2017-01-19

Progress in modeling through distributed collaboration: Concepts, tools, and category-learning examples

Wills, AJ

http://hdl.handle.net/10026.1/5443

10.1016/bs.plm.2016.11.007
Psychology of Learning and Motivation
Elsevier

Progress in modeling through distributed collaboration: Concepts, tools, and category-learning examples

Andy J. Wills[a], Garret O'Connell[a], C.E.R. Edmunds[a], Angus B. Inkster[a]

[a]*School of Psychology, Plymouth University, UK*

Abstract

Formal modeling in psychology is failing to live up to its potential due to a lack of effective collaboration. As a first step towards solving this problem, we have produced a set of freely-available tools for distributed collaboration. This article describes those tools, and the conceptual framework behind them. We also provide concrete examples of how these tools can be used. The approach we propose enhances, rather than supplants, more traditional forms of publication. All the resources for this project are freely available from the `catlearn` website `http://catlearn.r-forge.r-project.org/`

Contents

Email address: `andy@willslab.co.uk` (Andy J. Wills)

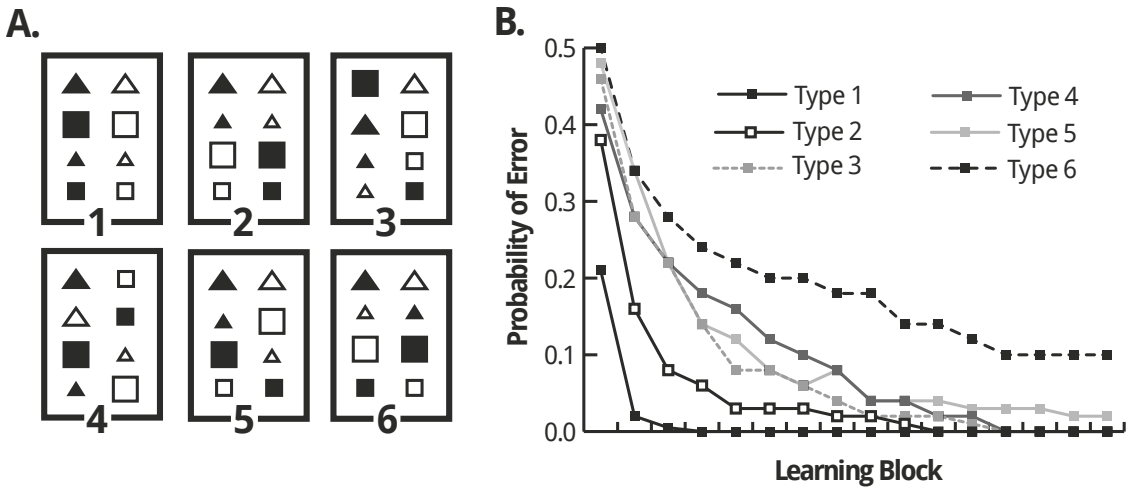

Figure 1: A. Six different ways to classify eight stimuli into two different groups; the stimuli take one of two values on each of three dimensions (color, size, shape). B. Mean errors in learning these category structures from feedback, as a function of amount of feedback (learning block). Reproduced from "On the Adequacy of Current Empirical Evaluations of Formal Models of Categorization" by A.J. Wills & E.M. Pothos, 2012, Psychological Bulletin, 138, 102–125. Copyright 2012 by American Psychological Association.

1. Introduction

A formal psychological model is one that unambiguously specifies transformations from independent variables to dependent variables (Wills & Pothos, 2012). For example, in category learning, category structure might be one independent variable and classification accuracy one dependent variable (see Figure 1). Wills & Pothos (2012) argued that: (1) formal models are important because they allow unambiguous comparison of the relative adequacy of theories, but that (2) little progress had been made because the comparisons had been too narrow. For example, Smith & Minda (2000) compared the performance of two models on variants of just one experiment.

The route to faster progress in formal modeling, according to Wills & Pothos (2012), is to compare models across a much broader set of phenomena. Over the last few years, we've been trying to put that idea into action. We've concluded that the goal is not achievable by any individual or small group within a reasonable amount of time (e.g. one career). Collaboration would solve this problem, but a suitable framework for efficient collaboration has been lacking until now. In the sections that follow, we introduce a conceptual framework, and a set of practical, freely-available tools, to support efficient distributed collaboration through technological means. We illustrate our ideas with examples from the field we know best, category learning, but the ideas are general purpose, and can be applied to any situation where formal models are evaluated against known phenomena. In providing this framework and these tools, we have lowered the "cost of entry" for collaboration. Our hope is that, by so doing, we will encourage others to adopt distributed collaboration as a central research methodology in the formal modeling of psychological processes.

The form of distributed collaboration we envisage would be a cultural shift for psychology. Such a shift would be desirable, as evidenced by how well distributed collaboration works in other fields (e.g. software development). However, the shift will only happen if distributed collaboration is compatible with the traditional metrics by which researchers gain employment, promotion, and esteem—an issue we return to throughout the article. Our central point, though, is that the formal modeling of mental processes is simply too large a project for any one person or lab. The choice is between collaboration or failure.

2. Concepts

We propose, and have initiated, an open archive of formal psychological models, plus independently-replicated data sets against which to test them, and archival records of those simulations. The structure

	ALCOVE	proto-ALCOVE	COVIS	⋮	⋮
S, H & J (1961)	1	0	1	·	·
M & S (1978)	1	0	NT	·	·
...	·	·	·	·	·
...	·	·	·	·	·

Figure 2: Illustration of a small part of the "Grid". Columns represent formal models, rows represent empirical phenomena. The cell contents indicate: ordinal success of model (1), ordinal failure of model (0), model not tested (NT). The examples in this figure are drawn from category learning, but the concept of the Grid applies across the formal modeling of psychological processes. S, H & J (1961) = Shepard et al. (1961), M & S (1978) = Medin & Schaffer (1978), ALCOVE (Attention Learning COVEring map, Kruschke, 1992), proto-ALCOVE (e.g. Johansen & Palmeri, 2002), COVIS (COmpetition between Verbal and Implicit Systems, Ashby et al., 1998). Image author: Andy J. Wills. CC BY 4.0.

and content of this open archive is based around five central concepts, which we discuss below: the "Grid" (Section 2.1), Canonical Independently Replicated Phenomena (CIRP, pronounced "syrup"[1], Section 2.2), open CIRP registration (Section 2.3), open-source model implementations (Section 2.4), and open simulation publication (Section 2.5), analogous to open data publication (e.g. Morey et al., 2016).

2.1. The Grid

The Grid is a way of representing the goal of broad comparisons of model adequacy, and progress toward that goal. It is also an archival record of the success or failure of simulations of empirical phenomena. Figure 2 illustrates one small part of the Grid.

Each column of the Grid represents a distinct formal model. The number of these columns, at least within the published literature, is knowable but large. The central problem, as we will cover in more detail in Section 2.4, is that software to implement these models is either not publicly available, or available but incompatible with implementations of other models.

Each row of the Grid represents an empirical phenomenon within the explanatory scope of at least one of the models. Determining what these rows should be is a more difficult task than it might first appear, as we will discuss in Section 2.2.

The matrix of cells that is defined by the rows (phenomena) and columns (models) of the Grid poses a challenge in terms of sheer scale. Even with ten models and ten empirical phenomena, the Grid makes explicit the need to run up to 10 models $\times$ 10 phenomena = 100 different simulations. Each journal article reporting work on a formal model has, perhaps, on average, 2–4 such simulations. Even if the 25 or more journal articles that would, at this rate, be needed to fill this grid[2] actually existed (and they most likely do not as yet), discovering what has already been done is laborious, as no search engine indexes journals in this way. For example, if one wanted to search for simulations of Medin & Schaffer (1978) with the ALCOVE model (Kruschke, 1992), one might use Web of Science to retrieve all papers that cite both those articles. As of 2016-07-14, this is 316 articles. So, at a minimum, one would have to read 316 abstracts to determine what simulations had been performed. In practice, the abstracts probably would not be conclusive in some cases, and one would have to retrieve the full text as well—and all this for one cell of the Grid.

[1] As the singular of phenomena is phenomenon, the acronym CIRP is correct for both singular and plural forms, cf. sheep.
[2] 100 simulations / 4 simulations per paper = 25 papers.

In summary, the Grid highlights the extent of the problem facing the formal modeling of psychological processes. The columns of the Grid (formal models) are not publicly available (or have other problems, see Section 2.4). Determining the rows of the grid (the phenomena) is not straightforward, as we discuss in Section 2.2. The cells of the Grid (simulations of phenomena) are a hurdle in terms of both numerosity and lack of an appropriate record system, as we have just discussed.

2.2. Canonical Independently Replicated Phenomena (CIRP)

We propose that each row of the Grid should be a Canonical Independently Replicated Phenomenon (CIRP). In this sub-section, we first justify and discuss the criterion of independent replication, and then explain what, by our definition, makes an independently-replicated phenomenon canonical. We also consider some consequences of adopting the CIRP approach.

We propose that each row of the Grid should be an independently-replicated phenomenon because modeling non-real results wastes time. The waste of effort increases as the number of models increases—10 models equals up to 10 wasted simulations for each non-real result. Non-real results also potentially lead to an inaccurate assessment of relative adequacy—a model should not be penalized for failing to accommodate a non-real result, yet inclusion of non-real results means this may happen. Thus, to guard against the wasted effort and potentially misleading conclusions that come from simulating non-real results, we propose a minimum standard that relative adequacy assessments of formal models be restricted to independently replicated phenomena.

