RESUMO
*Software livre e desenvolvimento de trabalhos científicos: o R como exemplo a ser seguido*

A replicabilidade de resultados de pesquisas é importante para assegurar que os procedimentos seguidos foram adequados. O uso de software livre facilita a replicabilidade dos resultados em nível profundo porque o próprio código fonte do software é disponibilizado para inspeção. Devido à dinâmica social de produção de software, a qualidade dos softwares livres, particularmente dos necessários ao trabalho científico, não é inferior à qualidade dos softwares proprietários, sendo, pois, recomendável o uso de software livre em trabalhos científicos. O R é apresentado como exemplo de software livre que facilita a replicabilidade de resultados em trabalhos que envolvem análise estatística de dados.

**Palavras-chave:** replicabilidade, software livre, R.

ABSTRACT

The replication of research results is important to ensure adequate procedures were followed. Free software makes replication easier because the software' source code is available for inspection. Due to the social dynamics of software development, the quality of free software, especially the applications used in scientific research, is usually higher than the quality of proprietary software. Hence it is recommended the use of free software in scientific research. R is presented as an example of free software that makes it easier to replicate the results of data analyses.

**Keywords:** replication; free software; R.

# Software Livre e Desenvolvimento de Trabalhos Científicos: o R como exemplo a ser seguido

Jakson Alves de Aquino[1]

## INTRODUÇÃO

Um dos critérios mais aceitos de cientificidade do conhecimento é a sua falseabilidade: as teorias científicas devem fazer previsões de fenômenos cuja ocorrência as corrobore e de fenômenos cuja ocorrência as falseiem. Mas quando a observação de um determinado fenômeno está de acordo com uma teoria, existem duas possibilidades: (1) a teoria previu corretamente o fenômeno ou (2) houve algum erro de observação que levou à produção de um dado falsamente positivo. O raciocínio inverso é válido para fenômenos que falseiem uma teoria: (1) ou a teoria está errada ou (2) um erro de observação produziu um falso negativo.

Para minimizar os riscos de produção de falsos positivos e de falsos negativos, o trabalho científico deve seguir uma série de procedimentos metodológicos gerais e de técnicas específicas para cada área de conhecimento. Um cuidado geral a se tomar é fazer uma descrição suficientemente detalhada dos procedimentos seguidos para se chegar a determinado resultado. Assim, outros cientistas poderão repetir o procedimento descrito e deverão encontrar os mesmos resultados. Novamente, o raciocínio anterior se aplica: se os resultados forem diferentes, ou o relatório de pesquisa original não explicitou de forma suficientemente clara e completa os procedimentos seguidos, ou o segundo cientista não seguiu os procedimentos corretamente.

É claro que o raciocínio acima é esquemático, e raramente os resultados são replicáveis com a facilidade implícita acima. Nas ciências sociais, são comuns e importantes pesquisas

1 Tem doutorado em ciências humanas pela Universidade Federal de Minas Gerais e é professor da área de Ciência Política do Departamento de Ciências Sociais da Universidade Federal do Ceará. Com experiência no desenvolvimento de modelos baseados em agentes, suas áreas atuais de interesse incluem cultura política, métodos quantitativos de pesquisa e políticas públicas. Email: jaa@ufc.br

de cunho mais qualitativo e exploratório, que permitem um rico conhecimento de algum aspecto delimitado e específico da realidade social, mas que não permitem fazer generalizações porque, devido ao pequeno número de casos, é difícil distinguir tendências gerais de idiossincrasias dos sujeitos estudados. Pesquisas qualitativas são mais apropriadas para o levantamento de hipóteses explicativas do que para o teste de hipóteses.

Mas, mesmo nas ciências sociais, existem pesquisas que, em princípio, deveriam ter seus resultados replicáveis. As pesquisas que fazem uso de procedimentos estatísticos para analisar um grande número de casos simultaneamente, se feitas corretamente, e se bem descritas nos seus relatórios de pesquisa, deveriam ser passíveis de replicação por outros cientistas (ROCHA et al., 2013). O R, um software livre amplamente usado na análise estatística de dados, é uma ferramenta útil na produção de pesquisas replicáveis.

Nas próximas seções, farei uma breve explanação sobre software livre e sobre as motivações de seus desenvolvedores e farei uma apresentação sucinta do R, sem a intenção de ensinar a usá-lo. Na conclusão, retomarei a discussão sobre replicabilidade, argumentando que software livre em geral, e o R em particular, são alternativas mais adequadas para o trabalho científico do que softwares proprietários.

