

RunMyCode.org: A Research-Reproducibility Tool for Computational Sciences

Christophe Hurlin

Department of Economics
University of Orléans, France
Christophe.hurlin@univ-orleans.fr

Christophe Pérignon

Finance Department
HEC Paris, France
perignon@hec.fr

Victoria Stodden

Department of Statistics
Columbia University
New York City, USA
vcs@stodden.net

1. Introduction

Research reproducibility can be vastly improved by the open availability of the code and data that generated the results. In this paper, we present a new web-based tool that aims to improve reproducibility in computational sciences. The RunMyCode.org website gives published articles a *companion webpage* from which visitors can (1) download the associated code and data and (2) execute the code in the cloud directly through the RunMyCode.org website. This permits results to be verified through the companion webpage or on a user's local system. RunMyCode.org also permits a user to upload their own data to the companion webpage to check the code by running it on novel datasets.

We present the structure of the RunMyCode.org system in Figure 1. Researchers provide the code and data associated with their publication. Users can either use the data provided by the researchers or provide their own. Then code and data are sent to the cloud. When the computation is done, the results are sent back to the user.

The RunMyCode concept can be viewed as an attempt to provide, on a large scale, an executable paper solution. The difference between this and the executable paper approach proposed by the scientific publishers (see for the instance the Elsevier's Executable Paper Grand Challenge, 2011, <http://www.executablepapers.com>) is that the companion webpage is not encapsulated within the text of a scientific publication. In that sense, a companion webpage can be considered as providing *additional* material for a scientific publication, in particular the digital objects that permit verification and replication of the published computational results.

Of course, being able to reproduce the main findings of scientific papers is important for the scientific community itself, but it also matters for the credibility of science in society. Furthermore, reproducibility is of primary importance for governments and corporations since it is a necessary condition to convert scientific ideas into economic growth. We summarize in Figure 2 how the RunMyCode website can improve transfer of technology from the academia to society (students, corporations, administrations, general public, etc). A key feature of the website is to reduce the technical cost for users to access and use a new scientific technique.

RunMyCode has three main objectives. The first is to allow researchers to quickly disseminate the results of their research to an international audience through an on-line service. This should lead to a notably increase in the citations of certain academic articles. Second, RunMyCode aims to provide a very large community of users – potentially beyond the academic sphere – with the ability to use the latest scientific methods in a user-friendly environment, for their own data and parameter values. To date, such analyses were impossible for users without the necessary computing skills to implement the methods in specific software. Third, it allows members of the academic community (researchers, editors, referees, etc.) to replicate scientific results and to demonstrate their robustness.

RunMyCode is an international academic project founded by economics and statistics professors from Columbia University, HEC Paris, and University of Orléans (France) and engineers from CNRS (the French National Science Foundation). RunMyCode is incorporated as a non-for-profit scientific association and is funded by universities, national research agencies, and foundations.

The rest of our paper is structured as follows. In Section 2, we explain why researchers should share their code and data and why they often do not. We explain in Section 3 why, on top of *sharing* code and data, making code running in the cloud is a further step toward full reproducibility. We then take economics as an example of computational science (Section 4) and we discuss the case of code and data sharing in this fields, as well as executable code. Section 5 focuses more specifically on RunMyCode and on its functioning, while Section 6 mentions several potential partnerships and further developments for the RunMyCode initiative.

2. Why (not) sharing code and data

There are many good reasons to share the code and data associated with a scientific paper. Lerner and Tirole (2002) show that in the context of open-source, researchers can benefit from enhancement of their reputation and that of their potential value on the labor market. The availability of data and codes is related not only to the reproducibility issue, but also to the dissemination and exploitation of academic research. Having access to such resources improves the visibility of articles and their impact on both scientific community and non-academic sphere. A recent example is the V-lab (*Volatility Lab*) website launched in 2012 by Nobel laureate Robert Engle at New York University in order to ease the diffusion of the systemic risk measures proposed by Engle and his co-authors.

