Ao instalar o pacote `sits`, é preciso especificar o `commit` da versão 1.12.6 que é reconhecida pelo `sits.rep`, como demonstra na linha 15 da Figura 4.3. É recomendável instalar as dependências na ordem informada.

Figura 4.3 - Dependência que devem ser instaladas para executar os experimentos do *script* disponibilizado pelo Simoes et al. (2019).

```
1  install.packages('lwgeom')
2  install.packages('tidyverse')
3  install.packages('rgeos')
4  install.packages('rversions')
5  install.packages('roxygen2')
6  install.packages('devtools')
7  install.packages('ensurer')
8
9  devtools::install_version('sf', '0.7-7', upgrade=FALSE, force=TRUE)
10
11 devtools::install_version('raster', '2.9-23', upgrade=FALSE, force=TRUE)
12 devtools::install_github("e-sensing/wtss")
13 devtools::install_github("pedro-andrade-inpe/sits.validate",
       upgrade=FALSE, force=TRUE)
14
15 devtools::install_github("e-sensing/inSitu", upgrade=FALSE, force=TRUE)
16
17 devtools::install_github("e-sensing/lucCalculus", upgrade=FALSE, force=
       TRUE)
18 devtools::install_github('e-sensing/sits', ref ='
       d92bdc2f9b0f6158a8ae7ae8b7e4544fcca8390a', upgrade = FALSE)
```

Fonte: Produção do Autor.

4.4 `Script` Reprodutível gerado pelo `sits.rep`

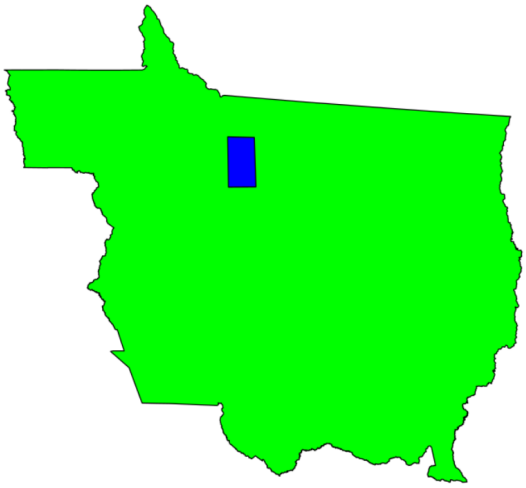
Nesta seção será apresentado o *script* reprodutível gerado pelo pacote `sits.rep` usando como base o *script* de classificação e pós-processamento disponibilizado por Simoes et al. (2019). Para construir esse *script*, foi necessário realizar adaptações no código original para funcionar no `sits.rep`. Além disso, o escopo da classificação foi simplificado para que o processamento pudesse ser mais rápido. Será realizado somente o primeiro processamento do *script* original e a quantidade de dados treinados e classificados foram reduzidas. Nesse estudo de caso, não será abordada a bifurcação dos experimentos em uma árvore de processos, já que o *script* de

Simoes et al. (2019) não aborda multi-dependência de experimentos dado um experimento pai, por se tratar de uma classificação final.

No *script* original, não existe a separação entre o processo de classificação e as etapas de pós-processamento. Todos os processos são tratados como uma sequência de comandos. Os resultados de cada processo interno são armazenados e usados como entrada para outros processos até que o resultado final seja gerado e finalizado o processamento principal. Para que funcione como esperado no `sits.rep`, cada um desses processos internos precisa ser separado em um experimento próprio.

O primeiro experimento executado é de classificação, mandatório para iniciar a árvore de experimentos no `sits.rep`. O processo de classificação foi adaptado para ser executado em um arquivo separado, uma exigência da função `sits.rep::classify()`. Outra adaptação foi realizada na área de cobertura. Ao invés de obter imagens de todo Estado do Mato Grosso, o escopo foi diminuído para abranger somente uma pequena região entre São José do Apuí e Novo Paraná (em azul na Figura 4.4). A Figura 4.5 apresenta o código usado para executar o *script* de classificação adaptado para o `sits.rep`. Após a execução, o resultado será armazenado em um diretório chamado `classification`, atrelado árvore `deep`. Além disso, para usar os dados produzidos por esta classificação como entrada para os próximos experimentos, será necessário apenas o nome deste experimento.

