



BIROn - Birkbeck Institutional Research Online

Yule, P. and Cooper, Richard P. (2003) Express: a web-based technology to support human and computational experimentation. *Behavior Research Methods, Instruments, & Computers* 35 (4), pp. 605-613. ISSN 0743-3808.

Downloaded from: <https://eprints.bbk.ac.uk/id/eprint/557/>

Usage Guidelines:

Please refer to usage guidelines at <https://eprints.bbk.ac.uk/policies.html>
contact lib-eprints@bbk.ac.uk.

or alternatively

**Birkbeck ePrints: an open access repository of the
research output of Birkbeck College**

<http://eprints.bbk.ac.uk>

Yule, Peter and Cooper, Richard P. (2003).
Express: a web-based technology to support
human and computational experimentation.
*Behavior Research Methods, Instruments, &
Computers* **35** (4) 605-613.

This is an exact copy of a paper published in *Behavior Research Methods, Instruments, & Computers* (ISSN 0743-3808). Copyright and all rights therein are retained by authors or by other copyright holders. All persons downloading this information are expected to adhere to the terms and constraints invoked by copyright. © 2003 Psychonomic Society, Inc.

Citation for this copy:

Yule, Peter and Cooper, Richard P. (2003). Express: a web-based technology to support human and computational experimentation. *London: Birkbeck ePrints*. Available at: <http://eprints.bbk.ac.uk/archive/00000557>

Citation as published:

Yule, Peter and Cooper, Richard P. (2003). Express: a web-based technology to support human and computational experimentation. *Behavior Research Methods, Instruments, & Computers* **35** (4) 605-613.

<http://eprints.bbk.ac.uk>

Contact Birkbeck ePrints at lib-eprints@bbk.ac.uk

Express: A Web-based technology to support human and computational experimentation

PETER YULE and RICHARD P. COOPER

Birkbeck College, University of London, London, England

Experimental cognitive psychology has been greatly assisted by the development of general computer-based experiment presentation packages. Typically, however, such packages provide little support for running participants on different computers. It is left to the experimenter to ensure that group sizes are balanced between conditions and to merge data gathered on different computers once the experiment is complete. Equivalent issues arise in the evaluation of parameterized computational models, where it is frequently necessary to test a model's behavior over a range of parameter values (which amount to between-subjects factors) and where such testing can be speeded up significantly by the use of multiple processors. This article describes Express, a Web-based technology for coordinating "clients" (human participants or computational models) and collating client data. The technology provides an experiment design editor, client coordination facilities (e.g., automated randomized assignment of clients to groups so that group sizes are balanced), general data collation and tabulation facilities, a range of basic statistical functions (which are constrained by the specified experimental design), and facilities to export data to standard statistical packages (such as SPSS). We report case studies demonstrating the utility of Express in both human and computational experiments. Express may be freely downloaded from the Express Web site (<http://express.psyc.bbk.ac.uk/>).

Many researchers and teachers of research methods now routinely use computer-based experimentation packages to generate and present a wide variety of experiments to participants via standard desktop computer hardware. However, most experiment presentation packages provide limited support for coordinating multiple participants and collating the resulting data. Typically, a data file is generated for each participant, and the experimenter must collect all of these files (often from multiple computers, especially if the experiment is run as part of a laboratory class) and merge those data files, using a utility program supplied with the experiment presentation package. The process of collating data in this manner is laborious, time-consuming, and prone to error. In the case of experiments with between-subjects factors, it also requires that the experimenter monitor group sizes and explicitly assign individuals to each group to ensure that group sizes are balanced across conditions.

Perhaps surprisingly, similar issues arise within the field of computational modeling (Yule & Cooper, 2001). Many computational models of cognitive processes contain parameters (e.g., learning rate or level of noise) on

which the model's behavior is dependent, and in these cases model evaluation requires exploring the behavior of the model for different values of the parameters. This is necessary to determine the best fit to the data, to establish independence of behavior from parameters thought to be noncritical, or to determine how the model behaves when impaired (see Plaut & Shallice, 1993, for an extended example of the methodology).

There is a direct correspondence between exploring the behavior of a model for different parameter values and performing a standard psychological experiment with human (or indeed animal) participants. Parameters correspond to between-subjects factors, and different values for the parameters correspond to different levels of those factors. As in a standard experiment, the factors may be ordinal (e.g., degree of connectivity, learning rate, etc.) or categorical (e.g., if alternative mechanisms or subprocesses within a model are being explored), and each run of the model for a given combination of parameter values may be thought of as corresponding to one experimental participant. The behavior of the model, as in the case of living participants, may be summarized through values of dependent variables, and within-subjects factors may be included in both human and model experiments by collecting values of the dependent variables over multiple trials or multiple blocks of trials. Finally, models frequently include random elements that affect their behavior (e.g., noise affecting activation values). In such cases, it is necessary to run the model multiple times for each combination of parameter values, to establish the model's mean behavior. Running the model N times

We are grateful to John Fox for his continuing support, to David Glasspool and Rachel McCloy for help with testing Express, and to Ulf-Dietrich Reips and William Schmidt for insightful comments on an earlier draft of this paper. This research was supported by EPSRC Grant GR/M89621. Correspondence concerning this article should be addressed to P. Yule, School of Psychology, Birkbeck College, University of London, Malet Street, London WC1E 7HX, England (e-mail: p.yule@bbk.ac.uk).

in this way corresponds to running a group of N participants in the same condition.