However, in practice, most researchers are still reluctant to share their code and data. Borgman (2007) identifies four major factors preventing systemic disclosure of code and

data: (1) lack of incentives (citations or promotion), (2) the effort required to clean data and codes, (3) the creation of a competitive advantage over other fellows, and (4) intellectual property issues. Similar impediments for reproducibility have been identified in previous work (Stodden 2009) in a survey of 723 American academic researchers. In her study, the main factors restraining researchers from making computer codes available are the time for documenting and preparing the codes (77% of subjects), the idea of having to answer questions from possible users (52%), and having no direct benefits (44%). A possible loss in future publications was also indicated as a subsequent factor by 30% of the researchers. Finally, in some research area, a significant fraction of research is conducted using proprietary data. For instance, Gandon (2010) report that 28% of the articles published in the top economics journal, the American Economic Review, used confidential data.

RunMyCode.org solves several of the problems given above confronting computational scientists who wish to engage in reproducible research. It removes the difficulty of hosting the code and data, it removes the difficulty of installing and running (even correct) code on a local computer system, and by providing the ability for users to execute the code in the cloud, it minimizes the amount of support coders and authors are asked to supply. RunMyCode.org also provides suggested citations, to help encourage a reward system that encourages code and data release, by giving credit for these scientific contributions. RunMyCode.org provides a public date of creation of the companion webpage, helping to ensure primacy to those who release code and data and encourage attribution. Perhaps most importantly, RunMyCode.org provides a central field-independent platform to facilitate both code and data sharing, and the verification of published computational results.

3. Why make code executable in the cloud?

We argue that sharing code and data would be a significant step towards research reproducibility. However, it may not be a sufficient one. A further step would be to make code running in the cloud. To make our point, we present a landmark experiment conducted by researchers in economics. McCullough, McGeary and Harrison (2006) aimed to reproduce the results of the 266 papers published in the Journal of Money, Credit and Banking between 1996 and 2003. The replication team only had to use online material associated with the 266 papers available on the journal website, which had a data availability requirement. Out of the 266 papers, 193 of them contain an empirical section and, as such, should have data and/code provided by their authors, in compliance with journal policy. In reality, 35% of the papers had no online material whatsoever, 5% had data but no code, and 4% had code written in languages not supported by the replication team. Other research confirms this is not a situation unique to economics (see Alsheikh-Ali (2011), Tenopir (2011), and Savage (2009)).

The main finding of this paper is that a small fraction of the papers with available data and code were reproduced to their full extent. Hence, sharing code and data may not always be a sufficient condition for engaging in reproducible research. Indeed, only 14 articles (7% of the sample) have been reproduced. Several reasons can explain this extremely low reproducibility rate. First, the authors of the original papers were not always careful enough when preparing the final version of the code and data uploaded on the journal website. Hence, this material is hard to use and results hard to reproduce. Second, there is typically

very little and often no explanation on how to use the online material (e.g. No readme file). This is due, in part, to the fact that the editorial boards provided no strict guidelines about the code and data submission process.

The case for executable script has recently been made in biostatistics (Peng, 2011). Indeed, in the journal *Biostatistics* each article receives a mark mentioned on the first page. “D” and “C” stand for available data and code, respectively, whereas “R” signifies a reproducible article. In the latter case, a “reproducibility review” (execution of the code on the original data) has been performed by the editor on the request of the author. The journal hence identifies four levels of reproducibility, from non-reproducibility to “gold standard”: (1) publication only, (2) publication and code, (3) publication, code and data, and (4) publication, and executable code and data.

4. An example of computational science: Economics

RunMyCode.org was first launched in economics and there are several reasons for that. Over the last few decades, economics has become more empirical and data-driven. Furthermore, economics is nowadays a highly computational discipline, far ahead of many other social sciences. Numerical computation is now ubiquitous in modern economics: statistical analysis, estimation, optimization, simulation, numerical equation solving, and the entire spectrum of econometrics. Barrou (2008) reports that the fraction of theoretical papers published in the top economics journal, *American Economic Review*, dropped from 70% in the 1970’s to 20% in the recent years. Further evidence is provided in the survey of Kim et al. (2006) of all the articles with more than 500 citations from top economics and management journals. They show that at the beginning of the 1970s, 77% of these papers were theoretical and 11% empirical. By the end of the 1990s, the proportions were reversed: 11% theoretical vs. 60% empirical.

