IEEE CIS

CDS NEWSLETTER

The Newsletter of the Technical Committee on Cognitive and Developmental Systems

Developmental Robotics
Machine Intelligence
Neuroscience
Psychology

Volume 13, number 2
Fall 2016

Editorial

Computational Modelling Across Disciplines

Pierre-Yves Oudeyer

Inria and Ensta ParisTech, France

Editor,
Chair of the Technical Committee on Cognitive and Developmental Systems, 2016

pierre-yves.oudeyer@inria.fr

Computational modelling of cognitive and developmental living systems bridges the diverse research areas and disciplines of the CDS scientific community. Computational models aim to be formal languages to articulate and compare theories and hypotheses, as well as to be used as experimental tools to investigate and help us understand complex dynamics in development. Also, because they are computational, they often find applications in building machines that can behave and adapt flexibly in the real world.

However, as discussed in the dialog initiated by Olivia Guest and Nicolas Rougier in this issue, they need to address several major challenges in order to achieve scientific impact: reproducibility, replicability, but also reusability in an interdisciplinary community. Indeed, one needs to ensure that models' implementations and experimentations match their high-level specifications. It is also key to conduct alternative implementations and experimentations to distinguish which aspects of these models are key concepts, and which others are tools for experimenting these concepts. Last but not least, models should be understandable and reusable by other researchers who are not always themselves computational experts, which is facilitated when they are delivered in a way that allows non-experts to directly "play" with these models. These issues are discussed in this dialog by Konrad Hinsen, Sharon Crooke, Gaël Varoquaux, Todd Gureckis and Alexander Rich, Robert French and Caspar Addyman, and Celeste Kidd.

In a new dialog initiation, Jun Tani, who has been studying recurrent neural networks models of sensorimotor development for the last 20 years, asks which ingredients are needed to enable neural architectures with capabilities of infant-like learning and development. In particular, he observes that recent deep learning advances are still lacking infant-like capabilities for learning incrementally from very little data, and asks whether computational models of staged development could enable progress towards infant-like lifelong deep learning. He also discusses the potential role of the predictive coding principle in development. Those of you interested in reacting to this dialog initiation are welcome to submit a response by May 30th, 2017. The length of each response must be between 600 and 800 words including references (contact pierre-yves.oudeyer@inria.fr).

CDS TC Community News

After two years of chairing the IEEE CIS Cognitive and Developmental Systems technical committee, I would like to welcome Kathryn Merrick as the new TC Chair in 2017, whose introduction message is below. It has been an honour for me to serve the community in this job, helping in the transition from the AMD to the CDS TC to broaden the scope and interdisciplinary bridges of this community, which I hope will foster the dissemination of developmental systems research and ideas across the behavioral, brain and cognitive sciences.

Links
Previous open-access editions of the newsletter can be found at: http://icdl-epirob.org/cdsnl
Web site of the IEEE TC on Cognitive and Developmental Systems: http://icdl-epirob.org/cdstc
IEEE ICDL-Epirob conference: http://www.icdl-epirob.org

Message From the New CDS TC Chair

Kathryn Merrick

School of Engineering and Information Technology, University of New South Wales, Canberra, Australia

Chair of the Technical Committee on Cognitive and Developmental Systems

k.merrick@adfa.edu.au

I am honoured to be appointed to the role of Chair of the IEEE CIS Technical committee on Cognitive and Developmental Systems for 2017. I would like to take this opportunity to thank Pierre-Yves Oudeyer for his work in the last two years, his advice to me over the past few weeks, and his continuing contribution to the community, and editing this newsletter!

This is an exciting period for autonomous systems research, with rapid developments in technologies such as self-driving cars and drones, as well as more advanced robots and industrial hardware. As these technologies are emerging, however, new questions are arising around legal, ethical and safety concerns, many of which influence our ability to trust these new technologies. The design of 'trusted autonomous systems' presents a new research challenge in itself (see the recent IEEE Access article by Abbass et al., vol. 4), but one to which we, as researchers in cognitive and developmental systems, are well placed to respond.

'Trust' has been studied in many different guises. Fault-tolerance, robustness and resilience in robotics and aviation, communication and negotiation in multi-agent systems, inter-organisational trust and employee motivation are among a wide range of topics that have been studied from the perspective of trust. In fact, work across the disciplines of engineering, computer science, cognitive science and psychology all contributes to our understanding of trust.

As researchers in cognitive and developmental systems, we can contribute to the design of future 'trusted autonomous systems' from at least two novel perspectives: First, our expertise in autonomous mental development can contribute to the design of adaptive, robust and fault-tolerant systems that can apply the results of life-long learning to respond appropriately when the unexpected occurs. Secondly, our expertise in cognitive modelling, computational neuroscience and developmental psychology can inform the design of machines that can model and recognise complex cognitive states of trust and motivation in humans and respond appropriately.

In 2017 my goals for the CDS Technical Committee are (1) to maintain our existing and successful task forces (2) to identify creative individuals to augment these with new task forces in complementary areas such as cognitive systems and computational neuroscience. The potential for cross fertilisation of ideas in these areas is the first step towards the next generation of developmental systems, capable of both development in their own right and exhibiting an understanding of the developmental processes that shape their human collaborators.

I look forward to working with you all in 2017.

Links
Previous open-access editions of the newsletter can be found at: http://icdl-epirob.org/cdsnl
Web site of the IEEE TC on Cognitive and Developmental Systems: http://icdl-epirob.org/cdstc
IEEE ICDL-Epirob conference: http://www.icdl-epirob.org

Table of Contents

Editorial

Pierre-Yves Oudeyer
Computational Modelling Across Disciplines 1

Kathryn Merrick
Message From the New CDS TC Chair 2

Dialogue

Olivia Guest, Nicolas P. Rougier
What is Computational Reproducibility? 4

Konrad Hinsen
Trust But Verify 5

Sharon Crook
Resources for Reproducibility and Rigor in Computational Neuroscience 6

Gaël Varoquaux
Beyond Computational Reproducibility, Let Us Aim for Reusability 7

Todd M. Gureckis, Alexander S. Rich
Computationally Reproducible Experiments 8

Robert M. French, Caspar Addyman
Practical Replication, Not Formal Reproducibility 9

Celeste Kidd
The Importance of Conceptual Replications for Testing Computational Cognitive Theories 10

Olivia Guest, Nicolas P. Rougier
Diversity in Reproducibility 12

New Dialogue Initiation

Jun Tani
Exploring Robotic Minds by Predictive Coding Principle 13

IEEE TCDS Table of Contents

Volume 8, Issue 3, September 2016 15

Volume 8, Issue 4, December 2016 17

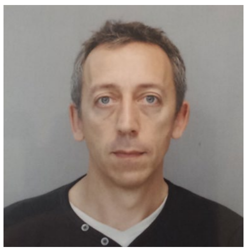
Dialogue

What is Computational Reproducibility?

Olivia Guest

Department of
Experimental Psychology
University College London,
United Kingdom

o.guest@ucl.ac.uk



Nicolas P. Rougier

INRIA Bordeaux Sud-Ouest,
Institut des Maladies
Neurodégénératives,
Université Bordeaux,
Bordeaux, France

nicolas.rougier@inria.fr

Computational modelling is the process by which phenomena found in complex systems are expressed algorithmically. The creation of such simulations is useful because it allows us to test whether our understanding is sophisticated enough to create credible working models of the phenomena we are studying. In neuroscience and cognitive science especially, computational modelling comprises more than just capturing a single phenomenon, it also implements a theory. It gives scientists a method of allowing their ideas to be executed, i.e., for emergent properties to appear when they are implemented and run (McClelland, 2009). In this context, a model is said to be *replicable* if experiments within it can be carried out successfully using the original codebase, with the implicit assumption that such a codebase is available.

