RatInABox: A unified Python framework for modelling spatial behaviour and neural data

Caswell Barry[1,2], Mehul Rastogi[1], William de Cothi[2], Claudia Clopath[1,3], Kimberly Stachenfeld[4,5], Tom George[1]

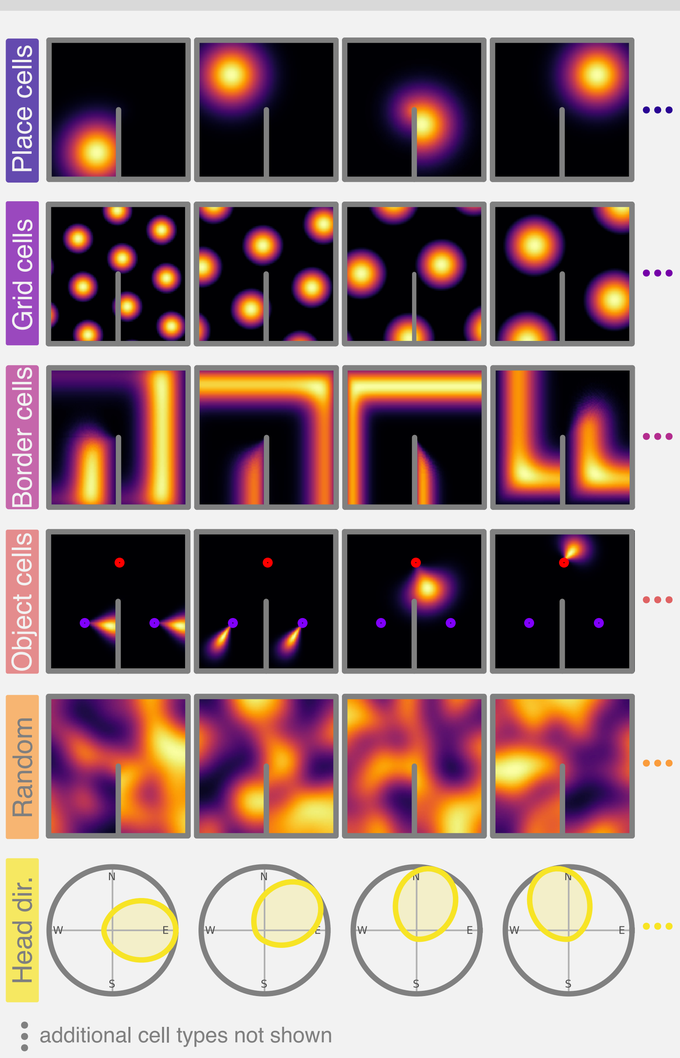
[1]Sainsbury Wellcome Centre, UCL, London [2]Dept. of Cell & Developmental Biology, UCL, London [3]Imperial College, London [4]Columbia University, New York [5]Google DeepMind, London

