

PRE VIVA VERSION

Please note this thesis has not yet been subject to final examination (viva voce). The outcome of the examination process typically involves some amendments to the thesis content.

ON STATE REPRESENTATIONS AND
BEHAVIOURAL MODELLING METHODS IN
REINFORCEMENT LEARNING

HENRIK SILJEBRÅT

GOLDSMITHS, UNIVERSITY OF LONDON

JULY 2022

A THESIS SUBMITTED FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

SUPERVISED BY PROFESSOR ALAN PICKERING

DEDICATION

Emma,

Jag ska berätta en sak för dig.

Vi fick aldrig chansen att följa Kahneman & Tverskys fotspår.

Det här verket är inte ens nära.

Men det är gjort. Jag hoppas du är stolt ändå.

Det här lilla ljuset i det vidunderliga okända är för dig.

ABSTRACT

Reinforcement learning (RL) – algorithms for learning from rewards – has proved successful in the cognitive sciences for explaining both neuronal signals and behaviour in animals, and for producing impressive results in artificial intelligence.

Essential to RL models are state representations. Based on what current state an animal or artificial agent is in, they learn optimal actions by maximizing future expected reward. But how are humans able to learn and create representations of states?

This thesis approaches this question from two fronts. First, we thoroughly investigate methods for fitting behavioural models to human lab data. In contrast to recent proposals, we find that the best methods for model selection – determining what model most likely generated some data – are based on maximum likelihood estimation, rather than Bayesian inference. We also demonstrate the importance of considering individual differences in model fitting: the model which best fits the performance of one participant may not fit the behaviour of another participant.

Second, we introduce Shapetask – a novel learning and decision-making task where participants must find hidden structure in a sequence, without the task explicitly rewarding the appropriate actions. We show that some humans can find this pattern, while RL cannot, unless equipped with appropriate state representations. We then show how previously proposed models that integrate RL with complex state representations can account for individual human behaviour in the Shapetask.

We argue our results add to the growing literature indicating a broader role for dopamine as one involving general sensory prediction errors, not just reward prediction errors. Further, we argue Shapetask holds promise for use in further research on the topic of state representation and task structure. Such research may illuminate the workings of animal brains, and contribute to artificial intelligence, where enhanced models of state representations could improve data efficiency and generalisability over current generation systems.

ACKNOWLEDGEMENTS

Thanks to my supervisor Alan Pickering. You have been very patient. Always able to bring my abstract ideas into shape (pun intended), and entertaining of my tangential interests and distractions. I am especially grateful you helped me through the worst of the pandemic with all your support.

Thanks to my family for your love and support, you are the best family I have ever had.

Thanks to my friends, older and newer. I would not have made it without you. I mean that. A, A, A, A, A, B, C, D, F, F, H, I, J, K, K, M, M, M, N, R, S, S, Y, Å - you're all great. I'm sure i forgot somebody. Sorry. Next round is on me.

Wednesday club, you're okay, I guess. Your partners are great though.

This document was typeset in Microsoft Word using the typographical look-and-feel *classicthesis* developed by André Miede and Ivo Pletikosić. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*". *classicthesis* is available for both LATEX and LYX: <https://bitbucket.org/amiede/classicthesis/>

CONTENTS

| | |
|-------|---|
| 1 | Introduction 21 |
| 1.1 | Chapter Overview 22 |
| 1.2 | Summary 24 |
| 2 | Background 25 |
| 2.1 | Reinforcement Learning and Conditioning 25 |
| 2.1.1 | A brief history of reward learning 25 |
| 2.1.2 | Rewards and Dopamine 28 |
| 2.1.3 | RL algorithms and dopamine function 30 |
| 2.1.4 | Action Selection 35 |
| 2.2 | Multiple Systems and Hierarchies 37 |
| 2.2.1 | Model-free and model-based signals and behaviour 38 |
| 2.2.2 | Dual systems in artificial intelligence 40 |
| 2.2.3 | Integrations of model-based and model-free 41 |
| 2.2.4 | Hierarchies 43 |
| 2.2.5 | Memory systems and beyond 46 |
| 2.2.6 | Summary and consolidation 50 |
| 2.3 | Representing the World as States and Tasks 52 |
| 2.3.1 | From single states to task sequences 54 |
| 2.3.2 | State representation in machine learning 56 |
| 2.3.3 | Attention to details 57 |
| 2.3.4 | Clustering states into beliefs 58 |
| 2.3.5 | Cognitive maps 60 |
| 2.3.6 | The successor representation 62 |
| 2.3.7 | Hierarchical state-action structure 63 |
| 2.4 | Summary and Discussion 64 |
| 3 | Methodological Introduction by analysing the bandit task 67 |
| 3.1 | What are models? 69 |
| 3.1.1 | Individual differences 70 |
| 3.2 | Models and task parameters 73 |
| 3.2.1 | Note on terminology 76 |
| 3.3 | Simulating performance in the bandit task 76 |
| 3.3.1 | Exploring effects of varying model parameters 77 |
| 3.4 | Exploring task parameters 80 |
| 3.4.1 | Varying bandit arm reward probabilities 81 |
| 3.5 | Recovering parameters with Maximum Likelihood Estimation 82 |
| 3.6 | Simulate and fit QL2 84 |
| 3.7 | Recovery quality with varying number of trials 88 |
| 3.7.1 | Confidence for individually fitted parameter values 94 |
| 3.7.2 | Standard errors with the Hessian 97 |