Independent replication is beginning to become accepted as a data quality standard in psychology (e.g. Ledgerwood, 2014; Pashler & Wagenmakers, 2012). However, a criterion of independent replication (no shared authors) is not always applied when choosing phenomena against which to evaluate formal models. For example, when the Generalized Context Model (GCM, Nosofsky, 1984) was introduced, it was evaluated against a single experiment which, at that time, had not been replicated (Shepard et al., 1961). More recently, in a review of experiments supportive of the COVIS model (Ashby et al., 1998), none of the attempted independent replications were cited (Ashby & Maddox, 2011). The intention here is not to single out these authors for criticism, but to illustrate a general issue. We are aware of relatively few cases in formal cognitive modeling where independent replication is applied as a data-quality criterion and, even in those cases, it is only applied to a subset of the phenomena examined (e.g. Love et al., 2004).

There are a couple of objections that might be raised against independent replication as a minimum data-quality standard for model evaluation. The first is that attempts to replicate are rare in psychology (Makel et al., 2012) and, where replications are attempted, they are often unsuccessful (Open Science Collaboration, 2015).[3] Thus, one fear is that applying a criterion of independent replication would leave no phenomena to examine within a particular domain. The rational response to such a fear is to determine whether one's fear is justified and, if it is, suspend formal modeling of that domain until such time as demonstrably real phenomena become available. Some domains of psychology contain independently replicated phenomena, as illustrated later in this section, and also in Section 4.3.

A second objection to using independent replication as a minimum data-quality standard is that it's hard to define what counts as a replication. Pashler & Harris (2012) make a distinction between direct and conceptual replications, and are critical of the latter. We agree with their central point but note that, in practice, there are some problems in distinguishing between a direct replication and a conceptual one. For example, if a direct replication is a study that is the same as the original study in every respect, then no replications are direct (for example, they test different participants). Defining a direct replication as one identical in every respect except that they sample different members of the same tightly-defined population (e.g. US undergraduates) is also likely to be prohibitively restrictive.

The concept of a CIRP is a pragmatic response to the perhaps inevitable uncertainty about what counts as a replication. The search for CIRP should, we argue, favor independent replications that are as close to being direct as the published literature will allow. The determination of CIRP is an area in which the concept of ordinal success (Wills & Pothos, 2012) is useful. No two studies in psychology have ever produced

[3]For an more extended discussion of this latter claim, see also Anderson et al. (2016) and Gilbert et al. (2016).

literally the same results at a quantitative level. In the CIRP approach, a replication is considered to be successful if the key results are statistically significant, and in the same direction as the original study.

Another important aspect of defining a CIRP is to accept that some changes to stimuli or procedure may lawfully affect otherwise robust results. For example, Medin & Schaffer (1978) is one of the most highly cited publications in the history of categorization research.[4] The Experiment 2 data set of that paper has been independently replicated a number of times (Blair & Homa, 2003; Minda & Smith, 2002; Nosofsky et al., 1992, 1994; Rehder & Hoffman, 2005). Although there is debate about the generality of the ordinal result across different stimulus sets and procedures (e.g. Nosofsky, 2000; Smith & Minda, 2000), there is consensus that the ordinal result is robust within the experimental conditions employed by Medin and Schaffer (e.g. Minda & Smith, 2002). Experiment 2 of Medin & Schaffer (1978) is thus an independently replicated phenomenon in category learning. It may have boundary conditions (it's likely most psychological phenomena do) but, when you run the experiment the way the original authors did, you can expect to get the results they did. This is a solid basis for the relative adequacy assessment of formal models. Note that, in the CIRP approach, the emphasis is on real effects, rather than effects that are both real and important. In a relative-adequacy assessment (Wills & Pothos, 2012), an effect is important if at least one current or future model cannot accommodate it. Therefore, it is seldom possible to know in advance whether an effect is important in this sense.

The CIRP approach has a couple of necessary consequences for model evaluation. First, it places relative adequacy assessments of formal models slightly behind the empirical front line; "off the razor's edge", as Medin (2011) put it. The necessary lag this introduces is probably a good thing—this kind of broad model comparison is time consuming and one needs to ensure it is signal, rather than noise, that is being fitted to the models.

The second necessary consequence of the CIRP approach is that there will always be more published experiments than CIRP. Unless the original experiment and its replication(s) are literally identical, this raises an issue—to which of the replications does one fit a model? One possibility is to fit each replication separately. Although such an approach has obvious advantages in terms of completeness, it has disadvantages in terms of clarity and efficiency. The CIRP approach involves identifying a replication that is representative of what is known empirically, and is sufficiently well specified in terms of psychological stimulus representation, and empirical results, to allow meaningful modeling. This is one sense in which a CIRP is a *Canonical* Phenomenon. Deciding which experiment among replications is canonical is a matter of judgment, and a judgment that is unlikely to receive consensual approval in every case. The CIRP approach does not solve this problem, but does require that the introduction of a CIRP is accompanied by an explanation of the choices made (so that the choices are transparent and open to rational challenge).

One strength of the CIRP approach is that it separates empirical phenomena from the interpretation placed upon those phenomena. A replication that is successful but whose authors favor a different explanation to the original study is still a successful replication. The theoretical legwork comes in the form of comparing formal models to phenomena. Separating phenomena from formal models should make it easier for groups with different theoretical approaches to work in a distributed-collaborative manner.

2.3. CIRP registration

In Section 2.2, we proposed that each row of the Grid should comprise a Canonical Independently Replicated Phenomenon (CIRP). The concept discussed in the current sub-section is that CIRP should be recorded in a freely available store of target data sets that is open not only for inspection, but also for addition and improvement by the community.[5]

The existence of such a store is important for distributed collaboration on the formal modeling of psychological processes. This is because relative-adequacy assessments of models require that the models have been assessed against a commonly-defined set of phenomena. If this is not apparent, imagine the alternative

[4] Web of Science reports 1457 citations as of 2016-06-15.

[5] One must also accept the possibility that a CIRP might be deprecated in the event a phenomenon turns out to be a false positive despite being independently replicated.

situation where model A is assessed against phenomena 1–3, and is found to accommodate 1 and 2 but not 3. Model B is assessed against phenomena 4–6, and is found to accommodate all three phenomena. Which is the more adequate model, A or B? The answer is one cannot tell, because model B has not been assessed against phenomena 1–3 and model A has not been assessed against phenomena 4–6.

We further propose that this freely-available store of CIRP should have a standard format, not only for the recording of target data, but also for concise documentation of each data set. Such documentation would need to specify, at a minimum, (1) the source of the data set, (2) references that establish independent replication of the phenomenon, and (3) reasons for choosing this particular data set as the CIRP from among the set of replications. We provide a more detailed specification of our proposed documentation format in Section 4.4. Such documentation is brief, and is not intended as a replacement for more fulsome publications in traditional outlets. Rather, our expectation is that CIRP registrations would be one tangible product of synthetic review articles published in high-quality journals. In the same way that data publication increases the citation rate of empirical reports (Piwowar et al., 2007), CIRP registration could act to increase the citation rate of review articles, as the review, not the CIRP documentation, would be the definitive reference for the means by which the CIRP had been determined.

2.4. Free, open-source model implementations

One of the main sources of inefficiency in formal modeling is the lack of publicly-available model implementations. For example, at the time of writing, the majority of the formal models of category learning had no publicly-available implementation.[6] The amount of effort this wastes is large. For example, most of the work involved in writing Edmunds & Wills (2016), which established that COVIS could accommodate the result shown in Figure 1, came from having to implement the COVIS model. The lack of a publicly-available implementation increases the entry cost to using a formal model and hence disincentivizes the exploration of models by those outside the research group that created them.

We thus propose that another key resource for distributed collaboration is a freely accessible store of model implementations. Below, we introduce and discuss the three attributes we consider essential in such an archive: open-source software (Section 2.4.1), archival storage (Section 2.4.2), and cross-compatibility (Section 2.4.3).

2.4.1. Open-source software

It is important for distributed collaboration that model implementations be open source. Open-source software can be examined by others, so they can understand how it works. The fact they can do this also increases the chances that bugs are detected. Just as important, the free availability of source code means others can build upon and improve what the author has provided, either in terms of improving the efficiency or flexibility of the implementation, or by using components of the code in their own development of other formal models. This approach to software development is often known as "free software", where "free" refers to freedom rather than cost; it reflects the freedom of users to do what they wish with the code (Stallman, 2015; Williams, 2002). The ALCOVE model (Attention Learning COVEring map, Kruschke, 1992) is an example of good practice in this regard because its source code is publicly available (Kruschke, 1991).

The source code provided by Kruschke is an example of good practice for another reason, too — it requires only open-source tools (e.g. Stallman & the GCC team, 2016) to turn the source code into a working model implementation. The importance and strength of this open-tools approach can be illustrated by comparison to the alternative—use of proprietary tools. For example, MATLAB (MathWorks, 2016) is a proprietary tool used by some modelers. Its source code is not available, so researchers cannot see for themselves how MATLAB works. They cannot modify or improve it, and they have to take on trust that it is error free. The vendor can change the way a proprietary tool operates such that things you've written no longer work, and decline to sell or support the older versions of the tool for which your code did work.[7] In contrast, with

[6] Examples include: COVIS (Ashby et al., 1998), ATRIUM (Erickson & Kruschke, 1998), KRES (Rehder & Murphy, 2003).

[7] This really happens. SPSS (IBM, 2016) will not open its own output files if they were generated before 2007 (U. Mass., n.d.). Hypercard programs cannot be run on modern hardware (Oren, 2004). Visual Basic 6 support was withdrawn (Microsoft, 2008), despite a long public campaign (Ramel, 2016). In contrast, ALCOVE was written on a discontinued computer architecture and operating system (Markoff, 1989; Wikipedia, 2016b), yet compiles with minimal modification on modern machines.

open-source tools, if the developers choose to change the tool in a way that causes you a problem, you can choose to continue to use the old version (the old version remains freely available indefinitely). Incidentally, users of MATLAB may wish to consider OCTAVE (Eaton, 2014), an open-source tool that runs MATLAB code with little to no modification.

In summary, we propose that a publicly-accessible archive of model simulations should be open source, both in terms of the model code itself, and in terms of requiring only open-source tools to turn the source code into a working model implementation.

