PSY 1401 Final Project
May 2, 2024
Jack Gabel, Sol Markman, Quilee Simeon, and Hokyung Sung

Chunking Behavior in the Simon Game

Abstract

The Simon Game is a classic children's toy in which users reproduce sequences of colors and sounds. Humans can perform remarkably long sequences in this game, with a world record of 84 steps. How is it possible to have such expansive working memory capacity in the Simon Game? We hypothesize that subjects 'chunk' the colors into patterns like 'loops' or 'repeats' which effectively allow them to expand their working memory capacity. In this study, we test the chunking hypothesis and seek to characterize the particular chunking algorithm used by subjects. We also implemented neural models that aim to formalize how chunks may be encoded in the brain. Two models leverage persistent-activity while our neural process model utilizes cognitive control and short-term synaptic plasticity to learn and reproduce sequences. The neural process model uses neural dynamics to encode transitions within the sequence, which may help enable high scores in the task. Overall, this work aims to shed light on the cognitive strategies and possible neural mechanisms underlying our subjects' impressive performance in the Simon Game.

Intro

Background on Simon Game

The Simon Game was patented in 1980 and remains a beloved children's toy. The device has a ring of four buttons that each have a particular tune and color associated with them. In a 'Simon says' style, a sequence of buttons is presented and the user responds by clicking the buttons to reproduce the sequence. Each turn, the sequence gains an additional button until the user makes a mistake and the game restarts. For example, the sequence will start out with one button (e.g. red). Then two buttons (red, blue). Followed by three buttons (red, blue, yellow), and so on.

Figure 1: A picture of the original Simon Game

In playing the game, we have noticed that users reproduce strikingly long sequences. For example, one researcher on this project has reached 35 steps! This capacity stands in stark contrast to traditional models of working memory which propose a much smaller capacity (Miller 1956). How can we reconcile human behavior in the Simon Game with the existing literature? We hypothesize that humans 'chunk' together patterns in the sequence, thus reducing the working memory load.

Chunking

We define chunking as a process of grouping data together according to specific patterns to reduce the description length of the data. The description length is the size of the code needed to store the data in memory. In this project, we aim to determine whether subjects chunk in the Simon Game. We specifically aim to identify *how* they chunk. For example, do they chunk together repeating colors? Circular loops of colors? Or other more complicated cognitive strategies? Finally, we aim to investigate how chunking might be implemented in the brain.

Evidence for Chunking

Multiple behavioral studies have suggested that human subjects chunk. For example, Mathy and Feldman (2011) asked subjects to remember sequential patterns of various lengths and compressibilities. They showed that subjects tend to remember more steps in the sequence for more compressible sequences. Amalric et al. (2017) also showed that humans chunk. They showed subjects short spatial patterns and asked them to predict future locations (i.e. complete the pattern). By analyzing errors, they found that subjects utilize symmetric and rotational patterns in the data.

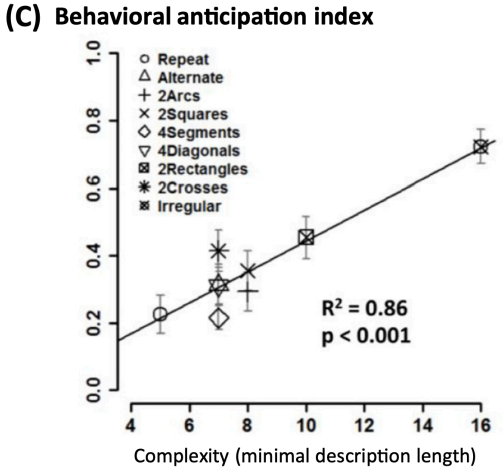


Figure 2: This figure shows the Mathy and Feldman (2011) data. Errors are plotted as a function of sequence complexity. Figure is from Dehaene et al. (2022).

There is also some tangential neural evidence for chunking. Shima et al. (2007) trained monkeys to produce four-step motor sequences. They found neurons tuned to sequential patterns of repetition (e.g. AAAA), alternation (e.g. ABAB), and pairs (e.g. AABB). These different patterns of movements are reminiscent of loops, repeats, or other types of chunking in the Simon Game.

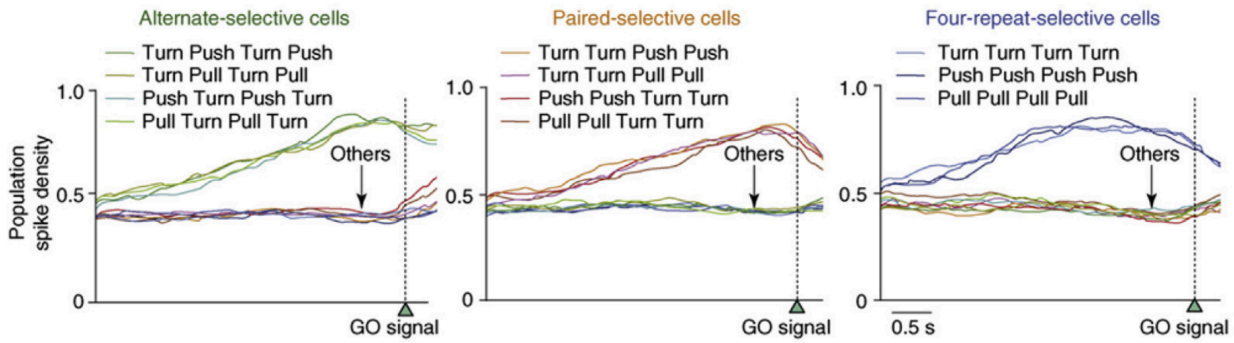


Figure 3: Neural activity tuned to specific patterns of movements (Shima et al. 2007)

Hypotheses

Behavioral

Our overarching behavioral hypothesis is that subjects chunk in the Simon Game. We will specifically seek to test exactly how subjects chunk. In other words, what is the subject's chunking strategy? We call these chunking strategies 'coding schemes'. Coding schemes explain how a raw sequence length (i.e. number of buttons pressed) is transformed into a shorter description that can be used to reproduce the full sequence. We define an array of five coding schemes, each of which serves as a behavioral hypothesis:

1) Lempel-Ziv (LZ) Complexity

LZ is a compression algorithm. It does not offer a code that can reproduce the full sequence, so we do not consider it as a realistic hypothesis for human behavior. However, it does offer a pre-existing compression metric, so it is useful to compare to our hand-crafted coding schemes below.