| | | |
|--------|---|-----|
| 3.8 | Recovery quality for various arm reward probabilities | 97 |
| 3.9 | Comparing MLE to Bayesian inference | 100 |
| 3.9.1 | Bayesian modelling tools | 103 |
| 3.9.2 | Fitting Bayesian models | 104 |
| 3.9.3 | Hierarchical MCMC compared to VBI | 110 |
| 3.10 | Model comparisons | 113 |
| 3.10.1 | Comparing models with BIC | 115 |
| 3.10.2 | More advanced model comparison methods | 116 |
| 3.10.3 | Collected methods for model comparisons | 117 |
| 3.10.4 | Comparing model comparison methods | 118 |
| 3.10.5 | Methodological considerations and conclusions | 123 |
| 3.11 | Applying methodology to human subjects | 125 |
| 3.11.1 | Model recovery check | 125 |
| 3.11.2 | Parameter recovery check | 126 |
| 3.11.3 | Model selections for Bandit dataset | 129 |
| 3.11.4 | Parameter value plots for Bandit dataset | 131 |
| 3.12 | Chapter summary | 133 |
| 4 | Stepping into States with Reversal Learning | 135 |
| 4.1 | Simulating performance in the reversal bandit task | 136 |
| 4.2 | Contrasting QL2 behaviour with human behaviour | 140 |
| 4.3 | Alternative algorithms without states | 141 |
| 4.3.1 | Dual α QL | 142 |
| 4.3.2 | Dual update QL | 143 |
| 4.4 | Adding states to Q-Learning | 144 |
| 4.5 | Algorithms that are more stateful to begin with | 146 |
| 4.6 | Behavioural comparison of agents | 149 |
| 4.7 | Parameter recovery performance | 151 |
| 4.8 | Model selection performance | 153 |
| 4.9 | Fitting human data in the Reversal Bandit task | 157 |
| 4.10 | Investigating the Worthy Bandit | 160 |
| 4.10.1 | Agent performance in the WorthyBandit | 161 |
| 4.10.2 | Human behaviour in the WorthyBandit | 169 |
| 4.10.3 | Model selection performance in the WorthyBandit task | 171 |
| 4.10.4 | Fitting human data in the WorthyBandit task | 176 |
| 4.11 | Chapter summary and Discussion | 177 |
| 5 | The Shape Sequence Task | 179 |
| 5.1 | Creating the Shape Sequence task | 180 |
| 5.1.1 | Scoring the Shapetask | 183 |
| 5.1.2 | Pilot version | 184 |
| 5.1.3 | Pilot version 2 | 189 |
| 5.1.4 | Balltask becomes Shapetask | 192 |

| | | |
|-------|---|-----|
| 5.1.5 | Defining behavioural groups in Shapetask | 198 |
| 5.1.6 | Summary of Shapetask results | 201 |
| 5.2 | RL behaviour in Shapetask | 202 |
| 5.2.1 | QL ₃ behaviour in Shapetask | 205 |
| 5.2.2 | Manipulating states to create SEQL ₃ | 209 |
| 5.2.3 | Simulating SEQL ₃ behaviour in Shapetask | 210 |
| 5.2.4 | General discussion of RL behaviour | 214 |
| 5.3 | Hierarchical RL | 215 |
| 5.3.1 | Applying HRL to Shapetask | 217 |
| 5.3.2 | HRL Results in Shapetask | 218 |
| 5.3.3 | Constraining structure | 224 |
| 5.3.4 | Overall behaviour of HRL-22 | 226 |
| 5.3.5 | HRL Discussion | 229 |
| 5.4 | The Successor Representation | 230 |
| 5.4.1 | Latent exploration of Gridworld mazes | 234 |
| 5.4.2 | Shapetask as a maze | 239 |
| 5.4.3 | Playing Shapetask with SRTD | 241 |
| 5.4.4 | Discussion of SRTD | 244 |
| 5.5 | Control group models | 245 |
| 5.6 | Model selection for Shapetask | 246 |
| 5.6.1 | Model selection performance with MLE fitting | 247 |
| 5.6.2 | Model selection with Approximate Bayesian Computation | 248 |
| 5.6.3 | Implementing ABC for Shapetask | 251 |
| 5.6.4 | ABC Model Selection Performance | 251 |
| 5.6.5 | Summarising Model Selection Performance | 259 |
| 5.7 | Fitting Models to Human Behaviour in Shapetask | 260 |
| 5.8 | Chapter summary and Discussion | 268 |
| 5.8.1 | Technical Discussion | 270 |
| 5.8.2 | General Discussion | 270 |
| 6 | Discussion and Future Work | 273 |
| 6.1 | Summary of Key Findings | 273 |
| 6.1.1 | Chapter Three | 273 |
| 6.1.2 | Chapter Four | 275 |
| 6.1.3 | Chapter Five | 277 |
| 6.2 | Limitations | 281 |
| 6.3 | Future Work | 285 |
| 6.3.1 | Task variations | 286 |
| 6.3.2 | Models | 287 |
| 6.4 | Final remarks | 289 |
| 7 | References | 291 |