Figura 4.4 - A área de interesse (em azul) corresponde uma pequena região do Estado do Mato Grosso (em verde), entre São José do Apui e Novo Paraná, presente no tile h12v10 do produto MOD13Q1.



Fonte: Produção do Autor.

Figura 4.5 - Trecho de código que usa o pacote `sits.rep` para a criação de um experimento de classificação.

```
1 library(sits.rep)
2 sits.rep::useTree("svm")
3 sits.rep::classify(script = "./classification.R")
```

Fonte: Produção do Autor.

O segundo experimento executado é de pós-processamento, para aplicação de um filtro Bayesiano sobre as imagens classificadas. A adaptação desse processo é realizada de maneira diferente do primeiro experimento, sendo necessário criar uma função separada com três parâmetros de entrada, como é apresentado a partir da linha 3 na Figura 4.6. O primeiro parâmetro corresponde aos arquivos das imagens de resultados do experimento anterior. O segundo aos dados em memória obtidos ao final da classificação. Por fim, o último parâmetro corresponde à localização de um diretório no qual a função deverá escrever os arquivos de resultado. Ao final do processamento, os resultados estarão escritos no diretório `pos_bayesan` no experimento `smooth`, um nó conectado ao experimento pai `classification`.

Figura 4.6 - Trecho de código que usa o pacote `sits.rep` para a criação de um experimento de pós-processamento.

```
1  library(sits.rep)
2
3  smooth <- function (input_file, input_rds, output) {
4      library(raster); library(sp); library(sits)
5
6      result <- sits_bayes_postprocess(
7          raster_class = input_rds$rasters.tb,
8          window = matrix(1, nrow = 3, ncol = 3, byrow = TRUE),
9          noise = 10, file = paste0(output, "/pos_bayesan"))
10
11     return(result)
12 }
13
14 sits.rep::pos_processing(parent = "classification",
15     process = "smooth", func = smooth)
```

Fonte: Produção do Autor.

O terceiro e último experimento executado foi a criação dos mosaicos. Esse experimento é construído de maneira similar ao anterior, executando o processamento de criação dos mosaicos a partir de uma função separada e armazenando os resultados no diretório informado pelo `output`. Esse experimento não faz uso do pacote `sits` para realizar as operações, portanto não gera um resultado em memória. Os únicos resultados desse experimento são os arquivos dos mosaicos criados. A Figura 4.7 mostra a criação do experimento `mosaic` usando as imagens de resultado do experimento anterior chamado de `smooth`.

Figura 4.7 - Código para a criação dos experimentos de classificação e pós processamento.

```
library(sits.rep)

mosaic <- function (input_file, input_rds, output) {

files_input <- list.files(input_file,
    pattern = ".*\\.tif", full.names = TRUE)

files_years <- gsub("^.*_[^_]{6}_[0-9]+_[0-9]+_[0-9]+_[0-9]+_([0-9]+)_.
    *\\.tif", "\\1", files_input)

    for (year in unique(files_years)) {
        year_list <- files_input[files_years == year]

        if(length(year_list) < 2)
            next

        res <- lapply(year_list, raster::raster)
        res$filename <- paste0(output, "/mosaic_",
            sprintf("MT_%s.tif",year))
        do.call(raster::merge, res)
    }
}

sits.rep::pos_processing(parent = "smooth", process = "mosaic",
    func = mosaic)
```

Fonte: Produção do Autor.

Após a criação do experimento `mosaic`, será criado um *script* reprodutível a partir desse experimento. A Figura 4.8 mostra o comando `makeExperimentRep`, que irá buscar todos os experimentos ancestrais e irá organizar os metadados para que seja possível reproduzir todo o processo de classificação e pós-processamento. O novo

diretório criado e chamado de `mosaic_rep` poderá ser compartilhado em uma base de dados aberta para que outros pesquisadores e instituições de interesse possam acessar esse experimento e reproduzi-lo de maneira automatizada.