LZ works by measuring how many different 'phrases' are in a sequence. Whenever the algorithm sees a new phrase, it gets added to a 'dictionary' and the complexity count increases by 1. For example, in the sequence 'ABABAB' we first encounter the character 'A'. Thus, 'A' is the first phrase in the dictionary. Next, we encounter 'B'. Since there is no existing 'B' in the dictionary, this phrase gets added. Then, we reach 'AB' which also gets added to the dictionary. The last part of the sequence is again 'AB'. Since we already have 'AB' in the dictionary, the last two characters do not increase the complexity count. In total, the count ends up being three (A, B, and 'AB').

2) Simple Repeats

This algorithm chunks together any chain of three or more of the same color. For example, the sequence, 'RRRBYG' would become '[R]3BYG'. The 'code length' for the original sequence would be six. Code length is the size of the code needed to reproduce the sequence. We consider ordinals to be size 1 and the repeat multiples to be size 1 (the sizes of other types of chunks are outlined below). The compressed sequence is therefore size five (4 ordinals plus 1 repeat multiple).

3) Complex Repeats

This scheme is an extension of the simple repeats algorithm that can chunk any repeating pattern, not just a single color. For example, ‘RGRGRG’ becomes ‘[RG]3,’ compressing from size six to three. Repeats can also be nested; for example, ‘RRRGGGRRRGGG’ (size 12) becomes ‘[[R]3[G]3]2’ (size 5), which is different from the simple repeats scheme which would only compress to ‘[R]3[G]3[R]3[G]3’ (size 8).

4) Alternations

This scheme chunks alternations in the sequence. An alternation is defined as an ‘ABA’ pattern with a fixed size of two, where ‘A’ and ‘B’ are any two colors. For example, ‘ABA’ would be chunked as an alternation, ‘[AB*]’ (size 2).

5) Cycles

This scheme introduces the concept of ‘cycles,’ which are determined by the colors’ position on the screen (or positions of the buttons on the device). In our implementation (Fig. 2), one full clockwise cycle starting at yellow would be ‘YBGR,’ while a counterclockwise cycle would be ‘YRGB.’ These would be coded as ‘[Y4+]’ and ‘[Y4-]’ respectively, where Y is the start color, four is the total number of items, and +/- denotes direction (clockwise/counter-clockwise). The size of a cycle is 2 + α, where α sets the contribution of the cycle direction to the size. In these results we set α = 0.5, following the logic that as a binary variable (clockwise or counterclockwise) it should be easier to remember than a color, which can be any one of four possibilities.

Neural

We propose three models for how Simon sequences are represented and chunked. Two are ‘persistent activity’ models (Slot model and Temporal model) and one is a short-term synaptic plasticity model. The details for these models are outlined in the Methods section below.

Methods

Behavioral Analysis

Why Use the Simon Game?

Empirically, we have observed that subjects can perform very long sequences through the Simon Game. While this game solicits fascinating behavior, we considered modifying the implementation for sake of simplicity. Specifically, we considered removing the gradual accumulation of items in the sequence (i.e. show the entire sequence at once and have the subject reproduce as much as they can remember). We tested this approach on one subject, and their performance declined precipitously. We need long sequences to be able to differentiate chunking hypotheses (see Figure 2 below), so we ruled out this approach. Another approach would be to remove the sequential aspect of the task and present the entire stimulus as a static image. A human subject did perform comparatively well with this task. However, in future work, we hope to extend this research to non-human primates to test our neural models. We have attempted this ‘entire-stimulus-at-once’ approach with non-human primate subjects and their working memory capacity peaks around only three items. We believe monkeys may be able to

achieve longer sequences through the traditional Simon-style item accumulation. We used the unaltered Simon Game for this project, such that we can have a suitable animal model for the human behavior.

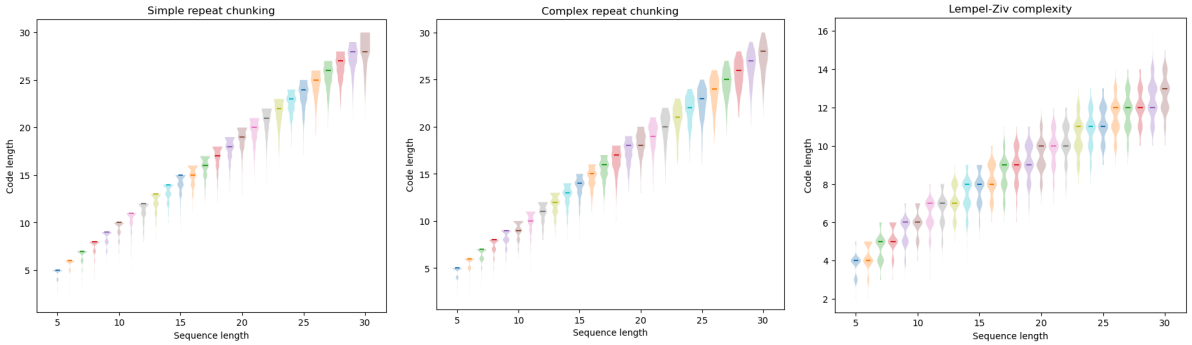


Figure 4: Three example coding schemes are shown. Shorter sequences show qualitatively less variance in their code length. Hence, our hypotheses about coding schemes can be differentiated better by behavioral performance data on longer sequences.

Task Implementation

We used the Pygame library to implement the Simon Game in Python. Sequences are randomly generated. The subject uses a computer trackpad to click the appropriate color. The code then logs the user's response, reaction time, and score. Score is the sequence steps that the user has achieved.

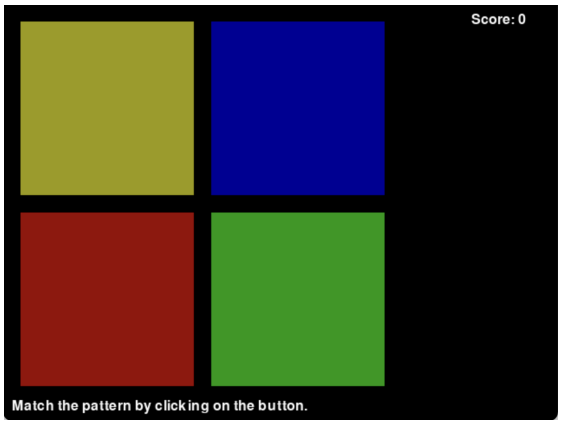


Figure 5: Our implementation of the Simon Game (Adapted from the *InventWithPython.com* version). Players use a mouse interface to reproduce the presented sequence of button clicks, which grows by one on each trial until an error is made and the game ends.

