gSRT-Soft: A generic software application and some methodological guidelines to investigate implicit learning through visual-motor sequential tasks

Stephanie Chambaron, Dominique Ginhac, Pierre Perruchet

Behavior Research Methods
2008, 40 (2), 493-502
doi: 10.3758/BRM.40.2.493

gSRT-Soft: A generic software application and some methodological guidelines to investigate implicit learning through visual–motor sequential tasks

Stéphanie Chambaron, Dominique Ginhac, and Pierre Perruchet
Université Libre de Bruxelles, Brussels, Belgium

Serial reaction time tasks and, more generally, the visual–motor sequential paradigms are increasingly popular tools in a variety of research domains, from studies on implicit learning in laboratory contexts to the assessment of residual learning capabilities of patients in clinical settings. A consequence of this success, however, is the increased variability in paradigms and the difficulty inherent in respecting the methodological principles that two decades of experimental investigations have made more and more stringent. The purpose of the present article is to address those problems. We present a user-friendly application that simplifies running classical experiments, but is flexible enough to permit a broad range of nonstandard manipulations for more specific objectives. Basic methodological guidelines are also provided, as are suggestions for using the software to explore unconventional directions of research. The most recent version of gSRT-Soft may be obtained for free by contacting the authors.

Implicit learning is usually defined as the process by which people learn without intent and without being able to clearly articulate what they are learning (for reviews, see Perruchet & Pacton, 2006; Shanks, 2005). The growing interest in implicit learning stems from its crucial role in the acquisition of one's mother language and in the development of other cognitive, social, and motor abilities. Another attractive feature of implicit learning is that it has proven to be relatively insensitive to age (e.g., Curran, 1997; D. V. Howard & J. H. Howard, 1989; Kotchoubey, Haisst, Daum, Schugens, & Birbaumer, 2000) and is preserved in a number of neuropsychological disorders (e.g., McDowall & Martin, 1996; Smith, Siegert, McDowall, & Abernethy, 2001; Stevens et al., 2002; Zillmer & Spiers, 2001). As a consequence, the phenomenon is a focus of investigation not only for laboratory researchers, but also for those oriented toward educational or clinical objectives.

Although several tasks have been used to investigate implicit learning (e.g., the artificial grammar learning task proposed by Reber, 1967, and the dynamic control task used by Berry & Broadbent, 1984), motor sequence-learning tasks are increasingly popular. In the most typical paradigm, coined the serial reaction time (SRT; Nissen & Bullemer, 1987) task, a target stimulus appears on successive trials at one of a limited number of positions. Participants are asked to react to the appearance of the target by pressing a key that spatially matches the location of the target on a keyboard. Unknown to participants, the sequence of events is not random. It usually consists of the continuous cycling of the same sequence. Learning is attested by the fact that reaction times (RTs) progressively decrease with practice of the repeated sequence and suddenly increase when a random sequence is unexpectedly inserted. This indicates that participants have acquired knowledge about the structured nature of the repeated sequence. However, even if it has been shown that participants demonstrate sequence learning, the debate about the nature of the acquired knowledge—implicit versus explicit—remains open. Consequently, various tests of awareness have been proposed for evaluating explicit and implicit sequence knowledge. First, Perruchet and Amorim (1992) developed a recognition task in which participants are presented some short sequences and have to discriminate between the sequences that follow the learned structure and the sequences that violate it. Subsequent studies have established that participants are able to recognize and discriminate correct sequences (Perruchet, Bigand, & Benoit-Gonin, 1997; Shanks, 2003; Shanks, Wilkinson, & Channon, 2003). Second, other researchers have proposed various generation tasks in which participants have to reproduce either the whole learned sequence or some fragments of it. In the case of the *free generation* task (Destrebecqz & Cleeremans, 2001; Perruchet & Amorim, 1992; Shanks & Johnstone, 1999), participants have to freely generate a sequence that is as similar as possible to the sequence learned during the training phase. This implies that participants have to remember a substantial amount of the structure of this learned sequence. On the contrary, in a *cued generation* task (Willingham, Nissen, & Bullemer, 1989), participants are presented ele-

S. Chambaron, schambar@ulb.ac.be

ments of the sequences and are instructed to press the button corresponding to where they think the next stimulus will appear. The stimulus remains present until the participant makes the correct response. A variation of this procedure is the *trial-by-trial sequence generation* task proposed by Wilkinson and Shanks (2004). Participants observe a short, five-element sequence of targets from the training sequence and then have to produce a single generation response corresponding to the correct continuation response. Norman, Price, Duff, and Mentzoni (2007) have proposed a novel generation task, the *generation rotation* task. This task is a modification of the existing trial-by-trial generation task, but with the addition of a randomly varying contextual cue. According to Norman et al., this task provides a more robust measure of explicit sequence knowledge.

Several reasons explain the success of the SRT paradigm. There is no doubt that sequential behavior is involved in virtually any world-wide ability, from language to the organization of movements, thus ensuring a good ecological validity to sequential tasks. The use of a visual–motor implementation makes a quantitative assessment of learning easy to get, and robust learning has proven to be possible within a short time in a large variety of populations, from children (Vinter & Perruchet, 2000) to elderly people (J. H. Howard & D. V. Howard, 1997). Another advantage of the SRT task over some other tasks of implicit learning is that participants are in truly incidental conditions of learning, because the effect of regularities can be assessed without participants having been informed about the presence of hidden regularities (Cleeremans, 1993; Destrebecqz & Cleeremans, 2001). Finally, it has been shown that the reliability of SRT tasks is pretty good in comparison with other tasks of implicit learning (Salthouse, McGuthry, & Hambrick, 1999). This property is essential when the aim of the researcher is to compare the learning abilities of different samples of participants and, moreover, when the residual learning abilities of patients need to be assessed on an individual basis.

A potential difficulty in the development of dedicated SRT task software is the large number of to-be-implemented procedures that are becoming increasingly complex. Indeed, since the initial study by Nissen and Bullemer (1987), SRT tasks have been the object of a large number of investigations that have led to both the emergence of a number of variants and the growing sophistication of methodological controls (e.g., Bischoff-Grethe, Goedert, Willingham, & Grafton, 2004; Chambaron, Ginhac, & Perruchet, 2006; Curran & Keele, 1993; Osman, Bird, & Heyes, 2005; Perruchet et al., 1997; Ziessler & Nattkemper, 2001).