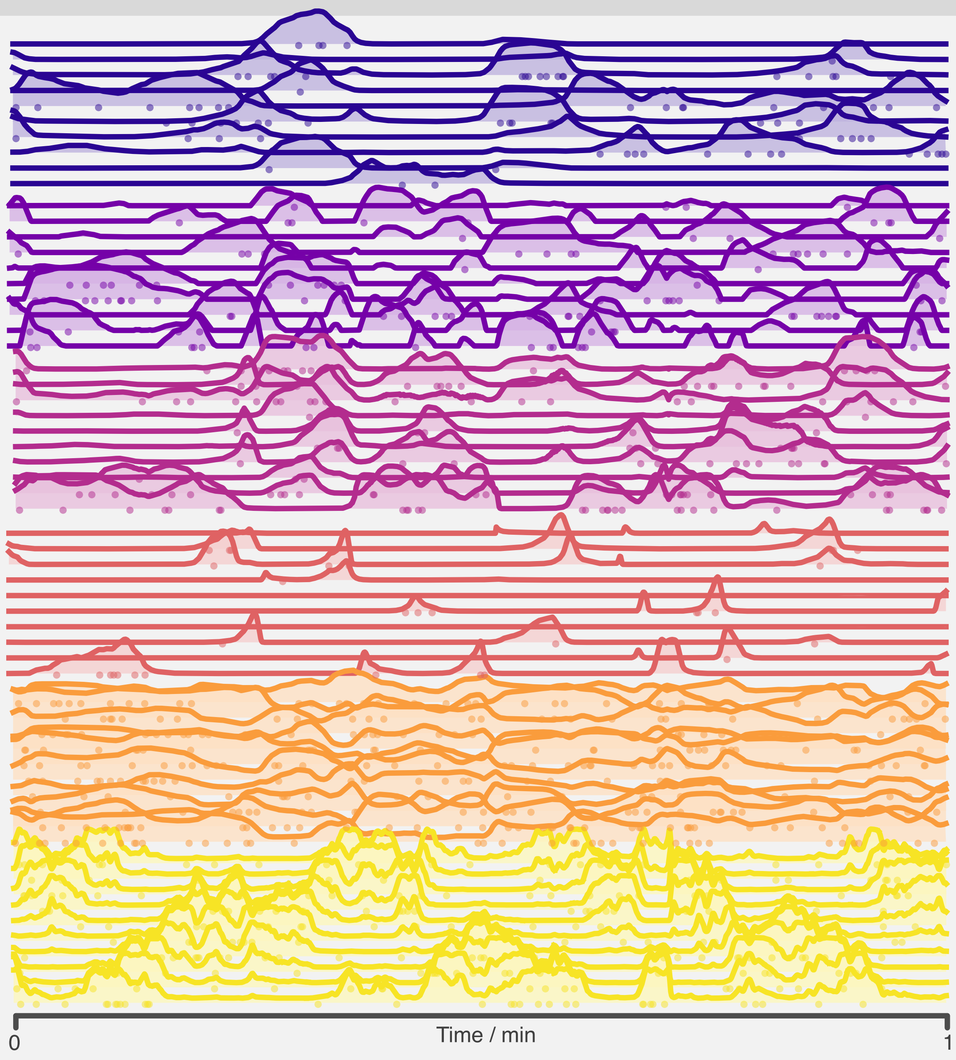
1 OVERVIEW

Navigation and the neural processes involved are complex but modelling them shouldn't be.

RatInABox is an **open-source Python package,** providing tools to help **improve**, **standardize** and **streamline** how we model navigation.

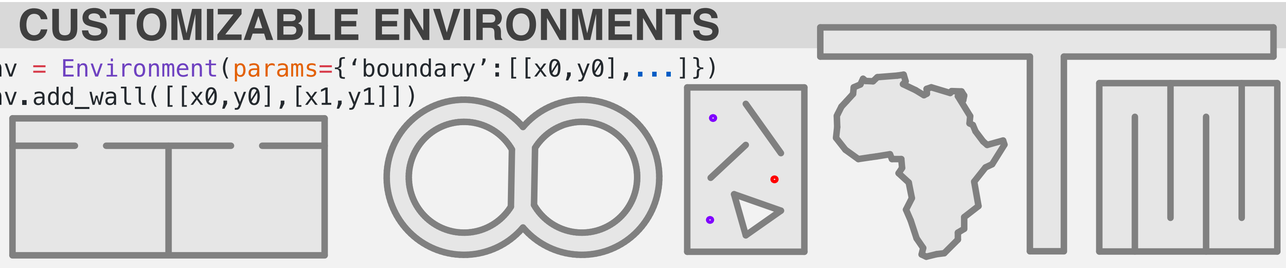
1. Construct **complex continuous environments.**
2. A framework for generating varied, **realistic, continuous motion trajectories.**
3. Efficient models of **known** and **bespoke cell types.**





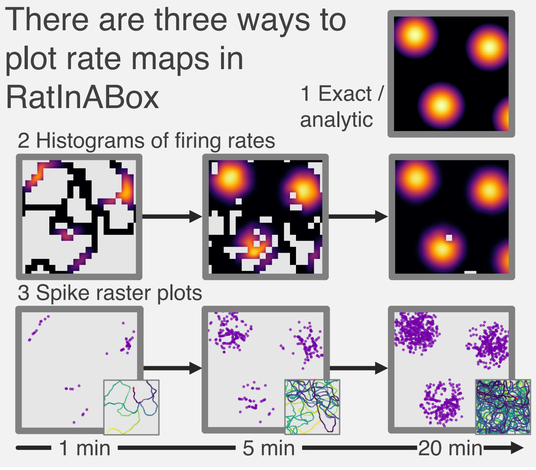
2 CUSTOMIZABLE ENVIRONMENTS

```
env = Environment(params={'boundary':[[x0,y0],...]})
env.add_wall([[x0,y0],[x1,y1]])
```



3 NEURAL RATE MAPS

There are three ways to plot rate maps in RatInABox



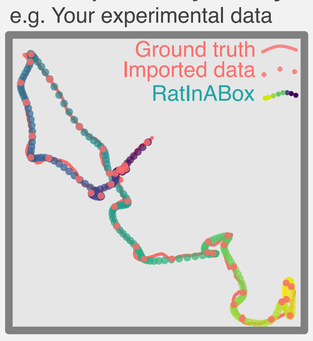
iPad

4 TRAJECTORY / POLICY CONTROL

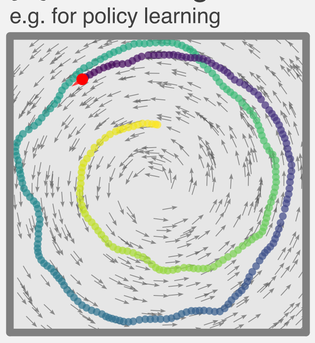
There are three ways to generate trajectories using RatInABox