Justification for Random Sequences

We considered using pre-generated sequences that were most likely to be chunked according to specific coding schemes. For example, the sequence 'Red, Red, Red, Red, Red, Red, Red' is highly chunkable according to the 'Simple Repeats' hypothesis. However, we ultimately decided to use randomly generated sequences. Using random sequences will allow our findings to explain human behavior in the real Simon Game, rather than on contrived sequences that are unlikely to occur in practice. Furthermore, we found that using random colors offers qualitatively different predictions of which sequences will be

more chunkable. In other words, we predicted that random sequences would be sufficient to differentiate our hypotheses.

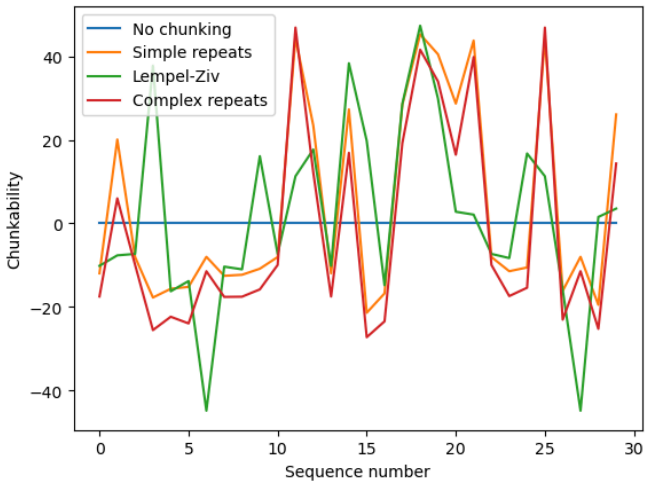


Figure 6: A set of 30 randomly generated sequences offers qualitatively different chunkability predictions.

Chunkability

Given a coding scheme, we define ‘chunkability’ as a measure of the deviation from the median code length for a particular sequence length. In other words, chunkability measures how atypical the code length is for a given sequence length, compared to the typical (median) code length expected under that coding scheme. Specifically, the way we compute the chunkability of a sequence *S* of length *L* is to generate 1000 random sequences of length *L* and compute their code lengths under the given coding scheme to define a distribution of code lengths. Then, we compare the code length of *S*, *C(S)*, to the distribution to get a percentile. Chunkability is defined as *50% – percentile(C(S))*, such that more positive chunkabilities mean shorter-than-expected code lengths and vice-versa. This definition ensures that with randomly generated sequences, there is no inherent positive correlation between chunkability and sequence length (Fig. 7).

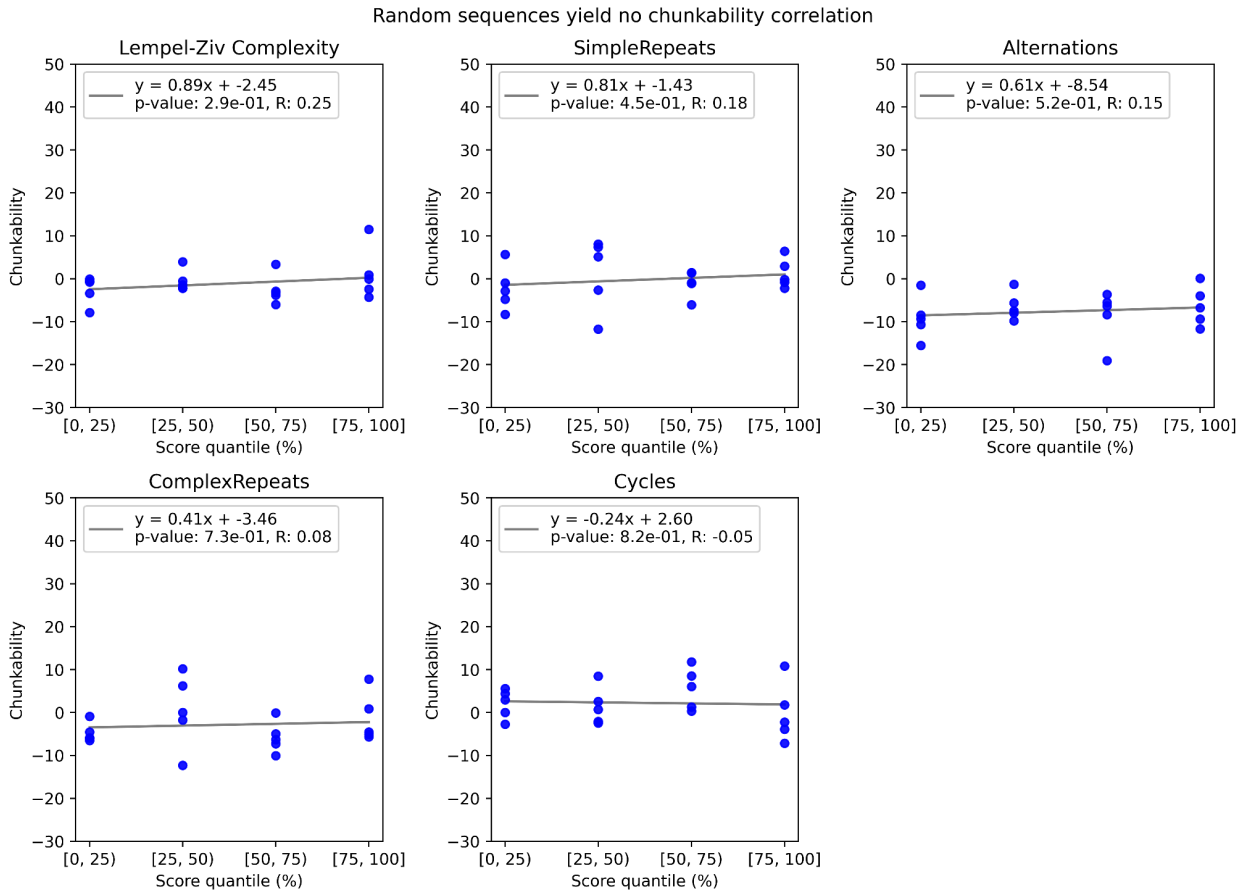


Figure 7: Random sequences yield no chunkability correlation. With 500 randomly generated sequences, we observe no correlation between chunkability and 'score,' which in this case is sequence length. This analysis was motivated by the analysis on the participants' data (Fig. 10) to show that the null hypothesis for each coding scheme, that participants' scores do not vary with that scheme's chunkability, is indeed no significant correlation.

