Assignment 1 — Matching Pennies: WSLS vs k-ToM-inspired belief learning

AUTHOR
Ramona

The Assignment Concept

This assignment asks for two things: (i) describe two plausible strategies for repeated Matching Pennies, and (ii) provide a simulation + visualization pipeline in a version-controlled repository.

The goal is not only "who wins". The goal is to connect behaviour to mechanistic claims under cognitive constraints (limited memory, perseveration, errors/noise), and to show those mechanisms are identifiable via model fitting.

Game and protocol

- Repeated Matching Pennies for **T = 60** trials.
- Each trial, one player is the **Matcher** (wants actions to match) and the other is the **Mismatcher** (wants actions to differ).
- After **30 trials** roles swap.

Why the role swap matters cognitively:

A final pay-off collapses behaviour into one number. The mid-game role swap changes the reward mapping (match vs mismatch). This tests whether a strategy represents task contingencies and re-maps behaviour when contingencies flip. That is a cognitive signature, not just "performance."

A useful reference point: in the ideal mixed-strategy equilibrium of Matching Pennies, expected pay-off is 0. Persistent deviations from 0 indicate exploitability under cognitive constraints (finite memory, noise, imperfect opponent modelling).

Strategy 1: WSLS (Win–Stay / Lose–Shift)

WSLS is a minimal, cognitively cheap strategy:

- After a win: repeat previous action (stay)
- After a loss: switch action (shift)

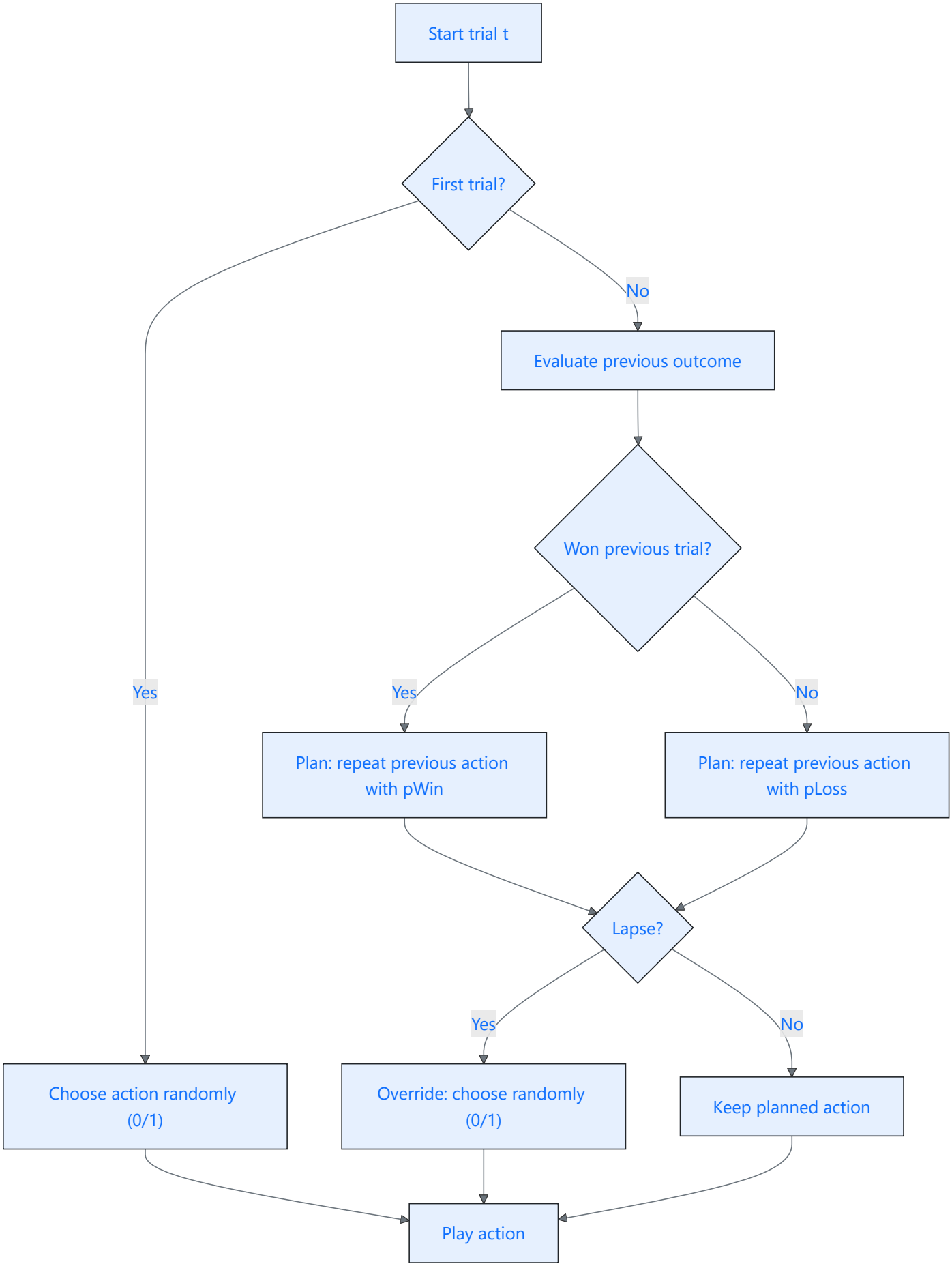
We implement WSLS probabilistically to encode cognitive constraints:

- `p_repeat_win`: how strongly the agent repeats after a win (perseveration / "win-stay")
- `p_repeat_loss`: how strongly the agent repeats after a loss (failure to shift)
- `lapse`: random responding (errors / distraction)

Cognitive constraints (operational):

- Memory requirement: last action (1 bit) + last outcome (1 bit).
- Computation: one conditional probability lookup ("win?") and a possible flip.
- Errors: implemented as `lapse`, mixing toward random responding.

Formalisation (diagram):



Strategy 2: k-ToM-inspired belief learning (interpretable)

In the course notes, Theory of Mind models are about explicitly modelling what the opponent will do.

To keep the model simple and interpretable (Assignment 1 scope), we use a k-ToM-inspired belief-learning strategy:

- Maintain a belief (b_t = P(1))
- Update belief with a delta rule (learning rate ()):
  - (b_{t+1} = b_t + , (_t - b_t))
- Choose a role-dependent best response:
  - Matcher: match the likely opponent action
  - Mismatcher: mismatch it
- Add cognitive noise via () (decision sharpness) and `lapse` (random errors)

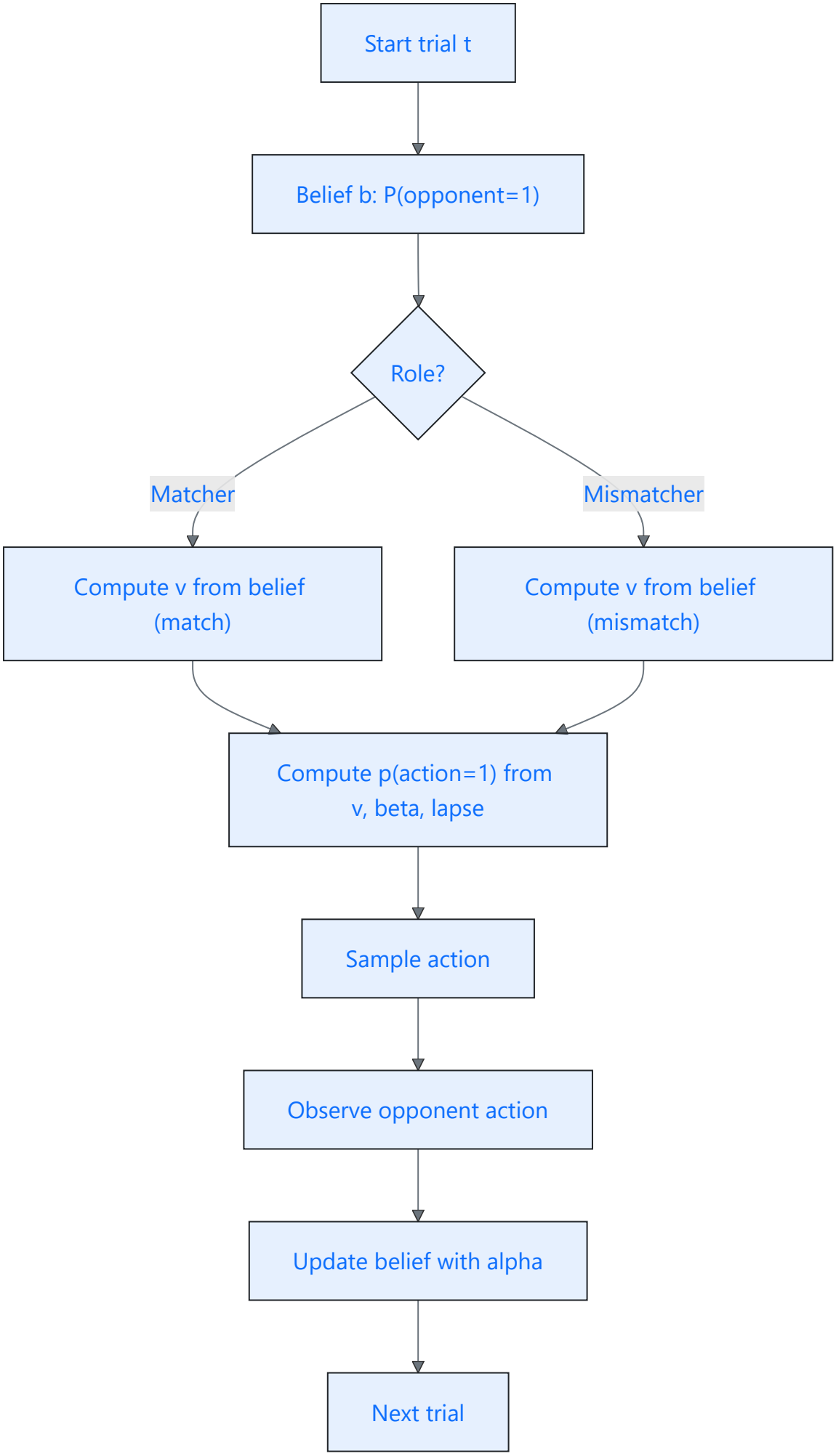
Parameters map cleanly to constraints:

- `alpha`: recency weighting / memory limitation (how quickly beliefs forget older evidence)
- `beta`: graded vs deterministic choice (noise / stochasticity)
- `lapse`: occasional random choices (errors / distraction)

Cognitive constraints (operational):

- Memory requirement: one scalar belief state (b_t) plus knowledge of current role.
- Computation: update a single belief and compute a best response given role.
- Errors: implemented via () and `lapse`.

Formalisation (diagram):



5W1H: Where Stan is used (and why it is not bolted-on)

Who uses Stan?

- The modeller uses Stan to make assumptions explicit and to connect simulated behavior back to interpretable parameters.

What is Stan used for?

- 1. Sampling heterogeneous agent parameters from explicit priors (generative story).
- 2. Fitting hierarchical models to action sequences to infer cognitive parameters (interpretation).
- 3. Comparing competing cognitive hypotheses with out-of-sample predictive fit (LOO).

When is Stan used?

- Before simulation: to draw agent parameters from priors (fixed_param).
- After simulation: to fit models back to the generated sequences (MCMC), then compute LOO.

Where is Stan in the pipeline?

- `assignment1/stan/` contains the generative prior models and the two hierarchical fit models.

Why is this necessary?

- Because strategy descriptions are not enough: we also need to show that the mechanisms are (i) identifiable from finite noisy data and (ii) predictively distinct.
- Parameter recovery (simulate → fit → recover true values) is a direct check that "the parameters mean what we claim they mean."
- LOO makes the two strategies explicit competing generative hypotheses, not just a tournament.

Priors used (summary)

Component	Parameter	Prior (generative)	Prior (fit, population-level)
WSLS	p_repeat_win	Beta(8, 2)	mu_win ~ Normal(0, 1.5) on logit scale
WSLS	p_repeat_loss	Beta(2, 8)	mu_loss ~ Normal(0, 1.5) on logit scale
WSLS	lapse	Beta(2, 10)	mu_lapse ~ Normal(-2, 1.5) on logit scale
Belief	alpha	Beta(2, 2)	mu_alpha ~ Normal(0, 1.5) on logit scale
Belief	beta	LogNormal(0, 0.5)	mu_log_beta ~ Normal(0, 1.5) on log scale
Belief	lapse	Beta(2, 10)	mu_lapse ~ Normal(-2, 1.5) on logit scale