Another reason that contributed to the development of an executable-code platform is the fact that scripts and data tend to be smaller than in many other computational sciences. There are some recent exceptions though in economics with datasets of several terabytes of high-frequency financial transaction data or shopper data at retailers.

A more recent and larger study than the Thursby one mentioned at the beginning of the chapter is that of McCullough, McGeary and Harrison (2006). They aimed to reproduce the results of all papers published in the *Journal of Money, Credit and Banking* between 1996 and 2003. In total, 193 empirical papers (which should include the data and codes used) are present among the 266 publications. The authors conclude that only 14 papers out of 193 (7%) are reproducible articles. According to the authors of the study, two main reasons explain their findings. First, authors do not always carefully prepare the datasets and associated code. Second, editors do not provide enough guidelines on how to prepare these resources so as to facilitate reproducibility analyses. To improve the replication of results, McCullough, McGeary and Harrison suggest to systemically create *readme* files and to use common formats for data.

Since the 2000’s, mainly top-ranked economics journals, such as *American Economic Review* and *Econometrica*, have created data and code/script archiving systems. However, as noted by McCullough and Vinod (2003), sharing code and data have remained on a voluntary basis

for a while. In 2004, the chief editor of the American Economic Review, Ben Bernanke, decided to make mandatory data and code submission after publication. Gandon (2010) studies the performance of this policy in 2007-2008 and shows that only 79% of the published papers could be replicated without contacting the authors. Another scientific policy recommendation would be to require the code and data associated with a scientific work prior to its publication (without making them publicly available yet).

While discussions among economics journals focused on disclosure of scripts and/or data, we are aware of only one paper advocating executable scripts. In a pioneering article, Phillips (2003), one of the best econometricians in the world, describes an internet service for automatic forecasting similar in some respects with the RunMyCode companion website concept. Phillips anticipates that the future of economic forecasting is in automatic internet-based econometric modeling, that he calls *Interactive Econometric Web Service* (IEWS). Phillips imagined a web-interface on which different forecasting methods are presented. The user is allowed to choose the parameters and options. The results are executed on a local server and displayed in the webpage as tables, graphs, etc. He then summarizes the advantages of his IEWS:

“Perhaps the main advantage of econometric web services of this kind is that they open up good practice econometric technique to a community of users, including unsophisticated users who have little or no knowledge of econometrics and no access to econometric software packages. Much as users can presently connect to financial web sites and see graphics of financial asset prices over user-selected time periods at the click of a mouse button, this software and econometric methodology make it possible for users to perform reasonably advanced econometric calculations in the same way. The web service can be made available on a 24/7 basis so that people can perform online calculations in presentations and lectures”. Phillips (2003), *The Law and Limits of Econometrics*, page 25.

Phillips’ paper has been a major source of inspiration for the RunMyCode project. The companion website proposed by RunMyCode can be seen as a generalization of Phillips’ IEWS.

5. How does RunMyCode.org work?

RunMyCode is based on the concept of a companion webpage associated with a scientific publication. It allows people to run on-line computer scripts associated with an article, the results being automatically displayed to the user as a SaaS (Software as a Service), or to download the script and demo data directly. The companion webpage is thought of as a frame of the scientific publication making it possible to both download the research resources associated with publications and/or to simply use them through the web to check the robustness, performance, and reproducibility of the results.

An example of a companion website is presented in Figure 3. A scientific paper’s companion webpage on RunMyCode.org is structured as follows. The upper panel displays information about the paper, including a direct link to the pdf file and the abstract, and the authors. The intermediate panel contains information about the coders (i.e., the researchers who wrote the code, and who may not be the original authors of the paper) along with a description of

the goal of the code. The lower panel allows the user to upload the data, selects models, and set parameters values. Finally, the green RunMyCode button launches the computation.