Figura 4.8 - Código para reprodução dos experimentos.

```
sits.rep::makeExperimentRep("mosaic", "mosaic_rep")
```

Fonte: Produção do Autor.

Com o *script* reprodutível, o pesquisador poderá baixar esse arquivo e reproduzir diretamente pelo `sits.rep`, ao executar o comando `reproduce`. A Figura 4.9 mostra o comando para reproduzir o experimento `mosaic` a partir do *script* reprodutível. Ao executar esse comando, será criado um container Docker com todo o ambiente necessário para executar o experimento.

Figura 4.9 - Demonstração de código do pacote `sits.rep` para reprodução dos experimentos.

```
library(sits.rep)
sits.rep::reproduce("mosaic_rep")
```

Fonte: Produção do Autor.

4.5 Discussão

O *script* apresentado por Simoes et al. (2020) possui vários dos elementos necessários para reproduzir os experimentos relatados na pesquisa e recriar os mapas de uso e cobertura da terra. As informações disponibilizadas são claras e dependendo do nível de conhecimento do leitor sobre a linguagem usada, os experimentos podem ser entendidos facilmente devido aos diversos comentários presentes no código que auxiliam na compreensão de cada passo executado.

Entretanto, existem algumas características identificadas que podem dificultar a reprodutibilidade completa do *script*. Por exemplo, o *script* não lida explicitamente

com o versionamento dos pacotes importados. Com isso, qualquer alteração realizada na lógica interna das funções em versões mais recentes dos pacotes, os processos poderão obter resultados diferentes que serão propagados para os próximos experimentos, causando um resultado diferente daquele apresentado na pesquisa. Além disso, caso o nome ou a assinatura de alguma das funções usadas seja alterado em versões mais novas, o *script* poderá não ser executado por algum erro de sintaxe na linguagem. Essas características delegam ao interessado a responsabilidade de buscar as versões corretas para cada dependência, seja através da consulta aos autores, ou tentando descobrir a versão disponível na época da publicação do artigo.

Outra característica identificada é que o *script* não explicita a semente aleatória. Esse valor é usado pelos modelos de aprendizagem de máquina para simular a inicialização dos parâmetros de maneira estocástica. Este é um exemplo de valor que dificilmente poderá ser recuperado após as classificações, se não ficar registrado no momento da execução do *script*. O impacto sobre os resultados produzidos poderá não ser grande, com algumas diferenças pontuais, porém ainda assim a reprodução não será exata.

Diferentemente de um ambiente onde o pesquisador possui total liberdade, sendo responsável pela reprodutibilidade, o `sits.rep` impõe algumas restrições. É necessário seguir o protocolo definido pelo pacote, separando os diferentes passos em experimentos. Adicionalmente, o `sits.rep` insere uma camada de complexidade sobre os experimentos, já que obriga o interessado ter a ferramenta Docker instalado em seu ambiente. Caso contrário, os experimentos não poderão ser reproduzidos.

Apesar desses pontos levantados, o `sits.rep` é vantajoso ao pesquisador por registrar os experimentos e todas as dependências necessárias para sua reprodução, incluindo as relações entre os próprios experimentos dinamicamente. É somente necessário conhecer o identificador único, ou melhor o nome, do experimento e seus dados podem ser automaticamente disponibilizados para a função executada. Essa característica facilita a análise exploratória dos dados ao permitir a criação de novos experimentos rapidamente apenas alterando os parâmetros. Além de tudo isso, ao usar o `sits.rep`, o pesquisador ganha um ambiente reprodutível totalmente dedicado ao experimento dele de maneira automatizada. Com isto, o pesquisador pode trabalhar exclusivamente na parte mais criativa do processo, sem preocupações com relação às questões de reprodutibilidade.

