BEST PRACTICES

Pedro M. Sousa Bezerra

1 Introdução

Este diário apresenta uma compilação de dicas úteis e boas práticas a serem adotadas para realizar pesquisas computacionais de forma reprodutível. Reprodutibilidade em pesquisas é uma questão que tem sido cada vez mais discutida, tendo em vista o questionamento acerca da validade de diversos trabalhos publicados em importantes revistas acadêmicas que não puderam ser reproduzidos. Entretanto, embora a prevenção contra fraudes em pesquisas seja o aspecto que trouxe à tona muitas das discussões sobre o assunto, fazer pesquisa reprodutível tem muitos outros benefícios. O principal deles é contribuir ao objetivo principal pelo qual se conduz uma pesquisa: permitir o avanço científico. A reprodutibilidade garante ao pesquisador que um trabalho, de sua própria autoria ou não, possa ser reproduzido e continuado, tornando o conhecimento científico mais acessível e difundido.

2 Ferramentas de Reprodutibilidade em Pesquisa Computacional

Diversas ferramentas estão disponíveis para auxiliar o pesquisador a realizar sua pesquisa de forma reprodutível. As ferramentas variam conforme a área de pesquisa e os elementos envolvidos na reprodutibilidade. Em pesquisa computacional, os principais elementos de reprodutibilidade são: dados, código, documentação, ambiente e *workflow*. É importante definir o mais cedo possível quais ferramentas serão adotadas para realizar o trabalho de forma reprodutível desde o começo. Tornar reprodutível um trabalho que já está pronto pode exigir muito esforço extra e muitas vezes se mostrar uma tarefa impossível, já que muitas coisas que deveriam ter sido registradas se perderam.

2.1 Dados

Serviços de armazenamento em nuvem, como *Dropbox* e *Google Drive*, não são adequados para publicar e distribuir os dados de uma pesquisa, pois a permanência dos arquivos não é garantida - os serviços podem ser descontinuados ou você pode perder o acesso a seus dados se a sua assinatura expirar ou se o espaço de armazenamento gratuito for reduzido futuramente. Além disso, é mais difícil manter os dados de forma organizada neste tipo de serviço. Procure utilizar repositórios cuja finalidade é armazenar dados para propósitos científicos. Por exemplo, na área de *Machine Learning*, temos o *UCI Machine Learning Repository* [1] e o *Kaggle* [2]. *Dryad* [3] e *figshare* [4] são outros exemplos de repositórios utilizados para disponibilizar dados científicos.

Procure disponibilizar os dados de forma clara e organizada. Caso algum tipo de pré-processamento de dados seja realizado, não se esqueça de disponibilizar os dados brutos e os dados processados. Outro ponto que deve ser levado em conta é a licença de distribuição dos dados. Se sua pesquisa utiliza dados que não foram coletados por você, verifique sob qual licença eles foram distribuídos e quais as consequências disto na sua pesquisa - pode ser que os dados não possam ser utilizados para o fim que você deseja, o que pode invalidar a publicação do seu trabalho. Se você coletou dados e vai distribui-los, certifique-se de utilizar uma licença adequada, especialmente se sua pesquisa envolver seres humanos ou animais, para proteger as partes envolvidas de eventuais litígios.

2.2 Código

Novamente, não é recomendado armazenar e distribuir seu código em serviços de armazenamento em nuvem, como *Dropbox* e *Google Drive*. Apesar deste tipo de serviço facilitar a sincronização de arquivos entre diferentes estações de trabalho, pode ser muito difícil reverter alterações indevidas no código ou retorná-lo a um estado anterior, tarefas recorrentes ao longo do desenvolvimento de um programa. Assim, é extremamente recomendado a utilização de uma ferramenta de controle de versionamento de código, como o Git. Com comandos simples, você pode sincronizar os arquivos entre diferentes máquinas, salvar o estado do seu código ao longo do desenvolvimento e reverter os arquivos para qualquer estado registrado, facilitando muito o desenvolvimento e a reprodutibilidade do trabalho. Plataformas

como *GitHub* [5] e *Bitbucket* [6] são exemplos de ferramentas que utilizam o Git para controle de versionamento, além de oferecer em conjunto repositórios para armazenar e distribuir o código fonte.

