Dear Prof. Mathis and Prof. Frank,

We thank the editors and reviewers for their time taken to review our manuscript, as well as their patience awaiting our response. In addition to addressing the specific points raised by the reviewers, we have taken this opportunity to describe and document a variety of other improvements made to the RatInABox software package which have broadened its utility. The core purpose of the software remains consistent, yet both the user base and feature set have significantly expanded since our initial submission. Supporting this growth has been a priority.

As such, we have revised the original manuscript, taking care to present a clear message suitable for *eLife's* broad audience. We now present it for re-submission with new text coloured blue and adjust/reworded text in orange in the main document. Below, we first describe the main changes made and then provide, for each reviewer, a detailed point-by-point response to their reports.

Significant changes made to the code base

- **Improved Environments** can now contain objects, curved/polygonal boundaries and holes.
- **New cell types** including object vector cells, agent vector cells, random spatial neurons, field of view encodings as well as non-linear and trainable neural network neurons.
- **New demo scripts** including those on splitter cells, conjunctive grid cells, deep learning, actor-critic reinforcement learning agents, vector cells and successor features.
- **PyPI installation and proper versioning** >>> pip install ratinabox
- **Multiple agents.** Environments now support containing multiple agents simultaneously.
- **Website:** https://ratinabox-lab.github.io/RatInABox
- **Formalisation of head direction.** Agents now have an independent head direction variable which is updated and used by egocentric cell types (e.g. head-direction cells).
- **New plotting functions.** For example users can plot the head direction selectivity of any neuron class, or the head-direction average selectivity of any neuron.
- **Many minor changes** including initialisation parameter checking, figure plotting improvements, API simplifications across many classes, bug fixes etc.

Significant changes made to the manuscript

- **Consistency update.** Significant changes have been made to the methods and throughout the manuscript to make it consistent with the current state of the software package, RatInABox v1.11.4.
- **Figure 1 refresh.** While the fundamental content of figure 1 remains unchanged, all panels have been refreshed to more accurately reflect new features.
- **Figure 2 panel e.** New panel showing the position analysis requested by reviewer 1
- **Figure 3 refresh and new panels.** Figure 3 has undergone a redesign, now featuring three additional panels (b, c, and d) and an inset panel (f) presenting novel analyses aimed at addressing the concerns and recommendations put forth by the reviewers. All original content of the figure 3 has been retained.
- **Staging:** A new introduction paragraph stages why this package is useful
- **Use-cases.** A new paragraph has been added to clarify the intended use case of RatInABox
- **Discussion of limitations.** Additional discussion of the limitations of RatInABox to address the recommendations of the editors and reviewer 2.
- **Comparisons.** Additional discussion comparing RatInABox to other techniques for generating behavioural and neural data.

> **Editors comments:**
>
> (1) We would like to see a better discussion of limitations in the Discussion, and some more staging of why this is useful in the Introduction.

We now openly discuss the limitations of the software package in the Discussion on page 9 in a new section of text as follows:

"Our package is not the first to model neural data[37,38,39] or spatial behaviour[40,41], yet it distinguishes itself by integrating these two aspects within a unified, lightweight framework. The modelling approach employed by RatInABox involves certain assumptions:

1. It does not engage in the detailed exploration of biophysical[37,39] or biochemical[38] aspects of neural modelling, nor does it delve into the mechanical intricacies of joint and muscle modelling[40,41]. While these elements are crucial in specific scenarios, they demand substantial computational resources and become less pertinent in studies focused on higher-level questions about behaviour and neural representations.
2. A focus of our package is modelling experimental paradigms commonly used to study spatially modulated neural activity and behaviour in rodents. Consequently, environments are currently restricted to being two-dimensional and planar, precluding the exploration of three-dimensional settings. However, in principle, these limitations can be relaxed in the future.
3. RatInABox avoids the oversimplifications commonly found in discrete modelling, predominant in reinforcement learning[22,23], which we believe impede its relevance to neuroscience.
4. Currently, inputs from different sensory modalities, such as vision or olfaction, are not explicitly considered. Instead, sensory input is represented implicitly through efficient allocentric or egocentric representations. If necessary, one could use the RatInABox API in conjunction with a third-party computer graphics engine to circumvent this limitation.
5. Finally, focus has been given to generating synthetic data from steady-state systems. Hence, by default, agents and neurons do not explicitly include learning, plasticity or adaptation. Nevertheless we have shown that a minimal set of features such as parameterised function-approximator neurons and policy control enable a variety of experience-driven changes in behaviour the cell responses[42, 43] to be modelled within the framework.

And have also added a new opening paragraph to better stage why RatInABox is useful (and computational modelling toolkits in general) to *eLife*'s broad audience:

"Computational modelling provides a means to understand how neural circuits represent the world and influence behaviour, interfacing between experiment and theory to express and test how information is processed in the brain. Such models have been central to understanding a range of neural mechanisms, from action potentials [1] and synaptic transmission between neurons [2], to how neurons represent space and guide complex behaviour [3,4,5,6,7]. Relative to empirical approaches, models can offer considerable advantages, providing a means to generate large amounts of data quickly with limited physical resources, and are a precise means to test and communicate complex hypotheses. To fully realise these benefits, computational modelling must be accessible and standardised, something which has not always been the case."

> (2) It would be great to address limitations on how the environment and cells can be modelled currently. Namely, how the cells can be modified to study influences of other internal (latent) states, such as reward or abstract coding, and dynamics. And please clarify what types of experimental data that can and cannot be modeled, the limitations of experimental estimates from the models, and importantly, what testable predictions of yet unknown results can be generated by this model, beyond replicating what is already known.

In the text described above and added to the discussion, we discuss the assumptions and limitations of RatInABox. Briefly these are: no low-level biophysical, biochemical, joint or muscle-level modelling, no 3D or higher dimensional / abstract environments, no discretised environments, no explicit sensory inputs (e.g. vision or olfaction).

Most of these limitations are, as we hope to have made clear in the manuscript, features not bugs. They were explicit choices to keep the RatInABox framework lightweight and easy to understand. We wanted to avoid "feature creep" and repeating work done by others. Interfacing RatInABox with other toolkits or exploiting the existing RatInABox API to add new features could overcome these limitations and we will happily consider these contributions as/when they are made.

Regarding "how the cells can be modified to study influences of other internal (latent) states, such as reward or abstract coding, and dynamics"*:* Briefly (and expanded on in the individual responses below), in a series of new tutorials we demonstrate how RatInABox can be used to model:

- Splitter cells[1] (i.e. where a latent factor determines cell firing dynamics)
- Conjunctive cells (where multiple primitive cell classes combine to give mixed selective cells e.g. jointly selective to speed and position)
- Reinforcement learning[2,3] (where reward is explicitly encoded and used to "train" representations and guide behaviour)
- Successor features[4] (where cell representations are dependent on statistics of the agents trajectory).
- …

To name a few. As such, the following text has been modified in the case studies, section 3.1.

"Additional tutorials, not described here but available online, demonstrate how RatInABox can be used to model splitter cells, conjunctive grid cells, biologically plausible path integration, successor features, deep actor-critic RL, whisker cells and more. Despite including these examples we stress that they are not exhaustive. RatInABox provides the framework and primitive classes/functions from which highly advanced simulations such as these can be built"

Furthermore, function-approximator neurons (such as the existing FeedForwardLayer, or the new NeuralNetworkNeurons) can be used to model dynamic systems either by (i) setting the input to a neuron layer as itself (a recurrent input) or (ii) setting the function approximator to be dynamic, e.g. a recurrent neural network. The following text has been added to section 2:

"Naturally, function-approximator Neurons can be used to model how neural populations in the brain communicate, how neural representations are learned or, in certain cases, neural dynamics. In an online demo we show how grid cells and head direction cells can be easily combined using a FeedForwardLayer to create head-direction selective grid cells (aka. conjunctive grid cells[10]). In Fig. 3d and associated demo GridCells provide input to a NeuralNetworkNeuron which is then trained, on data generated during exploration, to have a highly complex and non-linear receptive field. Function-approximator Neurons can themselves be used as inputs to other function-approximator Neurons allowing multi-layer and/or recurrent networks to be constructed and studied."