Implementation of Coding Schemes

The implementation of the coding schemes and all other code for this paper can be found in this Github repository: https://github.com/qsimeon/PSY1401Projects/tree/main/final_project.

Neural Models

Our neural models serve as hypotheses for how the 'chunked' Simon sequences are represented in the brain. The models focus on the time window immediately before the subject starts reproducing their final successful sequence. We selected this time window because it is the portion of the task where the subject is representing the entire sequence.

Slot Model

For sake of simplicity, this model assumes a subject with a working memory capacity of four chunks. It also assumes that the subject uses the 'simple repeats' coding scheme. The aim of this model

is to provide an implementation of basic chunking as it might occur in a non-human primate model (from whom the appropriate neural data could be collected).

The model has four populations (or slots) of neurons, representing four ordinals (or chunks) in a sequence. Population 1 represents the first ordinal, Population 2 represents the second ordinal and so on. In each population, there are four neurons which each prefer one of the four colors in the Simon Game. If the sequence is 'Red, Blue, Yellow, Green' then the neuron that prefers red in the first population will be activated. The neuron that prefers blue in the second population will be activated, and so on.

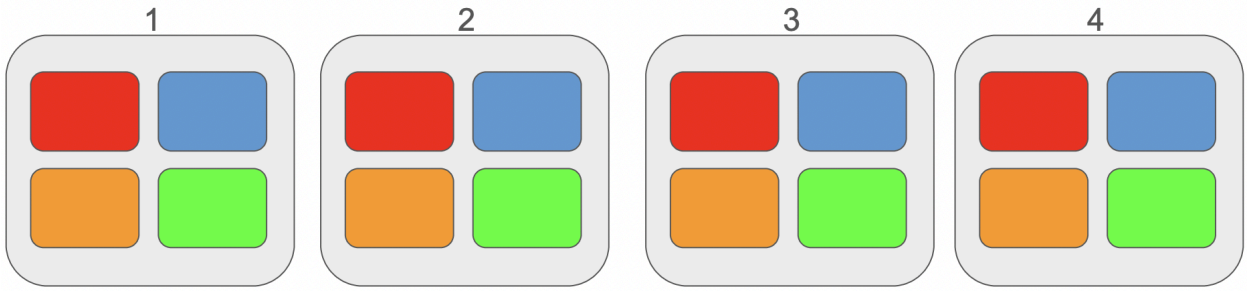


Figure 8: Four populations of four neurons representing ordinals in a sequence.

In our simulation, we will track the firing rates of these neurons for particular Simon sequences. The firing rates are sampled from a Poisson distribution where:

$\lambda_{i,j}(t)$ for neuron j in population i at time t is given by:

$$\lambda_{i,j}(t) = \lambda_{\text{base}} + \frac{\lambda_{\text{max}}}{1 + e^{-k(n_{i,j} - x_0)}}$$

- λ_{base} is the baseline firing rate when not activated by its stimulus.
- λ_{max} is the maximum possible increase in firing rate above the baseline.
- $n_{i,j}$ is the number of repeats of the stimulus activating neuron j in population i.
- k and x_0 are parameters that shape the sigmoid function, controlling the steepness and the midpoint of the response curve, respectively.

Temporal Model

This model makes the same simplicity assumption as the Slot model in that it only accounts for simple repeats. However, instead of encoding ordinality in slots, this model encodes ordinality in time. Similar to a phonological loop, the sequence gets repeated in working memory in a cyclical pattern. This cycle is implemented through a population of four neurons, with one neuron preferring each of the four colors in the Simon Game. If the sequence is 'Red, Blue, Yellow, Green' then the neuron preferring red will be activated, followed by the neuron preferring blue and so on. Once all four neurons have been sequentially activated, the sequence will restart. The model assumes that the subject has a maximum 'cycle duration' of 20 units of time, and that each chunk necessitates 5 units of time. As such, the subject can only remember a maximum of four chunks.

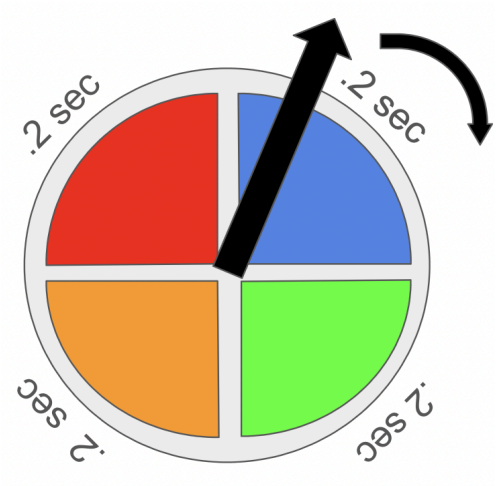


Figure 9: The Temporal model consists of a population of four neurons with each neuron preferentially activated by one of the four colors in the Simon Game. The neurons fire in order according to the order of the colors in the stimulus. More specifically, they fire in a cyclical pattern to maintain the memory.

In our simulation of the Temporal model, we will track the firing rates of these neurons for particular Simon sequences. The firing rates are sampled from a poisson distribution where:

$\lambda_i(t)$ of neuron i at time t is given by:

$$\lambda_i(t) = \lambda_{\text{base}} + \frac{\lambda_{\max}}{1 + e^{-k(n_i - x_0)}}$$

- λ_{base} is the baseline firing rate when not activated by its stimulus.
- $\lambda_{\max}$ is the maximum possible increase in firing rate above the baseline.
- n_i is the number of repeats of the stimulus activating neuron i.
- k and x_0 are parameters controlling the shape of the sigmoid function, adjusting the steepness and the midpoint, respectively.