There are just two differences between the standard experimental situation and the situation of model evaluation: participant type (living participants vs. computer models) and scale (tens or hundreds of participants vs. tens of thousands of model runs). The difference in scale arises because, even with just a few parameters, the space of model instances (i.e., the number of conditions) can quickly become large. For example, if there are just three parameters (e.g., connectivity, learning rate, and noise) having 4, 3, and 10 levels respectively, then 120 ($4 \times 3 \times 10$) model instances must be explored. With, say, 100 replications for each model instance, we find that we require 12,000 simulations to fully explore the model.

The scale of model evaluation means that thorough evaluation can require massive computing power and can generate several orders of magnitude more data than a standard experiment. With regard to the former, it is not uncommon in the computational literature for models to take weeks or months of processor time to develop and evaluate. Advances in computing technology have not helped reduce this time; rather, they have allowed ever more complex models to be evaluated. It is, therefore, common practice for modelers to run simulations on several computers simultaneously (when they are available), with each machine exploring a different region of the parameter space. Once all the machines are finished, the results can be collated. This use of multiple machines to speed up model evaluation raises exactly the same issues as those raised by current experiment presentation packages: The modeler must manually allocate machines to regions of the parameter space so that all combinations of parameter values are explored with approximately the same number of simulations (i.e., the modeler must manually balance the size of groups for each condition), and when the evaluation is complete, the modeler must manually collate the data from the various machines (or write a special purpose computer program to automate the job). The first of these issues can be complicated further by the fact that the available machines may run at different speeds, so optimizing the assignment of regions of parameter space to machines in such a way that everything is done in the least possible time is nontrivial.

This article describes Express, a Web-based system that automates the coordination of "clients" and the collation of the data that they generate. It has a variety of other features, including an experiment design editor, facilities for tabulation of data and export of raw data as text files, a range of common statistical tests, and an extensive help system. In the remainder of this article, the basic client-server model of Express is described. This is followed by a detailed discussion of the capabilities of the Express server and a description of the client-server interface. Two case studies, illustrating use of the server both in a standard experimental setting and as a harness for evaluating a computational model, are then de-

scribed. We conclude by discussing a number of residual issues, including security, system requirements, and the distribution policy.

EXPRESS: A GENERAL CLIENT-SERVER SOLUTION

The Internet provides a means by which the difficulties described above may be overcome. The essence of the approach adopted by Express is to treat both human participants interacting through experiment presentation software and computational models as clients that interact across the Internet (or across an intranet) with a server. The complete Express system consists of the Express server (a set of programs that run on a standard Web server and respond to Web requests) and an interface definition that specifies how clients may interact with the server.

The role of the Express server is to coordinate clients, independently of client type, and collate their results. When a client is run, it begins by querying the server for a set of parameters (typically values of the experiment's independent variables). It then operates on these parameters to produce a set of dependent measures (i.e., values for the experiment's dependent variables). These measures are then returned to the server, which stores them for analysis. In the case of experiments with within-subjects variables (where each client is required to perform under a variety of conditions), the server will return further parameters that define the next experimental condition. Client-server interactions will continue until the client has returned dependent measures for all the conditions. Figure 1 shows this interaction.

At present, two forms of clients are supported by the Express system: HTML/script clients, consisting of a system written in HTML and a browser scripting language (JavaScript or VBScript), for displaying an interface to a (presumably human) participant within a Web browser, and C/C++ clients, consisting of a computer program written in C or C++ and using a C Application Programming Interface (API) to interact with the Express server. This second form of client is particularly useful in computational modeling work, in which the model may be augmented to allow interaction with the Express server. However, it may also be of use for human experiments written within any programming language that allows interface to C. In either case, the server does not know or care whether the clients with which it interacts are humans performing a computer-based experiment or computer models attempting to simulate human behavior.

The use of a Web-based client-server system addresses the problems raised above and, hence, has major benefits for both standard experimental psychology and cognitive modeling. With respect to standard experimental psychology, it allows both the assignment of participants to groups and the collation of data to be automated. It also allows real-time monitoring of results, via

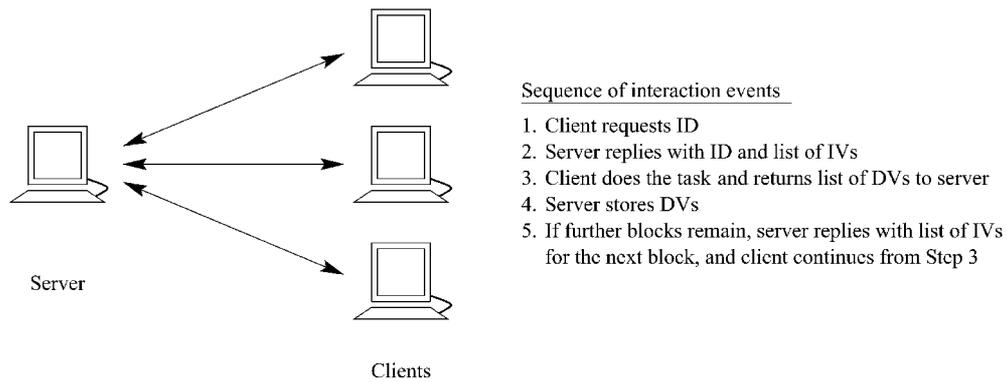


Figure 1. The relation of the server to clients and the sequence of interactions between the two. Clients may be human participants performing at a networked computer or computational models simulating some task. IV, independent variable; DV, dependent variable.