2.4.2. Archival storage

Although Kruschke's ALCOVE implementation is commendable for being open source and based on open tools, there are other respects in which it could be improved. For example, Kruschke's ALCOVE implementation is only available on his personal website (Kruschke, 1991). Contrast this to research article publication. Few authors or readers of research articles would be satisfied with a publication approach that involved posting only on a personal website. One central reason for the existence of journals, and other archival systems (e.g. arXiv; Ginsparg, 1999), is the security others gain from knowing that the material upon which they base their own research cannot disappear at the whim (or demise) of the original author. Computer code should be accorded no lesser status. Specifically, our proposal is that model implementations should be stored in a way that could reasonably be considered archival.

We further propose that the storage should not only be archival, but also have version control (e.g. Collins-Sussman et al., 2011). Version control means that any changes to the code since publication are recorded, that each update is assigned a unique version number, and that all versions remain publicly available. This means that researchers are able to say that their simulation was based on a particular numbered version of a model implementation, and be confident that this version will continue to be publicly available, even if superseded by a more recent version. This is essential for the reproducibility of published simulations.

In summary, our proposal is that model implementations should be stored in a system that could reasonably be considered to be archival, and which has effective version control. One example of good practice in this regard is the SUSTAIN model (Supervised and Unsupervised STratified Adaptive Incremental Network; Love et al., 2004), as archived by Gureckis (2014). Another is the DIVA model (Kurtz, 2007), as archived by Conaway (2016).

2.4.3. Cross-compatibility

Two model implementations, even those written in the same language, can operate in a sufficiently different way that their operation cannot be easily combined in a study of relative model adequacy. For example, Kruschke's ALCOVE code (Kruschke, 1991) reads parameters and stimulus inputs from a text file, working through the presented stimuli in order, and outputting each response to another text file. In contrast, the code for a different model used by O'Connell et al. (2016) generates the stimulus representations as part of the main code. Thus, in the former case, the model and data sets are separated while, in the latter, they are integrated. These kinds of differences substantially increase the effortfulness of assessing different models against a common set of phenomena. The solution is for different models to have, as far as is possible, a common structure for inputs and outputs. This is what we mean by cross-compatibility, and we propose that model implementations should be cross-compatible.

2.4.4. Summary

Distributed collaboration on the determination of relative model adequacy would be facilitated by the existence of a free, publicly-available, open-source, version-controlled archive of cross-compatible model implementations. As in the case of CIRP registration, such an approach can only get traction if it supplements rather than supplants traditional measures of academic esteem. It seems to us that such an archive meets that goal. In particular, it benefits those who have created these models by assisting the widespread adoption of the models by the community. This should in turn boost citations of the journal articles introducing and investigating those models. So, like CIRP registration, publicly-available model implementations are intended to enhance, rather than supplant, traditional publication routes.

2.5. Simulation publication

If the rows (CIRP) and columns (models) of the Grid (Figure 2) were freely available, there would still be a need for some mechanism by which the results of individual simulations (cells) could be archived. Given the already-discussed need for an archive of CIRP and model implementations, it seems efficient to locate a reproducible account of simulations in the same location. As in these other cases, it is important that this archive of simulations enhances rather than supplants traditional publication routes. We envisage that researchers will generate simulations in pursuit of some broader goals. For example, they may wish to compare the relative adequacy of a few models across a range of replicated phenomena. The results of this investigation would, one assumes, be the subject of peer-reviewed journal article publication in the normal way. What the archive of simulations would provide is the code necessary to reproduce the conclusion recorded in the Grid. The Grid, through the mechanism of registering simulations, also becomes a central searchable database of all relevant simulations. This should further enhance distributed collaboration and, by so doing, improve citation metrics.

2.5.1. A technical note about stimulus representations

This sub-section is a discussion of a technical issue that the casual reader can safely skim. The technical issue concerns the that fact that formal models of cognitive processes seldom start with a retinal representation of stimuli. Rather, they assume the presence of certain perceptual processes prior to the input of the model, and provide a somewhat more abstract representation of the experimental stimuli. For example, one common approach in the modeling of category learning is to represent each stimulus as a point in a multidimensional psychological stimulus space (Shepard, 1957, 1987). The input representations chosen by a modeler when simulating a given experiment are an essential component of the model specification. The simulation cannot be reproduced without the stimulus representations, and different stimulus representations could lead to different conclusions.

The technical question that arises is how one handles stimulus representations within an archive of simulations. One approach is to include stimulus representations in each of the individual simulation archives. However, we advocate a slightly different approach. Although different models sometimes use different input representations, it is also the case that some models are input compatible. For example, some versions of exemplar and prototype models use the same input representations and differ only in how that input is subsequently processed (e.g. Nosofsky, 1987). It therefore makes sense to publish stimulus representations separately from simulations, as simulations are specific to one model–CIRP combination (i.e. they concern one cell of the Grid), while stimulus representations can be common to multiple cells in a given row.

Separate publication of stimulus representations also provides a convenient location for brief documentation and justification of the choices made in constructing these stimulus representations. Documentation of a stimulus-representation archive would briefly explain the way in which the representation was determined, citing (where appropriate) the empirical data used in the construction of the representations (e.g. a multidimensional scaling solution).

2.6. Summary

We have outlined what we see as the necessary components for distributed collaboration on formal modeling in psychology. In summary, what is needed is a free, open, documented, version-controlled archive of (1) CIRP (Canonical Independently Replicated Phenomena), (2) cross-compatible model implementations, and (3) simulations.

3. Introduction to `catlearn`

In this section we introduce `catlearn`, the framework we have written to support distributed collaboration on formal modeling in psychology. `catlearn` is a free and open-source extension ("package") of the R language (R Core Team, 2016). Like R itself, `catlearn` is made available under the GPL (General Public License; Free Software Foundation, 2007), which ensures the software is, and will remain, free and open source. The project title, `catlearn`, was once an acronym (CATegory LEARNing), but the goals of

the framework have outgrown the acronym, and the word `catlearn` should now be treated as an arbitrary proper noun.

3.1. Why R?

Why is `catlearn` based around R? R is perhaps best known to psychologists as software for statistical analysis. It provides similar functionality to other statistical analysis software used by psychologists, but also many other features.[8] Of particular relevance to formal modeling, R supports non-linear optimization (the process by which formal models are typically fitted to data). Due to R's increasing popularity as an analysis tool for psychologists, situating `catlearn` within R brings the advantage of a familiar environment that contains many of the tools needed to inspect and analyze the output of model simulations. R is also easy to learn, relative to more low-level languages such as C.

Another strength of R is that it provides a simple system for documentation to be incorporated within `catlearn`. Preceding a command by a question mark, e.g. `?t.test` brings up documentation relevant to that command. This in-built documentation system provides an ideal means to document the rows (CIRP), columns (model implementations), and cells (simulations) of the Grid (Figure 2), and also to document the commands that generate input representations. R packages can also include datasets, which are loaded using the `data` command, e.g. `data(USArrests)`. This means that, when combined with the documentation system, R packages are well suited to CIRP registration.

A further strength of R is that it has dedicated version-controlled archival systems. Those wishing to contribute to `catlearn` can do so at any time using the version-controlled archive provided by R-Forge (Theußl & Zeileis, 2009). Users interested in the daily-updated development version of `catlearn` (the "unstable" version) can also download it from R-Forge. Users who prefer a slightly older but more fully tested version (the "stable" version) can download it from the Comprehensive R Archive Network (CRAN). CRAN is a robust version-controlled archive hosted simultaneously in over 120 different locations worldwide.

One possible objection to situating `catlearn` within R is that R code doesn't always run as fast as code written in some other languages. In many cases, this is not an issue from the user's perspective as code written in R still runs instantaneously. However, model implementations may, in some cases, benefit from being written in a compiled language. This can be achieved within R in a number of ways—for example, through use of the `Rcpp` package (Eddelbuettel & Francois, 2011; Eddelbuettel, 2013). `Rcpp` allows one to write time-critical pieces of code in `C++`, and integrate them seamlessly into R. R also links to a number of other languages, including FORTRAN, Java, Python, and MATLAB (via OCTAVE, Gaujoux, 2015).

3.2. Stateful list processors

The `catlearn` package, as described so far, provides the framework for a free, open, documented, version-controlled archive of CIRP, model implementations, input representations and simulations. One thing that the above discussion did not consider was cross-compatibility of models. In order to facilitate model comparisons, each model implementation should, as far as is possible, require similar inputs and provide similar outputs. Of course, this correspondence cannot be complete, as formal models differ in their assumptions about input representations, and have different parameters. While some differences are perhaps inevitable, it makes sense to eliminate unnecessary differences in implementation. In pursuit of that goal, model implementations within `catlearn` employ the *stateful list processor schema*. This schema is most easily described with the aid of a concrete example, which is provided in Section 4.

4. Examples

In the current section, we provide some specific examples of how `catlearn` can be used as an archive of model implementations, CIRP, and simulations. The examples are written from a user perspective, rather than a developer perspective. In other words, they assume a situation where you are using content already

[8] The Comprehensive R Archive Network (R Foundation, 2016) lists 8993 packages that extend the base functions of R (2016-08-20).