Cognitive Control-based Short-term Synaptic Plasticity (CC-STSP) Model

The previous models aimed to formalize hypotheses about the format in which individual chunks may be encoded in the persistent activity of a neural population dedicated to working memory during the delay period. We additionally explored an alternative, more comprehensive approach to modeling, which does not assume that the entire sequence is represented within persistent activity. This model instead encodes a sequence by keeping track of a high-level context, learning the transition weights to new items in the sequence, and switching to a new context if the transition weights have been saturated. We call this the **Cognitive control-based Short-term Synaptic Plasticity (CC-STSP) model.**

The model utilizes two modular systems – a working memory component (potentially consistent with accounts based on persistent activity) for keeping track of and switching between high-level contexts, and another recurrent output module that instantiates the context, which is trainable via short-term

synaptic plasticity that can learn the transitions within the sequences. Here we describe in detail how the model works.

A *context* is defined as the full transition function from one item to the next.

$$\mathcal{C} : \{r, b, g, y\} \rightarrow \{r, b, g, y\}$$

For instance, a randomly generated context c_1 might look like:

$$c_1(r) = b$$
$$c_1(b) = b$$
$$c_1(g) = r$$
$$c_1(y) = g$$

In our model, a weight matrix of size 4x4, W_1, instantiates the context, which is to be thought of as the recurrent output module that can be used to actually generate responses. Given one-hot vectors corresponding to each one of the possible items, applying the weight matrix outputs a vector of length 4, interpreted as the logits of the next item. We choose the argmax to be the output y of the context.

When presented with the first item in a sequence, say $x = r$, the 0th default context c_0 is added to the list of contexts, instantiated as a recurrent module with randomly initialized weights. This list of contexts define the working memory component, to be actively maintained via persistent activity. We do not specify the neural instantiation for this maintenance.

Given the first item, the recurrent output module now generates a “prediction” of the next item by applying W_0 to it. The model is then presented with the second item, say $y = b$. if b is already predicted by the context, then no weight update is required. Otherwise, the module will update the weights according to a simple learning rule for a multi-neuron perceptron, i.e.

$$W_0 \leftarrow W_0 + \alpha * x \otimes (y - \hat{y}).$$

Importantly, we repeat this update until the context correctly outputs $\hat{y} = b$, to ensure that the context will correctly output the next item. This constitutes the real time, “short-term synaptic plasticity”-based learning of the transition weights in the model.

After the learning-encoding phase, the model is then probed to generate sequences in response. We assume the very first item is a given. The model then applies the context to the first item to generate the prediction of the next item. Since by the learning phase, the context was trained to output the second item, it will reproduce the sequence correctly. This process of observing a new item in the sequence, updating the weights to predict the new item, and then generating a response based on the learned predictions generally characterizes the function of the recurrent output module.

The current context can be kept as long as the transitions in the sequences obey the same patterns for a given input item. For instance, a single context defined as above is sufficient for perfectly reproducing a cyclic sequence such as “gybrgybrgybrg…”, because the transition function is not changing and the context, i.e. the recurrent weights, can be recurrently applied indefinitely. Intuitively, just remembering where to go next is sufficient to remember a cycle.

When the model encounters a new transition that requires “unlearning” a previously learned transition, however, the model must switch to a new context, since the transition function, i.e. the context, has now “saturated”. The cognitive control module will thus add a new context to the actively maintained list of contents, and newly learn the transition function. Encoding the sequence proceeds in this manner by updating weights and creating new contexts as needed. During the response phase, all the model has to do is remember when in the sequence the context switched, and now use the weights of the new context to predict the next item in the new context.

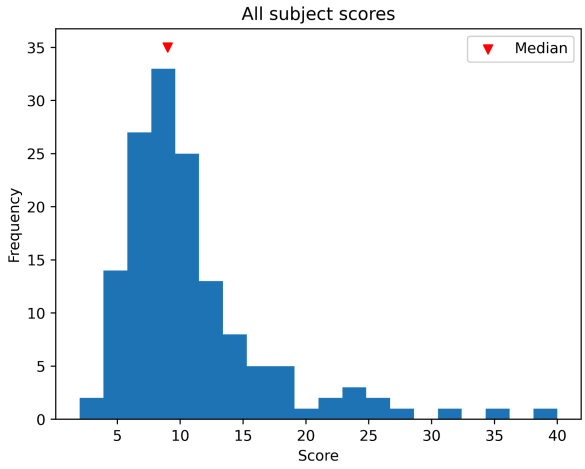
We assume a cap on the number of contexts that the model can actively maintain, set to 5 in our modeling implementation, roughly approximating the number of objects in working memory. In the response phase, we set up the simulation such that with probability $\{(\#\ contexts)/(max\ \#\ contexts)\}^2$, the model will “confuse” the context, selecting a random context to generate the next item from. In other words, the more contexts the working memory module is keeping track of, the more likely that the context (i.e. the transition function) may get “mixed up.”

In our current implementation of the CC-STSP model, we additionally allow for the possibility of reusing a context for “repeat” sequences. In general, when a new context begins, our model currently does not have a way in which a previously used context in the sequence gets reused. For repeats, however, since the context transition function is straightforward and reusable for repeats of any item ($c_n(x) = x$), we can plausibly suppose that whenever the model encounters a repeat of items, it chooses to reuse the context. This prevents the creation of a whole new context, and thereby has the effect of reducing the cognitive load on the model. In this sense, context-reuse may be an additional candidate for how chunking is implemented as a neural process within the brain.