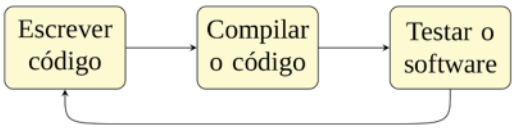
## 2. SOFTWARE LIVRE

Software é um conjunto de instruções usadas pelo computador para executar tarefas. A linguagem "compreendida" pelo computador é uma linguagem binária, extremamente difícil de ser lida por humanos, mesmo pelos mais bem treinados nessas linguagens. Mas quase todo software é escrito em uma linguagem de programação mais próxima da linguagem humana, usando termos em inglês que significam, por exemplo:

- Se x for igual a y, então, execute a seguinte tarefa...
- Repita a execução da tarefa seguinte até x se tornar igual a y...

O texto contendo as instruções escritas na linguagem de programação é chamado de código fonte. O código fonte de algumas linguagens de programação, como C, C++, Fortran e Pascal, devem ser traduzidos para linguagem de máquina (compilados) por um outro software (compiladores) antes de serem executadas pelo computador. Outras linguagens, como Perl, Python e R, têm seu código lido por um interpretador que se encarrega, no momento da execução, de fazer a tradução. Uma vantagem dos software produzidos a partir da compilação de uma linguagem de programação é que, por não precisarem de tradução no momento da execução, eles são geralmente executados mais rapidamente do que aqueles baseados em linguagens interpretadas. O desenvolvimento de software com linguagem que requer compilação, como ilustrado pela Figura 1, basicamente, segue o ciclo: (1) escrever em linguagem de alto nível (linguagem de programação) código novo ou correções a código já produzido; (2) usar o compilador para traduzir o código para linguagem de baixo nível (linguagem de máquina); (3) executar o código em busca de defeitos no funcionamento.

**Figura 1: Ciclo de desenvolvimento de código compilado**




Uma vantagem das linguagens interpretadas é que, por não precisarem ser compiladas, têm ciclos de desenvolvimento mais simples e rápidos.

Quando um software é escrito em linguagem que precisa ser compilada, o desenvolvedor tem a opção de distribuir apenas o código binário (o software já compilado), guardando para si o código fonte. O usuário recebe o software em linguagem binária, não tendo acesso ao código fonte, não podendo inspecionar o código para saber como ele funciona e, eventualmente, fazer ajustes necessários. O código fonte permanece como segredo do seu desenvolvedor. No caso de linguagens que são interpretadas no momento da execução, é necessário distribuir o próprio código fonte.

Os dois tipos software mais comuns, no que se refere à licença de uso e distribuição, são o software proprietário e o software livre. O primeiro tipo é desenvolvido por pessoas ou empresas que vendem aos usuários uma licença restrita de uso. O usuário é proibido de compartilhar o software com outras pessoas. Em alguns casos, é definido o número de cópias que podem ser instaladas. Ou seja, alguém que precise do software em casa e no local de trabalho, se não comprar duas licenças de uso, estará, efetivamente, proibido de compartilhar o software até consigo próprio.

As licenças dos softwares considerados livres também não dão aos usuários a propriedade sobre o software, mas permitem a livre distribuição do software e, mais do que isso, obrigam que o código fonte seja distribuído juntamente com o código compilado ou, pelo menos, esteja disponível para download sem nenhuma restrição de acesso. São permitidas a modificação do código fonte e a distribuição de cópias modificadas. A licença de software livre mais difundida, a GNU GPL, obriga que as versões do software com alteração mantenham a mesma licença, mas há outras licenças que permitem que o código fonte seja utilizado em softwares proprietários.