LIST OF FIGURES

- Figure 2.1 Markov Decision Process 31
- Figure 3.1 Overview of simulations (robots) as models of animal behaviour in the world 70
- Figure 3.2 Different scenarios of individual differences in some experimental task 72
- Figure 3.3 Performance summary for agents QL2 and RandomBias playing the Bandit task for 1000 trials 77
- Figure 3.4 Probability of picking the correct arm – the arm with highest reward probability – for parameter sweeps with QL2 (left) and RandomBias (right) 79
- Figure 3.5 QL2 performance across different number of task trials 80
- Figure 3.6 Impact of arm reward probability differences for the Bandit task 81
- Figure 3.7 Correlation plots between parameter values used in simulations (y-axis) and recovered parameter values found by fitting the QL2 model (x-axis) 86
- Figure 3.8 Likelihood surfaces for two separate choice sequences, both generated by the same QL2 agent 88
- Figure 3.9 Correlation plots of α values, where simulation values are on the y-axis and fitted values on the x-axis 90
- Figure 3.10 Violin plots showing distance between simulated α parameter and fitted α 91
- Figure 3.11 Correlation plots for β parameter where fitted values are on the x-axis and simulation values on the y-axis 92
- Figure 3.12 Violin plot showing distance between simulated and fitted β on y-axis and number of trials as categories on the x-axis 92
- Figure 3.13 Correlation plots for β parameter with fitted values on x-axis and simulation values on y-axis 93
- Figure 3.14 Histograms showing the distribution of distance between fitted α values and the α values used for simulations 95
- Figure 3.15 Histograms showing the distribution of distance between fitted β values and the β values used for simulation 96
- Figure 3.16 Correlation plots for fitted α on x-axis and simulation α on y-axis 98
- Figure 3.17 Violin plots showing distance between fitted and simulated α on y-axis for each arm reward probability difference on the x-axis 99
- Figure 3.18 Correlation plots for simulated β on y-axis and fit β on x-axis 99
- Figure 3.19 Violin plots showing distance between simulated and fit β on y-axis and arm difference categories on x-axis 100
- Figure 3.20 Histograms showing the parameter value distributions for the 50 subjects in our dataset 105
- Figure 3.21 Correlation plots for fitted β on x-axis and simulation β on y-axis 107
- Figure 3.22 Boxplots showing distance between simulated α and fitted α on y-axis, with number of trials on x-axis 108