As shown in Figure 1, RunMyCode plays the role of an intermediary between the researchers offering the code (which may, in some cases, be different from the authors of the publication) and the users (researchers, students, public administration, private firms, etc.). RunMyCode allows researchers to create a custom companion webpage online without any particular computing skills. This is a six-step process, each of them requiring the author to give some information about the publication and the co-authors, as well as a clear description of the variables and input parameters of the computer code. The author can declare five types of inputs: scalar, vector, matrix, text, choice list, file (in this case he or she defines the type of file, such as an image file). For any other type of inputs, the author is asked to give particular recommendation to our technical team. In contrast, no information about the output is required: the companion webpage reproduces the output of the computer code (tables, figures, numerical values, text, image, etc.) as it would appear on the researcher's personal computer. The final task of the researcher is to preview and validate their companion webpage.

The RunMyCode backend can take scripts or code, where code needs to be compiled before execution and scripts are interpreted at runtime only and need not to be compiled. Currently, it is possible to create a companion webpage from code written in C++, Fortran, MATLAB, R, and RATS. More software will be added in the near future, especially Python.

Note that the creation of a companion webpage does not typically require any modification to the original script/code and as such requires no additional effort from the researcher. The source scripts are simply encapsulated and sometimes transformed into an executable on the RunMyCode system. For instance, Matlab scripts are compiled and then run with the Matlab Compiler Runtime (MCR). The MCR is a standalone set of shared libraries that enables the execution of compiled Matlab applications that do not have Matlab installed. For other software (for instance specific econometric software such as RATS etc.), the scripts cannot be transformed into an executable file. In this case, the script is simply used in batch mode. The source codes are compiled according to the recommendation provided by the author, and when the code uses some specific libraries, we use exactly the same libraries. RunMyCode runs in the Linux environment. If a code runs on a specific Windows system, we emulate a virtual machine (Windows) with the same environment as that used by the author.

During this process (called pre-production), we may introduce some additional instructions in the original script if necessary in order to 1) link it with the inputs provided by the companion websites and 2) format the results in a pdf file. Indeed, once the job is executed, a post-treatment is done from the raw results issued from the software. This post treatment is done with LaTeX: all the numerical results (tables, etc.) and all the comments (text, etc.) produced by the codes are automatically saved in tex format. An automatic program compiles these results to produce a pdf file. The visual results (figure etc.) are saved in an eps format and included in the LaTeX file during the post treatment process. Currently they are published in the same pdf file as the other numerical results. In the future, we plan to improve this mechanism in order to produce the results and the figure in html.

During this pre-production process, we check the code to ensure that the required inputs match the descriptions and constraints provided by the author. Note that this is not a

scientific validation. We only check for typical bugs (infinite loops, etc.) for the duration of the computing process and for security (malicious codes). Note that RunMyCode is responsible of the security of the codes that are submitted to our cloud provider. But, the cloud provider has also its own security rules that are not specific to RunMyCode.

Once the website is created, it enters the validation stage. First, the authors or coders validate it. Then, the editorial team checks whether the topic complies with the editorial policy of the website, similar to arXiv for example or any peer-reviewed academic journal. Finally, a technical validation of the code is undertaken, which focuses on its robustness, security, CPU requirements, and computing time.

Once the validation step is completed, the code is uploaded on the cloud and the companion webpage goes online. Companion websites can be found directly on the web, through any search engine, or starting from the RunMyCode website. Each contributor within RunMyCode is given a unique profile called a “coder page.” This permits the researcher to find and connect with people working on similar or other interesting problems. Most importantly, it offers various statistics on the visibility of their websites: number of visits, number of executions of the code, number of downloads, etc.

Developing the concept of executable papers is an important issue nowadays for the major scientific editors worldwide. Pdf publications can no longer be considered as the ultimate stage in scientific research. For example, two major conferences called “Beyond the PDF” were organized in 2011 and 2013.¹ As another example, Reed-Elsevier issued a call for tenders in 2010, for the executable paper concept. Its objective was to find ways to easily replicate the results of scientific publications. Nevertheless, to our knowledge, no functional form of the executable paper concept has been proposed so far and no published article describes or proposes such services in economics and management.

As an illustration, we show in Figure 4 how a specific result can be reproduced with a RunMyCode companion website. Consider a given study in which one of the key results is a plot of the value of a Y variable that depends on an X variable. With the companion website, one can reproduce the result published in the original paper using the same parameter values as in the paper ($n = 100$ in this example). However, as shown in Figure 4, one can also launch the computation using different parameter values ($n = 50$) and see whether the key result is robust to a change in the value of one or several parameters.

