Building
Generally Capable AI Agents
with MineDojo

https://developer.nvidia.com/blog/building-generally-capable-ai-agents-with-minedojo/

Admin

- **Draft feedback from TAs going out today**
- Marks later this week …

- **Don’t forget weekly project standups!**

The Data of RL

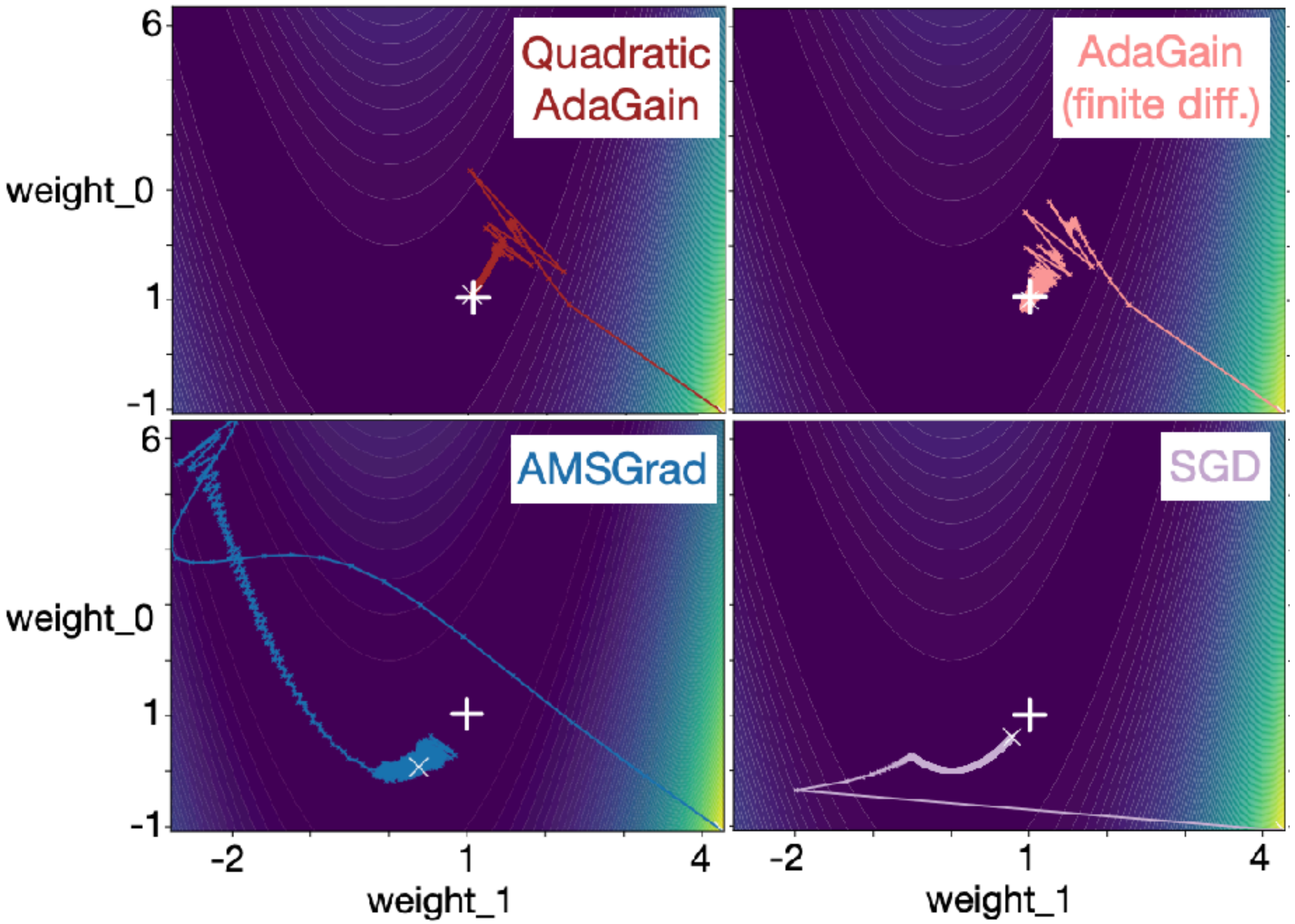
Imagine you developed an new algorithm

- One of the primary ways to understand and evaluate your new idea is via experiments
- There are many things you might want to know:
  - Is my implementation correct?
  - Does the method converge to the correct thing?
  - How does the performance vary as a function of initialization, hyper parameters, and design choices?
  - What are the limitations of the idea?
  - Lastly, if it is better in some measurable, reliable, relevant way?

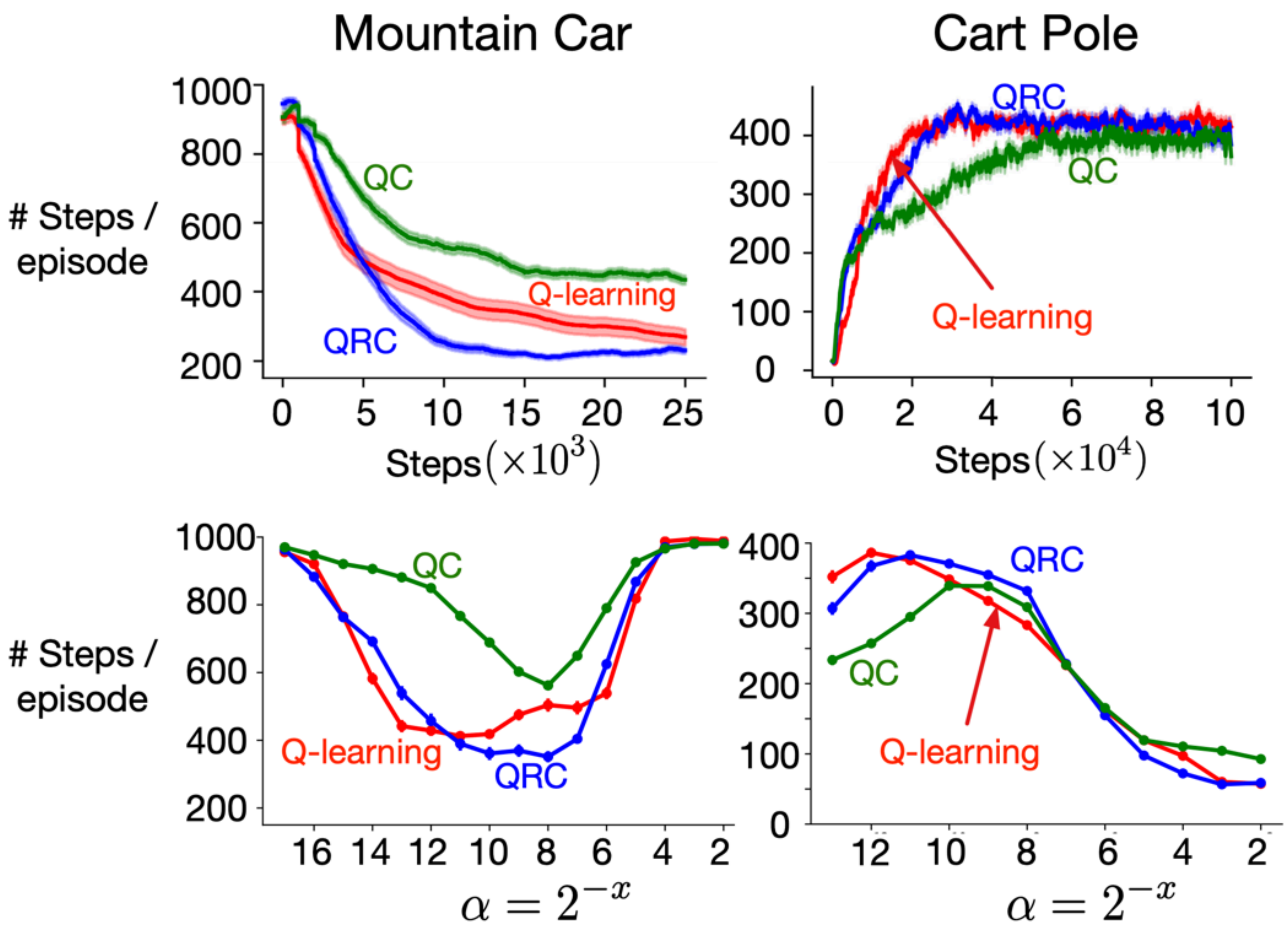
Start with the problem

- Common failure:
  - Spend time developing a new approach, and adjusting your experiments to illustrate the new approach works and works well
  - Someone points out a missing baseline or alternative approach
  - The baseline is better than all the other algorithms tested
- Alternative strategy:
  - Start with the open problem
  - Show that baselines fail or have some important limitations

Example: step-size adaption



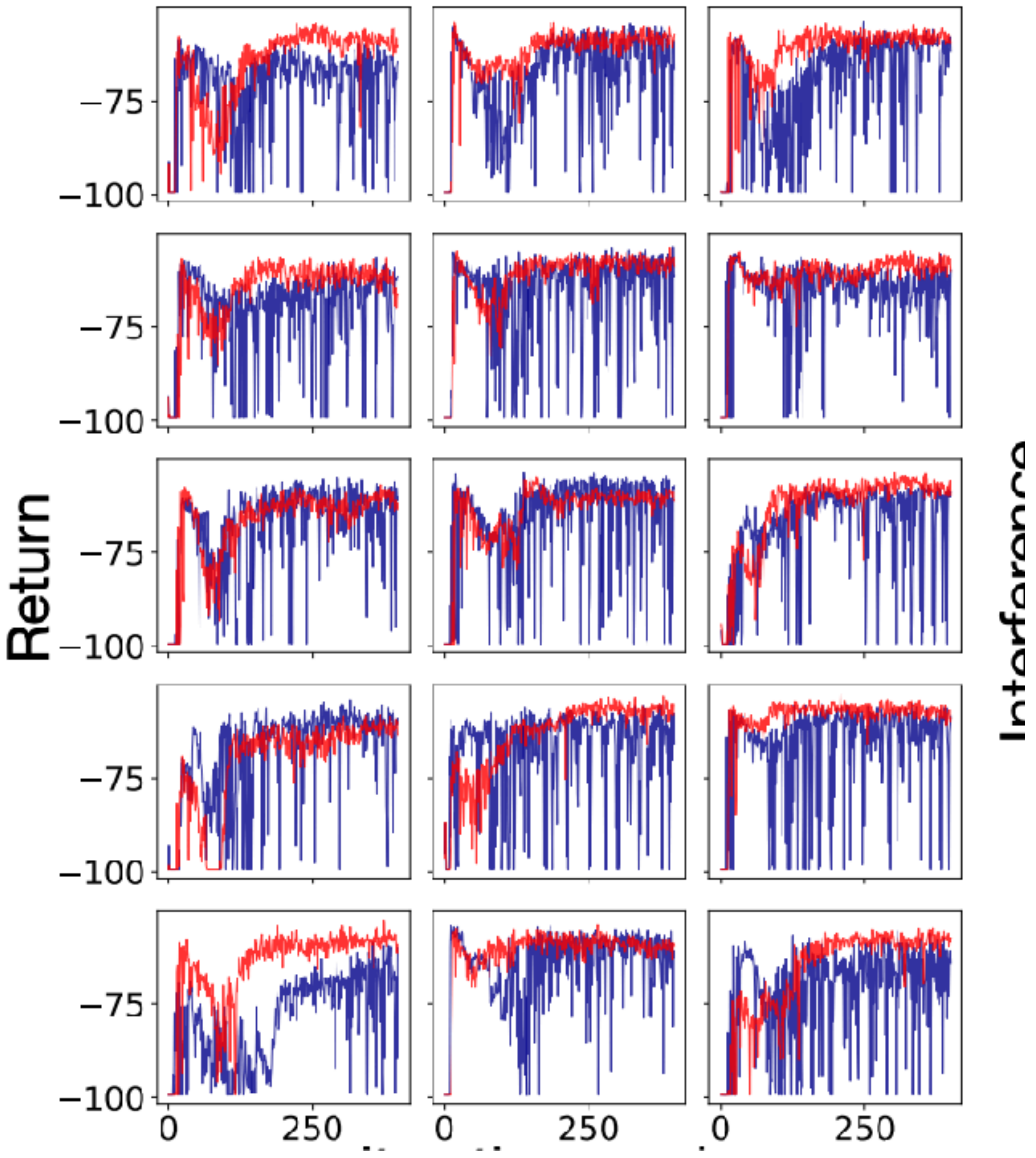
Example: sound off-policy control



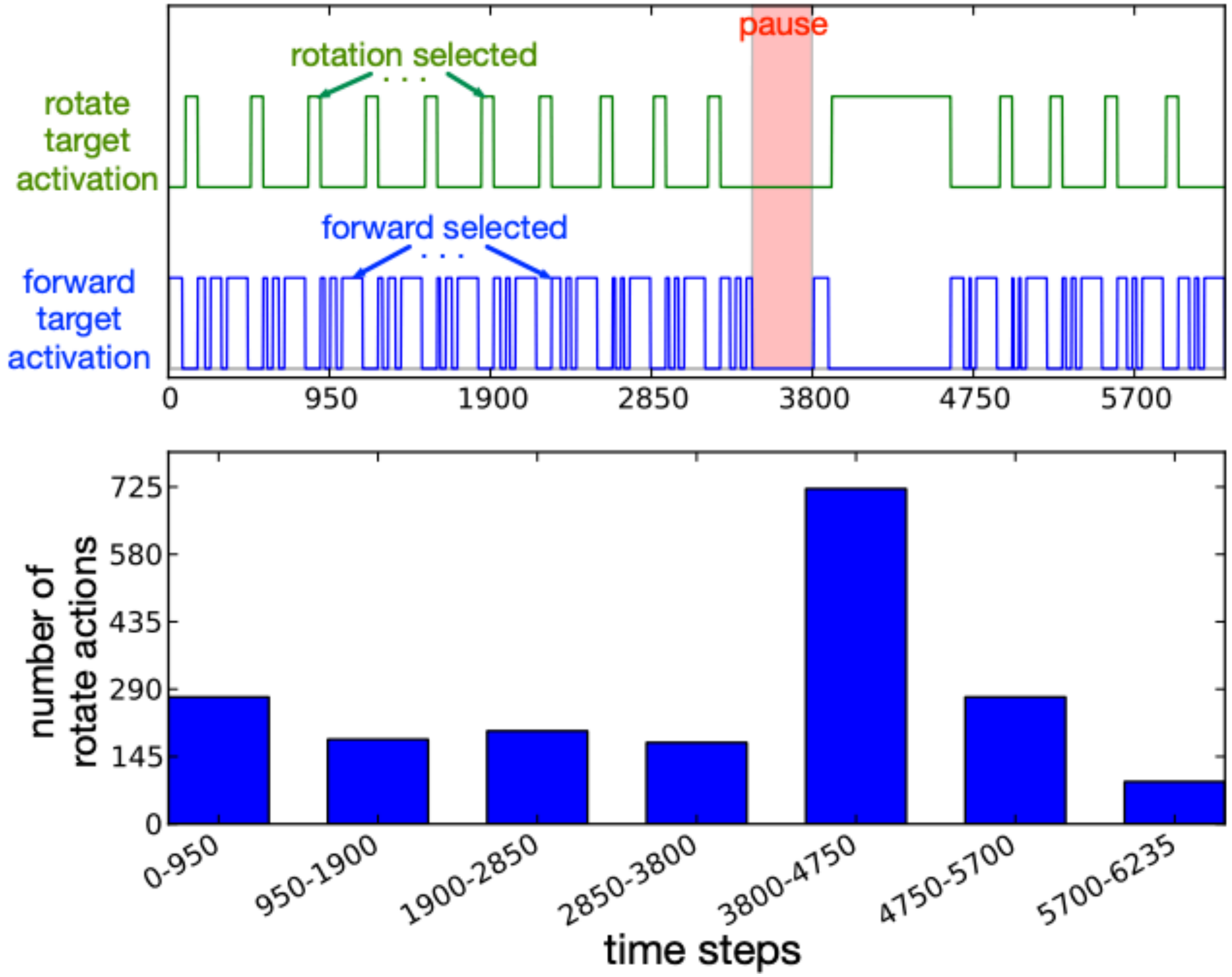
What to measure, what to plot?