5 CONCLUSÕES E TRABALHOS FUTUROS

Questões relativas à reprodutibilidade têm sido discutidas nas diversas áreas da ciência, levando à criação de ferramentas computacionais para auxiliar os pesquisadores a organizarem os artefatos de suas publicações e assim assegurar uma maior facilidade para se garantir a reprodução de pesquisas científicas. Neste contexto, o pacote `sits.rep` sistematiza a organização e armazenamento dos dados e códigos produzidos por estudos de classificações de uso e cobertura da terra usando o pacote `sits`, através de um ambiente desenvolvido exclusivamente com este propósito, diferentemente do que tem sido proposto na literatura.

Através do `sits.rep`, é possível automatizar a criação de experimentos reprodutíveis e compartilhá-los com outros pesquisadores. Devido à sobrescrita das funções usadas pelo *script* de classificação, é possível reaproveitar códigos e torná-los reprodutíveis sem qualquer alteração. O pacote `sits.rep` apresentado neste trabalho é um protótipo e não um produto finalizado. Ele está em constante desenvolvimento. Já é possível criar experimentos, executar *scripts* e encadear processos. Além disso, é possível reproduzir cada experimento de forma isolada e compartilhá-lo com outros pesquisadores. A partir da consolidação do `sits.rep`, acreditamos que a produtividade das equipes que desenvolvem códigos de classificação de uso e cobertura da terra aumente substancialmente, deixando os pesquisadores despreocupados com as questões de reprodutibilidade, podendo se dedicar exclusivamente em produzir melhores classificações.

Durante o desenvolvimento deste trabalho, foi publicado um artigo no XI `Workshop` de Computação Aplicada à Gestão do Meio Ambiente e Recursos Naturais (WCAMA), no qual é apresentado o pacote `sits.rep` e seus principais conceitos:

MARIANO, Rafael; QUEIROZ, Gilberto; ANDRADE, Pedro; SANTOS, Rafael. sits.rep: Pesquisa Reprodutível em Classificações de Uso e Cobertura da Terra. In: WORKSHOP DE COMPUTAÇÃO APLICADA À GESTÃO DO MEIO AMBIENTE E RECURSOS NATURAIS (WCAMA), 11. , 2020, Evento Online. Anais [...]. Porto Alegre: Sociedade Brasileira de Computação, 2020 . p. 51-60. ISSN 2595-6124. DOI: https://doi.org/10.5753/wcama.2020.11019.

Nos trabalhos futuros temos os seguintes objetivos:

- Executar análise de usabilidade da interface desenvolvida com usuários do pacote `sits`, com o objetivo de melhorar as funcionalidades desenvolvidas

e eventualmente levantar novas demandas para o pacote.

- Produzir um serviço em nuvem capaz de permitir que diferentes pesquisadores trabalhem em uma mesma classificação. Os experimentos serão executados em um servidor escalável capaz de processar grandes volumes de dados.

- Desenvolver funcionalidades para visualização da árvore de classificações, mostrando os nós com melhores resultados, bem como possibilitando a inclusão de novos nós de forma visual.

Todo o código fonte da ferramenta desenvolvida neste trabalho encontra-se disponível no GitHub seguindo um modelo de licença de código aberto (RafaMariano/sits.rep). O estudo de caso pode ser acessado na forma de um Jupyter Notebook (RafaMariano/sits_rep_estudo_de_caso). Os dados do MODIS utilizados no Capítulo 4 estão disponíveis para download no seguinte endereço: http://brazildatacube.dpi.inpe.br/bricks/data/MOD13Q1/.

REFERÊNCIAS BIBLIOGRÁFICAS

ANDRADE, P. **sits.validate**. 2018. Disponível em: <https://github.com/pedro-andrade-inpe/sits.validate>. Acesso em: 24 maio 2019. 23

BAKER, M. 1,500 scientists lift the lid on reproducibility. **Nature News**, v. 533, n. 7604, p. 452, 2016. 1

BARBA, L. A. Terminologies for reproducible research. **CoRR**, abs/1802.03311, 2018. Disponível em: <http://arxiv.org/abs/1802.03311>. 9, 10

BEAULIEU-JONES, B. K.; GREENE, C. S. Reproducibility of computational workflows is automated using continuous analysis. **Nature Biotechnology**, v. 35, n. 4, p. 342–346, 2017. 2