Figure 5 gives a representation of the RunMyCode system.

RunMyCode is based on a cloud computing architecture type and a message routing mechanism built on MOM (Message Oriented Middleware). The message includes the data and all the parameters needed to run the script on the cloud. For all the applications, cloud facilities are provided by the French National Research Agency (CNRS)’s TGE Adonis. The management of the jobs is done through the DTM (Distributed Task Manager) application provided by the TGE Adonis (CNRS). DTM is a lightweight tool for submitting and monitoring jobs through a local batch scheduler, gLite Grid and local Linux/Unix host. Jobs in DTM may consist of one or several tasks. The RunMyCode jobs are registered and then they are executed by DTM jobs agents in SGE or Grid. Once the post-treatment is ended, the website receives the information and displays the results to the user. If the user is still on the companion website, he can display the results by clicking on the button “view” of his/her

¹ See <https://sites.google.com/site/beyondthepdf/> and <http://www.force11.org/beyondthepdf2>

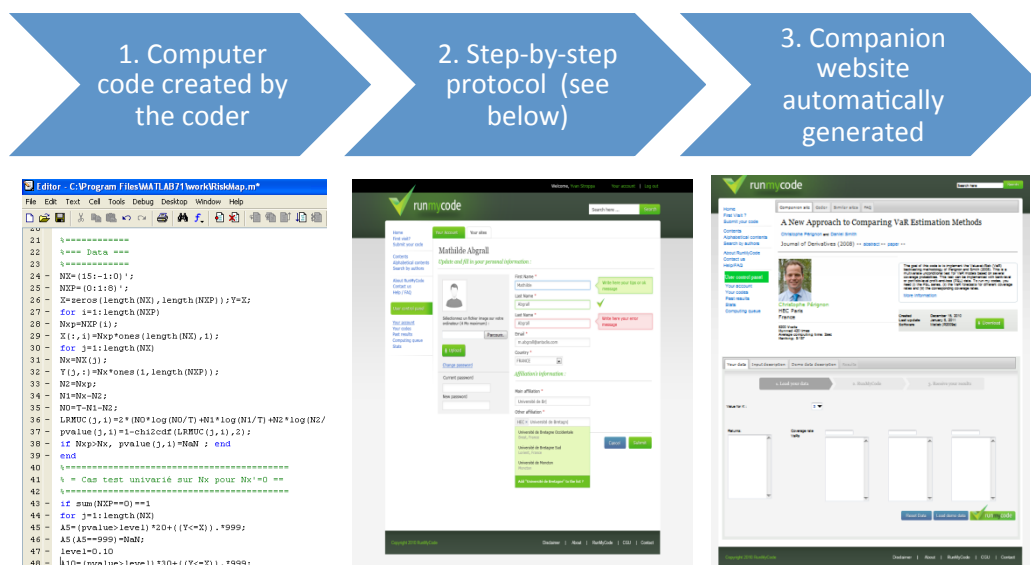
computing queue. If the user browses other sites, or if she/he is logged-out, he/she can retrieve his/her results on the tab “past results”.

There is no mechanism to check if the code halts in a reasonable time, since the length of the process may vary with the inputs or parameter choices provided by the user. In order to ensure that no process will hog the computational resources, we only fix a limit in term of CPU time (10 hours).

Currently, each job is submitted to a specific node of the computing cloud and so we do not use parallelization at this point. Because of this, the current architecture of RunMyCode generally does not provide better time performances that the user would have on his or her personal computer or personal system. On the contrary, the performance is generally worse due to the task scheduler, the check on the inputs etc. But, improvement of compute time is on our working agenda.

Box: What do I need to create a companion website?

