Supplementary Information:

Why Neurocognitive Creativity Research Needs More Computational Modeling

James Lloyd-Cox[1], Alan Pickering[1], Joydeep Bhattacharya[1]

[1]Department of Psychology, Goldsmiths University of London, London, UK

The attached code (in Matlab®; see https://github.com/Alan-Pickering/example-creativity-model) implements a simple toy model of within-category search processes. We have left extensive comments in the code but this supplementary text explain in greater detail how the provided model code works and how one might experiment with it. In the spirit outlined in the main article, our primary objective in this exercise has been to demonstrate a simple but formal model related to creative cognition. We have tried to do this in the most accessible and transparent fashion. The hope is to enable those new to formal computational modelling to get a clearer insight into the modelling process, rather than making any major claims for the specific features of this particular model. It also makes transparent the process of making assumptions and modelling choices inherent in every formal model.

This model was designed to simulate creativity tasks in which the instructions are to search a conceptual space for an unusual item or response (e.g., trying to come up with an unusual exemplar from the category of "fruit", where unusual is defined as a response that would have been suggested by very few people when asked to generate fruit exemplars).

Concept network as a multidimensional space

The central idea in this model is to represent the concept network (e.g., fruits) as an *n*-dimensional space. In the code provided, we simplify this space to just two dimensions (in the code *ndims*=2). Our first assumption (A1) is that the number of dimensions will not affect the qualitative behaviour of the model. We should investigate that assumption by running simulations using higher-dimensional models. In general, we recommend starting with simplifying assumptions but, where possible, one should test the impact of each assumption one makes. The model's key feature is that each exemplar is represented as a unique point in the space and the Euclidean distance[1] between any two exemplars reflects the overall strength of association between the exemplars. Our second assumption (A2) is that the free-wheeling, undirected flow of thought in this space will more likely move between exemplars strongly associated with one another (e.g., apple and orange; these examples will have a small Euclidean distance between the points they occupy). The construction of the model as outlined below ensures that the model generally behaves according to assumption A2.

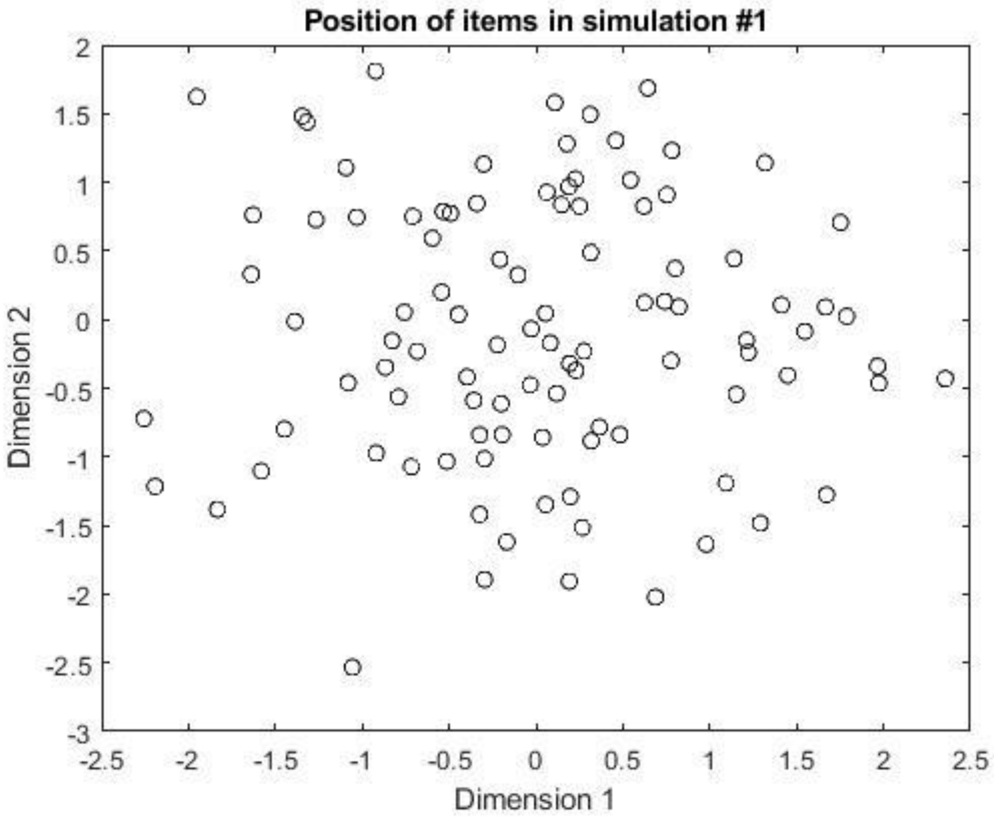
In this model space, the dimensions can be considered features over which items such as fruits might be associated. For example, one dimension might be "size", and because apple and orange are similar in size, the distance between them on size dimension would be small; alternatively put, their association in terms of size would be strong. Consider another dimension, "citrus-ness"; here, we