However, for models to be evaluated it is mandatory to ensure they are *reproducible* (Topalidou, Leblois, Boraud, & Rougier, 2015). That is, that they can be recreated based on their specification — the details deemed important enough to be included in the accompanying article (Hinsen, 2015). Ideally, this should be possible without contacting the authors for advice, and critically, without referring to the original code (Cooper & Guest, 2014). If the specification is sufficient to successfully recreate the codebase from scratch, then the model is said to be reproducible. This adds further credence to both the model and its overarching theoretical framework. If not, and the model cannot be recreated, then even if the experiments can be carried out successfully within the original codebase, the model is not reproducible (Crook, Davison, & Plesser, 2013).

How to share computational research?

Access to the original codebase is not always straightforward. There have been few substantial changes within scholarly communication and research dissemination since 1665, when the first academic journals (*Le Journal des Sçavans* and *Philosophical Transactions of the Royal Society*) were published. Dissemination of scientific discoveries via publishers continues to consist primarily of static text and figures. However, most research is underpinned by, if not wholly comprised of, code, which is inherently dynamic.

Given code forms the backbone of modern scientific research, it is perhaps unusual that its position within this framework is not clear. For example, it is not straightforward where codebases should be placed: in a footnote (with code assured to be available upon request), in supplementary materials, or in an online repository? Even though more journals are requesting code, as well as raw data, few publisherbacked repositories exist. It is striking that an overwhelming number of journals make no provisions for and offer little guidance on hosting these files or indeed facilitating access to them.

Is it time for progress?

The open source and open science communities proposed solutions to some of the aforementioned problems without publishers' aid nor mediation. Firstly, a set of new innovative software tools (e.g., the binder project) make modelling work more accessible. Secondly, some researchers have taken matters into their own hands and created resources for best practice (e.g., version control: Blischak, Davenport, & Wilson, 2016; Eglen et al., 2016; Wilson, 2016). While others lead by example: Ogrean et al. (2016) published an article with an interactive figure; and the LIGO Open Science Center released extensive amounts of data and code (*LIGO Open Science Center: Tutorials*, 2016). In the same vein, the ReScience journal encourages the reproduction of modelling work.

Is the scientific community ready to embrace and facilitate changes with respect to: associating articles with original codebases in a transparent way and, more broadly, making sure computational theories are well-specified and coherently implemented?

Blischak, J. D., Davenport, E. R., & Wilson, G. (2016, 01). A quick introduction to version control with git and github. PLoS Comput Biol, 12(1), 1-18. doi: 10.1371/journal.pcbi.1004668
Cooper, R. P., & Guest, O. (2014). Implementations are not specifications: Specification, replication and experimentation in computational cognitive modeling. Cognitive Systems Research, 27, 42 49. doi: 10.1016/j.cogsys.2013.05.001
Crook, S. M., Davison, A. P., & Plesser, H. E. (2013). 20 years of computational neuroscience. In M. J. Bower (Ed.), (pp. 73–102). New York, NY: Springer New York. doi: 10.1007/978-1-4614-1424-7_4
Eglen, S., Marwick, B., Halchenko, Y., Hanke, M., Sufi, S., Gleeson, P., Poline, J.-B. (2016). Towards standard practices for sharing computer code and programs in neuroscience. bioRxiv. doi: 10.1101/045104
Hinsen, K. (2015). Writing software specifications. Computing in Science & Engineering, 17(3), 54–61. doi: 10.1109/MCSE.2015.64
LIGO Open Science Center: Tutorials. (2016). https:// losc.ligo.org/tutorials/. (Accessed: 2016-0602)
McClelland, J. (2009). The place of modeling in cognitive science. Topics in Cognitive Science, 1(1), 11–38. doi: 10.1111/j.1756-8765.2008.01003.x
Ogrean, G. A., van Weeren, R. J., Jones, C., Forman, W., Dawson, W. A., Golovich, N., Ebeling, H. (2016). Frontier fields clusters: Deep chandra observations of the complex merger macs j1149.6+2223. The Astrophysical Journal, 819(2), 113. doi: 10.3847/0004 -637X/819/2/113
Topalidou, M., Leblois, A., Boraud, T., & Rougier, N. P. (2015). A long journey into reproducible computational neuroscience. Frontiers in Computational Neuroscience, 9(30). doi: 10.3389/fncom.2015.00030
Wilson, G. (2016). Software carpentry: lessons learned [version 2; referees: 3 approved]. F1000Research, 3(62). doi: 10.12688/f1000research.3-62.v2

Trust But Verify

Konrad Hinsen

Center of Molecular Biophysics, CNRS Orléans. France

hinsen@cnrs-orleans.fr

Computational reproducibility is a means to an end: establishing trust in scientific findings obtained with the help of computers. In the following, I will examine the issues raised by Guest and Rougier in this wider context.

Traditionally, trust in scientific research is derived from independent verification. The more people have verified an observation, an hypothesis, or a deduction, the more it is trusted. Conversely, a claim that nobody can verify is not considered science at all. For experimental work, replication is an important aspect of verification. Models and theories are verified both by checking their internal coherence and by comparing their predictions to reality. For computational research, there are no established verification practices yet, and that's what this dialog is really about.

Machines can perform long computations much faster and more reliably than any human, but that also means that no human can directly verify the outcome. The traditional way of establishing trust thus no longer works. The earliest debate on this problem that I am aware of was started by an opinion piece by De Millo, Lipton, & Perlis (1979) on the role of formal methods in program verification. More recently, automated proofs have caused a similar debate in mathematics (Wolchover, 2013). Philosophers of science have started to examine different aspects of this "epistemic opacity" of computation (Imbert, 2016).

In computational science, trust needs to be established at three levels. The most basic one is trusting the authors of a paper that they ran the computations they claim to have run, i.e. that they did not make basic mistakes such as using a wrong input file or a defective computer. Replicability goes a long way towards providing this level of trust. In the long run, I expect replicability to disappear from our list of worries: being a purely technical issue, it can be delegated to computers. One day, replicability will be checked automatically upon submission to code/data archives such as Zenodo.

The next level of trust is about the software correctly implementing the model it claims to implement. This is in my opinion the main challenge for computational science in the coming decades (Hinsen, 2016a). Reproducibility as defined by Guest and Rougier is an important criterion: if another team of researchers can write another piece of software that reimplements a published model and yields sufficiently similar results, this generates significant trust in the original work. However, this approach is not always applicable. Complex software can take many years to implement. Complex models cannot be adequately described in journal articles (Hinsen, 2016b). Inspiration for dealing with complexity can be found in software engineering techniques such as testing or formal verification, but their application to scientific software is not straightforward.

The final level of trust is about the model representing some part of reality to some useful degree. Some people confound it with the other two levels, advocating the validation of software and computations by direct confrontation with experimental observations. Models are then just stepping stones in writing software. Like Guest and Rougier, I do not find this point of view satisfying. Science progresses by the improvement of models, not software. The mere knowledge that some software makes good predictions of experimental observations contributes little to an understanding of the underlying phenomena.

The key to progress is, in my opinion, a change of attitude towards computational results. For this I consider it important to shift the focus of this dialogue from the technicalities to the fundamentals. What really matters is not sharing code or reimplementing models, but making computational results verifiable. Perhaps the best approaches to reach this goal haven't been invented yet. We should present verifiability as both a moral obligation and a technical challenge, and then count on human ingenuity to make it happen.

As reviewers and journal editors, we should ask authors to explain how their computations were validated and how their peers can verify their results independently. Ideally, papers about computational work should include a required section entitled "verification and validation". This would serve both as a carrot and a stick: it would force authors to think about the question, but also provide a venue for showcasing good ideas and techniques that others can draw inspiration from.