- There are always multiple views into an experiment
  - There are many dimensions over which a new idea might be relevant
- This about what aspect is relevant to you and your problem:
  - Final value-function/policy quality/accuracy
  - Speed of learning
  - Insensitivity to hyperparameters
  - Robustness
  - Problem specific metrics
- Just in case: plot everything!

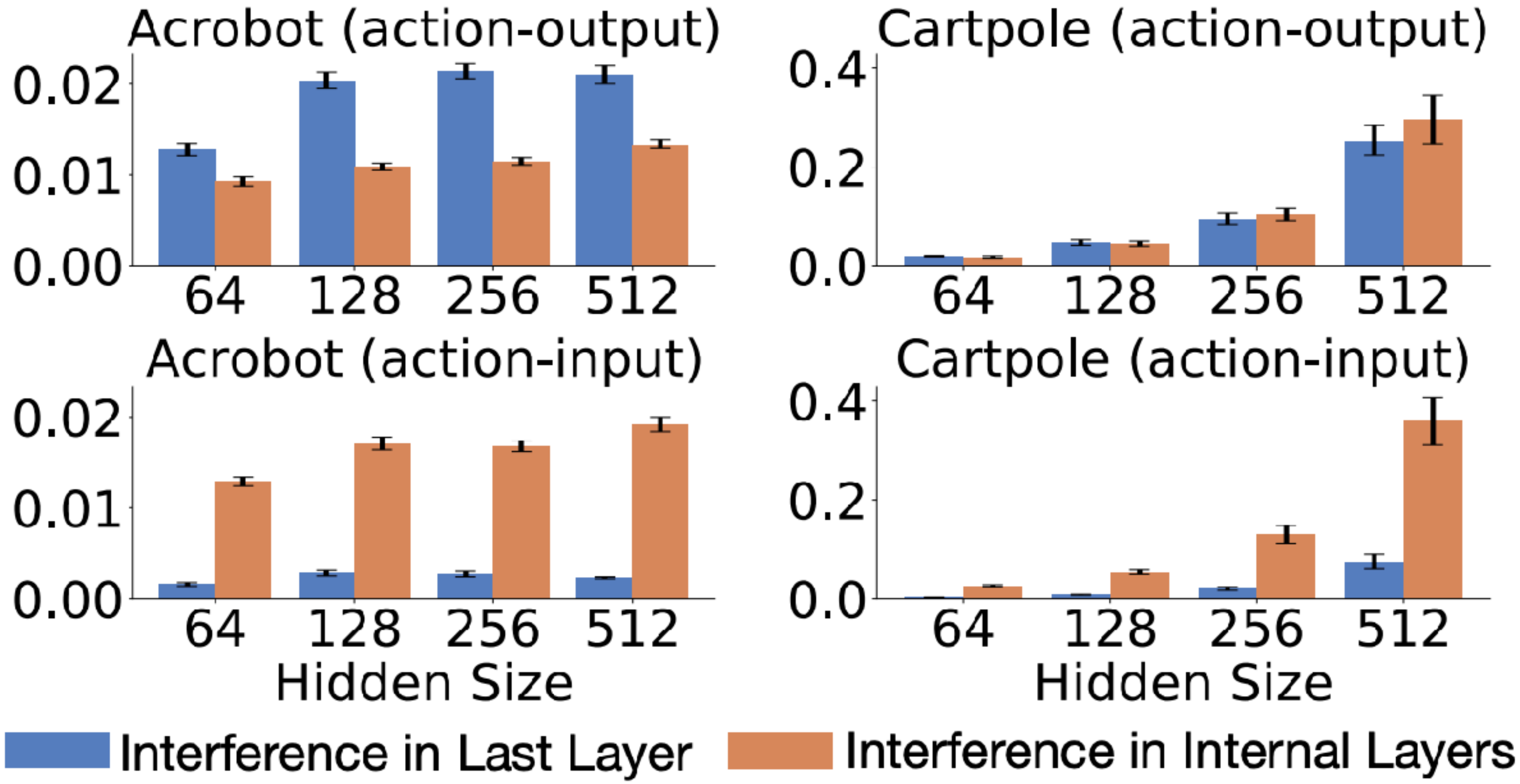
Example: a more stable control algorithm



Example: clear change in behavior



Example: where interference in happening in a network

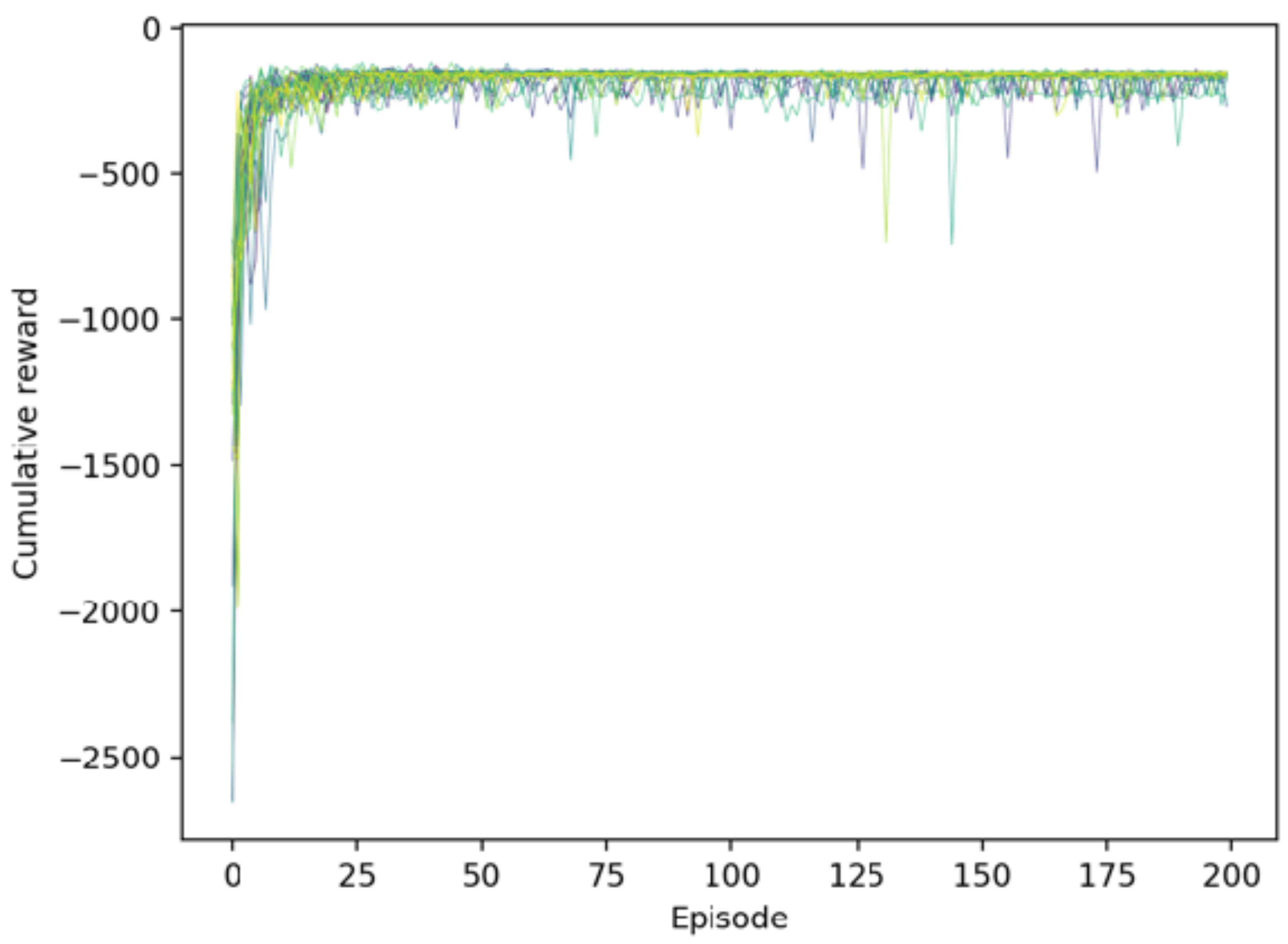


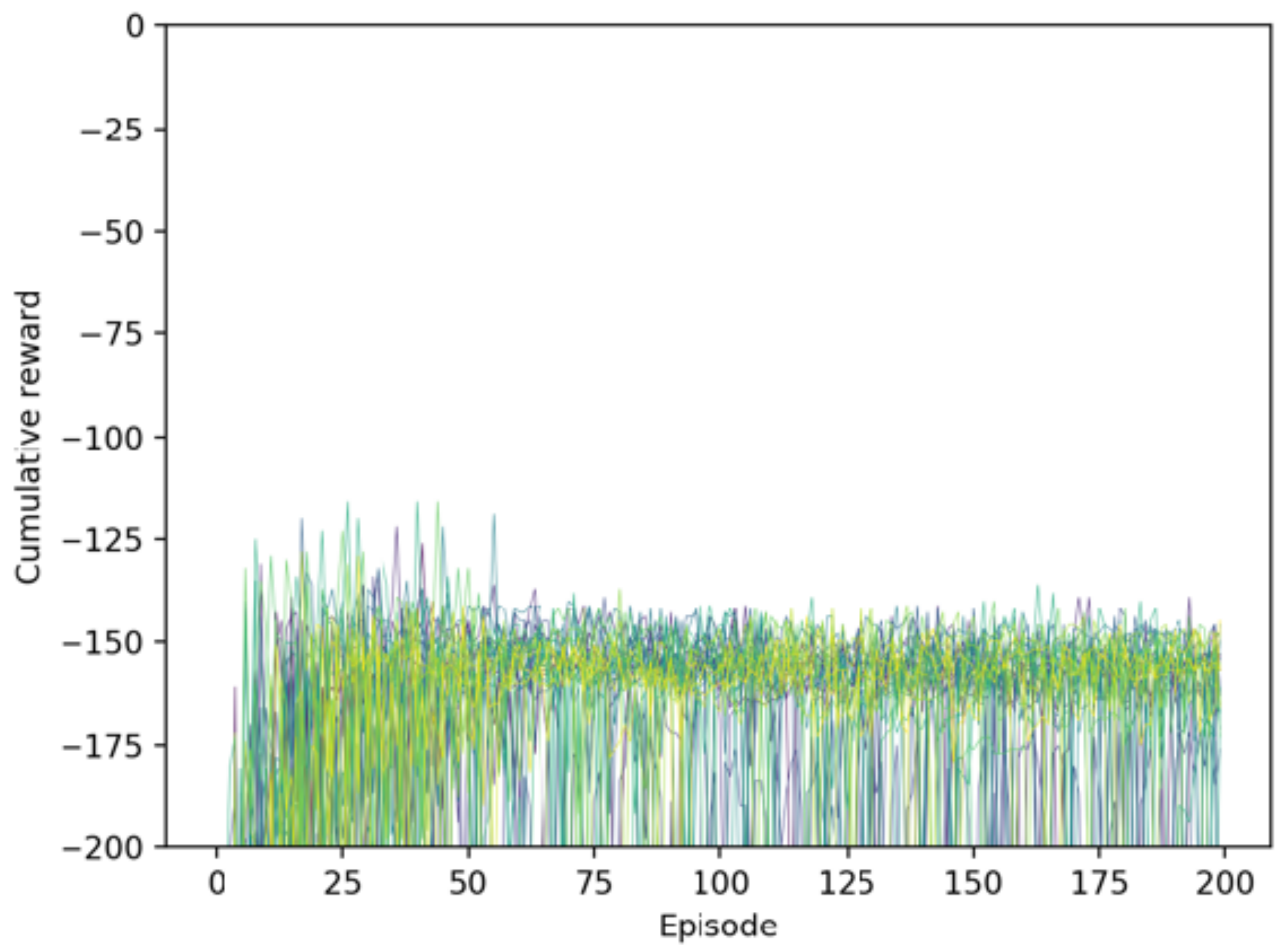
Ultimately we end up comparing things

- SOTA competitor, natural baseline, or calibration agent
- We need to measure something & compare agents
- This is not about winning and losing … its about telling the story of the data
- To tell the story accurately:
    - Properly report uncertainty & variation
    - Properly report how hard it was to get good performance
    - Properly report the impact of as many choices as you can
    - **Stretch**: properly reflect how well these algorithms might work in the real-world