[1] See https://en.wikipedia.org/wiki/Euclidean_distance.

expect the distance to be large as oranges are citrus fruits but apples are not. However, it seems likely that orange and apple would be close together on most of the model dimensions, so the overall Euclidean distance separating them (across dimensions) would be small in our model space. Thus, when one thinks of apple (as an example from the fruit category), one is likely to think of orange and vice versa.

Simulating the concept network using multivariate normal distributions

To generate the position of the items in the model space, we use a random number generation process. Specifically, we generated the exemplars using a multivariate normal random[2] generator (bivariate in this case as our space has two dimensions). Before explaining the simple model implementation procedure in Matlab (of note, the procedure will be similarly easy in most other coding languages), we need to unpack the assumptions that flow from this modelling choice. First, we need to set several items in our model space. We chose 100 (in the code, *nitems*=100) as a rough estimate of the total number of fruits an adult human might be able to name given enough time. Our third assumption (A3) is that the precise number chosen is not going to change the way the model would behave, so long as we avoid really small unrealistic values (<10). Supplementary Figure 1 (SF1) shows an example of a multivariate normal random sample of 100 items using 2 dimensions. One can see different samples by changing the *plot2show* control variable in the code provided. SF1 was generated using *plot2show*=1 and is the set of items used in simulation number 1.



Supplementary Figure 1: A random sample of 100 items generated using a multivariate normal random number generation process in 2 dimensions. The distances along each dimension are arbitrary and standardized.

[2] See https://en.wikipedia.org/wiki/Multivariate_normal_distribution

SF1 shows that most items are clustered close to the centre of the space, and the density of items gets less as we move outwards from 0 on either dimension. This implies that the sample items at the centre of the space have lots of closely associated items and that as one moves towards the edge of the space, each item has fewer close associates. In searching this space to find unusual items (which is the task we are trying to simulate), one needs to move towards the periphery of the space. Items at the periphery are less likely to be retrieved and thus, by definition, are more unusual items in this category.

The multivariate normal (MVN) random process means that, along each dimension, the distribution of item positions follows a univariate normal distribution. While a univariate random variable has a mean and a standard deviation (0 and 1 respectively, for a standard normal distribution), the MVN distribution has a mean vector and a variance-covariance matrix (*mu* and *sigma,* respectively, in the code). We set the means to be zero on each dimension and the item variances to be 1 (*itemvar* in the code). These are just standardized values and are not important. Nevertheless, they do allow us to scale other parameters in our model easily, given that we know, with these choices, that roughly 5% of our items will lie outside the values of -2 and 2 on each dimension.

We also can choose whether there is any covariation between the values on the separate dimensions of our space. In the model code, this is specified via *itemcov*. Our next assumption (A4) is that these two dimensions are not related; therefore, we set *itemcov* = 0. This ensures that the cloud of points in our 2-d space is roughly circular; non-zero values for the covariance would stretch the cloud of points into an elliptical shape. This is, of course, a simplifying assumption which we believe is almost certainly wrong even if the other aspects of the model might be useful: over all the dimensions on which fruits can vary, we feel confident that the distance between pairs of items on some dimensions will be correlated with their distances along some other dimensions (e.g., size will be loosely inversely correlated with the intensity of flavour, think melon vs blackcurrant or raspberry). Once again, tests of assumption A4 should be made if the simpler model proves useful. We need to use >2 dimensions to explore this assumption properly; with >2 dimensions, we can arrange it so that the degree of covariation between pairs of dimensions can vary over different pairs of dimensions.

