

An introduction to Reinforcement Learning

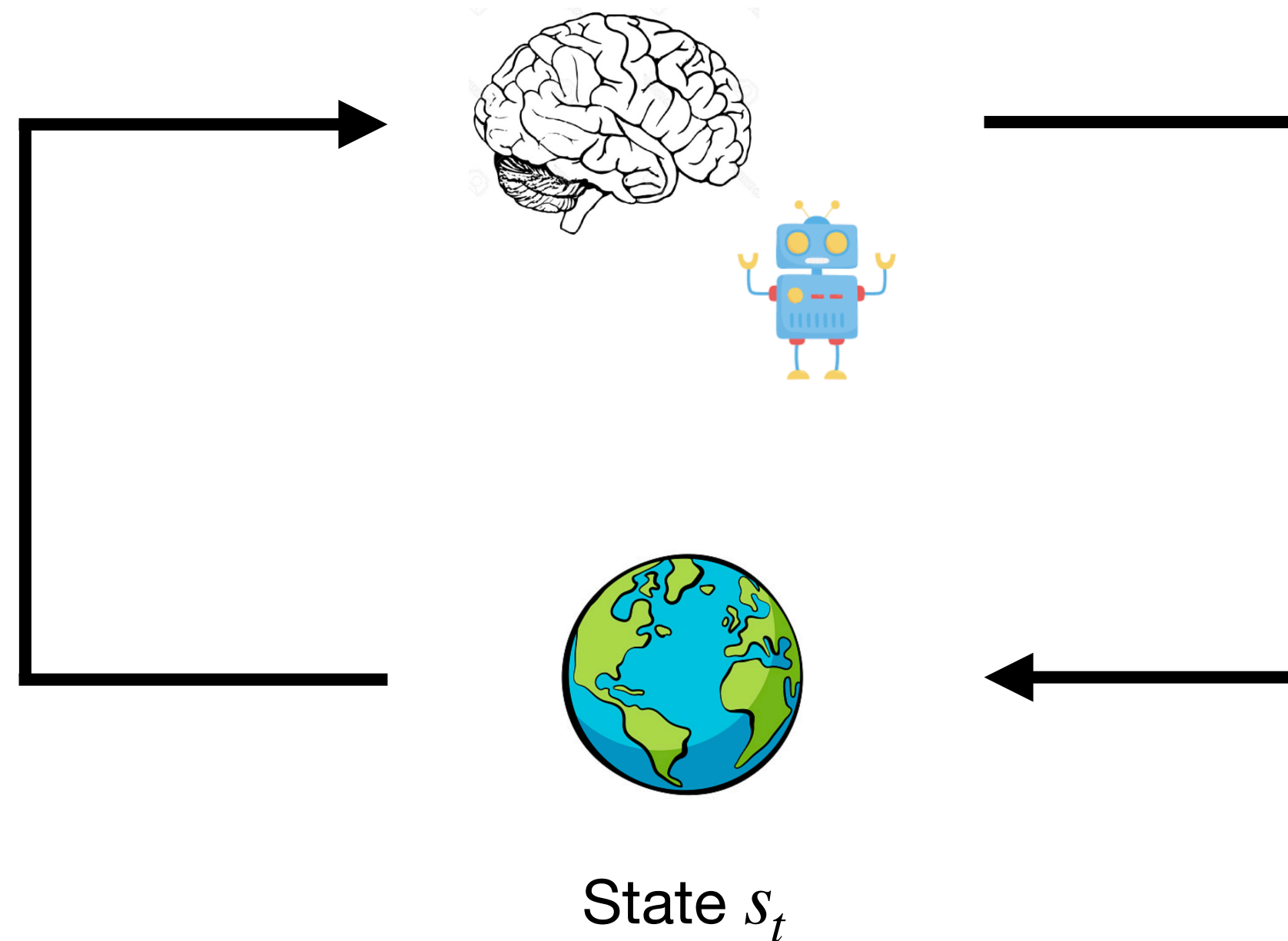
19th of July 2022

Recap: models in RL

Based on a reward signal, agents learn **values of actions/states**:

$$V_{\pi}(s) = \mathbb{E}_{\pi}[R | s_0 = s]$$

Reward r_t



Action is governed by a **policy**:

$$\pi(a, s) = P(a_t = a | s_t = s)$$

Action a_t

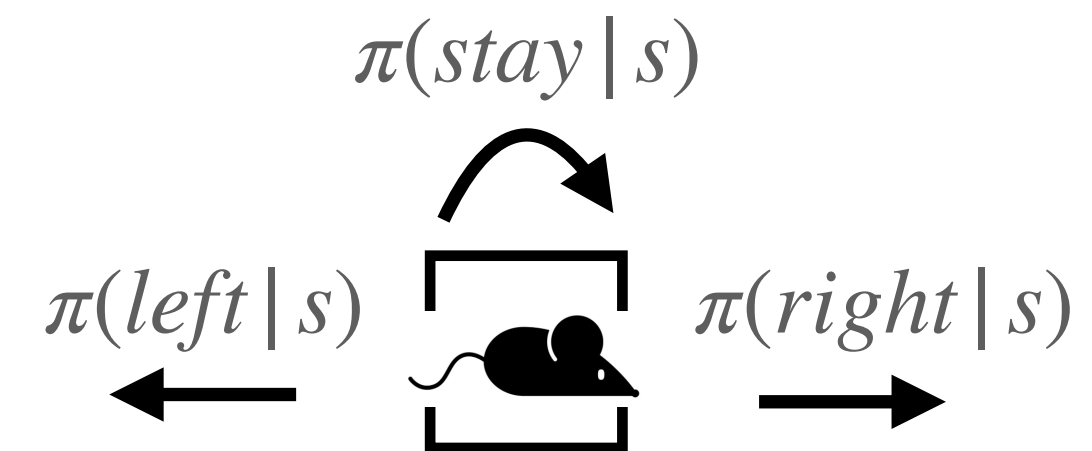
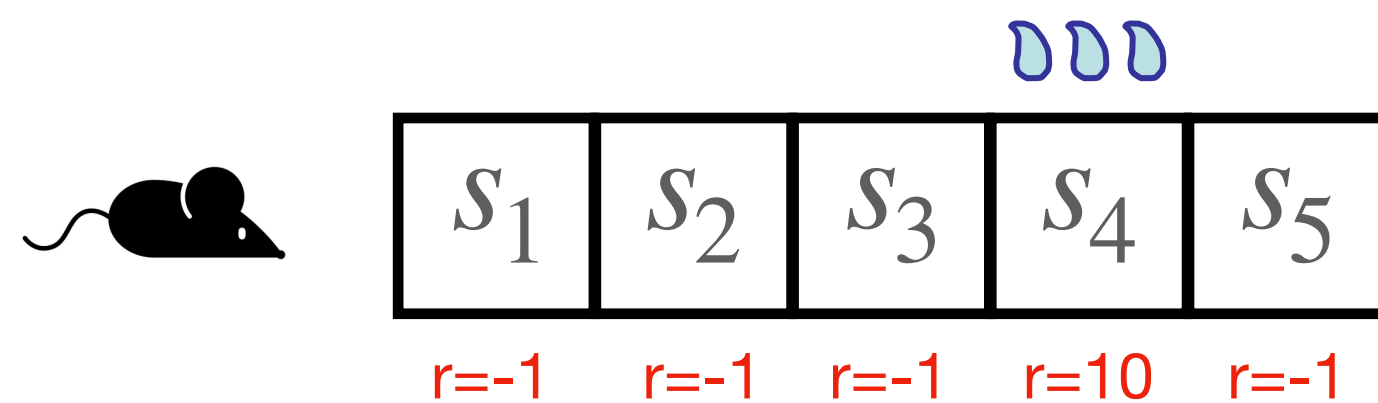
Agents can learn a **model of the environment** to make smarter decisions, e.g.:

$$P(s_{t+1} = s | s_t = s, a_t = a)$$

Recap: models in RL

Allows to specify all environment dynamics for RL problem:

$$P(s', r | s, a) = P(s_{t+1} = s', r_{t+1} = r | s_t = s, a_t = a)$$



Recap: models in RL

$$P(s', r | s, a) = P(s_{t+1} = s', r_{t+1} = r | s_t = s, a_t = a)$$

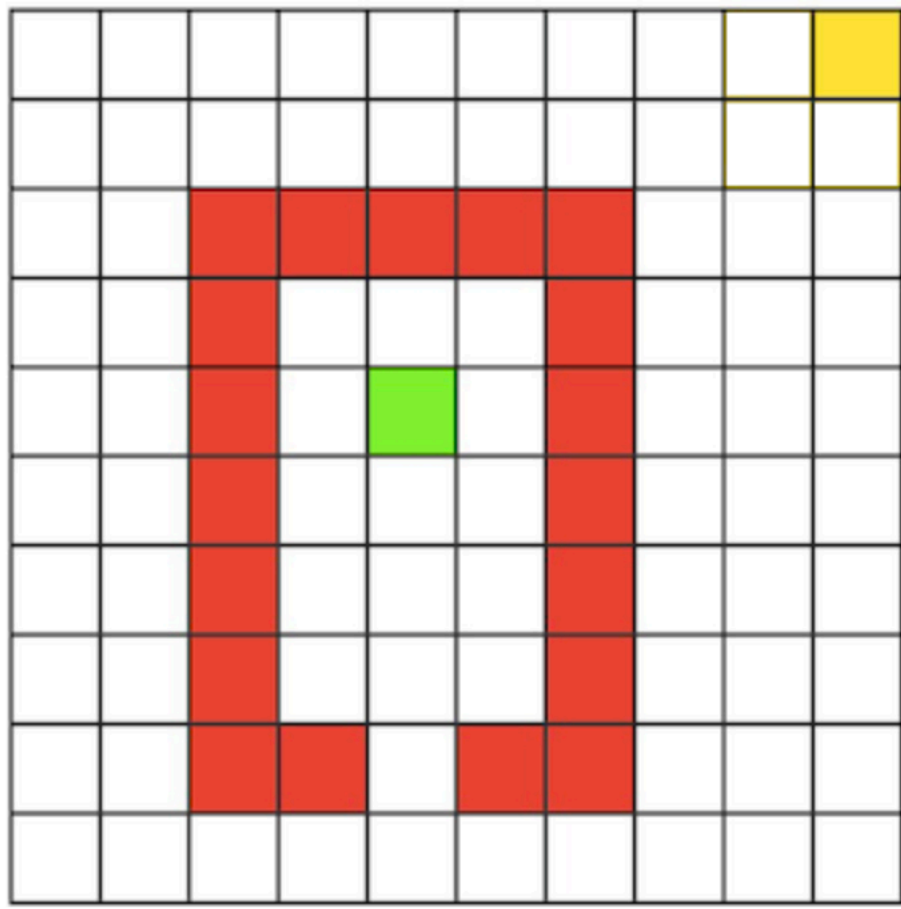
How can we make use of such models of the world?

Planning and **action selection**

Learning

- Key idea: store experiences in world model $P(s', r | s, a)$
- Sample from this model to generate extra learning data
- This is called **DYNA-Q...**

Recap: DYNA-Q



$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha (r_t + \gamma \max_a Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))$$

$$P(s', r | s, a) = P(s_{t+1} = s', r_{t+1} = r | s_t = s, a_t = a)$$

$$Model(S, A) \leftarrow R, S'$$

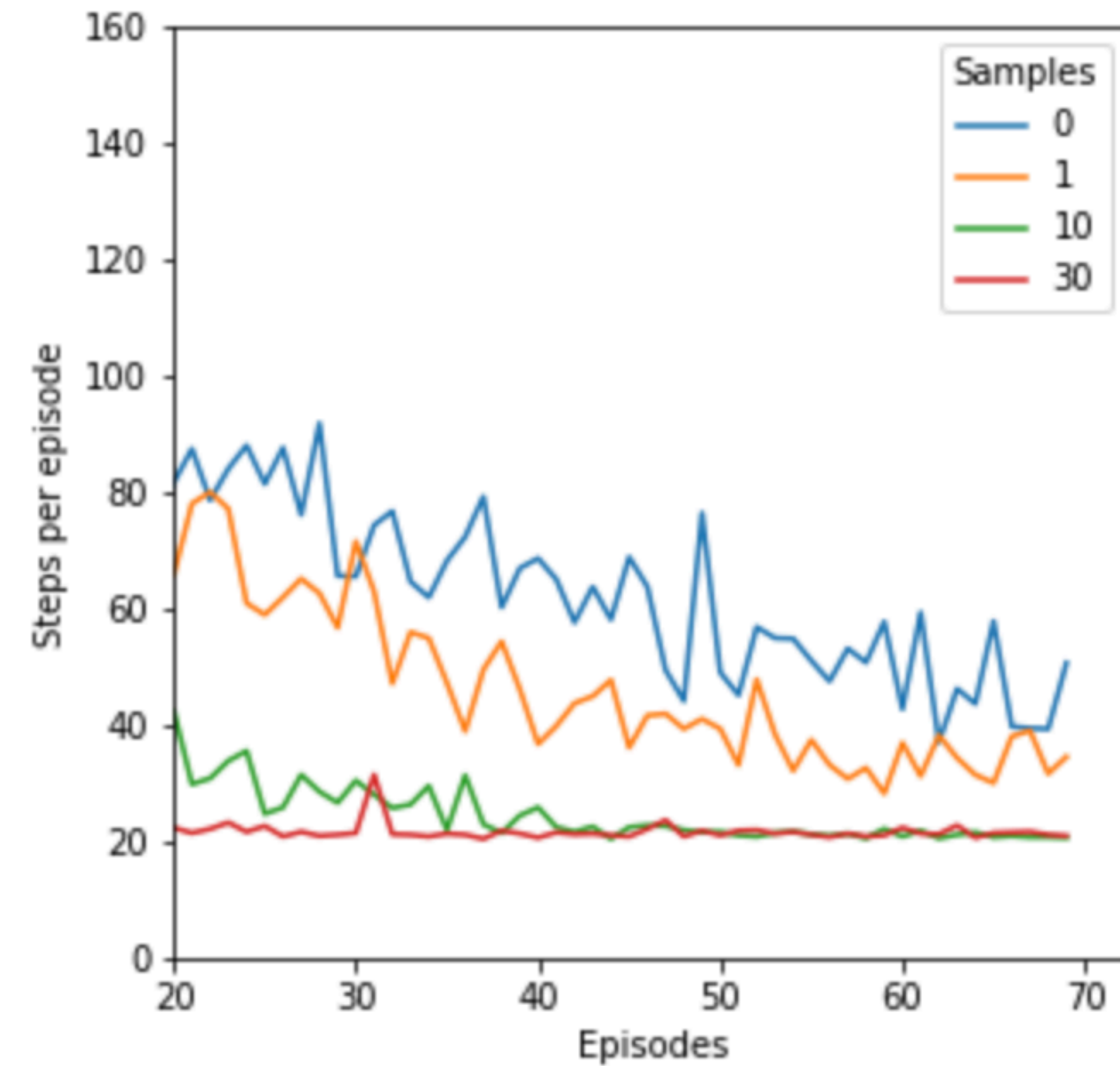
And during breaks ('offline rest'), they can sample from this experience and learn from these samples:

$S \leftarrow$ previously observed state

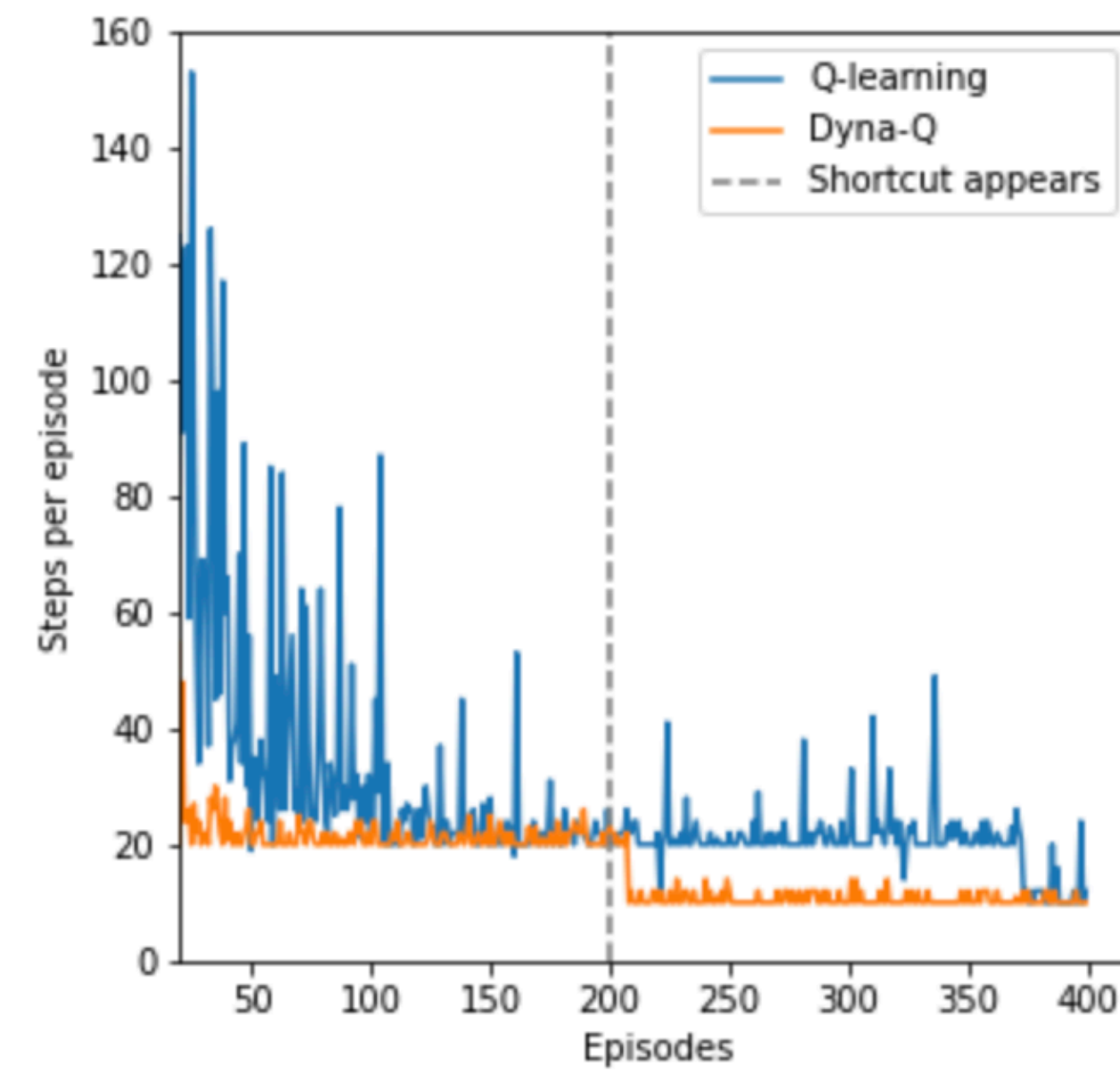
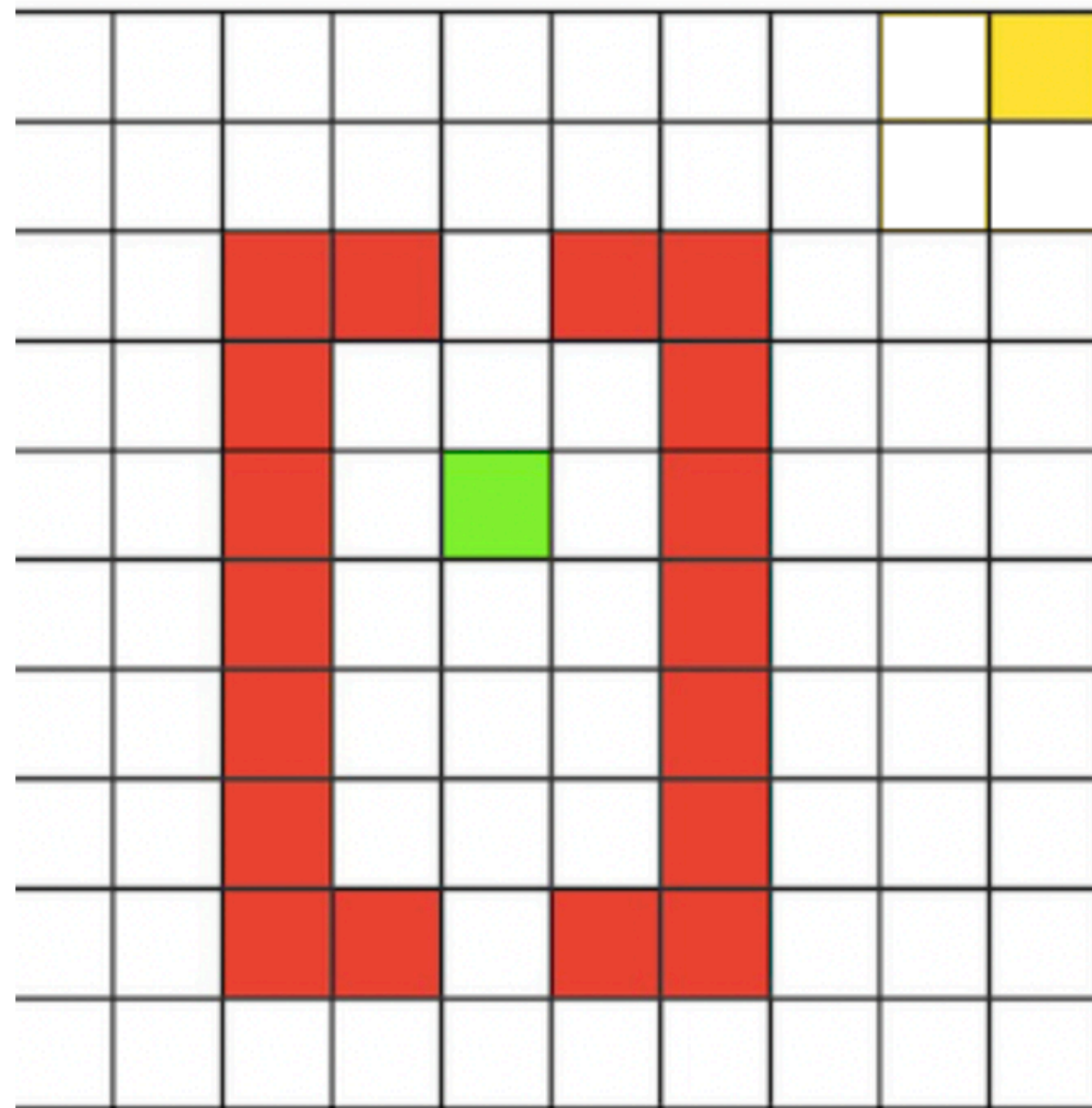
$A \leftarrow$ action previously taken in S