- Figure 3.23 Boxplots showing distance between simulated β and fitted β on y-axis, with number of trials on x-axis 108
- Figure 3.24 Box plots showing the distance between simulated parameter and fitted parameter on y-axis with each trial count category on the x-axis 110
- Figure 3.25 Line plots showing distance between simulated and fit β value on the y-axis and simulated β value on the x-axis. 111
- Figure 3.26 Scatterplots of distance between fitted and simulated β on y-axis and SD estimate on x-axis 112
- Figure 3.27 HDI check for β estimates for MCMC and VBI methods 113
- Figure 3.28 Left: Confusion matrix. Right: Inverse confusion matrix 115
- Figure 3.29 Plots showing what model was selected (left) and whether the model selected was correct (right) 119
- Figure 3.30 Parameter value distances for each method and agent/model 119
- Figure 3.31 Plots showing model selection results for dataset B across different model selection methods 120
- Figure 3.32 Parameter value distances between simulated parameter value and fitted parameter value for each method considered 120
- Figure 3.33 Model selection plots comparing different model selection methods for dataset C 121
- Figure 3.34 Parameter value distances between simulated and fitted values for QL2 (left, middle) and RandomBias (right) 121
- Figure 3.35 Model selection performance for the different methods 122
- Figure 3.36 Parameter value distances between simulated parameter value and fitted value 122
- Figure 3.37 Model selection plots for 1000 simulated subjects where half were QL2 agents and half were RandomBias 126
- Figure 3.38 Barplots showing details on how many of the misidentified cases were RandomBias misidentified as QL2 (left), or QL2 identified as RandomBias (right) 126
- Figure 3.39 Boxplots showing average distance between simulated parameter and fitted parameter across the methods 127
- Figure 3.40 Plots for QL2 α parameter for each method 127
- Figure 3.41 Plots for QL2 β parameter for each method 128
- Figure 3.42 Plots for RandomBias bias parameter for each method 128
- Figure 3.43 Model selection counts for the Bandit dataset 129
- Figure 3.44 Plot showing what action was selected 130
- Figure 3.45 VBAT-MLE model selections contrasted by task order 130
- Figure 3.46 Probability of model being QL2 across all subjects 131
- Figure 3.47 Plot showing fitted RandomBias parameter value for bias on y-axis, across all subjects (x-axis) 132
- Figure 3.48 Plot showing fitted QL2 α parameter value on y-axis for each subject on x-axis 132
- Figure 3.49 Plot showing fitted QL2 β parameter value on y-axis for each subject on x-axis 132
- Figure 4.1 Performance averaged across QL2 agents using parameter values from the entire parameter space 137

- Figure 4.2 Performance for parameter value combinations of QL2 α (x-axis) and β (coloured lines) 138
- Figure 4.3 Action selections for QL2 agents using $\alpha=0.4$ and β as per the plot legend 139
- Figure 4.4 Proportion of correct choices (y-axis) for QL2 agents using $\alpha=0.4$ and β as per the plot legend 139
- Figure 4.5 Average proportion of correct arm choices (y-axis) for human subjects performing the reversal bandit task 140
- Figure 4.6 Average arm (action) selection across subjects (y-axis) for each individual trial (x-axis) 141
- Figure 4.7 Proportion correct arm choices (y-axis) for Dual- α QL agents 142
- Figure 4.8 Proportion of correct choices (y-axis) for parameter combinations of Dual Update QL agent 144
- Figure 4.9 Action selections (y-axis, arm1 or arm2, 0 or 1 respectively) on each trial 146
- Figure 4.10 Proportion of correct choices (y-axis) in the ReversalBandit task for the HMM agent (left) and HMM- δ agent (right) 148
- Figure 4.11 Proportion of action choices (y-axis; arm1=0, arm2=1) on each trial (x-axis) for 100 HMM agents 149
- Figure 4.12 Proportion of correct choices (y-axis) in the ReversalBandit task for our agents and two human groups (x-axis) 150
- Figure 4.13 Proportion correct choices (y-axis) at different number of trials after each switch point (x-axis) 151
- Figure 4.14 Simulated parameter value (y-axis) plotted against fitted parameter value (x-axis) 152
- Figure 4.15 Simulated parameter value (y-axis) plotted against CBM-HBI fitted parameter value (x-axis) for subjects simulated with and fitted with HMM- δ 153
- Figure 4.16 Proportion of correctly identified cases (y-axis) for each method (colours as per legend) 155
- Figure 4.17 Proportion of cases (y-axis) for each simulated agent (x-axis) identified as what model (differently coloured bars as per legend) 156
- Figure 4.18 Number of subjects (y-axis) that were best fit with each model (x-axis) 158
- Figure 4.19 Model probability (y-axis) for each subject (x-axis) 159
- Figure 4.20 Reward magnitude (y-axis) for each card in the decks (x-axis) 161
- Figure 4.21 Total score in the task (y-axis) depending on how many cards are picked (x-axis) from each deck (coloured lines as per legend) 161
- Figure 4.22 Number of cards drawn (y-axis) from the good deck for each α (x-axis) and β (coloured lines as per legend) parameter value combination 163
- Figure 4.23 Total reward score (y-axis) for each α (x-axis) and β (coloured lines as per legend) parameter value combination 163
- Figure 4.24 Closer look at the parameter combination $\alpha=0.4$, $\beta=10$ with standard rewards 164
- Figure 4.25 Behavioural differences between winners and losers 165
- Figure 4.26 Performance for RandomBias agent (y-axis, total points scored) across the parameter value space (x-axis) 166
- Figure 4.27 Simulations for SEQL2 agent across the parameter space 167