Results

Behavioral Analysis

A



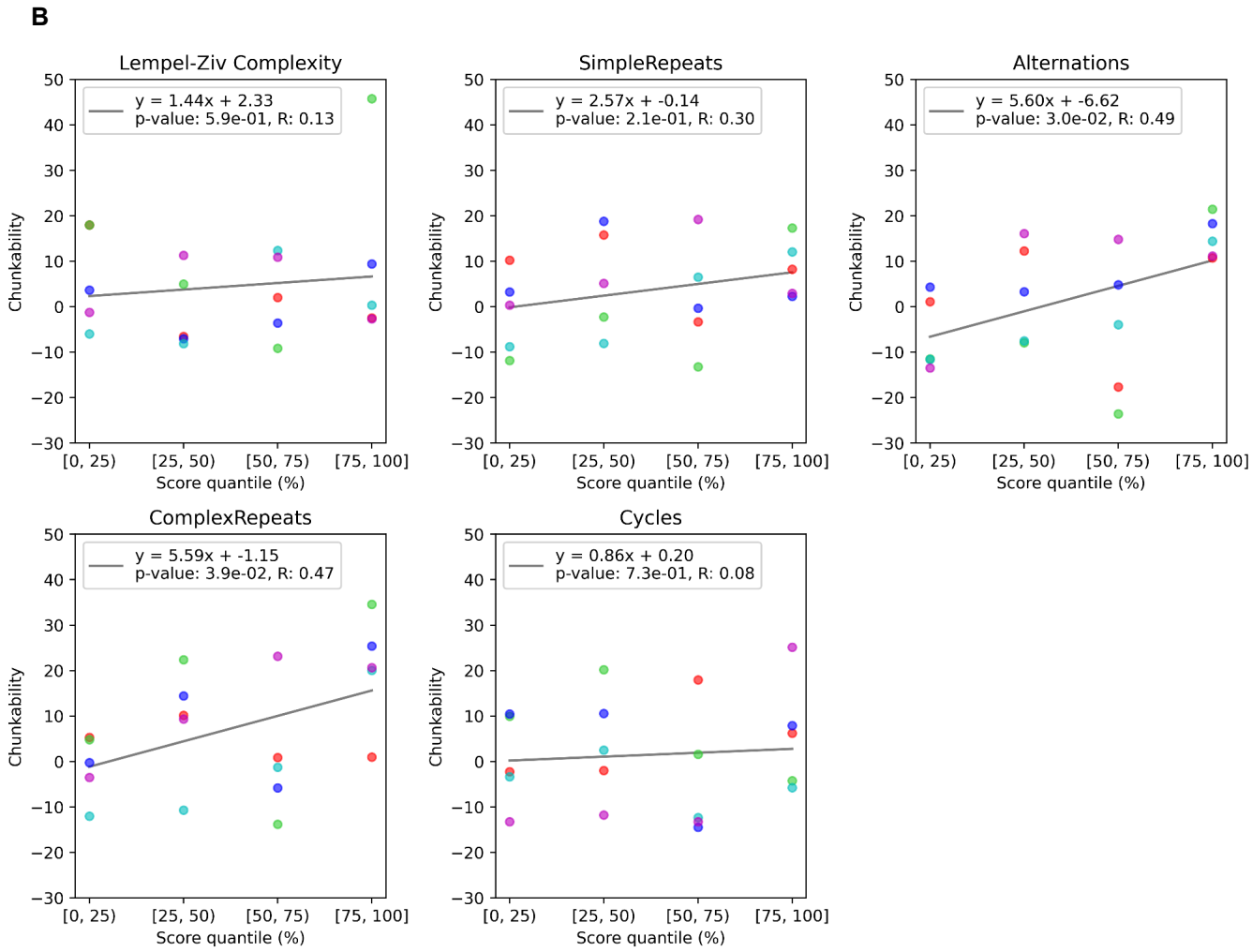


Figure 10: (A) Histogram of all subjects' scores. The median score is 9. (B) On average, participants' scores varied significantly with the alternations scheme chunkability and the complex repeats scheme chunkability, but not with Lempel-Ziv complexity, simple repeats scheme chunkability, and cycles scheme chunkability. The R-values for alternations and complex repeats were comparable, but slightly higher for alternations (0.49 vs. 0.47). Each color (red, green, blue, cyan, magenta) represents one subject. The data shown is from 123 games total; however, each subject played a different number of games (17, 18, 14, 50, 24, respectively). The subjects also had different score ranges and medians (19, 14, 7, 9, 8, respectively). To account for this, we averaged within-subject for each score quartile before performing the regression across all subjects.

The fact that we did not observe a significant effect for simple repeats does not necessarily mean that the subjects do not chunk simple repeats. Rather, it likely results from the fact that in the random Simon game sequences, simple repeats do not account for enough of the variation in the true difficulty of the sequences. The significant complex repeats effect indicates that adding the chunking of multi-color repeats (e.g., RBRB) and perhaps also nested repeats accounts for more of this variation. Overall, having more data from more subjects would allow us to make stronger claims about comparisons between these coding schemes.

Neural Models

Slot Model

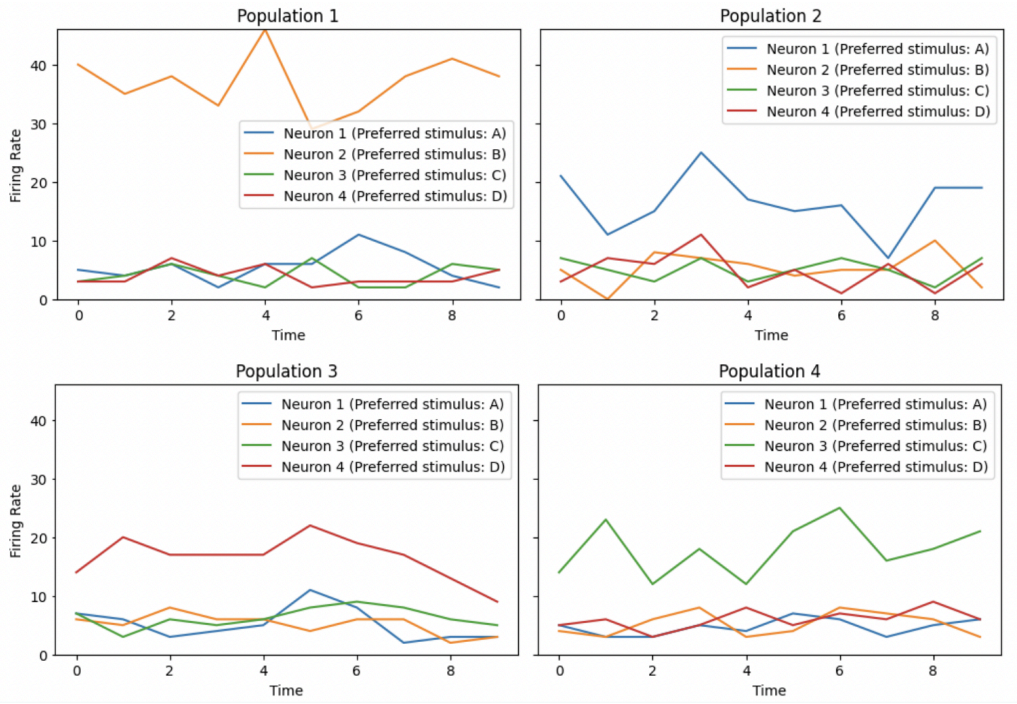


Figure 11: Four populations of simulated neurons responding to an example stimulus sequence, ‘BBBADCBDA’. The firing rates are determined according to the Slot model.