$R, S' \leftarrow Model(S, A)$

$$Q(S, A) \leftarrow Q(S, A) + \alpha [R + \gamma \max_a Q(S', A) - Q(S, A)]$$



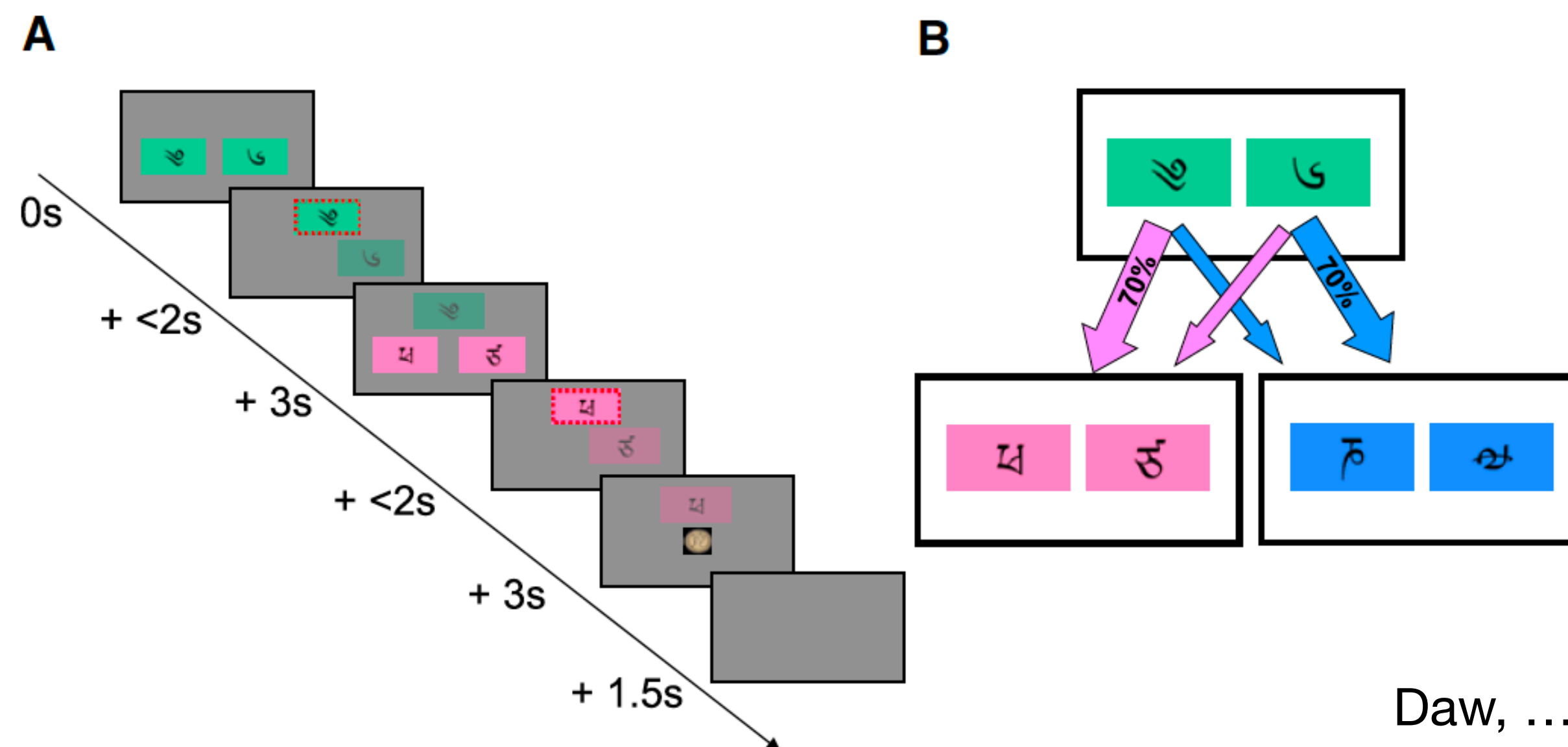
Recap: DYNA-Q



Model-free vs. Model-based control

Two-step task: one of the most iconic RL tasks

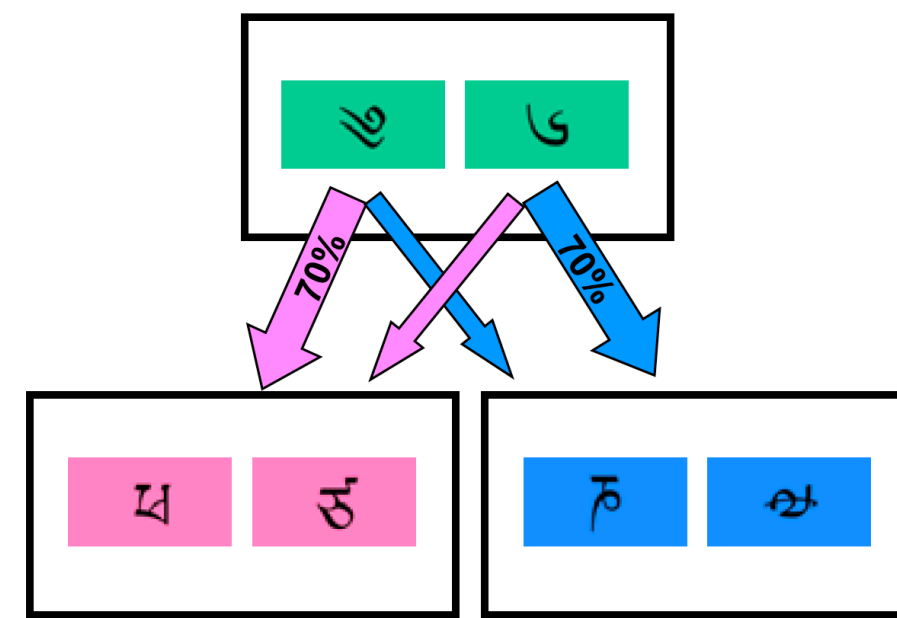
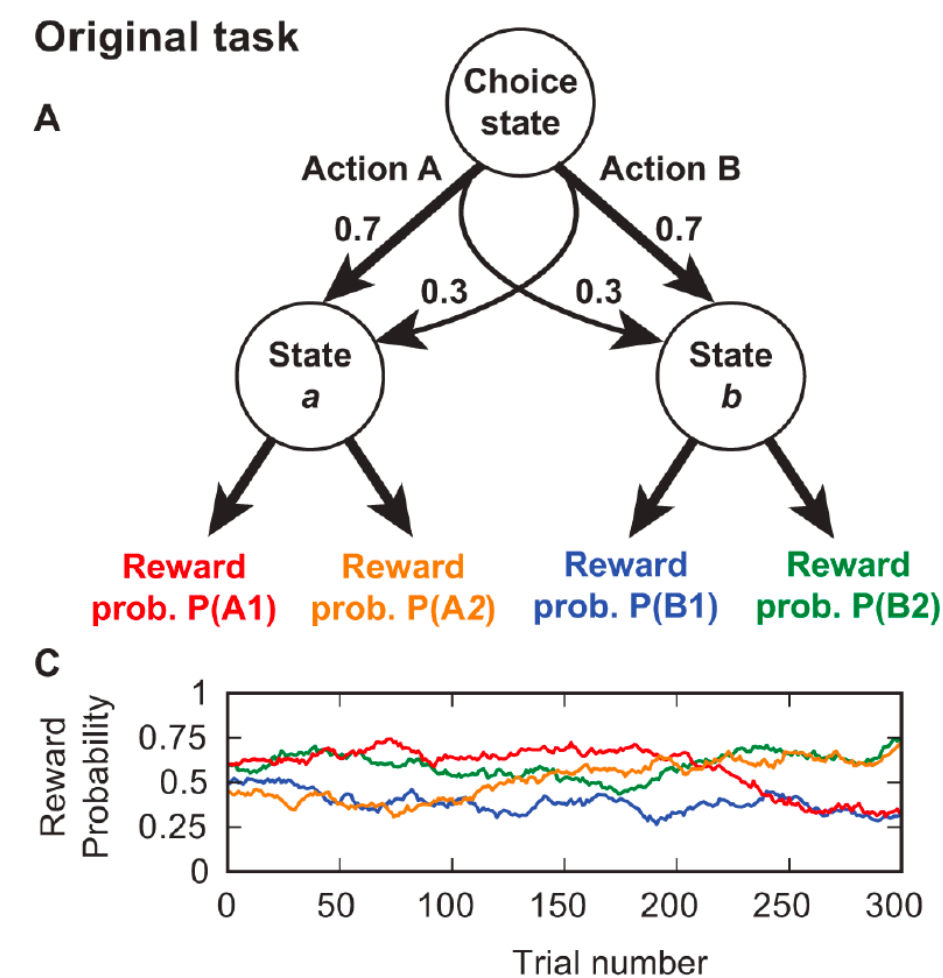
Choose twice between two options to obtain a reward



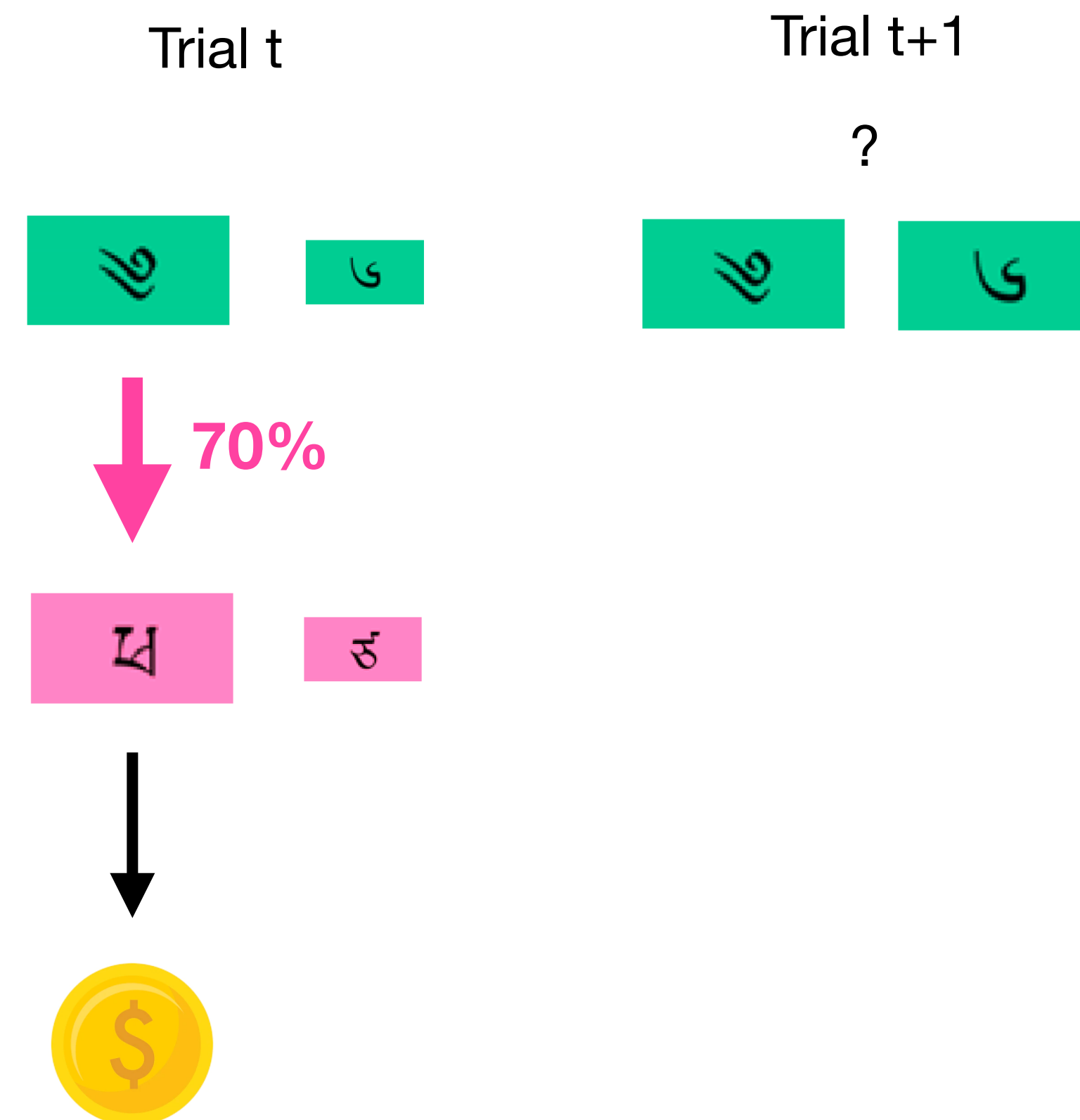
Key manipulation: **common** and **rare** transitions
- how should you update?

