Start recording …

Real-time learning, from human reward!

Predictive knowledge on a robot

“Wall ahead” is a sensorimotor fact

Predicting: Will rolling forward soon result in a bump?

bump pred bump data

Predicting right and left bumps

pred data

left bump

both bump

right bump

Admin

- **Next week is spring break; no lecture, no office hours**
- Session moderators for today: **Bashir, Zahra**
  - https://docs.google.com/spreadsheets/d/1dbmlvduupZUCDjxU4HW2_350OVrVG-g1FoEAG-uWhMk

The Data of RL

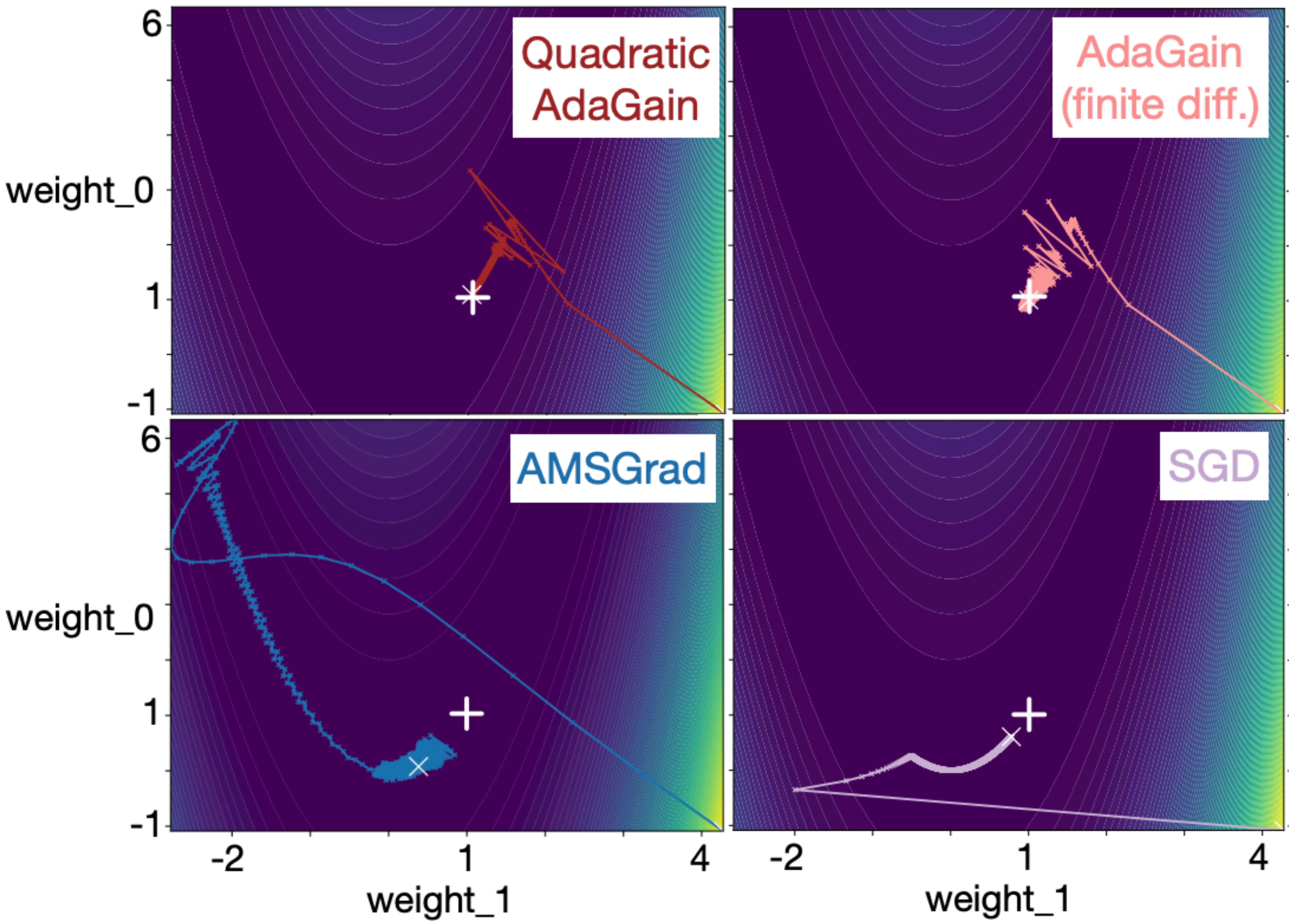
Imagine you developed an new algorithm

- 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?
  - The limitations of the idea?
  - Lastly, if it is better in some measurable, reliable, relevant way?

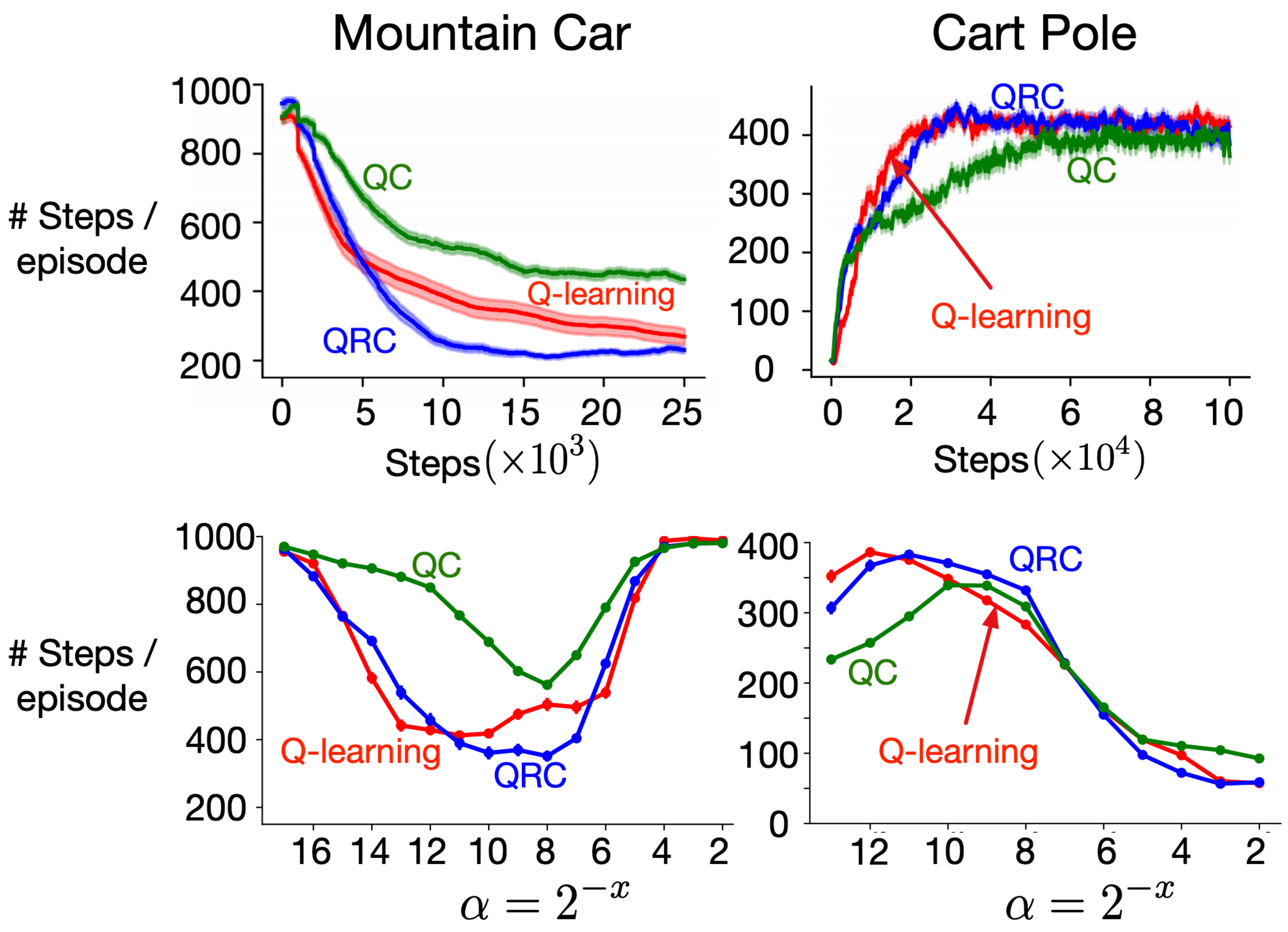
Start with the problem

- 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



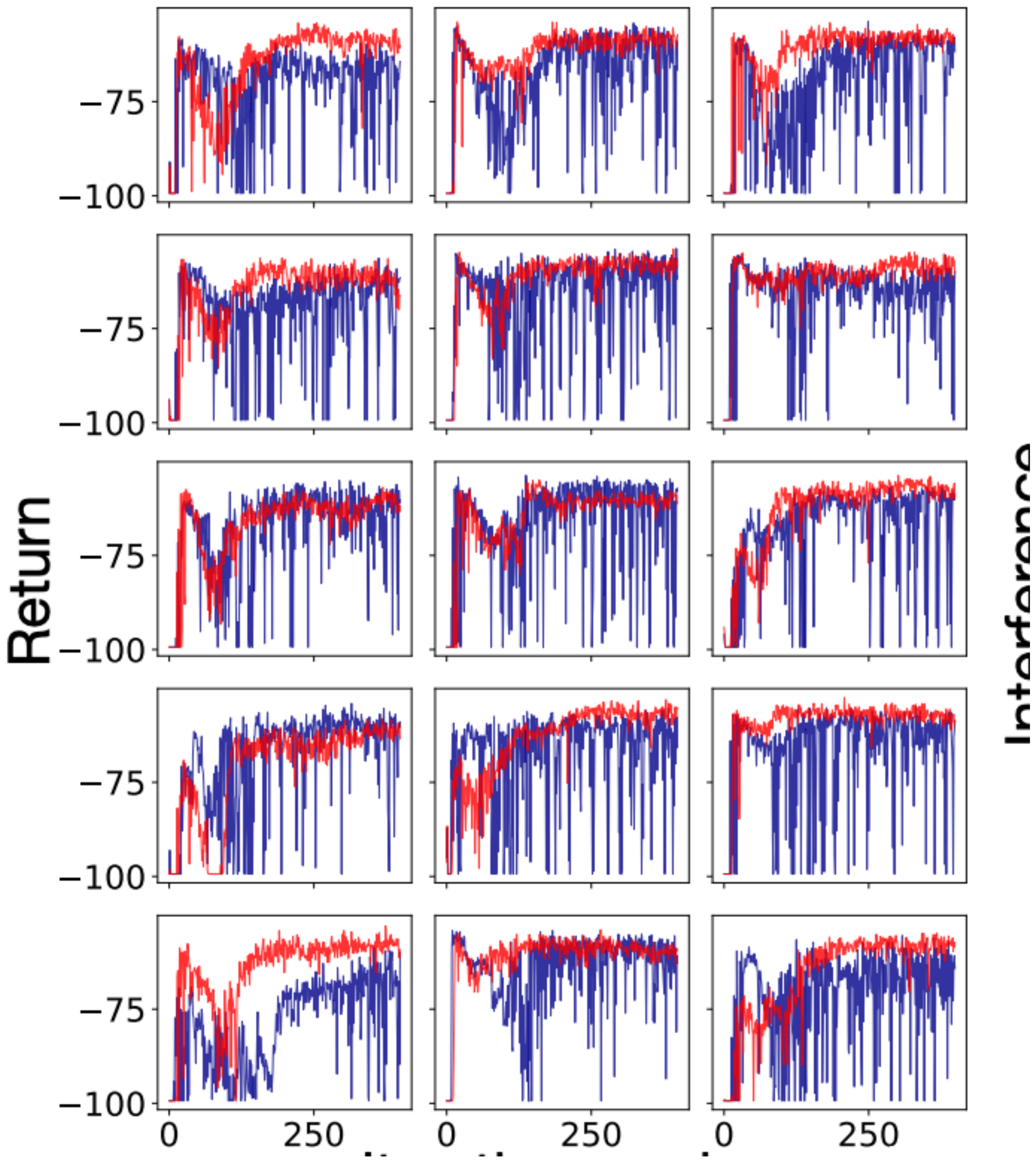
Example: sound off-policy control



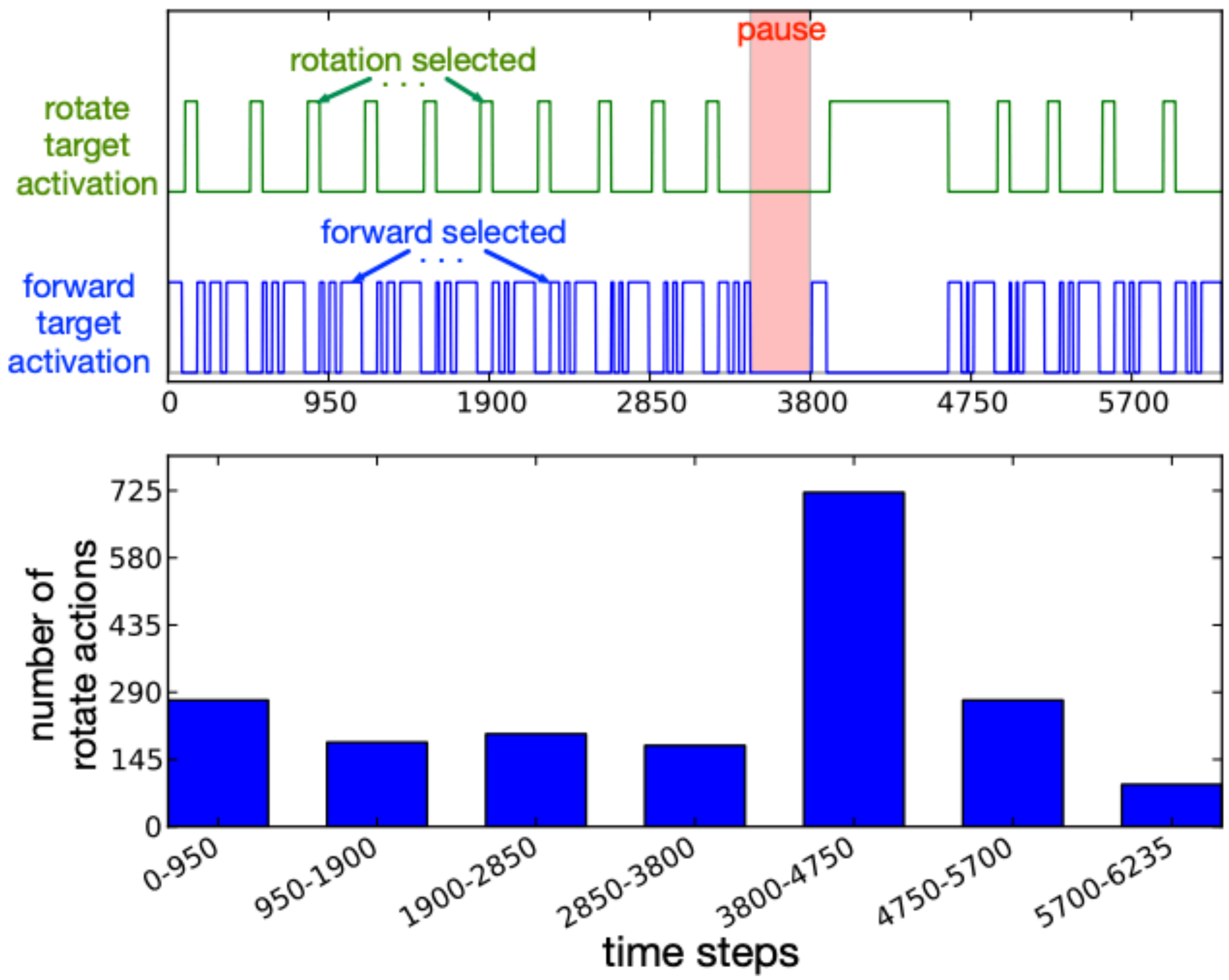
What to measure, what to plot?

- 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!