Muitos softwares livres são desenvolvidos por programadores individuais, mas projetos de maior complexidade, como Linux, LibreOffice e Firefox, podem ser vistos como esforços coletivos de resolver problemas comuns (HERTEL; NIEDNER; HERRMANN, 2003, p. 1162). O software livre, uma vez produzido e distribuído, pode ser usado por todos os interessados indistintamente. Ele é, portanto, um bem público, e uma análise da racionalidade de agentes egoístas nos faz esperar que bens públicos não serão providos (OLSON, 1999). Se é possível para um desenvolvedor de software vender cópias de seu produto em linguagem de máquina e com licenças que obrigam outros interessados a também comprarem o software, por que alguns desenvolvedores distribuem seus softwares gratuitamente, inclusive disponibilizando o código fonte? Por que não tentam obter algum rendimento do seu trabalho? Os produtores de software livre são, aparentemente, altruístas, mas o altruísmo não está presente de forma notável em outras indústrias, o que aumenta a necessidade de explicar a existência de software livre (WU; GERLACH; YOUNG, 2007). Diversos pesquisadores, por meio de entrevistas e questionários, fizeram levantamentos das principais motivações dos desenvolvedores de software livre, como veremos a seguir.

Um dos motivos que leva um programador a se envolver com o desenvolvimento de software livre é a necessidade de resolver um problema específico que lhe afeta (XU; JONES; SHAO, 2009). Muitas vezes, a forma mais fácil de fazer isso é escrever o próprio software, tomando como ponto de partida software livre já existente. O programador pode desenvolver um novo software ou adicionar uma nova funcionalidade a um software existente. Para tomar a decisão de desenvolver o software, o programador precisa fazer uma estimativa de quanto tempo decorrerá até que alguma outra pessoa execute a tarefa. Para alguns, será melhor esperar. Para outros, será melhor tomar a iniciativa de desenvolver (BITZER; SCHRETTL; SCHRÖDER, 2007). Os desenvolvedores de software livre se beneficiam diretamente do código que produzem porque o fazem para si próprios. Uma vez desenvolvido, o custo de disponibilizar o software na internet é baixo (LAKHANI; HIPPEL, 2003).

A necessidade pessoal pode ser uma explicação para muitos casos de contribuição, mas a disponibilização do trabalho para o público em geral ainda precisa ser explicada porque o desenvolvedor não tem obrigação de distribuir o software que produziu. Um dos motivos para o compartilhamento pode, de fato, ser chamado de altruísmo. Quanto mais os indivíduos percebem sua contribuição como significativa, mais ficam satisfeitos em contribuir, ou seja, há uma satisfação intrínseca em ajudar os outros (HERTEL; NIEDNER; HERRMANN, 2003, p. 1164).

Outra motivação é o posicionamento ideológico favorável ao software livre. A ideologia pode ser vista neste caso como a racionalização de reações emotivas ao debate software livre versus software proprietário. O software livre é por vezes visto como não alienante por permitir aos usuários programadores o domínio completo do software. É claro que esse argumento não se aplica com a mesma força aos usuários não programadores. Para Richard Stallman, um dos maiores defensores do software livre, as leis de patentes, que já foram úteis ao progresso, atualmente são entraves ao desenvolvimento da tecnologia. Toda invenção é baseada em trabalhos anteriores e, por isso, patente seria algo injusto. Não faria sentido uma pessoa monopolizar o direito a royalties de um produto por décadas apenas por ter feito pequenos acréscimos à sua tecnologia. Para Stallman, todo software deveria ser livre. Software proprietário seria uma imoralidade.

O prazer de realizar um trabalho criativo também foi identificado como um dos motivos para o desenvolvimento de software livre. Há uma zona de trabalho cujo nível de complexidade maximiza a satisfação com a realização da atividade. Se a tarefa for muito simples, é percebida como entediante. Se for muito complexa, o sujeito ficará frustrado com sua incapacidade de resolver o problema. Níveis intermediários de complexidade deixam o sujeito satisfeito com o próprio desempenho e aprendizado, com a sensação de que descobriu algo novo, de que superou um desafio. Segundo, Lakhani (2003), participantes de projetos de software livre podem estar em busca de tarefas que estejam de acordo com seu nível de habilidade em programação. Nestas circunstâncias, programar se torna algo divertido.

Quando um programador disponibiliza seu trabalho na internet, outros programadores interessados no desenvolvimento do mesmo tipo de software, como já mencionado, podem achar mais vantajoso contribuir para a melhoria do software já publicado do que desenvolver algo completamente novo. Em muitos casos, o programador que tomou a iniciativa de publicar seu código aprenderá técnicas novas com as contribuições recebidas, melhorando, sua habilidade pela revisão do seu trabalho por pares (LAKHANI, 2003).