Daw, ..., Dolan, Neuron, 2011

Two-step task: one of the most iconic RL tasks

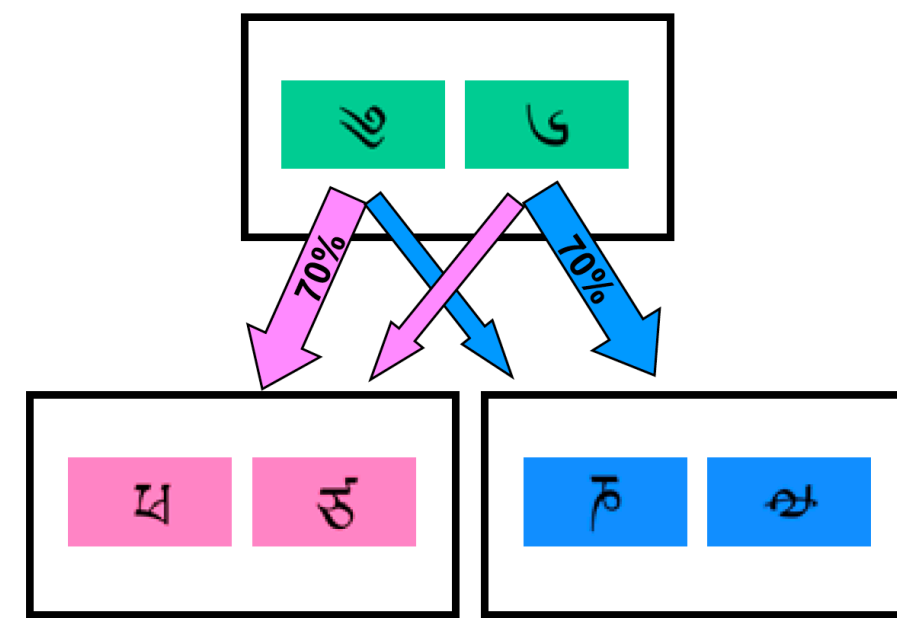
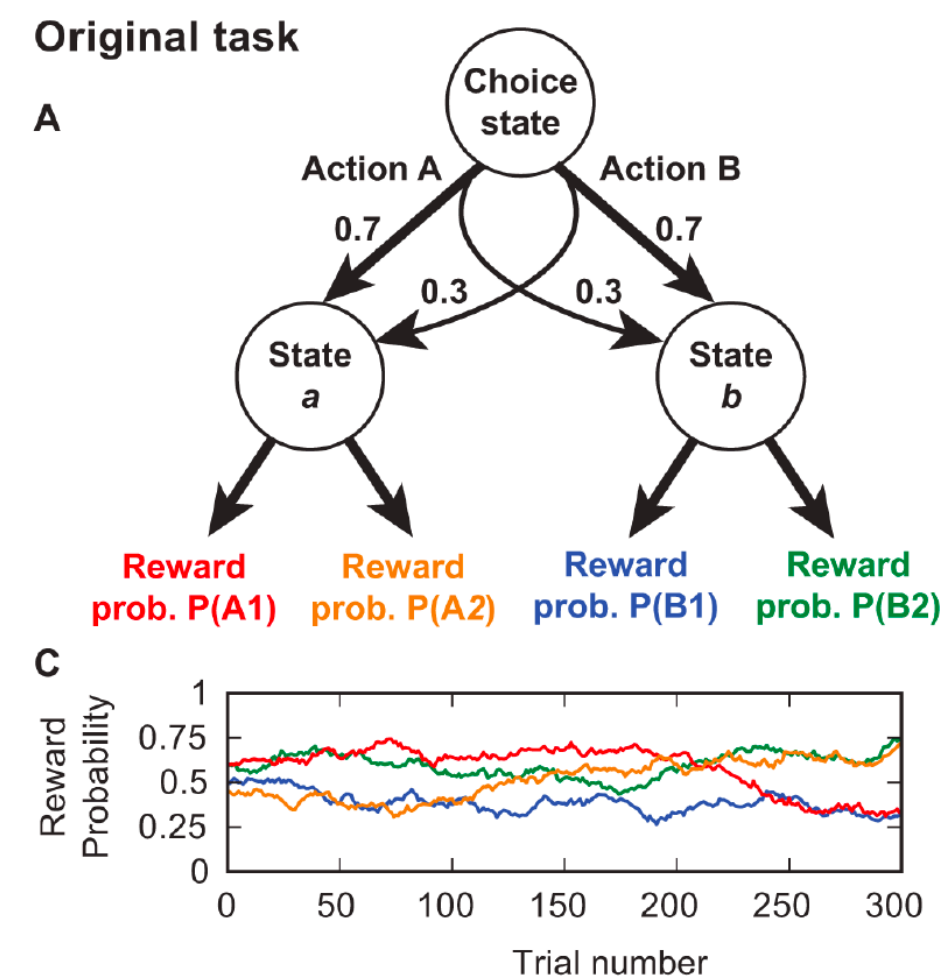


Akam, Costa, Dayan,
PLOS Computational Biology, 2015

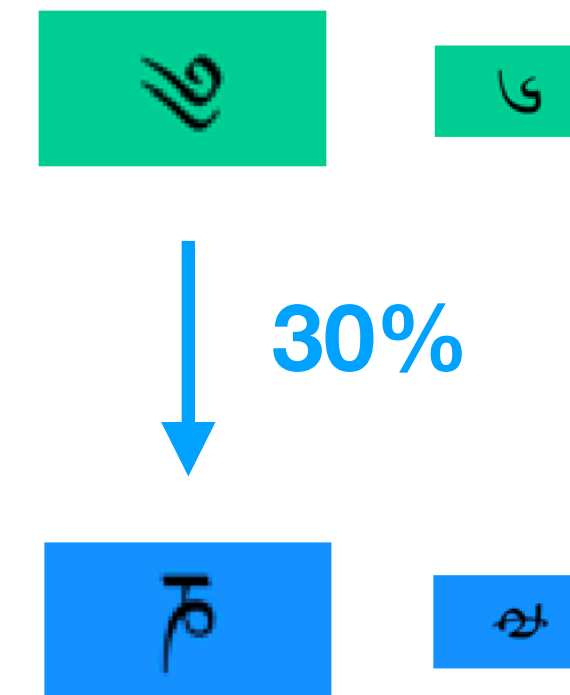


Which green option should the agent choose again
at trial t+1?

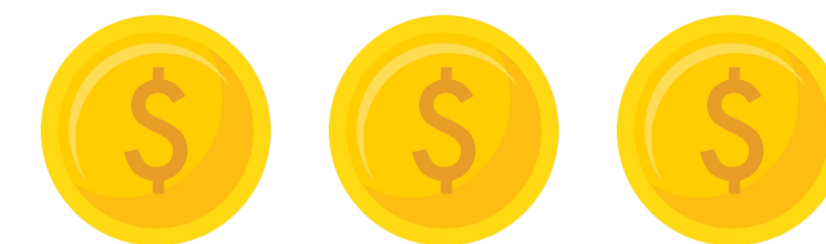
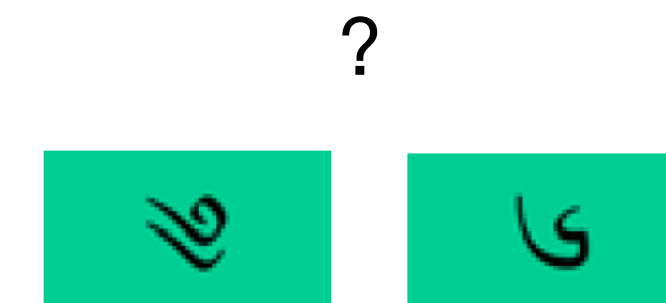
Two-step task: one of the most iconic RL tasks



Trial t



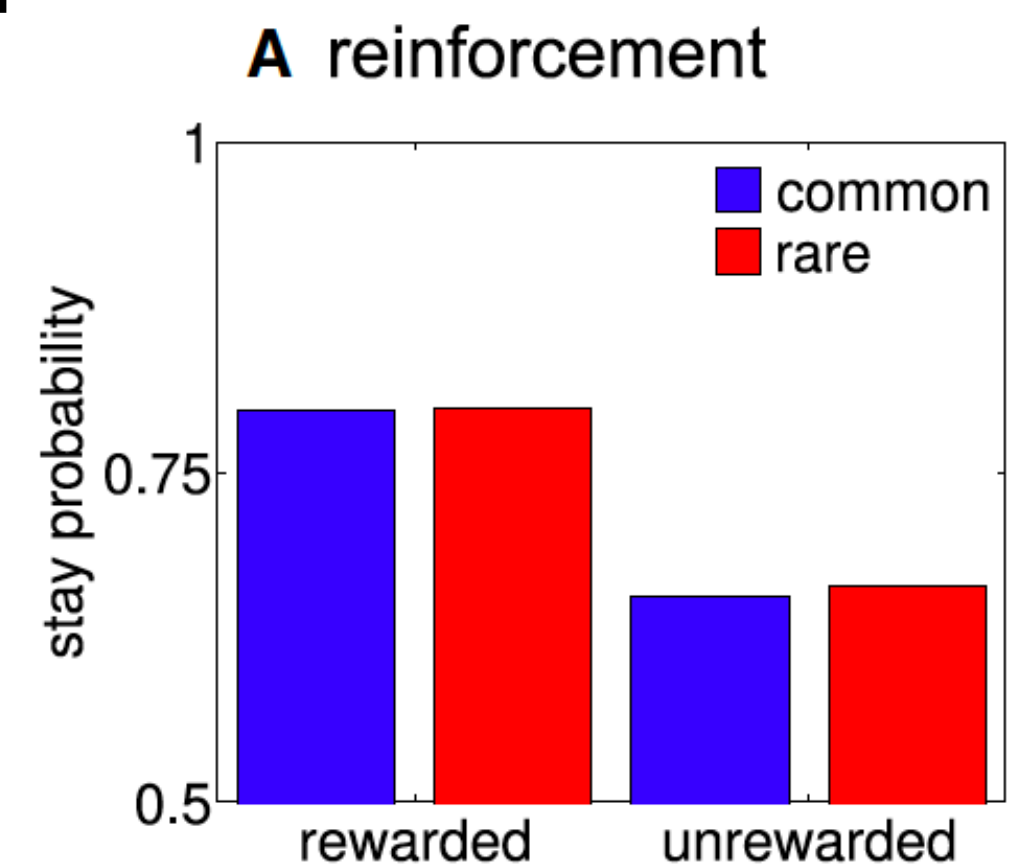
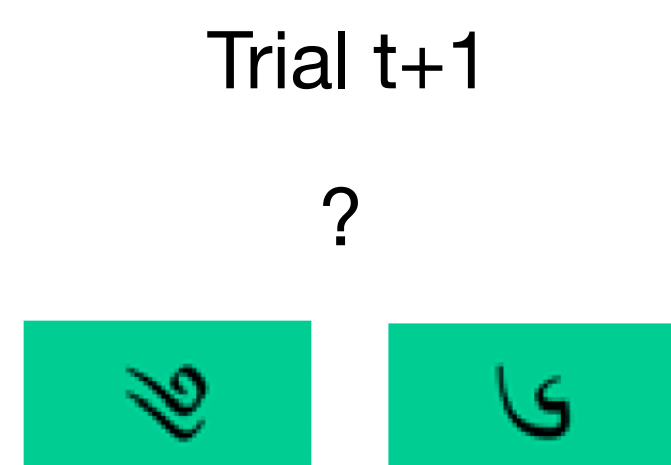
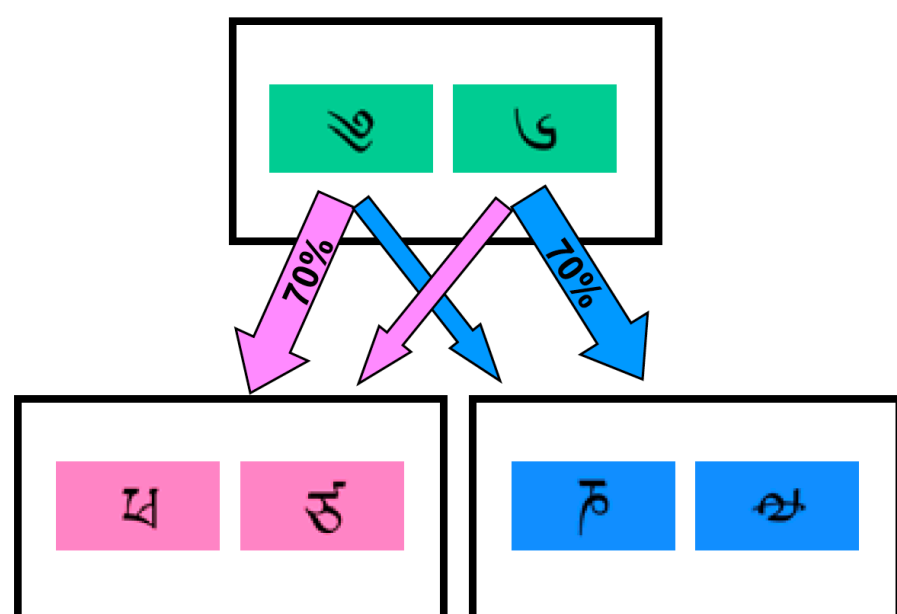
Trial t+1