Não deixe de escolher uma licença para o seu código ao distribui-lo publicamente. É importante para garantir que seu trabalho não seja utilizado para fins não desejados, e para isentá-lo da responsabilidade no caso de uso indevido. Reprodutibilidade e software livre andam de mãos dadas, então uma licença de código aberto é recomendada para facilitar a reprodução do seu trabalho. O site *Choose a License* [7] disponibiliza um guia prático e simples que ajuda você a escolher a licença de código aberto mais adequada para a sua situação. Para adicionar uma licença ao seu código, basta copiar o texto da licença escolhida e salvá-lo em um arquivo com nome "LICENSE.txt" na pasta raiz do seu repositório.

Para organizar seu repositório, é recomendado que você crie diretórios separados para cada tipo de arquivo. Uma sugestão de organização de diretórios é apresentada a seguir:

- **src**: contém os arquivos com código fonte de módulos e bibliotecas utilizadas pelo seu programa
- **deliver**: contém a versão final de arquivos a serem entregues, como notebooks e PDF's
- **dev**: contém versões anteriores de arquivos
- **data**: contém os dados que são utilizados pelo programa
- **figures**: contém figuras utilizadas e gráficos gerados pelo programa

Por fim, adicione um arquivo texto com nome "README.md" na pasta raiz do seu repositório e escreva instruções para as pessoas que visitam o seu repositório, com informações sobre o seu trabalho, referências, e passos a serem executados para reproduzir o programa. Não se esqueça de adicionar um diagrama ilustrando o *workflow* do seu programa, apresentando as suas entradas e saídas e as sequências de passos necessários para produzir os resultados. Você pode utilizar uma ferramenta como o *Draw.io* [8] para a confecção deste diagrama. O arquivo "README.md" deve ser escrito com a linguagem *Markdown*. Um guia de utilização da linguagem está disponível em *Markdown Guide* [9].

2.3 Documentação

Quanto mais automatizada for a geração dos resultados do seu trabalho, mais reprodutível ele será, e isto inclui a geração da documentação que irá apresentar os resultados da sua pesquisa. Dentre as ferramentas disponíveis para tal fim, podemos destacar ambientes de programação interativa, como o *Jupyter Notebook* [10] e *Google Colab* [11], que permitem a elaboração de notebooks em IPython com texto escrito em *Markdown* e código escrito em diversas linguagens de programação. Essas ferramentas são úteis para apresentar ou demonstrar de forma ilustrativa o trabalho realizado no formato de um *paper* executável, permitindo que o leitor possa interagir com o documento e verificar como alterações no código podem influenciar o resultado final. A ferramenta *nbconvert* [12] permite realizar a conversão de notebooks em IPython para LaTeX [13]. Assim, é possível produzir um documento PDF em formato de artigo científico a partir de um *paper* executável.

Em alguns casos, entretanto, um ambiente de programação interativa não é adequado para garantir a reprodutibilidade do trabalho, dependendo da complexidade do programa, como por exemplo no caso de códigos com tempo de execução muito elevado ou que dependam de aplicações externas. Nestes casos, o *paper* executável pode acabar não incluindo os resultados da pesquisa de forma completa. Assim, o uso de editores on-line em LaTeX, como o *Overleaf* [14], pode ser útil para complementar a documentação.

2.4 Ambiente

Para garantir que seu código seja reproduzido em diferentes ambientes de programação, é preciso manter uma lista completa e atualizada das dependências do seu programa e suas respectivas versões: sistema operacional, linguagens de programação, bibliotecas, pacotes e aplicações utilizadas. Esta lista deve estar presente no arquivo "README.md" do seu repositório.