the built-in statistical facilities. This has proven to be particularly useful when upward of 20 participants are completing an experiment simultaneously during laboratory classes, and real-time monitoring can provide an early indication of a bug in a client or a problem with an experimental design, allowing remedial action to be taken quickly. With respect to cognitive modeling, the client-server system makes large-scale parameter studies, with many parameters and/or levels, more practicable than they might otherwise be, since it enables the distributed parallel execution of the client on all available processors, to avoid the typically long execution times on single-processor systems.

The use of a client-server architecture is not new in experimental psychology; in fact, it is inherent in Web-based experiment presentation systems (e.g., Birnbaum, 2000; Birnbaum & Wakcher, 2002; Hewson, Yule, Laurent, & Vogel, 2003; Reips, 2001; Schmidt, 1997), and several systems of this type are available, such as WEX-TOR (Reips & Neuhaus, 2002),¹ and SurveyWiz and FactorWiz (Birnbaum, 2000).² However, to our knowledge, none of these systems offers explicit support for modeling applications or even real-time analysis facilities. The problem of large-scale parameter manipulation in distributed computing has also been addressed by a number of general purpose systems, such as screensaver supercomputing systems (such as the United Devices toolkit),³ and the Grid project.⁴ None of these systems has support for both human and model participants, since this is a rather specialized requirement of the social and cognitive sciences.

The aim of Express is to provide a general facility for all experimental purposes in the social and cognitive sciences. In principle, Express could even be used as a *back end* for existing stand-alone computer-based experiment presentation packages, such as E-Prime (MacWhinney, St. James, Schunn, Li, & Schneider, 2001), ERTS (Beringer, 1994), SuperLab (Haxby, Parasuraman, Lalonde, & Abboud, 1993) and PsyScope⁵ (Cohen, MacWhinney, Flatt, &

Provost, 1993); interfaces for these systems may be developed in future work.

In the case of cognitive modeling, this generalized approach has an additional methodological advantage: It allows both models and human participants to interact with the same experimenter system. Human and model behavior may be directly compared by declaring client type as a between-subjects factor and allowing clients of both types to perform the same task. This ensures that all the factors of the experimental design (including within-subjects and between-subjects variables, stimulus randomization, and dependent measures) are held fixed between the two cases, which in turn ensures maximum possible comparability between human and model data sets. Several authors (e.g., Anderson & Lebiere, 1998; Ritter, Baxter, Jones, & Young, 2000; Yule & Cooper, 2001) have recently argued that this is an important step in improving the methodology of cognitive modeling.

THE EXPRESS SERVER

The functions of the Express server may be divided into two categories: functions involved in experiment administration and functions involved in client-server interaction. Express is implemented as a pair of CGI (common gateway interface) server programs, one for each set of functions. Both programs share a set of configuration and data files.

The Experiment Administration System

The experiment administration system provides Web-based tools for specifying the design of an experiment and for monitoring the experiment while it is being run.

The design editor. The design editor appears as a Web page with a series of buttons on the left and one of a number of forms on the right (see Figure 2). The buttons provide access to the various aspects of the design that may be configured via the editor, and the forms