As described in the methods section, this simulation utilizes the ‘simple-repeats’ coding scheme and has a working memory capacity of four chunks. The stimulus sequence for the simulation (‘BBBADCBDA’) is therefore compressed to ‘[B]3, A, D, C’. In the first population of neurons, chunk 1 ([B]3) is represented through Neuron 2. This neuron is preferentially activated by stimulus, ‘B’. Since B is repeated three times in this case, the firing rate is scaled up according to the ‘n’ term in the model. The other three chunks are represented in their respective populations through more limited activation of their associated neurons. The activations are more limited because n = 1 for chunks 2, 3, and 4.

Temporal Model

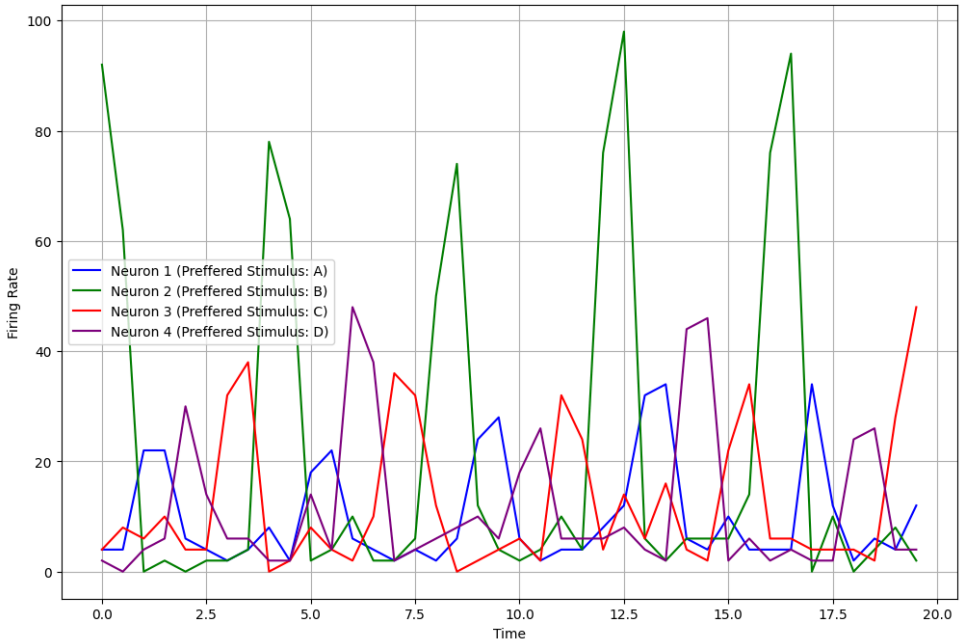


Figure 12: A population of simulated neurons responding to an example stimulus sequence, 'BBBADCBDA'. The firing rates are determined according to the Temporal model.

This simulation utilizes the same example sequence as the Slot model ('BBBADCBDA'). As described above, this simulation utilizes the 'simple-repeats' coding scheme and has a working memory capacity of four chunks. As with the Slot model, the stimulus sequence for the simulation ('BBBADCBDA') is compressed to '[B]3, A, D, C'. In the first population of neurons, chunk 1 ([B]3) is represented through Neuron 2. This neuron is preferentially activated by stimulus, 'B'. Since B is repeated three times in this case, the firing rate is scaled up according to the 'n' term in the model. The other three chunks are represented sequentially thereafter (in a cyclical pattern) through more limited activation of their associated neurons. The activations are more limited because n = 1 for chunks 2, 3, and 4.

Cognitive Control-based Short-term Synaptic Plasticity (CC-STSP) Model

The CC-STSP model was implemented as a neural process model that is runnable on arbitrary target sequences, and can generate behavior as a model for a human. Figure 13 shows the distribution of model scores (median: 10.0) on 500 randomly generated sequences.

The model also produces plausible behavior for particular sequences. For instance, of the 500 random sequences tested on the model, the following are the model's three highest-scoring sequences, which are arguably on the more "chunkable" end:

['rryyyybrybrryryrrby', 'bgbbyrrrgggggggyggyyg', 'ybrrrgrbbggyyrrgbgryr']

For each sequence, we can also obtain on which points a new context was created while performing the trial. These context-switching points can be interpreted as generating both psychological predictions about the delimiters of chunking in the mind, as well as neural predictions about the temporal pattern of cognitive control and error signals in the brain.

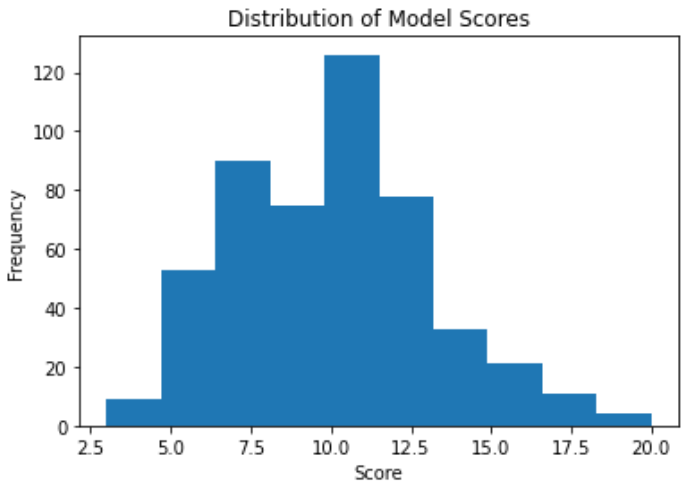


Figure 13: Histogram showing simulated scores of model behavior on 500 randomly generated sequences. (Median model score: 10.0)

The full implementation of the CC-STSP model is available in our github repository.

Discussion

Behavior Analysis

These results argue that subjects indeed chunk in the Simon Game. Humans perform significantly better on more chunkable sequences, for certain coding schemes. Specifically, we found that humans tend to chunk alternations and complex repeats. However, there was diverse inter-subject variability. For example, subjects with cyan and green dots in Figure 10 had significant effects chunking simple repeats, even though this effect did not generalize when we averaged across subjects.