Mas uma das motivações mais mencionadas pelos pesquisadores do tema é o interesse em melhorar a própria reputação como programador, embora os programadores não admitam isso com frequência. Os créditos pela contribuição em projetos de software livre ficam registrados online, acessíveis a qualquer empregador em potencial (LAKHANI, 2003). Qualquer pessoa pode acompanhar pela internet o histórico do desenvolvimento de um software livre. Mesmo quem não tem nenhum conhecimento de programação, pode avaliar a competência e capacidade de liderança de um programador a partir da análise do histórico de modificações no código e dos comentários às mudanças feitas por outros programadores. O histórico de contribuição a um projeto de software livre é um indicador de competência mais confiável para um empregador do que uma carta de recomendação escrita por um amigo generoso ou por um patrão ansioso por se livrar de um empregado problemático (ZEITLYN, 2003, p. 1290).

Não é apenas na produção de código que o trabalho de voluntários é importante para o desenvolvimento de software livre. O suporte ao software também costuma ser gratuito e realizado por voluntários, sendo os desenvolvedores do software, por vezes, minoria entre os que ajudam os usuários a tirar dúvidas. As razões para se ajudar nessa tarefa são semelhantes às motivações para desenvolver o software: motivos ideológicos (para ajudarem a causa do software livre), por premiação intrínseca (gostar de ajudar), para aumentar a reputação e melhorar as chances de conseguir emprego (LAKHANI; HIPPEL, 2003, p. 937). O custo de tempo para quem ajuda é pequeno (5 minutos em média) porque geralmente quem responde já sabe a resposta (LAKHANI; HIPPEL, 2003). Por um lado, a razão mais importante para os provedores de informações dedicarem tempo à leitura das perguntas postadas em fóruns e listas de discussão é para aprenderem com as respostas. Por outro lado, ao postarem suas próprias respostas, eles também se beneficiam com os comentários que se seguem (LAKHANI; HIPPEL, 2003).

Informação fornecida como resposta a perguntas em fóruns perde valor comercial porque se torna de domínio público. Mas se o indivíduo não postar sua resposta rapidamente, outro postará antes dele e, portanto, o valor comercial será perdido de qualquer maneira e a reputação do outro é que crescerá (LAKHANI; HIPPEL, 2003, p. 935). Esse é um dos motivos pelos quais o suporte gratuito a software livre é muitas vezes mais rápido e eficiente do que o suporte pago fornecido por desenvolvedores de software proprietário.

Além das motivações pessoais dos programadores, há outras razões objetivas favoráveis ao desenvolvimento de software livre. Bens materiais são de propriedade exclusiva, ou seja, se forem dados, vendidos ou trocados, deixam de pertencer a um proprietário e passam a pertencer a outro proprietário. Até mesmo quem empresta um objeto fica, temporariamente, sem a posse do mesmo. Software é apenas um tipo de informação (instruções para computadores), e, como todo tipo de conhecimento, pode facilmente ser compartilhado por um número indefinido de pessoas. A posse de um software por uma pessoa não necessariamente priva outras da posse do mesmo bem (ZEITLYN, 2003, p. 1287). Apenas a aceitação de algumas convenções sociais — como a de propriedade intelectual — e de legislação protegendo essas convenções torna possível a existência de software proprietário. Também são necessárias convenções sociais e leis para garantir a posse de objetos físicos, mas a própria natureza dos objetos e os interesses particulares dos indivíduos tornam a implementação dessas leis algo bem menos problemático.

Embora seja desenvolvido, majoritariamente, por voluntários, o software livre não fica aquém do software proprietário em termos de qualidade. Essa afirmação é verdadeira pelo menos para os programas que despertam o interesse de grande número de desenvolvedores e de usuários. Devido ao caráter público dos debates em torno do desenvolvimento do software, muitas alternativas de inovação e solução de problemas são discutidas, sendo escolhida a melhor (HERTEL; NIEDNER; HERRMANN, 2003). Se as decisões tomadas não forem às que conduzirem a um software mais eficiente e eficaz, os insatisfeitos têm total liberdade para fundar um novo projeto. Se o novo projeto realmente estiver no rumo certo, a maioria dos desenvolvedores abandonará o projeto antigo em favor do novo. Os usuários, embora não contribuam com código, ajudam a encontrar falhas e a testar o software nas mais diversas condições. Por isso, é importante que também sejam numerosos.