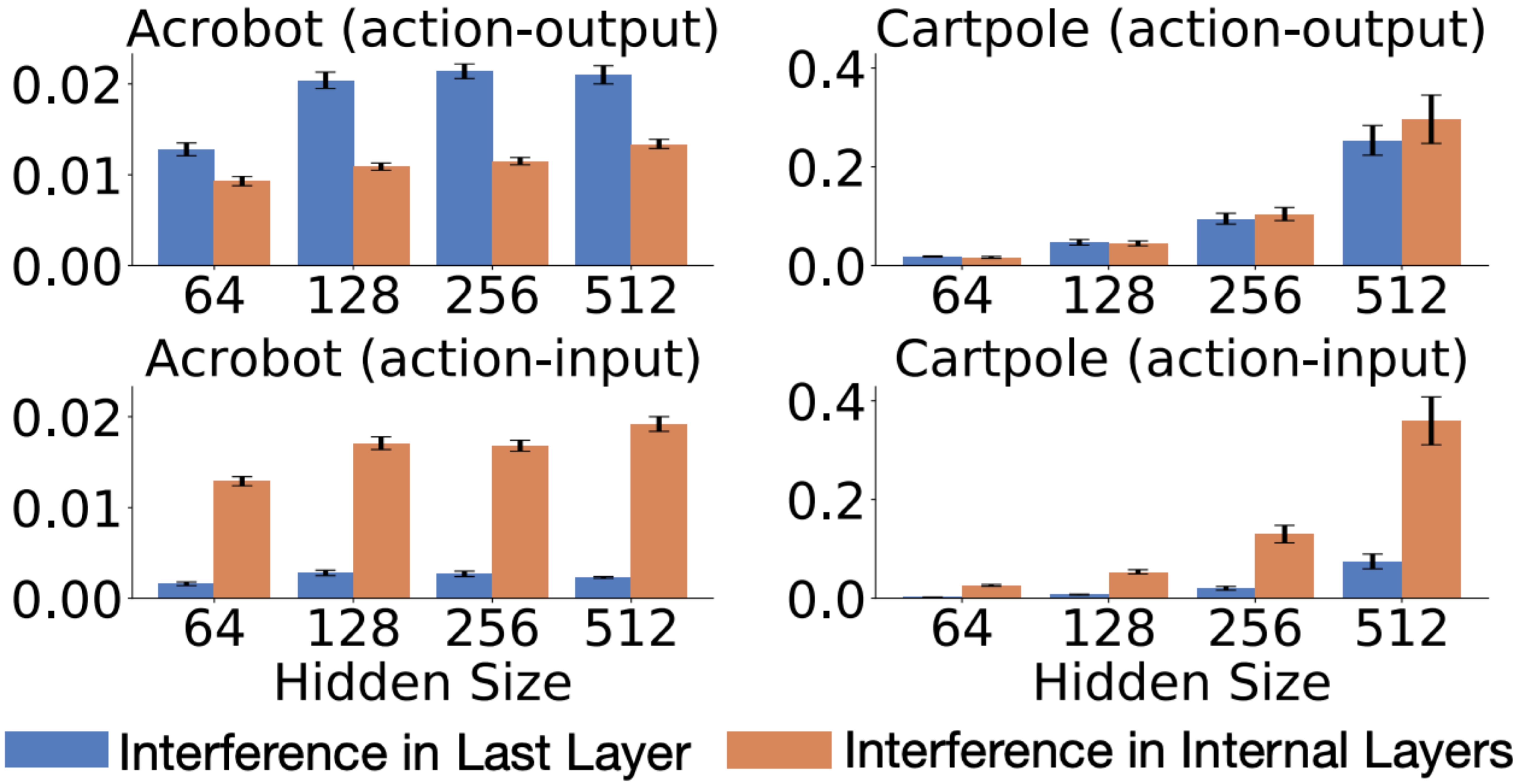
Example: a more stable control algorithm



Example: clear change in behavior



Example: where interference in happening in a network



Ultimately we end up comparing things

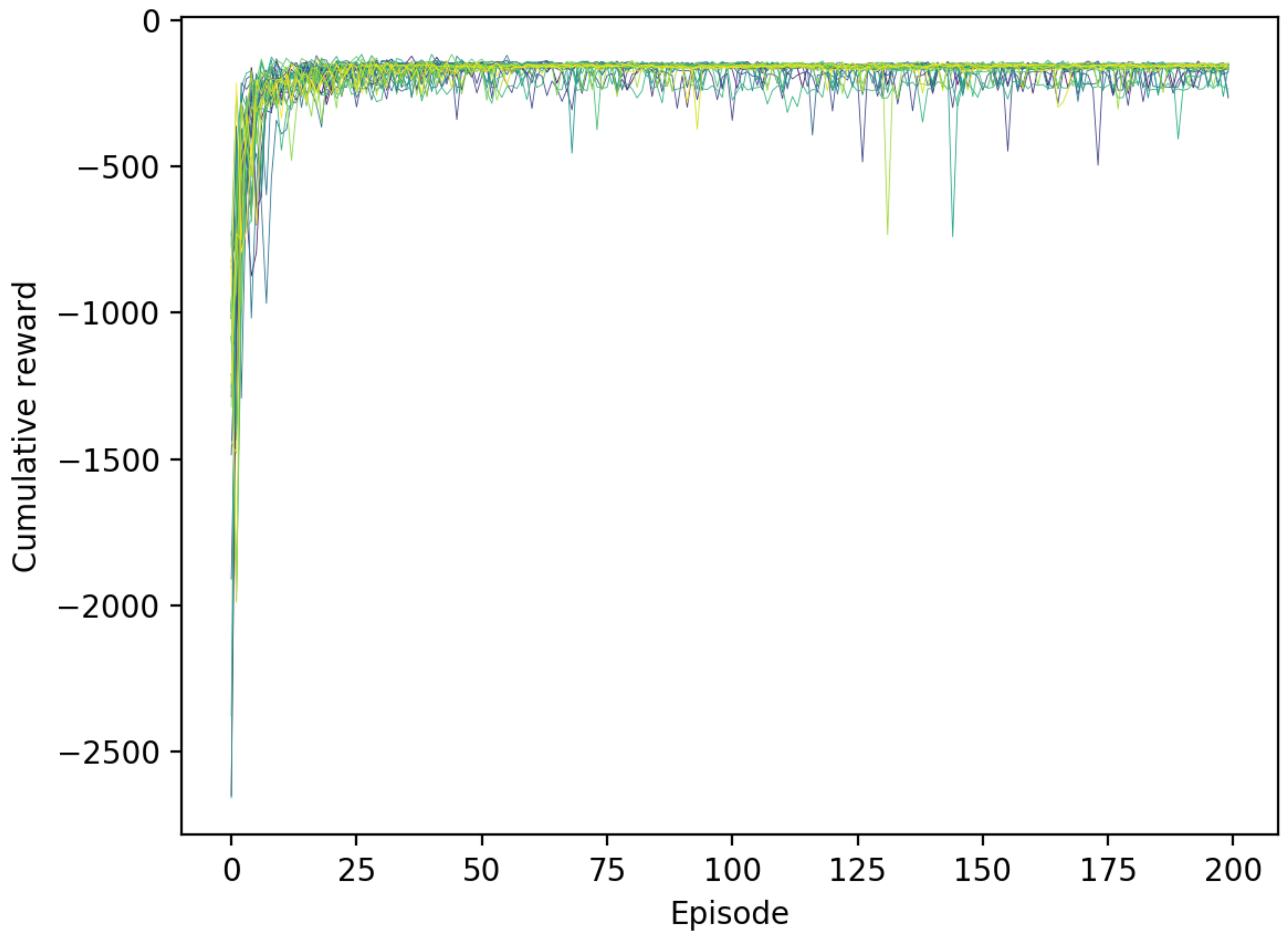
- SOTA competitor, natural baseline, or calibration agent
- We need to measure something & compare agents
- This is not about winning and losing … its about telling the story of the data
- To tell the story accurately:
    - Properly reflect uncertainty
    - Properly how hard it was to get good performance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

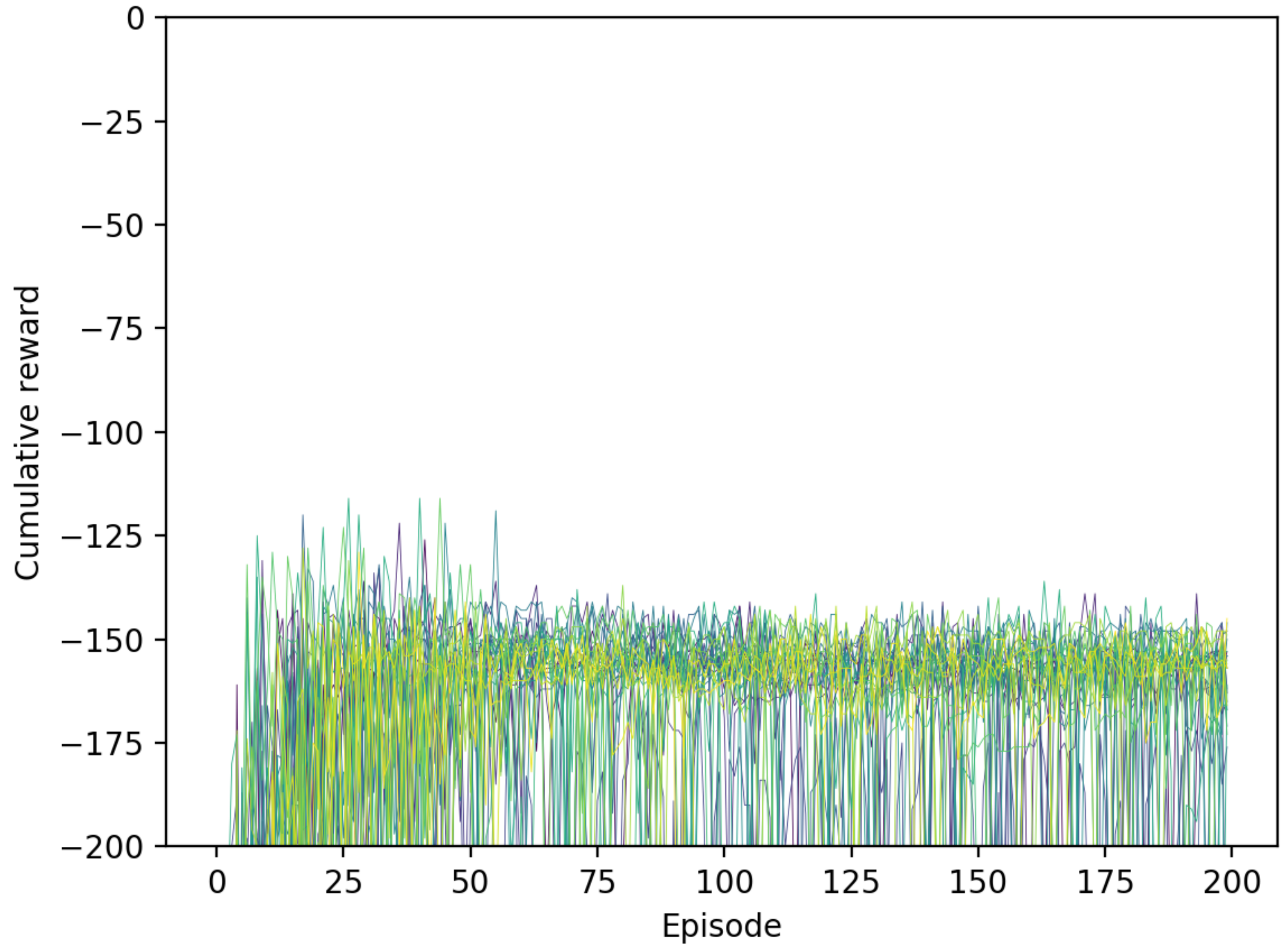
Ultimately we end up comparing things

- 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 reflect uncertainty
    - Properly how hard it was to get good performance
    - Properly reflect the impact of all choices
    - **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**
- Fixed start state, 0.5 decaying step size, 10 tilings 10x10