Random motion
Fitted to real rodent foraging

Import trajectory
e.g. Your experimental data



Control signal
e.g. for policy learning



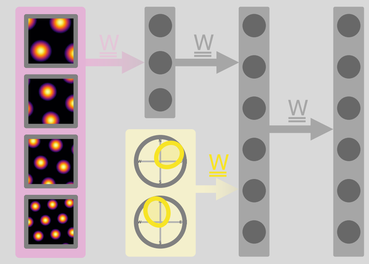
5 SIMPLE PYTHON API

```
>>> pip install ratinabox                                  # Install RatInABox

from ratinabox import Environment, Agent, PlaceCells

env = Environment(params={})                               # Default Environment
env.add_wall([[0.5,0.0],[0.5,0.5]])                        # ...add a wall
agent = Agent(env, params={})                              # Default Agent
placecells = PlaceCells(agent, params={'n':10}) # Default place cells

while agent.t < 60:                                        # Simulate (60 secs)
   agent.update()
   placecells.update()

agent.animate_trajectory()                                 # Animate trajectory
placecells.plot_rate_map()                                 # Plot rate maps
spikes = placecells.history["spikes"]                      # Extract spike data
```

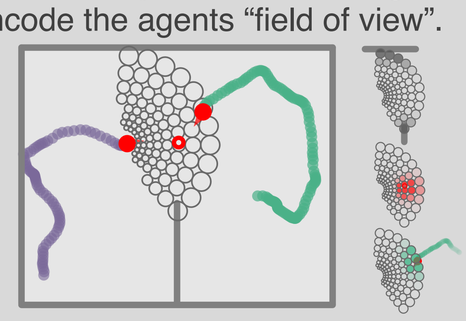
6 ADVANCED FEATURES AND MODELLING

Advanced features extend the package beyond its role as a standalone data generator. It can be used to build/simulate complex models of plasticity, learning, mixed representations, social behaviours and more…

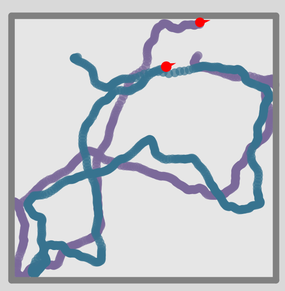
Parameterised layers
These neurons take other neurons as inputs. Inputs can be stacked indefinitely and combined to make "mixed" representations.



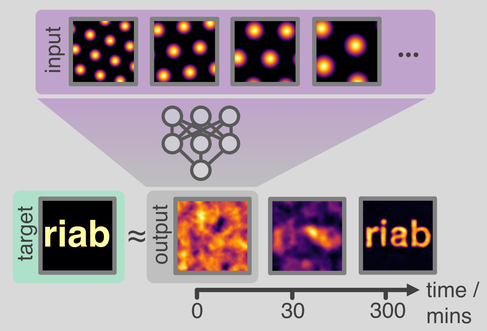
Egocentric representations
Respond to boundaries, objects and agents in a head-centred reference frame. They can be arranged to encode the agents "field of view".



Multiple Agents
In this example Agent 2 "chases" Agent 1.



Inbuilt deep learning functionality
These neurons map their inputs through a user-defined, trainable pytorch neural network.



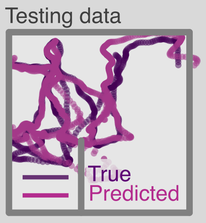
7 CASE STUDIES AND DEMOS

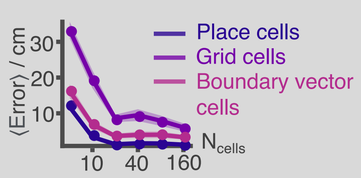
DECODING POSITION

Encoding: `BoundaryVectorCells()`
Data generation: `Random trajectory`
Model: `GP-regressor`

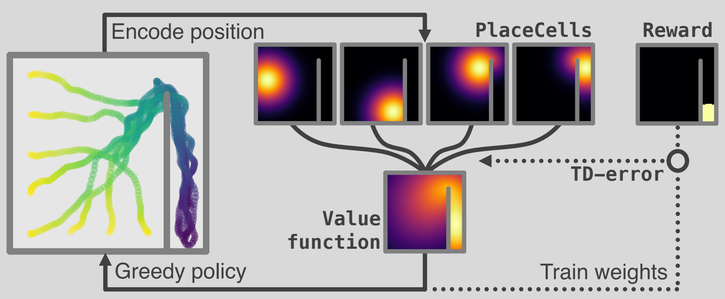
Training data

Testing data





REINFORCEMENT LEARNING



OTHER DEMOS ON…

Splitter cells, deep learning, mixed selective representations, path integration, actor-critic deep RL, successor representations, …

Github:

Publication:

Related packages: RL and discrete state spaces: Neuro-Nav, Juliani et al. (2022), Hippocampal model comparison: NeuralPlayground, Domine and Carrasco-Davis et al. (2023).

Acknowledgements: Thanks to Tom Burns, Gastón Sivori, Colleen Gillon, Peter Vincent, Ryan Young, Adam Tyson, Jeff Erlich, Quinn Lee and others for contributions and comments.

RiaB barry lab Sainsbury Wellcome Centre DeepMind CLOPATH LAB Computational Neuroscience