Finally, we already know that *"testable predictions of yet unknown results can be generated by this model"* as a few early citations of the package[8,9,10] (which have been added to the manuscript) demonstrate. However, primarily we think of RatInABox as a generator of models rather than a model itself, therefore making testable predictions is not the primary contribution and might seem out of place. We have modified the following sentence in the discussion to clarify this:

“Its user-friendly API, inbuilt data-plotting functions and general yet modular feature set mean it is well placed empower a wide variety of users to more rapidly build, train and validate models of hippocampal function[25] and spatial navigation[26], accelerating progress in the field.”

As well as the final paragraph of the discussion which was added to say:

“In conclusion, while no single approach can be deemed the best, we believe that RatInABox’s unique positioning makes it highly suitable for normative modelling and NeuroAI. We anticipate that it will complement existing toolkits and represent a significant contribution to the computational neuroscience toolbox.”

To clarify that normative modelling and NeuroAI are two key areas we anticipate RatInABox making contributions.

Reviewer #1 (Public Review):

In this work George et al. describe RatInABox, a software system for generating surrogate locomotion trajectories and neural data to simulate the effects of a rodent moving about an arena. This work is aimed at researchers that study rodent navigation and its neural machinery.

Strengths:
+ The software contains several helpful features. It has the ability to import existing movement traces and interpolate data with lower sampling rates. It allows varying the degree to which rodents stay near the walls of the arena. It appears to be able to simulate place cells, grid cells, and some other features.
+ The architecture seems fine and the code is in a language that will be accessible to many labs.
+ There is convincing validation of velocity statistics. There are examples shown of position data, which seem to generally match between data and simulation.

Weaknesses:
+ There is little analysis of position statistics. I am not sure this is needed, but the software might end up more powerful and the paper higher impact if some position analysis was done. Based on the traces shown, it seems possible that some additional parameters might be needed to simulate position/occupancy traces whose statistics match the data.

Thank you for this suggestion. We have added a new panel to figure 2 showing a histogram of the time the agent spends at positions of increasing distance from the nearest wall. As you can see, RatInABox is a good fit to the real locomotion data: positions very near the wall are under-explored (in the real data this is probably because whiskers and physical body size block positions very close to the wall) and positions just away from but close to the wall are slightly over explored (an effect known as thigmotaxis, already discussed in the manuscript).

As you correctly suspected, fitting this warranted a new parameter which controls the strength of the wall repulsion, we call this “wall_repel_strength”. The motion model hasn’t mathematically changed, all we did was take a parameter which was originally a fixed constant 1, unavailable to the user, and made it a variable which can be changed (see methods section 6.1.3 for maths). The curves fit best when wall_repel_strength ~= 2. Methods and parameters table have been updated accordingly.

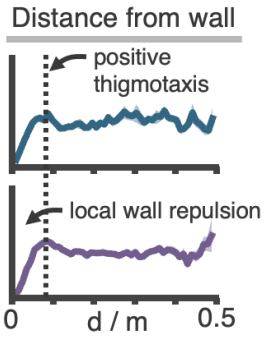


Fig. 2e Histogram of the area-normalised time spent in annuli at increasing distances, d, from the wall. RatInABox and real data are closely matched in their tendency to over-explore locations near walls without getting too close.

+ The overall impact of this work is somewhat limited. It is not completely clear how many labs might use this, or have a need for it. The introduction could have provided more specificity about examples of past work that would have been better done with this tool.

At the point of publication we, like yourself, also didn't know to what extent there would be a market for this toolkit however we were pleased to find that there was. In its initial 11 months RatInABox has accumulated a growing, global user base, over 120 stars on Github and north of 17,000 downloads through PyPI. We have accumulated a list of testimonials[5] from users of the package vouching for its utility and ease of use, four of which are abridged below. These testimonials come from a diverse group of 9 researchers spanning 6 countries across 4 continents and varying career stages from pre-doctoral researchers with little computational exposure to tenured PIs. Finally, not only does the community *use* RatInABox they are also *building* it: at the time of writing RatInABx has received logged 20 GitHub "Issues" and 28 "pull requests" from external users (i.e. those who aren't authors on this manuscript) ranging from small discussions and bug-fixes to significant new features, demos and wrappers.

Abridged testimonials:

- *"As a medical graduate from Pakistan with little computational background…I found RatInABox to be a great learning and teaching tool, particularly for those who are underprivileged and new to computational neuroscience."* - Muhammad Kaleem, King Edward Medical University, Pakistan
- *"RatInABox has been critical to the progress of my postdoctoral work. I believe it has the strong potential to become a cornerstone tool for realistic behavioural and neuronal modelling"* - Dr. Colleen Gillon, Imperial College London, UK
- *"As a student studying mathematics at the University of Ghana, I would recommend RatInABox to anyone looking to learn or teach concepts in computational neuroscience."* - Kojo Nketia, University of Ghana, Ghana
- *"RatInABox has established a new foundation and common space for advances in cognitive mapping research."* - Dr. Quinn Lee, McGill, Canada

The introduction continues to include the following sentence highlighting examples of past work which relied of generating artificial movement and/or neural dat and which, by implication could have been done better (or at least accelerated and standardised) using our toolbox.

"Indeed, many past[13, 14, 15] and recent[16, 17, 18, 19, 6, 20, 21] models have relied on artificially generated movement trajectories and neural data."

> + Presentation: Some discussion of case studies in Introduction might address the above point on impact. It would be useful to have more discussion of how general the software is, and why the current feature set was chosen. For example, how well does RatInABox deal with environments of arbitrary shape? T-mazes? It might help illustrate the tool's generality to move some of the examples in supplementary figure to main text - or just summarize them in a main text figure/panel.

Thank you for this question. Since the initial submission of this manuscript RatInABox has been upgraded and environments have become substantially more "general". Environments can now be of arbitrary shape (including T-mazes), boundaries can be curved, they can contain holes and can also contain objects (0-dimensional points which act as visual cues). A few examples are showcased in the updated figure 1 panel e.

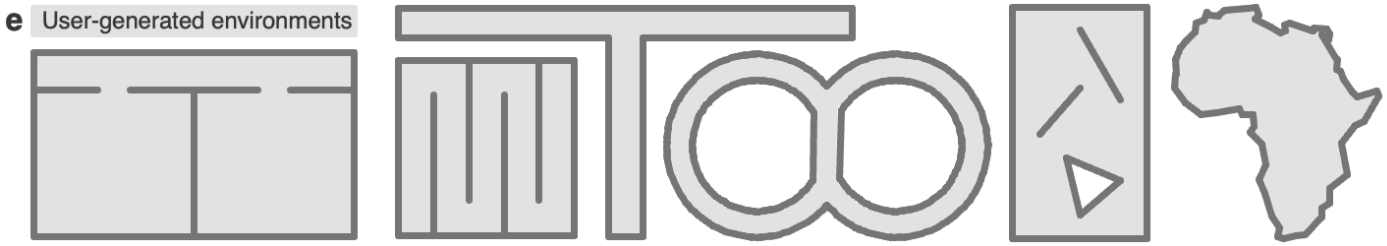


Fig. 1e: Users can easily construct complex Environments by defining boundaries and placing walls, holes and objects. Six example Environments, some chosen to replicate classic experimental set-ups, are shown here.