- Figure 4.28 Performance and behaviour of the HMM agent in the WorthyBandit task 168
- Figure 4.29 Performance and behaviour of the HMM- δ agent in the WorthyBandit task 169
- Figure 4.30 Human behaviour in the WorthyBandit task 170
- Figure 4.31 Score in the WorthyBandit task (y-axis) for each agent type (x-axis) 171
- Figure 4.32 Model selection plots for the two methods 173
- Figure 4.33 Proportion of cases (y-axis) for each agent type (x-axis) identified as what model type (different coloured bars as per legend) 174
- Figure 4.34 Probability (y-axis) for the model (coloured boxes as per legend) being the best fit for the group of simulated subjects of each agent type (x-axis) 175
- Figure 4.35 Count of model selections (y-axis) for each model (x-axis) for each method (coloured bars as per the legend) 176
- Figure 4.36 Histogram with number of subjects (y-axis) across the total score (x-axis) received in the experiment 177
- Figure 5.1 The basic structure of the shape sequence task 181
- Figure 5.2 Shapetask version 1, a.k.a. Balltask 185
- Figure 5.3 Summary scores for all subjects in the pilot version of Shapetask 187
- Figure 5.4 Averaged Likert choice (left) and Shift Predict score (right) for each pilot participant (coloured lines) and for each shape position (x-axis) 188
- Figure 5.5 Likert choice (left) and proportion of shift predict choices (right) for subject two in the pilot data 188
- Figure 5.6 Proportions (y-axis) across all subjects and trials for each score type (x-axis), separated by experiment version (rows) and ball position (columns) 190
- Figure 5.7 Proportion of choices being “shift predict” (y-axis) for each ball position (x-axis) 191
- Figure 5.8 Proportion of choices (y-axis) for each score type (x-axis) averaged across all subjects and trials 194
- Figure 5.9 Proportion of choices (y-axis) for each score type (coloured lines as per legend) averaged across all subjects 196
- Figure 5.10 Proportion of choices (y-axes) for each score type (columns) and each individual subject (separate lines) 197
- Figure 5.11 Shapetask groups 199
- Figure 5.12 Shapetask groups for individual subjects 200
- Figure 5.13 Simple maze task 203
- Figure 5.14 Comparison of QL2 (top) and QL3 (bottom) agents 204
- Figure 5.15 Q-Learning with future discount in simple maze example 205
- Figure 5.16 Overview of behaviour for all simsets 206
- Figure 5.17 Average score (y-axis) for each simset (coloured lines) and shape position (x-axis), separated by score type (columns) 206
- Figure 5.18 QL3 behaviour groups as per legend 207
- Figure 5.19 QL3 parameter values separated by behavioural group 208
- Figure 5.20 QL3 parameter value plots for each group 209
- Figure 5.21 SEQL3 performance in Shapetask for each simset (separate lines), averaged across all trials 211
- Figure 5.22 Behavioural groups for SEQL3 in Shapetask 212

- Figure 5.23 Parameter value correlations for SEQL3 separated by groups (colours per legend) 213
- Figure 5.24 Overview of HRL 216
- Figure 5.25 Overview of HRL performance in Shapetask 219
- Figure 5.26 HRL performance in Shapetask by group 219
- Figure 5.27 Parameter value pair plot for HRL in Shapetask 220
- Figure 5.28 Proportion of choices (y-axis) for each shape position (x-axis) and score type (columns) for each group (coloured lines as per legend) 221
- Figure 5.29 Proportion of Shift-predict (y-axis) for each shape position (x-axis) for example simset 222
- Figure 5.30 Taskset selections (y-axis) for each trial (x-axis) and subject type (rows) 223
- Figure 5.31 Behaviour of two example subjects from the same simset 223
- Figure 5.32 Taskset selections for best and worst subjects in best performing simset using HRL-22 224
- Figure 5.33 Taskset selections for best and worst subjects in best performing simset using HRL-22 and 270 trials in Shapetask 225
- Figure 5.34 Behaviour of the worst and best subjects in the best performing simset using HRL-22 and 270 trials in Shapetask 225
- Figure 5.35 Taskset selections for the best and worst subjects in best performing simset using HRL and 270 trials for Shapetask 226
- Figure 5.36 Behaviour of the best and worst subjects in best performing simset using HRL and 270 trials for Shapetask 226
- Figure 5.37 HRL-22 performance in Shapetask, separated by groups as per legend 227
- Figure 5.38 Pair plot comparing parameter value spaces for each group 228
- Figure 5.39 Gridworld maze 234
- Figure 5.40 Illustration of the Hotel California problem 236
- Figure 5.41 Overall SRTD behaviour in Shapetask 242
- Figure 5.42 SRTD behaviour grouped by winners and others 243
- Figure 5.43 SRTD behaviour grouped by high winners, winners and others 243
- Figure 5.44 SRTD parameter values for each group 244
- Figure 5.45 RandomBias behaviour in Shapetask 246
- Figure 5.46 Overall model selection performance for Shapetask 247
- Figure 5.47 Model selection performance in Shapetask 248
- Figure 5.48 Model selection performance when HRL is excluded from models fitted 252
- Figure 5.49 Model selection performance in Shapetask for ABC method 253
- Figure 5.50 Confusion matrices for model fitting of Shapetask 255
- Figure 5.51 Bayes Factor (y-axis) for the best (colours as per legend) and second best fitted models for each simulated model (x-axis) 256
- Figure 5.52 Behavioural curves for each simulated agent type (rows) showing proportion of choices (y-axis) for each shape position (x-axis) and score type (columns) 258
- Figure 5.53 Model selections for human subjects in Shapetask 261
- Figure 5.54 Boxplot showing Bayes Factor (y-axis) for each fitted model (colours as per legend) and Shapetask version (x-axis) 262