De Millo, R.A., Lipton, R.J., & Perlis, A.J. (1979). Social Processes and Proofs of Theorems and Programs. Communications of the ACM, 22.5, 271–280.
Wolchover, N. (2013). In Computers We Trust? Quanta Magazine, February 22. (https://www.quantamagazine.org/20130222-in-computers-we-trust/)
Imbert, C. (2016). Computer Simulations and Computational Models in Science. In Magnani & Bertolotti (eds.) Springer Handbook of Model-Based Science. ISBN 978-3-319-30526-4
Hinsen, K. (2016a). Verifiable Research: The Missing Link between Replicability and Reproducibility. The Winnower, July 8. (doi:10.15200/winn.146857.76572)
Hinsen, K. (2016b). Scientific notations for the digital era. The Self-Journal of Science, 27 April 2016. (http://sjscience.org/article?id=527)

Resources for Reproducibility and Rigor in Computational Neuroscience

Sharon Crook

School of Mathematical and Statistical Sciences & School of Life Sciences, Arizona State Universitv. USA

sharon.crook@asu.edu

Conceptually, computational reproducibility goes beyond the ability to re-create a simulation of a computational model. The word *reproducibility* evokes thoughts of the scientific method and the importance of independent evaluation of results by other members of the community. Science requires that we determine whether conclusions have been obtained using a rigorous process, and we must know whether results are robust to small changes in conditions.

Within the computational neuroscience community, recognition of the need to improve the role of computational models in science has led to activities devoted to promoting computational reproducibility. We can categorize these approaches in terms of independence from the original implementation of the computational model: 1) ensuring replication of simulation results within one's own research group, 2) independently obtaining the same results with the original code, 3) adapting the computational model to a different simulation platform or language, and 4) reproducing the modeling results in a completely independent manner.

The open source software movement has resulted in wide-spread use of web-accessible version control systems, which has impacted the scientific computation community. Consistent use of a version control system within a research group is one of the best ways to promote the first category listed here, as long as one is careful to include data and other necessary documentation. The use of an automated tool for tracking computational projects like Sumatra (Davison 2012) offers another approach.

Independent evaluation of a computational model using the original code requires model sharing. The growth of online databases and repositories for sharing models has led to improved accessibility of models. For example, ModelDB (Migliore et al. 2003) contains over 1000 curated, published models from neuroscience, and the BioModels Database (Le Novère et al. 2006) has over 1500 literature-based models of biological processes. However, these resources offer only a tiny fraction of published models, and more efforts towards model sharing are needed. One of the best ways to guarantee model sharing would be requiring submission to a publicly-accessible online database prior to publication. Dr. Gordon Shepherd of the SenseLab behind ModelDB is enthusiastic about working with appropriate journals to develop a system for sharing models at ModelDB in coordination with publication, although broad discussion would be needed to address issues related to model curation and code review.

Several efforts focus on creating resources that provide model descriptions that are simulator-independent, where the goal is to provide all of the details needed to implement and simulate models and their components, enabling model re-use and portability. For example, PyNN provides a Python package for simulator-independent specification of neuronal network models that can be run on four different widely-used simulation platforms (Davison 2008). Unlike this procedural approach, model description languages such as NeuroML are completely descriptive (Gleeson et al. 2010), providing great flexibility for generating code that can be used to simulate the model using many different languages and simulation platforms. NeuroML descriptions of models also can be used to automatically generate tables for publication that expose the internal properties of a model such as network connectivity patterns, parameters, and units. This aids model transparency and promotes the last category of reproducibility—computational reproducibility that is completely independent from the original implementation.

All of the approaches discussed here facilitate computational reproducibility; however, increasing the scientific impact of models across neuroscience also requires better descriptions of model assumptions, constraints, and validation. These important aspects of model development and model evaluation are critical for reproducibility. For data-driven models, which data were used to constrain model development? How well do emergent properties of the model dynamics match additional experimental data? Even when model development is theory-driven, modelers should describe carefully the assumptions and the process for creating and validating the model, improving transparency and rigor.

Davison AP (2012) Automated capture of experiment context for easier reproducibility in computational research. Computing in Science and Engineering 14: 48-56.
Migliore M, Morse TM, Davison AP, Marenco L, Shepherd GM, Hines ML (2003) ModelDB: making models publicly accessible to support computational neuroscience. Neuroinformatics 1(1):135-139.
Le Novère N, Bornstein B, Broicher A, Courtot M, Donizelli M, Dharuri H, Li L, Sauro H, Schilstra M, Shapiro B, Snoep JL, Hucka M (2006) BioModels Database: A free, centralized database of curated, published, quantitative kinetic models of biochemical and cellular systems. Nucleic Acids Research 34:D689-91
Davison AP, Brüderle D, Eppler JM, Kremkow J, Muller E, Pecevski DA, Perrinet L, Yger P (2008) PyNN: a common interface for neuronal network simulators. Front. Neuroinform. 2:11.
Gleeson, P., S. Crook, R. C. Cannon, M. L. Hines, G. O. Billings, et al. (2010) NeuroML: A Language for Describing Data Driven Models of Neurons and Networks with a High Degree of Biological Detail PLoS Computational Biology 6(6): e1000815.

Beyond Computational Reproducibility, Let Us Aim for Reusability

Gaël Varoquaux

Inria and INSERM, France

gael.varoquaux@inria.fr

Science is based on the ability to falsify claims. Thus, reproduction or replication of published results is central to the progress of science. Researchers failing to reproduce a result will raise questions: Are these investigators not skilled enough? Did they misunderstand the original scientific endeavor? Or is the scientific claim unfounded? For this reason, the quality of the methods description in a research paper is crucial. Beyond papers, computers—central to science in our digital era—bring the hope of automating reproduction. Indeed, computers excel at doing the same thing several times.

However, there are many challenges to computational reproducibility. To begin with, computers enable reproducibility only if all steps of a scientific study are automated. In this sense, interactive environments—productivity-boosters for many—are detrimental unless they enable easy recording and replay of the actions performed. Similarly, as a computational-science study progresses, it is crucial to keep track of changes to the corresponding data and scripts. With a software-engineering perspective, version control is the solution. It should be in the curriculum of today's scientists. But it does not suffice. Automating a computational study is difficult. This is because it comes with a large maintenance burden: operations change rapidly, straining limited resources—processing power and storage. Saving intermediate results helps. As does devising light experiments that are easier to automate. These are crucial to the progress of science, as laboratory classes or thought experiments in physics. A software engineer would relate them to unit tests, elementary operations checked repeatedly to ensure the quality of a program.

Once a study is automated and published, ensuring reproducibility should be easy; just a matter of archiving the computer used, preferably in a thermally-regulated nuclear-proof vault. Maybe, dear reader, the scientist in you frowns at this solution. Indeed, studies should also be reproduced by new investigators. Hardware and software variations then get in the way. Portability, ie achieving identical results across platforms, is well-known by the software industry as being a difficult problem. It faces great hurdles due to incompatibilities in compilers, libraries, or operating systems. Beyond these issues, portability faces in addition numerical and statistical stability issues in scientific computing. Hiding instability problems with heavy restrictions on the environment is like rearranging deck chairs on the Titanic. While enough freezing will recover reproducibility, unstable operations cast doubt upon scientific conclusions they might lead to. Computational reproducibility is more than a software engineering challenge; it must build upon solid numerical and statistical methods.

Reproducibility is not enough. It is only a means to an end, scientific progress. Setting in stone a numerical pipeline that produces a figure is of little use to scientific thinking if it is a black box. Researchers need to understand the corresponding set of operations to relate them to modeling assumptions. New scientific discoveries will arise from varying those assumptions, or applying the methodology to new questions or new data. Future studies build upon past studies, standing on the shoulders of giants as Isaac Newton famously wrote. In this process, published results need to be modified and adapted, not only reproduced. Enabling reuse is an important goal.

To a software architect, a reusable computational experiment may sound like a library. Software libraries are not only a good analogy, but also an essential tool. The demanding process of designing a good library involves isolating elementary steps, ensuring their quality, and documenting them. It is akin to the editorial work needed to assemble a textbook from the research literature.