Another possible obstacle against a still larger extension of SRT tasks, however, is the increasing difficulty of their implementation. SRT tasks require accurate time measurements, and this has been shown to be problematic in a multitask environment, such as Microsoft Windows. It has been recommended to use MS-DOS only, in which millisecond accuracy is possible (Myors, 1999). However, modern operating systems, such as Microsoft Windows XP, offer sophisticated mechanisms for recording time measures with submillisecond accuracy (more detailed explanations are given in the section dealing with records of RTs).

The purpose of the present article is to help researchers address all of these difficulties. We propose flexible, easy-to-use software and some guidelines for using it in accordance with up-to-date methodological principles.

OVERVIEW OF gSRT-SOFT

gSRT-Soft is a user-friendly application that makes it very easy to run classical SRT experiments, but is flexible enough to permit a broad range of nonstandard manipulations for more specific objectives. By default, the software has been designed to allow a straightforward and time-saving implementation of the most popular paradigms (e.g., a four-choice SRT task, with participants' responses being entered on the computer keyboard). Thus, it can be used, for instance, by a neuropsychologist wishing only to add a standard SRT task to a large-scale test battery of cognitive functions. However, it also integrates a number of features that enable users to implement nonstandard situations. For instance, the targets can be arranged in order to draw a labyrinth that covers all the bidimensional space of the screen, and a computer mouse or graphic tablet can be used as a manipulandum instead of the keyboard.

gSRT-Soft was developed in C++, which was chosen because it is a general-purpose programming language with high-level and low-level capabilities. The high-level capabilities allow easy development of large pieces of software, whereas the low-level capabilities allow the programmer to manage real-time aspects of the software, such as high-resolution measures of RTs.

The software is a Windows-based program using menus, buttons, and selection boxes accessible with a mouse. It includes two main menus—*Configuration* and *Participant*. Configuration determines the general parameters for a given experiment, which need to be set before the Participant menu can be accessed. First, the Configuration menu allows either the generation of a new configuration, the loading of an existing configuration stored on a file, or the saving of the current configuration to a file. These files are standard INI files frequently used for Microsoft Windows-based applications. In the second menu, the *New Participant* menu specifies the particular values used for each participant of a previously configured experiment.

Configuration

Figure 1 shows the available possibilities in the Configuration dialog box. The main choice concerns whether participants' responses are made on the keyboard or with a mouse. As anywhere else in the program, the user indicates his or her choice by clicking the appropriate radio button. When the keyboard option (default) is selected, two suboptions follow. Because an overwhelming proportion of SRT studies involve the same stimulus–response pattern (namely, four target locations and four spatially congruent response keys), this configuration is proposed as a default (the researcher can choose either the European AZERTY or the U.S. QWERTY keyboard). However, the configuration can be customized with regard to

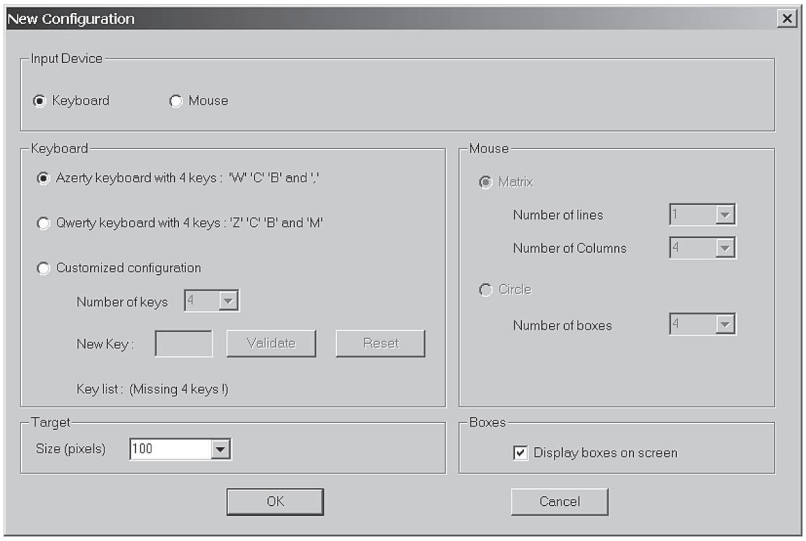

Figure 1. The New Configuration dialog box.

which keys match which target. The selection of the keys may serve several objectives. For instance, in most studies, there is a direct mapping between the location of the target on the screen and the spatial arrangement of the keys on the keyboard. Whether stimulus–response mapping is direct or not, however, has proven to be influential in a large number of motor control studies (e.g., Deroost & Soetens, 2006; Stöcker, Sebald, & Hoffmann, 2003). This software allows the exploration of the effect of this variable in SRT studies. It is possible, for instance, to pair the leftmost target with the rightmost key, or to design any other stimulus–response pairings.

During keyboard configuration, changing the number of keys is also possible, although this option may turn out to have limited utility. This is not because the nearly ubiquitous use of four target locations would be rooted in a principled advantage of this specific configuration; rather, it is due to the fact that using more than four keys causes a considerable slowing down of the responses and/or a dramatic increase in the error rate, for obvious reasons (for the use of six keys, see, e.g., Cleeremans, 1995; Heyes & Foster, 2002; Jiménez, Méndez, & Cleeremans, 1996). Limiting the number of possible events to four (or even six) can be damaging, restricting the ecological validity of the SRT task in a skill-learning context. Also, a sequence involving a larger number of different events allows the exploration of learning of much richer statistical regularities.

The use of a computer mouse is essentially aimed at overcoming the practical constraints linked to the use of a computer keyboard. The number of events is no longer limited, except by the spatial constraints linked to the positioning of the targets on a computer screen. In gSRT-Soft, the targets can be disposed either in a matrix (*X* rows and *Y* columns, maximum) or along a circle (*X* targets, maximum). The maximum values for rows and columns in a matrix layout or in a circle layout are dependent on screen resolution and target size. For example, with a standard XGA resolution (1,024 × 768 pixels) and a 100-pixel target size, the maximum number of rows is six and the maximum number of columns is nine. In the case of a circle layout, the maximum number of targets is 12. These maximum values are automatically evaluated in order to avoid an overlap between two consecutive locations on the screen.