Ainda que a sua lista esteja completa, diversos fatores podem dificultar a reprodução do seu código em outros computadores: por exemplo, seu sistema operacional pode conter algum pacote pré-instalado que você utiliza, mas que não fazia ideia da existência. Assim, para garantir a reprodutibilidade do trabalho, é extremamente recomendado o uso de ferramentas de virtualização, que irão permitir a execução do seu programa em um ambiente isolado. O *Docker* [15] surge como a principal ferramenta para distribuir imagens do ambiente de programação utilizado. De posse desta imagem, o usuário pode facilmente gerar um contêiner e executar o seu programa, mesmo sem ter as dependências instaladas, visto que um contêiner promove isolamento de recursos do sistema operacional a nível de software. Outra

opção é o uso de máquinas virtuais, que são bem mais pesadas do que contêineres, mas úteis nos casos em que se deseja simular também o isolamento em nível de hardware. Uma ferramenta comumente utilizada para este fim é a *Oracle VM VirtualBox* [16].

2.5 *Workflow*

As ferramentas de *workflow* (não confundir com o diagrama do *workflow* que deve ser adicionado ao seu repositório) se preocupam com a proveniência dos dados, ou seja, fazem um registro de tudo que foi necessário para produzir uma determinada saída a partir de uma entrada: dados brutos, código fonte, valores de parâmetros, ambiente de programação etc. Por mais que o controle de versionamento de código permita retornar o programa a estados anteriores, nem sempre é possível reproduzir resultados anteriores, mesmo tendo a mesma versão do código, devido aos outros fatores, que são devidamente registrados por uma ferramenta de *workflow*. Exemplos de ferramentas são o *Pachyderm* [17] e o *Sacred* [18]. Este último possui comandos simples de aprender, e se utilizado desde o princípio, requer pouco esforço extra para tornar qualquer execução do seu trabalho reprodutível. Já o *Pachyderm* facilita a utilização de *pipelines* de dados, e mostra-se mais útil quando seu código já se encontra em um estágio mais avançado de desenvolvimento.

3 Do's and Don'ts

Esta seção contém algumas dicas práticas do que se deve fazer e do que se deve evitar ao realizar pesquisa computacional de forma reprodutível.

3.1 Do's

- Se o seu programa lida com números aleatórios, tenha certeza de que todos os geradores de números aleatórios utilizados pelo seu código foram devidamente semeados e de que todos as sementes utilizadas foram devidamente registradas. Em Python, por exemplo, os geradores das bibliotecas *random*, *numpy.random* e *Tensorflow* são independentes e devem ser semeados caso utilizados.
- Fique atento aos ambientes de programação utilizados. Se você instalou o Anaconda e já possuía bibliotecas em Python antes de instalá-lo, você talvez tenha de instalá-las novamente usando o *conda*, pois o ambiente padrão de programação do Anaconda possui um repositório de bibliotecas diferente. Outra alternativa é adicionar o caminho padrão das bibliotecas do Sistema Operacional no ambiente padrão do Anaconda.
- Ao utilizar o Anaconda para instalar novas bibliotecas, certifique-se de que o diretório onde o Anaconda foi instalado possui permissões de escrita. Caso contrário, a instalação irá falhar.
- Sempre salve o estado do seu código ao final de uma sessão, mesmo que você tenha certeza que vai trabalhar na mesma máquina no dia seguinte. Um imprevisto pode modificar a sua rotina e comprometer seu dia de trabalho caso você trabalhe em uma máquina diferente com uma versão desatualizada do código. Se estiver utilizando Git, o que é recomendado, isto equivale a utilizar o comando *git push* para publicar on-line os *commits* que você realizou.
- Cuidado ao realizar operações matemáticas com variáveis que são utilizadas como índices de vetores ou listas, por exemplo. Em muitas linguagens de programação, índices devem ser variáveis inteiras, e operações como divisão podem converter uma variável para ponto flutuante e gerar erros no código. Certifique-se de converter a variável para inteira ao utilizá-la como índice.
- Se o seu código realiza operações de leitura e/ou escrita em arquivos localizados em diretórios externos ao do código fonte, certifique-se de que está utilizando o caminho relativo dos arquivos, e tenha cuidado ao mover pastas, pois o caminho relativo pode mudar.
- Para validar o seu programa, antes de executar os testes definitivos, execute testes com instâncias menores dos seus experimentos para ter certeza de que o código não contém erros. Assim, evita-se desperdício de tempo com longas execuções.
- Para verificar se o seu trabalho é reprodutível, execute desde o começo as suas próprias instruções de reprodução em outra estação de trabalho. Se possível, solicite a outra pessoa para fazê-lo por você.