- Figure 5.55 Behavioural plots for human subjects in Shapetask, grouped by fitted model as per colours in legend, with score type in columns and task version on rows 263
- Figure 5.56 Learning curves for human subjects in BOB task 264
- Figure 5.57 Individual summary scores for subjects playing Shapetask BOB 266
- Figure 5.58 Shapetask BOB subjects, coloured by fitted model (see legend) and separated by behavioural groups (rows) 267
- Figure 5.59 Learning curves for individual subject 38 in BOB-NR task, fitted best with SRTD model 268

LIST OF TABLES

- Table 3.1 Bayes Factor interpretation for values on log scale (left most column), raw ratio (middle column) 114
- Table 5.1 SR example using a gridworld maze 232
- Table 5.2 Overview of Shapetask as a maze 240

EQUATIONS

List of equations

- 2.1 Rescorla-Wagner 30
- 2.2 RL State value 32
- 2.3 RL State-action value 32
- 2.4 TD prediction error 33
- 2.5 Actor-critic Value update 33
- 2.6 Actor-critic Policy update 34
- 2.7 SARSA 34
- 2.8 Q-learning 34
- 3.1 QL2 74
- 3.2 SoftMax 74
- 3.3 Likelihood a posteriori 83
- 3.4 Likelihood 83
- 3.5 Maximum likelihood 84
- 3.6 Bayes' rule 101
- 3.7 Bayes Factor 113
- 3.8 Bayes Factor with uniform priors 114
- 3.9 Bayesian Information Criterion 115
- 4.1 State Enhanced QL2 145

| | |
|---|-----|
| 4.2 HMM posterior belief | 147 |
| 4.3 HMM probability of observation | 147 |
| 4.4 HMM probability of next state | 147 |
| 4.5 HMM Timestep change for γ | 148 |
| 4.6 HMM Probability of observation for WorthyBandit | 167 |
| 5.1 QL3 | 202 |
| 5.2 HRL | 215 |
| 5.3 SR value calculation | 231 |
| 5.4 SR successor representation update | 231 |
| 5.5 SR temporal difference error | 231 |
| 5.6 SRTD value calculation | 233 |
| 5.7 SRTD successor representation update | 233 |
| 5.8 SRTD weights update | 233 |
| 5.9 SRTD reward prediction error | 233 |
| 5.10 RandomBias control model | 245 |

CODE SNIPPETS

Code Snippet 3.1 Q-learning algorithm in Python 75

ETHICAL STATEMENT

Experiments in chapter five were approved by the ethics board of the Computing Department of Goldsmiths, University of London. No animals were harmed in making this work. However, we do rely on previous research where unnecessary harm was done to animals, for example lesion studies. We hope such harm may be avoided in the future.

1 INTRODUCTION

How do we find structure in the world around us?

When we are born, the world is blurry mess. The only shapes we can see in the fog of light are faces [1]. We are helpless. But through the combination of nature and nurture, we learn to make sense of our senses and control our limbs. We walk, we talk, we learn to see the birds and the bees.

As we grow, we also learn to look beyond the physical into the abstract. Like children learning how to count farther than the fingers on their hands. Somehow, we manage to find structure within ourselves, the world and the interaction in-between. We know what is important to recognise for the task at hand, but it may be irrelevant for another. We have to infer invisible truths, like reading someone's feelings on their face. Get it right, and you are rewarded with a hug.

This thesis interrogates these topics. More specifically, how do we learn and use structure – models of the world – to support reward-based learning?

In the last decade, reward-based learning algorithms in the form of reinforcement learning (RL) have demonstrated impressive feats such as playing video games at human level and beyond [4, 174, 196]. This is especially intriguing from a neurobiological perspective as we have quite strong evidence that these algorithms have neural correlates in the midbrain dopaminergic system of the mammalian brain [191]. This system projects, among other areas, to a large part of the basal ganglia, which is an evolutionarily old system of brain structures that exists in all vertebrates [103]. Reward-based learning can be found in very basic vertebrates [5], as well as invertebrates [17, 201]. Recent evidence suggests RL-like mechanisms exist even in insects [22]. Taken together, we thus have strong reasons to study RL more closely as one of the most basic forms of learning in humans and other animals.