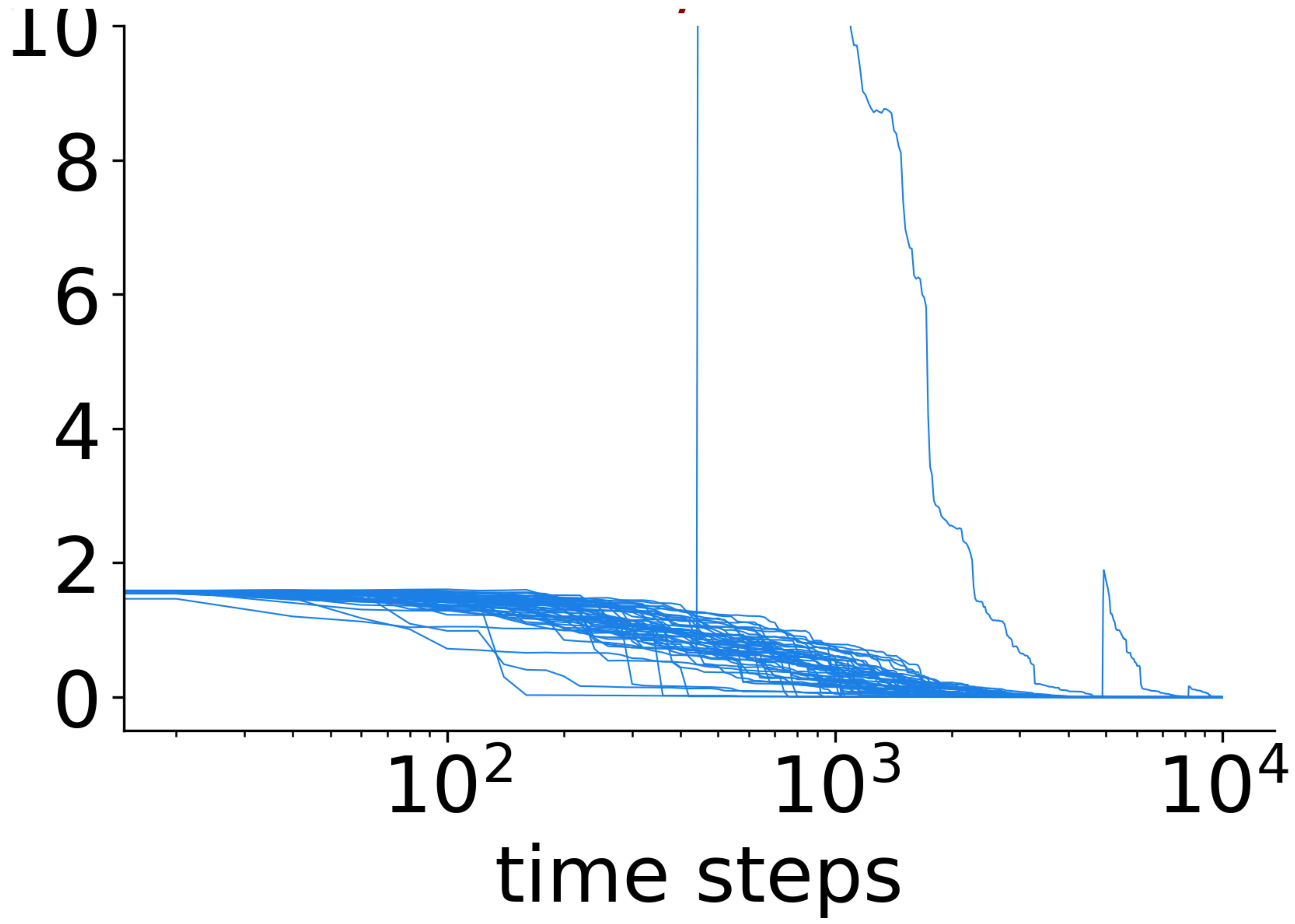




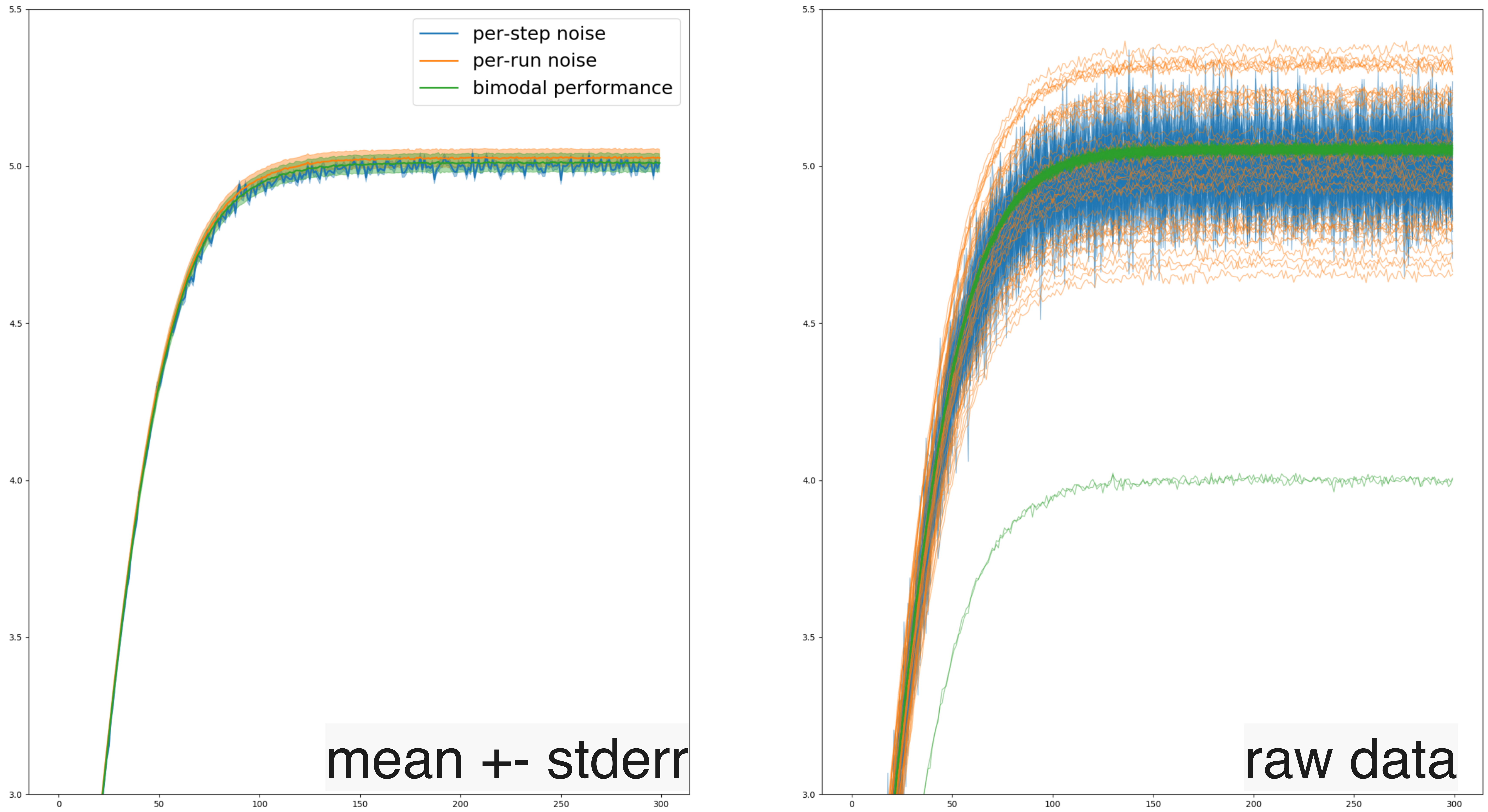
Without repetition we can say so little

- 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



The raw data can tell different stories

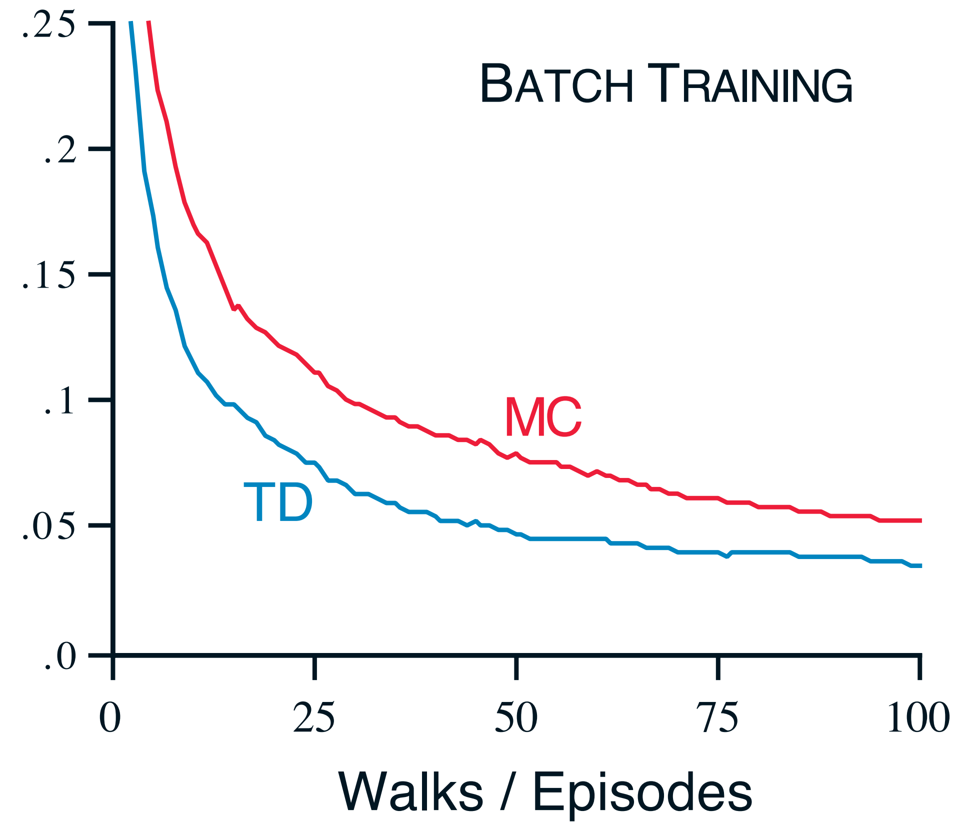


- 50 runs, 300 steps
- Credit: Andy Patterson

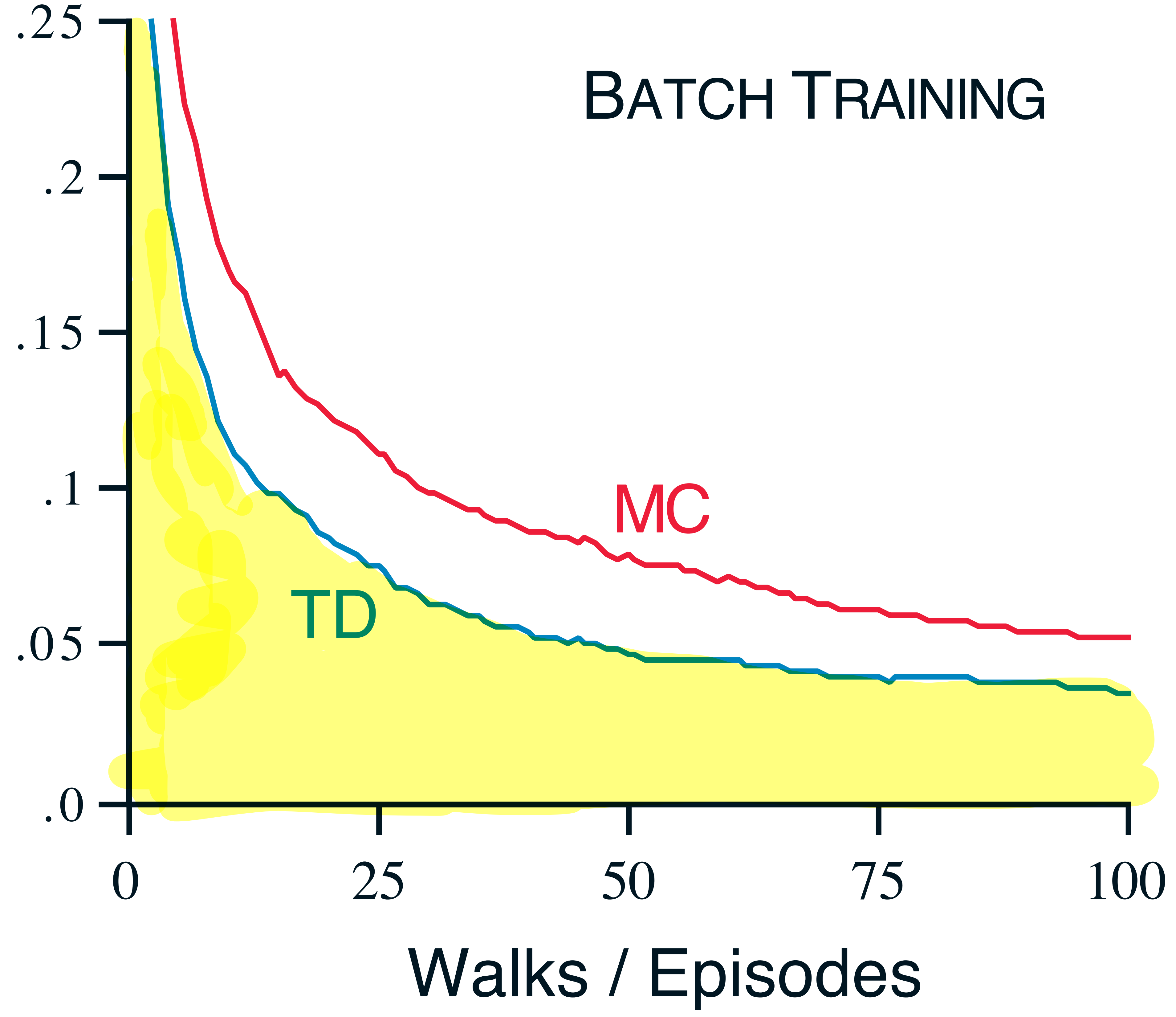
Which data/alg would you prefer?

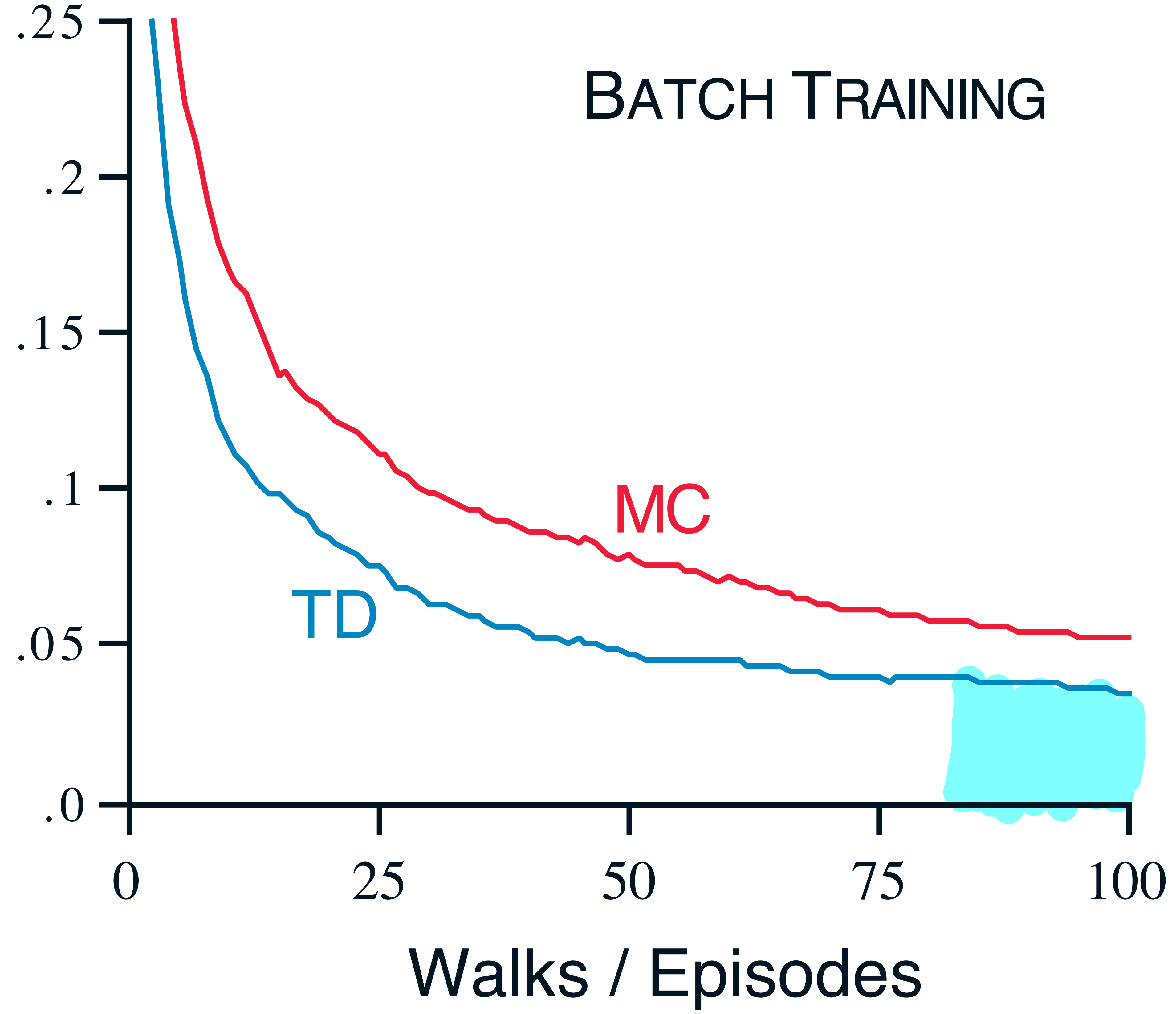
Agents & Environments are data generators

- 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?

Your questions

- Should we have a special conference / event for competitive testing?
- Generative vs Discriminative models: http://robotics.stanford.edu/~ang/papers/nips01-discriminativegenerative.pdf
  - There is a relationship here to TD vs MC
  - For the RL context think of Model-based methods (e.g., DP) vs model-free
- Models that allow temporal abstraction (thinking jumps); challenges
  - Discovery problem (where do the options come from)
  - Off-policy learning: learning option policy and models in parallel
  - Using them for planning: open question

Your questions

- In Deep RL the Matters paper: TRPO on swimmer -> bad policy
  - Can we check this automatically? What's the problem?
  - Can we build this into the environment? Isn't it already?
  - More examples
- Does Whiteson et al's generalized environments pose a problem for current theory of RL with function approximation?
- Is the average reward formalism or algorithms useful in episodic tasks?

Your questions

- Why do we need off-policy learning?
- Tips on writing
- Tips or ideas for visualization

Admin

- **Next week is spring break; no lecture, no office hours**
- Session moderators for today: **Tiriac,Valentin**
  - https://docs.google.com/spreadsheets/d/1dbmlvduupZUCDjxU4HW2_350OVrVG-g1FoEAG-uWhMk
  - **Your job is to ask questions and moderate discussion!**
  - If you cannot make your session, tell me ahead of time

Plan for today

- Continue our discussion about the distributions generated by RL experiments and why you should care
- A crash course in good presentations
- Discuss your questions & project standups

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?

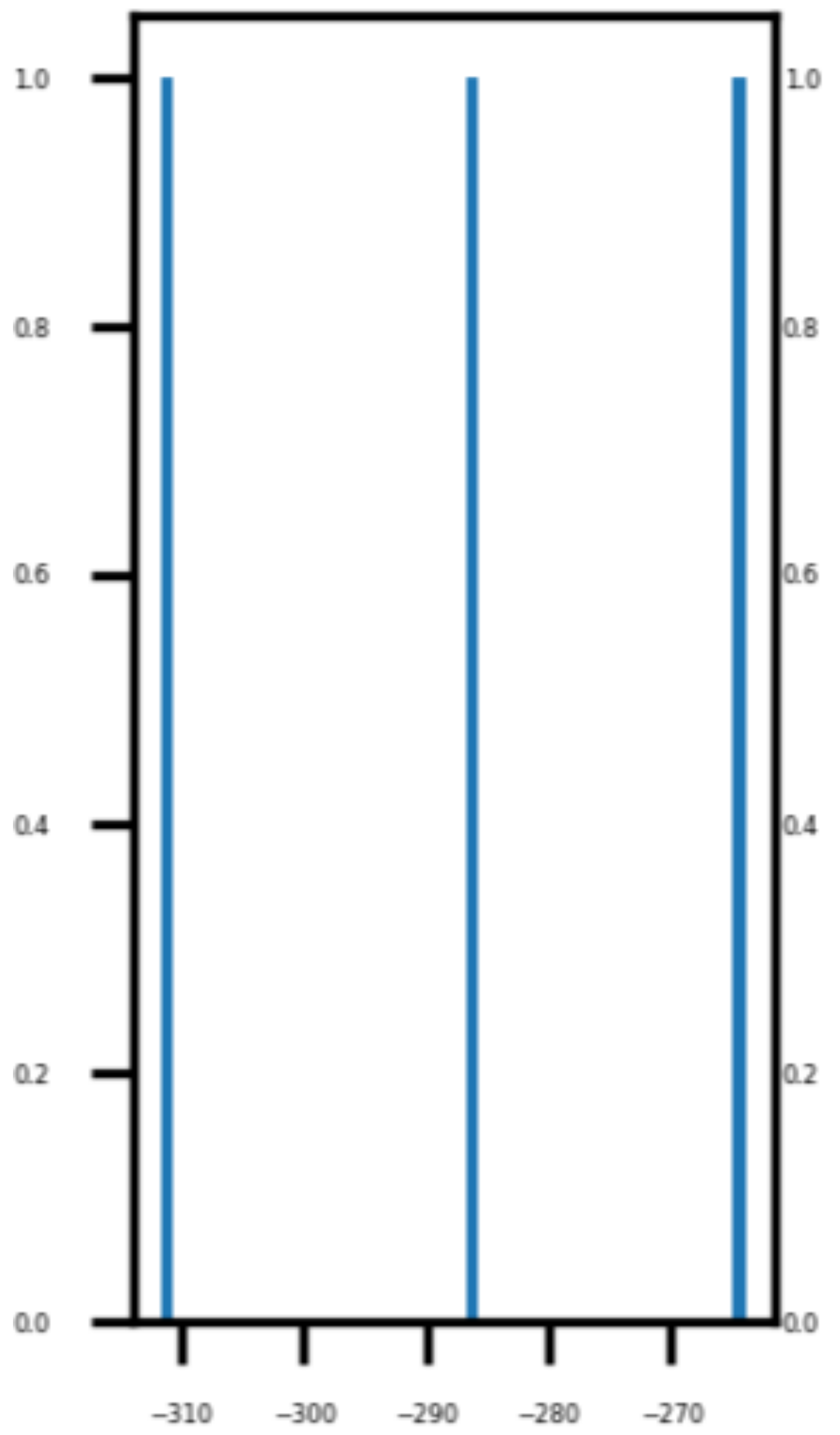
How many runs do we need?

- Common practice is 3
- 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?

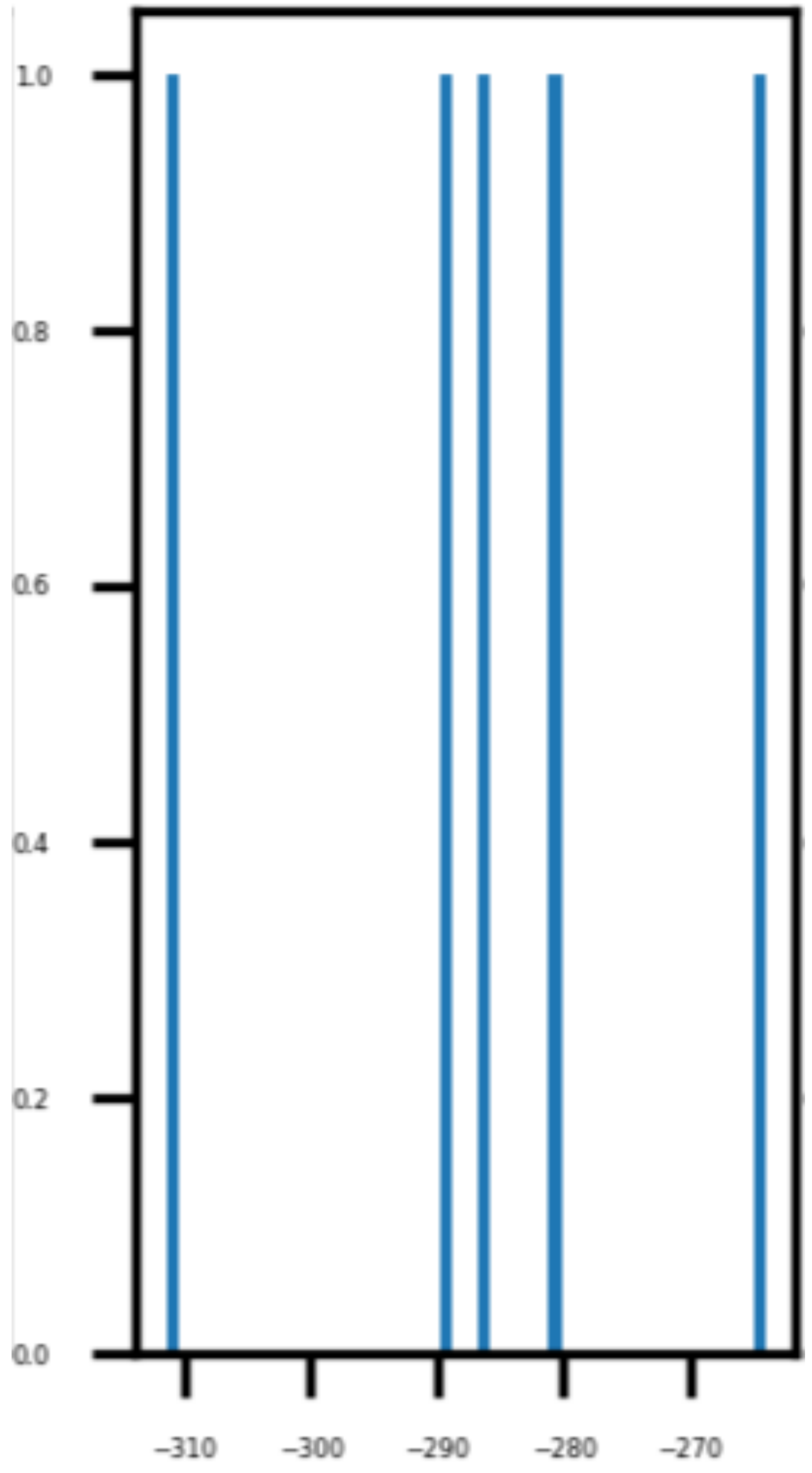
What if we did 3 runs?