3.2 Don'ts

- Não adie a escolha de decisões relativas às ferramentas de reprodutibilidade que você pretende utilizar. Se você começar a fazer pesquisa de forma reprodutível desde o começo, não haverá trabalho extra necessário para tornar o trabalho reprodutível futuramente. Além disso, algumas ferramentas de reprodutibilidade não

são aplicáveis em todos os contextos, dependendo da complexidade do projeto, do estágio de desenvolvimento do algoritmo e do número de pessoas envolvidas.

- Se você está escrevendo sua documentação em mais de uma língua, tenha certeza de que todas as sentenças foram devidamente traduzidas. Não deixe partes do texto disponíveis apenas em uma língua (ex: *paper* executável em inglês e instruções do repositório *GitHub* em português), para evitar erros de tradução por parte dos leitores que podem comprometer a reprodutibilidade.
- Ao utilizar Git para controle de versionamento do código, não deixe para fazer um *commit* apenas no final do dia. Não economize nos *commits*, por menor que seja a alteração que você fizer no seu código, já que, dependendo do seu programa, pequenas modificações podem modificar drasticamente os seus resultados. Quanto mais versões do seu código você tiver, mais protegido você estará.
- Se o seu programa lida com números aleatórios, tenha em mente que, em abientes de programação interativa, como *Jupyter Notebook*, executar células em ordens diferentes irá produzir resultados diferentes caso os geradores de números aleatórios não tenham sido re-semeados. Para reproduzir os resultados de um notebook, evite executar células fora de ordem.
- Evite confeccionar um *paper* executável que apresente apenas os resultados finais do seu trabalho que não refletem alterações nos trechos com código (ex: um *paper* com figuras que foram geradas por código externo ao *paper*). Não há sentido em utilizar um ambiente com programação interativa se o *paper* será redigido como se fosse um documento em LaTeX.
- Evite utilizar os valores padrão de funções provenientes de bibliotecas sem defini-los explicitamente, uma vez que os valores padrão podem mudar em versões futuras da biblioteca e modificar o comportamento da função. Se você realizou uma atualização de pacotes/bibliotecas recentemente e os seus resultados foram alterados, esta pode ser uma das razões.

Referências

[1] UCI Machine Learning Repository. `https://archive.ics.uci.edu/`.

[2] Kaggle: Your Home for Data Science. `https://www.kaggle.com/`.

[3] Dryad Digital Repository. `https://datadryad.org/`.

[4] figshare - credit for all your research. `https://figshare.com/`.

[5] GitHub: The world's leading software development platform. `https://github.com/`.

[6] Bitbucket: The Git solution for professional teams. `https://bitbucket.org/`.

[7] Choose a License: Choose an open source license. `https://choosealicense.com/`.

[8] Flowchart Maker & Online Diagram Software. `https://www.draw.io/`.

[9] Markdown Guide. `https://www.markdownguide.org/`.

[10] Project Jupyter. `https://jupyter.org/`.

[11] Google Colab. `https://colab.research.google.com/`.

[12] nbconvert: Convert Notebooks to other formats. `https://nbconvert.readthedocs.io/en/latest/`.

[13] LaTeX - A document preparation system. `https://www.latex-project.org/`.

[14] Overleaf, Editor LaTeX Online. `https://pt.overleaf.com/`.

[15] Get Started with Docker. `https://www.docker.com/get-started`.

[16] Oracle VM VirtualBox. `https://www.virtualbox.org/`.

[17] Pachyderm - Scalable, Reproducible Data Science. `https://www.pachyderm.io/`.

[18] Sacred's documentation. `https://sacred.readthedocs.io/en/latest/`.