The making of a companion website is the following. Users will be able to generate automatically their own companion website from their computer codes. They follow the following process:



In the step-by-step protocol, the coder provides information about (1) the scientific paper, (2) the coders, (3) the code and the software, (4) the inputs (e.g. variables), and (5) the outputs. The first step requires very standard information about the publication the name of the authors, the affiliations, the abstract, the DOI or the link to the publication (published article or working paper), some key words etc. The second step consists in listing the authors of the codes or the scripts. In order to avoid confusion with the authors of the scientific publication, we introduce use the term “coders.” Indeed, the coders may not be the authors of the publication, which is actually generally the case. The third step consists of declaring the main information about the code or the scripts. The coder has to upload all the required files (main codes and sub-files or library), eventually in zip format, for the execution. We also

ask for some information about the software used (the list is currently limited to Matlab, C, R, Fortran, C++, Python or RATS, although if the software can run on a linux system, RunMyCode can probably support it), the version, the architecture (32 or 64 bits) and the compiler (for the codes only). The coder has also to provide a description of the goal of the code that will be displayed on the companion website. This description may be different from the abstract of the paper and may be designed to give all the required information to the future user of the companion website. The coder has also the possibility of uploading a pdf file if this description is longer than 800 characters. Finally, the coder could also provide a copy of the results (pdf file) obtained with the demo data.

The fourth step is the most crucial. The coder is asked to describe all the inputs of the codes. For each input, the coder has to declare the type (scalar, text, vector, matrix, choice list or file), the label that it will be displayed on the companion website and the name of this variable in the code/script. The application checks in the main code if this name is present. Then, for each input the coder has to provide a value (for the scalar or text types) or a set of demo data. The coder could also provide a text description of these inputs and these demo data. These descriptions will be displayed on the companion website. Then, the coder gets a first visualization of the input form of his or her future companion website: each type of input is associated to a particular object (box for the scalar and text, choice list etc.). He/she has the possibility to design this form by dragging and dropping all these objects.

The last step consists in declaring the outputs. This step is very limited, since by default RunMyCode will reproduce the same presentation of the results (tables, figures etc.) as that the user would obtain on his personal computer. All these inputs are included in a pdf file. So this last step is only devoted to the cases where the code produces some numerical data useful to the user. In this case the coder declares the name and the label of all the corresponding variables in the code. The results contains in these variables will be stored in a csv file and can be downloaded by the user.

6. Partnerships and Expansion

To develop its operations, RunMyCode is currently partnering with scientific publishers; scientific association; editorial boards of scientific journals; conference and workshop organizers; pdf archives; digital archiving services.

References:

Dewald, W.G., J. Thursby and R.G. Anderson (1986), "Replication in Empirical Economics: The Journal of Money, Credit and Banking Project". *American Economic Review*, 76, 587-603.

Glandon, P. (2010), "Report on the American Economic Review Data Availability Compliance Project", http://www.aeaweb.org/aer/2011_Data_Compliance_Report.pdf.

Alsheikh-Ali AA, Qureshi W, Al-Mallah MH, Ioannidis JPA (2011) Public Availability of Published Research Data in High-Impact Journals. *PLoS ONE* 6(9): e24357. doi:10.1371/journal.pone.0024357

Tenopir C, Allard S, Douglass K, Aydinoglu AU, Wu L, et al. (2011) Data Sharing by Scientists: Practices and Perceptions. *PLoS ONE* 6(6): e21101. doi:10.1371/journal.pone.0021101

Savage CJ, Vickers AJ (2009) Empirical Study of Data Sharing by Authors Publishing in PLoS Journals. *PLoS ONE* 4(9): e7078. doi:10.1371/journal.pone.0007078

Ince, D.C., L. Hatton and J. Graham-Cumming (2012), "The Case for Open Computer Programs", *Nature*, 482, 485-488.

Kim, E.H., A Morse and L. Zingales (2006), "What Has Mattered to Economics since 1970?", *The Journal of Economic Perspectives*, 20(4), 189-202.

Lerner, J. and J. Tirole (2002), "Some Simple Economics of Open Source", *Journal of Industrial Economics*, 50(2), 197-234.

McCullough, B. D., K.A. McGeary and T. Harrison (2006), "Lessons from the JMCB Archive", *Journal of Money, Credit and Banking*, 38(4), 1093-1107.

Peng, R.D. (2011), "Reproducible Research in Computational Science", *Science*, 334, 1226-1229.