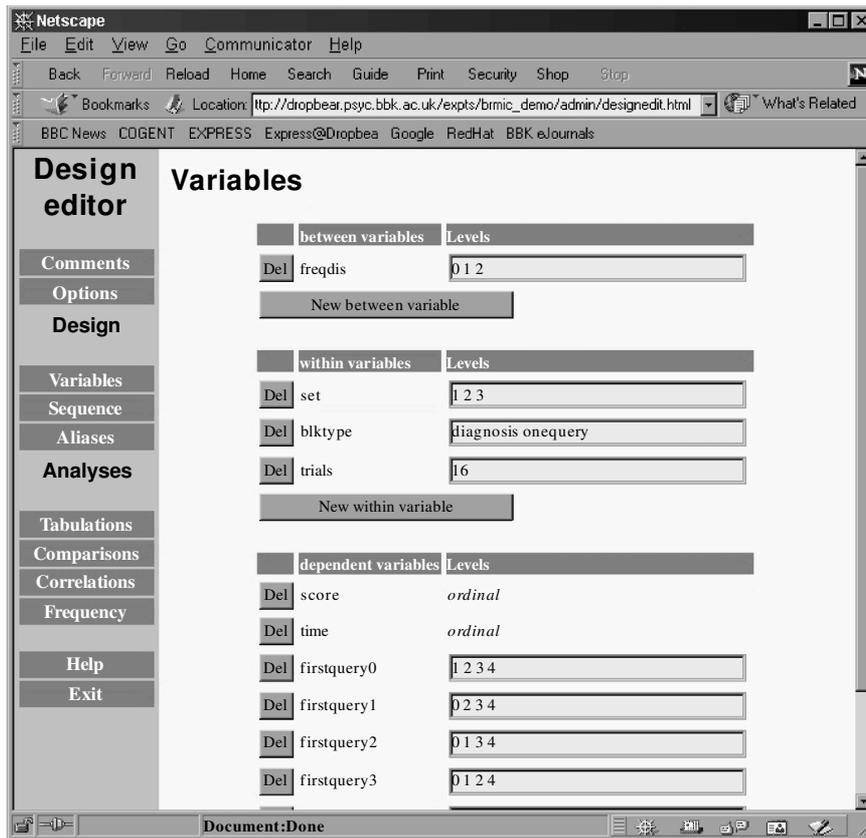


Figure 2. The Express design editor, showing the categories of design features on the left and the variables declaration form on the right.

allow the experimenter to specify details of those aspects. Specifically, the design elements allow the experimenter to do the following: provide a title and description of the experiment; declare independent (between-subjects and within-subjects) and dependent variables; declare scales and levels for all variable types; define pseudovariables or *aliases* (defined as functions of real variables); specify the sequential structure of the experiment in terms of blocks of trials and of trial-by-trial variable assignments and materials sequences; specify “include” files (typically containing HTML and JavaScript or VBScript, for use in human Web-based experiments); specify the number of replications in each between-subjects condition (typically used for model-based experiments); and specify various types of tabulations and statistical analyses based on previously declared variables. Information specified through the design editor is stored on the server as a design file. When the experimenter makes a request (e.g., by selecting to view a tabulation) or when a client makes a request (as will be described below), the server program uses the design file to determine how to fulfill the request.

Monitoring: Tabulations and statistical tests. The administration system allows real-time monitoring of an experiment’s progress. The form of this monitoring is

dictated by the tables and statistics specified within the design editor, as has been described above. Tabulations may show means, standard deviations, or other common descriptive statistics and may be defined for multiple independent and dependent variables. Statistics functions include a variety of standard parametric and nonparametric univariate and multivariate statistical tests. The integrity of these tests has been validated by comparing their results with those of SPSS on a range of test cases.

Tables and statistics are recalculated on request, allowing the experimenter to monitor results as the experiment progresses. Thus, the experimenter can use a Web browser on one machine to monitor the progress of a class of participants completing a lab. This may involve monitoring the number of participants who have completed each trial or monitoring trends in the data as they emerge.

Data may be displayed either in the form of HTML tables for human inspection or as tab-formatted plain text for export to commonly used statistical packages and spreadsheets, such as SPSS and Excel.

Other administrative functions. Express provides a variety of other administrative functions. These include the following: a status page (see Figure 3), which may be refreshed at any stage to show the progress of an experiment and which functions as a front end to the ex-

The screenshot shows a Netscape browser window with the URL `http://dropbear.psyc.bbk.ac.uk/expts/jdm6/index.cgi`. The page title is "Express Status". The main content area displays "JDM 6 (human)" and a description: "Continuing the series of JDM experiments. This one varies base-rates, in an effort to see if this has any effect on accuracy or querying strategies. The hypotheses are that it will not affect accuracy, but will affect strategy." Below this, it states "Experiment is running" and lists statistics:

- Assignment categories: 3
- Total completed jobs: 146
- Min completed jobs per category: 47
- Max completed jobs per category: 50
- Incomplete jobs: 2
- Excluded jobs: 58

A table titled "Incomplete jobs" is shown below:

ID	Host	Excluded	Comment	Trials
13397685	wallaroo.psyc.bbk.ac.uk	<input type="checkbox"/>	Test run for BRMIC screen dump	1
13463226	wallaroo.psyc.bbk.ac.uk	<input type="checkbox"/>	Test run for BRMIC screen dump	2

Figure 3. An Express status page, showing an experiment in progress. The experiment has three groups (one between-subjects variable with three levels). One hundred forty-six participants have completed the experiment, and another 2 are currently working on it. Data from 58 participants have been excluded.

periment administration system; a design summary page, which summarizes in one place key aspects of the experimental design, such as the number of factors and the levels of each factor; a host summary page to monitor the performance and activity of participating hosts; control functions for creating, copying, deleting, archiving, and renaming experiments (accessed through a special control panel); control functions for initialization of the experiment and starting and stopping execution; a mechanism for excluding selected participants from the analysis (allowing the experimenter to exclude outliers or participants whose data may be incomplete or otherwise suspect); and a contextual help system that provides documentation on all aspects of the server and that is keyed to automatically load the help text associated with the currently displayed administration page.

Client–Server Interaction

The experiment administration system is complemented by a client–server interaction system that responds to client requests. A client is any program that periodically delivers data to the server. As has been ex-

plained above, the data may be generated either by a human participant interacting through a front end that is presenting the experiment or by a computer model or other program using the C API.