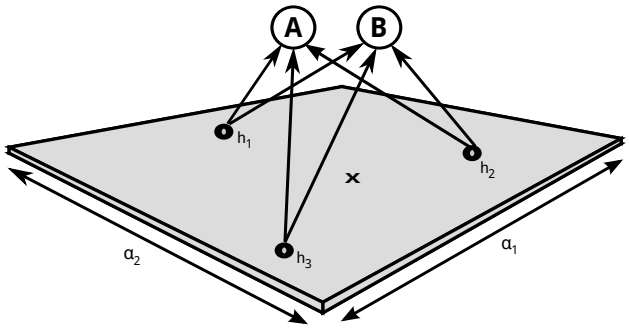

Figure 3: Architecture of the ALCOVE model (Kruschke, 1992). The gray quadrilateral is a three-dimensional depiction of a two-dimensional plane, representing a psychological stimulus space. Points h_1, h_2, and h_3 represent the location of radial-basis ("exemplar") units within that space. Point **x** represents the presented stimulus. The lettered circles are category representations, and the arrows connecting to them are variable-strength connection weights from the radial-basis units. α_1 and α_2 represent the attention allocated to each of the two dimensions of the space; the arrows beside these α illustrate that dimensional allocation acts to stretch and squash psychological space in ALCOVE. Image author: Andy J. Wills. CC BY 4.0.

present in the `catlearn` package, rather than adding new content yourself. Adding content to `catlearn` is discussed in Section 5.1.

Readers may find the following examples below easier to follow if they have a working version of R and the `catlearn` package in front of them. A guide on how to install both can be found at the `catlearn` project website (Catlearn Research Group, 2016). Once you have installed R and `catlearn`, type `library(catlearn)` in R to make `catlearn` available in your current session. Typing `?catlearn` then provides a brief help screen, with instructions of how to get a list of available commands. Each command then has its own help file (e.g. `?shin92`).

The examples that follow are drawn from category learning but—to reiterate—`catlearn` is a general-purpose framework for the formal modeling of psychological processes. Our examples come from category learning simply because that is the area with which we are most familiar. Our examples concern one of the central unresolved questions of category learning—are categories represented by prototypes (e.g. Reed, 1972) or by the storage of specific examples (e.g. Medin & Schaffer, 1978), or something else (e.g. Nosofsky et al., 1994)?

We start by describing the implementation of an exemplar model (Section 4.1), and a prototype model (Section 4.2), within `catlearn`. We then derive a CIRP (Section 4.3), and discuss its documentation within `catlearn` (Section 4.4). Next, we discuss an input-representation archive (Section 4.5), which is used by an archived simulation of our example CIRP with a formal model (Section 4.6). After that, we introduce the use of an Ordinal Adequacy Test to automatically evaluate whether the simulation successfully accommodates the CIRP (Section 4.7), and show how the Grid acts as a central record of simulation results (Section 4.8). Finally, we invite the reader to explore some of the other content of the `catlearn` package (Section 4.9).

4.1. ALCOVE model implementation

In this section, we describe the implementation of the ALCOVE model (Kruschke, 1992) within the `slpALCOVE` command of the `catlearn` package. The first sub-section is a mathematical description of ALCOVE, as some understanding of how ALCOVE works is required to understand how the `slpALCOVE` command works. This section can be skimmed by those already familiar with ALCOVE. For the mathematically unconfident, it should be sufficient to get the gist of this mathematical sub-section. The second sub-section describes the implementation of ALCOVE within `catlearn`.

4.1.1. Description of ALCOVE

ALCOVE (Kruschke, 1992) is one of the most influential models of category learning.[9] Figure 3 summarizes its architecture. ALCOVE is a connectionist model that assumes stimuli are represented as points

[9]Web of Science reports 814 citations as of 2016-06-16.

in a multidimensional psychological stimulus space (Figure 4A). Thus, each stimulus is represented by a vector, which we will denote here as $\mathbf{x}$. For example, for stimuli varying in size and angle, one might write $\mathbf{x} = \begin{pmatrix} 0.4 & 0.5 \end{pmatrix}$, where the two values represent the psychological size and angle of the presented stimulus.

Presentation of a stimulus leads to the activation of radial-basis nodes (Cheney, 1966). Radial-basis nodes, like stimulus representations, can be considered as points in stimulus space (Figure 3). In virtually all applications of ALCOVE there is exactly one radial-basis node for each unique training stimulus. Although these radial-basis nodes are often called "exemplar" nodes, this description is something of a misnomer as, in most applications, all the nodes exist before training begins. It is perhaps better to think of these radial-basis nodes as a simplification of the abstract concept behind ALCOVE, which is that there are radial-basis nodes randomly scattered across stimulus space (the "COVEring map" of ALCOVE).

However one prefers to think about it, the architecture of the radial-basis layer of ALCOVE is fully specified by the matrix $\mathbf{h}$, which has the same number of columns as there are radial-basis units (j columns), and the same number of rows (i rows) as there are psychological stimulus dimensions. For example, in a simple size-angle experiment with four stimuli, the architecture of the radial-basis layer might be described as

$$\mathbf{h} = \begin{pmatrix} 0.4 & 0.8 & 0.8 & 0.4 \\ 0.4 & 0.4 & 0.8 & 0.8 \end{pmatrix}$$

where each column represents the location of one training exemplar in stimulus space.

ALCOVE computes the activations of each of the radial basis nodes with the following equation:

$$a_j^h = exp[-c(\sum_i \alpha_i |h_{ji} - x_i|^r)^{q/r}] \tag{1}$$

This equation specifies that the activation of each radial-basis node is a decreasing function of its distance from the presented stimulus. Where $r = 2$, that distance is Euclidean (Figure 4B); where $r = 1$, the distance is city-block (Figure 4C). Euclidean distance is typically used for integral stimuli, city-block for separable stimuli (see Garner, 1976). Where $q = 1$, the decreasing function is exponential (Figure 4D); where $q = 2$, it is Gaussian. Exponential decay is typically used (Shepard, 1987); occasionally Gaussian decay is used where stimuli are highly confusable (Ennis, 1988).

Stimulus space can be uniformly contracted or expanded using c (see Figure 5C). c is largely treated as an arbitrarily variable parameter (Wills & Pothos, 2012), although psychologically it is intended to represent cognitive discriminability or memorability of stimuli, so if information about this could be derived independently for a set of stimuli then it would constrain model fitting somewhat (minimally, c would need to be a non-decreasing function of discriminability/memorability). In related models, application to amnesic data takes this form (e.g. Nosofsky & Zaki, 1998).

In Equation 1, α_i represents dimensional attention on dimension i. Dimensional attention acts as a multiplier to distance, stretching psychological space uniformly across one axis (see Figure 5B). The model

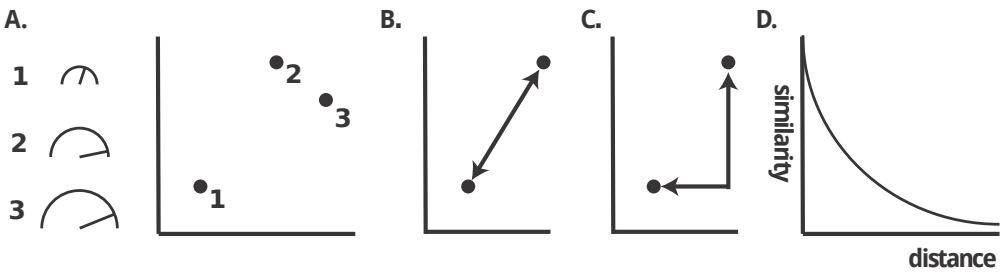

Figure 4: A. Representing the similarity structure of stimuli 1, 2, and 3 in a two-dimensional geometric space; in this example, the dimensions of this space are readily interpretable as size and angle. B. Euclidean distance ($distance^2 = x^2 + y^2$). C. City-block distance ($distance = x + y$). D. An exponential decay relationship between similarity and distance in psychological space. Reproduced from "On the Adequacy of Current Empirical Evaluations of Formal Models of Categorization" by A.J. Wills & E.M. Pothos, 2012, Psychological Bulletin, 138, 102–125. Copyright 2012 by American Psychological Association.

is typically initialised with equal attention to all dimensions, conventionally summing to unity, e.g. $\alpha = \begin{pmatrix}0.5 & 0.5\end{pmatrix}$.

The activation process represented by Equation 1 results in a vector of radial-basis node activations, e.g. $\mathbf{a^h} = \begin{pmatrix}0.4 & 0.8 & 0.4 & 0.8\end{pmatrix}$. Radial-basis node activation propagates forward to a set of output (category) nodes, which then have activation $\mathbf{a^o}$. There is one output node for each category, and each radial-basis node has one variable weight connection to each output node. The weight-state of the model is thus a matrix of the form:

$$\mathbf{w} = \begin{pmatrix}0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\end{pmatrix}$$

which has k rows (one for each ouput node) and j columns (one for each hidden node). Activation of the output nodes is calculated with the standard connectionist equation

$$a_k^o = \sum_j w_{kj} a_j^h \tag{2}$$

The forward propagation of activation ends with a standard exponential ratio rule to convert activation to response probability

$$P(K) = \frac{exp(\phi a_K^o)}{\sum_k exp(\phi_k^o)} \tag{3}$$

where ϕ is a non-negative response-scaling parameter. Low values of ϕ lead to approximately probability-matching behavior (category selection probability is proportional to the ratio of output node activations). High values of ϕ lead to approximately winner-take-all behavior (the category with the highest activation is always selected). We note in passing that this exponential ratio rule is probably a poor model of categorical decisions (Wills et al., 2000).

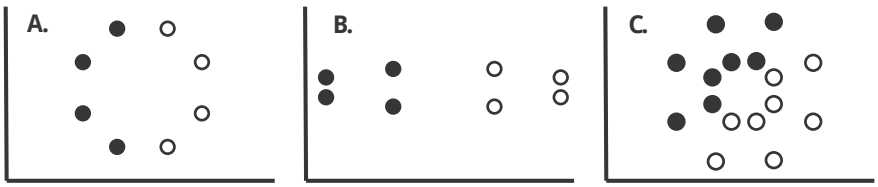

Figure 5: A. Geometric representation of two categories, each of four stimuli (category membership denoted by color of dot). B. Stretching along the x axis and compression along the y axis, thereby increasing within-category similarity and decreasing between-category similarity. C. Overall expansion of psychological similarity space. Reproduced from "On the Adequacy of Current Empirical Evaluations of Formal Models of Categorization" by A.J. Wills & E.M. Pothos, 2012, Psychological Bulletin, 138, 102–125. Copyright 2012 by American Psychological Association.