Science should value libraries made of code, and not only bookshelves. But they are expensive to develop, and even more so to maintain. Where to set the cursor? It is clear that in physics not every experimental setup can be stored for later reuse. Costs are less tangible with computational science; but they should not be underestimated. In addition, the race to publish creates legions of studies. As an example, Google scholar lists 28000 publications concerning compressive sensing in 2015. Arguably many are incremental and research could do with less publications. Yet the very nature of research is to explore new ideas, not all of which are to stay.

Computational research will best create scientific progress by identifying and consolidating the major results. It is a difficult but important task. These studies should be make reusable. Limited resources imply that the remainder will suffer from "code rot", harder and harder to reproduce as their software environment becomes obsolete. Libraries, curated and maintained, are the building blocks that can enable progress.

Computationally Reproducible Experiments

Todd M. Gureckis

Department of Psychology, New York University, USA

todd.gureckis@nyu.edu

Alexander S. Rich

Computation and Cognition Lab, New York University, USA

alexanders.rich@gmail.com

Guest and Rougier raise a timely issue regarding reproducible computational models. However, computational models are only as good as the empirical data they seek to explain. In psychology, and many other fields, there is currently a crisis of confidence in the quality of empirical research (Pashler and Wagenmakers, 2012). It appears that an unexpectedly large percentage of research studies do not obtain the same results when repeated by independent researchers (Open Science Collaboration, 2015).

Explanations of these failures often appeal to the difficulties in conducting experiments that are faithful to the original publication. For instance, a researcher might make an error in conducting the replication (e.g., altering the instructions). But even when a replication sticks closely to the text of a published method, there maybe unmentioned, implicit knowledge about how to conduct the study that is specific to a particular lab (Mitchell, 2014) or context (Van Bavel, 2016).

The replication debate helps us to refine the difference between replicability and reproducibility. A reproducible experimental method is one that is, in principle, possible to follow exactly. In contrast, a replicable experiment is reproducible but additionally depends on issues of sample size, statistical power, and the variability of the phenomena in question.

Many published experiments in psychology are not reproducible exactly from the methods section of the paper, limiting the feasibility of direct replication (Simons, 2014). There are simply too many variables that are not reported, from the mundane like the room temperature to the potentially crucial like experimenter demeanor. We suggest that *computationally reproducible experiments* represent one solution to this problem. Computationally reproducible experiments are experiments where every aspect of the interaction between the experimenter and the participant is controlled by a computer algorithm. While seemingly impractical (or even dystopian), this level of control is readily available in the form of web-based experiments.

Web-based experiments are psychological experiments which are conducted, most commonly over the Internet, using standard web technologies (e.g., Javascript, HTML) that run within a browser (e.g., Google Chrome). When a participant completes a web-based experiment every interaction must be programmatically scripted, from assignment to conditions and informed consent to task design and debriefing.

Web-based experiments have become a popular tool for behavioral research particularly due to the advent of crowd-sourcing systems like Amazon Mechanical Turk (Mason & Suri, 2012) which also help to standardize the method of subject recruitment and compensation. In an ideal example of this model, a researcher who conducted an experiment on Mechanical Turk could pass their experiment script to another researcher who, simply by hosting the code on a website (and recruiting a new sample from Mechanical Turk) could perform an exact replication of the design. In this case, the only difference between the original study and the replication are changes in the sample of participants on Mechanical Turk (or random noise). Other incidental variables (like air temperature) are explicitly not controlled, meaning they may not systematically differ between the original study and the attempted replication.

Today, researchers share the code for experiments informally though email. However, we have created a centralized system for sharing computationally reproducible experiments based on an open-source platform we developed called psiTurk (Gureckis et al., 2016). The psiTurk Experiment Exchange (https://psiturk.org/ee) allows researchers to download an experiment from the site and within minutes collect a new sample of data holding all elements of the method constant.

Of course, there will always be some studies that have to be performed in person, and in these cases video-taped protocols can be used to make the experimental methods more explicit and reproducible (Adolph, 2015). But to the extent that studies can be conducted online using a algorithmically scripted framework like psiTurk, the "implicit" or "unmentioned" component of experimental methods is removed aiding reproducibility.

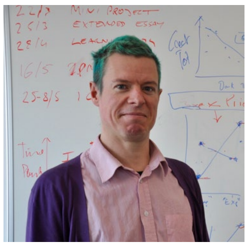
Gureckis, T.M., Martin, J., McDonnell, J., Rich, A.S., Markant, D., Coenen, A., Halpern, D., Hamrick, J.B., & Chan, P. (2016) psiTurk: An open-source framework for conducting replicable behavioral experiments online. Behavioral Research Methods, 48 (3), 829-842.
Pashler, H. & Wagenmakers, E-J. (2012) Editors' Introduction to the Special Section on Replicability in Psychological Science: A Crisis of Confidence? Perspectives in Psychological Science, 7(6), 528-530.
Open Science Collaboration. (2015). Estimating the reproducibility of psychological science. Science, 349(6251).
Van Bavel, J.J., Mende-Siedlecki, P., Brady, W.J. & Reinero, D.A. (2016). Contextual sensitivity in scientific reproducibility. Proceedings of the National Academy of Sciences,
Mitchell (2014). On the evidentiary emptiness of failed replications.
Simon, D.J. (2014). The value of direct replication. Perspective on Psychological Science, 9(1), 76-80.
Mason, W. & Suri, S. (2012). Conducting behavioral research on Amazon's Mechanical Turk. Behavioral Research Methods, 44(1), 1-23.
Adolph, K. E. (2016). Video as data: Sharing and repurposing children's behavior. APS Observer, 29, 23-25.

Practical Replication, Not Formal Reproducibility

Robert M. French

CNRS,
Dijon, France

robert.french@u-bourgogne.fr



Caspar Addyman

Psychology, Goldsmiths,
University of London,
United Kingdom

c.addyman@gold.ac.uk

"All models are wrong, but some are useful"
George Box (1979)

The distinction between replicability and reproducibility drawn by Guest and Rougier is arguably useful for initiating a discussion, but ultimately, we believe that their focus on reproducibility is misplaced. Models require more than their formal descriptions and even their code implementations. To move a field forward, models must be made as usable and testable as possible by providing ready access to a functioning implementation. If there are challenges with the replication and reuse of existing code, we should focus on pragmatic solutions for fixing on those problem, rather than attempting to introduce an abstract and formal Gold Standard.

The main reason for this was highlighted in our own 2012 Manifesto: "Most other researchers interested in your topic will not be modelers. Act accordingly." (p.334 Addyman & French, 2012). Today, we would double-down on that claim. Modelers talking to other modelers is all well and good, but if we want to take part in real, modern scientific discourse, modelers need to be able to talk to cognitive scientists, psychologists, industry and the public. We specified three goals, all of which are of crucial importance—namely, *(1) Let casual users run your simulations – easily; (2) Let motivated users run their own simulations and modify parameters; (3) Let other modelers use your code.*

This is not easy and there is no perfect solution. The problems with code becoming unusable are best addressed by improving approaches to coding. At present, our recommended way to do this is to view models as scientific software and follow best practices (Wilson et al, 2014). Keep code in a version-control system and follow established standards for documenting functions, encapsulating functionality and providing test cases.

In the future, notebook-based computing environments might provide a solution more in line with the ideal of literate programming. The concept of literate programming was introduced by computer scientist (and demigod), Donald E. Knuth, as a solution to this problem (Knuth, 1984). Code, macros and natural language are combined in a single document that acts as a human-readable explanation and machine-readable code. Numerous scientific notebook software projects (Beaker, Jupyter, RNotebooks, Wolfram Language) are working towards this goal.

These documents would complement journal articles rather than replace them. This means that papers should still be clear and comprehensive. A formal description of a model contains a lot of implicit assumptions based on authors' own knowledge. Try reading any paper in mathematics and theoretical computer science without familiarity with the field to see this. A psychologist reading most modeling papers probably experiences similar bewilderment.