Three different configurations (standard 4-position, 3 × 4 matrix, and 12-position circle) are depicted in Figure 2.

Note that with a matrix pattern, the sequence can be ordered in such a way that the target draws a labyrinth on the screen. A legitimate question, however, may be whether using a mouse recruits the same learning process as when keypressing is involved. To address this question,

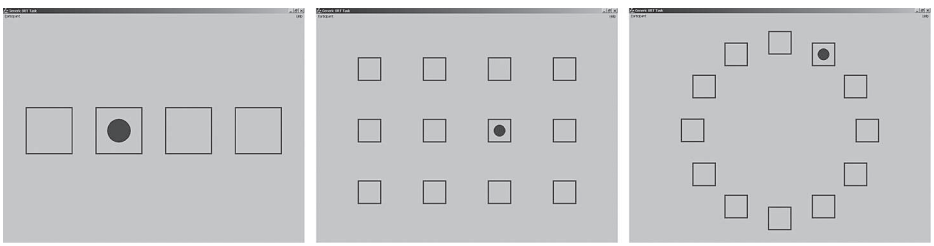

Figure 2. Different configurations for the targets.

we (Chambaron et al., 2006) have borrowed the design of a prior SRT study (Shanks, 2003), using either a keyboard or a mouse, as a function of participants. Although the mean RT was significantly larger with a mouse, evidence of learning was observed in both conditions, with no significant difference in the amount of learning. This result suggests that using a mouse may represent a promising alternative to the use of the keyboard whenever the researcher wishes to include more than a very few different targets in the trained sequence.

Finally, the size of the target needs to be defined. The default value is 100 pixels. However, the target can be reduced (to 1 pixel!) if the researcher wishes to degrade the perceptual discrimination of the stimuli—a procedure that may strengthen the role of location anticipation. Note that with the mouse option, a small target also increases motor accuracy constraints. The size of the target can also be increased when the opposite objectives are aimed. For obvious reasons, the maximum size depends on the number of targets and on their spatial configuration. When an illegal (i.e., too large) value is entered, the program returns an error message indicating the maximum value allowed, given the specific configuration.

By default, the possible locations of the targets are marked on the screen throughout the session by empty boxes in which the targets appear. However, an option can be selected in the Configuration menu, preventing these boxes from being displayed on the screen throughout the session.

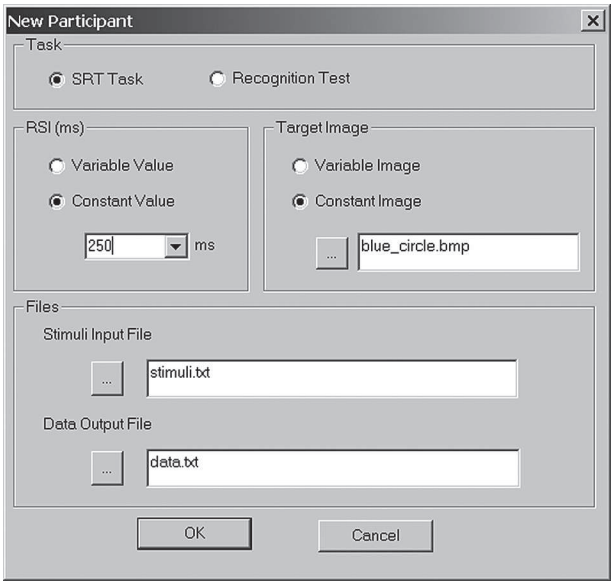

Figure 3. The New Participant dialog box.

New Participant

After the configuration is complete, clicking on the New Participant element menu opens the window displayed in Figure 3. The first choice concerns the task to be run. The label "SRT task" designates the main training task. This task may be followed by a recognition task. The general organization is the same for the two tasks.

The training sequences that a researcher may wish to explore may differ along a virtually unlimited number of features. They may be composed from the repetition of a sequence, but they also may be generated by a finite-state grammar (e.g., Cleeremans & McClelland, 1991) or other set of rules. They may be deterministic or probabilistic (see below), and they may differ in length and in a number of other parameters. This boundless variety makes it unmanageable to elaborate the sequence through an interactive set of options. The problem has been solved in gSRT-Soft by dissociating the generation of the sequence from the main program, with the consequence that the program reads only a previously prepared sequence of trials stored in the *stimuli input file*. A few typical sequences are included in the package (available on request). In most other cases, a standard spreadsheet, such as Microsoft Excel, prepares a set of original data quite well. For more sophisticated objectives, the researcher may use his or her preferred programming language, provided the resulting values are stored as a text file. In the case of a repetition of a sequence, the researcher only needs to generate a single text file that includes all the successive positions of the target. This text file will be used for all of the participants in the experiment. On the contrary, if the researcher designs individual, different blocks that vary randomly between participants, for example, the researcher must create one stimuli input file for each participant, and must include every block of stimuli in the desired order.

The stimuli input file must include a list of target locations. In the file, there is only one stimulus per line. The separator of the stimuli is the end-of-line (EOL) character. Instead of a stimulus, a line may include a "P" character, which indicates the presence of a self-paced pause. For instance, when the software reads the following 10 lines:

```
1
2
3
4
P
1
2
3
4
P
```

it generates two sequences, in which the target moves from the left to the right, and the two sequences are separated by a pause.

Moreover, concerning the recognition test, participants are classically presented with small sequence fragments and are asked to classify them as instances of the training material or not. In this case, the "R" character is inserted in the stimuli input file between each sequence to be recognized. The user decides the number of sequences to be recognized and the order in which these sequences are to be presented to the participants. In other words, the user has to prepare the set of sequences in the stimuli input file in such a way that "old" versus "new" sequences appear in a specific order. For instance, if we consider the sequence 1–2–3–4 to be the training sequence (i.e., *old* sequence), then the stimuli input file that reads as follows:

```
1
2
3
4
R
2
1
3
4
R
4
2
3
1
R
1
2
3
4
R
```

presents four sequences to be recognized in the following order: old, new, new, old. For simplicity, a two-alternative scale has been chosen in gSRT-Soft with default values for the question *Do you recognize this sequence?* and the two possible answers (*yes*, *no*). These default values can be modified by the user in the INI file. At present, gSRT-Soft does not allow recognition tasks based on gradual scales—for example, from 0 (*highly unfamiliar sequence*) to 10 (*highly familiar sequence*). Future releases may include such an option if this is requested.