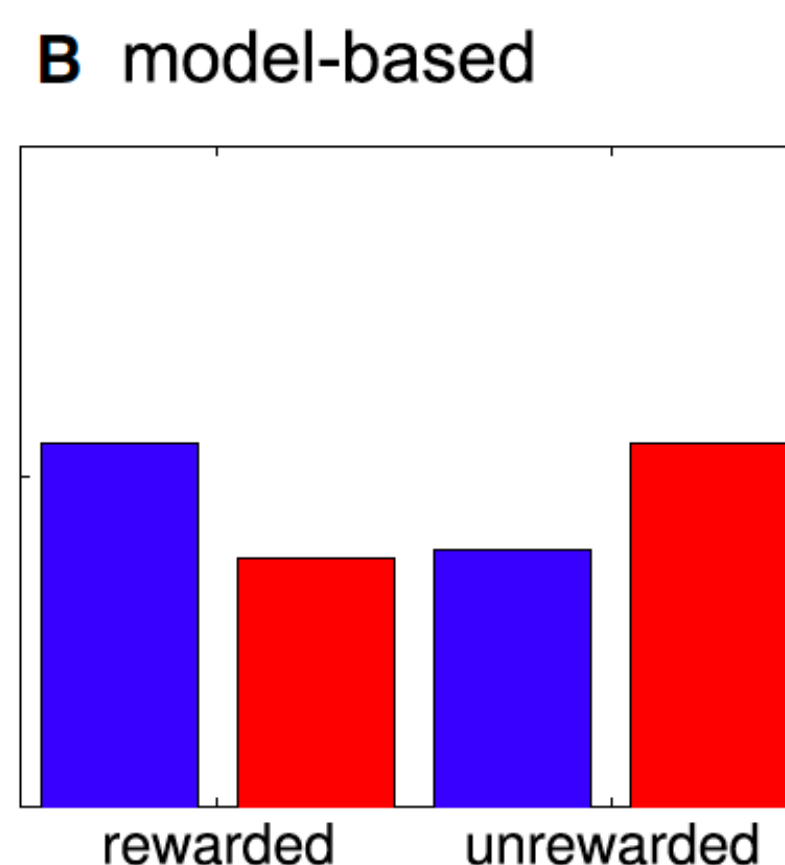
Which green option should the agent choose again at trial t+1?

Akam, Costa, Dayan,
PLOS Computational Biology, 2015

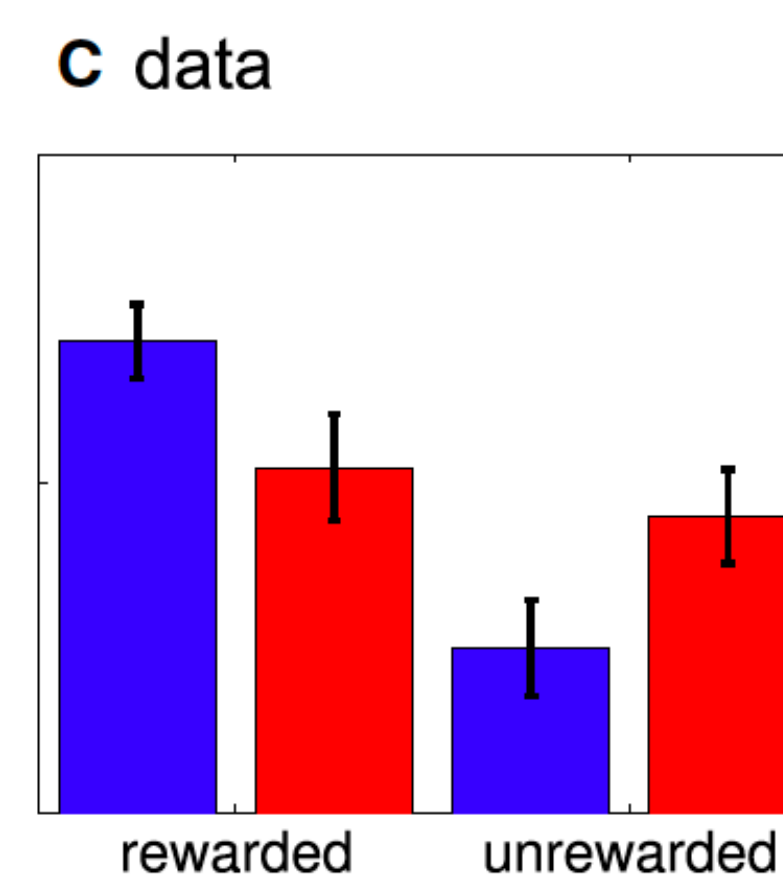
Two-step task: one of the most iconic RL tasks



Model-free RL agent: repeat what is rewarding



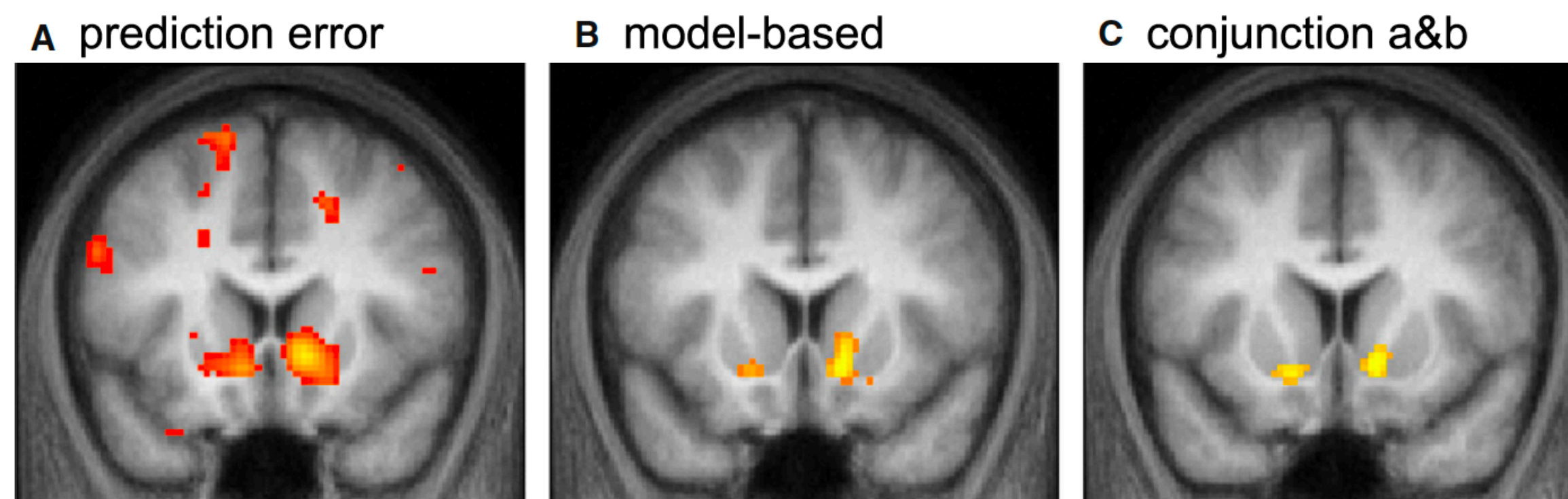
Model-based RL agent: repeat what is rewarding, but be clever