To further illustrate the tools generality beyond the structure of the environment we continue to summarise the reinforcement learning example (Fig. 3e) and neural decoding example in section 3.1. In addition to this we have added three new panels into figure 3 highlighting new features which, we hope you will agree, make RatInABox significantly more powerful and general and satisfy your suggestion of clarifying utility and generality in the manuscript directly. These include:

- Fig. 3b: Demonstration of this ability to control policy with a general control signal allowing users to generate arbitrarily complex motion trajectories

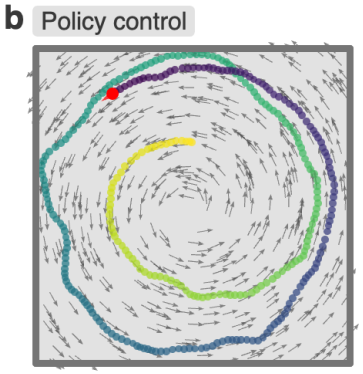


Fig. 3b: Movement can be controlled by a user-provided "drift velocity" enabling arbitrarily complex motion trajectories to be generated. Here we demonstrate how circular motion can be achieved by setting a drift velocity (grey arrows) which is tangential to the vector from the centre of the Environment to the Agent's position.

- Fig. 3c: New cell classes use egocentric boundary-, object- and agent-vector cells to efficiently encode what is in the agents "field of view".

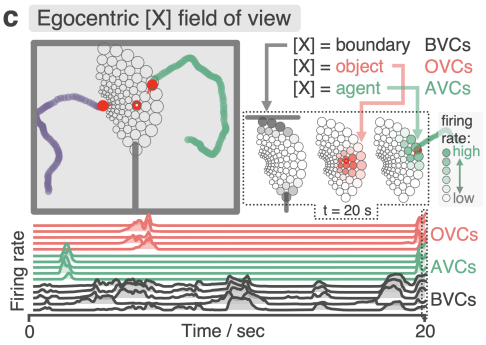


Fig. 3c: Egocentric VectorCells can be arranged to tile the Agent's field of view, providing an efficient encoding of what an Agent can 'see'. Here, two Agents explore an Environment containing walls and an object. Agent-1 (purple) is endowed with three populations of Boundary- (grey), Object- (red), and Agent- (green) selective field of view VectorCells. Each circle represents a cell, its position (in the head-centred reference frame of the Agent) corresponds to its angular and distance preferences and its shading denotes

its current firing rate. The lower panel shows the firing rate of five example cells from each population over time.

- Fig. 3d: Whilst most neurons have static receptive fields (e.g. place cells) we also provide two neuron classes whose receptive fields are defined by a parameterized function receiving the firing rate of other RatInABox neurons as inputs. In this example, grid cells are mapped through a feedforward neural network and trained to represent an arbitrary receptive field (the letter "riab"). These classes allow the construction of quite general neurons (a point we endeavour to stress in the manuscript) as well as the ability to study learning processes in the brain.

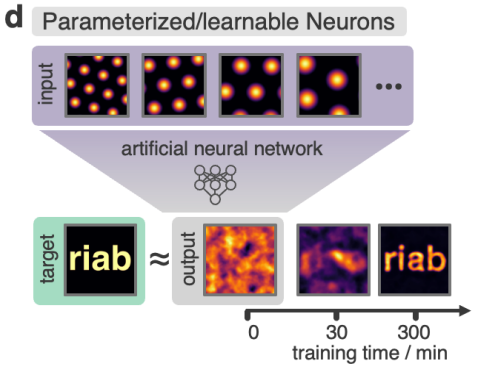


Fig. 3d: A Neurons class containing a feed forward neural network learns, from data collect online over a period of 300 minutes, to approximate a complex target receptive field from a set of grid cell inputs. This demonstrates how learning processes can be incorporated and modelled into RatInABox.

On the topic of generality, we wrote the manuscript in such a way as to demonstrate how the rich variety of ways RatInABox can be used without providing an exhaustive list of potential applications. For example, RatInABox *can* be used to study neural decoding and it *can* be used to study reinforcement learning but not because it was purpose built with these use-cases in mind. Rather because it contains a set of core tools designed to support spatial navigation and neural representations in general. For this reason we would rather keep the demonstrative examples as supplements and implement your suggestion of further raising attention to the large array of tutorials and demos provided on the GitHub repository by modifying the final paragraph of section 3.1 to read:

"Additional tutorials, not described here but available online, demonstrate how RatInABox can be used to model splitter cells, conjunctive grid cells, biologically plausible path integration, successor features, deep actor-critic RL, whisker cells and more. Despite including these examples we stress that they are not exhaustive. RatInABox provides the framework and primitive classes/functions from which highly advanced simulations such as these can be built."

> Reviewer #1 (Recommendations for the authors):
>
> + Minor
> + some copyediting could be useful: i.e. "This" on Intro line 4: antecedent?

We have replaced "This" with "This data" to clarify

> + Not every experimentalist will know what "one-hot" means

We removed this term. The sentence now reads "...or biologically unrealistic "gridworlds" with tabularised state spaces."

> + Claim in intro is overstated, that this data can be used to rapidly prove or disprove theoretical propositions - can you be more specific?

We relaxed this claim, rewording it to

“Not only is this data more cost-effective, quicker to acquire, and less resource-intensive than experimental data (no rats required), but it also offers the advantage of being flexibly hand-designed to support the validation or refutation of theoretical propositions.”

> **Reviewer #2 (Public Review):**
>
> George and colleagues present a novel open-source toolbox to model rodent locomotor patterns and electrophysiological responses of spatially modulated neurons, such as hippocampal "place cells". The present manuscript describes a comprehensive Python package ("RatInABox") with powerful capabilities to simulate a variety of environments, exploratory behaviors and concurrent responses of a variety of cell types. In addition, they provide the tools to expand these basics functions and potentially multiple different model designs, new cell types or more complex neural network architectures. The manuscript also illustrated several simple application cases. The authors have also created a comprehensive GitHub repository with more detailed explanations, tutorials and example scripts. Overall, I found both the manuscript and associated repository very clear, well written and easy the scrips easy to follow and implement, to a superior level of many commercial software packages. RatInABox fills several existing gaps in the literature and features important improvements over previous approaches; for example, the implementation of continuous 2D environments instead of tabularized state spaces. I believe this toolbox will be of great interest for many researchers in the field of spatial navigation and beyond and provide them with a remarkably powerful and flexible tool. I don't have any major issues with the manuscript. However, the manuscript can be further improved by clarifying some aspects of the toolbox, discussing its limitations and biological plausibility.
>
> Reviewer #2 (Recommendations for the authors):
>
> - With a few exceptions, the manuscripts lacks a comparison between the RatInABox and previous approaches to simulate behavioral and electrophysiological data. Such comparison can be added to the Discussion and will help many readers to appreciate the novelty and capabilities of this toolbox.