Furthermore, authors improve their models. When the model exists as version-controlled software, incremental changes to a model are acknowledged and leveraged. When it is not version-controlled, the formal description of the model can be distributed across numerous publications and, as such, becomes nearly impossible to implement. What, exactly, constitutes "the formal description" of the model and who will actually check that this description, potentially spread across half a dozen papers, is comprehensive and well-specified? Will reviewers and journal editors be required to re-implement the model from scratch before the paper is accepted?

To reiterate: formal descriptions of models are necessary. But equally important to the needs of on-going research are accessible, GUI-based, user-friendly implementations of those descriptions. Only then can the models be adequately tested and explored by other researchers. The usefulness of what we suggested can be illustrated by the following example. One of the authors of the present commentary implemented TRACX (French et al., 2011), a model of word segmentation and chunking, in JavaScript and posted it to the github.com code repository (Addyman: https://github.com/YourBrain/TRACX-Web). A graduate student at CMU wanted to explore various models of segmentation and chunking and she was able to directly use Addyman's online program.

However, she was unable to get the same results we got for Aslin et al. (1998). She wrote us and we suggested a number of reasons that she might not be getting the results we reported in our paper. She wrote back a few days later and said she had looked into all of that. So, French, who had originally written the TRACX code in Matlab, went back and looked at his files and, it turned out that, while the TRACX code worked fine, he had made a mistake in the Excel file he had used to calculate the averages reported in their 2011 paper!

This correction meant that for a paper French was finishing with a colleague, which included a comparisons between TRACX and an upgraded version of the model, TRACX2

(Mareschal & French, 2017), some of the data had to be changed. Most importantly, it prevented them from making the same calculation mistake twice. So the use of Addyman's on-line JavaScript code meant that the original error was rapidly corrected—something that would have *never* happened in the old way of doing things. And, most interestingly, correcting this error actually *improved* TRACX2's fit to data! In other words, the existence of online code for TRACX was able to fix an error—to be sure a small one, but the principle is the same—that made for better science.

To paraphrase George Box, all models may, indeed, be wrong, but to better understand precisely how they are wrong and how they can be made to better approximate reality, require a readily usable framework for testing them. Remember: "Most other researchers interested in your topic will not be modelers. Act accordingly."

Addyman, C., & French, R. M. (2012). Computational Modeling in Cognitive Science: A Manifesto for Change. Topics in Cognitive Science, 4(3), 332–341. http://doi.org/10.1111/j.1756-8765.2012.01206.x
Aslin, R. N., Saffran, J. R., & Newport, E. L. (1998). Computation of Conditional Probability Statistics by 8-Month-Old Infants. Psychological Science, 9(4), 321–324. http://doi.org/10.1111/1467-9280.00063
Box, G. E. P. (1979), "Robustness in the strategy of scientific model building", in Launer, R. L.;
Wilkinson, G. N., Robustness in Statistics, Academic Press, pp. 201–236.
Knuth, Donald E. (1984). Literate Programming. The Computer Journal. British Computer Society. 27 (2): 97–111. doi:10.1093/comjnl/27.2.97
Mareschal, D., & French, R. M. (2017). TRACX2: a connectionist autoencoder using graded chunks to model infant visual statistical learning. Phil. Trans. R. Soc. B, 372(1711), 20160057. http://doi.org/10.1098/rstb.2016.0057
Wilson, G., Aruliah, D. A., Brown, C. T., Chue Hong, N. P., Davis, M., Guy, R. T., … Johnson, J. (2014). Best Practices for Scientific Computing. PLoS Biology, 12(1), e1001745. http://doi.org/10.1371/journal.pbio.1001745

The Importance of Conceptual Replications for Testing Computational Cognitive Theories

Celeste Kidd

Brain and Cognitive Sciences,
University of Rochester,
Rochester, NY
USA

celestekidd@gmail.com

While code-sharing can ensure that computational theories are coherently implemented, computational reproducibility alone is insufficient for rigorous cognitive and developmental theory testing. *Conceptual* replications must also be part of the process.

A conceptual replication is one that tests the key, underlying ideas of a theory through a novel implementation. In experimental work, a conceptual replication might attempt to test the same hypothesis with a different experimental setup, procedure, or stimulus set. The analog for computational work is *re*implementation of the same underlying ideas in order to establish that what we think is important is actually the driving factor behind relevant dynamics. In other words, it is important to formalize and test the same theories again with completely novel approaches. For example, if a theory that proposes that human behavior optimizes some function (e.g., of reward or learning) is correct, two different models that utilize two different optimization schemes should both yield the same pattern of results. In this way, different types of models can be used to provide robust, convergent evidence to support or disprove particular theories.

Conceptual replications are especially important for theories that span multiple levels of analysis, such as those in cognitive and developmental science. Many, if not most, psychological theories are couched in terms that are substantially more abstract than a particular implementation. For instance, there have been decades of productive debates about whether human behavior optimizes a given objective function, independent of the method that it uses to do so. This type of high-level analysis of a complex system (i.e., the system's function and purpose) was said to fall at the *computational level* by David Marr, who distinguished it from lower levels of analysis (algorithmic and implementational) (Marr, 1982). Cognitive and developmental scientists most often care about questions at the computational level of analysis. Are learners optimal? Is attention rational? Is memory adaptive? While all of these questions are computational, the models that we use to test these theories must necessarily make some implementational assumptions. If the theories, however, are to be believed, the outcome should not depend on the particulars of the implementation.

While one single implementation provides an existence proof that an idea can work, it is important to establish that the computational-level theory is not critically dependent on the implementational details. If one kind of optimization algorithm successfully models human behavior but another does not, then the high-level theory cannot be about optimization but rather a *specific kind* of optimization. This means that the claims we want to make as cognitive scientists are inherently connected not just to a specific implementation working well, but the space of possible implementations that could work well.

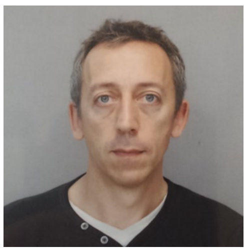
Reimplementation offers additional benefits beyond its theoretical contributions to theory testing. Solving a computational problem with a novel approach is also useful for detecting problems or bugs in a previous implementation. For example, if we thought that people optimized a given behavior, but discovered that only one kind of optimization algorithm worked well, we could discover that the real underlying process is not about optimization itself but about some incidental properties of one particular algorithm. Or, if two implementations gave two different answers, we might discover that there is a bug in one (or both) models, or that some of the assumptions underlying one (or both) implementations are incorrect. The same logic holds true of statistical analyses as well. Analogously, if a Pearson correlation is statistically reliable but a non-parametric one is not, that could suggest that the Pearson correlation is driven by an outlier. The ability of reimplementation to detect bugs becomes even more powerful when applied to complex models with many moving parts. Successful implementation requires all of the parts to work together. Further, writing code is usually easier and more pleasant than reading code, especially if you are reading specifically for bug identification. Reimplementation is a far more fun and fast way of checking computational ideas.

Marr, D. (1982), Vision: A Computational Approach, San Francisco, Freeman & Co

Diversity in Reproducibility

Olivia Guest

Department of Experimental Psychology University College London, United Kingdom
o.guest@ucl.ac.uk



Nicolas P. Rougier

INRIA Bordeaux Sud-Ouest, Institut des Maladies Neurodégénératives, Université Bordeaux, Bordeaux, France
nicolas.rougier@inria.fr

In our previous contribution, we proposed computational modelling-related definitions for replicable, i.e., experiments within a model can be recreated using its original codebase, and reproducible, i.e., a model can be recreated based on its specification. We stressed the importance of specifications and of access to codebases. Furthermore, we highlighted an issue in scholarly communication—many journals do not require nor facilitate the sharing of code. In contrast, many third-party services have filled the gaps left by traditional publishers (e.g., Binder, 2016; GitHub, 2007; Open Science Framework, 2011; ReScience, 2015). Notwithstanding, journals and peers rarely request or expect use of such services. We ended by asking: are we ready to associate codebases with articles and are we prepared to ensure computational theories are well-specified and coherently implemented?