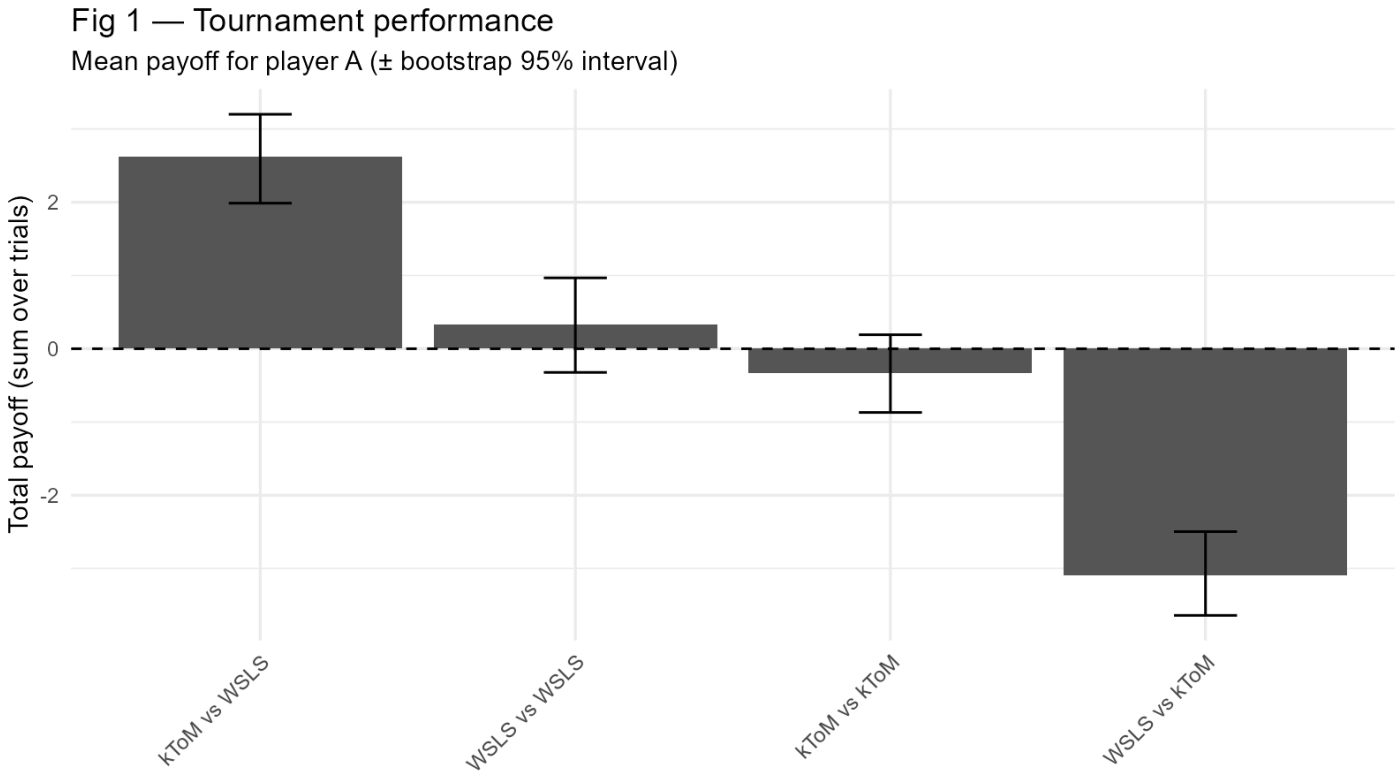
Results

Results logic:

We separate three questions so each figure has a clear purpose.

1. Performance: who tends to exploit whom (with uncertainty)?
2. Role sensitivity: what happens when contingencies flip at trial 30?
3. Mechanism: do fitted parameters recover the intended cognitive signatures, and do the models predict differently out of sample?

Fig 1 — Tournament performance (mean payoff + uncertainty)



Interpretation:

- This gives a coarse behavioral summary: expected payoff (with bootstrap uncertainty) for each matchup.
- Because Matching Pennies has a mixed-strategy equilibrium with expected payoff 0, systematic deviations from 0 indicate exploitability under constraints (finite memory, noise, imperfect modelling).

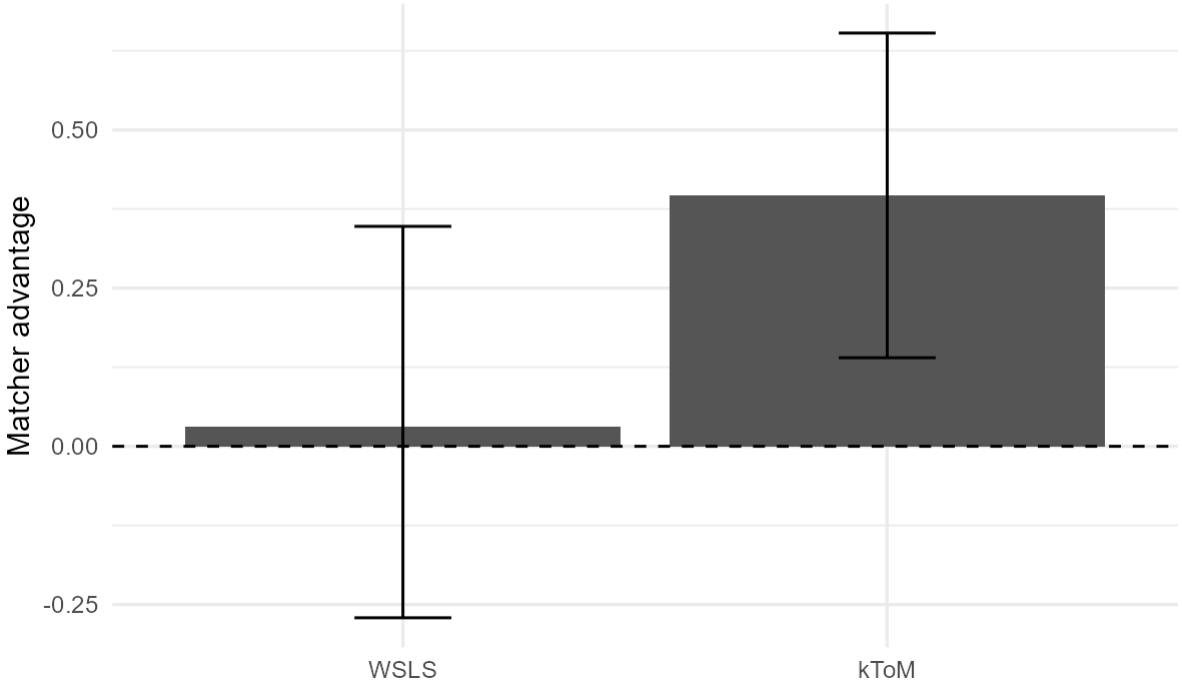
Expected pattern:

- If belief learning exploits WSLS regularities, belief-vs-WSLS should deviate from 0 in belief's favor (within uncertainty).
- Self-play near 0 is consistent with symmetry and limited exploitability (given noise/lapse).

Fig 2 — Role swap diagnostic (contingency remapping)

Fig 2 — Role sensitivity (Matcher advantage)

Payoff as Matcher minus payoff as Mismatcher (± bootstrap 95% CI)



Interpretation:

- The swap at trial 30 changes the utility mapping (match vs mismatch). This tests contingency re-mapping: does the policy adapt when "what counts as success" flips?
- WSLS is outcome-reactive: it conditions on win/loss, not on role explicitly. After the swap, WSLS can look superficially stable because it continues to follow "repeat after win, switch after loss," even though the meaning of "win" changed.
- The belief learner explicitly uses the role in its decision rule (best response depends on Matcher vs Mismatcher). A systematic role advantage (or a sharper post-swap shift) therefore supports the claim that it represents task contingencies, not only recent outcomes.
- If the plotted role advantage is near 0 with narrow intervals, the conclusion is not "no effect," but "roles are handled symmetrically / adaptation is fast under these parameters and noise."

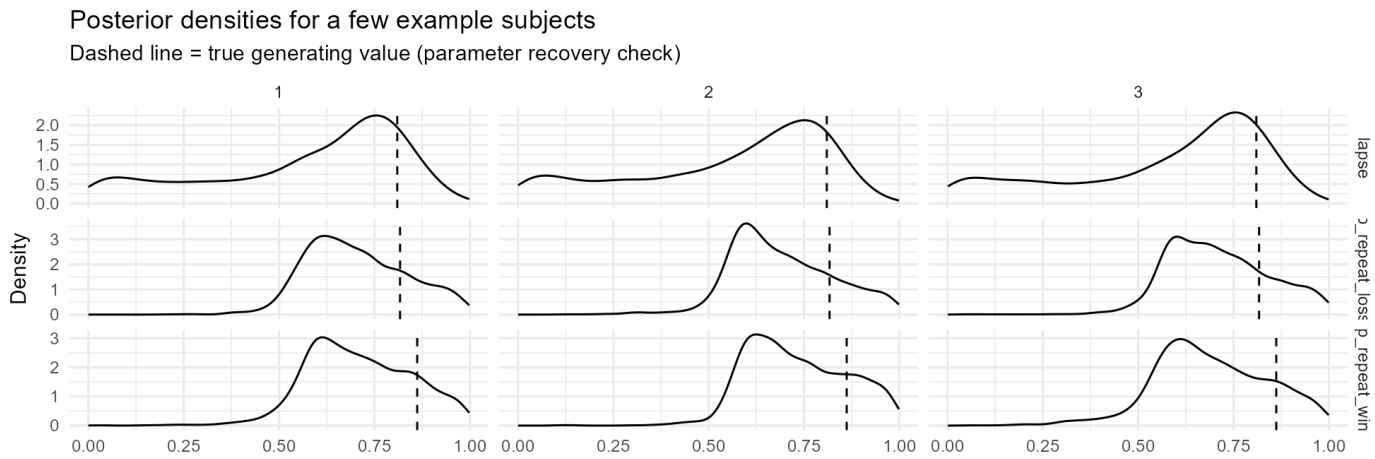
Optional diagnostic note:

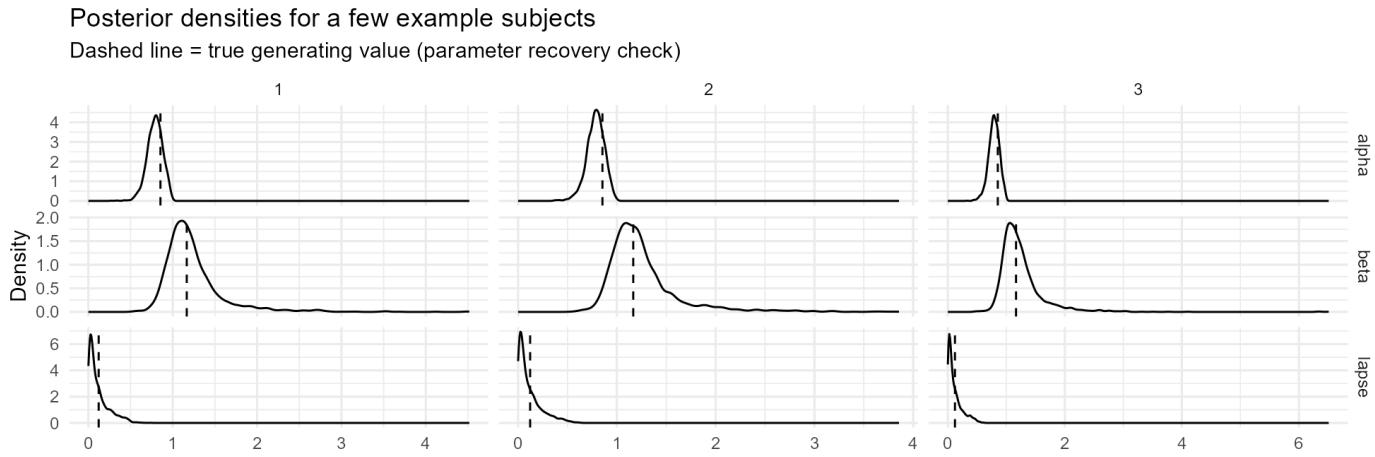
- Adaptation lag could be quantified by restricting the analysis to a small window right after trial 30 (e.g., trials 31–35). The current summary already captures the direction and uncertainty of remapping.

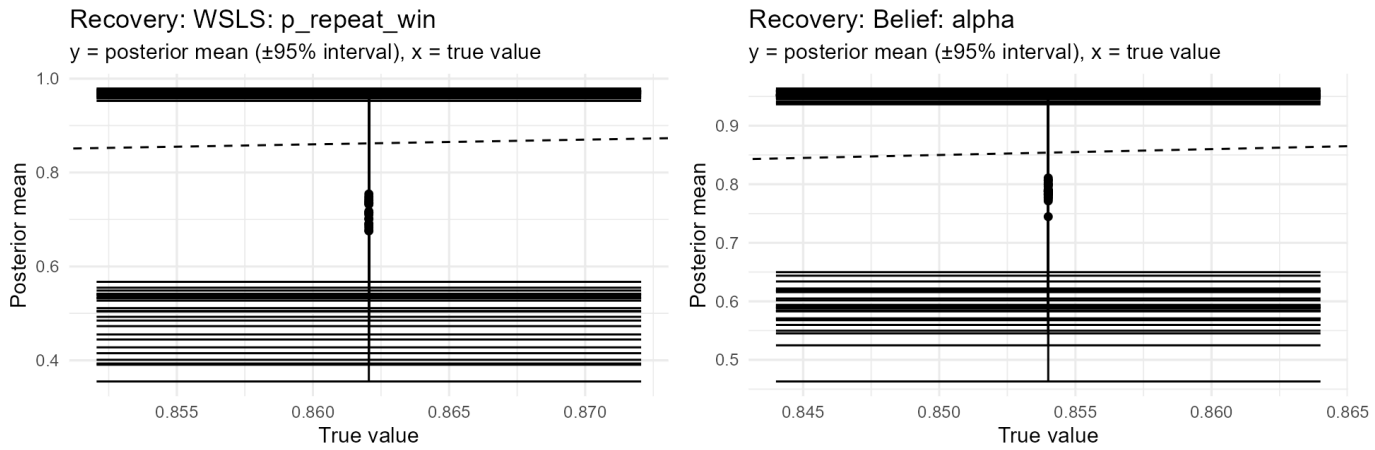
Fig 3 — Main interpretation figure (Chapter-5 style): posterior + recovery

This is the key mechanism figure.

- Top panels: posterior densities for a few example subjects with the true generating value (dashed line).
- Bottom panels: recovery scatter (true value vs posterior mean ± 95% interval).







Interpretation:

- Identifiability: posteriors concentrate around the true generating values for key parameters, implying the mechanisms are recoverable from sequences of length T=60 (given these priors).
- Cognitive signatures:
  - WSLS: `p_repeat_win` vs `p_repeat_loss` expresses reinforcement-like "stay/shift" asymmetry; `lapse` captures errors.
  - Belief learning: `alpha` expresses memory/recency weighting; `beta` expresses graded choice; `lapse` captures errors.