- 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)

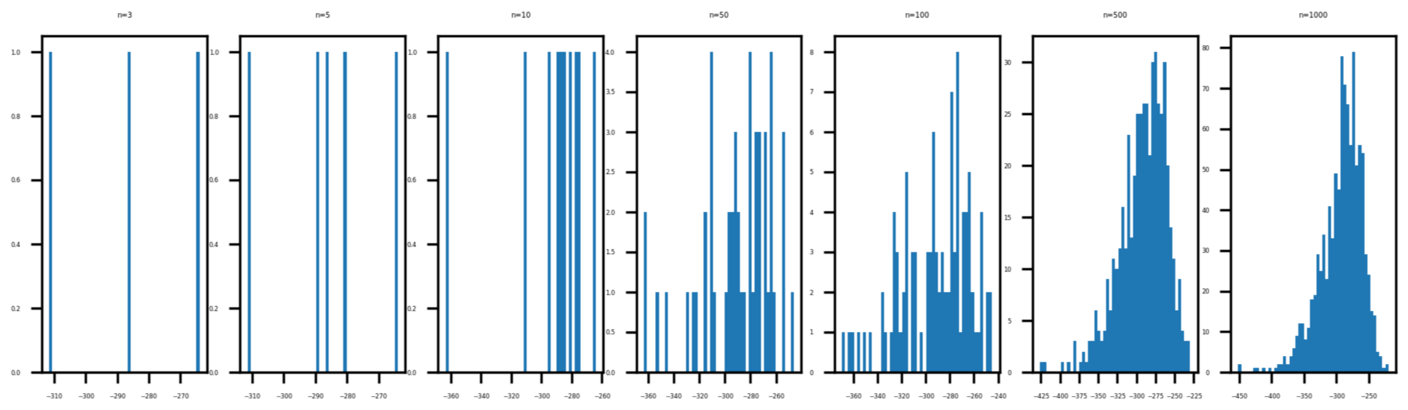


Many runs are needed to see the shape of the distribution

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

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

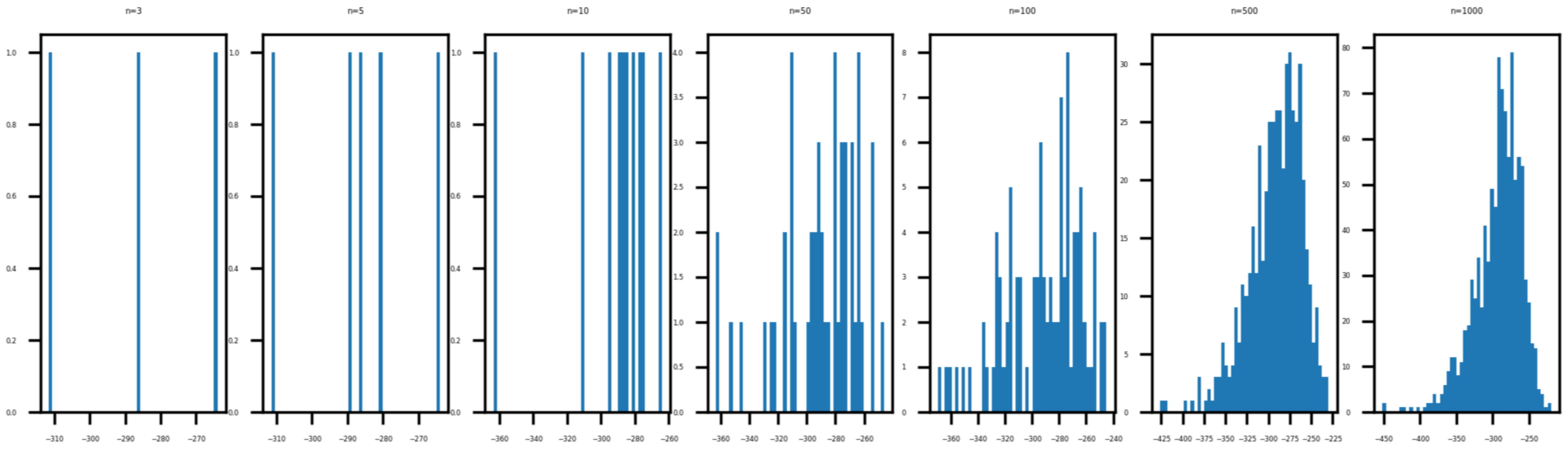
Environments design choices matter too



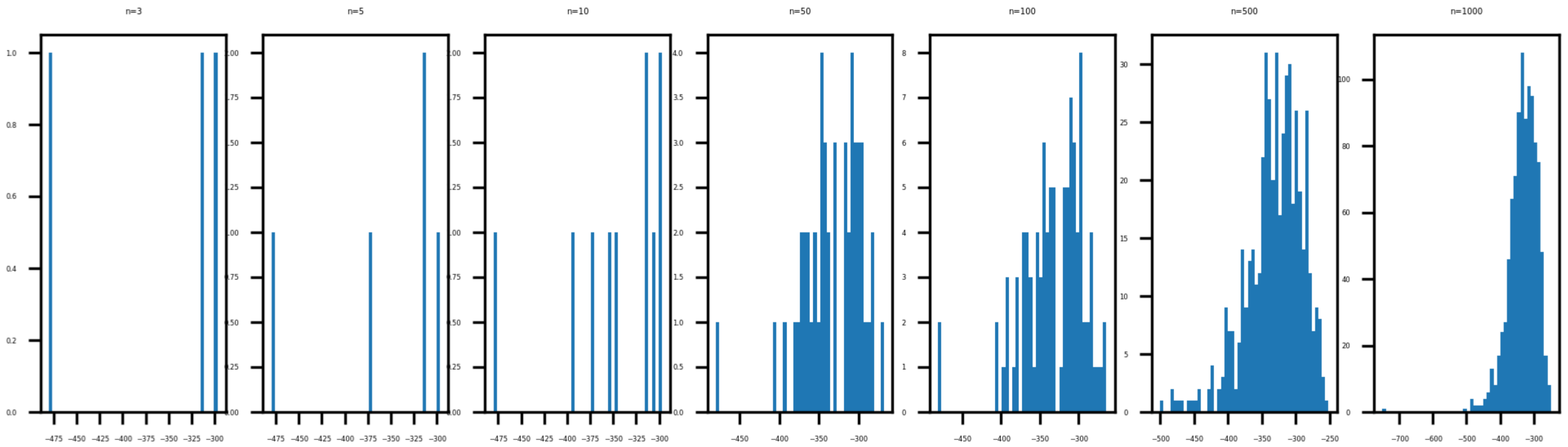
- Notice how the distribution was a bit skewed, not perfectly bell shaped
- We can get other distribution shapes by including cutoffs:
  - Restarting the episode if the agent reaches a max number of steps
  - This ensures the no episodes a really bad—might make bad agents look good
  - This gives free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

Cut-offs skew performance

Regular MC->



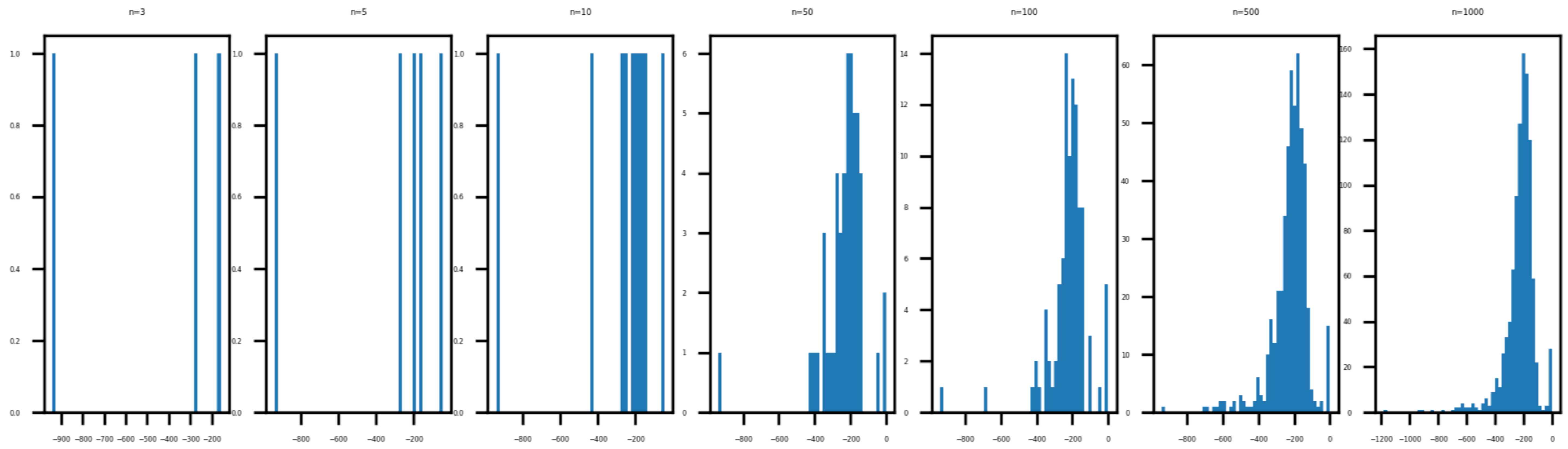
MC w cut-offs->



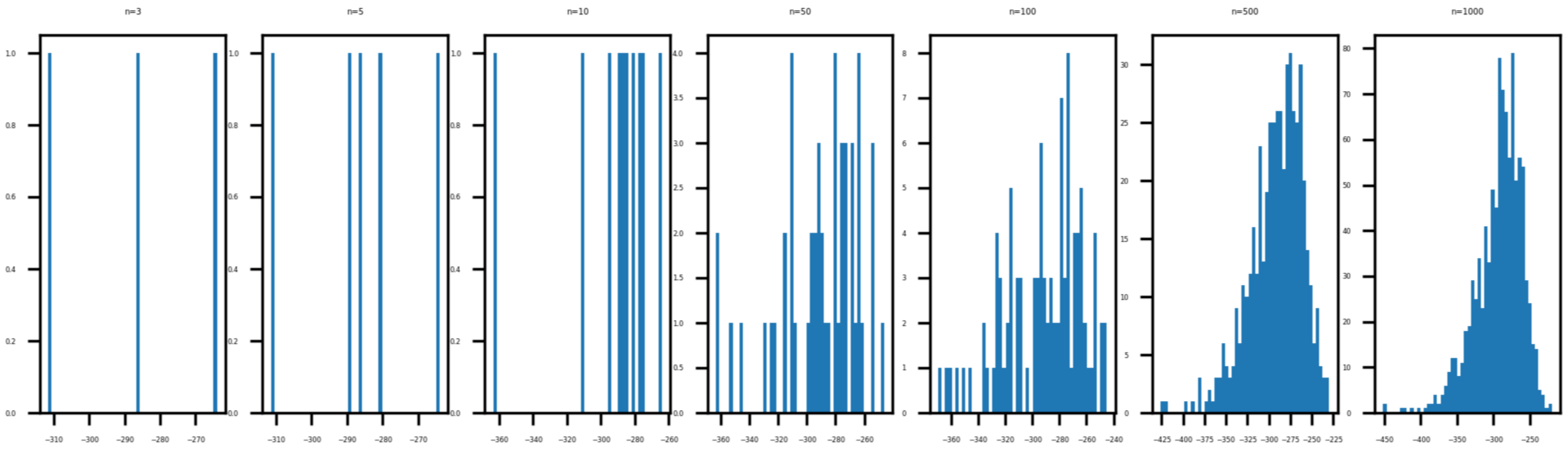
- 1000 step episode max

Every agent & environment pair can be different

- Same experiment and setup in Puddle world:

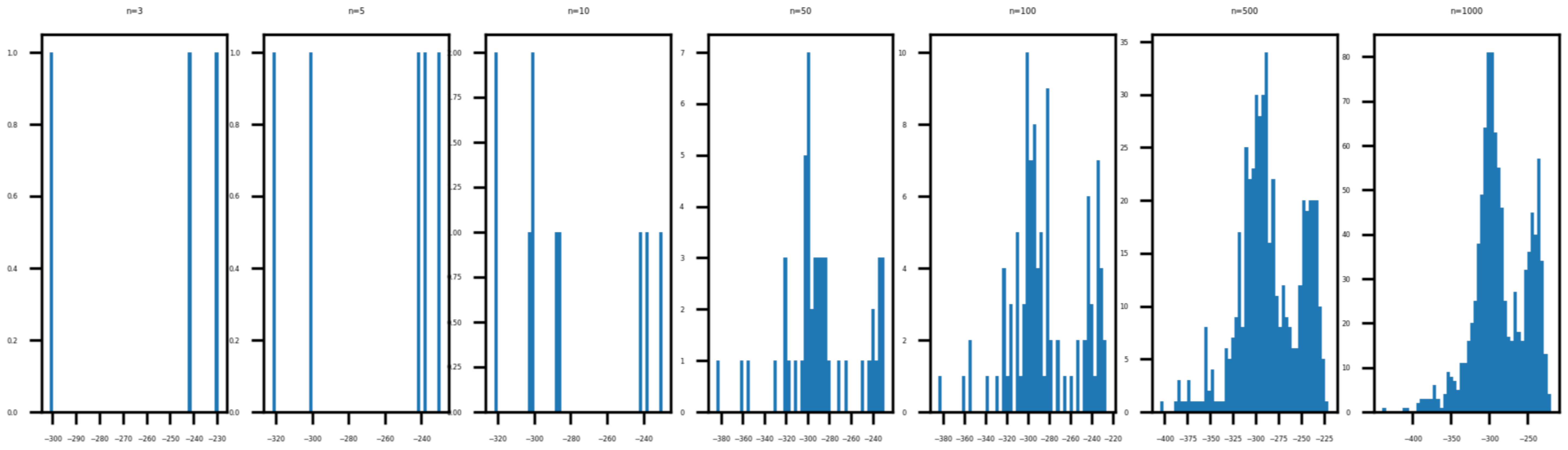


- Mountain car:

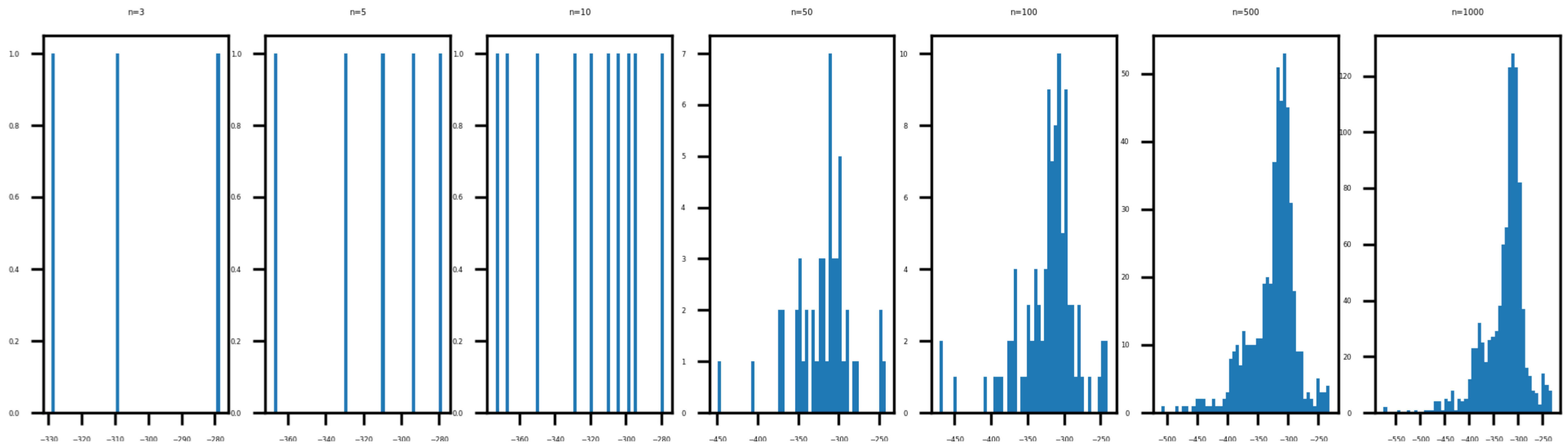


Design choices interact

- Mountain car with two different start states:

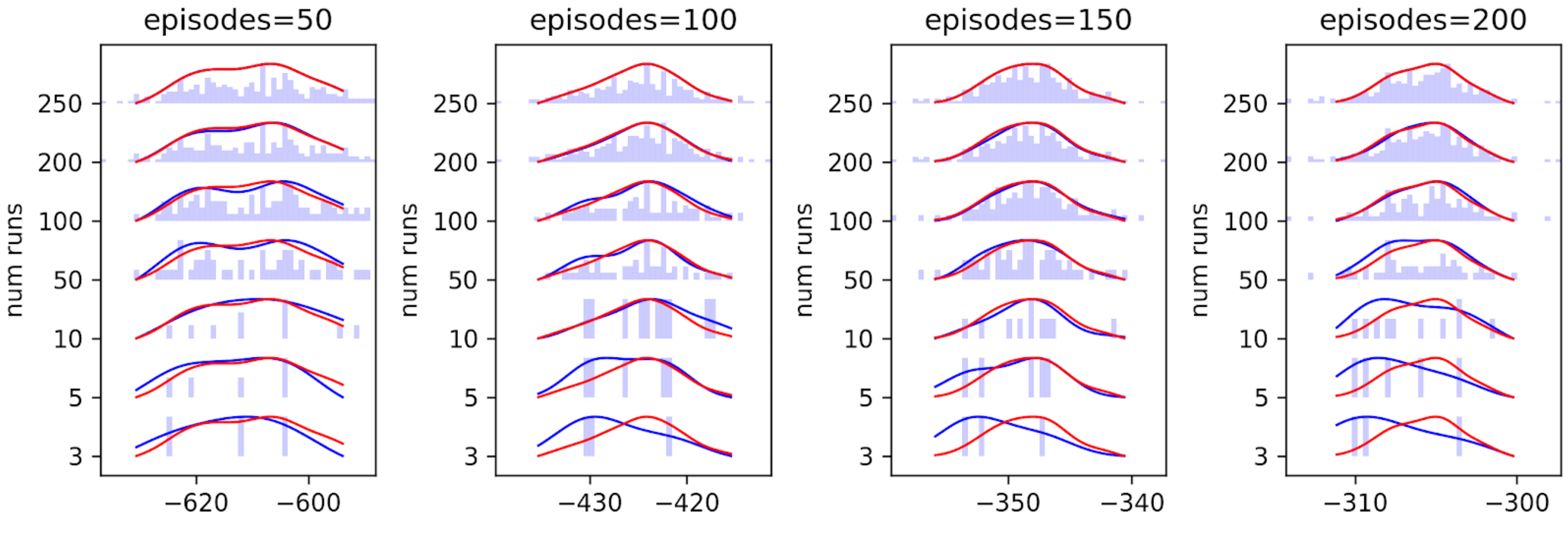


- Mountain car with two start states and cutoffs:

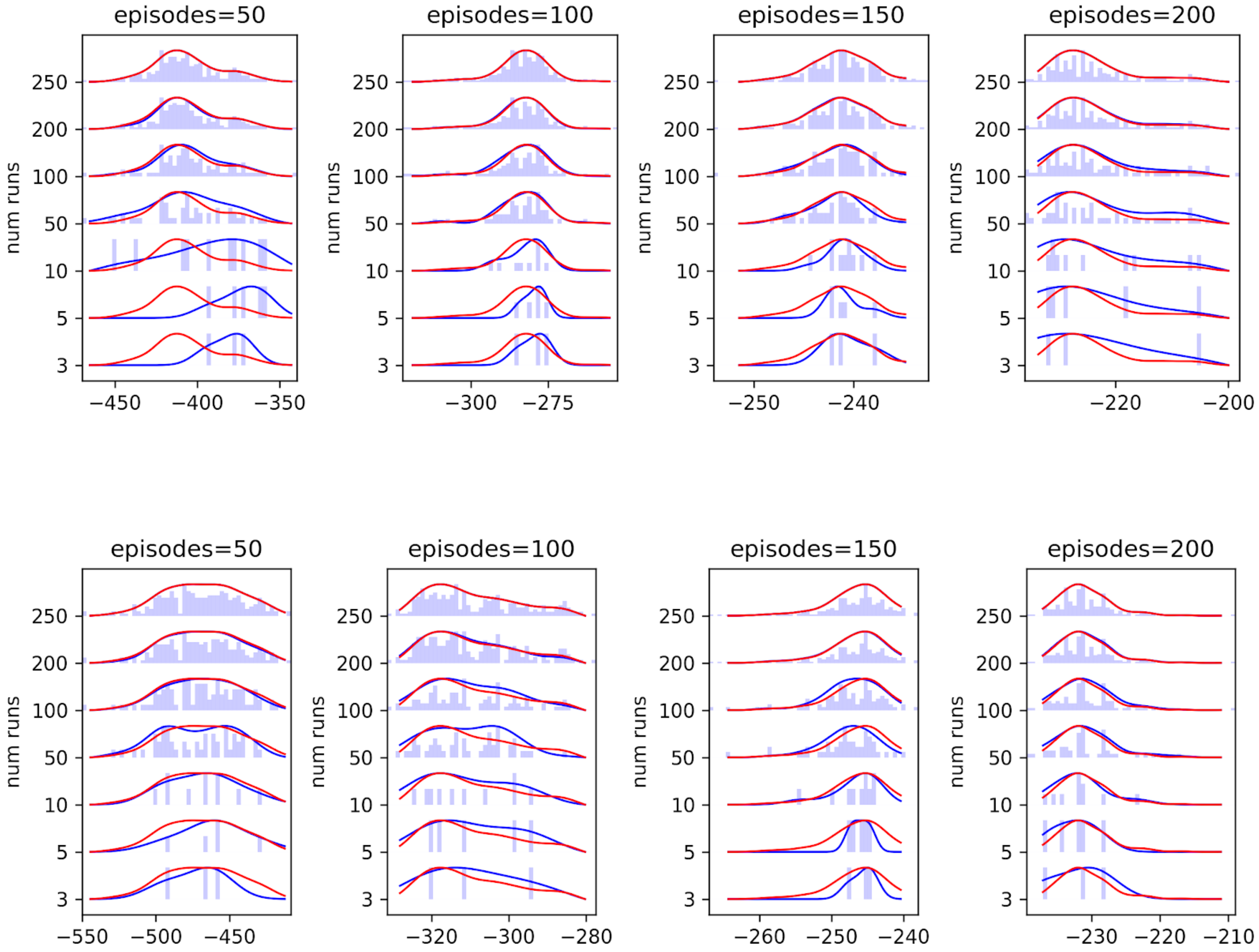


Experiment design choices interact too

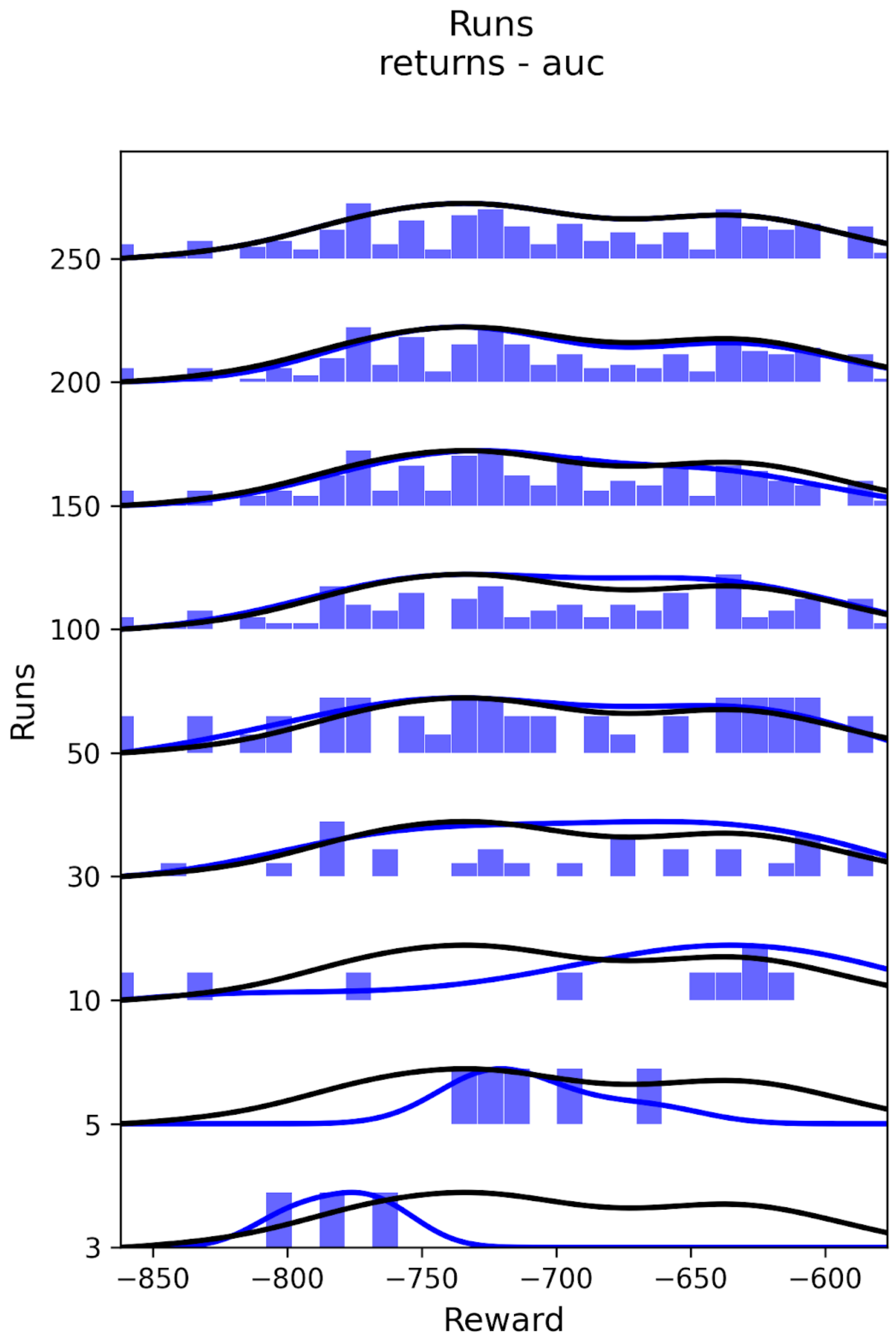
- In the prior plots we always ran 100k steps, and looked at the dist with more and more runs
- We can also look at the dist with more and more episodes (MC) …

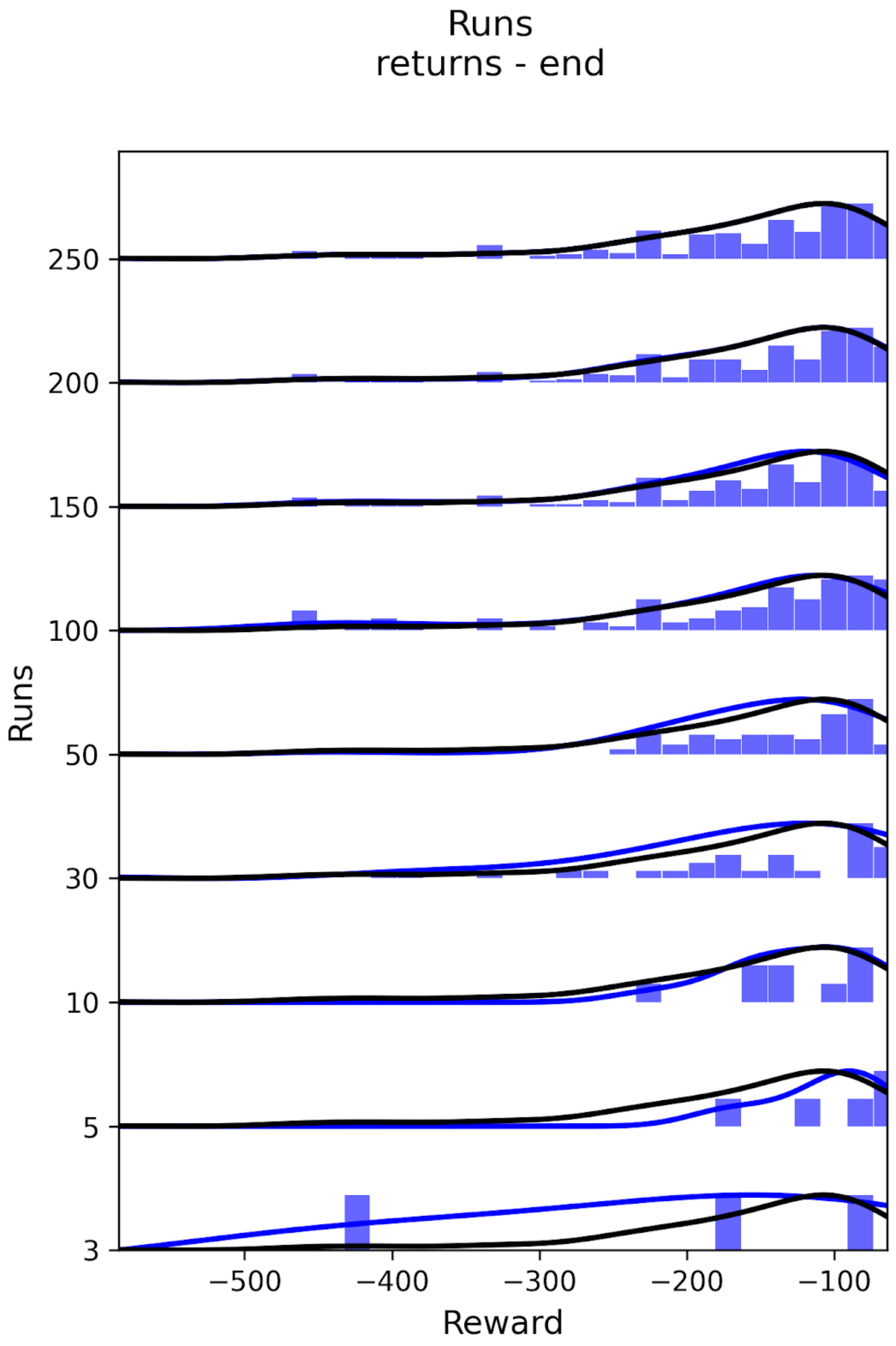


With and without cut-offs (median)

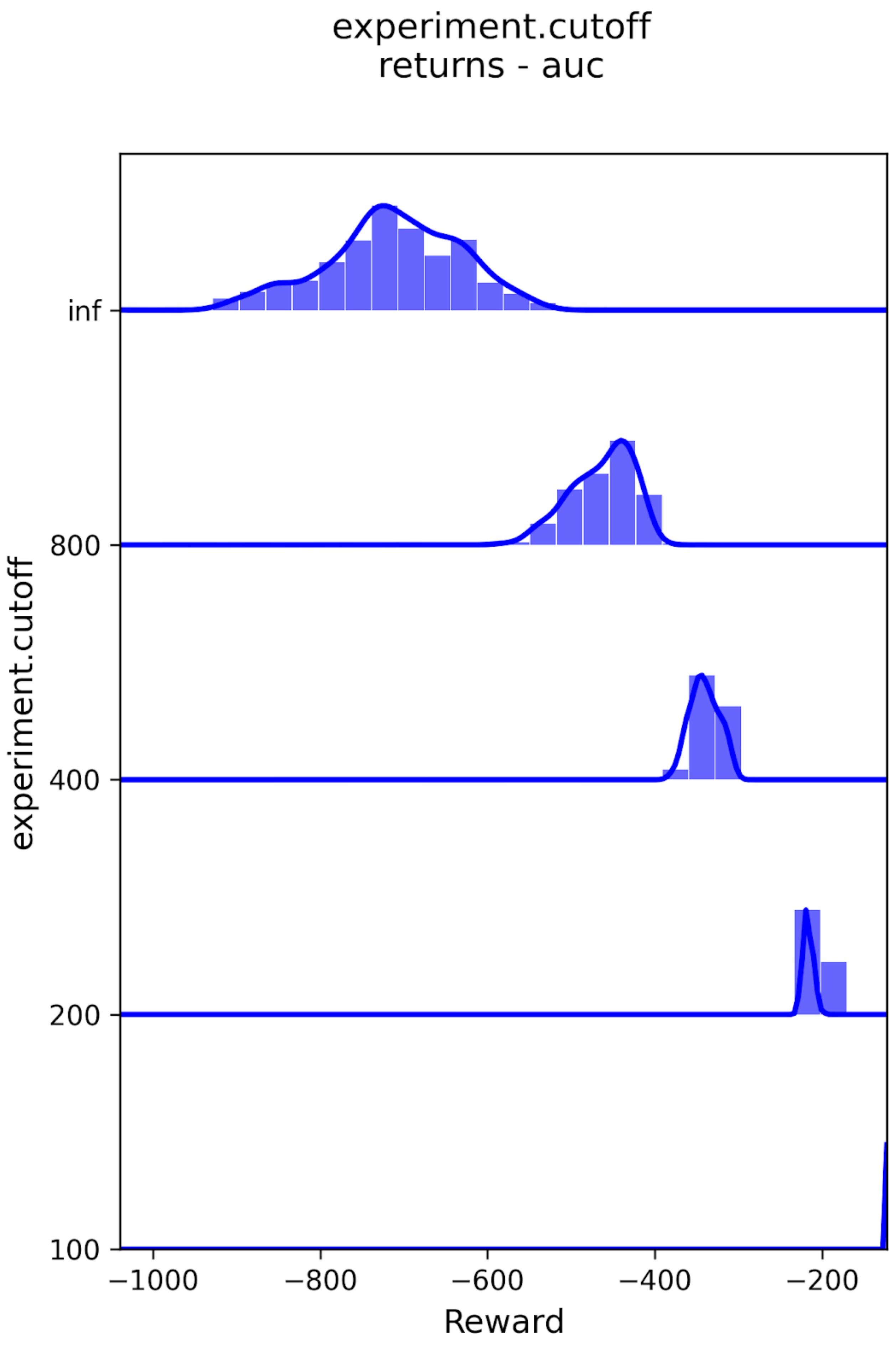


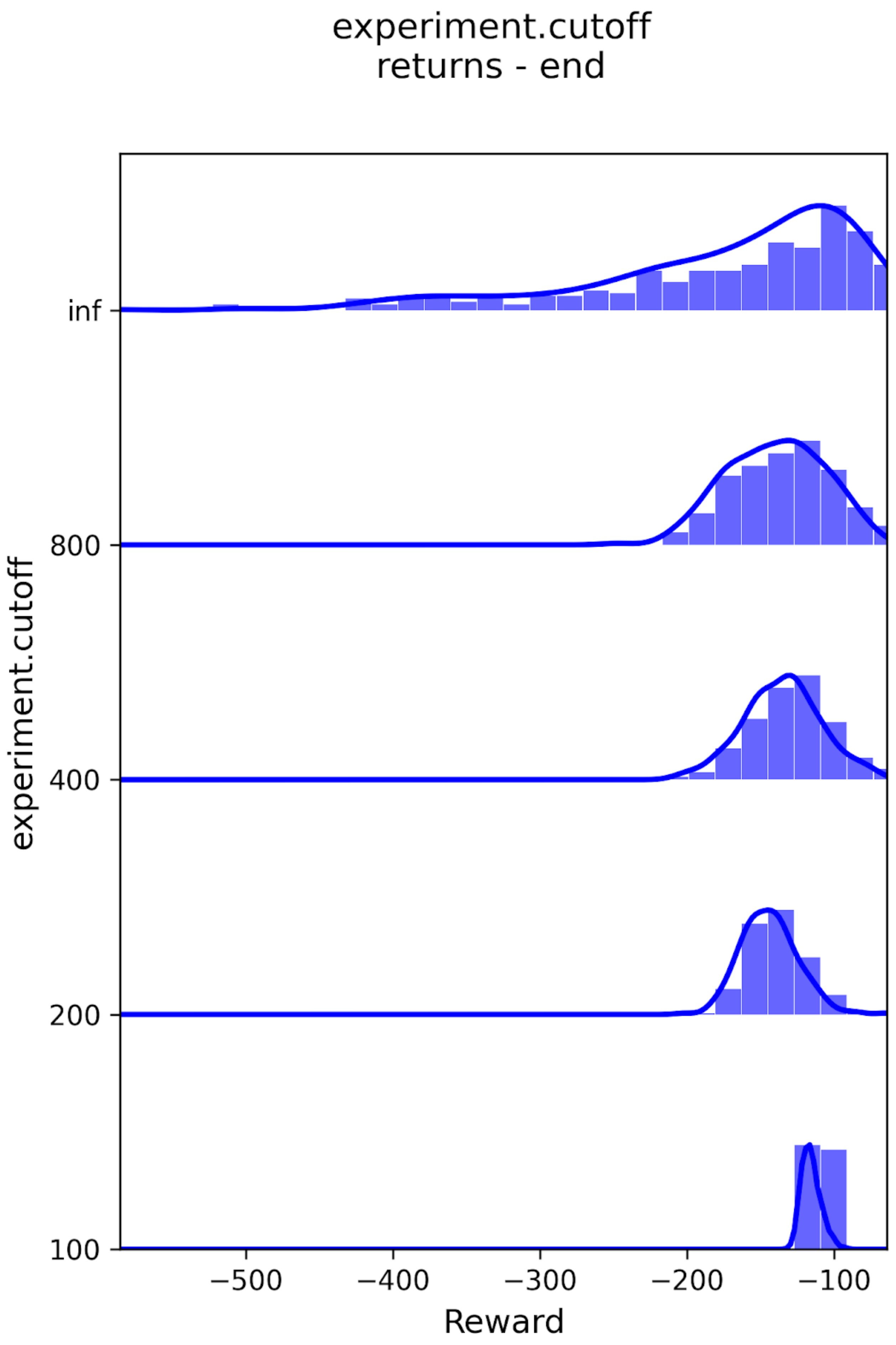
In puddle world we see impact of performance metric