The stimuli input file may be the same for all of the participants of a given experiment (note that by default, the program displays the path and the name of the last file that has been loaded, so this information does not need to be typed for each participant). However, in many cases, the stimuli input file differs from participant to participant, generally at the end of counterbalancing or randomizing.

An important parameter in the SRT task is the response–stimulus interval (RSI), which is the time that elapses between the participant's response (which, as a rule, triggers the suppression of the current target) and the onset of the next target. This value is usually kept constant throughout the session. A standard value of 250 msec is set by default in the software. However, it may be interesting to use variable RSIs (see below). In this case, the sequence of RSIs needs to be prepared. In the stimuli input file, each stimulus is followed by the associated RSI value. For instance, consider the following configuration:

```
1 0
2 500
3 0
4 0
P
1 0
2 500
3 0
4 0
P
```

There is no delay between Locations 1 and 2, nor between Locations 3 and 4, but there is a 500-msec interval between Locations 2 and 3.

Another important parameter in gSRT-Soft is the possibility of displaying any bitmap image on the screen as a target stimulus. Some basic bitmap files are distributed with the package, including geometric shapes (circle, square, triangle) in various colors (blue, yellow, red, black, etc.). However, any bitmap file can be used as a target. The image is automatically resized in order to be correctly displayed in the boxes on the screen. In the same manner as with the RSI value, it is possible to choose either a constant picture for all the stimuli or a variable picture associated to each stimulus. Consider the following example:

```
1 grey_circle.bmp
2 black_circle.bmp
3 yellow_circle.bmp
4 blue_circle.bmp
P
1 grey_triangle.bmp
2 black_triangle.bmp
3 yellow_triangle.bmp
4 blue_triangle.bmp
P
```

In the first sequence 1–2–3–4, a circle of different colors (grey, black, yellow, blue) is used, whereas in the second sequence, a triangle is displayed.

This option is innovative, making gSRT-Soft flexible enough to allow the programming of novel SRT procedures. We can imagine realizing SRT tasks with sequences of letters, words, or pictures. Such a possibility is used in the serial naming task developed by Goschke and Bolte (2007) in order to investigate implicit learning of repeated sequences of abstract semantic categories. In this task, participants have to name pictures of objects displayed on a screen (e.g., table, shirt, etc.). Unknown to the participants, the semantic categories of the objects (e.g., furniture, clothing) follow a repeating sequence.

For obvious reasons, we envision generating some sequences based on variable RSI values and variable target images. In such a case, the stimuli input file must be written in the manner of the following:

```
1 0 grey_circle.bmp
2 500 black_circle.bmp
3 0 yellow_circle.bmp
4 0 blue_circle.bmp
P
1 0 grey_triangle.bmp
2 500 black_triangle.bmp
3 0 yellow_triangle.bmp
4 0 blue_triangle.bmp
P
```

However, the present release of gSRT-Soft does not allow the display of multiple stimulus displays at the same time, as was seen in the procedure used by Norman et al. (2007). In their experiments, the four possible target positions were indicated by four shape outlines (heart, circle, square, cross) colored red, green, or blue on a white background. On each

trial, one shape was filled solid with the same color as its outline, and the other three remained unfilled. The filled shape was the target stimulus. Such an option will be considered with much attention by gSRT-Soft developers for future releases if enough researchers request it.

Before running the experiment, the path and the name of the data output file need to be specified. If this information is kept unchanged from one participant to the next, the data will be appended in the specified file. First, the file contains various information, such as the name and version number of the software, the date, and the time the task was performed. Moreover, the configuration parameters of the experiment are stored: input device, configuration of the input device, number of boxes, and size of the target. Then, the data about the participant are saved. They include the nature of the task, the name of input and output files, the type of RSI, and the target images. Finally, in the case of the training phase, the collected data are presented in a matrix comprising, on each row, the index of the trial, the location of the stimulus for a trial, the participant's answer, a binary value in the *good* column indicating whether this response is correct (1) or not (0), the RTs in milliseconds, the RSI (if variable), and the target image (if variable). The example matrix in Figure 4 includes eight rows corresponding to the eight trials previously described. Between the fourth and the fifth trial, a pause is inserted, and no data is saved in this case.

In the case of the recognition task, two different kinds of data are automatically recorded. First, RTs are stored in a matrix similar to the one previously described for the practice phase. Second, for each sequence to be recognized, the response given by the participant is stored in the output file.

As depicted in Figure 4, RTs are stored in milliseconds, with a high degree of accuracy. High-resolution timing is supported in modern operating systems (such as Microsoft Windows XP) that offer sophisticated mechanisms for recording time measures at submillisecond accuracy. In our case, high-resolution timing is supported by the two main following functions: *QueryPerformanceCounter* and *QueryPerformanceFrequency*. The first call, QueryPerformanceCounter, returns the amount of time that has elapsed since the system was booted. This amount of time is expressed in ticks of the processor clock. The second function, QueryPerformanceFrequency, returns the frequency of the processor clock. To retrieve the elapsed time of a code section, one has to get the actual value of the high-resolution performance counter immediately before and immediately after the section of code to be timed. The difference of these values would indicate the number of clock ticks that had elapsed while the code executed. The elapsed time can be computed then, by dividing this difference by the frequency of the processor. Such a method allows gSRT-Soft to record RTs with very high accuracy.