The primary functions of the client–server interaction system are to control client access and assign clients to appropriate experimental conditions, to control the sequence of trials within a client, and to collate data returned by clients. Client requests are implemented as HTTP FORM submissions. Thus, HTML/script experiments may be implemented using standard HTML FORM elements or raw URLs. In such cases, the server responds to client requests by generating an HTML page (based on a template provided by the experimenter) containing a list of variable assignments embedded within a script environment. The API used for server interaction by models and experiments written in C includes functions that encode and submit client requests, wait for a response from the server, and decode the resulting variable assignments.

Participant identifier assignment and access control. Before a client begins an experiment, it should request a participant identifier (PID) from the Express

server. The server will respond with a unique identifier code, and a sequence of $\langle Var = Val \rangle$ pairs specifying levels of between- and within-subjects variables. This allows the client to begin the experiment.

Between-subjects balancing. If there are between-subjects variables, the server assigns PIDs to each combination of levels of these variables (i.e., each between-subjects condition) in a balanced way. When a client requests a new PID, the PID is assigned to the condition that currently has the fewest PIDs assigned to it (or if more than one such condition exists, to a randomly selected condition from this set). This method combines randomness of assignment of PIDs to conditions with evenness of assignment and ensures that group sizes are balanced whenever possible.

The participant assignment method is also able to cope with cases in which a between-subjects variable is a natural group variable, rather than an experimenter-assigned variable (e.g., sex). If a client request for a PID specifies a value for one or more between-subjects variables, participant assignment is constrained to the conditions defined by those values.

If the experimenter has specified a maximum number of replications in the experimental design and there are no conditions with fewer than this number of completed clients, a request for a new PID will yield a PID of 0 (zero). This indicates that the experiment is complete and that no further clients are required. This mechanism errs on the side of overassignment, in that it continues issuing new (nonzero) PIDs until all replications are *complete*. Replications in progress are discounted. This ensures that the requested number of replications is met, even if one or more clients fail, for one reason or another, to complete the experiment. However, it can lead to overassignment, where one or more conditions end up with more replications than requested. If necessary, the participant exclusion system (as used for excluding outliers or otherwise suspect data) may be applied to rebalance group sizes and exclude any excess replications from all the analyses.

Sequencing. When the client initiates the experiment or submits data (as at the end of a trial), the server must provide the next set of within-subjects variable assignments to the client (or a stop signal if there are no further within-subjects assignments for the current client). Each set of variable assignments defines a single trial, and each PID can participate in multiple trials, one after the other. Although between-subjects variable assignments are necessarily constant across trials for the same PID, within-subjects variables are typically varied across trials.

Express supports the specification of one or more series of blocks, where each block contains a series of trials. There is full control over within-subjects variable assignments both between and within blocks, and different types of trials may be presented in the same block. If desired, trial order can be randomized within each block. Finally, different sequences of blocks can be associated with levels of a between-subjects variable, allowing between-subjects order counterbalancing.

Data collation. When the client submits data, the server saves them as a data record, using the client's PID to ensure that data records from the same participant are appropriately tagged. The primary mechanism saves the values of declared dependent variables; the built-in tabulation and statistics facilities may be configured to use these data as described above. Express also supports a secondary mechanism for saving arbitrary raw or uninterpreted data.

Restarting client sessions. Sometimes a client session can be interrupted due to software or hardware error or because a participant inadvertently closes the browser window. Express supports a means of continuing an interrupted experiment, provided the original PID is supplied. This presents the last, uncompleted trial again, after which the sequence continues as normal. An additional mechanism allows the client to save some state information to the server on each trial, enabling even those clients that maintain their own state information across trials to be restarted properly.

TWO CASE STUDIES

Case Study 1: Harnessing a Standard Experiment

Much of our own empirical work has been concerned with categorization and information-seeking behavior in such tasks as medical diagnosis (e.g., Cooper & Yule, 1999; Cooper, Yule, & Fox, 2003; Yule, Cooper, & Fox, 1998). This work has been supported by successive versions of the Express server, driving a client written in JavaScript and HTML and presented through a standard Web browser.

The experiments have consisted of between four and six blocks, each of between 12 and 20 trials. The participants' goal was to learn the associations or correspondences between symptom patterns and hypothetical diseases, whose correspondences were generally not deterministic: Symptoms were typically unreliable indicators of diseases. Two types of block have been used. In one, participants are presented with full information about all the symptoms (i.e., which symptoms are present and which are absent), and they are required to use this information to make their diagnoses. Feedback indicating the correct diagnosis is then given, allowing them to learn the symptoms for that disease. In the other type of block, participants are given just one presenting symptom (e.g., *The patient has headache*) and must query additional symptoms (e.g., *Is vomiting present?*) before making a diagnosis. Again, feedback indicating the correct diagnosis is then given.

A relatively complex JavaScript/HTML client was used for these experiments. The client creates a new browser window that is divided horizontally into three sections. The top portion shows a row of colored rectangles depicting cards corresponding to each symptom. Each card indicates whether the symptom is present, absent, or (in the case of blocks in which only the presenting symptom is given) unknown, and the symptom order

is randomized within the row on each trial. Cards are sensitive to mouse clicks, so that clicking on a symptom whose status is unknown reveals that symptom's status. The lower third of the window contains a randomized row of cards representing diseases, which are also sensitive to mouse clicks. The middle panel of the window is reserved for feedback that is provided after the participant makes a diagnosis. Prior to each block, block-specific instructions are presented in the client window.