Although the discovery of the neurobiological correlates of RL, now often described as the reward prediction error hypothesis of dopamine [232], is a big success in the cognitive sciences, there are still many outstanding questions. There is disparate evidence for what specific algorithm(s) are used – can one algorithm explain all varieties of learning mechanisms or are there several? Are there

hierarchical organizations in the brain that may explain these disparate results, and in what way would the higher order systems be involved? How may such interacting systems allow animals to create and use structure from and in complex real-world environments [113, 136, 190, 227]?

Investigating these questions is the main goal of this thesis. The underlying hypothesis we will work from is one where RL is a basic form of learning, which is supported by multiple higher-order, and probably evolutionary newer, systems. We assume these higher order systems create, use and manipulate “state representations” appropriate for the current task, and feed these to the RL system. We base these hypotheses partly on the above-mentioned strong evidence of RL as a basic form of learning in many animals. Partly, we are inspired by recent proposals that the dopaminergic system may indeed code for general prediction errors, not just rewards [46, 88, 115, 219].

Our approach to studying these questions is to introduce a new decision-making task called the shape sequence task. We show how this task is appropriate to study the questions we are interested in. We will introduce models found in existing literature and test their accounts of how complex state representations may explain our results from experiments with human participants.

However, in order to test scientific models of the world on theories of human brain models of the world, we need data modelling methods. We compare and contrast common methods for modelling behavioural data in existing human behavioural datasets and show the importance of fully testing such methods before applying them. We show the best performing methods are not those one may expect. Based on those results we also highlight the need to consider individual differences in task performance, which can be marked. These differences thus change what models may fit best to the data from an individual participant.

1.1 CHAPTER OVERVIEW

Following this introductory chapter, in chapter two, we start investigating the question of state representations by connecting them to the concepts of model-free and model-based RL: learning action values directly from experience or learning/using a model of

the world, respectively. From there we will see how the picture is perhaps even more complex, involving many systems and hierarchies. Common to each idea is that animals such as human beings manage to create useful representations of incoming sensory information, representations then used for learning and decision making.

In chapter three, we survey some of the methodological landscape of fitting models to data. We show how often this is a technically difficult process (for real lab data at least) and how important it is to fully test the algorithm- and task combination under consideration. This may sound like a well-studied subject, but we think there is surprisingly little published work addressing these questions. Published papers often use brief descriptions of their models and fitting methods because they use “standard methods”. However, in our experience, even using such standard methods reveal many difficulties that are under-emphasized in the existing literature. Adding to that, we further show, using a standard two-armed bandit task, with both simulated and human subjects, how modern methods that often are promoted as better, may not actually always work that well with real data.

In chapter four, we introduce the concept of states by modelling data from two kinds of reversal learning tasks. We further investigate the methodological aspects found in chapter three and apply our findings to datasets with human participants. We show that it is possible individuals use different strategies approaching a task, that are all successful, highlighting the need for individual differences in model fitting.

In chapter five, we introduce the novel shape sequence task – or Shapetask for short. We explain the process of developing the task and show how and why this task is appropriate for the research questions under investigation. We present results from experiments with human subjects playing various versions of Shapetask, showing how some participants are able to find the underlying pattern of the task surprisingly quickly. We then show how standard RL with manipulated state representations is needed to account for humans solving the task. Models from the literature proposing how such state representations may be connected with neurobiological correlated are introduced, and their behaviour in Shapetask simulated. Finally, we

introduce another numerical method for model fitting, required to fit one of the models under consideration. Using this method, and knowledge gained in previous chapters, we fit the collected models to our human data and discuss the results. We argue Shapetask holds great prospects for further research in the field.

1.2 SUMMARY

We aim to investigate how humans are able to find task structure and apply appropriate state representations to successfully solve tasks. Our most important contributions to this field are two-fold.

First, we investigate and critique existing methods for fitting behavioural models to human lab data. We find that the best methods are not the ones that are usually promoted. We also highlight the importance of considering individual differences in model fitting: the model which best fits the performance of one participant may not be able to fit the behaviour of another participant.

Second, as noted above, we introduce a novel decision-making task called Shapetask, and show how it can be useful for investigating state representations in humans. We test how well previously proposed models that integrate RL with complex state representations may account for human behaviour in the Shapetask.

We argue our results adds to the growing literature indicating a broader role for dopamine as one involving general sensory prediction errors, not just reward prediction errors. Further, we argue Shapetask holds promise for use in further research on the topic of state representation and task structure. Such research is important, as it may not only elucidate the workings of the animal brain but also provide valuable contributions to artificial intelligence, where improved models of state representations could vastly improve data efficiency and generalisability over current generation systems.