Scope of Evaluation

Dialogue contributions include proposals for: intermediate levels between replicability and reproducibility (Crook, Hinsen); going beyond reproducibility (Kidd); encompassing computational science at large (Gureckis & Rich, Varoquaux); and addressing communities as a function of expertise (French & Addyman). On the one hand, some replies discuss evaluation more broadly, empirical data collection, and software engineering. On the other hand, some delve into the details of evaluating modelling accounts. We will discuss the former first.

In Varoquaux's contribution, reproducibility includes replicability and code rot (e.g., in fMRI: Eklund, Nichols, & Knutsson, 2016). However, the titular computational reproducibility is orthogonal to maintaining a re-usable codebase. Software and hardware inevitably go out of fashion meaning codebases expire. Nevertheless, the overarching theory encapsulated by modelling software could withstand the effects of entropy if specified coherently, e.g., early artificial neural network codebases are not required to understand nor reproduce these models. Indubitably, there is a balance to be struck between reimplementation and re-use.

In contrast, Gureckis and Rich extend their scope to the empirical replication crisis in psychology. They mention that implicit knowledge often goes unpublished and thus only fully automated on-line experiments are computationally reproducible psychology.

Epistemically, empirical and software replication and reproduction are distinct from their modelling-related counterparts — they are six related endeavours. The difference between software for science (e.g., a statistical test) and science that is software (e.g., a cognitive model) is an important one to underline. In the former case the code is a tool, in the latter it constitutes an experiment. Notwithstanding, all such evaluations have scientific merit.

Levels of Evaluation

We mentioned two of the levels in which modelling work is evaluated. Unanimity is reached on replication as a minimum check, however some dialogue contributions go further. To wit, Hinsen separates this endeavor into three steps. Specifically we must check that a model is: bug-free; reproducible as presented; congruent with empirical data. These roughly map onto the levels of talking about modelling work more generally, as Kidd notes (Marr, 1982).

Implementation Level

With respect to the implementation level, as Crook explains, re-running code both within a lab and by others allows for checking for bugs and, importantly, if assumed-to-be-irrelevant variables, e.g., the random seed, are not driving the results. This also ensures documentation is appropriate. Success at this level indicates a model is replicable.

Model Level

To evaluate the quality of the specification, we may rewrite, i.e., reproduce, the model from scratch. This provides evidence for or against depending on the reimplementation's success. As Kidd mentions, and as we discovered (Cooper & Guest, 2014), this process allows us to: discern when implementation details must be elevated to the theory level and vice versa; evaluate the specification; and uncover bugs.

Theory Level

Many methods exist for testing theories. One such method involves computationally implementing a theory—another is to test predictions by gathering empirical data. As Crook points out, such data is also used to evaluate models and should be associated with the original article and codebase. In such cases, empirical data requires re-collecting. This is because if the phenomenon to-be-modelled, Hinsen warns, does not occur as described by the overarching theoretical account, then both theory and model are brought into question. "A* is a model of [...] A to the extent that [we] can use A* to answer questions [...] about A." (Minsky, 1965, p. 426)

Conclusions

Even though definitions for terms across the replies do not fully converge[1], all contributors agree that change is needed and imminent. A notable divergence of opinion can be found in the reply by French and Addyman, who believe specifications are less vital than we do. Importantly, we agree on some fundamentals: sharing codebases; linking articles with codebases; and reproducing models (e.g., ReScience, 2015).

In response to our question: Hinsen proposes modellers include a specific article section on evaluation; while Crook lists community-driven initiatives for sharing codebases and specifications. Crook hopes, as we do, for topdown publisher-enforced sharing of resources in partially-centralised repositories. However, this does not preclude, and may in fact require, grassroots demands. If the scientific community rightfully yearns for change, we are required to act to make this happen.

Binder. (2016). mybinder.org. Retrieved 2017-01-02, from http://mybinder.org/

Cooper, R. P., & Guest, O. (2014). Implementations are not specifications: Specification, replication and experimentation in computational cognitive modeling. Cognitive Systems Research, 27, 42-49. doi: 10.1016/ j.cogsys.2013.05.001

Eklund, A., Nichols, T. E., & Knutsson, H. (2016). Cluster failure: Why fMRI inferences for spatial extent have inflated false-positive rates. Proceedings of the National Academy of Sciences, 113(28), 7900-7905. Retrieved from http://www.pnas.org/ content/113/28/7900.abstract doi: 10.1073/ pnas.1602413113

GitHub. (2007). GitHub. Retrieved 2017-01-02, from http:// github.com/

Marr, D. (1982). Vision: A Computational Investigation into the Human Representation and Processing of Visual Information. Freeman.

Mesnard, O., & Barba, L. A. (2016, May). Reproducible and replicable CFD: it's harder than you think. ArXiv e-prints.

Minsky, M. (1965). Matter, mind and models. In International federation of information processing congress.

Open Science Framework. (2011). OSF. Retrieved 2017-01-02, from http://osf.io/

Patil, P., Peng, R. D., & Leek, J. (2016). A statistical definition for reproducibility and replicability. bioRxiv. Retrieved from http://biorxiv.org/ content/early/2016/07/29/066803 doi: 10 .1101/066803

ReScience. (2015). The ReScience Journal. Retrieved 2017-01-02, from http://rescience.github.io/

1. We do not wish to prescriptively enforce our terms and definitions—and we are open to suggestions, especially based on the use of such terms by computationally-based disciplines (e.g., Mesnard & Barba, 2016; Patil, Peng, & Leek, 2016).

New Dialogue Initiation

Exploring Robotic Minds by Predictive Coding Principle

Jun Tani

Cognitive Neuro-Robotics Lab.
Korean Advanced Institute of Science and Technology
Okinawa Institute of Science and Technology

tani1216jp@gmail.com

This dialogue discusses the topic of predictive coding in developmental robotics, highlighted from my newly published book (Tani, 2016).

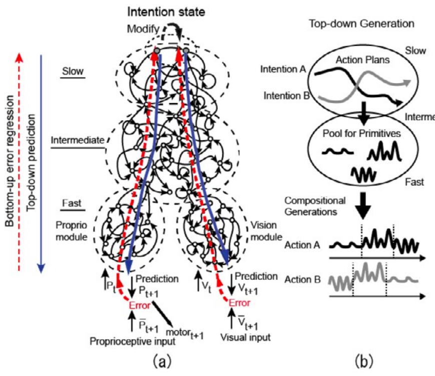
The book proposes that the mind is comprised of emergent phenomena, which appear via intricate and often conflictive interactions between the top-down intention for acting on the external world and the bottom-up recognition of the resultant perceptual reality. It is presumed that the skills for generating complex actions as well as knowledge and concepts for representing the world naturally develop through entangled interactions between these two processes. This hypothesis has been evaluated by conducting nearly two decades of neurorobotics experiments using various recurrent neural network models based on the principle of predictive coding.

Is predictive coding a paradigm shift in developmental or learning robots?

The idea of sensory-motor mapping has dominated for a long period in the study of behavior-based robotics. However, robots based on just sensory-motor mapping schemes cannot achieve human-level thinking and acting because they should be much more proactive toward the future as well as reflective of the past. In predictive coding, the intention for an action is generated with prediction of the action's consequence. Likewise, the recognition of the actual consequence in the open environment reflects on the current intention by means of the error regression with the prediction error.

Is implementation by RNN using error backpropagation through time (BPTT) essential?