SOME METHODOLOGICAL GUIDELINES

The remainder of this article outlines some basic methodological principles for making the best use of the software. We focus on the most standard SRT paradigms, which are also those on which our methodological knowledge is the most developed. To give a first hint about the difficulties inherent in planning an SRT experiment, let us consider the seminal study by Nissen and Bullemer (1987), in which the same 10-element sequence (4–2–4–1–3–2–4–3–2–1) was continuously repeated. Note that this sequence includes Locations 2 and 3 three times, and Locations 1 and 4 two times. This raw frequency information can be used by participants to improve their performance (Shanks, 2003). Moreover, even if one considers that sequential information is learned, it remains difficult to gain more knowledge about the learning capabilities of participants. An essential question about sequence learning is whether participants take into account the information provided by the immediately preceding event, by the two prior events, or by still higher order information. In the sequence used by Nissen and Bullemer, Location 3 allows one to predict Location 2, and thus it is possible that improved performance simply reflects knowledge of the first-order dependency 3–2. On other parts of the sequence, however, predicting the next trial requires considering at least two, and occasionally three, successive events (predicting the event following 3–2 implies considering whether the prior location is 1 or 4). The sequence makes it impossible to know whether participants actually learn more than first-order dependency rules.

The Order of the Dependency Rules

The example above makes it obvious that a proper assessment of what participants learn requires that the order of the dependency rules be homogeneous throughout the sequence, although a few authors have used hybrid sequences embedding relations of different order (e.g., Cohen, Ivry, & Keele, 1990). Many well-controlled studies now employ a 12-element sequence of targets known as the second-order

Index	Stimuli	Answer	Good	RT	RSI	Target
1	1	1	1	838.247	0	grey_circle.bmp
2	2	2	1	330.454	500	black_circle.bmp
3	3	3	1	598.474	0	yellow_circle.bmp
4	4	4	1	750.548	0	blue_circle.bmp
5	1	1	1	794.300	0	grey_triangle.bmp
6	2	2	1	362.484	500	black_triangle.bmp
7	3	3	1	686.425	0	yellow_triangle.bmp
8	4	4	1	694.526	0	blue_triangle.bmp

Figure 4. Example of a data output file.

conditional (SOC) sequence (Reed & Johnson, 1994). An example of the SOC sequence is 3–1–4–3–2–4–2–1–3–4–1–2 (Shanks, 2003). In this sequence, participants cannot learn to predict the next element on the basis of its raw frequency, because each location occurs an equal number of times (3). Neither can they predict the next element on the basis of the immediately preceding element, because all of the possible successions occur equally (e.g., 2 may be followed by 1, 3, and 4; note that a ubiquitous constraint in SRT tasks is that the target does not appear in the same location on two successive trials). In SOC sequences, two prior elements of context (prior locations) are required in order to fully predict the next target. For instance, 4–2 is always followed by 3, and, in addition, 3 is always preceded by 4–2. SOC sequences are now prevalent; some studies have used first-order conditional (FOC) sequences, in which the nature of any element is fully predicted by a single prior element (e.g., Schvaneveldt & Gomez, 1998, Experiment 1). Note that those sequences are especially easy to learn. Indeed, it is sufficient to learn pairwise relations, and, moreover, the length of the sequence is limited to the number of possible locations. For instance, with four locations, only four-element FOC sequences are possible (e.g., 2–4–3–1).

Generating a Transfer Sequence

The prior section concerned the choice of the to-be-learned sequence. At first glance, examining whether participants' speed improves throughout the session could provide a reliable measure of sequence learning. This is not the case, however, because performance improvement may be due to other factors—for instance, nonspecific familiarization with the task. In order to reveal learning, RTs on the trained sequence need to be compared with RTs on another sequence. This other sequence may be randomly generated. However, the procedure is not optimal, because it is possible for a random sequence to share some of the features displayed in the to-be-learned sequence. The best method consists of carefully selecting another sequence, called the *transfer sequence*. Considering the SOC sequence above (hereafter SOC1), a convenient transfer sequence (hereafter SOC2) would be 4–3–1–2–4–1–3–2–1–4–2–3 (Shanks, 2003; Wilkinson & Shanks, 2004). Worthy of note, SOC1 and SOC2 differ only by their second-order transitions (note that they are also equal with respect to other potentially influential features, such as the number of back-and-forth movements). Thus, comparing RTs on these two sequences should provide a reliable measure of whether participants have learned second-order dependency rules. An additional precaution consists of counterbalancing SOC1 and SOC2—half of the participants being trained with SOC1 (and tested with SOC2), and the other half being trained with SOC2 (and tested with SOC1)—in order to cancel the effect of a possible difference in difficulty between the two sequences. The principles described here for SOC sequences are obviously generalizable to any other sequences.

When Should the Transfer Sequence Be Introduced?

In Nissen and Bullemer (1987), a separate group of participants was trained with random sequences. Beyond the need for a large number of participants, a between-participants design is ill suited to neuropsychological investigations, in which an individual assessment of learning ability is often desirable. Most recent studies use a within-participants design. However, they often differ with regard to the moment at which the transfer sequence is displayed during the training session.

The most standard procedure consists in showing the transfer sequence toward the end of the training session. Assuming that the whole session is divided into n blocks, the transfer sequence may be introduced on block $n-1$. The RT average on the transfer block is then compared with the RT average on the surrounding blocks (block $n-2$ and block n). Learning is shown by the presence of a selective increase of RTs (or error rate) on block $n-1$ in relation to RTs on the surrounding blocks. Although widespread, this method is limited by the fact that it provides no indication of the time course of learning, hence making results heavily dependent on the (largely arbitrary) choice of n. It is quite possible to imagine that a continuous measure of learning would have revealed differences in the learning curves of two groups of participants who are found to perform at the same level on a final test.

When learning curves appear desirable, displaying the transfer sequence (or parts of it) throughout the training session provides a solution. In a few studies (e.g., Meulemans, Van der Linden, & Perruchet, 1998), a random sequence is intercalated between successive occurrences of the to-be-learned sequence. However, this method requires the use of an ever-changing random sequence to prevent learning of the transfer sequence; we saw above that this choice is not optimal for assessing the exact content of learning. In another method, developed by Schvaneveldt and Gomez (1998), an element of the repeated sequence is randomly substituted with an element of the transfer sequence. Using the SOC1 and SOC2 sequences reported above as training and transfer sequence, respectively, a sample of the final sequence may be, for instance, 3–1–4–*2*–1–3–4–*3*–2–4–2–1– . . . , in which the italicized locations are transfer elements (i.e., they respect the second-order dependency rules of SOC2, whereas the other locations respect the rules of SOC1). Averaging—for each block of trials—the RTs on regular elements on the one hand and the RTs on transfer elements on the other allows us to obtain two separate curves, the difference of which provides evidence of learning.