BEGLEY, C. G.; ELLIS, L. M. Raise standards for preclinical cancer research. **Nature**, v. 483, n. 7391, p. 531–533, 2012. 2

BEGLEY, C. G.; IOANNIDIS, J. P. Reproducibility in science: improving the standard for basic and preclinical research. **Circulation Research**, v. 116, n. 1, p. 116–126, 2015. 7

BOETTIGER, C. An introduction to docker for reproducible research. **SIGOPS Operating Sytems Reviews**, v. 49, n. 1, p. 71–79, jan. 2015. ISSN 0163-5980. Disponível em: <http://doi.acm.org/10.1145/2723872.2723882>. 7, 24, 25

BOUTHILLIER, X.; LAURENT, C.; VINCENT, P. Unreproducible research is reproducible. In: CHAUDHURI, K.; SALAKHUTDINOV, R. (Ed.). **Proceedings of the 36th International Conference on Machine Learning**. PMLR, 2019. (Proceedings of Machine Learning Research, v. 97), p. 725–734. Disponível em: <http://proceedings.mlr.press/v97/bouthillier19a.html>. 9

CAMARA, G.; ASSIS, L. F.; RIBEIRO, G.; FERREIRA, K. R.; LLAPA, E.; VINHAS, L. Big earth observation data analytics: matching requirements to system architectures. In: ACM SIGSPATIAL INTERNATIONAL WORKSHOP ON ANALYTICS FOR BIG GEOSPATIAL DATA, 5., 2016. **Proceedings...** New York, NY, USA: ACM, 2016. p. 1–6. 4

CAMARA, G.; SIMOES, R.; ANDRADE, P. R.; MAUS, V.; SáNCHEZ, A.; ASSIS, L. F. F. G. de; ALVES, L.; CARVALHO, A. X. Y. de; MACIEL, A. M.;

VINHAS, L.; QUEIROZ, G. R. de. **e-sensing/sits: Version 1.12.5**. Zenodo, dez. 2018. Disponível em: <https://doi.org/10.5281/zenodo.1974065>. 4, 20

CHIRIGATI, F.; SHASHA, D.; FREIRE, J. Reprozip: Using provenance to support computational reproducibility. In: USENIX CONFERENCE ON THEORY AND PRACTICE OF PROVENANCE, 5., 2013. **Proceedings...** USA: USENIX Association, 2013. p. 1. 3

COLLABORATION, O. S. An open, large-scale, collaborative effort to estimate the reproducibility of psychological science. **Perspectives on Psychological Science**, v. 7, n. 6, p. 657–660, 2012. 7

COLLABORATION, O. S. et al. Estimating the reproducibility of psychological science. **Science**, v. 349, n. 6251, p. aac4716, 2015. 7

DAHLMAN, C.; TANO, A.; VISHWANATH, T. **World development report 1998/1999: knowledge for development**. Washington: World Bank, 1998. 1

DEEPSENSE.AI. **neptune.ml**. 2018. Disponível em: <https://ui.neptune.ml/-/explore>. Acesso em: 10 mar. 2019. 19, 20

DOCKER. **Docker overview**. 2021. Disponível em: <https://docs.docker.com/get-started/overview/>. Acesso em: 05 fev. 2021. 24

_____. **Get started, part 1: orientation and setup**. 2021. Disponível em: <https://docs.docker.com/get-started/>. Acesso em: 05 fev. 2021. 25

_____. **What is a container? A standardized unit of software**. 2021. Disponível em: <https://www.docker.com/resources/what-container>. Acesso em: 05 fev. 2021. 24, 25

FECHER, B.; FRIESIKE, S. Open science: one term, five schools of thought. In: BARTLING, S.; FRIESIKE, S. (ED). **Opening science**. [S.l.]: Springer, 2014. p. 17–47. 7

FEHR, J.; HEILAND, J.; HIMPE, C.; SAAK, J. Best practices for replicability, reproducibility and reusability of computer-based experiments exemplified by model reduction software. **arXiv preprint arXiv:1607.01191**, 2016. 7

FREIRE, J.; BONNET, P.; SHASHA, D. Computational reproducibility: state-of-the-art, challenges, and database research opportunities. In: ACM