We have added the following paragraph to the discussion to contrast RatInABox to previous approaches and highlight assumptions/limitations of the package:

“Our package is not the first to model neural data[37, 38, 39] or spatial behaviour[40, 41], yet it distinguishes itself by integrating these two aspects within a unified, lightweight framework. The modelling approach employed by RatInABox involves certain assumptions:

1. It does not engage in the detailed exploration of biophysical[37, 39] or biochemical[38] aspects of neural modelling, nor does it delve into the mechanical intricacies of joint and muscle modelling[40, 41]. While these elements are crucial in specific scenarios, they demand substantial computational resources and become less pertinent in studies focused on higher-level questions about behaviour and neural representations.
2. A focus of our package is modelling experimental paradigms commonly used to study spatially modulated neural activity and behaviour in rodents. Consequently, environments are currently restricted to being two-dimensional and planar, precluding the exploration of three-dimensional settings. However, in principle, these limitations can be relaxed in the future.
3. RatInABox avoids the oversimplifications commonly found in discrete modelling, predominant in reinforcement learning[22, 23], which we believe impede its relevance to neuroscience.
4. Currently, inputs from different sensory modalities, such as vision or olfaction, are not explicitly considered. Instead, sensory input is represented implicitly through efficient allocentric or egocentric representations. If necessary, one could use the RatInABox API in conjunction with a third-party computer graphics engine to circumvent this limitation.

5. Finally, focus has been given to generating synthetic data from steady-state systems. Hence, by default, agents and neurons do not explicitly include learning, plasticity or adaptation. Nevertheless we have shown that a minimal set of features such as parameterised function-approximator neurons and policy control enable a variety of experience-driven changes in behaviour the cell responses[42, 43] to be modelled within the framework.

In conclusion, while no single approach can be deemed the best, we believe that RatInABox's unique positioning makes it highly suitable for normative modelling and NeuroAI. We anticipate that it will complement existing toolkits and represent a significant contribution to the computational neuroscience toolbox."

> - An important strength of the toolbox is its capability to simulate with ease realistic animal exploratory patterns. In comparison, the simulation of electrophysiological cell responses is more simplistic. It would be useful to better describe the assumptions and limitations taken in the simulations of cell types and briefly compare them with the well-known experimental evidence. For example, place fields are pre-determined by the model's parameters and completely stable. However, it is well known that place fields developed with experience (e.g., change locations and sizes, pop in/out). The paper claims that it can "concurrently simulate neuronal activity data" in "configurable" environments. It should be clarified if this model can capture the developmental part of place cells, or only the "stable" state.

Assumptions and limitations are described now in the discussion as shown in the section above including a statement clarifying that default cells are steady-state but evolving receptive fields can be modelled too.

RatInABox default cell classes (like PlaceCells, GridCells) are, as you point out, "steady-state", so don't by default model adaptation, remapping etc. however these processes can be modelled within the framework quite easily. For example a demo[4] is included which shows how to model successor representations (a popular model for place cells) and shows them "evolving" over time. Another demo shows how to build splitter cells[1] which fire depend on the agents history. Additionally, PlaceCells now have a "remap()" method (released in v1.6.0) which shuffles their receptive fields. All said, we are confident RatInABox can easily extend to modelling learning processes in the brain and non-static receptive fields even if default cell types don't include these a priori. Minor text changes throughout the manuscript highlight the generality/extensibility of RatInABox and the features which enable this. Additionally the following paragraph in section 3.1 was modified to clarify this:

"Additional tutorials, not described here but available online, demonstrate how RatInABox can be used to model splitter cells, conjunctive grid cells, biologically plausible path integration, successor features, deep actor-critic RL, whisker cells and more. Despite including these examples we stress that they are not exhaustive. RatInABox provides the framework and primitive classes/functions from which highly advanced simulations such as these can be built."

Finally, RatInABox is open source. This means users can always write their own cell classes which can be arbitrarily complex, time varying etc. This does not merely kick the can down the road. By writing bespoke cells which follow our minimal API they can then immediately benefit from the rest of the RatInABox package functionality (complex Environments, random motion model, plotting functions, multilayer model building etc.). We already know of many users taking this approach of using RatInABox API to construct their own more advanced cell types and we support this.

> As another example, the authors showed that the decoding error of Agent position is almost 0 with >= 20 place cells (Figure S1c), which is significantly less than that using real neural data. At least,

> there should be some discussion of where this difference may arise. Can noise be added to cell tuning curves? E.g. in the form of out-of-field spikes or trial-by-trial variability.

It is absolutely true that this decoding error from our highly simplified analysis is significantly lower than what would be achieved using a comparable number of *real* neurons. There are many reasons for this: real neurons use spikes not rates, real neurons are noisy and neurons may be jointly encoding multiple variables at once, not just position. All of these can be modelled in RatInABox. For example users could extract spikes and train a spike-based decoder. Following your comment noise can now be natively added to all cell types (new as of v1.1.0) as described in a new methods subsection (see 6.3). Mixed selective Neurons can easily be built using function approximator classes as now discussed in the manuscript and demonstrated in a tutorial[6].

As you recommended, we have added the following sentence to the supplementary section to discuss where these differences arise and how they can be studied within the framework.

"Decoding errors in Fig. S1c are smaller than would be expected if one decoded from equivalently sized populations of real hippocampal neurons. There are likely many reasons for this. Real neurons are noisy, communicate sparsely through spikes rather than rates and, most likely, jointly encode position and many other behaviourally relevant (or irrelevant) variables simultaneously. All of these factors could be straightforwardly incorporated into this analysis using existing RatInABox functionality."

On the topic of generality we have been working to make RatInABox more versatile. An illustrative case in point is the new NeuralNetworkNeurons class. These neurons take *other* neurons as inputs and map them through a small (pytorch) neural network embedded within them. Beyond this they follow the standard RatInABox API and can be used exactly as any other cell would be. In a new panel of figure 3 (and an associated demo notebook[7]) we demonstrate these by training them to learn a non-linear target function from a set of grid cell inputs. Additionally in a different demo[3] we show how they can support deep reinforcement learning. As always, the point here is not the specifics of the example but to show that with this neurons class (and the analogous FeedForwardLayer) users are not restricted to our predefined list but can in principle create or "learn" arbitrary neural classes with their own complex receptive fields and perhaps even dynamics. This is clarified in the following text in modified/added-to section 2:

Customizable and trainable Neurons: Any single toolkit cannot contain all possible neural representations of interest. Besides, static cell types (e.g. PlaceCells, GridCells etc.) which have fixed receptive fields are limiting if the goal is to study how representations and/or behaviour are learned. RatInABox provides two solutions: Firstly, being open-source, users can write and contribute their own bespoke Neurons (instructions and examples are provided) with arbitrarily complicated rate functions.

Secondly, two types of function-approximator Neurons are provided which map inputs (the firing rate of other Neurons) to outputs (firing rate) through a parameterised function which can be hand-tuned or trained to represent an endless variety of receptive field functions including those which are mixed selective, non-linear, dynamic and non-stationary.

- FeedForwardLayer: Calculates a weighted linear combination of the input Neurons with optional bias and non-linear activation function.
- NeuralNetworkNeurons: Inputs are passed through a user-provided artificial neural network.

Naturally, function-approximator Neurons can be used to model how neural populations in the brain communicate, how neural representations are learned or, in certain cases, neural dynamics. In an online demo we show how grid cells and head direction cells can be easily combined using a FeedForwardLayer to create head-direction selective

grid cells (aka. conjunctive grid cells[10]). In Fig. 3d and associated demo GridCells provide input to a NeuralNetworkNeuron which is then trained, on data generated during exploration, to have a highly complex and non-linear receptive field. Function-approximator Neurons can themselves be used as inputs to other function-approximator Neurons allowing multi-layer and/or recurrent networks to be constructed and studied.