It is worth stressing that dispatching transfer items within the trained sequence not only opens a window on the level of learning reached at those points, but also changes the to-be-learned material. Instead of being continuously cycled in deterministic ways, the sequence becomes probabilistic. Assuming that 10% of the transfer elements have been randomly introduced, the location of the next target at any point in the sequence can be predicted only with a probability of .90. The few available studies that use both deterministic and probabilistic sequences (e.g., Shanks, Channon, Wilkinson, & Curran, 2006) show that learning occurs for both types of sequences. Nevertheless, unsurprisingly, learning probabilistic sequences appears more difficult. The level of difficulty obviously

depends on the proportion of transfer elements introduced during training. For instance, Schvaneveldt and Gomez (1998) reported that they had failed to obtain a reliable performance improvement with 20% of transfer trials with SOC sequences, but that they were successful with 10%. Shanks et al. (2006) reported quick learning of SOC sequences with 15% of transfer elements—even in amnesic patients. The fact that the repetition structure is less salient is often interpreted as an advantage, with the idea that the difficulty of detecting the repetition structure may disrupt the explicit mode of learning (e.g., Shanks et al., 2006). An additional advantage of probabilistic sequences is that they may be more representative of sequential events in the real world (Hunt & Aslin, 2001).

The Number of Trials per Block and the Number of Blocks

As a rule, the whole training session is divided into blocks of about 100 trials separated by self-paced pauses. The exact number of trials per block often depends on particularities of the repeated sequence. For instance, with 12-element sequences, the length of the block may be set to 96, in order to include eight full sequences. It is also possible to add a few random trials at the beginning of each block, in order to make the repetition structure less salient. The number of blocks is more difficult to select. It depends on the nature of the repeated sequence, the sample of participants, the objective of the researchers, and so on. However, it is worth stressing that studies using a small number of blocks (e.g., Perruchet et al., 1997) or using probabilistic sequences that allow the elaboration of learning curves (e.g., Shanks et al., 2006) have revealed that learning emerges very early during the session, and often does not improve with further practice. Of course, this does not mean that further practice does not induce any changes. It is possible, for instance, that performance in subsequent tests of explicit knowledge depends on the amount of practice. However, if the main objective of the researchers is to give evidence of implicit learning through RT measures, using only a few blocks of trials may be sufficient. For instance, with SOC sequences, three or four 100-trial blocks appear sufficient for getting evidence of reliable learning.

The Response–Stimulus Interval (RSI)

Most SRT studies have used RSIs around 200 or 250 msec. However, a few studies have used longer intervals. For instance, Frensch, Buchner, and Lin (1994) used RSIs of up to 1,500 msec and still observed learning. On the other hand, a few studies have used a 0-msec RSI, with various outcomes. Perruchet et al. (1997) reported no RT improvement with intact explicit knowledge; Destrebecqz and Cleeremans (2001) reported an exactly inverse pattern; and Shanks et al. (2006) reported improved performance both on RTs and on tests of explicit learning. Because these empirical results do not provide us with reliable guidelines for a choice, it may be better to start from theoretical considerations. It may be thought that a long RSI gives participants the opportunity of elaborating conscious strategies, rehearsing the sequence, or engaging in other controlled activities that are generally construed as being undesirable in the context of implicit learning studies. The choice of RSI values depends on the purpose of the experiment. Indeed, with gSRT-Soft, researchers interested in comparing performance patterns across different levels of awareness have the capability of using a between-participants RSI manipulation.

The Tests of Explicit Knowledge

SRT studies have often been used to compare motor performance in the incidental training task—often construed to be a measure of implicit knowledge—with performance in a subsequent task aimed at capturing participants' explicit knowledge about the structure of the training materials. There is consensus that the test of explicit learning needs to be as sensitive as possible in order to reveal even fleeting evidence of explicit knowledge (see Shanks & St. John, 1994, about the sensitivity criterion). In this regard, a yes/no recognition test appears to be a reasonable choice. Small parts of the repeated sequence, intermixed with small parts of the transfer sequence, are displayed to participants, who have to respond to the target just as they had in the training phase. Note that the resulting RTs provide an additional measure of motor performance that does not necessarily converge with performance data collected during the training session (see Shanks et al., 2006). The length of the to-be-recognized sub-sequences generally comprises between three and six trials. After each sub-sequence, participants are asked to judge whether they had seen the target during the training session.

Various generation tasks are also used to assess the amount of explicit knowledge. In this type of task, the relationship between the target appearance and participants' responses is reversed in such a way that participants' keypresses (or, alternatively, clicks in a target, when the mouse option is selected) elicit the appearance of the selected target on the screen. In free generation tasks, participants are asked to reproduce at best the sequence they saw during the training session. This task may be thought of as a recall task adapted to the case of sequential materials (for a detailed analysis of the resulting data, see Perruchet & Amorim, 1992). However, other instructions are possible. Participants may be told to generate the first sequence that comes to mind, and, in this case, the generation task becomes a nominally implicit task. In the case of cued or trial-by-trial generation tasks, additional cues are given to the participants, whose task is to produce the correct continuation response in the sequence. If performance is at chance, this indicates that participants have learned implicitly. However, there is much evidence that participants are able to generate the learned sequence, suggesting that learning requires at least some explicit knowledge (Perruchet & Amorim, 1992; Shanks & Johnstone, 1999).

The use of this kind of generation task does not show the researcher whether good performance is due only to explicit knowledge or is partly mediated by implicit knowledge. In order to measure the respective contributions of implicit and explicit knowledge, the process dissociation procedure (PDP) initially implemented by Jacoby (1991) can be used. By varying the instructions given to partici-

pants, the procedure is aimed at dissociating conscious and automatic influences. In the inclusion task, participants are asked to generate the learned sequence by remembering it as clearly as possible. The exclusion task, however, requires participants to avoid generating the regularities embedded in the learned sequence. Destrebecqz and Cleeremans (2001, 2003) were the first to conduct a series of experiments applying the PDP to SRT tasks. They demonstrated that, under certain circumstances, participants could not avoid reproducing the learned sequence, despite having been instructed not to do so.