As already noted, generating the MVN distributed items is simple: it is a single line of code once we have the parameters described above. In Matlab, we use the *mvnrand* command and write (line 105):-

```
itemvals=mvnrnd(mu, sigma, nitems);
```

It is important to note that we do not specify a set of fruits in this model or try to set their associative closeness to one another to reflect some objective reality. Our assumptions and model specification create a set of exemplars that we propose could represent any set of exemplars in a category of finite size. An obvious alternative would be to create an associative network for a specific category with the weights of association (distances) between items being set to “realistic” values. This could be done by testing the associative strength between exemplar pairs for real categories using lexical information or experimental data involving human participants. The weights used would then be set to be proportional to the measures of associative strength obtained. This sort of approach has been used in past computational models of creativity (see Box 2 in the main text for examples). This is a more complex approach, and we could test whether our simplifying assumptions lead to a model which is capable of producing simulated behaviour similar to that produced using a more elaborate model based on “real” associative weights.

Modeling free-wheeling associative thoughts

The next key aspect of the model is our choice for implementing the free association of thoughts. This is intended to capture one facet of the dual-process models discussed in the main text (see Box 1 in particular): the "spontaneous", or "generative", or "automatic" flow of ideas during the search for a creative response. We did this using a random walk[3]. Random walks have been used quite extensively in modelling behaviour in varied fields within psychology [1-5].

In the model, there is a loop of 200 simulations (*nsims*=200), and each simulation can be thought of as a simulated participant attempting to generate an unusual fruit (one that would be thought of by as few other participants as possible). The choice of 200 is fairly arbitrary but, given the extensive use of random variables in the model, it needs to be large enough to give representative outcomes when aggregated across all simulations. Within each simulation, there is an inner loop of up to 10000 steps (*nsteps*=10000). Each step is one step of the random walk. The number of steps is initially set to be large, although we adjust this to a lower number (1500) for reasons explained below. The length of a timestep is arbitrary, but one could rescale the numbers of steps into response times (e.g., 100 steps equates to 1 second) so that the simulated response times are of the right magnitude. The walk has to start at an initial position. In the simplest version of the model, we assume (A5) that the walk starts at the centre of the space: (0,0) in two dimensions (coded as *mystart*). This makes sense because if we are asked to think of fruits, it is highly likely that we would first think of common exemplars at the centre of our space.

Another feature is the walk step size, i.e., the amount that the walk might move in each direction on a single step. Bearing in mind that 95% of the items lie along dimension values in the range -2 to 2 (see above), we set the step size to 0.05 for each dimension (coded as *stepsize*). To make the walk random, we used a uniform random number generator to create a random move direction for each of the *nsteps* (=10000) steps of a simulation along each of the *ndims* (=2) dimensions. The direction for each step on each dimension was either -1, 0 or +1 (each occurring randomly with equal probability), generating 9 possible moves on each step (3*3 over the two dimensions). To do this, we used the *randi* command in Matlab to create a matrix of values with 10000 rows and 2 columns for each simulation (called *mymoves*):

```
mymoves=randi([-1 1],nsteps,ndims);
```

The actual walk is thus a combination of the direction specified by *mymoves* multiplied by the amount moved in that direction, specified by *stepsize*. On the *k*-th step of the walk, the variable *currpos* keeps track of the current 2-d position of the walk iteratively, thus:-

```
currpos=currpos+stepsize.*(mymoves(k,:));
```

Clearly, the larger the values used for *stepsize, the mor*e space will be covered by the random walk. All other things being equal, our *model intuition*[4] is that larger *stepsize* values will enable the simulated participant to encounter more unusual (peripheral) items more quickly.

[3] https://en.wikipedia.org/wiki/Random_walk

[4] A model intuition is what effect we think the model parameter will have. It is important to test these out. Formal modelling allows one to move from intuitions to clear predictions when we run the simulations.

Retrieval of candidate items

The next aspect of the model is how we interrupt the free-associative process of the random walk with attempts to retrieve candidate items. The way we capture this in our model is that, every so often, the walk pauses and the simulated participant attempts to retrieve an example item from the current position of the random walk in our space of items. The frequency with which this attempt at retrieval occurs is controlled by a parameter in the code called *walkfor* (a default value of 50 steps). This means that after every *walkfor* random walk steps, an attempt at item retrieval is made. This feature of the model embeds another assumption (A6); namely, that the spontaneous free-association processes are in alternating phases with memory retrieval and subsequent evaluative processes. This is a feature present in some theoretical accounts in the literature (see [33] in the main article).