Different versions of the experiment have used different independent variables. Between-subjects variables have included the matrix of conditional probabilities of symptoms, given diseases, and the base rates of diseases. Within-subjects variables have included block number (i.e., level of practice) and block type. Dependent variables have included diagnostic accuracy over the block (i.e., the percentage of correct trials in each block), block time, and (in query blocks) the first symptom queried for each presenting symptom.

The design is clearly complex. All of these complexities were programmed through a mix of JavaScript and HTML, with the Express server specifying values of the independent variables on each block and collating values for all the dependent variables.

The experiments were run within a student laboratory class, using the departmental intranet. Over 100 participants completed each experiment, normally in batches of up to 25 participants (using the 25 networked PCs in the lab). The participants generally took between 20 and 30 min to complete the experiment, and the progress of the entire class was monitored throughout the session by the instructors from a further networked PC.

The results of the experiments were interesting in their own right (see Cooper & Yule, 1999; Cooper et al., 2003; Yule et al., 1998), but for present purposes it is the successful use of the Express server as a harness for the experiment that is of primary interest. Most functions of the server were employed, including balancing of participant numbers in each between-subjects condition, sequence control, data collation, and analysis. The monitoring functions proved to be particularly useful, allowing the experimenter to see patterns in the data as they emerged and allowing the full dataset to be analyzed and presented to the class (using a standard PC projector attached to a PC showing the relevant Express administration page) as soon as the final participant completed the experiment.

Case Study 2: Harnessing a Computational Model

The use of Express in the evaluation of a computational model is illustrated by our work on modeling action selection and its breakdown following neurological damage (Cooper, Schwartz, Yule, & Shallice, 2003). The model consists of three interactive activation networks, with excitatory and inhibitory links between the networks. A number of parameters govern the activation dynamics within each network, and additional parameters govern the degree of interaction between the networks. The result is a model with over a dozen parameters, each

ranging in value from zero to one. (See Cooper & Shallice, 2000, for a detailed description of an earlier version of the model.) When the model is run, it generates a sequence of actions. The dependent variables relate to the integrity of that action sequence and include measures of the frequency of different types of action error, including, for example, omission errors (leaving out a required action) and perseverative errors (repeating an action or group of actions unnecessarily). The questions of interest concern the effects of various parameters on the profile of errors produced by the model and the relation of such error profiles to data generated by neurologically impaired individuals (see Schwartz et al., 1998).

The model was originally written in C as a stand-alone application, and initial evaluation used standard techniques (i.e., running the model over portions of the parameter space and collating the results manually). More recently, however, the API has been used to allow evaluation via the Express server. The dependence of the model on several different parameters, individually and in combination, has since been explored. Thus, one "experiment" varied the level of decay in all networks, with 20 different decay values and 50 replications for each value. A second experiment varied the ratio of top-down to bottom-up influences in the primary network over a similarly sized parameter space. Since this ratio is not a simple parameter of the model (it is a function of two complementary parameters) a pseudovalue was defined within the experiment's design file to allow manipulation of the ratio as if it were a parameter of the model. A third experiment varied four parameters simultaneously (levels of noise in two key networks and the levels of interaction between these two networks). In all cases, an additional independent variable—whether distractor objects were available within the model's simulated environment—was varied, allowing the model's behavior under various conditions to be compared with data from human participants under the two conditions (see Schwartz et al., 1998).

Our most extensive simulation experiments have involved over 15,000 runs of the model, with each run generating values for 10 dependent variables. These runs were conducted on a heterogeneous network of seven Linux processors (ranging from a 200-MHz Pentium Pro to two 1.3-GHz Athlon processors), and each was complete within 8 h. Previously, this kind of evaluation exercise would involve either running the model for several days or manually assigning different parts of the parameter space to different machines and manually collating the results from the various machines once all the machines had completed their assigned jobs. Express therefore greatly simplifies the model evaluation process.

A further benefit of Express in the evaluation of computational models relates to its mechanism for job assignment (i.e., for assigning new clients to between-subjects conditions). When a new client requests a PID, Express determines the set of conditions with the fewest complete jobs and assigns the new client to a random member of that set. Hence, in cases involving many between-subjects

conditions (i.e., large, possibly multidimensional parameter spaces), the conditions are filled in uniformly (rather than, e.g., from one corner of the parameter space to the opposite corner). The result is that a picture of model performance over the entire parameter space emerges relatively quickly. This can allow the model evaluator to abort an evaluation exercise relatively early in processing if it appears unpromising or, conversely, to extend the parameter space being scanned if the evaluation appears promising in some region. Our evaluation of the action selection model has employed both of these techniques.