In ALCOVE, learning is driven by "teacher" (t) values. The presence of a category label is represented by a teacher signal of 1; absence of a category label is typically represented by -1. The teacher is typically considered to be "humble". This means that if the output activation is more extreme than the +1/-1 teaching value, then the ouput activation is used as the teaching signal.

Learning of connection weights from radial-basis nodes to output nodes uses a standard summed-error term:[10]

$$\Delta w_{kj} = \lambda_w (t_k - a_k^o) a_j^h \tag{4}$$

where λ_w is the associative learning-rate parameter, which can range from 0 to 1. This equation acts to change connection weights in the direction that most rapidly reduces error.

[10]See Le Pelley (2004) for a discussion of summed- and separate- error term equations.

Attentional weights are also learned. This is achieved by the backpropagation of error (Rumelhart et al., 1986; Werbos, 1974) to the radial-basis nodes in the standard manner:

$$b_j = a_j^h \sum_k (t_k - a_k^o) w_{kj} \tag{5}$$

This back-propagated error is then used to change the attentional weight for each dimension:

$$\Delta\alpha_i = -\lambda_\alpha \sum_j b_j c |h_{ji} - x_i| \tag{6}$$

where λ_α is the attention learning-rate parameter, which again can range from zero to one. Implementations of ALCOVE constrain attentional weights to be non-negative. ALCOVE's attentional learning system acts to stretch and squash psychological stimulus space (Figure 5B) in the directions that most rapidly reduce error.

4.1.2. Implementation of ALCOVE

The function `slpALCOVE` in the `catlearn` package implements ALCOVE as a stateful list processor.[11] In this section, we explain how `slpALCOVE` works and, by so doing, also introduce the more general concept of a stateful list processor—a structure that is well suited to many formal models of cognitive processes.

As with all R commands, the `?` query returns help documentation for `slpALCOVE` (type `?slpALCOVE` to see it). Although complete, such documentation is typically concise rather than tutorial in nature. Where possible, one should cite, within the help documentation, another tutorial publication. In the case of `slpALCOVE`, the current article serves that purpose.

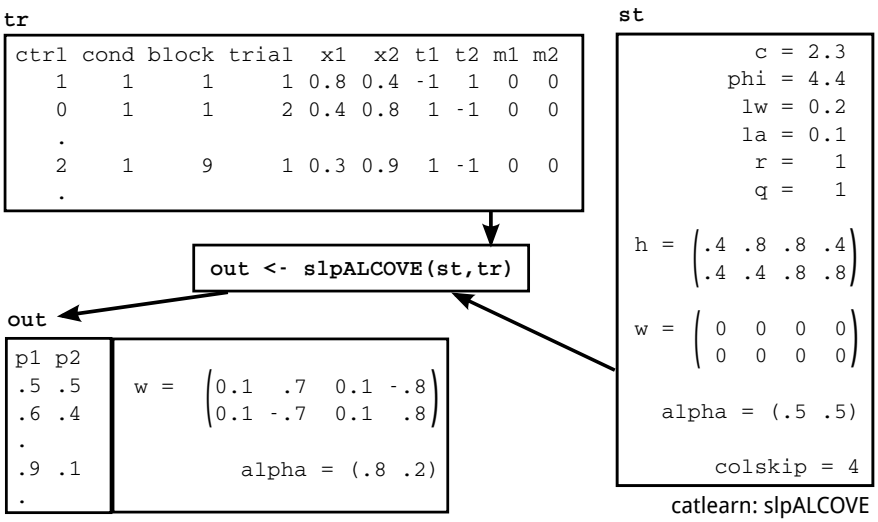

Figure 6: ALCOVE implemented as a stateful list processor in catlearn. Image author: Andy J. Wills. CC BY 4.0.

Figure 6 provides an illustration of the basic operating principles of the `slpALCOVE` stateful list processor. The `slpALCOVE` function takes two primary inputs from the user: `st` and `tr`. Input `st` ("state") is a list containing the model parameters and the model's initial state. For `slpALCOVE`, the parameters are *c, r, q, phi* (ϕ), *lw* (λ_w), *la* (λ_α), and *h*. These parameters were defined above. List `st` also contains the model's initial state: *w* contains the initial connection weights, and *alpha* (α) contains the initial attentional weights. These model states were defined above. The only other entry in `st` is *colskip*, which is an instruction to the model implementation to ignore the first N columns in the training matrix.

[11] The `slp` in `slpAlCOVE` stands for Stateful List Processor

Object `tr` ("training") is a matrix, where each row is one trial that is presented to the network. The nature of a list-processor architecture is that the model processes all of these trials, and in the order they are presented. The ordering of trials (e.g. through randomization) is undertaken by other functions that generate `tr`, not by the list processor.

The first column of `tr`, *ctrl* ("control"), is normally zero, but can be set to other values to change the mode of operation of the model mid-list. The options currently implemented in `slpALCOVE` are: 1 = reset the model to its initial state, 2 = freeze learning on the current trial. The latter option allows no-feedback test phases to be implemented in the standard way (i.e. by running the model with learning rates set to zero). The former option allows the list processor to run the same model with the same parameters on multiple runs of the experiment. This can be useful for averaging out order effects.

After *ctrl*, there are a variable number of optional columns that can contain any numerical information the user wishes. Typically, these will be used to contain information such as experimental condition, block, and trial. These columns are ignored by the list processor, it just needs to be told how many columns there are to ignore, using the *colskip* parameter; *colskip* should be set to equal the number of optional columns, plus one. Setting *colskip* incorrectly can lead to unpredictable behavior.

After these optional columns, the next set of columns contain the input representation, x, and then the teaching signals, t, both of which were defined above. The final set of columns, m, should in most circumstances be set to zero. Setting an m column to 1 indicates that, on that trial, that stimulus dimension was not presented. This is useful in some cases where stimuli have multiple components, not all of which are presented on every trial.

In addition to `st` and `tr`, the `slpALCOVE` function takes a number of other optional input arguments. These set the operating mode for the model. Most models have a number of different variants, and ALCOVE is no exception. If these options are not set (as in the example in Figure 6), `slpALCOVE` runs the version of ALCOVE specified in the previous section. For the sake of brevity, the alternative options are not discussed here, but are documented in `?slpALCOVE`.

So, `slpALCOVE` takes `tr` and `st` as input. The simulation is run by creating `tr` and `st`, and then entering the command `slpALCOVE(st,tr)`. A concrete example follows later in this article. When the simulation completes it returns a list, as illustrated in Figure 6. The list has three components: p, w, and *alpha*. Component p is a matrix that has one column for each category unit of the model, and one row for each trial of the simulation. The numbers in the matrix are the predicted probability of each category response on each trial. Matrix p can easily be recombined with the training matrix, `tr`, for easier interpretation of the model's predictions.[12] This combined matrix can then be analyzed using the same techniques as used to analyze participant data in R (e.g. through the use of the `aggregate` command).

The other two parts of the output list are w and *alpha*, which give the values of the connection and attentional weights at the end of the simulation. This information can be used to explore how the model has learned the category structure with which it was presented.[13] The returning of final w and *alpha* is the property of `slpALCOVE` that leads us to describe it as a *stateful* list processor. In other words, it returns not only the model's predictions but also its (final) state. The stateful property of the implementation is important because it avoids an otherwise serious limitation of the list-processor architecture. Specifically, in non-stateful list processors, the input to a model cannot be made contingent on its previous output.

In category learning, the most common example in which input needs to be contingent on output is in training to criterion. In modeling training to criterion, one might run the model until the probability of a correct response exceeds some threshold (e.g. 0.99). A list processor cannot do this because the number of training items has to be specified from the outset (it's the number of rows in the *tr* matrix). A stateful list processor can handle training to criterion because it returns its final state. For example, to model training to criterion where the criterion is checked once per block of trials, one presents a single block of trials to `slpALCOVE`, and checks whether the returned probabilities exceed the criterion. If they do not, one sets the initial state of the model (w, *alpha*) to the values returned by `slpALCOVE`, and runs another block of trials.

[12] `out <- slpALCOVE(st, tr); prob <- out$p; out2 <- cbind(tr,prob)`

[13] For more detailed exploration, the *xtdo* option can be used to output the model's state on every trial, see `?slpALCOVE`.

Thus, the model can "pick up where it left off" because it can be returned to the state it was in at the end of the last block.

4.2. proto-ALCOVE model implementation

Although ALCOVE is typically considered as an exemplar-like model, it is also possible to use it as a prototype model (e.g. Johansen & Palmeri, 2002) The change is straightforward, one simply uses the category prototypes as the locations for the radial-basis units. Specifically, each category unit is connected to exactly one radial-basis unit, with that radial-basis unit located at the average position of the category examples in psychological stimulus space. Thus, `slpALCOVE` can be used as a model implementation for both ALCOVE and proto-ALCOVE; all that differs is the contents of the h matrix passed to the model.

4.3. Derivation of a CIRP

In this section, we provide an example of a segment of a review article that uses `catlearn` as an archival source. The review segment does not contain the introductory information typically expected of a full review article, nor does it explore future avenues for research. A full review article would of course contain such information, which is why we describe what follows merely as a "segment". Although the information provided is correct and complete to the best of our knowledge, the primary purpose for showing it here is to provide a concrete context for our later examples of how to use the `catlearn` package. Note that `catlearn` permits very concise referencing of archival material; this is important because space in high-profile journals is scarce. Indeed, in this review segment, all archival information is provided in the caption of Figure 7. In the sections following this review segment, we describe the registration of the CIRP, the input representation, and the simulations, within the `catlearn` package.

4.3.1. Category size

Category size is the number of examples of a category that have been presented to the participant. The category-size effect (e.g. Homa et al., 1973, 2008) is the phenomenon that, as category size increases, the accuracy of generalization to new members of that category also increases. The category-size effect has been previously argued to support prototype theory (e.g. Homa et al., 1973), while others have argued that exemplar theories can also accommodate this effect (Shin & Nosofsky, 1992).