This panel added to figure 3 demonstrate them in action (it is exactly replicated in this demo):

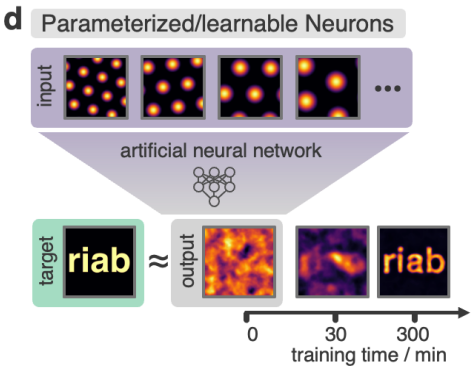


Fig. 3d: A Neurons class containing a feed forward neural network learns, from data collect online over a period of 300 minutes, to approximate a complex target receptive field from a set of grid cell inputs. This demonstrates how learning processes can be incorporated and modelled into RatInABox.

> The manuscript also lacks a description of the limitations of the approach. The authors should clarify what types of experimental data that can and cannot be modelled here, the limitations of experimental estimates from the models, and more importantly, what testable predictions of yet unknown results can be generated by this model, beyond replicating what is already known.

We already know that *"testable predictions of yet unknown results can be generated by this model"* as some early citations of the package[8,9,10] (which have been added to the manuscript) demonstrate. However, we think of RatInABox as a generator or models rather than a model itself, therefore making testable predictions is not the primary contribution and might seem out of place. We have modified the following sentence in the discussion to clarify this:

"Its user-friendly API, inbuilt data-plotting functions and general yet modular feature set mean it is well placed empower a wide variety of users to more rapidly build, train and validate models of hippocampal function[25] and spatial navigation[26], accelerating progress in the field."

As well as

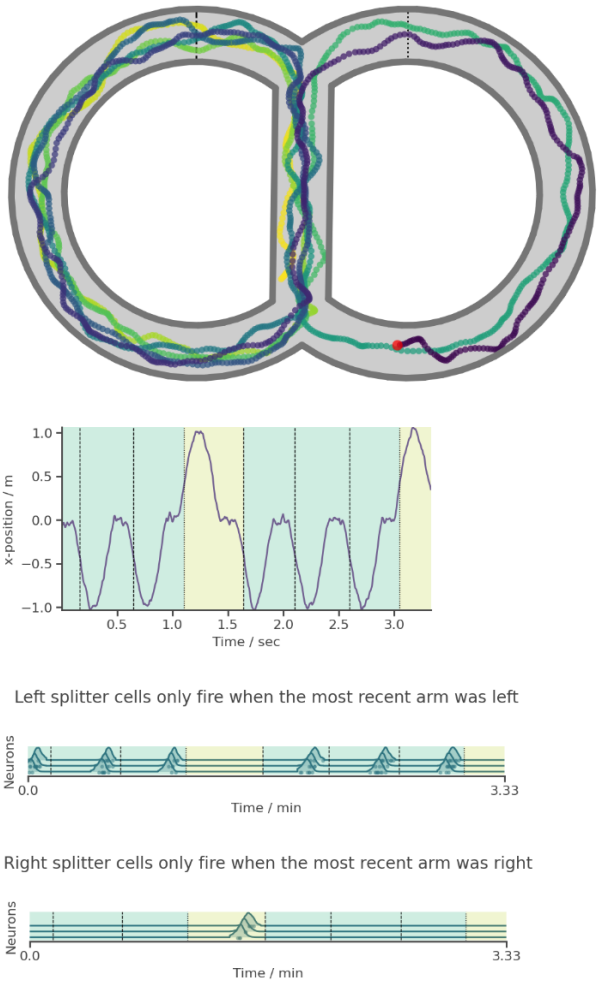
"In conclusion, while no single approach can be deemed the best, we believe that RatInABox's unique positioning makes it highly suitable for normative modelling and NeuroAI. We anticipate that it will complement existing toolkits and represent a significant contribution to the computational neuroscience toolbox."

> For example, can RatInABox be used to model episodic coding in the hippocampus (e.g., splitter cells)?, as task states can often be latent and non-Markovian. Can it be used to model the "remapping" of place cells (and other cell types) in response to changes in the environment?

All this stuff falls well within the scope of RatInABox but we'd rather not go down the rabbit hole of one-by-one including – at the level of user-facing API – all possible biological phenomena of interest. Instead the approach we take with RatInABox (which we think it's better and more scalable) is to provide a

minimal feature set and then *demonstrate* how more complex phenomena can be modelled in case studies, supplementary material and demos.

As you mentioned splitter cells we went ahead and made a very basic simulation of splitter cells which is now available as a Jupyter demo on the GitHub[1]. In this demo, the Agent goes around a figure-of-eight maze and "remembers" the last arm it passed through. A bespoke "SplitterPlaceCell" uses this to determine if a splitter cell will fire or not. Of course this is highly simplistic and doesn't model the development of splitter cells or their neural underpinnings but the point is not to provide a completely comprehensive exploration of the biological mechanisms of splitter cells, or any other phenomena for that matter, but to demonstrate that this is *within scope* and give users starting point. We also have a new demo for making conjunctive grid cells[6] by combining grid cells and head direction cells, and many more. This figure shows a snippet of the splitter cell demo.



Likewise, remapping could absolutely be modelled with RatInABox. In fact we study this in our own recent paper[8] which uses RatInABox. By the way, PlaceCells now contain a naive "remap()" method which shuffles cell locations.

We repeat here the modified paragraph in section 3.1 pointing users to these teaching materials:

"Additional tutorials, not described here but available online, demonstrate how RatInABox can be used to model splitter cells, conjunctive grid cells, biologically plausible path integration, successor features, deep actor-critic RL, whisker cells and more. Despite including these examples we stress that they are not exhaustive. RatInABox provides the framework and primitive classes/functions from which highly advanced simulations such as these can be built."

> Minor:
> - The difference between SpeedCells and VelocityCells should be better explained

SpeedCell (a single cell) encodes the speed of the agent whereas VelocityCells (there may be many) are jointly dependent on speed and direction. We have tried to make this clearer by making VelocityCells inherit from HeadDirectionCells and rewritten and reordered their descriptions to read:

"• HeadDirectionCells: Each cell has a preferred direction. The firing rate is given by a von Mises distribution centred on the preferred direction.
• VelocityCells: Like HeadDirectionCells but firing rate scales proportional to speed.
• SpeedCell: A single cell fires proportional to the scalar speed of the Agent."

The methods for these three classes have also been rewritten and should now be easier to interpret.

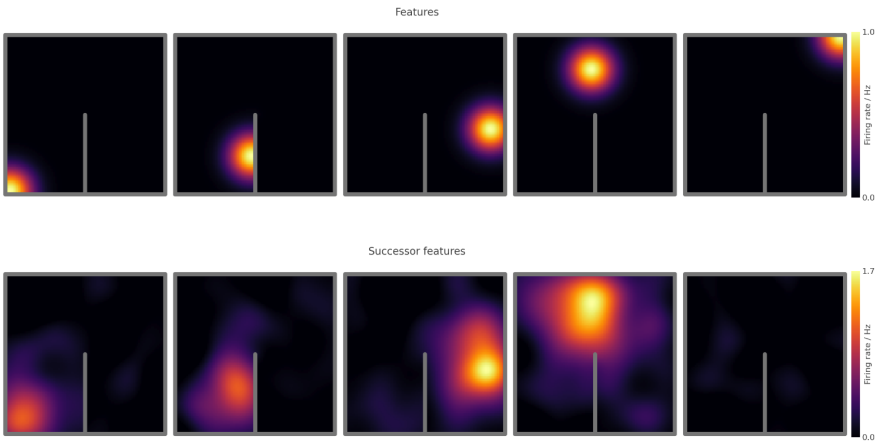
> - How is the theta LFP simulated? Is it a fix frequency or does it change with animal speed as it is typically observed?