While this data does suggest that we use particular types of chunks, there are many other coding schemes that could be tested. For instance, coding schemes that combine together different chunking strategies might better capture the experimental data. Future work should formalize additional coding schemes.

Neural Models

Slot Model and Temporal Model

These models make several neural and behavioral predictions. First, they predict that subjects will have a cap on the number of chunks that they can represent. The exact capacity limit would depend on the subject's number of slots (for the slot model) or cycle duration (for the temporal model). In this case, the models are implemented to represent a maximum of four chunks. Therefore, both models predict that subjects would only be able to remember the first seven letters (i.e. the first four chunks) of the example sequence 'BBBADCBDA', if they were given a structurally similar sequence in the Simon Game.

Our primary neural hypothesis is that chunking is implemented through the firing rates of neurons. Both models predict that there will be neurons whose firing rates scale with the number of repeats. This neural mechanism allows for the system to represent longer sequences without needing additional cells, slots, or time in the cycle durations. The models are implemented such that firing rate scales with repeats according to a sigmoid function. This implementation is biologically realistic in that it places a ceiling on firing rate, however it predicts that the subject will have difficulty deciphering the exact number of repeats when a color is repeated many times (activations for higher numbers of repeats will be increasingly similar). This behavioral prediction was not tested here, but could be tested in the future, particularly with monkey subjects.

Finally, the Temporal model predicts cyclical activity, while the Slot model predicts discrete neural populations for each chunk. These predictions may help to differentiate the two models in neural data.

Cognitive Control-based Short-term Synaptic Plasticity (CC-STSP) Model

This model takes a qualitatively different approach from the above models, as it presents a fully runnable model that implements the entire process of stimulus observation to response generation. As such, the model makes direct behavioral and neural predictions.

The model can be used to assess various psychological hypotheses behaviorally. The current implementation assumes hand-picked parameters for maximum number of contexts, probabilities for context confusion, the size and depth of recurrent weights, and more. Additionally, there are choices regarding more high-level structural constraints of the model, e.g. what kinds of features to extract given a sequence beyond using the last item, which types of contexts to allow re-use, whether to allow contexts to interfere with each other upon saturation of contexts, etc. Such modeling variables collectively constitute psychological hypotheses about how humans may solve the task. Given this, if we have sufficient behavioral data, we can fit the model's parameters to the data and validate on a held-out test set, and see which of the instantiations best explain and predict human behavior. Such an approach may also be used to explain individual variability. Notably, the model suggests that context-reuse may be an important mechanism with which chunking may be implemented as a neural process.

Neurally, the primary difference between this model and previous models is that it utilizes short-term learning of weights in a dynamic recurrent system to account for sequential working memory. Measurement or inference of synaptic weight changes may validate aspects of the hypothesis. For instance, given a highly predictable sequence within the current context, there may be minimal short-term weight changes since the model predicts that no weight updates are required. On the other hand, when a transition is encountered for the first time within that particular context, neural observations may show elevated cognitive control and synaptic weight changes. Importantly, the model can be thought of as specifically generating temporally fine-grained predictions about the presence of a specific item-invariant error or "switch-context" signal at certain points in the sequence. If such signals indeed do exist whenever there are context-switches as predicted by the model, it will provide strong evidence that indeed cognitive control plays a major role in the Simon Game.

Collectively, this model proposes a strategy for solving Simon Game where the sequence is encoded in the dynamics of the recurrent output module, namely within its trainable weights, rather than via persistent activity. The working memory component, instead of tracking items or chunks of items *per se*, is viewed as a cognitive controller, keeping track of transitions between contexts and deciding to switch to a new context given an "error" that requires overriding learned transitions. According to this

view, the capacity limit (or the excess capacity) in the Simon's task does not stem only from the capacity limit of working memory, but potentially also from the capacity limit of the weight space.

Limitations and Future directions

Behavioral Analyses

One limitation of this study is the small sample size. Also, the subject pool was not blinded to the research questions. Future work should recruit more subjects. With a larger dataset, it might become tractable to reverse engineer coding schemes by identifying chunks through reaction times. Lastly, it might be interesting to investigate what other factors besides chunking help subjects to remember long sequences in the Simon Game. For example, do subjects leverage the multi-sensory evidence that the task provides (i.e. both color and sound)?

Neural Modeling

The primary limitation for our modeling is the lack of available neural datasets to test our hypotheses. Our lab is interested in collecting electrophysiology data from non-human primates in the Simon Task, but it remains to be seen whether monkeys are capable of performing sufficiently long sequences to identify chunking. Another limitation is that the models are incomplete in various ways.

While the Slot and Temporal models present specific hypotheses about the format of encoding of items during the delay period, the models do not provide any account of how the neurons could come to represent the items in such a way, nor how downstream regions may utilize the neurons to generate output. The CC-STSP model aims to overcome these limitations by outlining a fully runnable model that implements the entire process of stimulus observation to response generation. However, integral components of the model, such as the cognitive control architecture or the mechanism of context reuse, have yet to be distilled into concrete neural models. Formalizing the conceptual theory in neural instantiations would be the right step going forward.

Conclusion

The Simon Game is an iconic children's game. This work argues that chunking explains our surprisingly lengthy Simon sequences. In contrast to existing literature, this work suggests that working memory does not have fixed item capacity. Rather, working memory can leverage chunking to reduce the description length of sequential information. Our proposed coding schemes and neural models offer explanations for how this chunking may be implemented in our minds and brains. These algorithms and mechanisms may extend beyond the Simon Game to underlie a variety of situations where humans deploy high-capacity working memory.

Sources

Mathy, Fabien, and Jacob Feldman. "What's magic about magic numbers? Chunking and data compression in short-term memory." *Cognition* 122.3 (2012): 346-362.

Miller, George A. "The magical number seven, plus or minus two: Some limits on our capacity for processing information." *Psychological review* 63.2 (1956): 81.

Dehaene, Stanislas, et al. "The neural representation of sequences: from transition probabilities to algebraic patterns and linguistic trees." *Neuron* 88.1 (2015): 2-19.

Dehaene, Stanislas, et al. "Symbols and mental programs: a hypothesis about human singularity." *Trends in Cognitive Sciences* 26.9 (2022): 751-766.

Shima, Keisetsu, et al. "Categorization of behavioural sequences in the prefrontal cortex." *Nature* 445.7125 (2007): 315-318.