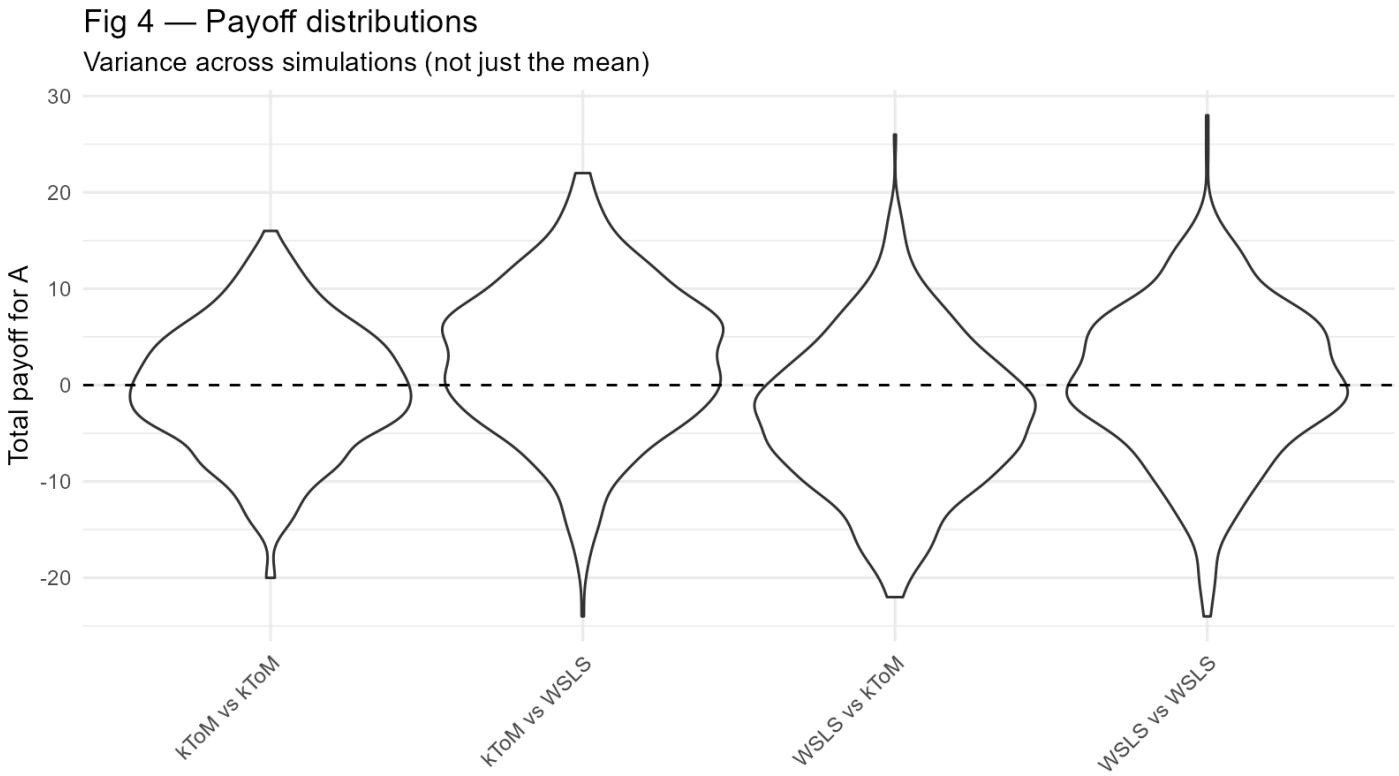
What would count as a failure:

- Broad posteriors that do not concentrate near the true values (or intervals that systematically miss) would suggest the parameters are not identified at this trial length, or that priors dominate the likelihood.

Note on recovery plots:

- Recovery scatter is most diagnostic when true values span a wide range. If true values are clustered (by design or by narrow priors), posterior concentration in the density panels carries more interpretive weight than the slope of the recovery scatter.

Fig 4 — Payoff distributions (variance / robustness)



Interpretation:

- Mean pay-offs can hide instability. Distributions show whether a strategy's advantage is robust or driven by a few extreme simulations.

LOO model comparison (table + dot)

We fit both hierarchical models to both datasets and compare out-of-sample predictive fit (LOO).