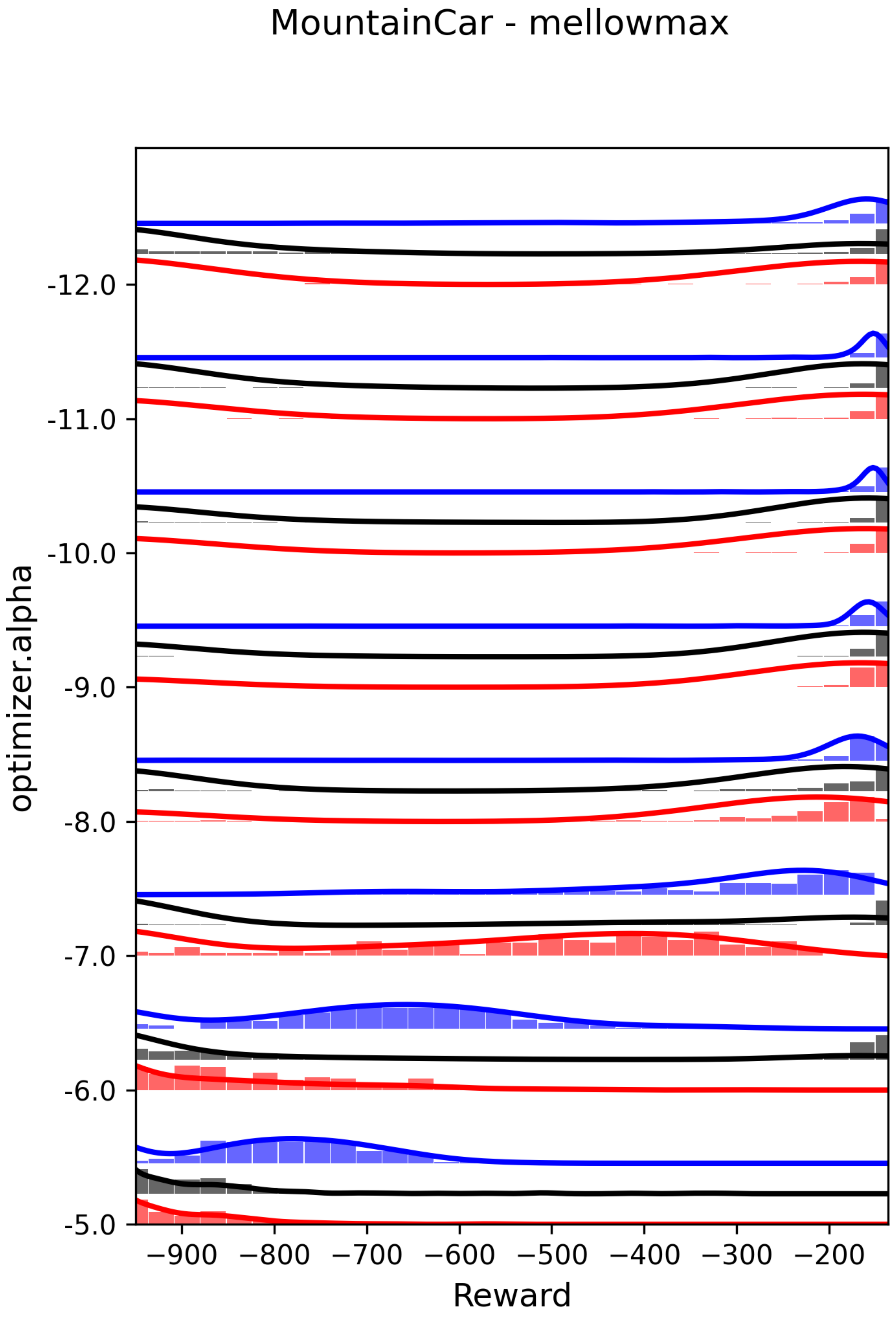


A closer look at cut-offs in puddle world





Bi-modality can even happen without explicit effort

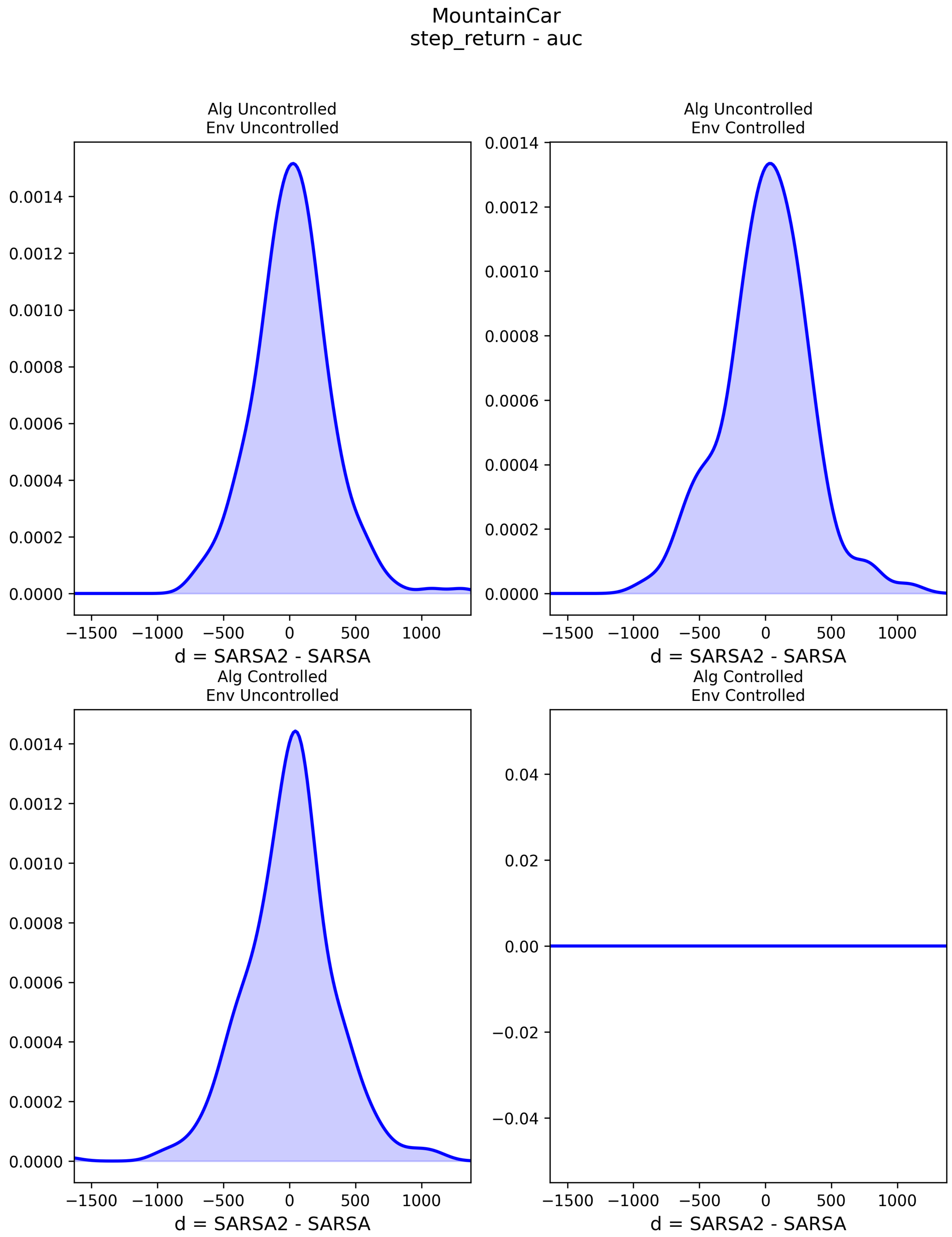


- Mountain car with 3 different algorithms and a Neural Network (2 layer, 32 hidden units, relu)
- Max episode length=1000, 100k steps total
- Agent hypers:
  - epsilon=0.1
  - Adam with beta_1 = 0.9 and beta_2=0.999
  - buffer_size = 4000, batch_size=32
  - No target nets

Controlling randomness

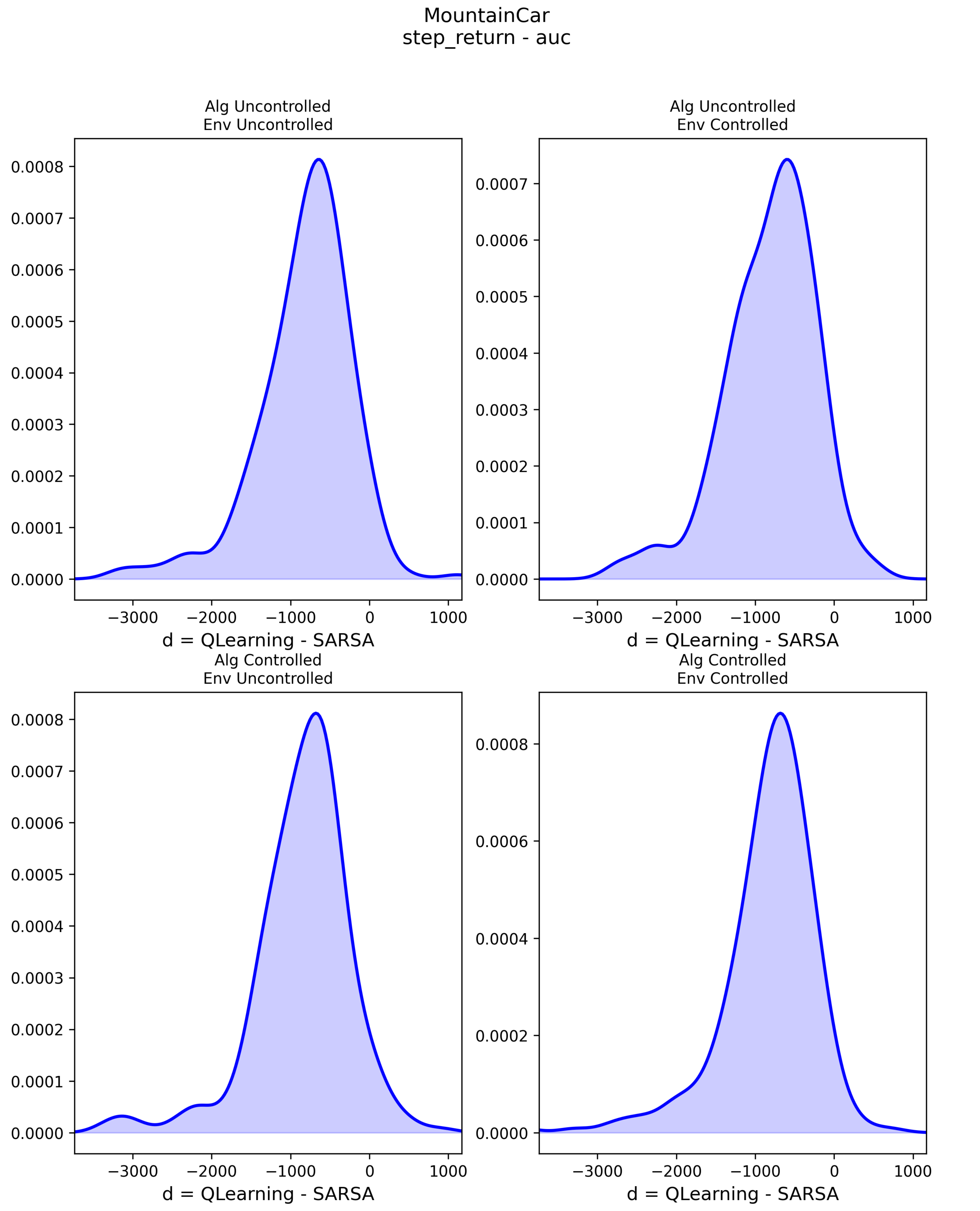
- Typically both the agent and environment have different sources of randomness:
    - In mountain car the start states, and epsilon in the agent for example
- We can decide to control these sources of randomness or not:
    - Controlled means the seed to the agent/env random number generator is set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

Controlling randomness: comparing the same algorithm (250 runs)



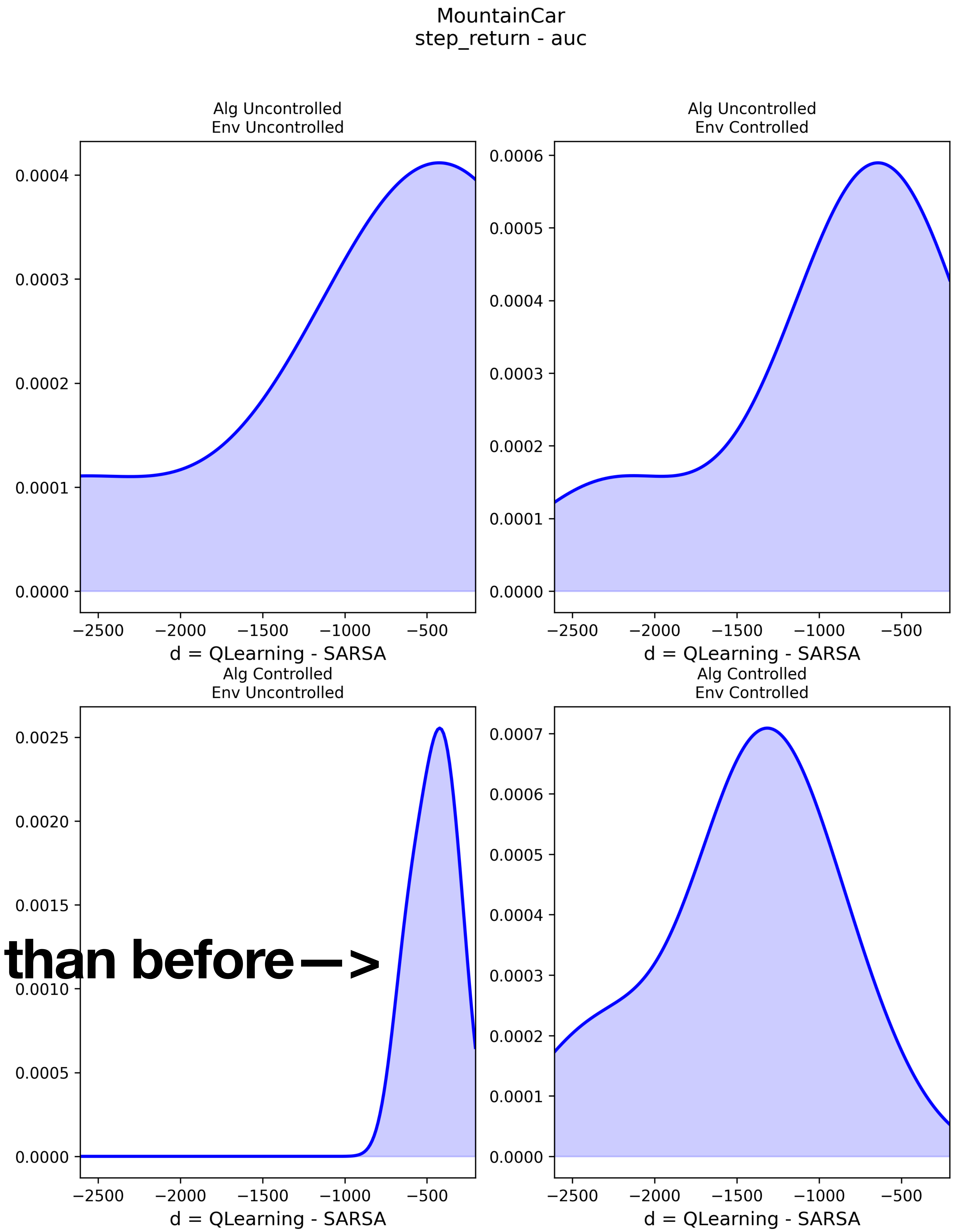
Controlling randomness: comparing Q-learning and Sarsa

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 population, 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 differently!