Evidence for the category-size effect comes from two different, independently-replicated, procedures. In the first, within-subjects, procedure (e.g. Breen & Schvaneveldt, 1986; Homa et al., 1973; Shin & Nosofsky, 1992) participants learn to categorize members of categories of different sizes. For example, category A might have 3 members, category B 6 members, and category C 9 members. In a subsequent test phase, participants are asked to classify novel members of these categories. The central result is that test-phase performance is better for novel members of the larger categories.

One potential shortcoming of the within-subjects procedure is that this observed accuracy-size relationship could be due to a bias to classify new items into the category that was most likely to occur during training. Some within-subject demonstrations of the category-size effect provide evidence against a response bias explanation (e.g. Homa & Chambliss, 1975), but this aspect of the within-subjects procedure has not been independently replicated.

The second, between-subjects, procedure provides an alternative solution to the response-bias issue (e.g. DiNardo & Toppino, 1984; Homa et al., 1987; Shin & Nosofsky, 1992). In this procedure some participants are trained on small categories, while others are trained on larger categories. For example, one group might experience two categories, each of size 3, while another group experiences two categories, each of size 10. In a subsequent test phase, the size-10 group is more accurate in their classification of novel stimuli. Both prototype and exemplar theories can accommodate this ordinal pattern, as illustrated in Figure 7. Shin & Nosofsky (1992) was registered in `catlearn` as a CIRP due to the availability of a multi-dimensional scaling solution in that paper (data that is useful in the generation of input representations for some formal models).

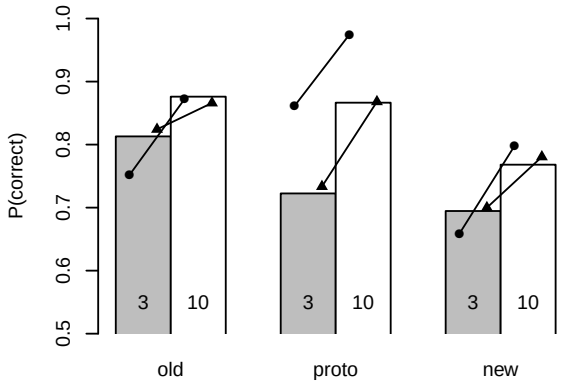

Figure 7: Effects of category size on response accuracy in category learning. Gray bars are the size-3 condition, white bars are the size-10 condition. The pairs of bars show the effect of category size on classification accuracy at test on: (a) the items seen during training ("old"), (b) the prototypes from which those training items were generated ("proto"), and (c) novel stimuli generated from the same prototypes ("new"). Circles are proto-ALCOVE's predictions; triangles are ALCOVE's predictions. Empirical data are from Shin & Nosofsky (1992, Table 10), where "old" is the mean across all old stimuli, etc. In this experiment, participants classified nine-vertex polygon stimuli. `catlearn` archive IDs: 1, 2. Image author: Andy J. Wills. CC BY 4.0.

4.4. CIRP registration

The CIRP derived in the above review segment has been archived in the `catlearn` package as `shin92`. Typing

```
data(shin92)
```

will load the relevant empirical results, which can be viewed in the normal ways (e.g. through the RStudio interface). Typing

```
?shin92
```

brings up the CIRP registration document, which has the following principal sections: Description, Format, Details, Author, Source, and References. *Description* contains a brief (one-paragraph) summary of the CIRP. *Format* describes the content of each of the columns of the dataset. *Details* briefly summarizes how the CIRP was derived, and provides a reference for a more detailed derivation (in this case, the current article). *Details* also briefly summarizes the structure of the experiment, although for fuller details the reader is expected to read the original journal article, Shin & Nosofsky (1992) in this case. *Source* gives the reference from which the data were drawn, while *References* contains any other references used in this CIRP registration document. *Author* provides the name and email address of the person registering the CIRP in `catlearn`. Sometimes, as in the present case, the registrant will be unconnected with the authors of the original experiment.

4.5. Input representation archive

The `catlearn` package also contains a command that generates input representations for the `shin92` CIRP. Typing

```
shin92train()
```

produces its default output, which is eight blocks of training and three blocks of test for the size-3 condition of the CIRP. This is the same number of blocks as in the experiment. Trial order is randomized. The output is in a format directly compatible with `slpALCOVE`, and uses multidimensional scaling co-ordinates from a similarity-scaling study reported in the same paper as the CIRP. Typing

```
?shin92train
```

calls up a help file that concisely documents the function. The format is similar to that of the CIRP registration document: Description, Usage, Arguments, Details, Author, and References. *Description* is a single-paragraph summary of the command's purpose. *Usage* shows the arguments taken by the command, and their default values. For example, one part of the *Usage* entry is `condition = "equal3"`. This states that one argument of the command is `condition`, which has a default value of `equal3`. As described in the *Arguments* section, the `condition` argument sets which experimental condition to simulate. The *Arguments* section concisely describes every argument specified in *Usage*. *Details* is used to document the meaning of the columns of the matrix produced by the command. *Author* and *References* serve the same functions as in the CIRP registration document.

4.6. Simulation archive

The `catlearn` package contains the two simulation archives used to generate Figure 7: `shin92exalcove` and `shin92protoalcove`. Here, we'll only look at `shin92exalcove`, but they both operate in a similar way. Simulation archives are intended to run automatically without any interaction from the user. Hence, typing

```
out <- shin92exalcove()
```

will run the simulation and store the results. It may take a second or two to run, as the simulation involves 100 simulated participants per condition, which are averaged to produce the output. Type

```
out
```

to display the results. The output of simulation archives produce predictions in the same format and order as the CIRP that is the subject of the simulation. Thus, assuming you have the CIRP data loaded—type `data(shin92)`— it is a simple matter compare the predictions with the observations. For example,

```
plot(shin92$c2acc,out$c2p); abline(0,1)
```

will produce a standard scatterplot of predictions against observations. Quantitative measures of fit can also be easily obtained. For example,

```
cor(shin92$c2acc,out$c2p)^2
```

reports r^2 for the fit (0.97), and

```
ssecl(shin92$c2acc,out$c2p)
```

reports the sum of squared errors (0.27). Typing

```
?shin92exalcove
```

produces a help file, with the normal sections: Description, Usage, Arguments, Details, Author, References. The meaning of these sections should be clear from the description of corresponding help files for CIRPs and input representations.

Simulation archive functions always have default model parameters, but it is easy to run the simulation with different parameters. For example, try

```
out2 <- shin92exalcove(params = c(0.1,0.6,0.1,0.8))
```

In this simulation, the c parameter has been much reduced, making the stimuli more confusable. Typing

```
cor(shin92$c2acc,out2$c2p)^2
```

shows that the correlation between model and data is now also reduced (0.86). The fact that these parameters are available to the user makes it possible to use a simulation archive function as the subject of a model-fitting procedure, such as non-linear optimization using the `optim` command, or other interesting approaches such as parameter space partitioning (Pitt et al., 2006). However, such procedures are outside the scope of the current paper.

The open-source nature of `catlearn` means that the precise decisions made by the researcher archiving the simulation are available for inspection. The code for the `shin92exalcove` simulation is quite short (26 statements) and can been seen by typing `shin92exalcove`.[14] The brevity of this simulation archive is due both to the compactness of R, and the fact that the CIRP, input representations, and model implementations are contained within their own documented functions.

4.7. Ordinal adequacy test

Wills & Pothos (2012) proposed the relative adequacy evaluation of multiple formal models against a broad range of known phenomena. Such a comparison is rendered more straightforward by employing the principle of ordinal adequacy (i.e. does the model reproduce the ordinal effects defined by the CIRP?). One of the chief advantages of ordinal adequacy over more quantitative measures is that it provides a clear, binary, answer to the question of whether model X accommodates phenomenon Y. When the phenomenon is defined ordinally, the model either can, or cannot accommodate it. This contrasts with more continuous measures of it, such as r^2, where it is not possible to say what specific level of r^2 represents a meaningful failure to accommodate a phenomenon, or whether a model with an r^2 of .98 is in any meaningful sense providing a more adequate account than a model with an r^2 of .95 (or whether it is, for example, better at fitting noise).

Wills & Pothos (2012) discuss this point at length, and we will not cover the detail of the argument again here. Rather, the current article is concerned with the practical issue of how one defines ordinal adequacy. In a rich data set such as `shin92`, there are many ordinal relationships; which of these should a model be evaluated against? Fortunately, the CIRP documentation provides an answer to this question—the independently replicated result is that performance on the novel stimuli from the 10-item category is more accurate than performance on the novel stimuli from the 3-item category. Such a test for ordinal adequacy can be defined algorithmically, and hence tested automatically. This is what OAT (Ordinal Adequacy Test) commands within `catlearn` do.

For the `shin92` CIRP, the OAT is provided by `shin92oat`, with the usual documentation available via `?shin92oat`. OAT work in the same way as other evaluations of model fit. For example, recall that in an earlier section we calculated the r^2 between the model's output and the CIRP. OAT work in the same way, except that they already include within them a definition of the critical ordinal properties of the empirical data set, and hence only take the model output as an argument. As previoulsy specified, the model output must have the same format and be in the same order as the CIRP data set. Thus, assuming the output of one's chosen model is contained in the variable `out` (see earlier example), the command

```
shin92oat(out)
```

returns `TRUE` indicating the model, with the default parameters, passes the ordinal adequacy test. Referring to our previous example, we can see that making the stimuli more confusable leads to the model failing the ordinal adequacy test

```
shin92oat(out2)
```

[14] Viewing the source code like this removes comments; the comments can be viewed on R-Forge.

Hence, the model's ordinal adequacy is parameter dependent.