Which items might be retrieved at each attempt? Following the underpinning logic of the model – i.e., that the distance in the model space represents the closeness of the association of an item – we implemented this using a competitive probabilistic retrieval process based on the relative distance of items from the current position of the walk. This means that the items nearest to the current position are compared in terms of their relative distances, and the probability of their retrieval is directly linked to those relative distances (closer items being more likely to be retrieved). The retrieval competition is limited to those items which are within a specific Euclidean distance of the current position. This selection of potential items for retrieval is controlled by a parameter, denoted *closeto* in the code (default =1 distance unit). This parameter is used by computing the Euclidean distance between each item in the space and the current position (computed as *eucdist*, see line 140), and then computing a "logical filter" (*choicefilt* in the code) with value = 1 for those items with a Euclidean distance (ED) less than *closeto*, and 0 otherwise. The filter is used later on in the code to restrict the choice function to apply to only those items with values of *choicefilt*=1. In mathematical terms, we can define a set *S* of potentially retrievable items where the Euclidean distance of item *i* from the current walk position, ED_i is smaller than *closeto* for all items *i* in the set *S*.[5]

The default value of *closeto* represents a wide search radius given that 95% of the items are within a circle of radius 2 units from the centre of the space. One might imagine that different individuals might vary in the value of *closeto* that they use. Our next model intuition is that someone with an ability to think more creatively might have a higher value of *closeto*, than their less creative counterpart, and so such a person would include more potential items in their retrieval searches. The more creative person (based upon their higher value of *closeto*) would, according to the model, be more likely to alight upon a more creative (unusual) choice of fruit in a fixed amount of thinking time.

The actual formula used to compute the probabilities of retrieval of the items lying within *closeto* distance units of the current position was based upon a widely adopted choice function in psychological modelling: the softmax function.[6] Specifically, the softmax formula used to define the probability of retrieving item *i*, given that *i* is a member of the set S of potentially retrievable items, is as follows:-

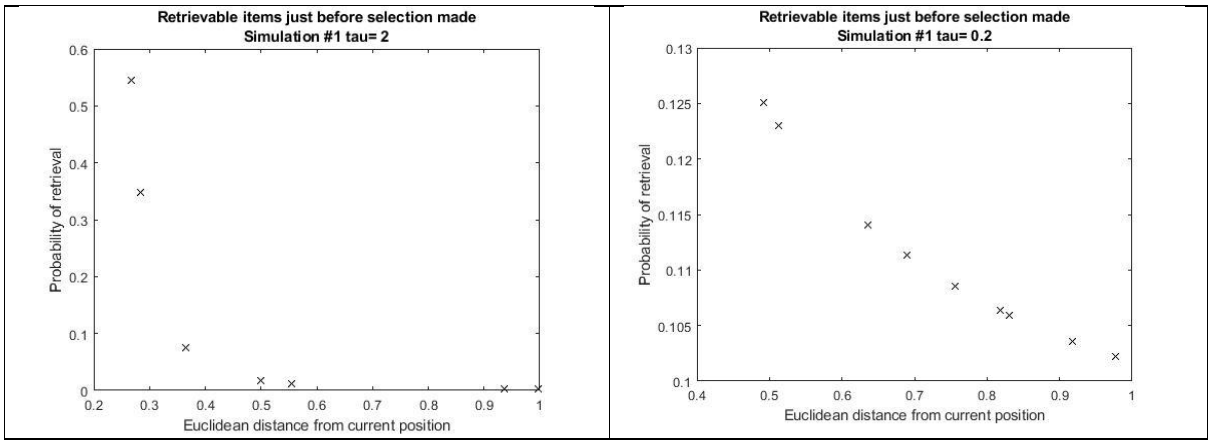
$$p(R_i \mid i \in S) = \frac{e^{\tau/ED_i}}{\Sigma_{j\in S}\, e^{\tau/ED_j}}$$

[5] In mathematical notation if item *i* is a member of a set S, this can be written as $i \in S$

[6] https://en.wikipedia.org/wiki/Softmax_function

The above formula suggests that the smaller the Euclidean distance of a candidate item *i* from the current position, the higher the probability that it is retrieved amongst all the competitor items in set *S*. In the code, it is computed across lines 157-159. The following lines of code (165-167) use a random number generator to retrieve a specific item in accordance with the probabilities returned by the softmax function.

