A very simple and fast way to access and validate algorithms in reproducible research

G. Stegmayer, M. Pividori and D.H. Milone

sinc(*i*), Research Institute for Signals, Systems and Computational Intelligence,

CONICET/FICH-UNL

Ciudad Universitaria UNL - (3000) Santa Fe - Argentina

gstegmayer@sinc.unl.edu.ar

Abstract

The reproducibility of research in bioinformatics refers to the notion that new methodologies/ algorithms and scientific claims have to be published together with their data and source code, in a way that other researchers may verify the findings to further build more knowledge upon them. The replication and corroboration of research results are key to the scientific process and many journals are discussing the matter nowadays, taking concrete steps in this direction. In this journal itself, a very recent opinion note has appeared highlighting the increasing importance of this topic in bioinformatics and computational biology, inviting the community to further discuss the matter. In agreement with that article, we would like to propose here another step into that direction with a tool that allows the automatic generation of a web interface, named *web-demo*, directly from source code in a very simple and straightforward way. We believe this contribution can help make research not only reproducible but also more easily accessible. A web-demo associated to a published paper can accelerate an algorithm validation with real data, wide-spreading its use with just a few clicks.

Keywords: *reproducible research, web-demo, open-source*

1. Introduction

Reproducing computational results as part of the publication process, as well as for replicating algorithms and validation of meaningful results, has been a matter of interest to the academic community and widely discussed from some years now, receiving increasing attention lately [1,2,4,5,6]. Many important journals are discussing the matter and have taken concrete steps in this direction, allowing articles to be supplemented with online material. For example, since October 2014 papers in *Nature* journals should make computer code accessible [3]. However, as stated very recently in an opinion note of this journal itself [10], the availability only of the source code is not enough to guarantee its reproducibility. The authors have also highlighted that this topic is of particular interest to the bioinformatics community nowadays, due to the fact that computational biology methodologies and algorithms are becoming more and more complex and hard to reproduce and validate. In fact, reproducing computational results in any discipline is generally difficult and a very time-consuming task. It should be noted that full and open access to the original source code alone, although the simplest way of making research available, has many issues associated. Even when results can be replicated, the local setup of a desktop version takes a considerable time and effort. There are, often, many tools dependencies and operating systems restrictions to be solved. A certain compiler must be installed in order to run the code, with many parameters that have to be configured manually, often involving installation of internal and external libraries, as well as the setup of toolboxes and packages that often require a corresponding license. Not to mention dependability on operating system type and version. These are all complex and very time-consuming tasks, which usually turn to be a long trial-and-error process. And all of this is necessary even if a reader or a reviewer just wants to make a simple and preliminary trial test of a source code to determine if it runs correctly, and if it is really capable of processing some data.

In order to address these issues, a few approaches can be found. For example, it has been suggested that taking advantage of recent increases in Internet bandwidth, disk space and open

source software, research can be made effectively reproducible through virtual reference environments [10]. The authors proposed that every publication presenting computational results should be accompanied with a virtual environment (virtual machine) stored “in the cloud” where it can be executed and tested. This environment should be a minimal implementation of the software in order to reproduce some or all of the results of a publication. Other authors have highlighted the need for further efforts for making data and source code more integrated with publications [7,8,9]. For example, in 2011, the computer science community addressed the question of how to reproduce computational results actually “within” a research article. Executable papers have been proposed to do that [11], where authors have the ability to embed pieces of executable code and data into a paper. This allows readers to explore the computational section of an article at the same time they are accessing it. The code can be executed actually while reading the article, which provides readers (and reviewers) with an interesting and seamless experience. Unfortunately, it has to be said that this proposal has remained closed to the computer science community alone and this initiative has not been extensively used since proposed, mainly because it is rather complex and assumes an expert user. The author must specify manually a certain programming language and corresponding version from a closed list to run the code, as well as perform the configuration of many parameters that are required. In order to correctly embed the script into a paper or webpage, a specific series of commands must be used. Besides, access to external software such as for the folding of RNA sequences, the alignment and comparison of sequences, or access to a genome annotation database, has not been explicitly considered. These issues show the need for a new tool, simple to use and configure, intuitive for any user, as well as fast to execute. For example, this tool should be capable of automatically detecting inputs and outputs from the source code. Then, it should provide an adequate interface to test the code. Furthermore, web access to this interface would be very important in order to allow simple and fast testing, as well as worldwide access. In fact, the web is easy to access for anybody, anywhere. With a web interface, a script

can be tested in just two clicks. However, the generation of such web interfaces for a given source code is not simple nor direct, and it would require, again, expertise on web programming and time.

As a solution to these issues, we would like to present an automatic generator of web interfaces. It simply receives as input an original source code, it automatically detects the inputs and outputs of the algorithm and generates a simple and adequate web interface for it, named web-demo, together with an URL to access it. It is really just that simple, and fast. The source code of the web-demo builder is available to download for free and can be installed in any personal computer or server. Furthermore, it can also be accessed directly in the cloud.