SIGMOD INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA, 2012. **Proceedings...** [S.l.]: ACM, 2012. p. 593–596. 7

GENTLEMAN, R.; LANG, D. T. Statistical analyses and reproducible research. **Journal of Computational and Graphical Statistics**, v. 16, n. 1, p. 1–23, 2007. Disponível em: <https://doi.org/10.1198/106186007X178663>. 2, 7

GERTLER, P.; GALIANI, S.; ROMERO, M. How to make replication the norm. **Nature**, v. 555, n. 7698, p. 580–580, 2018. 1

GIL, Y.; DAVID, C. H.; DEMIR, I.; ESSAWY, B. T.; FULWEILER, R. W.; GOODALL, J. L.; KARLSTROM, L.; LEE, H.; MILLS, H. J.; OH, J.-H.; PIERCE, S. A.; POPE, A.; TZENG, M. W.; VILLAMIZAR, S. R.; YU, X. Toward the geoscience paper of the future: best practices for documenting and sharing research from data to software to provenance. **Earth and Space Science**, v. 3, n. 10, p. 388–415, 2016. Disponível em: <https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1002/2015EA000136>. 2, 7

GOMES, R. **Docker para desenvolvedores**. [S.l.]: Leanpub, 2017. 24

GOODMAN, S. N.; FANELLI, D.; IOANNIDIS, J. P. A. What does research reproducibility mean? **Science Translational Medicine**, v. 8, n. 341, p. 341ps12–341ps12, 2016. ISSN 1946-6234. Disponível em: <https://stm.sciencemag.org/content/8/341/341ps12>. 9

GOVONI, M.; MUNAKAMI, M.; TANIKANTI, A.; SKONE, J. H.; RUNESHA, H. B.; GIBERTI, F.; PABLO, J. D.; GALLI, G. Qresp, a tool for curating, discovering and exploring reproducible scientific papers. **Scientific Data**, v. 6, p. 190002, 2019. 3, 12, 13

GREFF, K.; KLEIN, A.; CHOVANEC, M.; HUTTER, F.; SCHMIDHUBER, J. The sacred infrastructure for computational research. In: PYTHON IN SCIENCE CONFERENCE, 16., 2017. **Proceedings...** [S.l.], 2017. p. 49–56. 3, 16, 17

HEALTH, N. I. of et al. **Principles and guidelines for reporting preclinical research**. Maryland, USA: National Institutes of Health, 2015. 2

HRYCIUK, R. **Neptune – machine learning platform**. [s.n.], 2016. Disponível em: <https://deepsense.ai/neptune-machine-learning-platform-post/>. Acesso em: 25 abr. 2019. 18

IOANNIDIS, J. P. A. Why most published research findings are false. **PLOS Medicine**, v. 2, n. 8, 08 2005. Disponível em: <https://doi.org/10.1371/journal.pmed.0020124>. 1

KNUTH, D. E. Literate programming. **The Computer Journal**, v. 27, n. 2, p. 97–111, 1984. 11

KULKARNI, N.; ALESSANDRÌ, L.; PANERO, R.; ARIGONI, M.; OLIVERO, M.; FERRERO, G.; CORDERO, F.; BECCUTI, M.; CALOGERO, R. A. Reproducible bioinformatics project: a community for reproducible bioinformatics analysis pipelines. **BMC Bioinformatics**, v. 19, n. 10, p. 5–13, 2018. 25

LANDAU, W. M. The drake r package: a pipeline toolkit for reproducibility and high-performance computing. **Journal of Open Source Software**, v. 3, n. 21, 2018. Disponível em: <https://doi.org/10.21105/joss.00550>. 3

MCNUTT, M. Reproducibility. **Science**, v. 343, n. 6168, p. 229–229, 2014. ISSN 0036-8075. Disponível em: <https://science.sciencemag.org/content/343/6168/229>. 1

NATIONAL ACADEMIES OF SCIENCES ENGINEERING MEDICINE. **Reproducibility and replicability in science**. The National Academies Press, 2019. ISBN 978-0-309-48616-3. Disponível em: <https://www.nap.edu/catalog/25303/reproducibility-and-replicability-in-science>. 9, 10