A notable advantage of RNN models is that they are differentiable. If the whole network is built on a set of modular RNNs—for instance one RNN for each sensory modality of a robot, one to learn multi-modality associations, and one for executive control—the whole also becomes differentiable. In this situation, a prediction error appearing at a particular spatio-temporal point in the perceptual flow can be distributed into the whole network retrospectively using error backpropagation through time. If the whole network activity is imposed with particular macroscopic constraints such as multiple timescales (for instance, different local subnetworks functioning at different timescales) or multiple spatial scales (for instance, different local connectivity distribution among subnetworks), some meaningful structures such as spatio-temporal hierarchy can self-organize as the result of end to end learning on this differentiable network. This type of development by means of the downward causation cannot be expected if the whole system is composed of patchy assemblies of different computational schemes.



(a) Predictive coding implemented by multiple timescales RNN and (b) self-organization of functional hierarchy for action generation.

Is staged development essential?

It is fair to say that the recent success of deep learning is owed to a few researchers who have strongly believed for decades that the error backpropagation applied to differentiable networks is the most effective machine learning scheme. Now, we witness that convolutional neural networks, long-term and short-term memory as well as neural Turing machine built on this idea show significant learning performance by using millions of training data available on the internet.

However, this deep learning approach supported by usage of huge amount of data cannot be applied directly to developmental robots because they are constrained by the so-called poverty of stimulus, just like human infants. For both robots and infants the amount of experience in the real world is quite limited. Still at least for infants, skills and knowledge can be developed adequately with generalization even under such conditions. As pointed out by many others, it is expected that learning in one developmental stage can provide a "prior" for the one in the next stage thus drastically reducing freedom of learning. By this means, generalization with less amount of tutoring experience becomes possible. Based on this conception, developmental stage would proceed from physical embodiment levels to more symbolic ones.

Tutoring should require a lengthy period wherein physical interactions between robots and tutors involve “scaffolding”: guiding support provided by tutors that enables the bootstrapping of cognitive and social skills required in the next stage.

Can robots attain free will and consciousness?

For robots built on predictive coding, action and thoughts are generated as emergent phenomena when dense interactions between the top-down and the bottom-up process are developed in circular causality. It has been shown that chaos developed in the higher cognitive levels drives the spontaneous generation of the next intentional action, which will then be modified by means of minimizing the resultant conflictive error with the outer world (Tani, 2016). It is speculated that the spontaneity in generating the next intention by chaos might account for the unconscious generation of free will reported by Benjamin Libet whereas effortful process of minimizing the conflictive error does the same for the postdictive conscious awareness of it. When robotic minds are built on such emergent phenomena, those robots could have subjective experiences, just like us.

Tani, J. (2016). Exploring Robotic Minds: Actions, Symbols, and Consciousness as Self-Organizing Dynamic Phenomena. Oxford University Press.

IEEE TCDS Table of Contents

Volume 8, Issue 3, September 2016

Emergence of Altruistic Behavior Through the Minimization of Prediction Error
Jimmy Baraglia, Yukie Nagai, Minoru Asada

The emergence of altruistic behavior in infants fosters their social development and supports their involvement in our society. Altruistic tendencies, intended to benefit others with no apparent rewards, are also very useful for social robots that are designed to be used in our households. Yet, to make robots capable of learning how to help others as infants do, it is important to understand the mechanisms and motives responsible for the development of altruistic behavior. Further, understanding the mechanisms behind the early development of pro-social behavior would be a great contribution to the field of developmental psychology. To these ends, we hypothesize that infants from 14 months of age help others to minimize the differences between predicted actions and observations, that is, to minimize prediction errors. To evaluate our hypothesis, we created a computational model based on psychological studies and implemented it in real and simulated robots. Our system first acquires its own sensory-motor representation by interacting with its environment. Then, using its experience, the system recognizes and predicts others' actions and uses this prediction to estimate a prediction error. Our experiments demonstrated that our robots could spontaneously generate helping behaviors by being motivated by the minimization of prediction errors.

Interplay of Rhythmic and Discrete Manipulation Movements During Development: A Policy-Search Reinforcement-Learning Robot Model
Valentina Cristina Meola, Daniele Caligiore, Valerio Sperati, Loredana Zollo, Anna Lisa Ciancio, Fabrizio Taffoni, Eugenio Guglielmelli, Gianluca Baldassarre

The flexibility of human motor behavior strongly relies on rhythmic and discrete movements. Developmental psychology has shown how these movements closely interplay during development, but the dynamics of that are largely unknown and we currently lack computational models suitable to investigate such interaction. This work initially presents an analysis of the problem from a computational and empirical perspective and then proposes a novel computational model to start to investigate it. The model is based on a movement primitive capable of producing both rhythmic and end-point discrete movements, and on a policy search reinforcement learning algorithm capable of mimicking trial-and-error learning processes underlying development and efficient enough to work on real robots. The model is tested with hand manipulation tasks ("touching," "tapping," and "rotating" an object). The results show how the system progressively shapes the initial rhythmic exploration into refined rhythmic or discrete movements depending on the task demand. The tests on the real robot also show how the system exploits the specific hand-object physical properties, some possibly shared with developing infants, to find effective solutions to the tasks. The results show that the model represents a useful tool to investigate the interplay of rhythmic and discrete movements during development.

Nonparametric Bayesian Double Articulation Analyzer for Direct Language Acquisition From Continuous Speech Signals
Tadahiro Taniguchi, Shogo Nagasaka, Ryo Nakashima

Human infants can discover words directly from unsegmented speech signals without any explicitly labeled data. Current machine learning methods cannot efficiently estimate language model (LM) and acoustic model (AM) and discover words directly from continuous human speech signals in an unsupervised manner. To solve this problem, we propose an integrative generative model that combines an LM and an AM into a single generative model called the hierarchical Dirichlet process hidden LM (HDP-HLM). The HDP-HLM is obtained by extending the hierarchical Dirichlet process hidden semi-Markov model (HDP-HSMM) proposed by Johnson et al. An inference procedure for

the HDP-HLM is derived using the blocked Gibbs sampler originally proposed for the HDP-HSMM. This procedure enables the simultaneous and direct inference of LM and AM from continuous speech signals. Based on the HDP-HLM and its inference procedure, we develop a novel machine learning method called nonparametric Bayesian double articulation analyzer (NPB-DAA) that can directly acquire LM and AM from observed continuous speech signals. By assuming HDP-HLM as a generative model of observed time series data, and by inferring latent variables of the model, the method can analyze latent double articulation structure, i.e., hierarchically organized latent words and phonemes, of the data in an unsupervised manner. We also carried out two evaluation experiments using synthetic data and actual human continuous speech signals representing Japanese vowel sequences. In the word acquisition and phoneme categorization tasks, the NPB-DAA outperformed a conventional double articulation analyzer and baseline automatic speech recognition system whose AM was trained in a supervised manner. The main contributions of this paper are as follows: 1) we develop a probabilistic generative model that integrates LM and AM, i.e., HDP-HLM; 2) we derive an inference method for this, and propose the NPB-DAA; and 3) we show that the NPB-DAA can discover words directly from continuous human speech signals in an unsupervised manner.

Evolutionary Fuzzy Integral-Based Gaze Control With Preference of Human Gaze

Bum-Soo Yoo, Jong-Hwan Kim

Research on developing human-like gaze control has been carried out to enhance human-robot interaction. From the viewpoint of a large consistency of human gaze, conventional research had focused on predicting where humans usually pay attention to. However, gaze control is a cognitive process that can even produce different scanpaths from the same visual information. In this paper, an evolutionary fuzzy integral-based gaze control algorithm with preference is proposed. It produces various scanpaths according to the preference of human gaze. The proposed gaze control algorithm evaluates each pixel point with fuzzy measures and fuzzy integral, and produces a scanpath through repeated selections considering memory and bio-inspired processes. The produced scanpath is transformed into a fixation map and compared with a scanpath obtained from a human subject by the earth mover's distance. Based on the comparison, quantum-inspired evolutionary algorithm gradually develops preference of human gaze to produce a scanpath similar to the human scanpath. The effectiveness of the proposed algorithm is demonstrated by comparing a human scanpath with a scanpath produced from the algorithm using the developed preference. The applicability of the proposed algorithm is also demonstrated by applying the developed preference to gaze control for learning from demonstration.