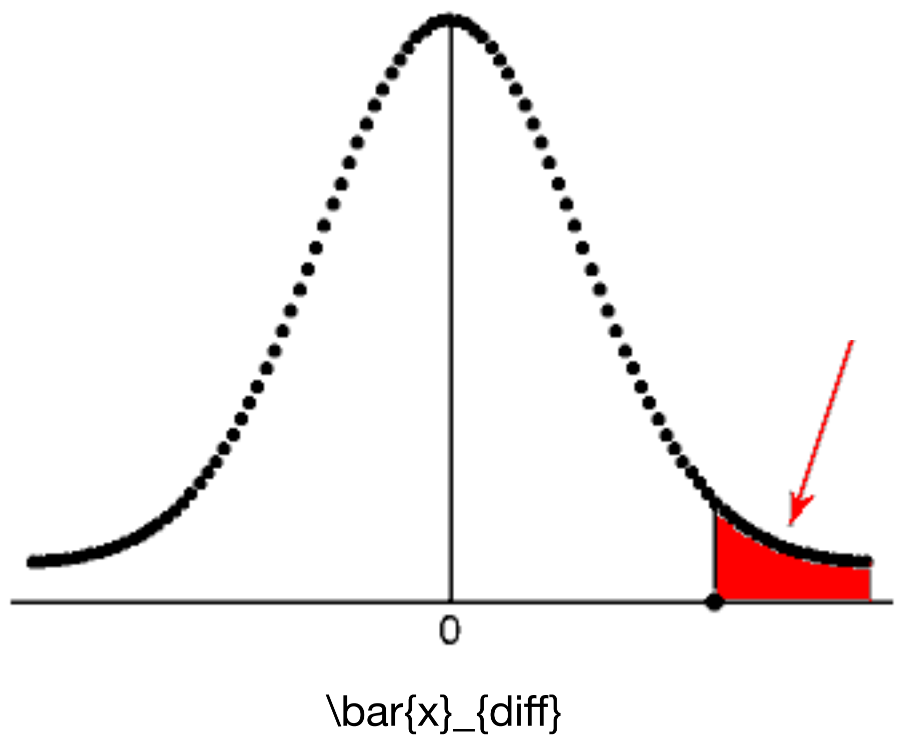
If your p-value is high, then your evidence (data) does not provide enough support to reject the null

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** $X_{diff,1}, X_{diff,2}$ 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~~

Break time

Then part II: giving presentations

Tips for giving research talks

Giving a good talk is hard!

- It is stressful; there are factors out of your control
- You are worried if your content is correct/accurate and if your style is clear and effective
- Everyone is different styles and preferences
- It is very easy to sit back and critique someone’s talk—much easier than giving a good one yourself
- **Worst of all: we have to perform**
- Good talks require a balance of: good content, dynamic delivery, clear & simple imagery, and lots empathy for the audience

High-level strategy: be simple and direct

Assume the audience will not follow a complex story

- **You are too close:** You know so much about the details over your work
- The audience barely knowns anything: about your specific project
- They are easily distracted ad easily confused
- Make the talk structure simple
- Make the messages simple and direct: don’t imply say what you mean
- Try to get across ONE (15min or less); TWO (20-30min); THREE (>30min) main messages

Talk structure

Tell them where things are going, again and again

- Start with a title slide:
  - Explain the title—like define the words
  - Retell the story of the research; perhaps how it all start or the main learning
  - Note collaborators (with pictures) and where you did the work

Talk structure

Motivate your work strongly

- Use the next few slides to define your problem more informally
- Focus on why the problem is interesting and hard
- Focus on why solving it matters (useful in another algorithm, real application…)
- Use pictures and diagrams to help people visualize the story
    - Less text in this part of the talk is better
- **Try to think of an overall theme or story to help keep attention**
- Strongly connect with prior work
- This is your introduction

Make an outline

Tell them where you are going again

- E.g.: “1. Why we need step-size adaption, and what has been done before; 2. Step-size adaption for online temporal difference learning, … “
- Avoid meaningless categories, like: “1. Motivation; 2. Related work; 3. Algorithm …”
- Helps the read know what is coming next and also functions as the main take-home messages of the talk
- I like to do it as:

  ☐ Empty boxes

  ☑ That get checked off as we go through the talk

Successes and lessons related specifically to:

- [ ] Hardware
- [ ] Data collection and control
- [ ] Learning setup
- [ ] Evaluating progress

EXAMPLE

Never define more than you need

There is a jargon budget and a notation budget

- Think of the minimal set of notations and equations your can use to convey the technical aspects of your work
- Same goes for terminology:
    - Do you need to define “online-aware” algorithms
    - Or can you always just describe what that means in simple plain English words
- This minimalist principle applies to figures, diagrams, results, algorithms and conclusions

Continually check in on your audience

You have already lost them

- Go back to your outline and recap what stage of the talk we are at
  - What are the conclusions so far
- Plan to ask the audience questions; perhaps with slides
- Have single slide messages (a slide with one line of text in large font)
- Never show them equations, diagrams or figures you plan to rush through or skip
  - “Anyway lets skip to the bottom of the derivation”
  - “Ignore all those other lines and subplots …”
- Take the time to view the work from the audiences perspective

Describing algorithms

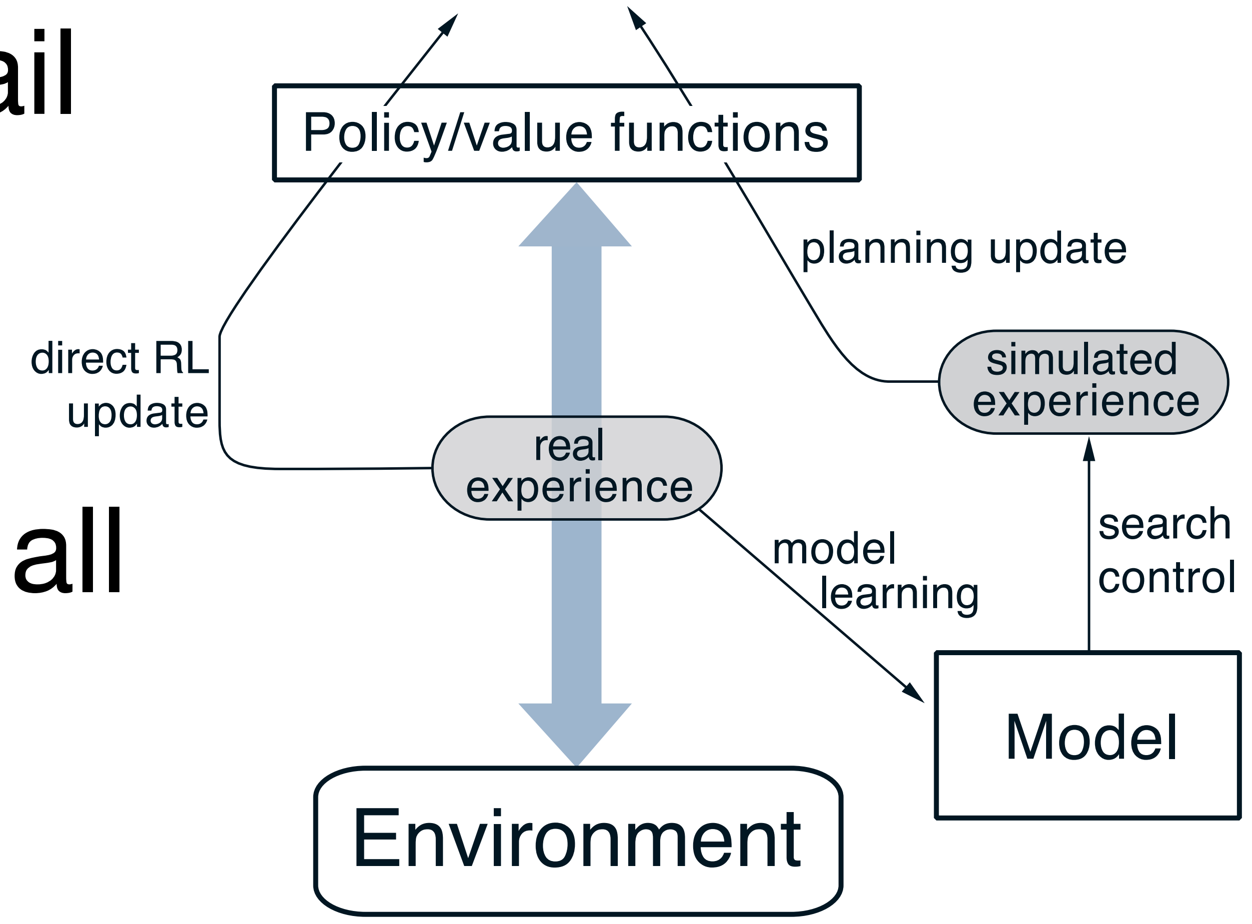
Think of the key details

Off-policy n-step Sarsa for estimating $Q \approx q_* $ or q_π

Input: an arbitrary behavior policy b such that $b(a|s) > 0$, for all $s \in \mathcal{S}, a \in \mathcal{A}$
Initialize $Q(s,a)$ arbitrarily, for all $s \in \mathcal{S}, a \in \mathcal{A}$
Initialize π to be greedy with respect to Q, or as a fixed given policy
Algorithm parameters: step size $\alpha \in (0,1]$, a positive integer n
All store and access operations (for S_t, A_t, and R_t) can take their index mod $n+1$

Loop for each episode:
    Initialize and store $S_0 \neq$ terminal
    Select and store an action $A_0 \sim b(\cdot|S_0)$
    $T \leftarrow \infty$
    Loop for $t = 0, 1, 2, \ldots$:
    | If $t < T$, then:
    | Take action A_t
    | Observe and store the next reward as R_{t+1} and the next state as S_{t+1}
    | If S_{t+1} is terminal, then:
    | $T \leftarrow t+1$
    | else:
    | Select and store an action $A_{t+1} \sim b(\cdot|S_{t+1})$
    | $\tau \leftarrow t - n + 1$ (τ is the time whose estimate is being updated)
    | If $\tau \geq 0$:
    | $\rho \leftarrow \prod_{i=\tau+1}^{\min(\tau+n, T-1)} \frac{\pi(A_i|S_i)}{b(A_i|S_i)}$ $(\rho_{\tau+1:\tau+n})$
    | $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n, T)} \gamma^{i-\tau-1} R_i$
    | If $\tau + n < T$, then: $G \leftarrow G + \gamma^n Q(S_{\tau+n}, A_{\tau+n})$ $(G_{\tau:\tau+n})$
    | $Q(S_\tau, A_\tau) \leftarrow Q(S_\tau, A_\tau) + \alpha\rho\,[G - Q(S_\tau, A_\tau)]$
    | If π is being learned, then ensure that $\pi(\cdot|S_\tau)$ is greedy wrt Q
    Until $\tau = T - 1$

- Code blocks are rarely useful
- Try to describe at a high level the main ideas of the algorithm
- Walk through the algorithm in increasing levels of detail:
    - Slide 1: three line English description of the algorithm
    - Slide 2: sub-bullets describing each component in more detail
    - Slide 3: perhaps introduce notation and equations
- Try to make a block diagram of your algorithm and refer to it at all three levels of the description



Describing algorithms

If you are gonna do it…

Off-policy n-step Sarsa for estimating $Q \approx q_*$ or q_π

Input: an arbitrary behavior policy b such that $b(a|s) > 0$, for all $s \in \mathcal{S}, a \in \mathcal{A}$
Initialize $Q(s, a)$ arbitrarily, for all $s \in \mathcal{S}, a \in \mathcal{A}$
Initialize π to be greedy with respect to Q, or as a fixed given policy
Algorithm parameters: step size $\alpha \in (0, 1]$, a positive integer n
All store and access operations (for S_t, A_t, and R_t) can take their index mod $n + 1$

Loop for each episode:
    Initialize and store $S_0 \neq$ terminal
    Select and store an action $A_0 \sim b(\cdot|S_0)$
    $T \leftarrow \infty$
    Loop for $t = 0, 1, 2, \ldots$:
    | If $t < T$, then:
    | Take action A_t
    | Observe and store the next reward as R_{t+1} and the next state as S_{t+1}
    | If S_{t+1} is terminal, then:
    | $T \leftarrow t + 1$
    | else:
    | Select and store an action $A_{t+1} \sim b(\cdot|S_{t+1})$
    | $\tau \leftarrow t - n + 1$ (τ is the time whose estimate is being updated)
    | If $\tau \geq 0$:
    | $\rho \leftarrow \prod_{i=\tau+1}^{\min(\tau+n, T-1)} \frac{\pi(A_i|S_i)}{b(A_i|S_i)}$ $(\rho_{\tau+1:\tau+n})$
    | $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n, T)} \gamma^{i-\tau-1} R_i$
    | If $\tau + n < T$, then: $G \leftarrow G + \gamma^n Q(S_{\tau+n}, A_{\tau+n})$ $(G_{\tau:\tau+n})$
    | $Q(S_\tau, A_\tau) \leftarrow Q(S_\tau, A_\tau) + \alpha\rho\,[G - Q(S_\tau, A_\tau)]$
    | If π is being learned, then ensure that $\pi(\cdot|S_\tau)$ is greedy wrt Q
    Until $\tau = T - 1$

- Cover up parts of the code and uncover them as you go
- Use color, highlighting and blocks to emphasize particular details
- Walk them through the algorithm; take the agent’s point of view
- Tell them why it is import to understand the algorithm at this level of detail

Question yourself, for other

Think about what might be confusing to the audience

- As you go through your slides
- As you practice your presentation
- Write down questions an outsider might wonder about:
  - “Why couldn’t we just use RMSProp here?”
  - “I don’t see why the Hessian would not be invertible?”
- Raise these questions in your talk and answer them:
  - “You might be wondering …”
- **Related**: always arrange your slides and bullets to answer: “what would they want to know next?”

Presenting empirical results

Take it slow

- First discuss the overall objective of the experiments:
  - “We ran three experiments to investigate the our new ….”
- Do everything one at a time:
  - One experiment at a time
  - First the problem /environment: described without reference to the agent
  - Then the solution methods
  - Then the way the experiment was set up & run and how the results we processed
  - One thing per slide. For example:

Empirically evaluating AdaGain

- We conducted experiments in two domains:
  - A state-less tracking problem
  - A multi-step prediction problem using real robot data
- The objective of these experiments were (one for each problem):
  - To investigate how AdaGain adapts in a continual learning setting compared with existing approaches
  - To evaluate how AdaGain performs with high-dimensional, non-stationary data

State-less tracking problem

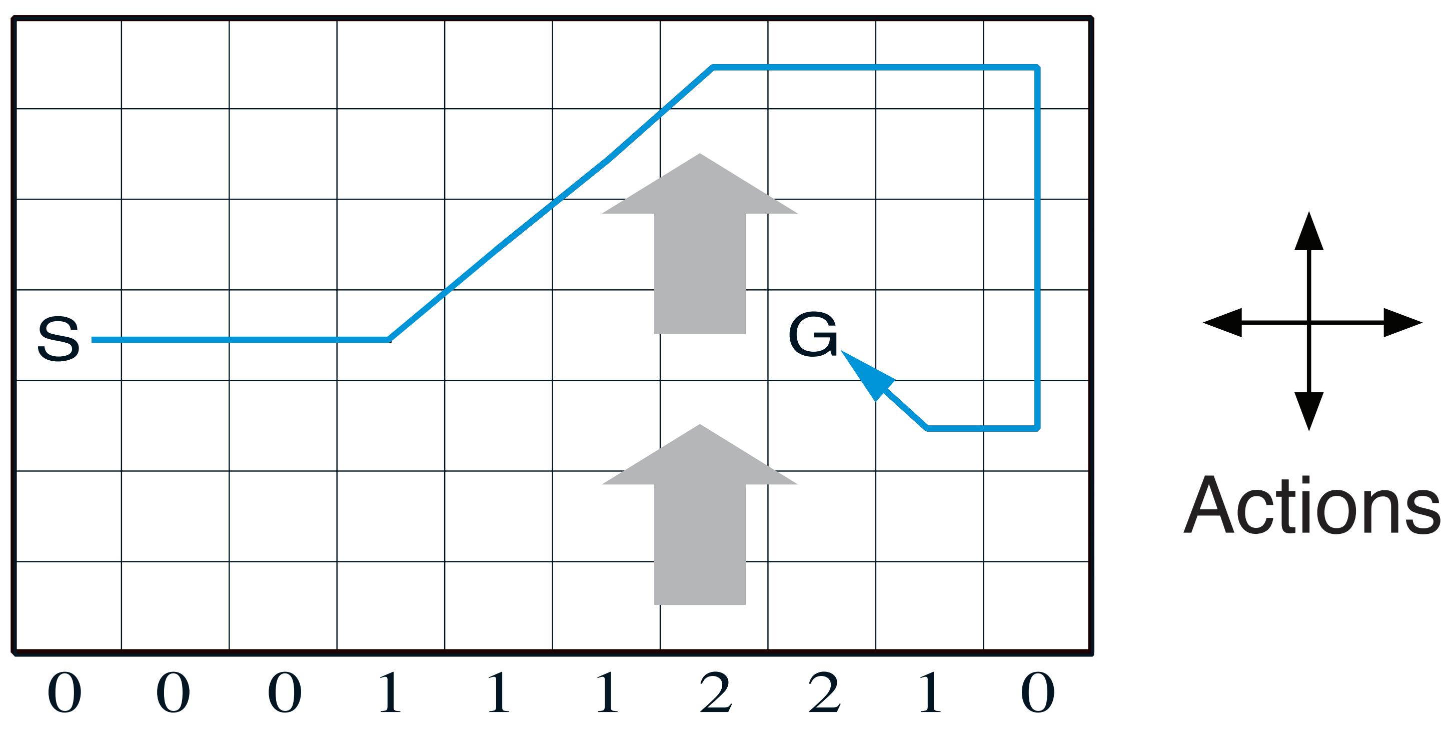
- In this problem the objective is to track the expected value of the target signal, where the underlying process follows a random walk
- *Define the problem precisely …*