NOSEK, B. A.; ALTER, G.; BANKS, G. C.; BORSBOOM, D.; BOWMAN, S. D.; BRECKLER, S. J.; BUCK, S.; CHAMBERS, C. D.; CHIN, G.; CHRISTENSEN, G.; CONTESTABILE, M.; DAFOE, A.; EICH, E.; FREESE, J.; GLENNERSTER, R.; GOROFF, D.; GREEN, D. P.; HESSE, B.; HUMPHREYS, M.; ISHIYAMA, J.; KARLAN, D.; KRAUT, A.; LUPIA, A.; MABRY, P.; MADON, T.; MALHOTRA, N.; MAYO-WILSON, E.; MCNUTT, M.; MIGUEL, E.; PALUCK, E. L.; SIMONSOHN, U.; SODERBERG, C.; SPELLMAN, B. A.; TURITTO, J.; VANDENBOS, G.; VAZIRE, S.; WAGENMAKERS, E. J.; WILSON, R.; YARKONI, T. Promoting an open research culture. **Science**, v. 348, n. 6242, p. 1422–1425, 2015. ISSN 0036-8075. Disponível em: <https://science.sciencemag.org/content/348/6242/1422>. 2

NÜST, D.; GRANELL, C.; HOFER, B.; KONKOL, M.; OSTERMANN, F. O.; SILERYTE, R.; CERUTTI, V. Reproducible research and giscience: an evaluation using agile conference papers. **PeerJ**, v. 6, p. e5072, 2018. 7

NÜST, D.; KONKOL, M.; PEBESMA, E.; KRAY, C.; SCHUTZEICHEL, M.; PRZIBYTZIN, H.; LORENZ, J. Opening the publication process with executable research compendia. **D-Lib Magazine**, v. 23, n. 1/2, 2017. 3, 10, 11

O'NEILL, G.; BORRELL-DAMIAN, L.; KUSTER, S.; SWART, M. **Open science definition**. 2015. Disponível em: <https://www.fosteropenscience.eu/taxonomy/term/100>. Acesso em: 20 maio 2019. 8

OSTERMANN, F. O.; GRANELL, C. Advancing science with vgi: Reproducibility and replicability of recent studies using vgi. **Transactions in GIS**, v. 21, n. 2, p. 224–237, 2017. 7

PENG, R. D. Reproducible research in computational science. **Science**, v. 334, n. 6060, p. 1226–1227, 2011. ISSN 0036-8075. Disponível em: <http://science.sciencemag.org/content/334/6060/1226>. 1, 7

PIERRO, B. de. Comunicação científica sem barreiras. **Pesquisa FAPESP**, n. 276, 2019. Disponível em: <https://revistapesquisa.fapesp.br/2019/02/08/comunicacao-cientifica-sem-barreiras/>. Acesso em: 23 maio 2019. 2

PLESSER, H. E. Reproducibility vs. replicability: a brief history of a confused terminology. **Frontiers in Neuroinformatics**, v. 11, p. 76, 2018. ISSN 1662-5196. Disponível em: <https://www.frontiersin.org/article/10.3389/fninf.2017.00076>. 9

PONTIKA, N.; KNOTH, P.; CANCELLIERI, M.; PEARCE, S. Fostering open science to research using a taxonomy and an elearning portal. In: INTERNATIONAL CONFERENCE ON KNOWLEDGE TECHNOLOGIES AND DATA-DRIVEN BUSINESS, 2015. **Proceedings...** New York: Association for Computing Machinery, 2015. Disponível em: <https://doi.org/10.1145/2809563.2809571>. 8, 9

PRINZ, F.; SCHLANGE, T.; ASADULLAH, K. Believe it or not: how much can we rely on published data on potential drug targets? **Nature Reviews Drug Discovery**, v. 10, n. 9, p. 712, 2011. 1