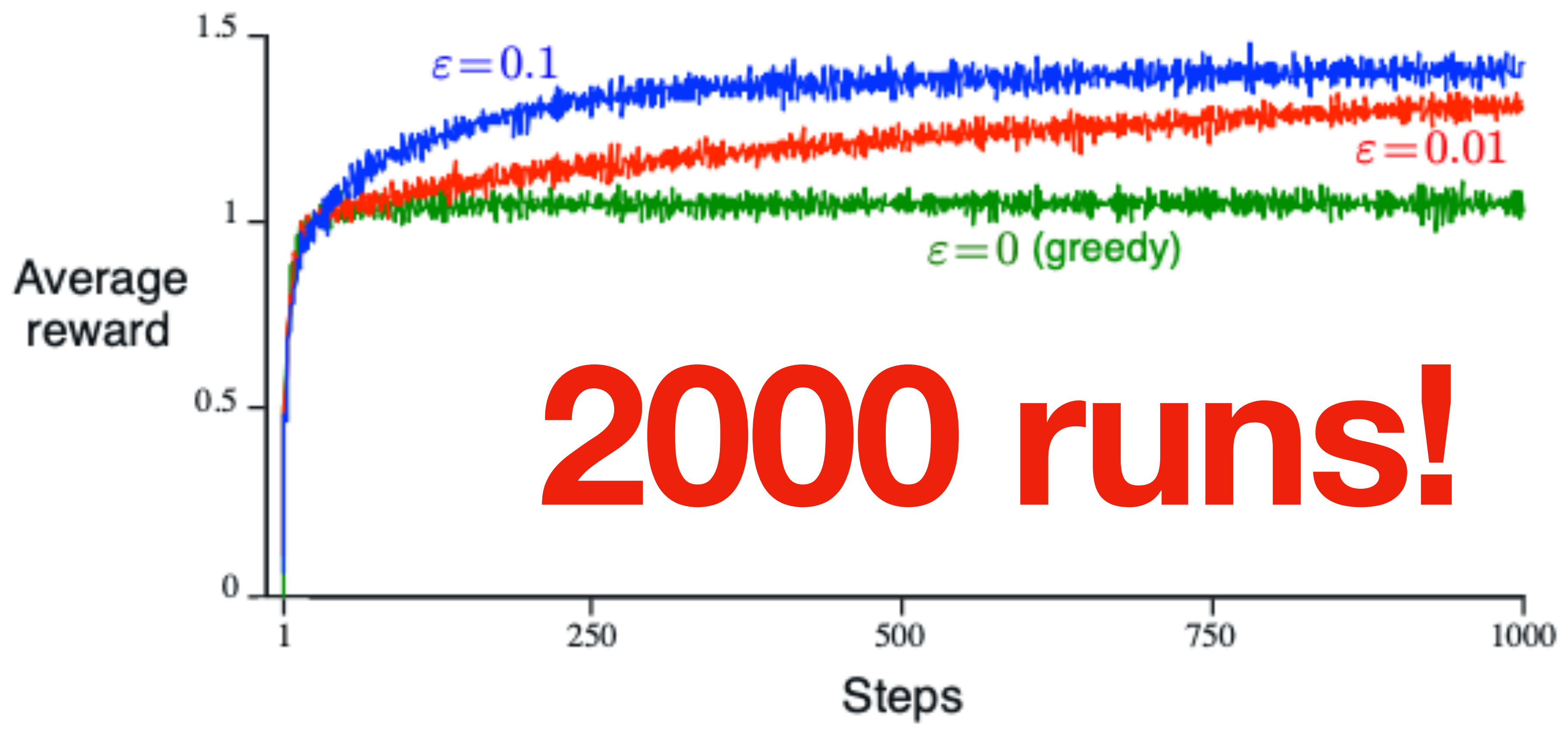
Are our algorithms practically useful?

- **Mountain Car, Sarsa(lambda) with tile coding — pretty much the best you can do on MC**
- Fixed start state, 0.5 decaying step size, 10 tilings 10x10



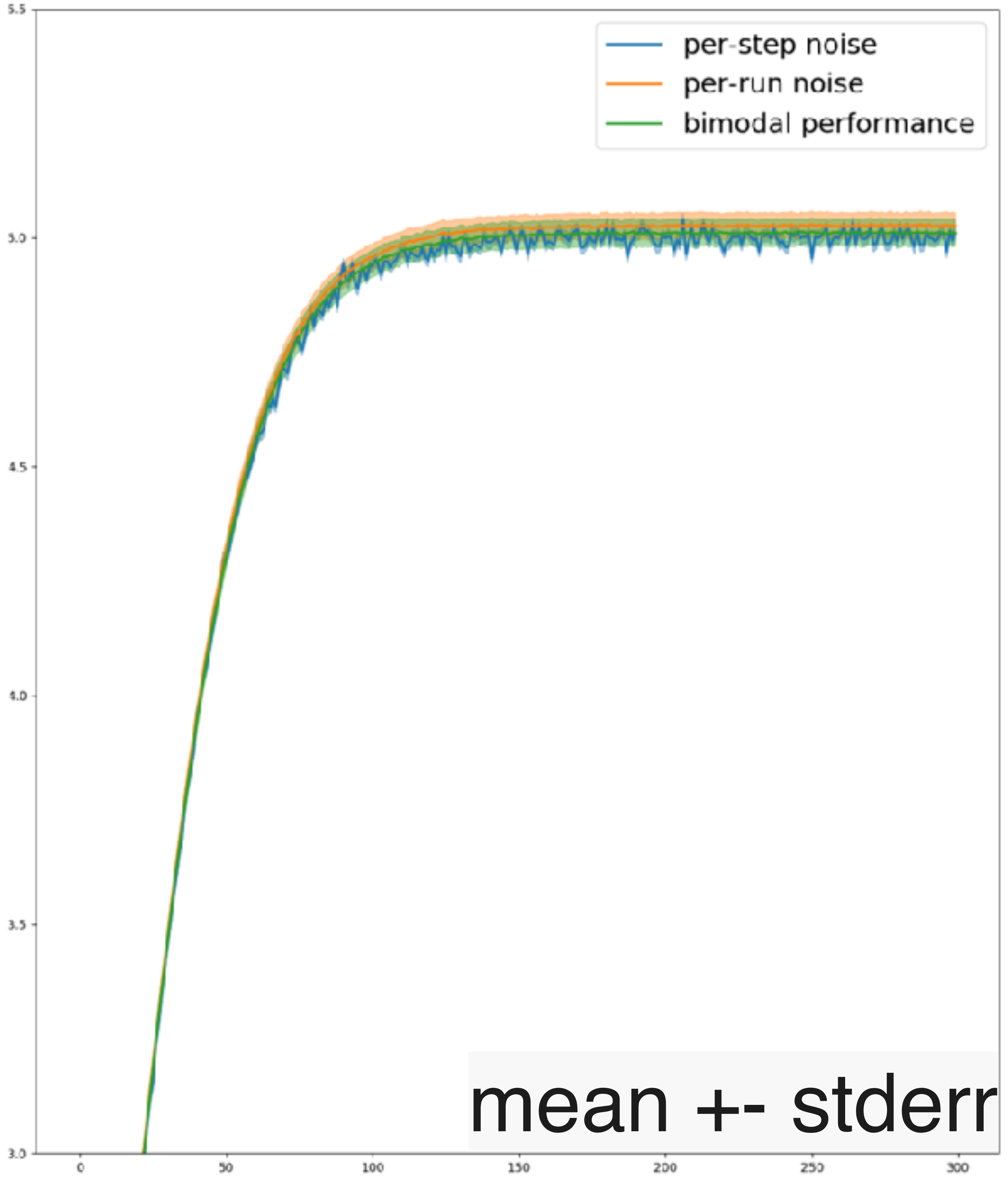


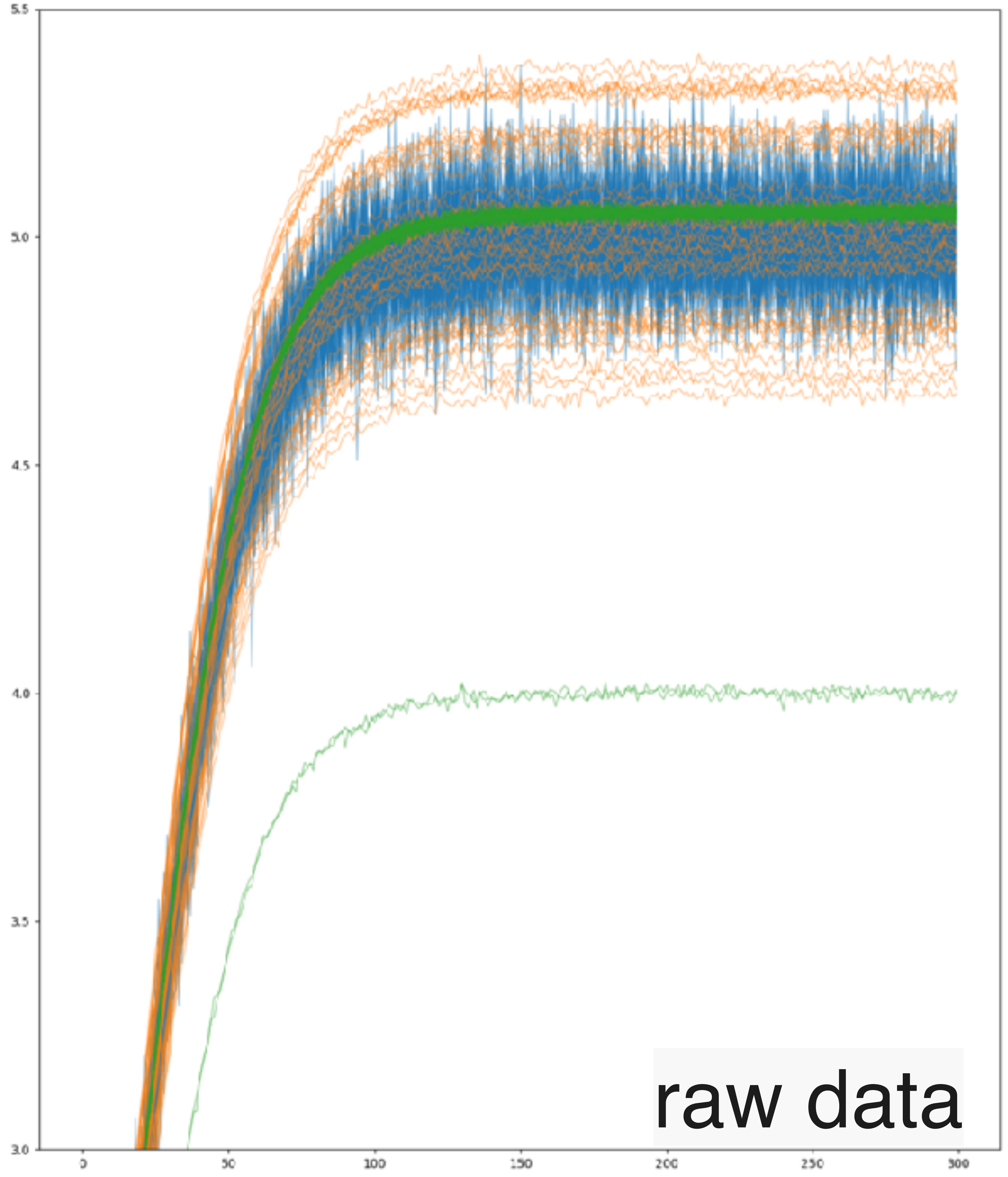
Without repetition we can say so little



- Experiment repetition is so important
- We don't want the results to be skewed by one algorithm getting lucky
  - Remember the MAB in Sutton&Barto…on some runs greedy is optimal
- We want to use statistical tools to talk about aggregate performance
- Hopefully we can build more reliable algorithms
- But we often need to look deeper to understand the mean & variance