OAT functions also provide a convenient place to locate the calculation of summary results, and the `xtdo` (eXTenDed Output) parameter provides a standard way of doing this. For example, the output of simulation `shin92exalcove` runs to 46 lines, which does not aid visualization of the main patterns. The review segment presented earlier in the article summarized the model's output with six figures (see Figure 7). The calculation of these figures can be reproduced thus

```
shin92oat(out,xtdo=TRUE)
```

Finally, note that the central conclusion of the review segment—that both ALCOVE and proto-ALCOVE accommodate the observed results—can now be reproduced by two, very brief commands that run in a few seconds:

```
shin92oat(shin92exalcove()); shin92oat(shin92protoalcove())
```

4.8. The Grid

The Grid (Figure 2) is included as stored data within the `catlearn` package. It is loaded by the command

```
data(thegrid)
```

and can be viewed in the normal way. The documentation can be read by typing `?thegrid`. The columns of `thegrid` record, in turn, a unique ID number for the simulation, the CIRP simulated (e.g. `shin92`), a descriptive name for the model tested (e.g. "protoALCOVE"), the result of the ordinal adequacy test (1,0), the command for the simulation archive (e.g. `shin92exalcove`) and the command for the OAT (e.g. `shin92oat`). The structure of the `catlearn` package means that the results of these simulations can be automatically verified by running the listed simulation command through the listed OAT command. This provides a mechanism by which the accuracy of the Grid can be automatically maintained; these checks will be performed prior to each stable release of `catlearn`.

Given we have been visualizing The Grid as a matrix throughout the article, it may be slightly surprising to see that it is represented as a list. There are two reasons for this. First, large matrices quickly become unweildy. Second, and more subsantively, a list permits more than one entry per cell. This is likely to become important in future as, for example, increasing data quality leads to more detailed OAT or as more ambitious multi-CIRP simulations are run to minimize the problem of arbitrarily variable parameters.[15]

4.9. Other examples

We only had space to discuss one CIRP, one model, and one simulation archive in this article, but the current content of `catlearn` is more extensive than this. As an exercise, use `thegrid` and the help functions within `catlearn` to investigate the rest of the content.

5. Overview and conclusion

In this section we summarize how the concepts and framework of `catlearn` covered in the previous sections work in practice. The perspective is slightly different to that taken in the Examples section. In that section, we focused on the user's perspective—how to use content already available in `catlearn`. In the current section, we briefly focus on how to achieve specific goals through contributing new content to the `catlearn` project; for further details, see Catlearn Research Group (2016). We then finish with some conclusions and speculations.

[15]The problem of arbitrarily variable parameters is a serious one, but a discussion is beyond the scope of the current article; see Wills & Pothos (2012)

5.1. Contributing to `catlearn`

Here are the primary ways of contributing content to `catlearn`:

I've conducted an independent replication. If this phenomenon is not yet documented in `catlearn`, write a CIRP for it (see Sections 2.3, 4.4). Cite the unique CIRP ID you have created (e.g. `jones2017`) in your article, and use the mailing list (Catlearn Research Group, 2016) to encourage others to simulate it. If you have conducted a replication of a CIRP already in `catlearn`, update the CIRP documentation to include a citation of your manuscript, and cite the existing CIRP ID in your manuscript.

I'm writing a review article. If you have identified one or more Canonical Independently Replicated Phenomena (CIRP), register or update CIRP entries within `catlearn` (see above).

I'm writing a modeling paper. Use `catlearn` as an archive for your simulations. More specifically: (1) If your paper identifies one or more CIRP, register or update them within `catlearn` (see above). (2) Use the models already within `catlearn` to run your simulations (if the model does not exist in `catlearn`, see below). (3) Archive each simulation of a CIRP in `catlearn` (see Section 2.5, 4.6). (4) Archive each input representation (see Section 2.5.1, 4.5). (5) Provide OAT functions to automatically test each model for ordinal adequacy against a CIRP registered within `catlearn` (see Section 4.7). (6) Register each simulation in the Grid (see Section 2.1, 4.8). Cite the unique simulation IDs from the Grid in your article.

I'm developing a formal model. Make your model available within `catlearn`. When you publish your model, cite your `catlearn` model ID (e.g. "slpNewModel") in your manuscript. Bear in mind that models within `catlearn` are implemented as stateful list processors (see Sections 3.2, 4.1.2).

I want to use an existing model not included in catlearn. Implement the model within `catlearn` yourself (see above), or encourage the original authors to do so, or use the mailing list (Catlearn Research Group, 2016) to request that someone else in the `catlearn` project does the development work.

5.2. Conclusion

We have argued that making efficient progress in the formal modeling of psychological processes requires a technological mechanism that supports distributed collaboration. Indeed, we believe the scope of the task before the formal modeling community is so large that the choice is between collaboration or failure. We have provided a viable technological mechanism for distributed collaboration in the `catlearn` package, and we have illustrated its use through an example from the modeling of category learning. The `catlearn` package is free, open source, and publicly available.

We're aware that the approach we are proposing might be seen as controversial. Psychologists, one might argue, don't work together in this way. To the extent this is true, we hope that, by taking on the initial work ourselves, we can catalyze a change in our culture. The success of the kind of approach we advocate has clear precedents in other fields. Perhaps the most striking example is the development of Linux.

The development of Linux started with the GNU project (Stallman, 1983). Later, Linus Torvalds started to write a kernel (Torvalds, 1991), which ultimately meant that Linux could become a free-standing computer operating system. Linux is now the most used operating system in the world—it runs all Android smart phones, it's behind around half of all internet servers, and dominates the embedded systems market. The development of Linux happened through the distributed collaboration of thousands of programmers, taking the freely available code, improving it, and sharing their improvements back to the community. The Linux kernel, from around 10,000 lines of code in 1991, has reached, as of June 2015, almost 20 million lines of code, contributed by around 14,000 programmers (Wikipedia, 2016a).

Linux demonstrates that competition and collaboration need not be mutually exclusive—many of the programmers contributing to Linux are employed by commercial organizations, who presumably feel that supporting Linux is to their competitive advantage. If we can convince psychologists that distributed collaboration is the best way to overcome the limited progress of the last four decades, perhaps formal theories of cognitive processes will see an explosion of development over next 30 years comparable to the explosion seen by Linux in the last 30. Let's start working together.

Acknowledgments

The authors thank Eric S. Raymond and Richard Stallman for their inspirational work on the philosophy and practice of free and open source software.

References

Anderson, C., Bahnik, S., Barnett-Cowan, M., Bosco, F., Chandler, J., . . . , & Zuni, K. (2016). Response to comment on "Estimating the reproducibility of psychological science". *Science*, *351*, 1037.3.

Ashby, F. G., Alfonso-Reese, L., Turken, A., & Waldron, E. (1998). A neuropsychological theory of multiple systems in category learning. *Psychological Review*, *105*, 442–481.

Ashby, F. G., & Maddox, W. T. (2011). Human category learning 2.0. *Annals of the New York Academy of Sciences*, *1224*, 147–61.

Blair, M. R., & Homa, D. (2003). As easy to memorize as they are to classify: The 5–4 categories and the category advantage. *Memory and Cognition*, *31*, 1293–1301.

Breen, N., & Schvaneveldt, R. W. (1986). Classification of empirically derived prototypes as a function of category experience. *Memory and Cognition*, *14*, 313–320.

Catlearn Research Group (2016). The catlearn project. `http://catlearn.r-forge.r-project.org/`.

Cheney, E. (1966). *Introduction to approximation theory*. New York: McGraw-Hill.

Collins-Sussman, B., Fitzpatrick, B. W., & Pilato, C. M. (2011). Version control with subversion. `http://svnbook.red-bean.com/`. Accessed: 2016-06-15.

Conaway, N. (2016). MATLAB scripts to run DIVA simulations. `https://github.com/nolanbconaway/DIVA`.

DiNardo, M. J., & Toppino, T. C. (1984). Formation of ill-defined concepts as a function of category size and category exposure. *Bulletin of the Psychonomic Society*, *22*, 317–320.

Eaton, J. (2014). GNU Octave. `https://www.gnu.org/software/octave/`.

Eddelbuettel, D. (2013). *Seamless R and C++ integration with Rcpp*. New York: Springer.

Eddelbuettel, D., & Francois, R. (2011). Rcpp: Seamless R and C++ integration. *Journal of Statistical Software*, *40*, 1–18.

Edmunds, C. E. R., & Wills, A. J. (2016). Modeling category learning using a dual-system approach: A simulation of Shepard, Hovland and Jenkins (1961) by COVIS. In *Proceedings of the 38th Annual Conference of the Cognitive Science Society*. Cognitive Science Society.

Ennis, D. M. (1988). Confusable and discriminable stimuli: Comment on Nosofsky (1986) and Shepard (1986). *Journal of Experimental Psychology: General*, *117*, 408–411.

Erickson, M. A., & Kruschke, J. K. (1998). Rules and exemplars in category learning. *Journal of Experimental Psychology: General*, *127*, 107–140.

Free Software Foundation (2007). GNU general public license. `https://www.gnu.org/licenses/gpl.html`. Accessed: 2016-06-15.

Garner, W. R. (1976). Interaction of stimulus dimensions in concept and choice processes. *Cognitive Psychology*, *123*, 98–123.

Gaujoux, R. (2015). RcppOctave: Seamless Interface to Octave – And Matlab. `https://cran.r-project.org/web/packages/RcppOctave/index.html`.

Gilbert, D. T., King, G., Pettigrew, S., & Wilson, T. D. (2016). Comment on "Estimating the reproducibility of psychological science". *Science*, *351*, 1037.2.

Ginsparg, P. (1999). arXiv.org. `http://arxiv.org/`. Accessed: 2016-06-15.

Gureckis, T. M. (2014). sustain_python. `https://github.com/NYUCCL/sustain_python`. Accessed: 2016-06-15.

Homa, D., Burruel, L., & Field, D. (1987). The changing composition of abstracted categories under manipulatiosns of decisional change, choice difficulty, and category size. *Journal of Experimental Psychology: Learning, Memory and Cognition*, *13*, 401–412.