QUEIROZ, G. R. de; FERREIRA, K. R.; VINHAS, L.; CAMARA, G.; COSTA, R. W. da; SOUZA, R. C. M. de; MAUS, V. W.; SANCHEZ, A. Wtss: um serviço web para extração de séries temporais de imagens de sensoriamento remoto. In: SIMPÓSIO BRASILEIRO DE SENSORIAMENTO REMOTO, 2015. **Anais...** São José dos Campos: INPE, 2015. p. 7553–7560. 20

RANIS, G.; IRONS, M.; HUANG, Y. Technology change: sources and impediments. In: FITZGERALD, V.; HEYER, J.; THORP, R. (ED.). **Overcoming the persistence of inequality and poverty**. [S.l.]: Springer, 2011. p. 45–72. ISBN 978-1-349-32087-5. 1

REICH, M.; LIEFELD, T.; GOULD, J.; LERNER, J.; TAMAYO, P.; MESIROV, J. P. Genepattern 2.0. **Nature Genetics**, v. 38, n. 5, p. 500–501, 2006. 14, 15

RESEARCH INFORMATION NETWORK NESTA. **Open to all? Case studies of openness in research**. London: RIN, 2010. 7

SALMI, J. **Study on open science. Impact, implications and policy options**. 2015. Disponível em: <https://op.europa.eu/en/publication-detail/-/publication/b0906f2d-8907-11e5-b8b7-01aa75ed71a1>. Acesso em: 20 maio 2019. 7

SIMOES, R.; MACIEL, A.; ANDRADE, P.; SANTOS, L.; CAMARA, G.; PICOLI, M. **Satellite image time series classification script of Mato Grosso State in Brazil: 2001-2017**. Zenodo, jun. 2019. Disponível em: <https://doi.org/10.5281/zenodo.3237686>. xiv, 43, 44, 45, 46, 49, 50

SIMOES, R.; PICOLI, M.; CâMARA, G.; MACIEL, A.; SANTOS, L.; ANDRADE, P.; IPIA, A.; FERREIRA, K.; CARVALHO, A. X. Land use and cover maps for mato grosso state in Brazil from 2001 to 2017. **Scientific Data**, v. 7, 2020. 43, 44, 47, 53

SOILLE, P.; BURGER, A.; MARCHI], D. D.; KEMPENEERS, P.; RODRIGUEZ, D.; SYRRIS, V.; VASILEV, V. A versatile data-intensive computing platform for information retrieval from big geospatial data. **Future Generation Computer Systems**, v. 81, p. 30 – 40, 2018. ISSN 0167-739X. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0167739X1730078X>. 4

STODDEN, V.; MIGUEZ, S. Best practices for computational science: software infrastructure and environments for reproducible and extensible research. **Journal of Open Research Software**, v. 2, n. 1, 2013. 7

SUBRAMANIAN, V. R.; SAKO, Y. **Omniboard**. 2018. Disponível em: <https://vivekratnavel.github.io/omniboard/#/README>. Acesso em: 02 mar. 2019. 18

TOMMASO, P. D.; CHATZOU, M.; FLODEN, E. W.; BARJA, P. P.; PALUMBO, E.; NOTREDAME, C. Nextflow enables reproducible computational workflows. **Nature Biotechnology**, v. 35, n. 4, p. 316–319, 2017. 3

VASILEVSKY, N. A.; BRUSH, M. H.; PADDOCK, H.; PONTING, L.; TRIPATHY, S. J.; LAROCCA, G. M.; HAENDEL, M. A. On the reproducibility of science: unique identification of research resources in the biomedical literature. **PeerJ**, v. 1, p. e148, set. 2013. ISSN 2167-8359. Disponível em: <https://doi.org/10.7717/peerj.148>. 1, 7

VAUX, D. L.; FIDLER, F.; CUMMING, G. Replicates and repeats—what is the difference and is it significant?: a brief discussion of statistics and experimental design. **EMBO Reports**, v. 13, n. 4, p. 291–296, 2012. 7

VICENTE-SÁEZ, R.; MARTÍNEZ-FUENTES, C. Open science now: a systematic literature review for an integrated definition. **Journal of Business Research**, v. 88, p. 428–436, 2018. 8