The raw data can tell different stories



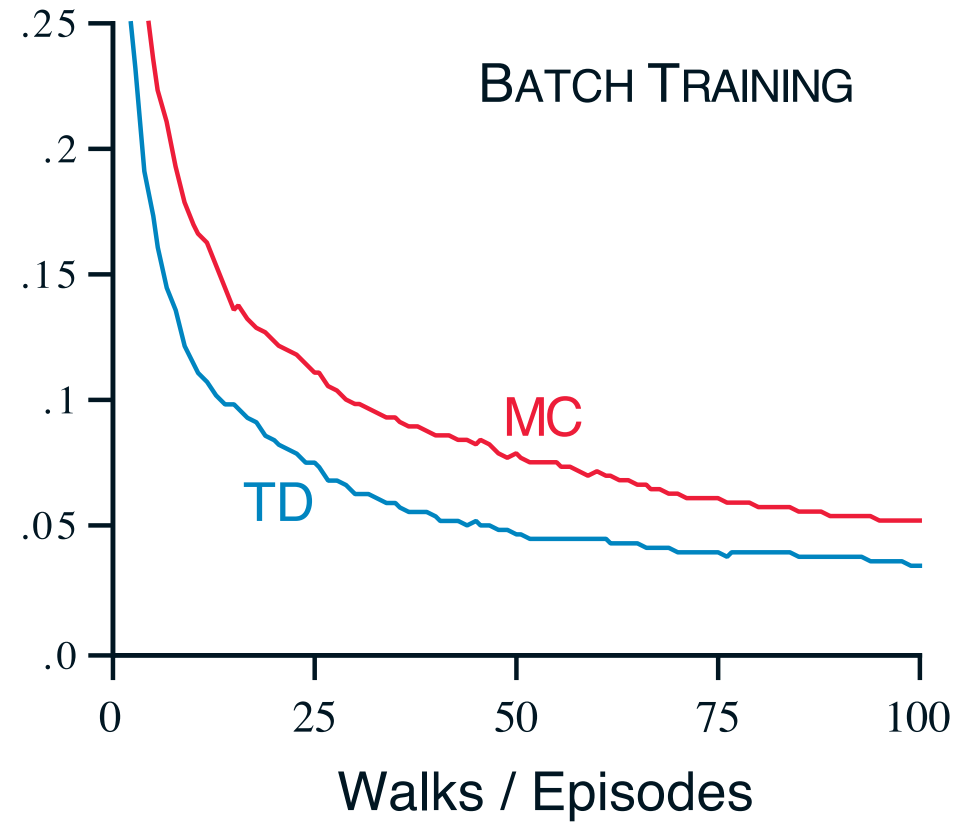


- 50 runs, 300 steps
- Credit: Andy Patterson

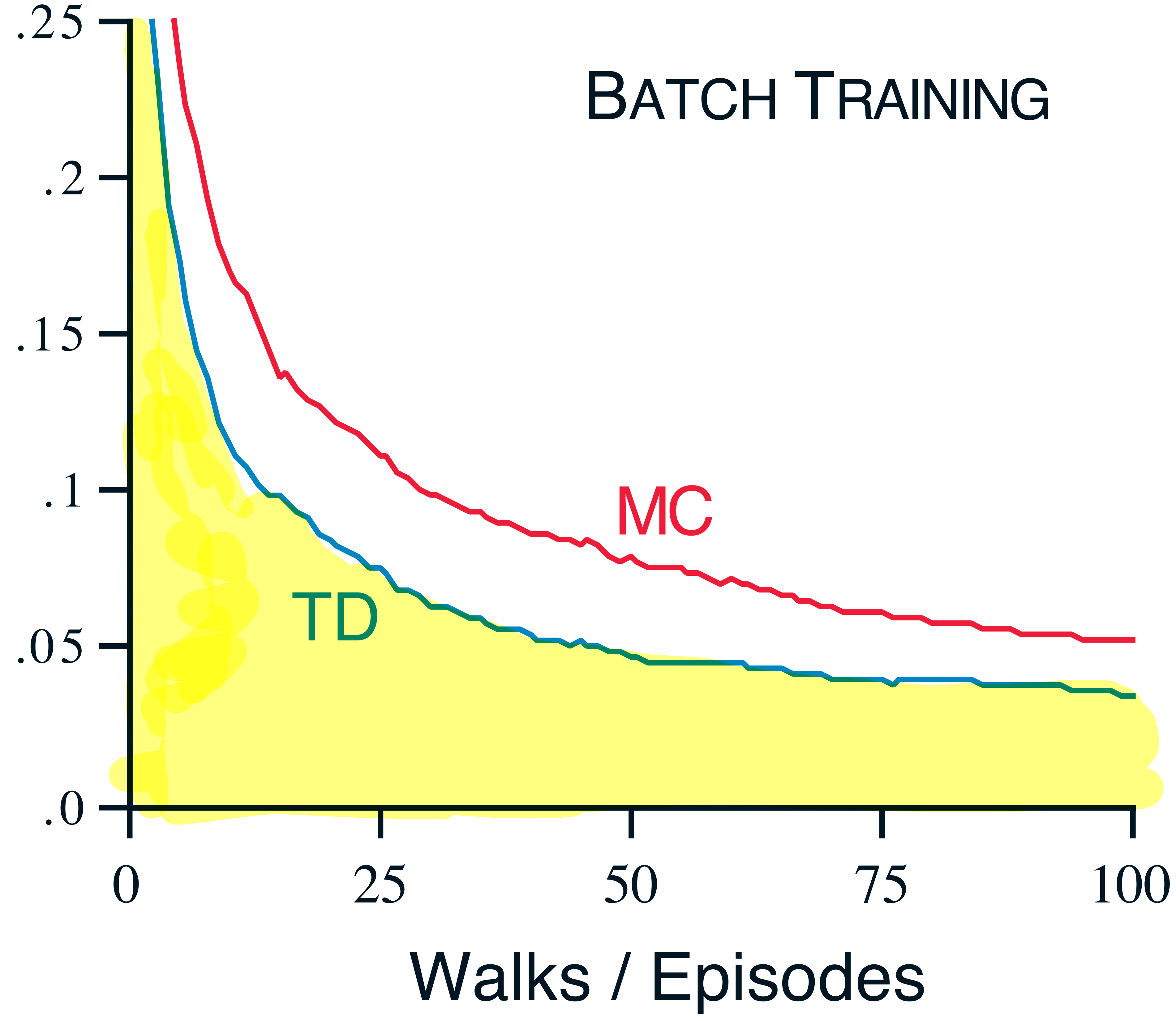
Which data/alg would you prefer?

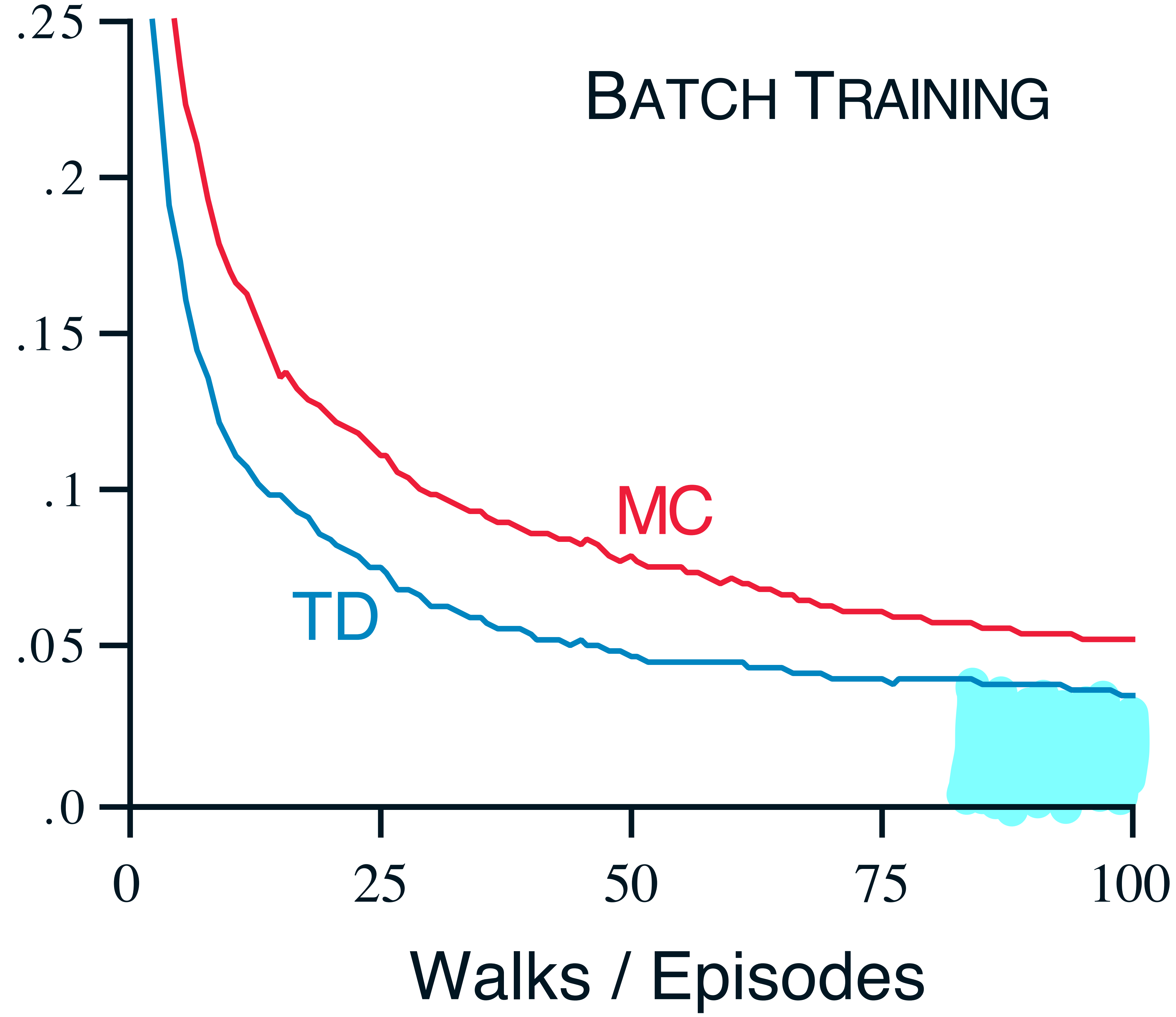
Agents & Environments are data generators

- If we want to make statistical statements about the data, then we have to understand what it looks like
- **We want to turn a learning curve for a single run into a number**
- The first step is deciding on a measure of performance:
  - Total area under the learning curve (AUC)
  - AUC of the large x% of the data
- Other measures focused on stability are also possible but we will start with the classic ones



Getting one number





These are importantly different when sweeping hyper-parameters

The distribution of performance

- Given a set of AUC, one for each run, what does the distribution of those numbers look like?
  - Bell shaped / Normal /Gaussian
  - Skewed
  - Multi-modal
  - Flat or point mass?
- **Practical tip:** set the seed for the environment and the agent independently, and use the run number for reproducibility
- What should we do about the hyper parameters?

The distribution of performance

- Given a set of AUC, one for each run, what does the distribution of those numbers look like?
    - Bell shaped / Normal /Gaussian
    - Skewed
    - Multi-modal
    - Flat or point mass?
- **Practical tip:** set the seed for the environment and the agent independently, and use the run number for reproducibility
- What should we do about the hyper parameters?

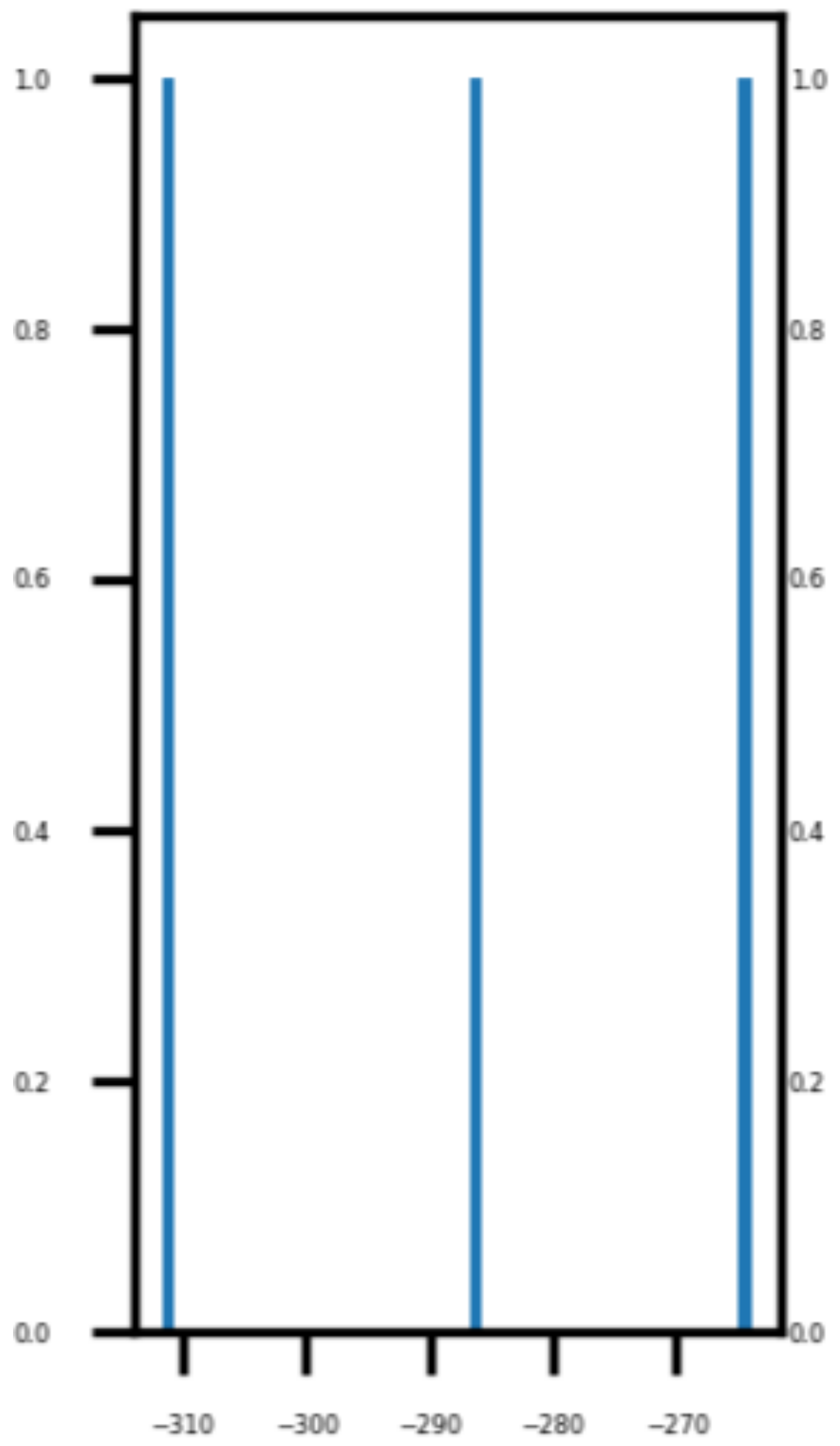
How many runs do we need?

- Common practice is 3, 5, … maybe 8
- In the literature you can find up to thousands of runs

- Let's run an experiment:
  - Mountain Car with random starts
  - Sarsa(lambda) with tile coding — reasonable hyper parameter choices
  - We will plot mean episodic return over 250 episodes
- What story does the data tell?

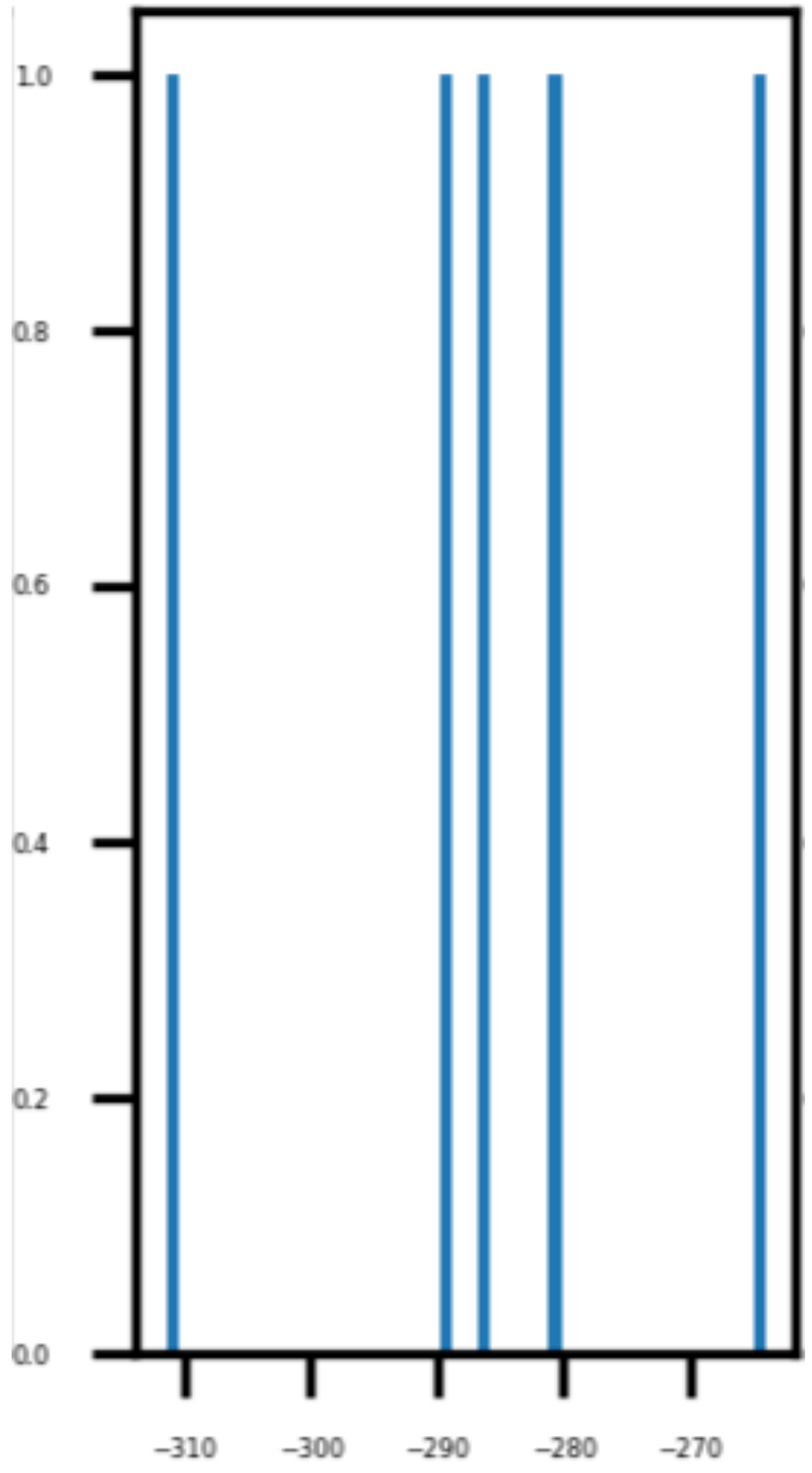
What if we did 3 runs?