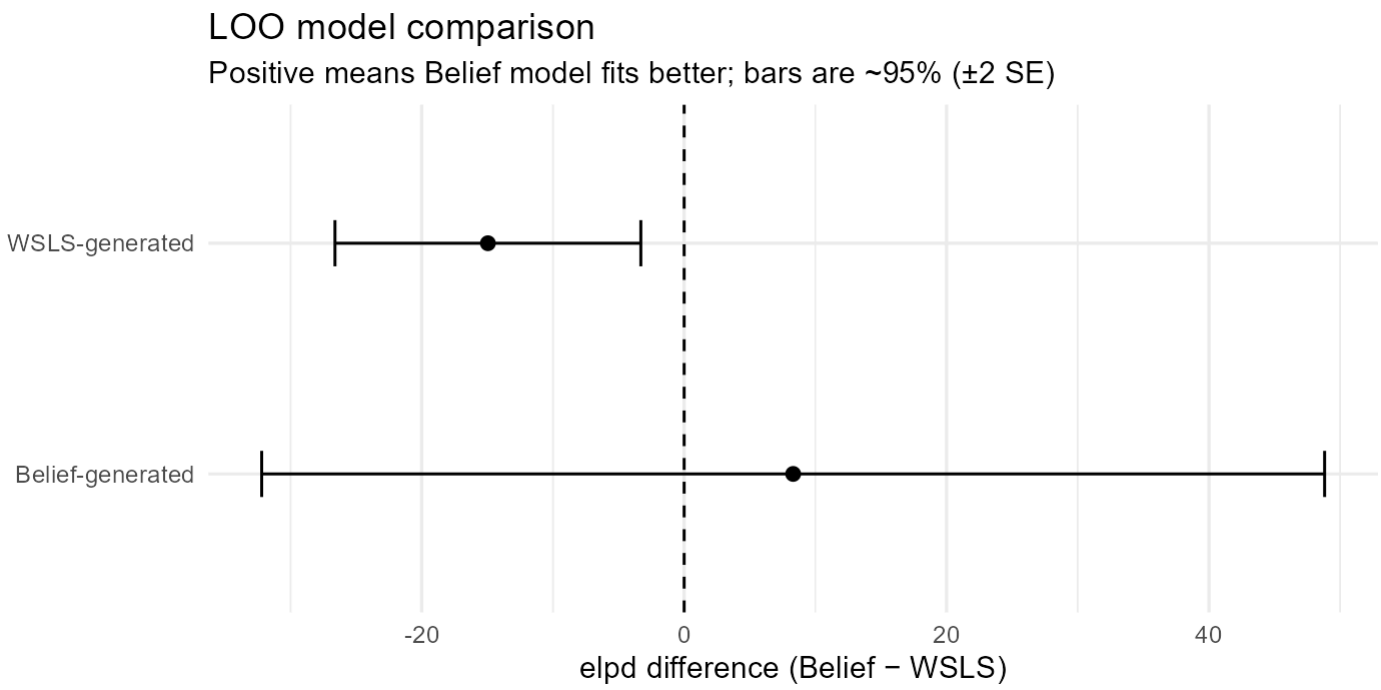
Interpretation:

- Reported quantities are differences in expected log predictive density (elpd) with a standard error (SE).
- If an elpd difference is small relative to its SE, evidence for a better model is weak at this sample size / noise level.

Decision rule (heuristic):

- Evidence is "clear" when |elpd_diff| is meaningfully larger than its SE (a common practical heuristic is about 2×SE); otherwise the comparison is ambiguous at this T and noise level.

```
# A tibble: 4 × 10
  elpd_diff se_diff elpd_loo se_elpd_loo p_loo se_p_loo looic se_looic model
      <dbl>   <dbl>    <dbl>       <dbl> <dbl>    <dbl> <dbl>    <dbl> <chr>
1      0       0      -1025.       5.77  10.8    0.112  2051.    11.5  model1
2    -15.0     5.78   -1040.       0.818  1.31   0.0338 2081.     1.64 model2
3      0       0       -926.      14.4    6.53   0.150  1853.    28.9  model2
4     -8.31    4.86    -935.      14.2    9.30   0.205  1869.    28.4  model1
# ℹ 1 more variable: dataset <chr>
```



Interpretation:

- Whether the WSLS model predicts WSLS-generated behavior better than the belief model, and vice versa.
- This turns “two strategies” into explicit competing generative hypotheses, beyond payoff.

Link to repository

https://github.com/ramona-tanovic/ACM.git

Conclusion

The role swap serves as a diagnostic manipulation: it separates strategies that merely react to outcomes (WSLS) from strategies that represent contingency structure (k-ToM-inspired belief learning). Stan is not an add-on here: hierarchical fits show that the intended parameters are identifiable from finite, noisy sequences (parameter recovery), and LOO formalizes whether the fitted mechanisms are predictively distinguishable beyond payoff summaries. Together, performance, role-swap diagnostics, and posterior signatures support an interpretation in terms of cognitive constraints (memory/recency, perseveration, and noise), rather than a purely descriptive tournament result.