Generation tasks are not implemented in the current release of gSRT-Soft. Nevertheless, actual development of the gSRT-Soft focuses on the implementation of a free generation task and a trial-by-trial generation task. Moreover, these two tasks could be used with inclusion and exclusion instructions. In the case of the free generation task, participants will be instructed to freely generate the learned sequence (inclusion condition) or to generate a different sequence (exclusion condition). In the case of the trial-by-trial generation task, participants will be asked to respond to a five-element sequence and to produce the sixth element. In the inclusion condition, participants are instructed by a green question mark displayed on the screen to produce the next element of the training sequence. In the exclusion condition, participants have to generate a different continuation response. In this case, a red question mark will appear on the screen.

Further releases of gSRT-Soft that include these new features will soon be available.

SUMMARY

In short, the gSRT-Soft should allow the user to run sequence-learning experiments quite easily and with excellent time accuracy. Obviously, the value of the results will depend on the care with which each researcher plans his or her research. We have provided some guidelines for designing experiments involving standard paradigms. There is no doubt that SRT studies, and learning studies in general, raise tricky methodological problems when new issues are explored. We hope that the capabilities of the software will encourage researchers to investigate unexplored directions of research in learning.

Evaluation copies of gSRT-Soft may be obtained free of charge by contacting the authors. Future releases of gSRT-Soft will be published as free software under the terms of the general public license (see www.gnu.org for more details about the GPL). Binary software and source code of the SRT task software will be available from a dedicated Web site. The authors of gSRT-Soft ask that publications involving the use of the original or modified versions of the software cite this article.

AUTHOR NOTE

This work was supported by the Centre National de la Recherche Scientifique (CNRS, UMR 5158 and UMR 5022), the Université de Bourgogne (LE2I and LEAD), and the Region de Bourgogne (AAFE). The authors also thank Axel Cleeremans and the anonymous reviewers for their help at various stages of elaboration. Correspondence concerning this article should be addressed to S. Chambaron, Cognitive Science Research Unit, Université Libre de Bruxelles, Av. F. D. Roosevelt, 50, CP 191, 1050 Brussels, Belgium (e-mail: schambar@ulb.ac.be).

REFERENCES

Berry, D. C., & Broadbent, D. E. (1984). On the relationship between task performance and associated verbalizable knowledge. *Quarterly Journal of Experimental Psychology*, **36A**, 209-231.

Bischoff-Grethe, A., Goedert, K. M., Willingham, D. T., & Grafton, S. T. (2004). Neural substrates of response-based sequence learning using fMRI. *Journal of Cognitive Neuroscience*, **16**, 127-138.

Chambaron, S., Ginhac, D., & Perruchet, P. (2006). Is learning in SRT tasks robust across procedural variations? In R. Sun & N. Miyake (Eds.), *Proceedings of the 28th Annual Conference of the Cognitive Science Society* (pp. 148-153). Mahwah, NJ: Erlbaum.

Cleeremans, A. (1993). Attention and awareness in sequence learning. In *Proceedings of the 15th Annual Conference of the Cognitive Science Society* (pp. 330-335). Hillsdale, NJ: Erlbaum.

Cleeremans, A. (1995). Implicit learning in the presence of multiple cues. In *Proceedings of the 17th Annual Conference of the Cognitive Science Society* (pp. 298-303). Mahwah, NJ: Erlbaum.

Cleeremans, A., & McClelland, J. L. (1991). Learning the structure of event sequences. *Journal of Experimental Psychology: General*, **120**, 235-253.

Cohen, A., Ivry, R. I., & Keele, S. W. (1990). Attention and structure in sequence learning. *Journal of Experimental Psychology: Learning, Memory, & Cognition*, **16**, 17-30.

Curran, T. (1997). Effects of aging on implicit sequence learning: Accounting for sequence structure and explicit knowledge. *Psychological Research*, **60**, 24-41.

Curran, T., & Keele, S. W. (1993). Attentional and nonattentional forms of sequence learning. *Journal of Experimental Psychology: Learning, Memory, & Cognition*, **19**, 189-202.

Deroost, N., & Soetens, E. (2006). Spatial processing and perceptual sequence learning in SRT tasks. *Experimental Psychology*, **53**, 16-30.

Destrebecqz, A., & Cleeremans, A. (2001). Can sequence learning be implicit? New evidence with the process dissociation procedure. *Psychonomic Bulletin & Review*, **8**, 343-350.

Destrebecqz, A., & Cleeremans, A. (2003). Temporal effects in sequence learning. In L. Jiménez (Ed.), *Attention and implicit learning* (pp. 181-213). Amsterdam: John Benjamins.

Frensch, P. A., Buchner, A., & Lin, J. (1994). Implicit learning of unique and ambiguous serial transitions in the presence and absence of a distractor task. *Journal of Experimental Psychology: Learning, Memory, & Cognition*, **20**, 567-584.

Goschke, T., & Bolte, A. (2007). Implicit learning of semantic category sequences: Response-independent acquisition of abstract sequential regularities. *Journal of Experimental Psychology: Learning, Memory, & Cognition*, **33**, 394-406.

Heyes, C. M., & Foster, C. L. (2002). Motor learning by observation: Evidence from a serial reaction time task. *Quarterly Journal of Experimental Psychology*, **55A**, 593-607.

Howard, D. V., & Howard, J. H., Jr. (1989). Age differences in learning serial patterns: Direct versus indirect measures. *Psychology & Aging*, **4**, 357-364.

Howard, J. H., Jr., & Howard, D. V. (1997). Age differences in implicit learning of higher order dependencies in serial patterns. *Psychology & Aging*, **12**, 634-656.

Hunt, R. H., & Aslin, R. N. (2001). Statistical learning in a serial reaction time task: Access to separable statistical cues by individual learners. *Journal of Experimental Psychology: General*, **130**, 658-680.

Jacoby, L. L. (1991). A process dissociation framework: Separating automatic from intentional uses of memory. *Journal of Memory & Language*, **30**, 513-541.