A key parameter in the softmax function is τ (in the code, this is *tau*), and it can take values of zero or greater. This parameter has different names in modelling contexts (e.g., inverse temperature, stochasticity or exploration-exploitation parameter), but it simply controls how noisy the choice process is. As τ gets larger then the closest item is more and more likely to be retrieved (i.e., a more deterministic choice), even when it is only slightly closer than the next nearest competitor. By contrast, as τ approaches zero, all competitor items tend to be chosen with similar probabilities irrespective of their relative distances (more random, noisy choice). We can see this effect of τ in Supplementary Figure 2 (SF2) during one retrieval decision during simulation number 1.



Supplementary Figure 2: The effect of parameter τ on retrieval probabilities as a function of Euclidean distance (ED) from the current walk position. Note the difference in y-axis scale across the two panels. In the left panel ($\tau = 2$), there are 7 potentially retrievable items, but the closest two, with EDs < 0.3 have the greatest chance of being retrieved (0.55 and 0.35 approximately). Items with ED ≥ 0.5 have virtually a zero probability of retrieval. In the right panel ($\tau = 0.2$) all 9 potentially retrievable items have a similar probability of being retrieved (0.1 < p < 0.13) irrespective of their EDs from the current position, even though the range of EDs goes from just below 0.5 to almost 1.

Once again, one has a strong model intuition that the parameter τ should directly affect the breadth of the search of the category space and thus the ability to generate more creative solutions. When τ is smaller, then more items can be retrieved at any position of the random walk than for higher values of τ. This means that, over a fixed period, the random walk has a greater chance of retrieving more peripheral items (i.e., more unusual, creative choices) for lower values of τ, all other things being equal. We can think of τ in conjunction with the value of *closeto* (already discussed) as opening up the retrieval process to a broader range of possible items. In terms that have been widely used in the creativity literature, one might view these two parameters as reflecting the degree of inhibition in memory retrieval; specifically, the combination of large *closeto* and low τ equates to weak inhibition.

Response selection as a controlled decision process

The final part of the model is the decision process used to decide if a retrieved item is "unusual enough" to be worthy of being given as a response. This represents the other facet of dual-process theories: the "deliberate", or "evaluative", or "controlled" process.

If a retrieved item is considered unusual enough, then it will be given as a response (e.g., I have though of the fruit "durian" and I am happy to give this as my example of an unusual fruit). If it is not deemed unusual enough, then the random walk resumes *from the position of the retrieved item*[7] until a future retrieval attempt is made, *walkfor* random walk steps later. What decision rule might a participant use to decide that a retrieved item was unusual enough? We considered that it must be some simple property of the retrieved item that a simulated participant could use to decide upon unusualness. For example, perhaps after retrieving an item, the participant finds that it brings to mind very few close associates, then one might decide that it is worth offering as a creative response. As a simple proxy for this, we used an alternative related decision rule (controlled in the code by the parameter *respmethod* taking a value of 1; *respmethod*=2 has a different effect see below): the retrieved item has to be more than a threshold Euclidean distance from the centre of the space. The threshold distance is specified by a parameter called *respthresh* in the code. At the start of each simulation, *respthresh* is set to *respthreshbase* (=2.0 by default). We chose this value in light of the parameters chosen for the multivariate normal distribution of the items (which force 95% of the items to lie between -2 and 2 on each dimension). If the retrieved item has an ED from the centre which exceeds the value of *respthresh*, then the response is made and the current random walk stops (controlled in the code by setting *endkflag* to 1). The simulation loop records the response information and then moves on to the next simulated participant.

To reflect the potential effect of time pressure on the task we assumed (A7) that participants might relax their decision threshold the longer they could not produce a suitable response. We simulated this by having a decrement to the value of *respthresh* (called *threshdrop* in the code, default value =0.01). The decrement is applied every 100 steps of the random walk if a response has not been made (in the code this is controlled by setting a parameter, *threshtime* = 100). This is a minor feature of the model and we can explore its effect by setting *threshdrop*=0 in the model.