In this case it's very simply just a fixed frequency. We'd be excited to see users contribute speed-varying theta models which would be fairly straightforward to do however we'd rather avoid going down that rabbit hole right now as we want to keep the package non-specialised.

> - Can the FeedForwardLayer be also used to simulate inhibitory inputs?

Of course, users would just need to restrict the weights to be negative. For now we don't intend to provide this functionality (i.e. Dale's law) at the level of the user-facing API but two lines of code would suffice

> - In one of their previous publications (Stachenfeld et al., NatNeurosci 2017), they modelled place cells as successor representations, which can explain the backward skewing around barriers. Here place fields rely on only the location and environmental shapes. The authors could discuss how their current model is different from and/or related to the SR model that they previously proposed.



As you point out default PlaceCells in RatInABox are static and policy-*independent* whereas successor features are policy-*dependent* and skew backwards over time to reflect a predictive function of where the agent will be in the near future. Both can be modelled with RatInABox as we now demonstrate in a tutorial[4]. The new class SuccessorFeatures takes in a set of basis features and a set of target features (these could be static PlaceCells) and linearly combines the basis features to best approximate the successor features of the target features (maths provided on the demo). The weights are learned using a variant of temporal difference learning. This figure shows a snippet from the successor feature demo.

> **Reviewer #3 (Public Review):**
>
> George et al. present a convincing new Python toolbox that allows researchers to generate synthetic behavior and neural data specifically focusing on hippocampal functional cell types (place cells, grid cells, boundary vector cells, head direction cells). This is highly useful for theory-driven research where synthetic benchmarks should be used. Beyond just navigation, it can be highly useful for novel tool development that requires jointly modeling behavior and neural data. The code is well organized and written and it was easy for us to test.
>
> We have a few constructive points that they might want to consider.
>
> - Right now the code only supports X,Y movements, but Z is also critical and opens new questions in 3D coding of space (such as grid cells in bats, etc). Many animals effectively navigate in 2D, as a whole, but they certainly make a large number of 3D head movements, and modeling this will become increasingly important and the authors should consider how to support this.

Agents now have a dedicated head direction variable (before head direction was just assumed to be the normalised velocity vector). By default this just smoothes and normalises the velocity but, in theory, could be accessed and used to model more complex head direction dynamics. This is described in the updated methods section.

In general, we try to tread a careful line. For example we embrace certain aspects of physical and biological realism (e.g. modelling environments as continuous, or fitting motion to real behaviour) and avoid others (such as the biophysics/biochemisty of individual neurons, or the mechanical complexities of joint/muscle modelling). It is hard to decide where to draw but we have a few guiding principles:

1. RatInABox is most well suited for normative modelling and neuroAI-style probing questions at the level of behaviour and representations. We consciously avoid unnecessary complexities that do not directly contribute to these domains.
2. Compute: To best accelerate research we think the package should remain fast and lightweight. Certain features are ignored if computational cost outweighs their benefit.
3. Users: If, and as, users require complexities e.g. 3D head movements, we will consider adding them to the code base.

For now we believe proper 3D motion is out of scope for RatInABox. Calculating motion near walls is already surprisingly complex and to do this in 3D would be challenging. Furthermore all cell classes would need to be rewritten too. This would be a large undertaking probably requiring rewriting the package from scratch, or making a new package RatInABox3D (BatInABox?) altogether, something which we don't intend to undertake right now. One option, if users really needed 3D trajectory data they could quite straightforwardly simulate a 2D Environment (X,Y) and a 1D Environment (Z) independently. With this method (X,Y) and (Z) motion would be entirely independent which is of unrealistic but, depending on the use case, may well be sufficient.

Alternatively, as you said that many agents effectively navigate in 2D but show complex 3D head and other body movements, RatInABox could interface with and feed data downstream to other softwares (for example Mujoco[11]) which specialise in joint/muscle modelling. This would be a very legitimate use-case for RatInABox.

We've flagged all of these assumptions and limitations in a new body of text added to the discussion:

"Our package is not the first to model neural data[37, 38, 39] or spatial behaviour[40, 41], yet it distinguishes itself by integrating these two aspects within a unified, lightweight framework. The modelling approach employed by RatInABox involves certain assumptions:

1. It does not engage in the detailed exploration of biophysical[37, 39] or biochemical[38] aspects of neural modelling, nor does it delve into the mechanical intricacies of joint and muscle modelling[40, 41]. While these elements are crucial in specific scenarios, they demand substantial computational resources and become less pertinent in studies focused on higher-level questions about behaviour and neural representations.
2. A focus of our package is modelling experimental paradigms commonly used to study spatially modulated neural activity and behaviour in rodents. Consequently, environments are currently restricted to being two-dimensional and planar, precluding the exploration of three-dimensional settings. However, in principle, these limitations can be relaxed in the future.
3. RatInABox avoids the oversimplifications commonly found in discrete modelling, predominant in reinforcement learning[22, 23], which we believe impede its relevance to neuroscience.
4. Currently, inputs from different sensory modalities, such as vision or olfaction, are not explicitly considered. Instead, sensory input is represented implicitly through efficient allocentric or egocentric representations. If necessary, one could use the RatInABox API in conjunction with a third-party computer graphics engine to circumvent this limitation.
5. Finally, focus has been given to generating synthetic data from steady-state systems. Hence, by default, agents and neurons do not explicitly include learning, plasticity or adaptation. Nevertheless we have shown that a minimal set of features such as parameterised function-approximator neurons and policy control enable a variety of experience-driven changes in behaviour the cell responses[42, 43] to be modelled within the framework.

> - What about other environments that are not "Boxes" as in the name - can the environment only be a Box, what about a circular environment? Or Bat flight? This also has implications for the velocity of the agent, etc. What are the parameters for the motion model to simulate a bat, which likely has a higher velocity than a rat?

Thank you for this question. Since the initial submission of this manuscript RatInABox has been upgraded and environments have become substantially more "general". Environments can now be of arbitrary shape (including circular), boundaries can be curved, they can contain holes and can also contain objects (0-dimensional points which act as visual cues). A few examples are showcased in the updated figure 1 panel e.

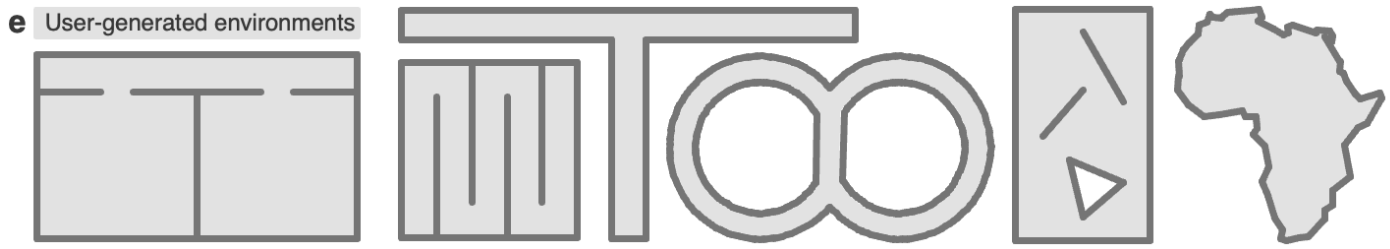


Fig. 1e: Users can easily construct complex Environments bydefining boundaries and placing walls, holes and objects. Six example Environments, some chosen to replicate classic experimental set-ups, are shown here.