## 3. R COMO EXEMPLO

Escrito, em sua maior parte, em C, tem versões previamente compiladas distribuídas gratuitamente para Linux, Windows e Mac OS X. Trata-se de um software livre, estando seu código disponível para download no sítio do programa.[2] O R é um interpretador da linguagem de mesmo nome, ou seja, é o software utilizado para converter para linguagem de máquina e imediatamente executar código escrito em R.

A linguagem R deriva da linguagem S, usada no software proprietário S-Plus. O software R foi inicialmente desenvolvido por Robert Gentleman e Ross Ihaka, professores do Departamento de Estatística da Universidade de Auckland, na Nova Zelândia. Atualmente, um grupo de 21 desenvolvedores, incluindo os dois pioneiros, tem direito de escrita no código fonte do R. As discussões acerca do desenvolvimento ocorrem publicamente em uma lista de correio eletrônico, havendo também listas de suporte aos usuários (inclusive, uma lista brasileira, sediada na Universidade Federal do Paraná).

Além do R propriamente e de uma seleção de cerca de 25 pacotes adicionais, a equipe do R mantém na internet um repositório com milhares de pacotes desenvolvidos e mantidos por voluntários. Essa rede de distribuição de software é conhecida como Rede Abrangente de Arquivos do R, ou CRAN, na sigla em inglês e, no momento em que este texto era escrito, podiam ser encontrados na CRAN 7.264 pacotes. Os arquivos são sincronizados diariamente por uma rede de servidores de páginas de internet que espelham o repositório. No Brasil, cinco instituições públicas integram a rede.[3] O objetivo da constituição da rede é permitir aos usuários instalarem e atualizarem pacotes a partir de um servidor próximo, permitindo downloads mais rápidos, reduzindo o tráfego de dados na internet e evitando demanda excessiva ao servidor central.

Os pacotes submetidos à CRAN são revisados para garantir que se integrarão bem ao R. Todas as funções públicas (que são visíveis para o usuário) devem estar documentadas. A equipe do CRAN não verifica se o pacote, de fato, executa corretamente os procedimentos anunciados na documentação. Cabe ao usuário entrar em contato com o autor do pacote para comunicar a existência de problemas ou solicitar o desenvolvimento de novas funcionalidades. Ao fazer isso, mesmo que não tenha conhecimento suficiente para pessoalmente desenvolver um pacote para o R, o usuário estará contribuindo para a melhoria do software

2 http://www.r-project.org

3 Fiocruz, UESC, UFPR, USP Piracicaba e USP São Paulo.

existente e, portanto, aumentando não só a própria produtividade mas também a de outras pessoas que, no futuro, poderiam vir a enfrentar o mesmo problema ou necessitar das mesmas funcionalidades.

O R é um programa de linha de comando. Isso significa que ele recebe comandos que devem ser enviados linha por linha. Ao ser iniciado, o R apresenta um símbolo de prompt, >, indicando que ele não está executando nenhum comando e, portanto, está pronto para receber novas instruções. Programas de linha de comando não têm uma interface gráfica com menus para a execução de ações, sendo necessário memorizar os comandos. Se o usuário não digitar comandos, nada acontecerá. Em programas com um sistema de menus, basta procurar para encontrar os comandos necessários para a maioria das tarefas de análise dos dados.

Para quem desenvolve um software de linha de comando, entretanto, adicionar menus correspondentes aos comandos exige um enorme esforço adicional. Além de definir uma função, seria preciso definir um item de menu que executaria a função, com a especificação de uma caixa de diálogo que permitisse o controle de todos os argumentos opcionais da função. Para o programador, desenvolver uma função é um trabalho estimulante porque é o enfrentamento de um desafio. Adicionar menus seria trabalhoso e mecânico, ou seja, algo extremamente aborrecido e que poucos farão voluntariamente.

Apesar de facilitar o uso inicial de um software, o uso de um sistema de menus e caixas de diálogo num trabalho científico de análise de dados resulta em um obstáculo importante para que se atenda a uma das recomendações da metodologia científica: a replicabilidade de resultados. Após executar centenas ou milhares de cliques em itens de menu e botões, nem mesmo o próprio pesquisador sabe mais exatamente o que fez para chegar aos seus resultados. E o pior, se tiver cometido qualquer erro durante o procedimento, se descobrir o erro, terá que repetir a execução de todos os cliques (o que pode levar dias). Com a produção de um script da análise dos dados, basta corrigir o erro e executar novamente o script (o que geralmente demora alguns segundos). Todo o procedimento de análise fica disponível para inspeção por outros pesquisadores, que podem mais facilmente atestar a correção da análise realizada ou apontar falhas. O rápido descobrimento e correção de falhas implica num mais rápido progresso na produção do conhecimento. Um trabalho científico que publica, como anexo, os scripts elaborados para a análise dos dados é mais facilmente falseável do que outro que não faz o mesmo e, portanto, atende melhor a um dos critérios mais aceitos de cientificidade.