The main idea behind the web-demo builder is to provide, not only the original implementation of a software or algorithm through its open source code, but also the replicability of the corresponding results through the automatic generation of a web interface to that source code. Although a web interface does not show the technical details behind a code, it can provide simpler and wider access to it. This can certainly favour its accessibility, dissemination and use. All of these can be achieved in a very simple and fast way with the proposed web-demo builder. Finally, it should be highlighted that a web-demo could be generated as part of a published paper under review in order to validate data and code in a very short period of time, certainly decreasing reviewers workload.

2. Web-demo builder: a fast and simple step towards reproducible research and more easy access

The web-demo builder is a web application designed to easily generate web interfaces from code written in R, Python or Matlab, which are among the most popular programming platforms used in the bioinformatics community.

With the web-demo builder, the author of a paper can make his research not only reproducible but also quickly accessible and easy to test with several possible inputs. Once the author generates a web-demo with the web-demo builder, the web-demo users (for example, a journal reviewer or a reader) can try it, with their own inputs (data and algorithm parameters) or by using sample files that can be included into the web-interface. In the creation of the web-demo, the author can use any of the environments that are already available inside the current version of the web-demo builder, with the standard configuration for R, Python and Matlab for bioinformatics.

The web-demo builder takes a source code and produces a web-demo, an accessible and simple web interface to the code, in just two steps. The author only needs to write a main function with corresponding inputs and outputs, and pack the code into a zip file. In step 1, the zip is uploaded to the web-demo builder, which automatically detects inputs and outputs. Step 2 consists in specifying the type of inputs and outputs: an input parameter might be a simple string or number, or a file; the output type could be a number, a text table, or an image with, for example, a histogram or a plot showing a convergence or error evolution of a function. These two simple steps produce a web-demo, accessible by a single URL, which is the web interface that can be used by journal reviewers or readers. The resulting web-demo is a plain web interface to interact with: fill the input fields and press the submit button to execute the script and obtain the outputs in the same web page. It should be noted that each web-demo generated is “frozen” with its own environment (in its own software container). This is an advantage because the web-demo will continue to work, providing the same results of the original paper, in spite of any package upgrade in the underlying system.

All of this is done without special effort nor knowledge in web development by the author. From a journal reviewer or reader viewpoint, who accesses the web-demo only needs to specify the

corresponding inputs (in addition, sample files can be provided to quickly test it). After that, the algorithm runs into a virtual environment (e.g. a software container, which is a modern and lightweight virtualization technology) in the server, and the outputs are shown in the web interface. Note that the user of the web-demo does not need to know nor download the toolboxes or configure any other tools employed by the author of the paper, he or she just interacts with the web interface. This represents a remarkable reduction of effort for the first glimpse of, for example, a reviewer, who is still able to download the full source code behind the web-demo to further investigate its implementation. Furthermore, if the author includes private data inside the web-demo, a reviewer can just run it, changing for example some parameters of the algorithm, and obtain the results, but never accessing the data directly, thus preserving its confidentiality.

More languages can be easily incorporated to the web-demo builder through pieces of code named *connectors*, which define the format of the inputs and outputs of a function, and the command line to run a script in the corresponding language. Also, code written in different languages could be called from the main script, provided that the corresponding interpreter is installed in the server or virtual environment where a web-demo runs. Advanced authors can provide a special environment configuration for running their code with anything needed installed (extra packages and/or external software)[1]. However, in most of the cases, authors have to upload only source code, using the provided standard environments.

It is possible to take a quick look at the web-demo builder for R, Python and Matlab, by accessing it online as an instance of Amazon Elastic Compute Cloud[2]. The general

1 Detailed instructions are available at https://bitbucket.org/sinc-lab/webdemobuilder/wiki/Home, section Create your own virtual environments.

2 http://sinc.unl.edu.ar/papers/bib-2015-1-amazonaws

configurations and packages for each language installed in this instance are detailed in the project web-page[3].

Very simple scripts to generate web-demos are provided in the supported languages. These scripts deal with three steps in miRNA prediction: miRNA Folding in Matlab, miRNA Feature extraction in Python and miRNA Clustering in R. These examples are available at https://bitbucket.org/sinc-lab/webdemobuilder/wiki/Home. More examples of generated web-demos can be found in http://fich.unl.edu.ar/sinc/web-demo. The web-demo builder is available under an open source license at https://bitbucket.org/sinc-lab/webdemobuilder/src.

Conclusions

We propose a very simple and fast way of enhancing reproducible research, with the extra advantage of making it easily accessible and usable through a web interface. The main contribution of this proposal is the automatic generation of this web interface. This way, researchers in bioinformatics can concentrate on making efficient algorithms and they do not have to worry about how to make them accessible through a user interface. Furthermore, a journal reviewer or reader can have a quick look at the algorithm in a seamless way.