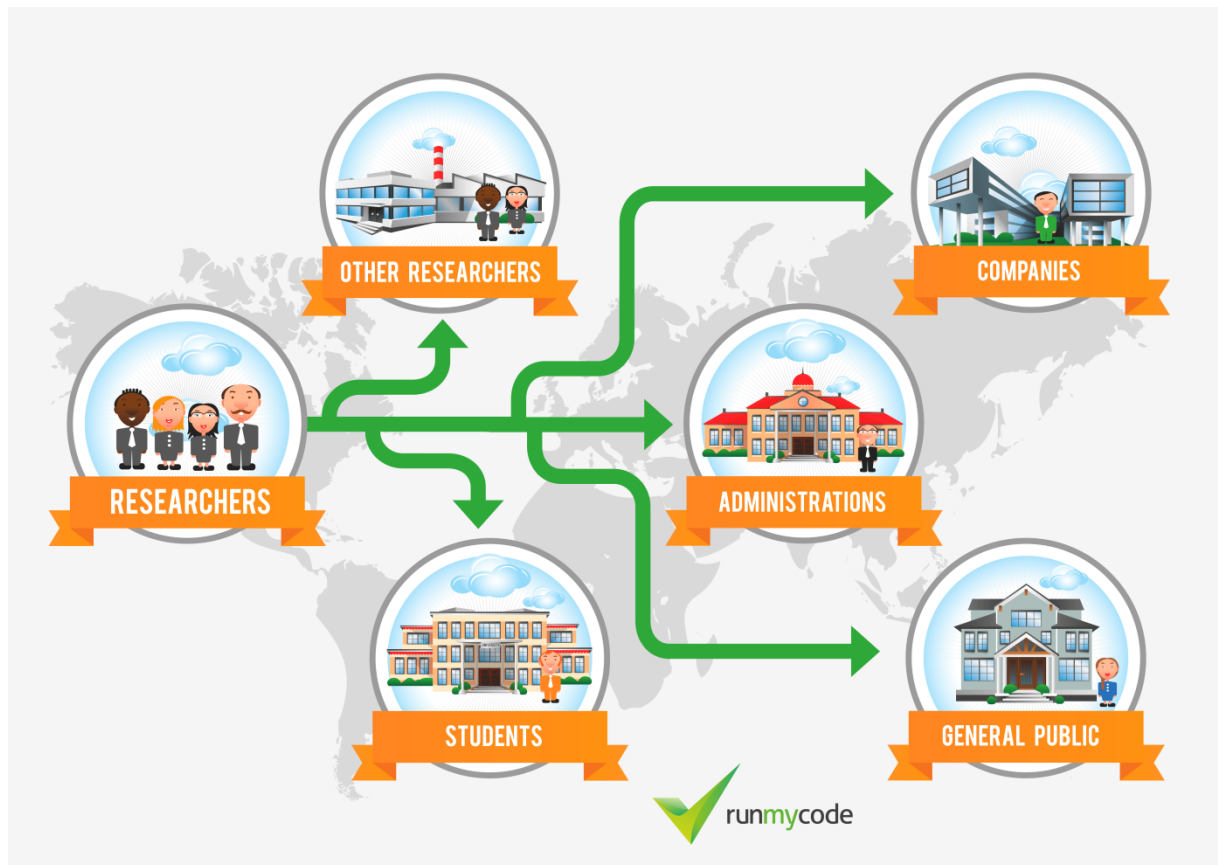
Phillips, P.C.B. (2003), "Law and Limits of Econometrics", *The Economic Journal*, 113, 26–52.

Figure 1: The RunMyCode System



Note: Researchers provide the code and data associated with their publication. Users can also provide their own data, which are sent to the cloud along with the computer code. When ready, the results are sent back to the user.

Figure 2: Improving Transfer of Technology



Note: The RunMyCode website aims to improve transfers of technology within academia (researchers to researchers and researchers to students), from the academia to companies, as well as from the academia to society (administrations, general public).

Figure 3: Example of a scientific paper’s companion webpage on RunMyCode.org

runmycode

Search here

Search

Home

First visit?