Uma outra vantagem do uso de scripts é a possibilidade de reaproveitamento de código quando há a necessidade de se fazer análise de dados semelhante a outra realizada anteriormente. Por exemplo, após passar vários dias elaborando o código para a análise dos dados eleitorais de uma eleição específica, é possível adaptar o código para os dados de outra eleição em uma ou duas horas.[4]

É possível combinar códigos R e LaTeX[5] num único documento, chamado Rnoweb. Do-

4 Compare, por exemplo, a análise dos dados eleitorais que fiz em 2010 e em 2014, disponíveis em http://xxx.xxxxx.xxx.xx/xxxxxx.xxx.

5 LaTeX é uma linguagem usada para editoração de texto, possuindo comandos para produzir negrito, itálico, espaçamento entre linhas, tabelas, bem como inserir referências bibliográficas, etc... A linguagem LaTeX é muito usada nas ciências exatas, entre outros motivos, pela facilidade de edição de fórmulas mate-

cumentos Rnoweb são processados pelo próprio R, sendo gerado um novo documento com código LaTeX puro. O output gerado pelo código R é inserido nesse novo documento. Existem vários pacotes do R que geram o código LaTeX de uma tabela a partir de objetos do R que possam ser interpretados como constituídos de linhas e colunas (matrizes, bancos de dados e sumários de análises de regressão, por exemplo). Com o uso do pacote knitr, as figuras geradas pelo código R são automaticamente salvas no formato PDF e o código LaTeX necessário para a inserção das figuras é adicionado ao documento LaTeX. Além disso, é possível inserir o valor de qualquer objeto do R diretamente no texto sendo escrito. Pode-se, por exemplo, usar código do R para gerar o número correspondente a um coeficiente de regressão. É claro que é muito mais simples copiar manualmente o coeficiente de regressão do que programar o R para gerá-lo, mas a vantagem deste procedimento é que o número será automaticamente atualizado sempre que houver qualquer alteração na manipulação ou na análise dos dados. O documento LaTeX gerado pelo R é processado por outro programa (pdflatex ou xelatex), sendo gerado um arquivo PDF. Esquematicamente, o processo ocorre como ilustrado pela Figura 2.

**Figura 2: Processamento de código Rnoweb**


A desvantagem do uso desse sistema é a necessidade de programar não apenas em R, mas também em LaTeX. A vantagem é que não é necessário copiar e colar ou inserir manualmente tabelas e figuras. Ou seja, não existe necessidade de manualmente substituir todas as tabelas e figuras após uma alteração de última hora na análise de dados porque temos a certeza de que todas as tabelas e figuras correspondem à última versão do script.

## 4. CONCLUSÃO

A replicabilidade de experimentos e análises estatísticas é um fator importante na produção de conhecimento científico porque quando um experimento ou análise de dados é replicado temos maior segurança de que os procedimentos seguidos foram bem descritos, podendo ser devidamente avaliados por outros cientistas, o que reduz o risco dos resultados encontrados serem apenas erros metodológicos. Com software proprietário, a comunidade científica até pode replicar resultados utilizando o mesmo software, mas sem acesso ao código fonte dos softwares utilizados, a inspeção é incompleta. Software livre, por definição, tem o código fonte disponível para inspeção por qualquer um que tenha competência para examiná-lo, e muitos cientistas estão suficientemente bem preparados para fazer isso. Essa disponibilidade do código fonte aumenta a profundidade da inspeção possível dos procedimentos seguidos em experimentos e análises de dados. Software livre, portanto, é mais consonante com o princípio de replicabilidade do que software proprietário.