Really data: a mix of both

Two-step task: one of the most iconic RL tasks

Model-free and model-based prediction errors in ventral striatum



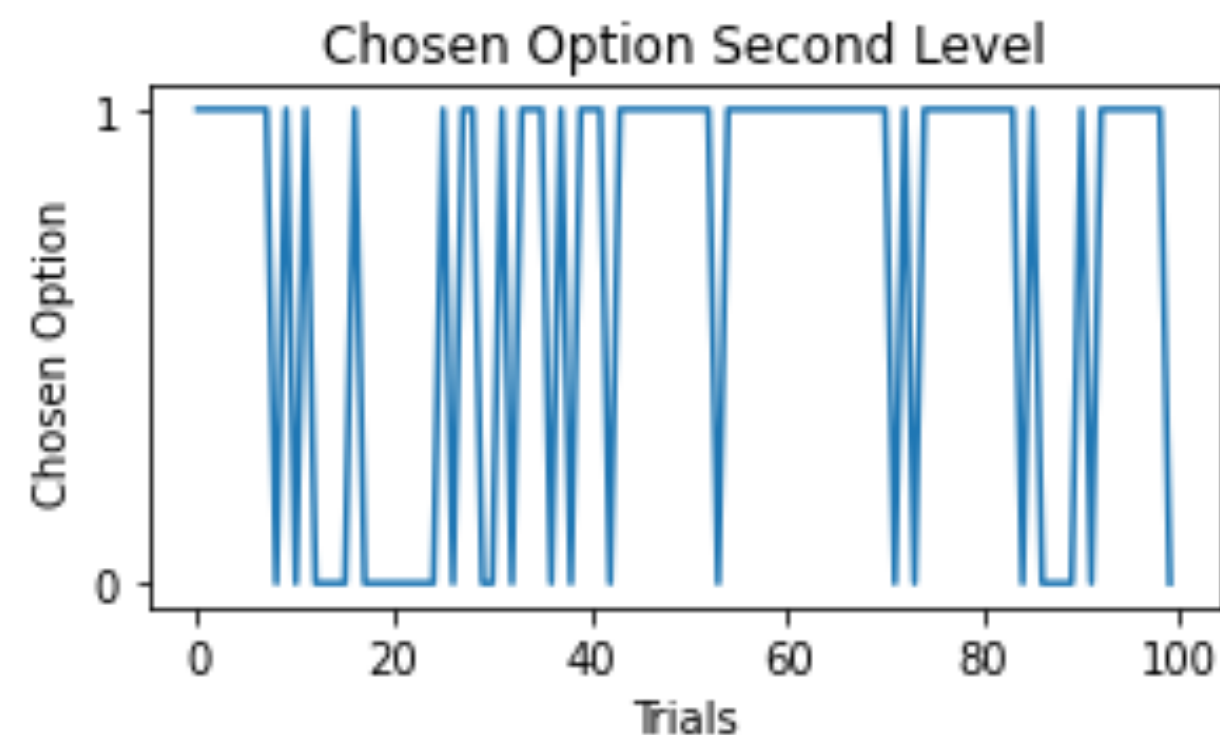
Coding: 2-Step

https://github.com/schwartenbeckph/RL-Course/tree/main/2022_07_19

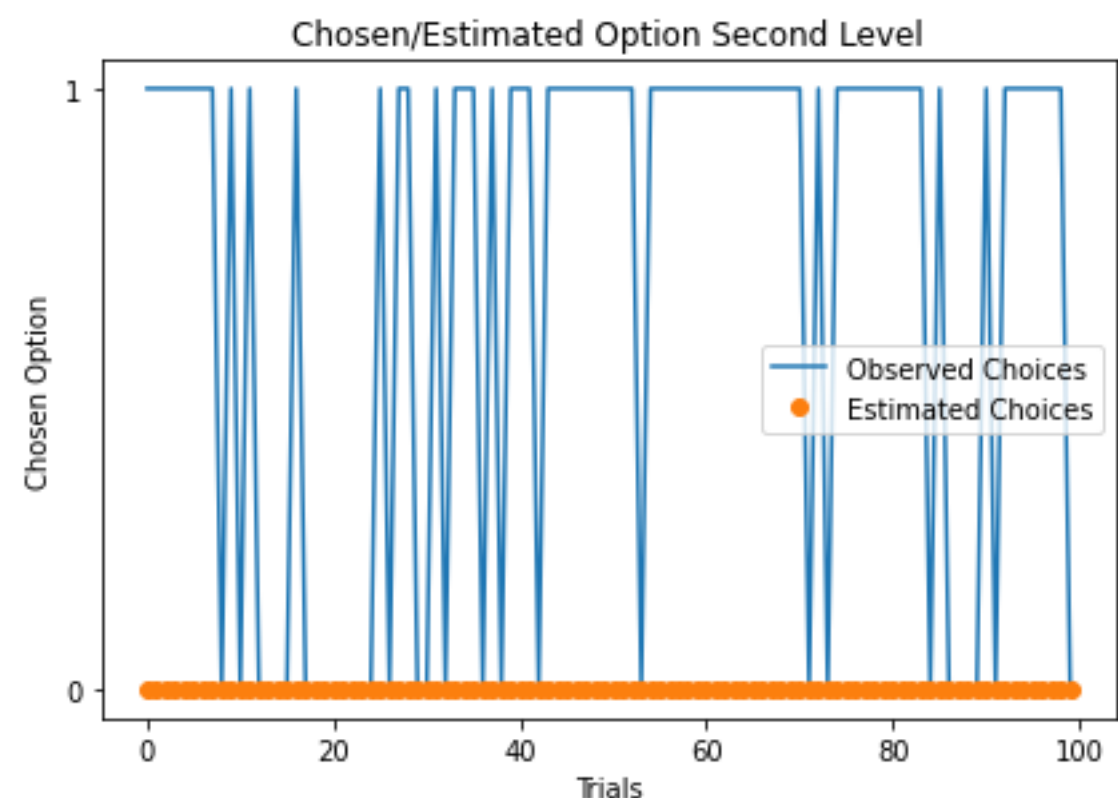
Model-fitting (a crash course)

Problem: how do we find the best parameters for a given model?

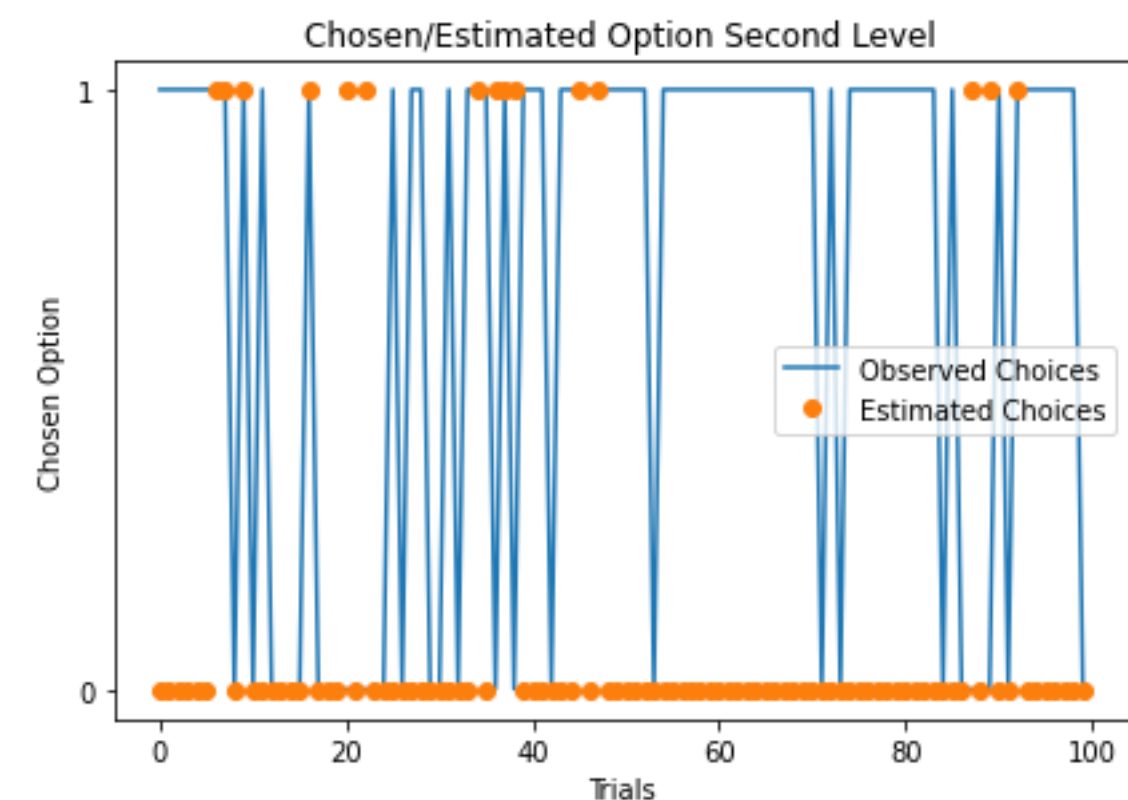
Assume your participant behaves like this:



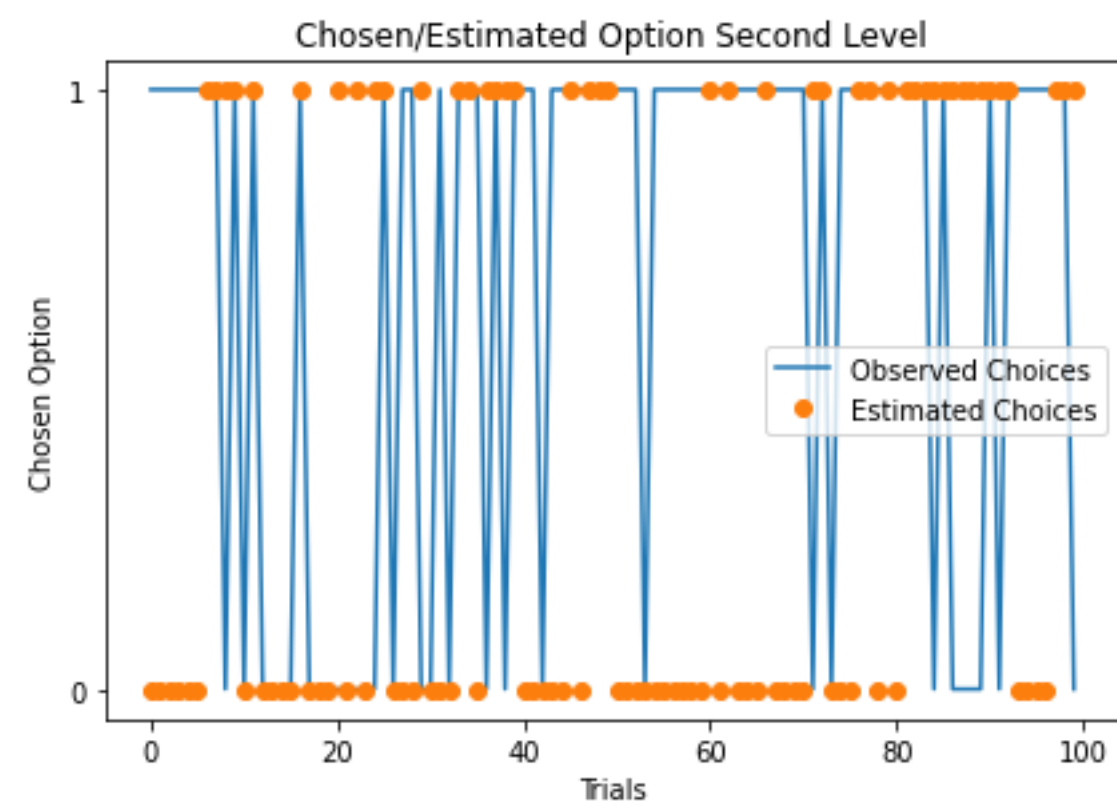
(Here: data generated with $\alpha = 0.5$ and $\beta = 5$)



($\alpha = 0$ and $\beta = 5$)



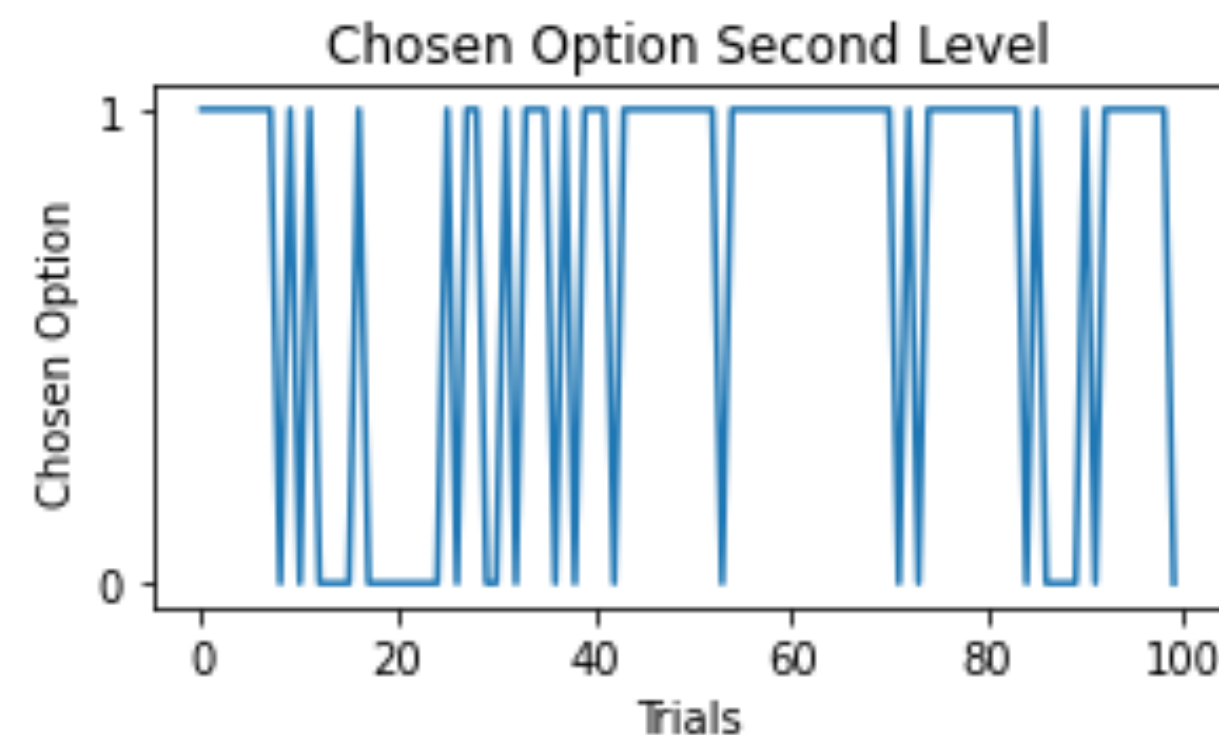
($\alpha = 1$ and $\beta = 5$)



($\alpha = 0.55$ and $\beta = 5.06$)

How will we do this?

Assume your participant behaves like this:



(Here: data generated with $\alpha = 0.5$ and $\beta = 5$)

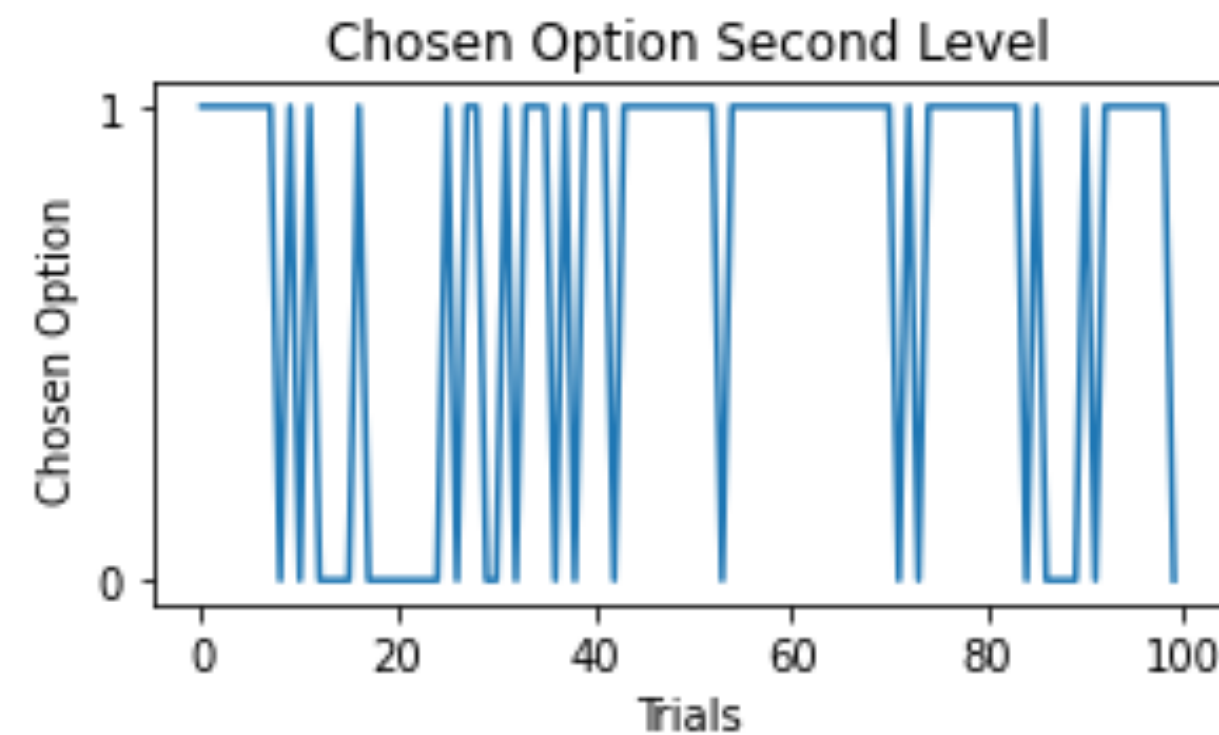
Idea (grossly simplified!):

- i) start with some initial params, e.g. α_1, β_1
- ii) simulate behaviour using α_1, β_1
- iii) assess the difference between real and simulated behaviour
- iv) Move α_1, β_1 around a bit to match real behaviour better
- iv) Do this often, until you can't really improve any more

In essence: find parameters that **maximise the probability of observing** every single choice

How will we do this?

Assume your participant behaves like this:



(Here: data generated with $\alpha = 0.5$ and $\beta = 5$)

Formally (don't take too seriously!):

$$\operatorname{argmax}_{\hat{\alpha}, \hat{\beta}} P(\text{data} \mid \hat{\alpha}, \hat{\beta})$$

Often it is convenient to work in log-space:

$$\operatorname{argmax}_{\hat{\alpha}, \hat{\beta}} \sum_i \log P(\text{data}_i \mid \hat{\alpha}, \hat{\beta})$$

In essence: find parameters that **maximise the probability of observing** every single choice

Coding: 2-Step

https://github.com/schwartenbeckph/RL-Course/tree/main/2022_07_19