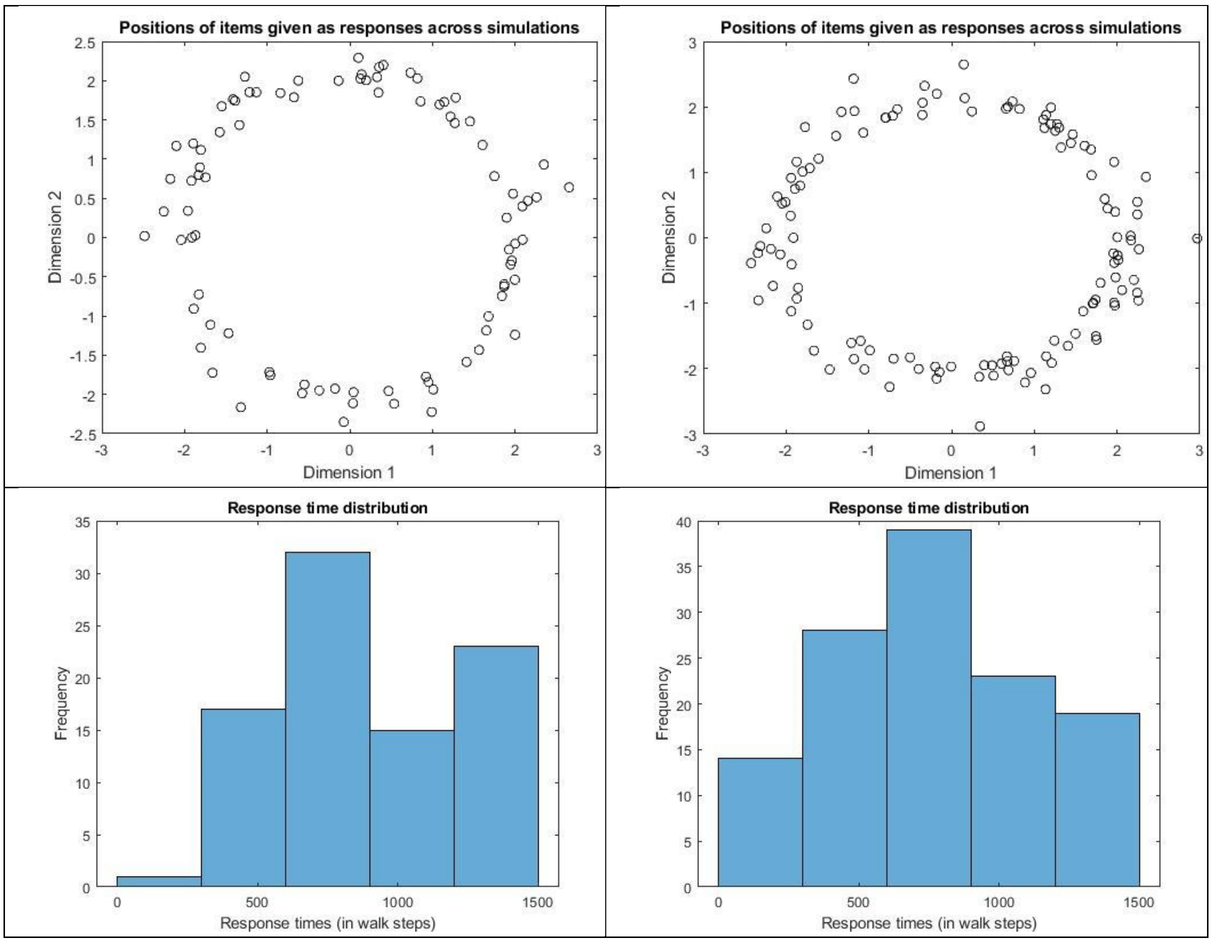
Exploring the effects of parameters

It is relatively easy to explore the effect of the model parameters, which are likely to affect the ability to give more creative responses. As noted above, for example, we have clear model intuitions for the effects of *stepsize*, *closeto*, and *tau*. Thus, it is essential to test these model intuitions, along with the effects of the assumptions listed above. For example, below, we show the effect of changing the value of *tau*.