Homa, D., & Chambliss, D. (1975). The relative contributions of common and distinctive information on the abstraction from ill-defined categories. *Journal of Experimental Psychology: Human Learning and Memory*, *104*, 351–359.

Homa, D., Cross, J., Cornell, D., Goldman, D., & Shwartz, S. (1973). Prototype abstraction and classification of new instances as a function of number of instances defining the prototype. *Journal of Experimental Psychology*, *101*, 116–122.

Homa, D., Proulx, M. J., & Blair, M. R. (2008). The modulating influence of category size on the classification of exception patterns. *Quarterly Journal of Experimental Psychology*, *61*, 425–443.

IBM (2016). SPSS v. 24. `http://www.ibm.com/analytics/us/en/technology/spss/`.

Johansen, M. K., & Palmeri, T. J. (2002). Are there representational shifts during category learning? *Cognitive Psychology*, *45*, 482–553.

Kruschke, J. K. (1991). ALCOVE.c. `www.indiana.edu/~kruschke/articles/ALCOVE.c`. Accessed: 2016-06-15.

Kruschke, J. K. (1992). ALCOVE: An exemplar-based connectionist model of category learning. *Psychological Review*, *99*, 22–44.

Kurtz, K. J. (2007). The divergent autoencoder (DIVA) model of category learning. *Psychonomic Bulletin and Review*, *14*, 560–576.

Le Pelley, M. E. (2004). The role of associative history in models of associative learning: A selective review and a hybrid model. *Quarterly Journal of Experimental Psychology*, *57B*, 193–243.

Ledgerwood, A. (2014). Introduction to the special section on advancing our methods and practices. *Perspectives on Psychological Science*, *9*, 275–277.

Love, B. C., Medin, D. L., & Gureckis, T. M. (2004). SUSTAIN: a network model of category learning. *Psychological Review*, *111*, 309–332.

Makel, M., Plucker, J., & Hegarty, B. (2012). Replications in psychology research: How often do they really occur? *Perspectives on Psychological Science*, *7*, 537–542.

Markoff, J. (1989). Eight desktop computers introduced by Digital. `http://www.nytimes.com/1989/01/11/business/company-news-8-desktop-computers-introduced-by-digital.html`. Accessed: 2016-06-15.

MathWorks (2016). MATLAB R2016a. `http://mathworks.com/products/matlab`. Accessed: 2016-06-15.

Medin, D. L. (2011). Comments on models and categorization theories: the razor's edge. In E. M. Pothos, & A. J. Wills (Eds.), *Formal Approaches in Categorization* (pp. 325–331). Cambridge, UK: Cambridge University Press.

Medin, D. L., & Schaffer, M. M. (1978). Context theory of classification learning. *Psychological Review*, *85*, 207–238.

Microsoft (2008). Microsoft product lifecycle: Visual Basic 6. `https://support.microsoft.com/en-us/lifecycle/search`.

Minda, J. P., & Smith, J. D. (2002). Comparing prototype-based and exemplar-based accounts of category learning and attentional allocation. *Journal of Experimental Psychology: Learning, Memory and Cognition*, *28*, 275–292.

Morey, R. D., Chambers, C. D., Etchells, P. J., Harris, C., Hoekstra, R., Lakens, D., Lewandowsky, S., Morey, C. C., Newman, D. P., Schonbrodt, F. D., Vanpaemel, W., Wagenmakers, E.-J., & Zwaan, R. A. (2016). The peer reviewers' openness initiative: incentivizing open research practices through peer review. *Royal Society Open Science*, *3*, 150547.

Nosofsky, R. M. (1984). Choice, similarity, and the context theory of classification. *Journal of Experimental Psychology: Learning, Memory and Cognition*, *10*, 104–114.

Nosofsky, R. M. (1987). Attention and learning processes in the identification and categorization of integral stimuli. *Journal of Experimental Psychology: Learning, Memory and Cognition*, *13*, 87–108.

Nosofsky, R. M. (2000). Exemplar representation without generalization? Comment on Smith and Minda's (2000) Thirty categorization results in search of a model. *Journal of Experimental Psychology: Learning, Memory and Cognition*, *26*, 1735–1743.

Nosofsky, R. M., Kruschke, J. K., & McKinley, S. C. (1992). Combining exemplar-based category representations and connectionist learning rules. *Journal of Experimental Psychology: Learning, Memory and Cognition*, *18*, 211–233.

Nosofsky, R. M., Palmeri, T. J., & McKinley, S. C. (1994). Rule-plus-exception model of classification learning. *Psychological Review*, *101*, 53–79.

Nosofsky, R. M., & Zaki, S. R. (1998). Dissociation between categorization and recognition in amnesic and normal individuals: An exemplar-based interpretation. *Psychological Science*, *9*, 247–255.

O'Connell, G., Myers, C., Hopkins, R., McLaren, R. P., Gluck, M. A., & Wills, A. J. (2016). Amnesic patients show superior generalization in category learning. *Neuropsychology*, *in press*.

Open Science Collaboration (2015). Estimating the reproducibility of psychological science. *Science*, *349*, aac4716.

Oren, T. (2004). A eulogy for Hypercard. `http://due-diligence.typepad.com/blog/2004/03/a_eulogy_for_hy.html`.

Pashler, H., & Harris, C. (2012). Is the replicability crisis overblown? Three arguments examined. *Perspectives on Psychological Science*, *7*, 531–536.

Pashler, H., & Wagenmakers, E.-J. (2012). Editors' introduction to the special section on replicability in psychological science: A crisis of confidence? *Perspectives on Psychological Science*, *7*, 528–530.

Pitt, M. A., Kim, W., Navarro, D. J., & Myung, J. I. (2006). Global model analysis by parameter space partitioning. *Psychological Review*, *113*, 57–83.

Piwowar, H. A., Day, R. S., & Fridsma, D. B. (2007). Sharing detailed research data is associated with increased citation rate. *PLOS One*, *2*, e308.

R Core Team (2016). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing Vienna, Austria. URL: `https://www.R-project.org/`.

R Foundation (2016). CRAN: The Comprehensive R Archive Network. `https://cran.r-project.org/`.

Ramel, D. (2016). Microsoft doesn't budge on classic Visual Basic. `https://adtmag.com/articles/2016/05/24/microsoft-visual-basic.aspx`. Accessed: 2016-06-15.

Reed, S. K. (1972). Pattern recognition and categorization. *Cognitive Psychology*, *3*, 382–407.

Rehder, B., & Hoffman, A. B. (2005). Thirty-something categorization results explained: Selective attention, eyetracking, and models of category learning. *Journal of Experimental Psychology: Learning, Memory and Cognition*, *31*, 811–829.

Rehder, B., & Murphy, G. (2003). A knowledge-resonance (KRES) model of category learning. *Psychonomic Bulletin and Review*, *10*, 759–784.

Rumelhart, D. E., Hinton, G., & Williams, R. (1986). Learning internal representations by error propagation. In D. E. Rumelhart, & J. McClelland (Eds.), *Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1* (pp. 318–362). Cambridge, MA: MIT Press.

Shepard, R. N. (1957). Stimulus and response generalization: a stochastic model relating generalization to distance in psychological space. *Psychometrika*, *22*, 325–345.

Shepard, R. N. (1987). Toward a universal law of generalization for psychological science. *Science*, *237*, 1317–1323.

Shepard, R. N., Hovland, C. L., & Jenkins, H. M. (1961). Learning and memorization of classifications. *Psychological Monographs*, *75*, Whole No. 517.

Shin, H. J., & Nosofsky, R. M. (1992). Similarity-scaling studies of dot-pattern classification and recognition. *Journal of Experimental Psychology: General*, *121*, 278–304.

Smith, J. D., & Minda, J. P. (2000). Thirty categorization results in search of a model. *Journal of Experimental Psychology: Learning, Memory and Cognition*, *24*, 1411–1436.

Stallman, R. M. (1983). `https://groups.google.com/forum/#!msg/net.unix-wizards/8twfRPM79u0/1xlglzrWrU0J`. Accessed: 2016-06-16.

Stallman, R. M. (2015). *Free software, Free society: Selected essays of Richard M. Stallman*. Boston, MA: Free Software Foundation.

Stallman, R. M., & the GCC team (2016). GCC, the GNU compiler collection. `https://gcc.gnu.org/index.html`. Accessed: 2016-06-15.

Theußl, S., & Zeileis, A. (2009). Collaborative software development using R-Forge. *The R Journal*, *1*, 9–14.

Torvalds, L. (1991). `https://groups.google.com/forum/#!topic/comp.os.minix/dlNtH7RRrGA[1-25]`.

U. Mass. (n.d.). SPSS legacy viewer for Windows. `http://www.umass.edu/statdata/software/downloads/SPSSLegacy/`. Accessed: 2016-06-15.

Werbos, P. J. (1974). *Beyond regression: New tools for prediction and analysis in the behavioral sciences*. Ph.D. thesis Harvard University.

Wikipedia (2016a). Linux. `https://en.wikipedia.org/wiki/Linux`. Accessed: 2016-06-16.

Wikipedia (2016b). Ultrix. `https://en.wikipedia.org/wiki/Ultrix`. Accessed: 2016-06-15.

Williams, S. (2002). Free as in freedom – Richard Stallman's crusade for free software. `http://www.jus.uio.no/sisu/free_as_in_freedom.richard_stallman_crusade_for_free_software.sam_williams/sisu_manifest.html`.

Wills, A. J., & Pothos, E. M. (2012). On the adequacy of current empirical evaluations of formal models of categorization. *Psychological Bulletin*, *138*, 102–125.

Wills, A. J., Reimers, S., Stewart, N., Suret, M. B., & McLaren, I. P. L. (2000). Tests of the ratio rule in categorization. *Quarterly Journal of Experimental Psychology*, *53A*, 983–1011.