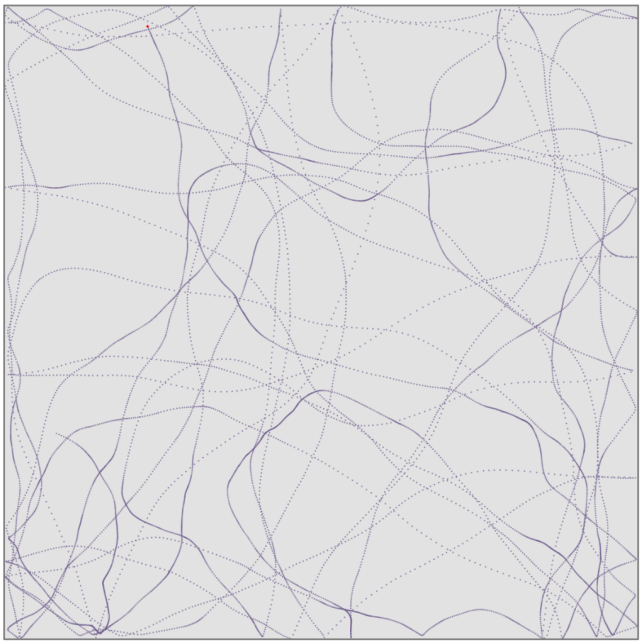
Whilst we don't know the exact parameters for bat flight users could fairly straightforwardly figure these out themselves and set them using the motion parameters as shown in the table below. We would guess that bats have a higher average speed (speed_mean) and a longer decoherence time due to increased

inertia (speed_coherence_time), so the following code might roughly simulate a bat flying around in a 10 x 10 m environment. The table shows all Agent parameters which can be set to vary the random motion model.

```python
Env = Environment(params = {'scale':10})
Bat = Agent(Env,
            params = {
                'dt':0.01,
                'speed_mean':5,
                'speed_coherence_time':1,
            })

while Bat.t < 60:
    Bat.update()

fig, ax = Bat.plot_trajectory(framerate=100)
```



Agent()				
dt	`dt`	Time discretisation step size (s).	0.01	$\mathbb{R}^+$
τ_v	`speed_coherence_time`	Timescale over which speed (1D or 2D) decoheres under random motion (s).	0.7	$\mathbb{R}^+$
σ_v (2D) μ_v (1D)	`speed_mean`	2D: Scale Rayleigh distribution scale parameter for random motion in 2D. 1D: Normal distribution mean for random motion in 1D (ms^{-1}).	0.08	2D: $\mathbb{R}^+$ 1D: $\mathbb{R}$
σ_v	`speed_std`	Normal distribution standard deviation for random motion in 1D (ms^{-1}).	0.08	$\mathbb{R}^+$
τ_ω	`rotational_velocity_coherence_time`	Rotational velocity decoherence timescale under random motion (s).	0.08	$\mathbb{R}^+$
σ_ω	`rotational_velocity_std`	Rotational velocity Normal distribution standard deviation (rad s^{-1}).	$2\pi/3$	$\mathbb{R}^+$
λ_{thig}	`thigmotaxis`	Thigmotaxis parameter.	0.5	$0 < \lambda_{thig} < 1$
d_{wall}	`wall_repel_distance`	Wall range of influence (m).	0.1	$\mathbb{R}^+$
s	`walls_repel_stregnth`	How strength walls repel the `Agent`. 0 = no wall repulsion.	1.0	$\mathbb{R}_0^+$
k	`drift_to_random_strength_ratio`*	How much motion is dominated by the drift velocity (if present) relative to random motion.	1.0	$\mathbb{R}_0^+$

> - Semi-related, the name suggests limitations: why Rat? Why not Agent? (But its a personal choice)

We came up with the name “RatInABox” when we developed this software to study hippocampal representations of an artificial rat moving around a closed 2D world (a box). We also fitted the random motion model to open-field exploration data from rats. You’re right that it is not limited to rodents but for better or for worse it’s probably too late for a rebrand!

> - A future extension (or now) could be the ability to interface with common trajectory estimation tools; for example, taking in the (X, Y, (Z), time) outputs of animal pose estimation tools (like DeepLabCut or such) would also allow experimentalists to generate neural synthetic data from other sources of real-behavior.

This is actually already possible via our “Agent.import_trajectory()” method. Users can pass an array of time stamps and an array of positions into the Agent class which will be loaded and smoothly interpolated along as shown here in Fig. 3a or demonstrated in these two new papers[9,10] who used RatInABox by loading in behavioural trajectories.

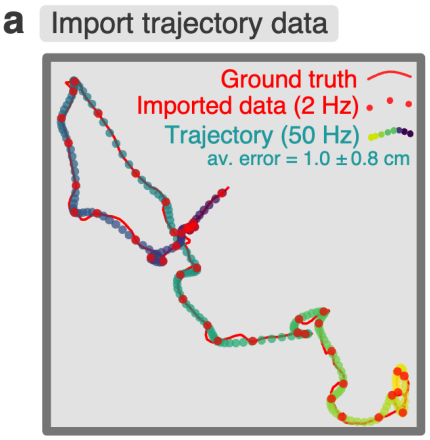
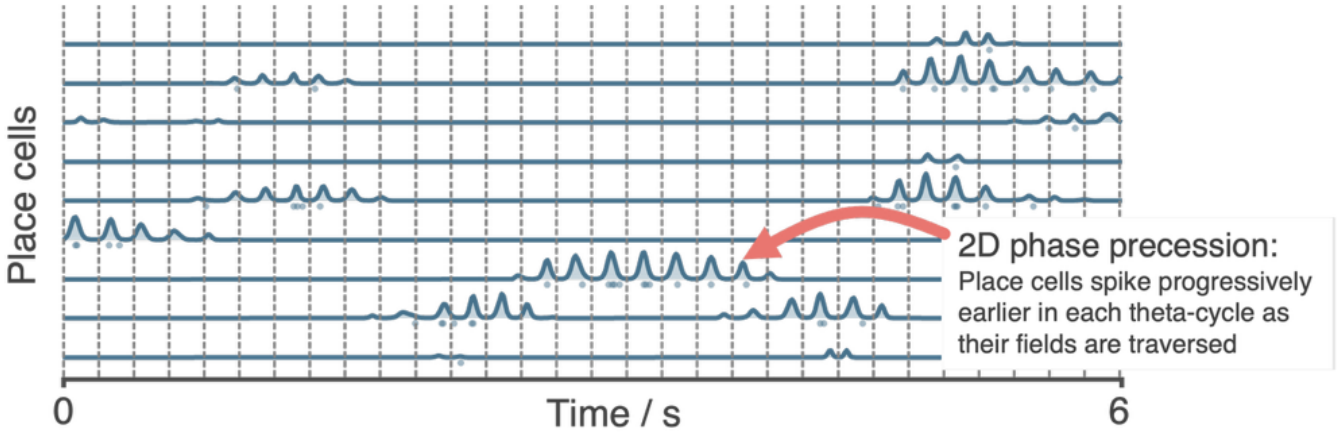


Fig. 3a: Low temporal-resolution trajectory data (2 Hz) imported into RatInABox is upsampled ("augmented") using cubic spline interpolation. The resulting trajectory is a close match to the ground truth trajectory (Sargolini et al.) from which the low resolution data was sampled.

> - What if a place cell is not encoding place but is influenced by reward or encodes a more abstract concept? Should a PlaceCell class inherit from an AbstractPlaceCell class, which could be used for encoding more conceptual spaces? How could their tool support this?

In fact PlaceCells already inherit from a more abstract class (Neurons) which contains basic infrastructure for initialisation, saving data, and plotting data etc. We prefer the solution that users can write their *own* cell classes which inherit from Neurons (or PlaceCells if they wish). Then, users need only write a new get_state() method which can be as simple or as complicated as they like. Here are two examples we've already made which can be found on the GitHub:

Phase precession: PhasePrecessingPlaceCells(PlaceCells)[12] inherit from PlaceCells and modulate their firing rate by multiplying it by a phase dependent factor causing them to "phase precess".