Algorithms compared

- We compared AdaGain with a mixture of well-know methods from deep learning, as well as several older methods from the meta-descent literature
- In particular we compared: AdaGain, RMSProp, Adam, SMD, AdaDelta, IDBD, …
- *Explain any particular details of interest*
- *Highlight key hyper-parameters or implementation details*
- *This might take multiple slides*

Experiment #1: state-less tracking

- We ran each step-size method for 1billion time-steps
- We computed the average reward per step, and average the results over 50 independent runs
- We set the hyper-parameters of each method with an extensive sweep:
    - *Describe parameters, ranges, and selection criteria*
- *Use pictures*

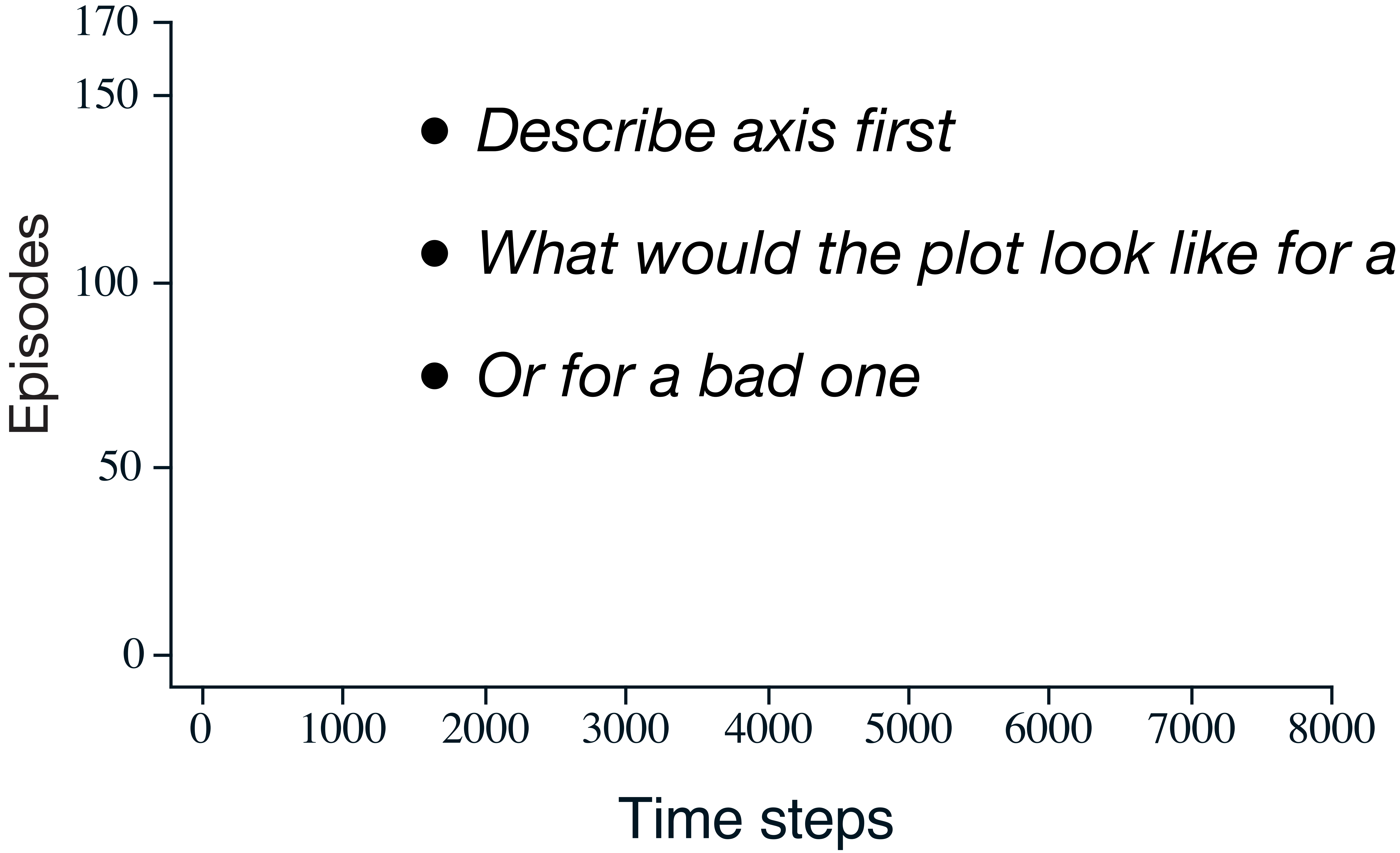


Experiment #1: results

- One plot at a time
- Start with a simple plot that is easy to describe
- Use annotations, colours, arrows, and animations to control the flow of information and keep everything manageable
- Tell the audience simple things like “up on this plot is good”
- Put the main message of the result at the bottom of the slide…build it in last

Experiment #1: results (only one plot per slide)

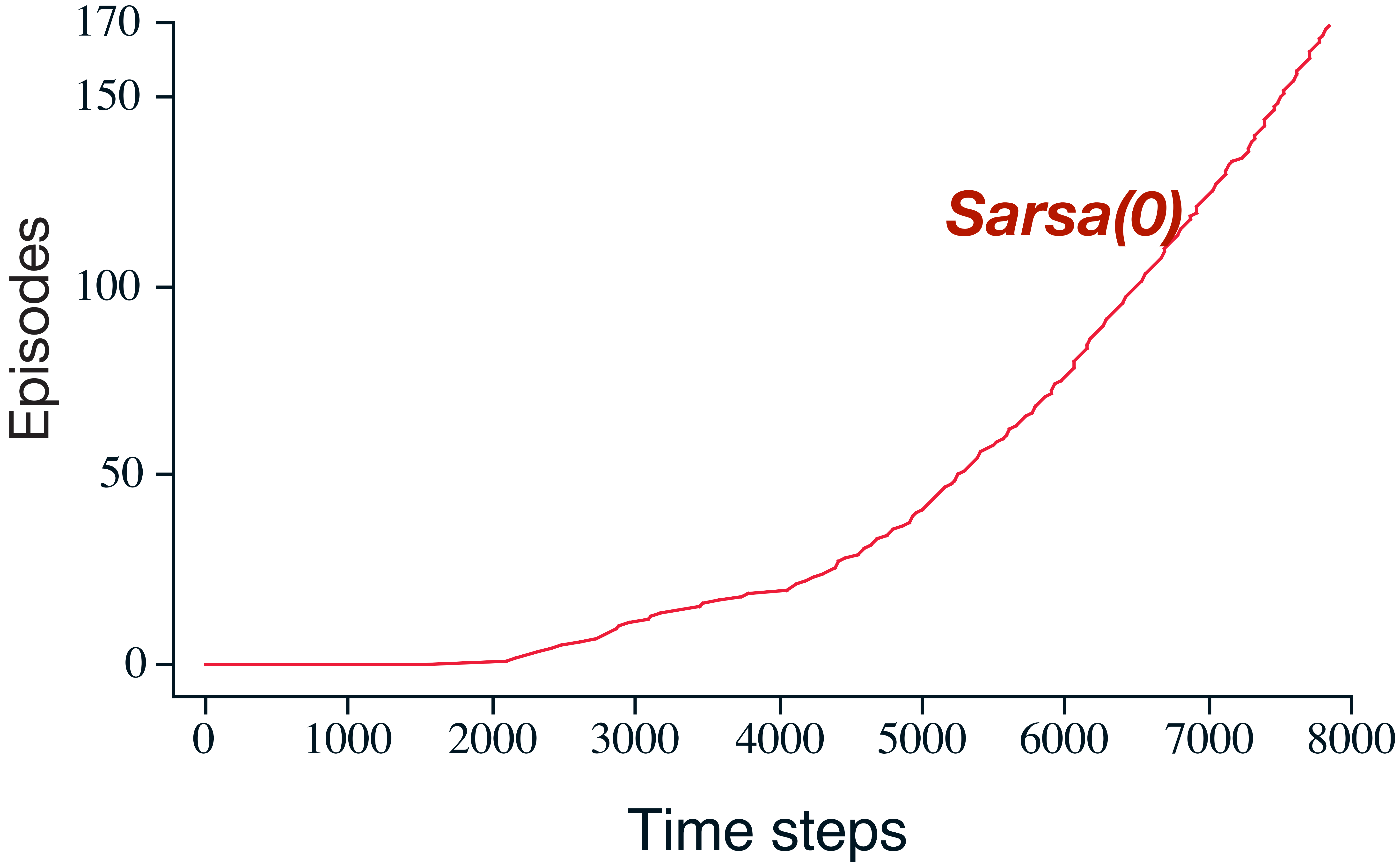
- *Describe axis first*
- *What would the plot look like for a good algorithm?*
- *Or for a bad one*



Experiment #1: results (only one plot per slide)

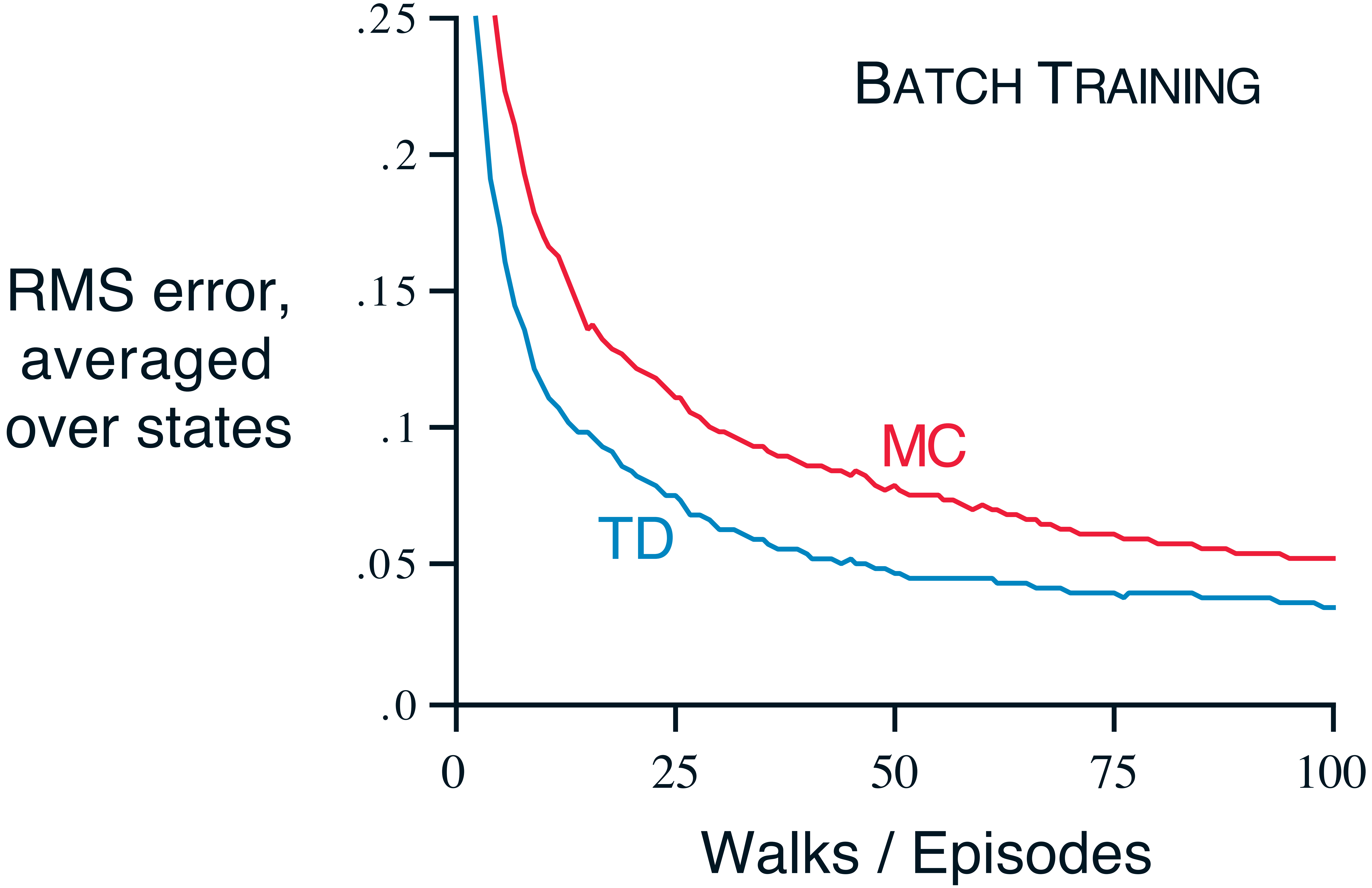
- *Define error bars if included*

Big text



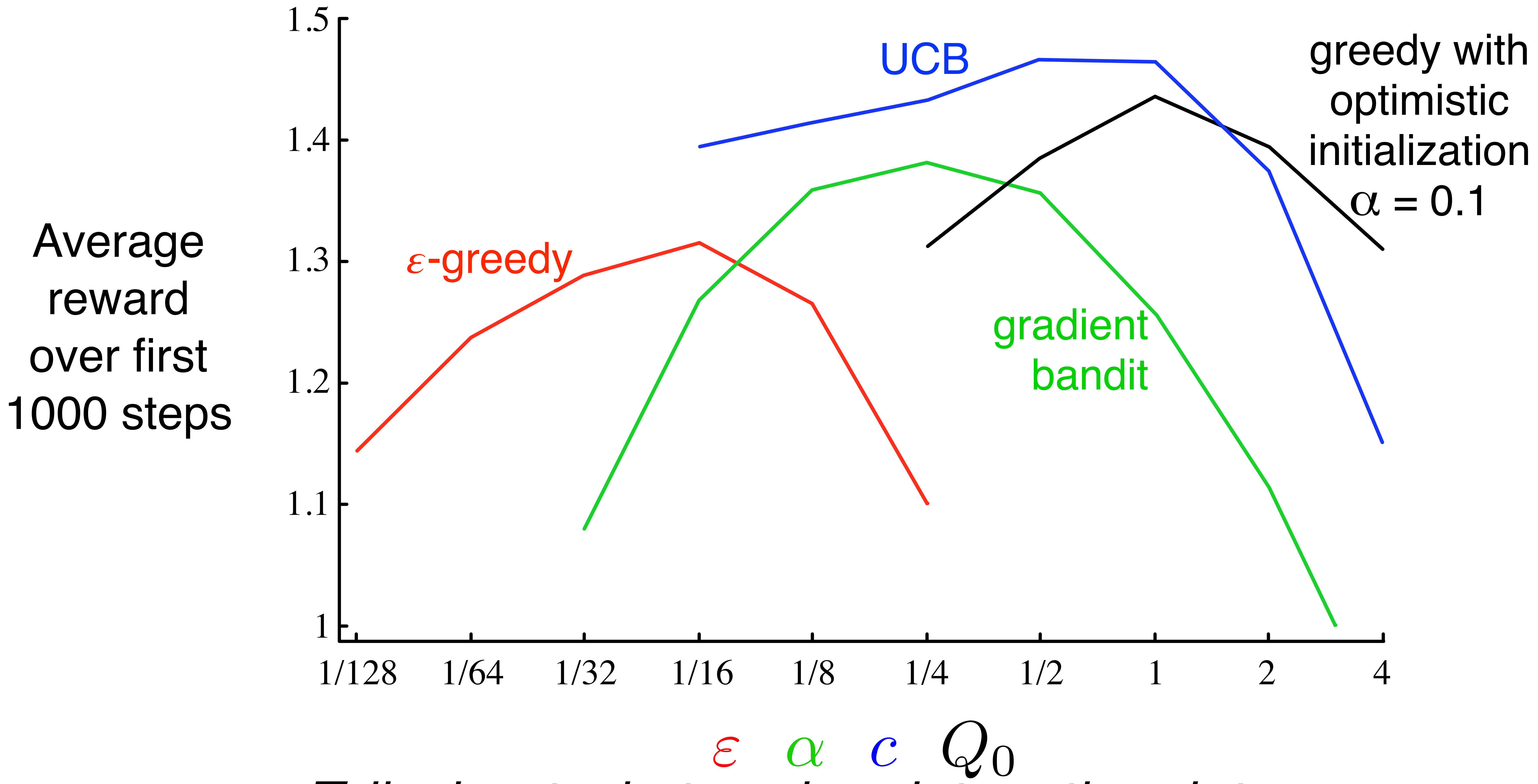
- *Sarsa complete episodes at a faster and faster rate*

Experiment #1: results (now comparing algs)



- *TD learns faster and reaches lower error…*

Experiment #1: results (progress to the complex)



- *Talk about what each point on the plot means, and the shape*

Summarize the results and make conclusion

- DON’T OVER CLAIM
- Tell the high-level take home messages
  - Ref specific results as needed
- Talk honestly about limitations and negative results
- We want to tell the story of what happened, **not sell** someone something
  - Presumably you are giving the talk about something with interesting results not “I invented TD++ and its always worse than TD”—bad topic

Wrap it up and look to the future

- Revisit your outline and take home messages
  - Tell the audience we got though everything I wanted to discuss
  - This is the way I think you should think about it
- Finish with Future-work / limitations
  - All good work is limited—you had to narrow the scope to make progress
  - Future work is often about lessening some of those limitations

Low-level advice

- Be prepared for lots of questions: everyone will think and do differently
    - That doesn’t matter, but its essential you can explain the “why” of all your choices
- One idea per slide
- Slide titles should be thought of as conclusions or topic sentences
- **SLOW DOWN but be excited**
- Use running examples “imagine you are driving home in the rain”; keep going back to the example
- Use **bold**, *italics*, and color whenever you want
- There are lots of rules: length of slide titles, bullet punctuation & grammar, slide #’s
    - **Low priority compared to constructing a simple and engaging talk**

The main goal is getting everyone to understand what you did and why

The secondary goal is making them believe its a significant contribution

Project standup

- 30second to 5 minute summary of your project
- Thing you are most focused on now
- Open question for the group:
  - Anything you are currently stuck on?