The scale of these simulation experiments raised several performance-related issues. First, the experiments generated up to 5 MB of data each. With such large amounts of data, there is a noticeable slowing in server response times. However, Express was still able to generate all necessary tables of statistics within 15 sec. (This time is, of course, a function of the machine running the Web server—in our case, a Pentium III with dual 1-GHz processors, although a less powerful machine will suffice, such as the 200-MHz Pentium Pro we started with.) Second, since the individual jobs were small (requiring less than 5 sec of processor time on our fastest machines), and since multiple jobs were running in parallel, the Express server was required to handle frequent interactions. Often these interactions overlapped, with requests from multiple clients being served simultaneously. This stressed the abilities of the server to function in parallel, but the server was, in all cases, able to respond satisfactorily. Finally, there were occasional problems with clients failing (e.g., through memory management problems in the client code and network failure). Because data from each job are registered on the server when the job is completed, such failures led to the loss of data from, at most, one job per client (i.e., less than 10 sec of processing). In any case, Express's job allocation mechanism ensured that no "holes" in the scanning of the parameter space arose through such failed clients.

RESIDUAL ISSUES AND CONCLUDING REMARKS

We have presented Express, a Web-based experiment server for the coordination of both human and computational experiments. Express automatically manages between-subjects variable assignment and data collation and allows real-time monitoring of participant progress. In addition, it makes the parallel execution of computational experiments as easy as serial execution, largely overcoming the practical limitations traditionally associated with intensive model evaluation techniques. The server can provide these benefits, through the use of the API, to virtually any computational model in the cognitive sciences. In this final section, we consider several residual issues relating to the use of Express in both human and computational settings and its relation to other relevant software.

Comparative Modeling and Model Validation

We have seen that model parameters can be varied automatically to allow comparison of the same model across different sets of parameter settings. But it is also possible to run experiments comparing different models, by using the natural group variables facility; to do this, each model just needs to declare its identity (a level of a natural group variable) at assignment time. The same method can be used to compare the performance of human participants with one or more models. Again, each distinct model type corresponds to a level of the natural group variable, and *human* corresponds to another level.

Natural group variables, although not randomly assigned, are nonetheless frequently analyzed as ordinary between-subjects variables in psychological research. We merely suggest extending this practice to encompass a wider range of natural group variables applying to models. Such an approach could be used to identify significant differences in performance among models or between models and humans.

Security

Because Express is a Web-based system, it may be accessed using a standard Web browser from any machine connected to the Internet. This can be very convenient, since it allows the experimenter to access the data remotely and participants located at remote sites to complete the experiment. However, it also means that, if appropriate security measures are not taken, malicious (or merely curious) individuals may access an experiment, view sensitive data, and even possibly damage data.

In order to guard against this possibility, Express has been designed to support a range of security policies, via the Web server's access control mechanism. The simplest security policy is unrestricted access, allowing everyone access to read and modify all administration settings and data. This is clearly inadequate for most, if not all, purposes. Beyond this, an experimenter may adopt a range of policies, including unrestricted read access coupled with restricted write access and complete restriction of both read and write mechanisms. Because the client-server interaction script is separate from the administration system, they can be treated separately—for example, to allow unrestricted access to the experimental client while controlling access to the administration system.

JavaScript

Arguably, a possible weakness of the system is its reliance on JavaScript for client systems. Any Web-based client must make at least some use of JavaScript (or VBScript) to communicate with the Express server. Thus, clients cannot be completely developed in HTML. Although this makes client development more difficult, most clients will require JavaScript for their own purposes (e.g., form validation, timings). It therefore seems pointless to try to eliminate the use of JavaScript for client-server communication.

Buchanan and Reips (2001) have shown that a small minority of users, but especially those with a high educational level, tend to disable JavaScript in order to avoid pop-ups and other on-line irritations, so such people might be excluded from script-reliant experiments. However, scripting is enabled by default on modern browsers, and those who elect to disable it should be able to reenable it temporarily if this is required for an experiment. This issue, therefore, does not raise insuperable difficulties.

Availability and System Requirements

The Express server is currently available only for various flavors of UNIX (currently, Linux, Solaris, and MacOS X), running standard Web server software (e.g., Apache, but others should be usable). We are investigating the feasibility of porting the server to Microsoft Windows.

Access to the Express administration system requires a networked machine with a standard Web browser (e.g., Version 4 or higher of Netscape Navigator or Internet Explorer, on any platform). The administration system can run with JavaScript disabled on the administrator's Web browser if necessary, but with a less user-friendly interface and without design-editing capabilities.

Binary versions of the Express server (Version 1), as well as ANSI C source code and documentation for the client-side API, may be downloaded for free from the Express project Web site at <http://express.psyc.bbk.ac.uk/>.

CONCLUSIONS

We have argued that Express provides a general server facility to support experimentation with human and model participants. Unlike other Web-based human experimentation systems, and in order to maximize the flexibility of client design, Express makes no attempt to generate client-side code. It is, therefore, explicitly compatible with the use of nonbrowser-based clients, such as models. It also seems to be unique in supporting extensive statistical analysis of the collected data. Express provides the multiprocessing capability required for modeling, along with psychological experimental design principles, making it a good domain-specific solution for general psychological experimentation and model evaluation purposes.