Splitter cells: Perhaps users wish to model PlaceCells that are modulated by recent history of the Agent, for example which arm of a figure-8 maze it just came down. This is observed in hippocampal "splitter cell". In this demo[1] SplitterCells(PlaceCells) inherit from PlaceCells and modulate their firing rate according to which arm was last travelled along.

> - This a bit odd in the Discussion: "If there is a small contribution you would like to make, please open a pull request. If there is a larger contribution you are considering, please contact the

> corresponding author3" This should be left to the repo contribution guide, which ideally shows people how to contribute and your expectations (code formatting guide, how to use git, etc). Also this can be very off-putting to new contributors: what is small? What is big? we suggest use more inclusive language.

We've removed this line and left it to the GitHub repository to describe how contributions can be made.

> - Could you expand on the run time for BoundaryVectorCells, namely, for how long of an exploration period? We found it was on the order of 1 min to simulate 30 min of exploration (which is of course fast, but mentioning relative times would be useful).

Absolutely. How long it takes to simulate BoundaryVectorCells will depend on the discretisation timestep and how many neurons you simulate. Assuming you used the default values (dt = 0.1, n = 10) then the motion model should dominate compute time. This is evident from our analysis in Figure 3f which shows that the update time for n = 100 BVCs is on par with the update time for the random motion model, therefore for only n = 10 BVCs, the motion model should dominate compute time.

So how long should this take? Fig. 3f shows the motion model takes ~10^{-3} s per update. One hour of simulation equals this will be 3600/dt = 36,000 updates, which would therefore take about 72,000*10^{-3} s = 36 seconds. So your estimate of 1 minute seems to be in the right ballpark and consistent with the data we show in the paper.

Interestingly this corroborates the results in a new inset panel where we calculated the total time for cell and motion model updates for a PlaceCell population of increasing size (from n = 10 to 1,000,000 cells). It shows that the motion model dominates compute time up to approximately n = 1000 PlaceCells (for BoundaryVectorCells it's probably closer to n = 100) beyond which cell updates dominate and the time scales linearly.

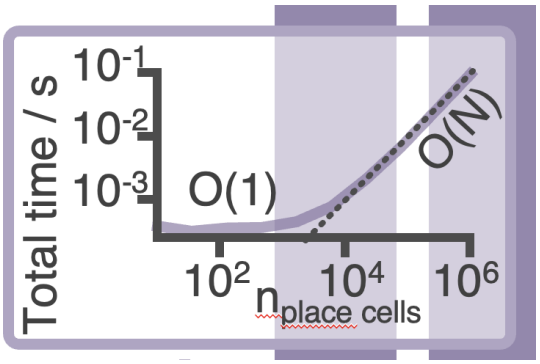


Fig 3f: Inset shows how the total update time (random motion model and place cell update) scales with the number of place cells.

These are useful and non-trivial insights as they tell us that the RatInABox neuron models are *quite efficient* relative to the RatInABox random motion model (something we hope to optimise further down the line). We've added the following sentence to the results:

"Our testing (Fig. 3f, inset) reveals that the combined time for updating the motion model and a population of PlaceCells scales sublinearly O(1) for small populations n > 1000 where updating the random motion model dominates compute time, and linearly for large populations n > 1000. PlaceCells, BoundaryVectorCells and the Agent

motion model update times will be additionally affected by the number of walls/barriers in the Environment. 1D simulations are significantly quicker than 2D simulations due to the reduced computational load of the 1D geometry."

And this sentence to section 2:

"RatInABox is fundamentally continuous in space and time. Position and velocity are never discretised but are instead stored as continuous values and used to determine cell activity online, as exploration occurs. This differs from other models which are either discrete (e.g. "gridworld" or Markov decision processes) or approximate continuous rate maps using a cached list of rates precalculated on a discretised grid of locations. Modelling time and space continuously more accurately reflects real-world physics, making simulations smooth and amenable to fast or dynamic neural processes which are not well accommodated by discretised motion simulators. Despite this, RatInABox is still fast; to simulate 100 PlaceCell for 10 minutes of random 2D motion (dt = 0.1 s) it takes about 2 seconds on a consumer grade CPU laptop (or 7 seconds for BoundaryVectorCells)."

> - Regarding the Geometry and Boundary conditions, would supporting hyperbolic distance might be useful, given the interest in alternative geometry of representations (ie, https://www.nature.com/articles/s41593-022-01212-4)?

Whilst this would be very interesting it would likely represent quite a significant edit, requiring rewriting of almost all the geometry-handling code. We're happy to consider changes like these according to (i) how simple they will be to implement, (ii) how disruptive they will be to the existing API, (iii) how many users would benefit from the change. If many users of the package request this we will consider ways to support it.

> - In general, the set of default parameters might want to be included in the main text (vs in the supplement).

We also considered this but decided to leave them in the methods for now. The exact value of these parameters are subject to change in future versions of the software. Also, we'd prefer for the main text to provide a low-detail high-level description of the software and the methods to provide a place for keen readers to dive into the mathematical and coding specifics.

> - It still says you can only simulate 4 velocity or head directions, which might be limiting.

Thanks for catching this. This constraint has been relaxed. Users can now simulate an arbitrary number of head direction cells with arbitrary tuning directions and tuning widths. The methods have been adjusted to reflect this (see section 6.3.4).

> - The code license should be mentioned in the Methods.

We have added the following section to the methods:

6.6 License

RatInABox is currently distributed under an MIT License, meaning users are permitted to use, copy, modify, merge publish, distribute, sublicense and sell copies of the software.

Reviewer response references and links

1. https://github.com/RatInABox-Lab/RatInABox/blob/main/demos/splitter_cells_example.ipynb
2. https://github.com/RatInABox-Lab/RatInABox/blob/main/demos/reinforcement_learning_example.ipynb
3. https://github.com/RatInABox-Lab/RatInABox/blob/main/demos/actor_critic_example.ipynb
4. https://github.com/RatInABox-Lab/RatInABox/blob/main/demos/successor_features_example.ipynb
5. https://ratinabox-lab.github.io/RatInABox/why-riab/testimonials.html
6. https://github.com/RatInABox-Lab/RatInABox/blob/main/demos/conjunctive_gridcells_example.ipynb
7. https://github.com/RatInABox-Lab/RatInABox/blob/main/demos/deep_learning_example.ipynb
8. Tom George, Kim Stachenfeld, Caswell Barry, Claudia Clopath, and Tomoki Fukai. A generative model of the hippocampal formation trained with theta driven local learning rules. In Thirty-seventh Conference on Neural Information Processing Systems, 2023. URL https://openreview.net/forum?id=yft4JlxsRf.
9. J. Quinn Lee, Alexandra T. Keinath, Erica Cianfarano, and Mark P. Brandon. Identifying representational structure in ca1 to benchmark theoretical models of cognitive mapping. October 2023. doi: 10.1101/2023.10.08.561112. URL http://dx.doi.org/10.1101/2023.10.08.561112.
10. Guillaume Etter, Suzanne van der Veldt, Coralie-Anne Mosser, and Sylvain Williams. A population code for idiothetic representations in the hippocampal-septal circuit. November 2023. doi: 10.1101/2023.11.10.566641. URL http: //dx.doi.org/10.1101/2023.11.10.566641.
11. https://mujoco.org/
12. https://github.com/RatInABox-Lab/RatInABox/blob/main/ratinabox/contribs/PhasePrecessingPlaceCells.py