- Histogram of mean episodic return over 100k steps (around 250 episodes)



What if we did 5 runs?

- Histogram of mean episodic return over 100k steps (around 250 episodes)

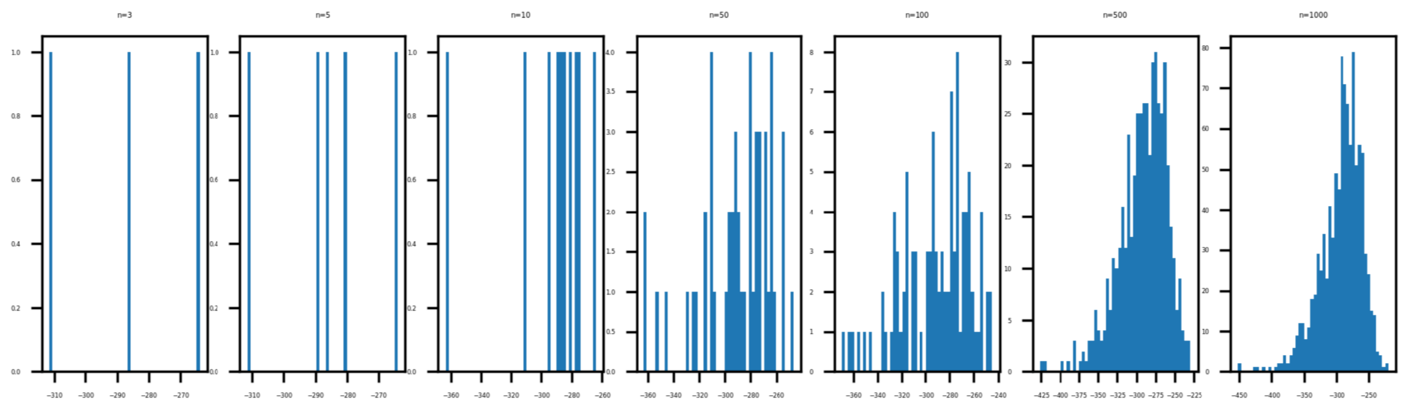


Many runs are needed to see the shape of the distribution

- Histogram of mean episodic return over 100k steps (around 250 episodes)

- Estimating the agent's performance accurately requires many independent repetitions of the experiment

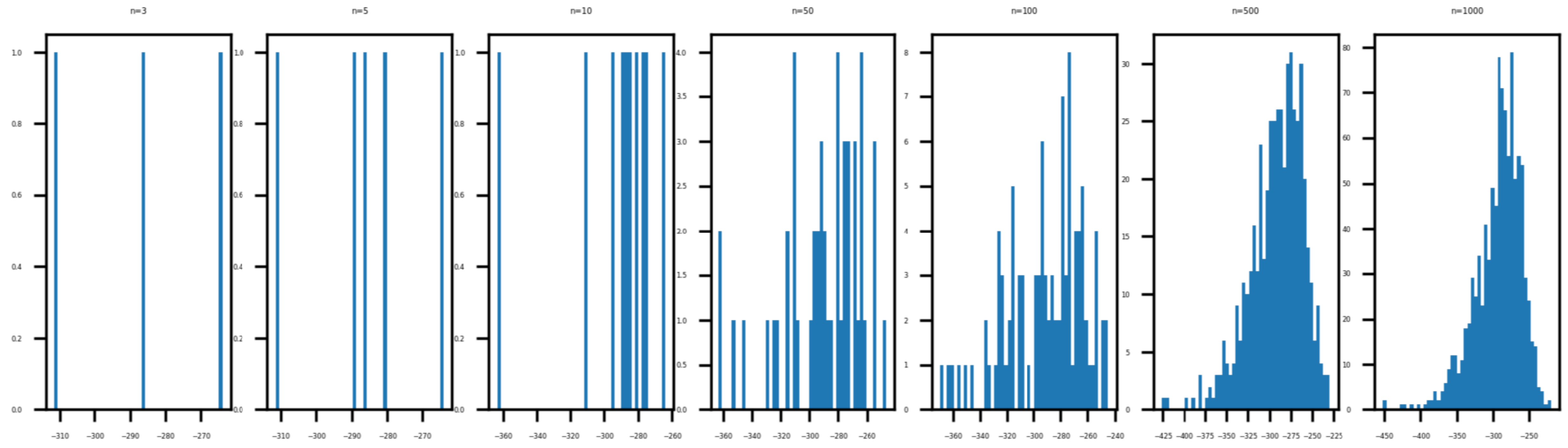
Environments design choices matter too



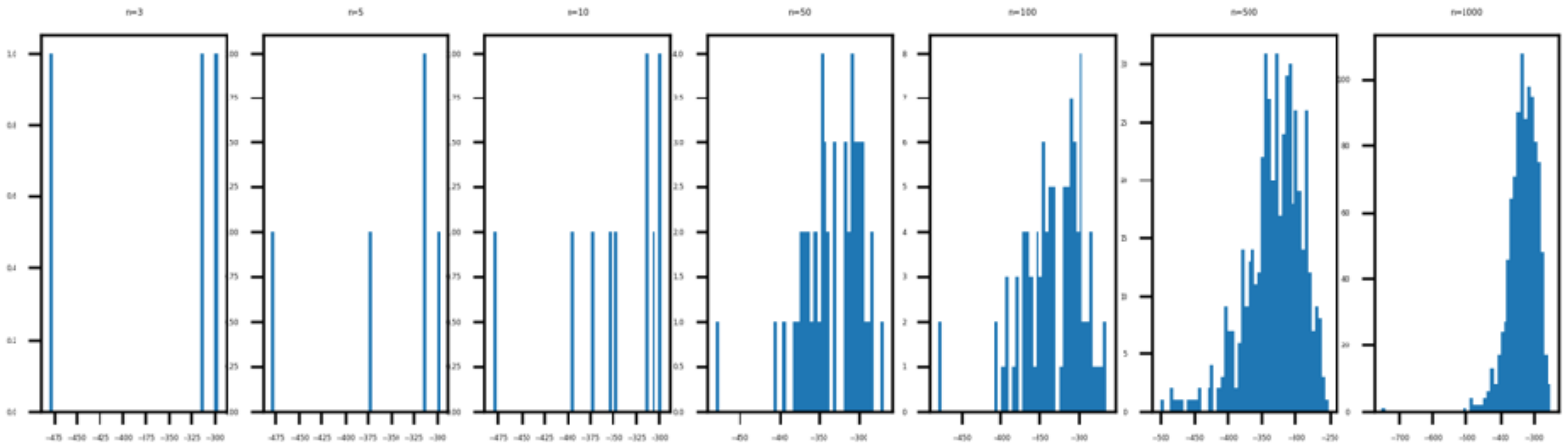
- Notice how the distribution was a bit skewed, not perfectly bell shaped
- We can get other distribution shapes by including **cutoffs**:
  - Restarting the episode if the agent reaches a max number of steps
  - This ensures the no episodes a really bad—**might make bad agents look good**
  - This gives free exploration—especially if random starting states are used

Cut-offs skew performance

Regular MC->



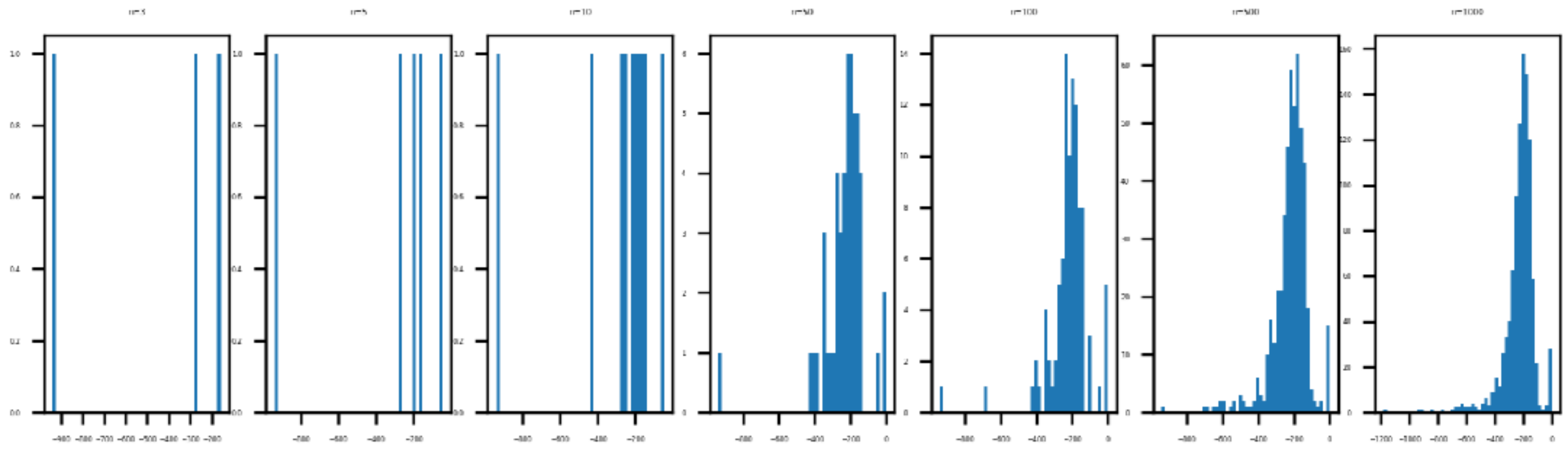
MC w cut-offs->



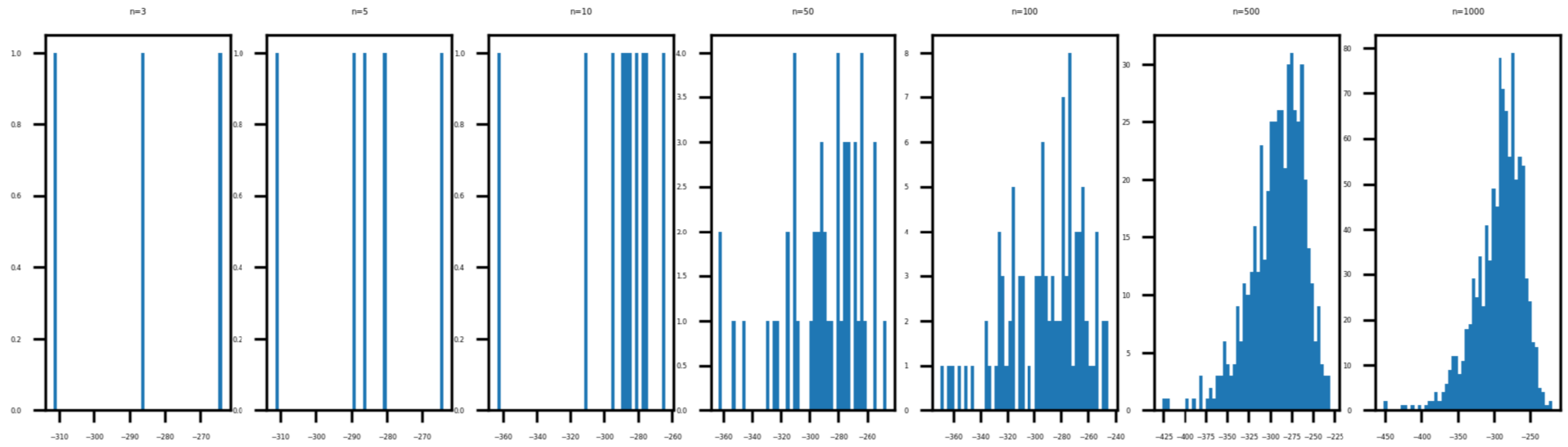
- 1000 step episode max

Every agent & environment pair can be different

- Same experiment and setup in Puddle world:

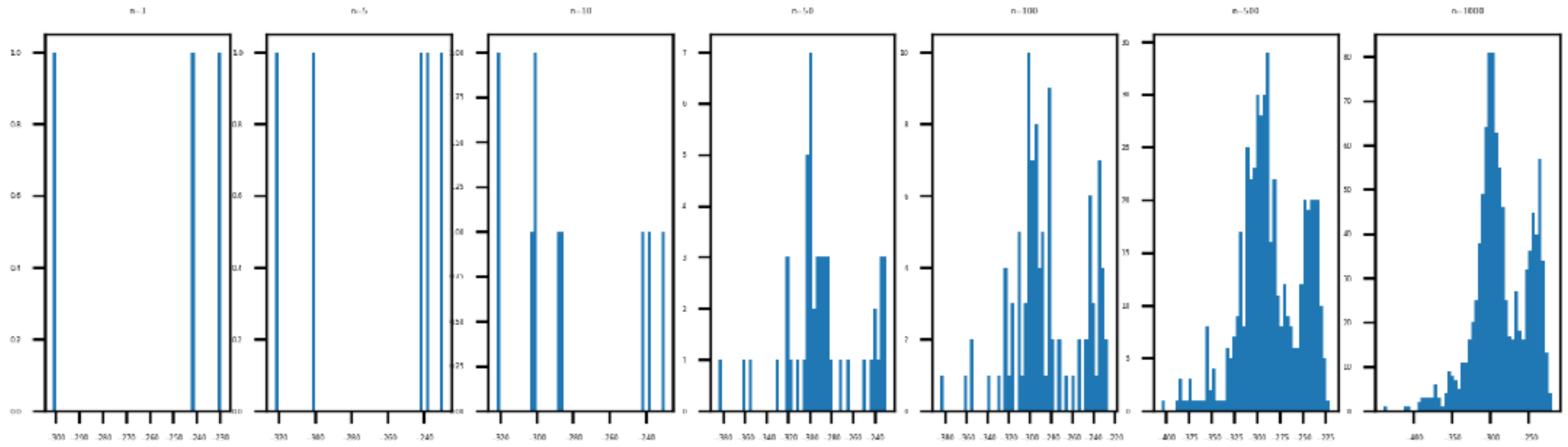


- Mountain car:

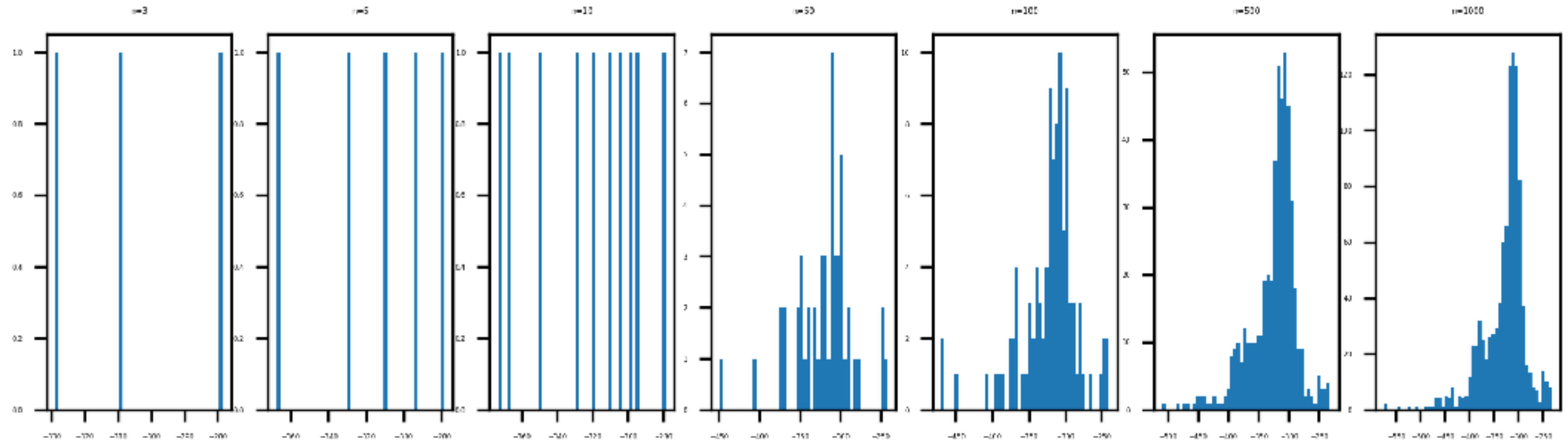


Design choices interact

- Mountain car with two different start states:

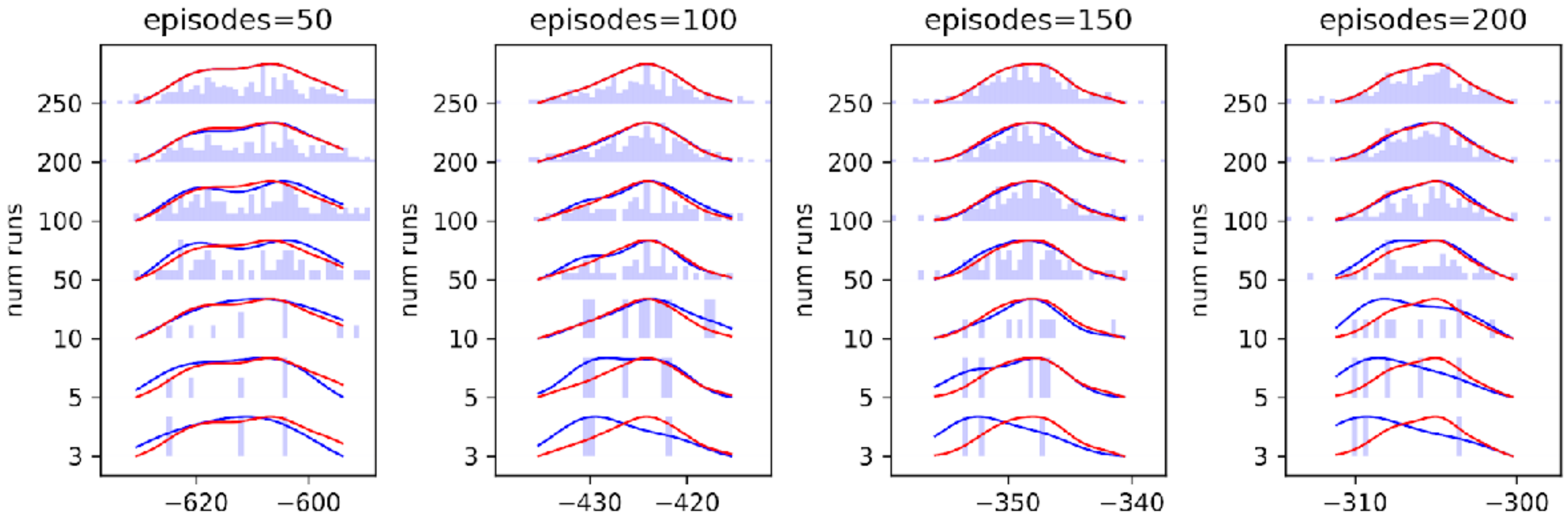


- Mountain car with two start states and cutoffs:

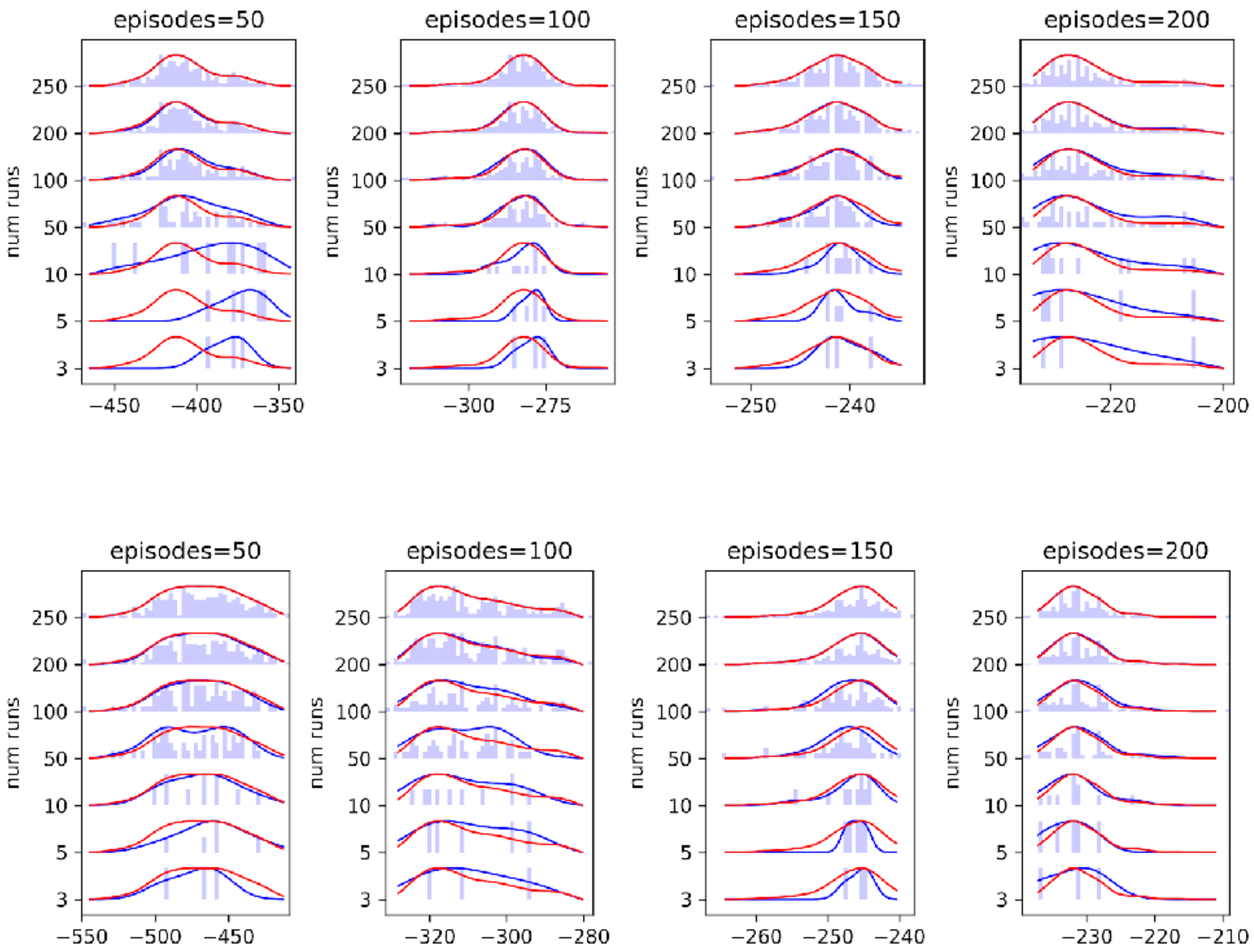


Experiment design choices interact too

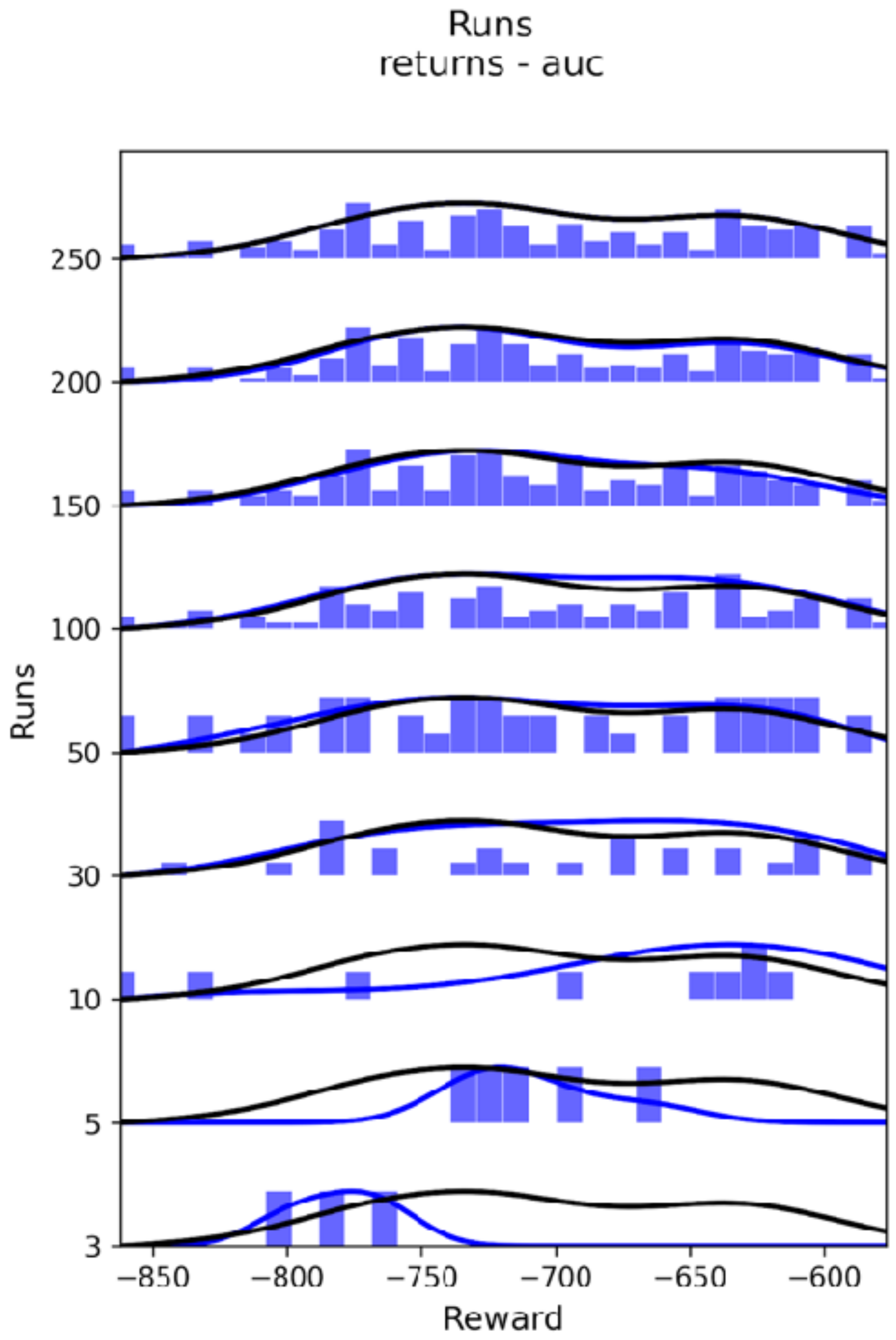
- In the prior plots we always ran 100k steps, and looked at the dist with more and more runs
- We can also look at the dist with more and more episodes (MC) …