É claro que um nível mais profundo de replicabilidade pode não ser justificativa suficiente para o uso de um software livre se sua qualidade for inferior à de softwares proprietários. Mas, como vimos, software livre usualmente tem alta qualidade, superando suas alternativas proprietárias, principalmente nos casos em que os usuários são numerosos e compe-

máticas.

tentes como programadores. Esse é justamente o caso dos cientistas de muitas áreas. Especificamente, muitos estatísticos e alguns cientistas sociais são também competentes como programadores. Mesmo os que aprenderam a programar apenas na própria linguagem R contribuem usando o programa e alguns de seus milhares de pacotes adicionais e, eventualmente, fazendo sugestões de melhoria aos desenvolvedores do software. Por essas razões, o R se tornou um bom exemplo de software livre de alta qualidade.

A comunidade de usuários do R é numerosa, e crescente, enfrentando, entretanto, alguns obstáculos a uma maior difusão entre cientistas sociais brasileiros. Nossos cursos de graduação e de pós-graduação na área de ciências sociais são, em sua maioria, deficientes no ensino de metodologia quantitativa (SOARES, 2005). Os estudantes são pouco ou nada treinados em projetos de pesquisa em que o número de casos analisados é grande (centenas ou milhares de casos) e em que o objetivo da pesquisa é fazer generalizações. Em poucos cursos, há ênfase no uso de ferramentas estatísticas. Os cientistas sociais brasileiros estão mais habituados a trabalhar com textos e discursos do que com ferramentas lógicas e matemáticas. Assim, o esforço inicial para usar um software como o R, que exige habilidade de programador, é maior para estudantes de ciências sociais do que para estudantes de outras áreas.

Além disso, os softwares proprietários concorrentes do R investem em propaganda e em técnicas comerciais de fidelização. É comum, por exemplo, a distribuição de licenças de uso para estudantes a baixo custo ou mesmo gratuitas. Após aprenderem a fazer suas análises de dados com um software, o pesquisador precisa estar muito motivado a usar o R para deixar temporariamente a fase produtiva em que se encontra e voltar à fase de aprendizado. Mas, no longo prazo, o esforço compensa.

E, na verdade, o esforço não precisa ser tão grande como se pensa. Existe muito material de livre acesso na internet ensinando a dar os primeiros passos no R, e existem pacotes do R — como o Deducer e Rcmdr — que criam menus semelhantes aos de softwares proprietários, exibindo exemplos de código que facilitam o aprendizado ao tornar menor o número de comandos a serem memorizados pelos iniciantes.

## REFERÊNCIAS

BITZER, Jürgen; SCHRETTL, Wolfram; SCHRÖDER, Philipp J.H. Intrinsic motivation in open source software development. Journal of Comparative Economics, v. 35, p. 160–169, 2007.

HERTEL, Guido; NIEDNER, Sven; HERRMANN, Stefanie. Motivation of software developers in Open Source projects: an Internet-based survey of contributors to the Linux kernel. Research Policy, v. 32, p. 1159–1177, 2003.

LAKHANI, Karim. Why hackers do what they do: uderstanding motivation effort in free/open software projects. Working Paper of the MIT Sloan School of Management, v. 4425–03, 2003.

LAKHANI, Karim R; HIPPEL, Eric von. How open source software works: "free" user-to--user assistance. Research Policy, v. 32, p. 923–943, 2003.

OLSON, Mancur. A lógica da ação coletiva: os benefícios públicos e uma teoria dos grupos sociais. São Paulo: Edusp, 1999. [1965].

ROCHA, Enivaldo Carvalho da Rocha et al. A importância da replicabilidade na ciência política: o caso do SIGOBR. Revista Política Hoje, v. 22, n. 2, p. 213–229, 2013.

SOARES, Gláucio Ary Dillon. O calcanhar metodológico da ciência política no Brasil. Sociologia, Problemas e Práticas, n. 48, p. 27–52, 2005.

WU, Chorng-Guang; GERLACH, James H. b; YOUNG, Clifford E. An empirical analysis of open source software developers' motivations and continuance intentions. Information & Management, v. 44, p. 253–262, 2007.

XU, Bo; JONES, Donald R.; SHAO, Bingjia. Volunteers' involvement in online community based software development. Information & Management, v. 46, p. 151–158, 2009.

ZEITLYN, David. Gift economies in the development of open source software: anthropological reflections. Research Policy, v. 32, p. 1287–1291, 2003.


UNIVERSIDADE
FEDERAL
DE PERNAMBUCO

CFCH
CENTRO DE FILOSOFIA E CIÊNCIAS HUMANAS

Departamento de
Ciência Política

Programa de Pós-Graduação
em Ciência Política