REFERENCES

- ANDERSON, J. R., & LEBIERE, C. J. (1998). *The atomic components of thought*. Mahwah, NJ: Erlbaum.
- BERINGER, J. (1994). ERTS: A flexible software tool for developing and running psychological reaction time experiments on IBM PCs. *Behavior Research Methods, Instruments, & Computers*, **26**, 368-369.
- BIRNBAUM, M. H. (2000). SurveyWiz and FactorWiz: JavaScript Web pages that make HTML forms for research on the Internet. *Behavior Research Methods, Instruments, & Computers*, **32**, 339-346.
- BIRNBAUM, M. H., & WAKCHER, S. V. (2002). Web-based experiments controlled by JavaScript: An example from probability learning. *Behavior Research Methods, Instruments, & Computers*, **34**, 189-199.
- BUCHANAN, T., & REIPS, U.-D. (2001). Platform-dependent biases in online research: Do Mac users really think different? In K. J. Jonas, P. Breuer, B. Schauenburg, & M. Boos (Eds.), *Perspectives on Internet*

- research: Concepts and methods*. Available on line at <http://www.psych.uni-goettingen.de/congress/gor-2001/contrib/buchanan-tom>.
- COHEN, J., MACWHINNEY, B., FLATT, M., & PROVOST, J. (1993). PsyScope: An interactive graphic system for designing and controlling experiments in the psychology laboratory using Macintosh computers. *Behavior Research Methods, Instruments, & Computers*, **25**, 257-271.
- COOPER, R. P., SCHWARTZ, M., YULE, P., & SHALLICE, T. (2003). *The simulation of action disorganisation in complex activities of daily living*. Manuscript submitted for publication.
- COOPER, R. P., & SHALLICE, T. (2000). Contention scheduling and the control of routine activities. *Cognitive Neuropsychology*, **17**, 297-338.
- COOPER, R. P., & YULE, P. (1999). Comparative modelling of learning in a decision making task. In M. Hahn & S. C. Stoness (Eds.), *Proceedings of the Twenty-First Annual Conference of the Cognitive Science Society* (pp. 120-125). Mahwah, NJ: Erlbaum.
- COOPER, R. P., YULE, P., & FOX, J. (2003). Cue selection in category learning: A systematic comparison of three theories. *Cognitive Science Quarterly*, **3**, 143-182.
- HAXBY, J. V., PARASURAMAN, R., LALONDE, F., & ABBODD, H. (1993). SuperLab: General-purpose Macintosh software for human experimental psychology and psychological testing. *Behavior Research Methods, Instruments, & Computers*, **25**, 400-405.
- HEWSON, C., YULE, P., LAURENT, D., & VOGEL, C. (2003). *Internet research methods: A practical guide for the social and behavioural sciences*. London: Sage.
- MACWHINNEY, B., ST. JAMES, J., SCHUNN, C., LI, P., & SCHNEIDER, W. (2001). STEP—A System for Teaching Experimental Psychology using E-Prime. *Behavior Research Methods, Instruments, & Computers*, **33**, 287-296.
- PLAUT, D. C., & SHALLICE, T. (1993). Deep dyslexia: A case study of connectionist neuropsychology. *Cognitive Neuropsychology*, **10**, 377-500.
- REIPS, U.-D. (2001). The Web Experimental Psychology Lab: Five years of data collection on the Internet. *Behavior Research Methods, Instruments, & Computers*, **33**, 201-211.
- REIPS, U.-D., & NEUHAUS, C. (2002). WEXTOR: A Web-based tool for generating and visualizing experimental designs and procedures. *Behavior Research Methods, Instruments, & Computers*, **34**, 234-240.
- RITTER, F. E., BAXTER, G. D., JONES, G., & YOUNG, R. M. (2000). Supporting cognitive models as users. *ACM Transactions on Computer-Human Interaction*, **7**, 141-173.
- SCHMIDT, W. C. (1997). Word-Wide Web survey research: Benefits, potential problems, and solutions. *Behavior Research Methods, Instruments, & Computers*, **29**, 274-279.
- SCHWARTZ, M. F., MONTGOMERY, M. W., BUXBAUM, L. J., LEE, S. S., CAREW, T. G., COSLETT, H. B., FERRARO, M., FITZPATRICK-DE SALME, E. J., HART, T., & MAYER, N. H. (1998). Naturalistic action impairment in closed head injury. *Neuropsychology*, **12**, 13-28.
- YULE, P., & COOPER, R. P. (2001). Towards a technology for computational experimentation. In E. M. Altmann, A. Cleeremans, C. D. Schunn, & W. D. Gray (Eds.), *Proceedings of the Fourth International Conference on Cognitive Modeling* (pp. 223-228). Mahwah, NJ: Erlbaum.
- YULE, P., COOPER, R. P., & FOX, J. (1998). Normative and information processing accounts of medical diagnosis. In M. A. Gernsbacher & S. J. Derry (Eds.), *Proceedings of the Twentieth Annual Conference of the Cognitive Science Society* (pp. 1176-1181). Mahwah, NJ: Erlbaum.

NOTES

1. <http://www.genpsylab.unizh.ch/wextor/index.html>.
2. <http://psych.fullerton.edu/mbirnbaum/programs/>.
3. United Devices: <http://www.ud.com/>.
4. Grid forum: <http://www.gridforum.org/>.
5. See <http://www.pstnet.com/E-Prime/e-prime.htm>, <http://www.erts.de>, <http://www.superlab.com>, and <http://psyscope.psy.cmu.edu>, respectively.