[7] The position of the random walk is supposed to represent where one's thoughts currently are at within the category space. Thus, if one has retrieved an item then the position of that item seems a reasonable choice for the current "position" of your thoughts. In the code the current walk position is moved to the position of the retrieved item on line 177. So, strictly, the model is a random walk punctuated with jumps to retrieved items.

It was noted above that the maximum number of time steps (*nsteps*) per simulation was set to 10000. This large value was used to develop and set the values of the model parameters. This parameter setting was chosen because it allowed every simulated subject to retrieve an unusual item across the whole range of *tau* values used (0.2 to 2). To test the model, we inspected the response time distributions achieved when 10000 walk steps per simulation were permitted. Based on the mean response times over 200 simulations (generally around 1600-1700 steps), we set the maximum response time to 1500 timesteps in our test simulations. This change was made to reflect our assumption (A8) that, under a small amount of time pressure, not all subjects would be able to select a genuinely unusual response (using the *respthresh* decision rule). That is, thy know that all the candidate responses they think of in the time allowed are likely to have been thought of by lots of other people. The model does not yet implement what response they offer in these circumstances. Perhaps they would give one of the less unusual items previously retrieved but still in their (working) memory. As the simulations below confirm, using the 1500 step maximum per simulation ensures that not every simulated participant can produce an unusual item (i.e., one that passes the decision threshold) in the time allowed.

Supplementary Figure 3 (SF3) shows the simulation results for two very different *tau* values (0.2 vs. 2.0). Our model intuition above was that a lower value of *tau* would lead to more creative responses. In our simulation, this intuition would be confirmed if a higher proportion of the 200 simulated participants (with *tau*=0.2) can give a response that passes the *respthresh* decision rule compared with 200 simulated participants (with *tau*=2). The simulation confirmed our intuition: 88/200 simulated participants (with *tau*=2) were able to generate an unusual response. By comparison, 123/200 simulated participants (with *tau*=0.2) were able to generate an unusual response. The responses made were generally in very similar average positions in the model space see SF3) and were all towards the periphery of the space, in keeping with the nature of the *respthresh* decision rule. In addition to the greater number of responses in the low *tau* condition, SF3 shows that the responses were made more rapidly in the low *tau* simulations (an average of 734 random walk steps, s.d.=358) compared with the high *tau* simulations (average= 878 walksteps, s.d.=364). Thus, in respect of *tau*, our model intuitions are confirmed. We leave it as an exercise for the interested reader to explore the effects of *stepsize* and *closeto* in relation to the model intuitions offered above.



Supplementary Figure 3: The effect of parameter τ on the ability to make an unusual category response in the time allowed. The leftmost panels are for *tau* = 2.0, and the rightmost panels are for *tau* = 0.2. The upper row records the position of the response items made (note the greater number of responses made for the 200 simulations with tau = 0.2). The bottom row records the distribution of response times for the responses given.

Making new predictions with the model and testing them

It is an essential first step to show that the model behaves in the ways our intuitions suggested it would. As already noted, one must also explore, as fully as possible, the impact of the numerous assumptions and choices made in developing the model. However, for the model to be useful, it should lead to novel predictions for real creative behaviours that can be tested in actual human participants. Below we illustrate how even this simple model can generate testable predictions.

The model can easily be made to simulate a fluency task as well (where one has to name as many exemplars of the category as you can in a fixed time). In fact, the provided code already records the number of category exemplars retrieved during the “think of an unusual fruit” simulation. To simulate a fluency task, the “unusual response” decision process must be turned off. This can be achieved by setting *respmethod* to have a value of 2 (instead of the usual value of 1; *respmethod*=2 makes no decision about whether a retrieved item is unusual). Then one can run the code with *nsteps*=1500 (to represent the fixed amount of response time for the fluency task) with *tau*=0.2 vs 2.0. Across 200 simulations, the mean number of unique items retrieved for *tau*=0.2 was 22.1

(s.d.=2.5). For *tau*=2.0, the average number of unique items retrieved was 16.9 (s.d.=3.5). The model thus shows that variation in parameter *tau* can underlie an ability to generate an unusual response more often and be more fluent in generating category exemplars. We leave it as an exercise for the interested reader to see if the same patterns over both tasks can be obtained using variation in the parameters *closeto* and *stepsize*.