Key Points

1. Reproducibility, which is a valuable property of any scientific work, can be improved by making new research easily accessible to any user. Quick and easy access to interact with novel algorithms is very important for the bioinformatics community nowadays.
2. Access to a web page has become a universal resource, making web technologies the ideal way to wide spread results. However, the development of a web page is neither a simple task nor a common ability among researchers in bioinformatics.

3 For R 3.0: https://bitbucket.org/sinc-lab/webdemo-base-r3, for Python 2.7: https://bitbucket.org/sinc-lab/webdemo-base-python2, for Matlab 2013a: https://bitbucket.org/sinc-lab/webdemo-base-matlab, and for all the three languages: https://bitbucket.org/sinc-lab/webdemo-base-matlab-python2-r3

sinc(*i*) Research Center for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
G. Stegmayer, M. Pividori & D. H. Milone; "A very simple and fast way to access and validate algorithms in reproducible research."
Briefings in Bioinformatics,, Vol. 17, No. 1, pp. 180-183, 2016.

3. This work presents a web tool which can automatically create web interfaces from source code: the w*eb-demo builder*. It enables paper authors to show and share their research work quickly through the web, even if they don't have knowledge about web development. A web-demo can be accessed by reviewers and readers, which can quickly and easily run the code provided by the author through an adequate web interface.

4. The generated web-demos run in a virtual environment of the server. This virtual environment is provided by default for the web-demo builder or it can be uploaded by the author, with a specific operating system, tools, libraries, data, etc.

6. The web-demo builder allows, not only to fully reproduce results in a minimal environment precisely described, but also to quickly test an algorithm with no effort in the installation and configuration from the raw source code.

Funding and Acknowledgments

This work was supported by National Scientific and Technical Research Council (CONICET) [PIP 2013-2015 117], Universidad Nacional del Litoral (UNL) [CAI+D 2011 548] and Agencia Nacional de Promoción Científica y Tecnológica (ANPCyT) [PICT 2014 2627]. The authors would like to acknowledge Juan Pablo Vidocevich for developing part of the web-demo builder.

References

1. Bissell, M. Reproducibility: The risks of the replication drive. Nature 2013; 503:333-4.

2. Nature editorial. Journals unite for reproducibility. Nature 2014. November; 515: 7. doi:10.1038/515007a.

3. Nature editorial. Code share. Nature 2014. October; 514: 536. doi:10.1038/514536a.

4. Vandewalle, P., Kovacevic, J. and Vetterli, M. Reproducible Research in Signal Processing - What, why and how. IEEE Signal Processing Magazine 2009, 26(3): 37-47.

5. Huang Y., Gottardo R. Comparability and reproducibility of biomedical data. Briefings in Bioinformatics 2013;14(4):391-401. doi:10.1093/bib/bbs078.

6. Piwowar H., Day R.S., Fridsma D.B., Sharing detailed research data is associated with increased citation rate. PLoS ONE 2007; 2(3): e308.

7. Jasny BR, Chin G, Chong L, Vignieri S. Again, and Again, and Again … Introduction to special section on Reproducible Research. Science 2011; 334(1): 1225.

8. Peng R.D.. Reproducible Research in Computational Science. Science 2011; 334(1): 1226-1227.

9. Sandve G.K. Nekrutenko A., Taylor J., Hovig E. Ten Simple Rules for Reproducible Computational Research. PLoS Computational Biology 2013, 9(10): 1003285.

10. Harley D.G., Budden D.M. and Crampin E.J. Virtual Reference Environments: a simple way to make research reproducible. Brief Bioinform 2014; doi:10.1093/bib/bbu034.

11. Nowakowskia, P., Ciepielaa, E., Harężlaka D. et al., The Collage Authoring Environment, Procedia Computer Science 2011, 4(1): 608-617, doi:10.1016/j.procs.2011.04.064.

Georgina Stegmayer is Assistant Professor in the Department of Informatics at Universidad Nacional del Litoral (UNL), and Researcher at the National Scientific and Technical Research Council (CONICET), Argentina. Georgina's research involves machine learning, data mining and pattern recognition in bioinformatics.

Milton Pividori is a Ph.D. student in the Research Institute for Signals, Systems and Computational Intelligence, sinc(i), at CONICET-UNL. His research involves data mining and bioinformatics, with particular focus on consensus clustering.

Diego Milone is Full Professor in the Department of Informatics at UNL and Independent Research Scientist at CONICET. He is Director of the sinc(i) Institute. His research interests include statistical learning, signal processing, neural and evolutionary computing, with applications to biomedical signals and bioinformatics.

Research Institute for Signals, Systems and Computational Intelligence

Research at sinc(i) aims to develop new algorithms for machine learning, data mining, signal processing and complex systems, providing innovative technologies for advancing healthcare, bioinformatics, precision agriculture, autonomous systems and human-computer interfaces. The sinc(i) was created and

is supported by by two major institutions of higher education and research in the country: the National University of Litoral (UNL) and the National Scientific and Technical Research Council (CONICET).