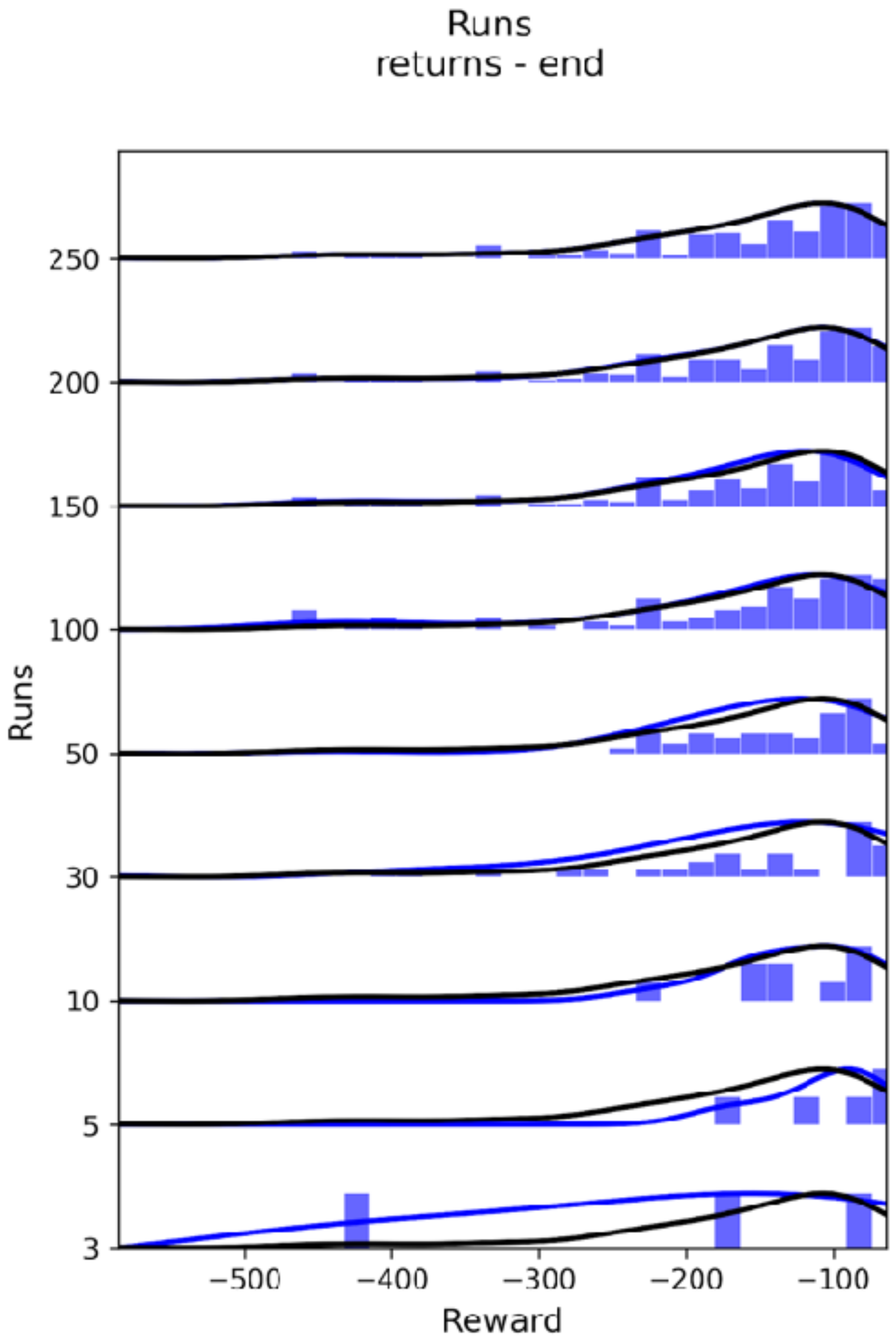


With and without cut-offs

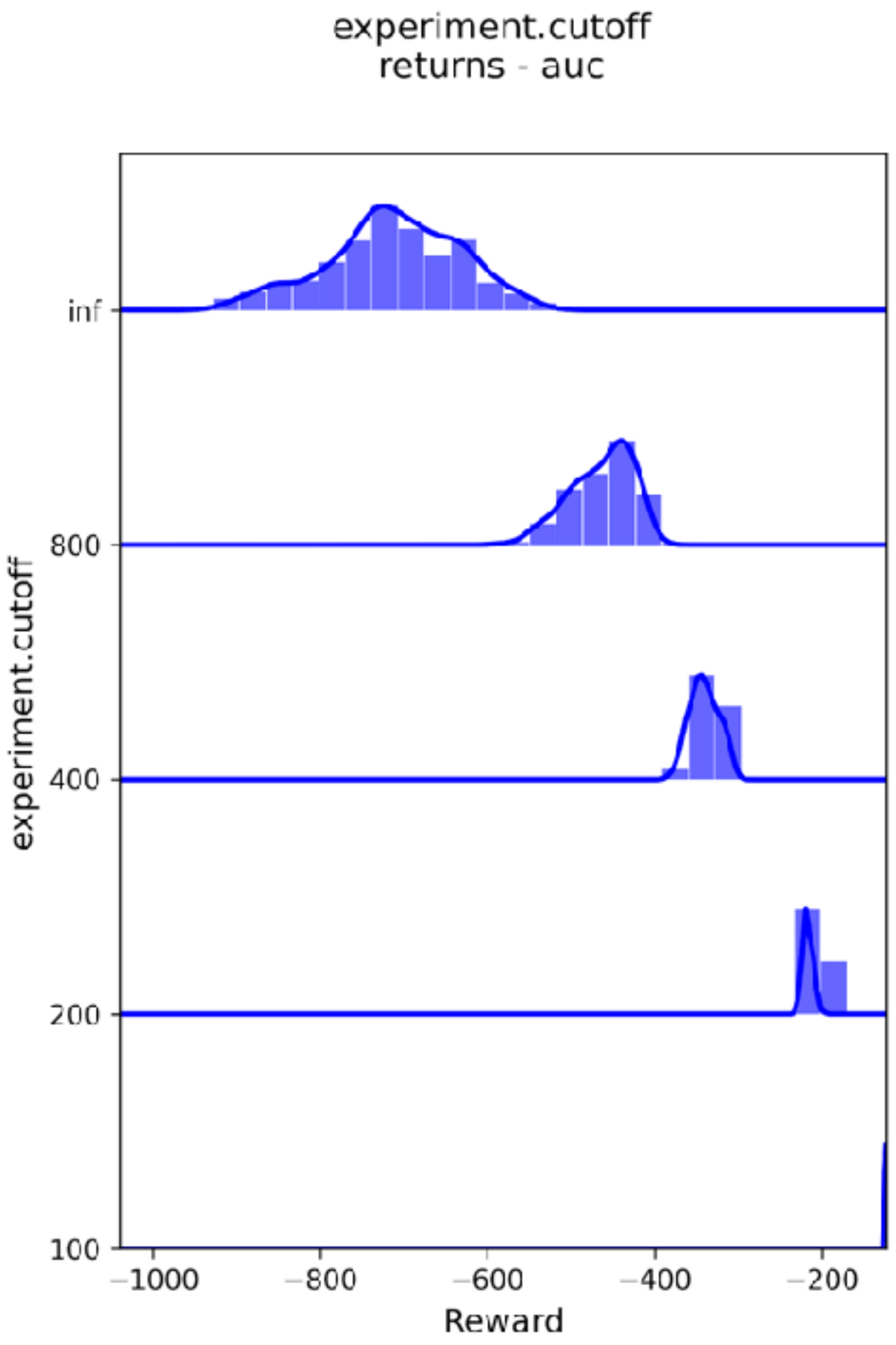


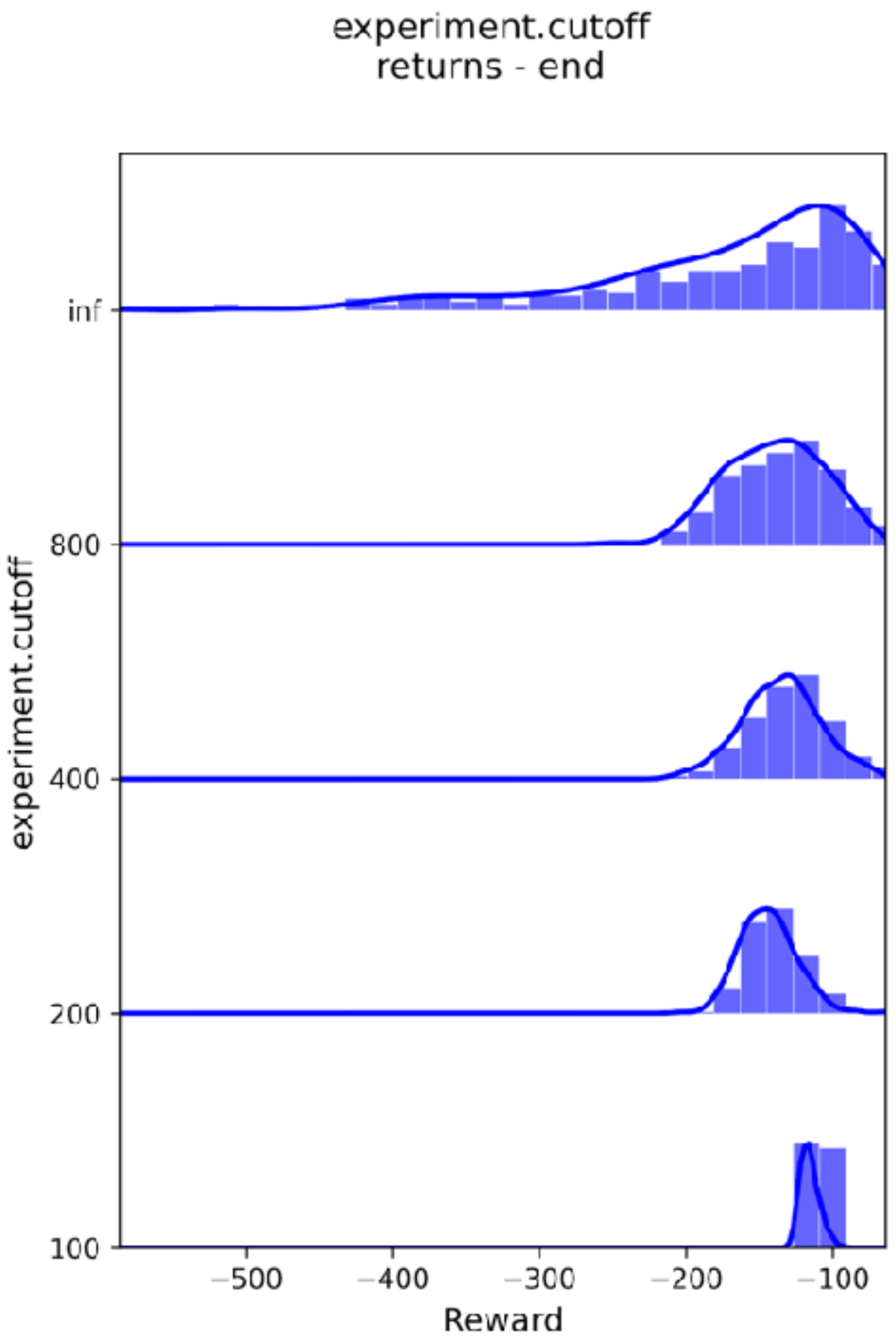
In puddle world we see impact of performance metric



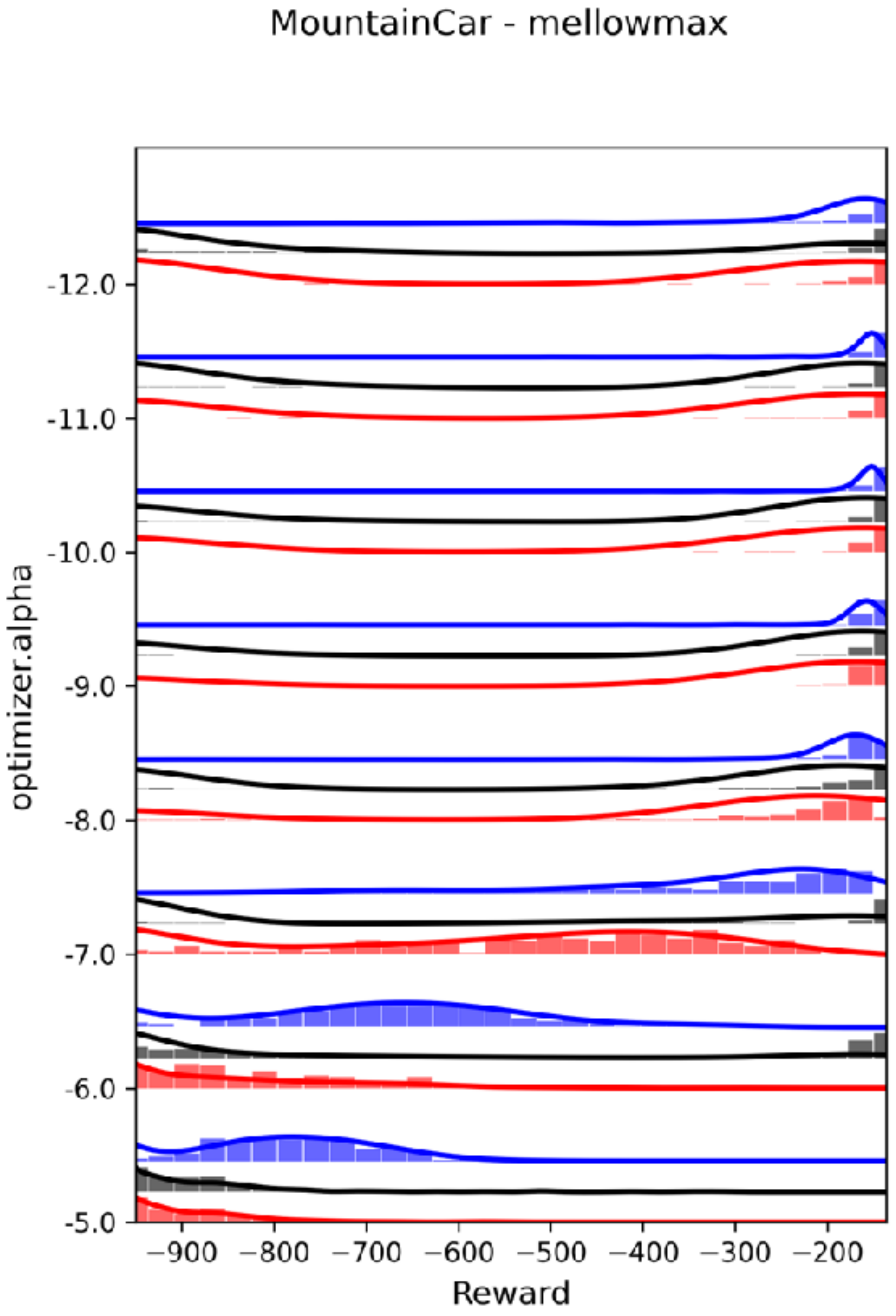


A closer look at cut-offs in puddle world





Bi-modality can even happen without explicit effort

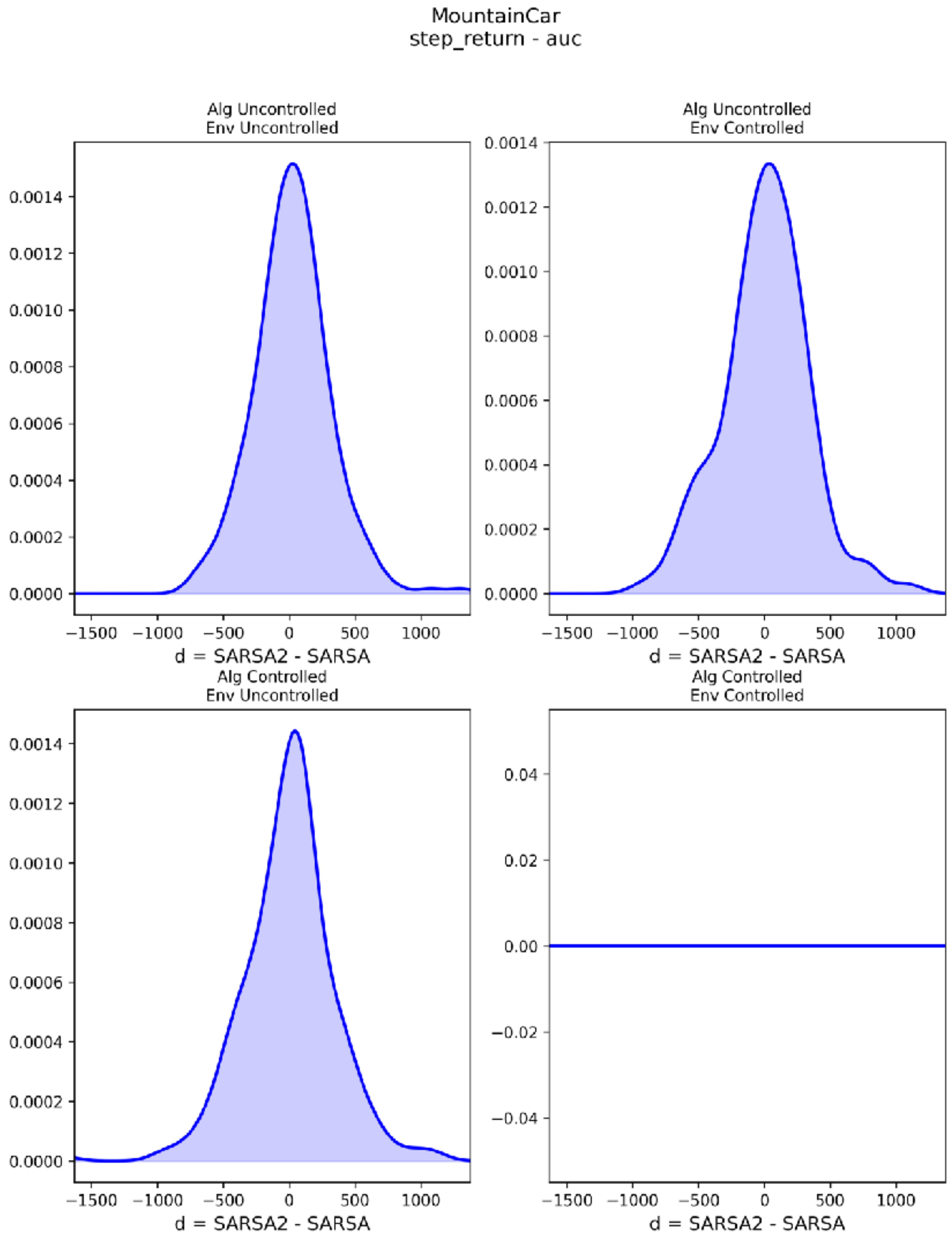


- Mountain car with 3 different algorithms and a Neural Network (2 layer, 32 hidden units, relu)
- Max episode length=1000, 100k steps total
- Agent hypers:
    - epsilon=0.1
    - Adam with beta_1 = 0.9 and beta_2=0.999
    - buffer_size = 4000, batch_size=32
    - No target nets