Of course, one might argue that by using verbal reasoning alone, one could have arrived at the prediction that more creative people would generate more unusual responses and be more fluent (i.e., retrieving more exemplars from the category). Having a formal model allows one to explore this predicted effect more rigorously and thoroughly. In the main article, the idea was briefly noted that a strategic search along one dimension of the category might help find unusual items. For example, one might think of exotic locations that one has visited and thereby recall unusual fruits experienced in those locations. A simple way to give dimensional directionality (of this kind) in the search could be to make the step sizes for the random walk different for each dimension. In the code provided, the step size along dimensions 1 and 2 was equal (0.05). With the same average stepsize, a more directed walk would be achieved with step sizes of 0.025 and 0.075 (or vice versa). For low *tau* settings (*tau*=0.2), this had little effect on the number of unusual responses made (in fact, they decreased slightly to 118/200 simulations c.f. 123/200 simulations with equal step sizes). For high tau settings (*tau*=2.0), unequal stepsizes increased the number of unusual responses achieved to 103/200 (c.f. 88/200 with equal stepsizes). This is a novel prediction of the model: in the "find an unusual exemplar task", people with lower creativity (higher *tau*) are more likely than their more creative counterparts (low *tau*) to benefit from a suggestion to use a strategy of focusing their search along one/some feature dimension(s). This prediction could be tested with actual participants by testing them under conditions when provided with a dimensional search strategy by the experimenter and comparing the results with performance under a control condition where no such strategy was given. Strictly the prediction applies only to those whose high vs. low creativity stems from processes captured by the parameter *tau* (which controls the noisiness of the exemplar retrieval process).

Model limitations

There are many limitations and unrealistic features of the current model. There is not space here to consider them all. Two striking illustrative examples of limitations are noted. First, the random walk phases in each simulation are of a fixed duration (controlled by the parameter *walkfor*). If these phases are intended to represent periods of mind-wandering around the conceptual space, then it seems unrealistic that these periods would all be of the same duration. An easy fix would be to use a Gaussian random variable (with mean and standard deviation) for *walkfor*, so that in a particular simulation, the number of steps of the random walk between each retrieval attempt would vary randomly about the mean value. The parameters of the random variable could vary across different simulated individuals.

Secondly, and more importantly, the item retrieval process takes no time in the model. Therefore, the model should be extended to include a retrieval time component. Such a component should ensure that the pattern of EDs of potentially retrievable items influences the time taken for the retrieval in a principled way. For example, one would imagine that a pattern of EDs in the left panel of SF2 (two exemplars with small EDs and high probabilities of retrieval; the rest further away and with very low probabilities of retrieval) would lead to quite different retrieval times than the pattern in the right panel (no close exemplars and all exemplars have a similar probability of retrieval). A pervasive finding is that response times are slower for more difficult decisions [6]. It is relatively straightforward to incorporate retrieval times into the model in a realistic way; for example, one might use a so-called accumulator model [7].

Supplementary References

1) Mathys, C., Daunizeau, J., Friston, K. J., & Stephan, K. E. (2011). A Bayesian foundation for individual learning under uncertainty. *Frontiers in Human Neuroscience*, *5*, 39.

2) Nosofsky, R. M., & Palmeri, T. J. (1997). An exemplar-based random walk model of speeded classification. *Psychological Review*, *104*(2), 266–300.

3) Ratcliff, R. (1978). A theory of memory retrieval. *Psychological Review*, 85, 59-108.

4) Simkin, M. V., & Roychowdhury, V. P. (2014). Stochastic modeling of a serial killer. *Journal of Theoretical Biology*, *355*, 111–116.

5) Tuerlinckx, F., Maris, E., Ratcliff, R., & De Boeck, P. (2001). A comparison of four methods for simulating the diffusion process. *Behavior Research Methods, Instruments, & Computers: A Journal of the Psychonomic Society*, *33*(4), 443–456.

6) Ratcliff, R., & McKoon, G. (2008). The diffusion decision model: theory and data for two-choice decision tasks. *Neural Computation*, *20*(4), 873–922.

7) Ratcliff, R., & Starns, J. J. (2013). Modeling confidence judgments, response times, and multiple choices in decision making: Recognition memory and motion discrimination. *Psychological Review*, *120*(3), 697–719.