Our offering

Submit your code

Search by themes

Advanced search

Help/FAQ

Our partners

The team

Contact us

Companion site

Coders


Similar sites

FAQ

How to Forecast Long-Run Volatility: Regime Switching and the Estimation of Multifractal Processes

By Laurent E. Calvet, and Adlai J. Fisher

Journal of Financial Econometrics (2004) [Abstract](#) [Paper](#)




Laurent E. Calvet

HEC Paris

France

Coder Page



Adlai J. Fisher

University of British Columbia

Canada

Coder Page

230 Visits

68 Runs

Downloads N/A

Average computing time N/A

Rating 5

Created June 16, 2011

Last update June 16, 2012

Software Matlab R2009

Download

230 Visits

68 Runs

Downloads N/A

Average computing time N/A

Rating 5

Twitter

Facebook

Google+

Your data

Inputs description

Demo data description

Results

1. Load your data

2. RunMyCode

3. Receive your results

Returns (centered)

Number of volatility frequencies

Starting values for optimization (optional)

Load demo data

Preview data

Reset data

runmycode





Copyright 2012 RunMyCode - v1.010

About

Privacy Policy

Terms of Use

Contact



HEC PARIS

tge

DONIS

ALFRED P. SLOAN FOUNDATION

Figure 4: Reproduce and Generalize Computational Results



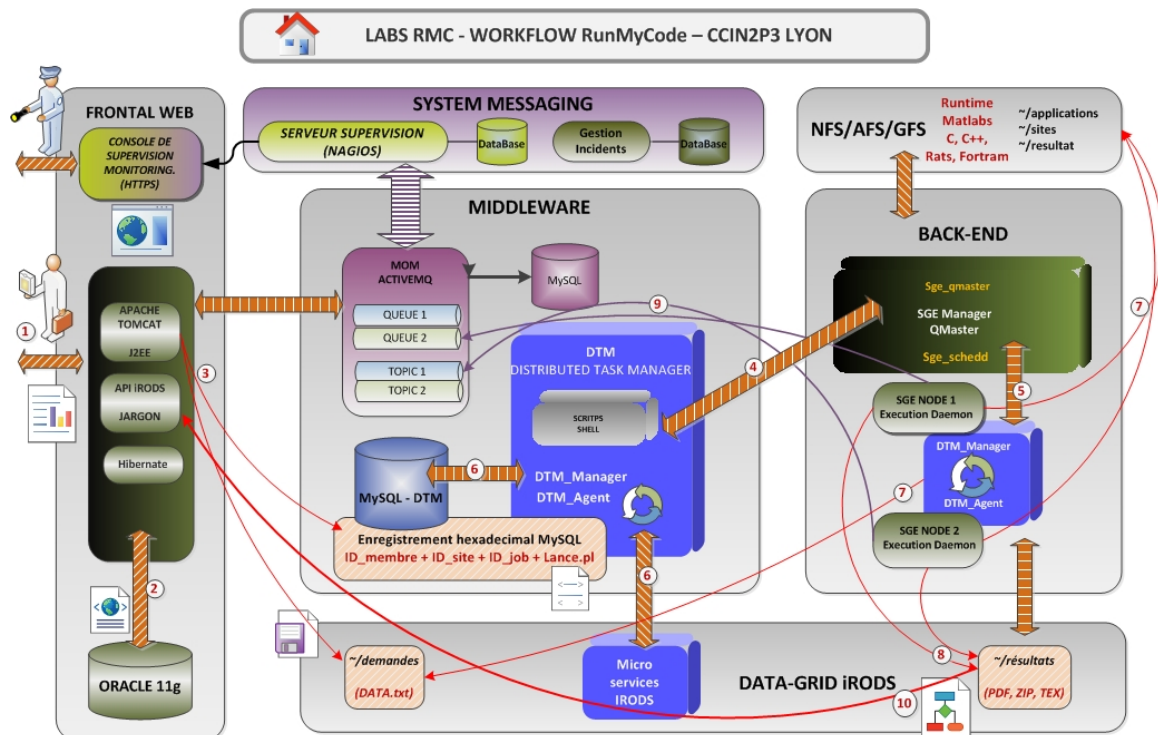


Figure 5: RunMyCode System Workflow