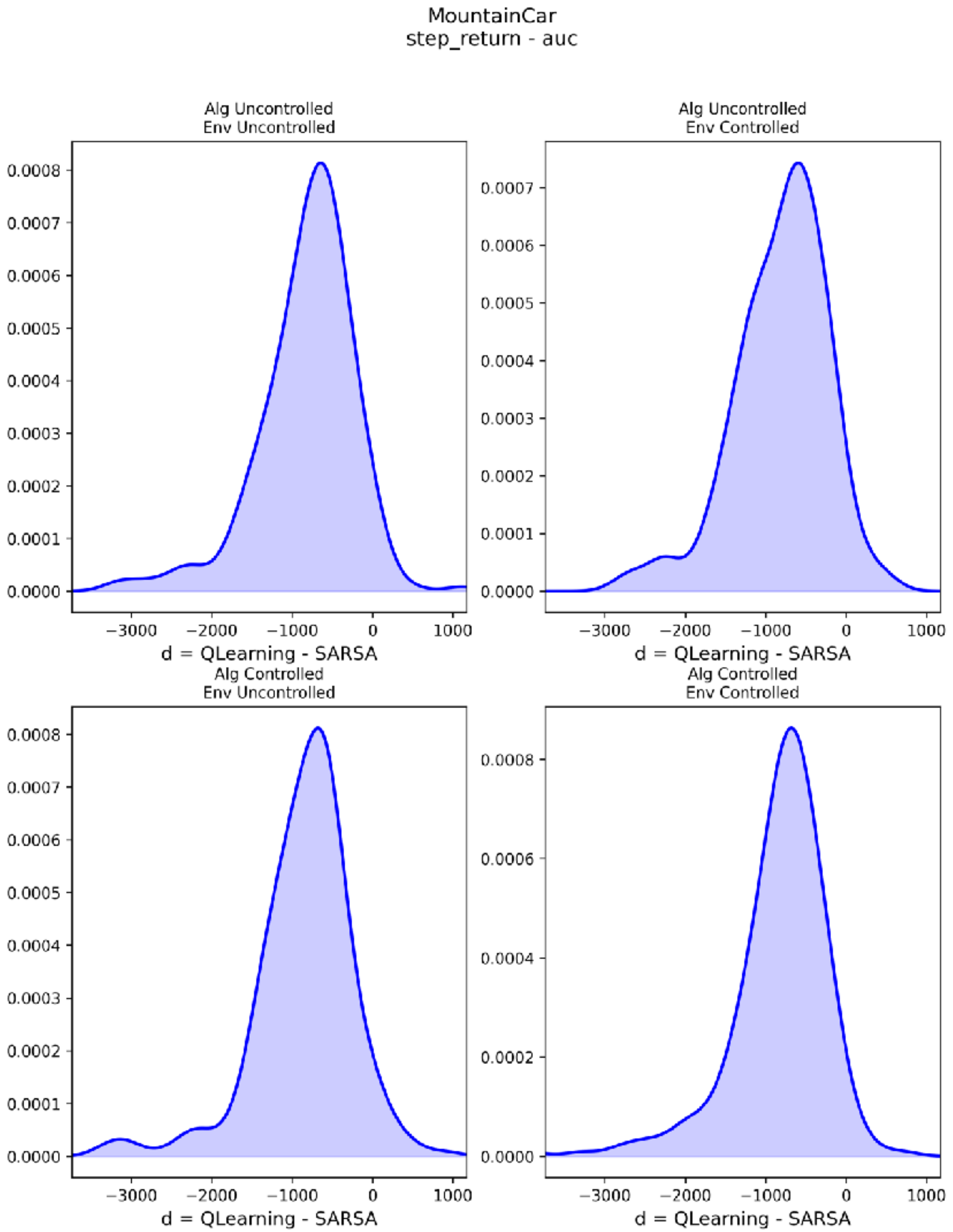
Controlling randomness

- Typically both the agent and environment have different sources of randomness:
  - In mountain car the start states, and epsilon in the agent for example
- We can decide to control these sources of randomness or not:
  - Controlled means the seed to the agent/env random number generator is set with the run_number
- There are 4 possibilities for controlling and not controlling each

Controlling randomness: comparing the same algorithm (250 runs)



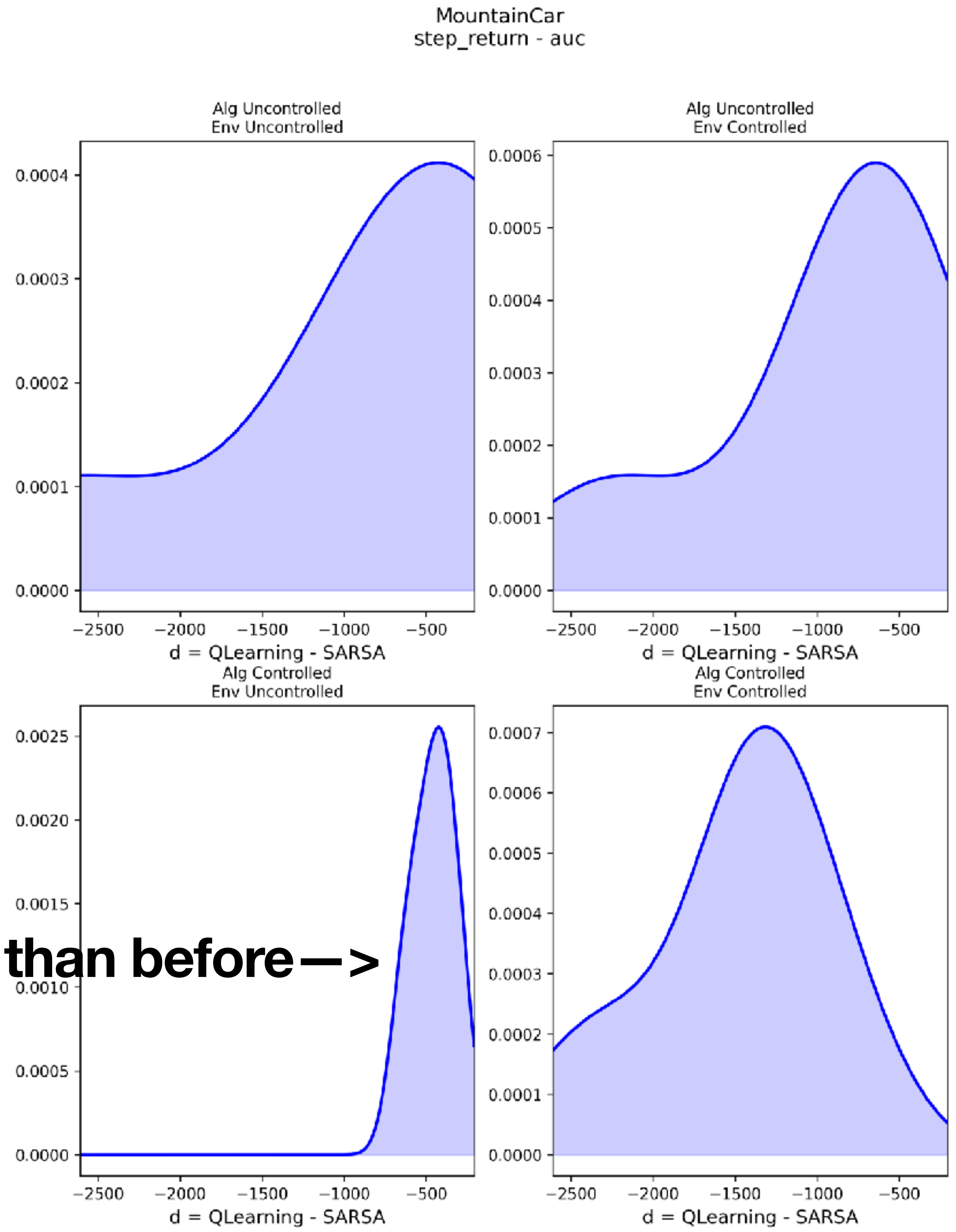
Controlling randomness: comparing Q-learning and Sarsa



Sarsa > Qlearning here

Controlling randomness: comparing Q-learning and Sarsa

but with only 5 runs



Qlearning looks better than before—>

Why it all matters

- We can’t always show all the data
- Worse: depending on experiment, environment, and agent design choices the data will all be different
- We will be left with mountains of data; dozens of plots
- That’s no fun for us, and certainly no good for a paper
- We want to aggregate the data, and use statistical tools like hypothesis tests and confidence intervals to make broader conclusions

You can’t just compute error bars and report p-values blindly

Hypothesis testing

- Let's say we draw samples from two populations, with true means m_0 and m_1
- We estimate the mean of each population: $\bar{x}_0$, $\bar{x}_1$
- Then we want to determine if the populations have different means
- We use a hypothesis test:
  - Null hypothesis: $m_0 - m_1 = 0$ (the true means are the same)
  - Alternative hypothesis: $m_0 - m_1 \neq 0$ (the true means differ)
    - **We want to reject the null hypothesis!**

How probable is it to observe this sample or a more extreme one, given that there is **no true difference in the performances of both algorithms?**

The **p-value is that probability:** to reject the null we want it to be extremely unlikely that we observe differences in the sample means given that the algorithms indeed perform the same!

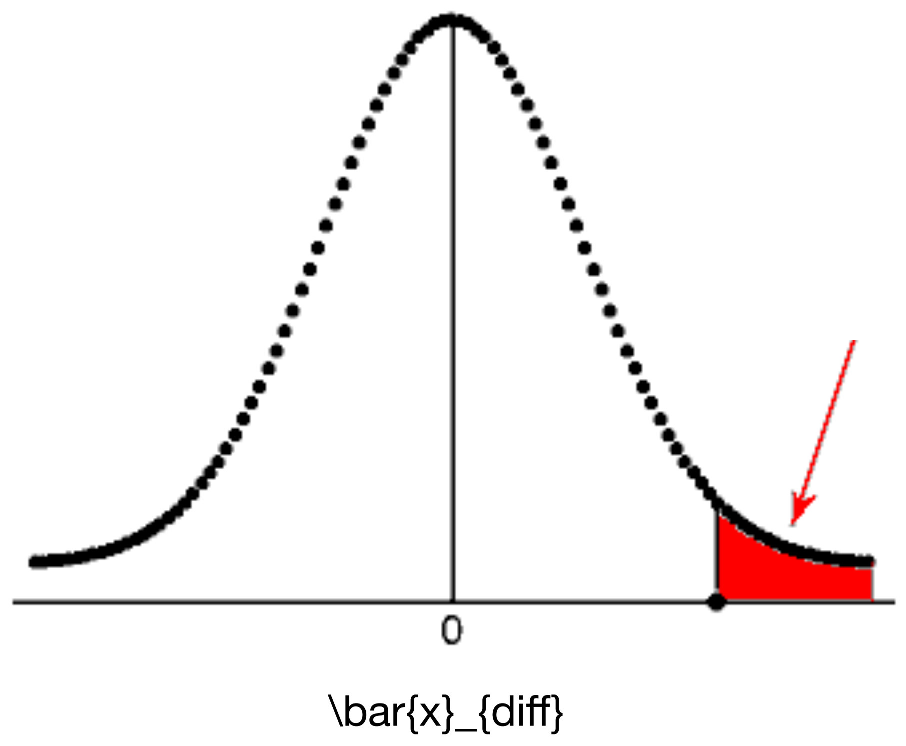
If your p-value is large, then your evidence (data) does not provide enough support to reject the null

Hypothesis testing

- Let X_1 be the random variable denoting the performance of algorithm_1
- Let X_2 be the random variable denoting the performance of algorithm_2
- If we assume X_1 and X_2 are normally distributed
  - Therefore $X_{diff} = X_1 - X_2$ is normally distributed
- We want many samples of X_{diff} (say 30 or more)

Hypothesis testing procedure

- **Let** $X_{diff,1}, X_{diff,2}, \ldots$ be a sequence of RV representing runs of the experiment and $\bar{X}_{diff}$ = average of $X_{diff,1:n}$
- **True distribution** over the differences: $\bar{X}_{diff} \sim p_{true}, i.e., p(\bar{x}_{diff})$ is density
- **Sample** $\bar{x}_{diff,0}$ // we run an experiment
- **Assume** null hypothesis: p_{null} is defined such that $\mathbb{E}[\bar{X}_{diff}] = 0$
  - This is the hypothesized model of p_{true}
  - E.g., p_{null} might be a mean-zero Gaussian over $\bar{x}_{diff}$
- **Question**: how likely is $\bar{x}_{diff,0}$ under H_0
  i.e., how likely is it that we would see $\bar{x}_{diff,0}$ or a more extreme value:
  $p_{null}(\bar{X}_{diff} > \bar{x}_{diff})$ (if unlikely, then our model likely wrong)



Is the difference significant?

A difference is called significant at significance level \alpha/2 when the p-value is lower than \alpha/2

Key assumptions in hypothesis testing

- We most often use a t-test (and standard error bars)
- ~~**They assume the distributions of performance are Normal**~~
- Performance is measured at random and independently from one another (each agent)
- Same sample size
- Continuous and bounded performance distributions
- ~~**Equal standard deviations**~~