Lifelong Augmentation of Multimodal Streaming Autobiographical Memories

Maxime Petit, Tobias Fischer, Yiannis Demiris

Robot systems that interact with humans over extended periods of time will benefit from storing and recalling large amounts of accumulated sensorimotor and interaction data. We provide a principled framework for the cumulative organization of streaming autobiographical data so that data can be continuously processed and augmented as the processing and reasoning abilities of the agent develop and further interactions with humans take place. As an example, we show how a kinematic structure learning algorithm reasons a-posteriori about the skeleton of a human hand. A partner can be asked to provide feedback about the augmented memories, which can in turn be supplied to the reasoning processes in order to adapt their parameters. We employ active, multimodal remembering, so the robot as well as humans can gain insights of both the original and augmented memories. Our framework is capable of storing discrete and continuous data in real-time. The data can cover multiple modalities and several layers of abstraction (e.g., from raw sound signals over sentences to extracted meanings). We show a typical interaction with a human partner using an iCub humanoid robot. The framework is implemented in a platform-independent manner. In particular, we validate its multi platform capabilities using the iCub, Baxter and NAO robots. We also provide an interface to cloud based services, which allow automatic annotation of episodes. Our framework is geared towards the developmental robotics community, as it: 1) provides a variety of interfaces for other modules; 2) unifies previous works on autobiographical memory; and 3) is licensed as open source software.

GRAIL: A Goal-Discovering Robotic Architecture for Intrinsically-Motivated Learning

Vieri Giuliano Santucci, Gianluca Baldassarre, Marco Mirolli

In this paper, we present goal-discovering robotic architecture for intrisically-motivated learning (GRAIL), a four-level architecture that is able to autonomously: 1) discover changes in the environment; 2) form representations of the goals corresponding to those changes; 3) select the goal to pursue on the basis of intrinsic motivations (IMs); 4) select suitable computational resources to achieve the selected goal; 5) monitor the achievement of the selected goal; and 6) self-generate a learning signal when the selected goal is successfully achieved. Building on previous research, GRAIL exploits the power of goals and competence-based IMs to autonomously explore the world and learn different skills that allow the robot to modify the environment. To highlight the features of GRAIL, we implement it in a simulated iCub robot and test the system in four different experimental scenarios where the agent has to perform reaching tasks within a 3-D environment.

Volume 8, Issue 4, December 2016

Affordance Research in Developmental Robotics: A Survey

Huaqing Min, Chang'an Yi, Ronghua Luo, Jinhui Zhu, Sheng Bi

Affordances capture the relationships between a robot and the environment in terms of the actions that the robot is able to perform. The notable characteristic of affordance-based perception is that an object is perceived by what it affords (e.g., graspable and rollable), instead of identities (e.g., name, color, and shape). Affordances play an important role in basic robot capabilities such as recognition, planning, and prediction. The key challenges in affordance research are: (1) how to automatically discover the distinctive features that specify an affordance in an online and incremental manner and (2) how to generalize these features to novel environments. This survey provides an entry point for interested researchers, including: (1) a general overview; (2) classification and critical analysis of existing work; (3) discussion of how affordances are useful in developmental robotics; (4) some open questions about how to use the affordance concept; and (5) a few promising research directions.

Selective Attention by Perceptual Filtering in a Robot Control Architecture

François Ferland, François Michaud

Modern autonomous robots must integrate multiple perceptual and behavioral modalities to be useful in our daily lives. Such integration is constrained by the limited onboard computing capacity of robotic platforms. To alleviate this issue, perceptual filtering, a selective attention mechanism, can be used to efficiently manage computing resources based on what the robot has to accomplish. This paper describes our implementation of perceptual filtering in a robot control architecture, implemented using robot operating system (ROS), and how it can dynamically optimize the use of the computing resources available on the robot. Our perceptual filtering mechanism is demonstrated and validated using a mobile humanoid platform integrating autonomous and teleoperated navigation, QR code recognition, face recognition, and sound localization capabilities.

Training Agents With Interactive Reinforcement Learning and Contextual Affordances

Francisco Cruz, Sven Magg, Cornelius Weber, Stefan Wermter

In the future, robots will be used more extensively as assistants in home scenarios and must be able to acquire expertise from trainers by learning through crossmodal interaction. One promising approach is interactive reinforcement learning (IRL) where an external trainer advises an apprentice on actions to speed up the learning process. In this paper we present an IRL approach for the domestic task of cleaning a table and compare three different learning methods using simulated robots: 1) reinforcement learning (RL); 2) RL with contextual affordances to avoid failed states; and 3) the previously trained robot serving as a trainer to a second apprentice robot. We then demonstrate that the use of IRL leads to different performance with various levels of interaction and consistency of feedback. Our results show that the simulated robot completes the task with RL, although working slowly and with a low rate of success. With RL and contextual affordances fewer actions are needed and can reach higher rates of success. For good performance with IRL it is essential to consider the level of consistency of feedback since inconsistencies can cause

considerable delay in the learning process. In general, we demonstrate that interactive feedback provides an advantage for the robot in most of the learning cases.

Spatial Concept Acquisition for a Mobile Robot That Integrates Self-Localization and Unsupervised Word Discovery From Spoken Sentences

Akira Taniguchi, Tadahiro Taniguchi, Tetsunari Inamura

In this paper, we propose a novel unsupervised learning method for the lexical acquisition of words related to places visited by robots, from human continuous speech signals. We address the problem of learning novel words by a robot that has no prior knowledge of these words except for a primitive acoustic model. Furthermore, we propose a method that allows a robot to effectively use the learned words and their meanings for self-localization tasks. The proposed method is nonparametric Bayesian spatial concept acquisition method (SpCoA) that integrates the generative model for self-localization and the unsupervised word segmentation in uttered sentences via latent variables related to the spatial concept. We implemented the proposed method SpCoA on SIGVerse, which is a simulation environment, and TurtleBot2, which is a mobile robot in a real environment. Further, we conducted experiments for evaluating the performance of SpCoA. The experimental results showed that SpCoA enabled the robot to acquire the names of places from speech sentences. They also revealed that the robot could effectively utilize the acquired spatial concepts and reduce the uncertainty in self-localization.

Decoding EEG in Cognitive Tasks With Time-Frequency and Connectivity Masks

Junhua Li, Yijun Wang, Liqing Zhang, Andrzej Cichocki, Tzyy-Ping Jung

Electroencephalogram (EEG) is a measurable window looking into brain dynamics. Brain activities may exhibit different representations while executing different cognitive tasks, which can be recognized by decoding EEG. This is crucial for constructing a brain-computer interface (BCI), which directly bridges between the human brain and external experiments or devices for communication or function restoration. This paper proposed a mask-based approach integrating time-frequency mask (TFM) and connectivity mask (CM) to improve BCI performance. The TFM method does not require the discriminative time-frequency points to be centralized together as the specific-frequency-specific-time (SFST) method does. It can also achieve good performance when discriminative features are scattered. Moreover, this paper also developed a CM method in the spatial domain to extract interchannel connectivity features. The performance of these methods was quantitatively evaluated on three datasets involving different cognitive tasks: 1) a pointing movement dataset; 2) a self-paced finger-tapping dataset in BCI competition II; and 3) a slow cortical potential dataset in BCI competition II. Empirical results of this paper showed that the TFM method outperformed the SFST method on all three datasets and achieved comparable performance to the winning methods in the two BCI competition datasets. The performance was further improved by combining TFM and CM, exceeding that of the winning methods in the BCI competition datasets.

Editor	**Pierre-Yves Oudeyer,** Inria and Ensta ParisTech, France, pierre-yves.oudeyer@inria.fr
Editorial assistant	**Fabien Benureau,** Inria and CNRS, France
	ISSN 1550-1914