Jiménez, L., Méndez, C., & Cleeremans, A. (1996). Comparing direct and indirect measures of sequence learning. *Journal of Experimental Psychology: Learning, Memory, & Cognition*, **22**, 948-969.

Kotchoubey, B., Haisst, S., Daum, I., Schugens, M., & Birbaumer, N. (2000). Learning and self-regulation of slow cortical potentials in older adults. *Experimental Aging Research*, **26**, 15-35.

McDowall, J., & Martin, S. (1996). Implicit learning in closed head

injured subjects: Evidence from an event sequence learning task. *New Zealand Journal of Psychology*, **25**, 1-6.

MEULEMANS, T., VAN DER LINDEN, M., & PERRUCHET, P. (1998). Implicit sequence learning in children. *Journal of Experimental Child Psychology*, **69**, 199-221.

MYORS, B. (1999). Timing accuracy of PC programs running under DOS and Windows. *Behavior Research Methods, Instruments, & Computers*, **31**, 322-328.

NISSEN, M. J., & BULLEMER, P. (1987). Attentional requirements of learning: Evidence from performance measures. *Cognitive Psychology*, **19**, 1-32.

NORMAN, E., PRICE, M. C., DUFF, S. C., & MENTZONI, R. A. (2007). Gradations of awareness in a modified sequence learning task. *Consciousness & Cognition*, **16**, 809-837.

OSMAN, M., BIRD, G., & HEYES, C. (2005). Action observation supports effector-dependent learning of finger movement sequences. *Experimental Brain Research*, **165**, 19-27.

PERRUCHET, P., & AMORIM, M.-A. (1992). Conscious knowledge and changes in performance in sequence learning: Evidence against dissociation. *Journal of Experimental Psychology: Learning, Memory, & Cognition*, **18**, 785-800.

PERRUCHET, P., BIGAND, E., & BENOIT-GONIN, F. (1997). The emergence of explicit knowledge during the early phase of learning in sequential reaction time tasks. *Psychological Research*, **60**, 4-13.

PERRUCHET, P., & PACTON, S. (2006). Implicit learning and statistical learning: One phenomenon, two approaches. *Trends in Cognitive Sciences*, **10**, 233-238.

REBER, A. S. (1967). Implicit learning of artificial grammars. *Journal of Verbal Learning & Verbal Behavior*, **5**, 855-863.

REED, J., & JOHNSON, P. (1994). Assessing implicit learning with indirect tests: Determining what is learned about sequence structure. *Journal of Experimental Psychology: Learning, Memory, & Cognition*, **20**, 585-594.

SALTHOUSE, T. A., MCGUTHRY, K. E., & HAMBRICK, D. Z. (1999). A framework for analyzing and interpreting differential aging patterns: Application to three measures of implicit learning. *Aging, Neuropsychology, & Cognition*, **6**, 1-18.

SCHVANEVELDT, R. W., & GOMEZ, R. L. (1998). Attention and probabilistic sequence learning. *Psychological Research*, **61**, 175-190.

SHANKS, D. R. (2003). Attention and awareness in "implicit" sequence learning. In L. Jiménez (Ed.), *Attention and implicit learning* (pp. 11-42). Amsterdam: John Benjamins.

SHANKS, D. R. (2005). Implicit learning. In K. Lamberts & R. Goldstone (Eds.), *Handbook of cognition* (pp. 202-220). London: Sage.

SHANKS, D. R., CHANNON, S., WILKINSON, L., & CURRAN, H. V. (2006). Disruption of sequential priming in organic and pharmacological amnesia: A role for the medial temporal lobes in implicit contextual learning. *Neuropsychopharmacology*, **31**, 1768-1776.

SHANKS, D. R., & JOHNSTONE, T. (1999). Evaluating the relationship between explicit and implicit knowledge in a sequential reaction time task. *Journal of Experimental Psychology: Learning, Memory, & Cognition*, **25**, 1435-1451.

SHANKS, D. R., & ST. JOHN, M. F. (1994). Characteristics of dissociable human learning systems. *Behavioral & Brain Sciences*, **17**, 367-447.

SHANKS, D. R., WILKINSON, L., & CHANNON, S. (2003). Relationship between priming and recognition in deterministic and probabilistic sequence learning. *Journal of Experimental Psychology: Learning, Memory, & Cognition*, **29**, 248-261.

SMITH, J., SIEGERT, R. J., MCDOWALL, J., & ABERNETHY, D. (2001). Preserved implicit learning on both the serial reaction time task and artificial grammar in patients with Parkinson's disease. *Brain & Cognition*, **45**, 378-391.

STEVENS, A., SCHWARZ, J., SCHWARZ, B., RUF, I., KOLTER, T., & CZEKALLA, J. (2002). Implicit and explicit learning in schizophrenics treated with olanzapine and with classic neuroleptics. *Psychopharmacology*, **160**, 299-306.

STÖCKER, C., SEBALD, A., & HOFFMANN, J. (2003). The influence of response-effect compatibility in a serial reaction time task. *Quarterly Journal of Experimental Psychology*, **56A**, 685-703.

VINTER, A., & PERRUCHET, P. (2000). Implicit learning in children is not related to age: Evidence from drawing behavior. *Child Development*, **71**, 1223-1240.

WILKINSON, L., & SHANKS, D. R. (2004). Intentional control and implicit sequence learning. *Journal of Experimental Psychology: Learning, Memory, & Cognition*, **30**, 354-369.

WILLINGHAM, D. B., NISSEN, M. J., & BULLEMER, P. (1989). On the development of procedural knowledge. *Journal of Experimental Psychology: Learning, Memory, & Cognition*, **15**, 1047-1060.

ZIESSLER, M., & NATTKEMPER, D. (2001). Learning of event sequences is based on response-effect learning: Further evidence from a serial reaction task. *Journal of Experimental Psychology: Learning, Memory, & Cognition*, **27**, 595-613.

ZILLMER, E. A., & SPIERS, M. V. (2001). *Principles of neuropsychology*. Belmont, CA: Wadsworth.

(Manuscript received May 9, 2007;
revision accepted for publication December 21, 2007.)