2 BACKGROUND

In this chapter we provide an overview of the theory of reinforcement learning (RL), and its relation to other areas of the cognitive sciences. We start with a brief history where we connect RL to its origins of animal learning and conditioning. We then look closer at the neurobiological connections, and how studies have shown conflicting evidence of potential multiple types of RL. We discuss how and if multiple systems are involved in reward-based learning, which will take us into contact with memory, hierarchies and artificial intelligence research. From there, we narrow our focus to state representations, present recent findings on this topic and identify areas of open questions. With all the pieces in place, we can then frame and ask our research question.

We do not aim to be comprehensive, rather this chapter is meant as a tour to set the stage for later chapters. Most, if not all, of the topics we cover are active fields of research on their own.

2.1 REINFORCEMENT LEARNING AND CONDITIONING

2.1.1 A BRIEF HISTORY OF REWARD LEARNING

Conditioning is the process of associating stimuli with other stimuli such as rewards, for example a bell with food, a case called classical or Pavlovian conditioning. The other main variation is instrumental or operant conditioning where reward is given only when a certain behaviour has been performed, for example pressing a lever.

Conditioning was first studied more than a hundred years ago. Thorndike [263] was one of the pioneers, establishing the “law of effect” which states that behavioural responses that lead to pleasure in a certain situation are more likely to occur again in the same situation, and behavioural responses leading to displeasure are less likely to occur again in the same situation. This laid the groundwork for Pavlov, Watson and Skinner and the so-called behaviourist school of psychology that dominated much of the early to mid-20th century [186]. Behaviourists – especially radical ones – believed internal states of animals were not important, only basic association learning between stimulus and response were important. For example, Pavlov [198] trained dogs to learn that shortly after the sound of a bell, food

would arrive. After sufficient training, the bell alone was enough to make the dogs salivate.

With time, other viewpoints came to light. Tolman [269] proposed, based on experiments with rats, the concept of “cognitive maps”. When trained to find food at one end of a T-maze, rats would at test be put in a T-maze turned 180 degrees without food. If it was indeed simple stimulus-response behaviour, the rats would turn to the same direction they turned when learning. Instead, they turned towards where the food would be spatially. By the late 1970s a more balanced view had emerged, where association learning was seen as important, but influenced by and interacting with internal processes. The seminal work of this more balanced view is the book by Mackintosh [162], where the importance of processes like attention to stimuli and generalisation were stated. As we shall see below, many of the questions raised by Mackintosh are ones we are still studying today.

Also in the 1970s, a fundamental link between motivation and dopamine was found, mainly based on experiments with the effect of dopamine influenced drugs, Parkinson’s disease and dopamine-lesioned rats [23]. Later in the 1980s, using direct neuron spike recordings in monkeys, dopamine cell firing gave insights into dopamine’s role in invigorating the animal’s current behaviour [23].

RL was first studied in the early 1980s as a formalization of conditioning [260] and built on rules of learning and prediction error formalised a decade earlier by Rescorla & Wagner [211]. In short (but see section below on algorithms), an unexpected reward or a reward bigger in value than expected, leads to a positive prediction error. An unexpected lack of reward or one lower in value than expected leads to a negative prediction error. These prediction errors are viewed as the main drivers of learning under reinforcement and learning stops when prediction errors are zero. Importantly, there are two main categories of RL algorithms: model-free (MF) and model-based (MB) [258]. MF algorithms directly update values of actions based on experience with the world, while MB algorithms have an internal model of the world that is updated from experience with the world, and then the model is used to predict the values of actions.

In early-mid 1990s it was found that RL, specifically temporal difference RL (TDRL; a model-free algorithm able to take expected future rewards into account [259]), could explain midbrain

dopaminergic activity results from neuron recordings in animals [19, 230, 233]. The neuronal activity did not just predict upcoming reward based on a stimulus (like the bell and food for Pavlov's dogs mentioned above), but in the absence of expected reward, activity would decrease. In other words, dopamine activity reflected the reward prediction error of RL.

Since then, many studies have confirmed the connection between dopaminergic cell responses and the prediction of rewards, leading to the reward prediction error (RPE) hypothesis of dopamine [116, 232, 266]. These prediction errors are reflected in the phasic (see next section) increase or decrease in dopamine cell firing, respectively for positive and negative reward prediction errors. The RPE hypothesis entails that this very time-specific change in phasic dopamine cell firing serves as a reinforcement signal for learning. To connect back to the more general results on motivation from the 1970s and 1980s mentioned above, [23] explains that the lack of dopamine can be seen as a RPE that is constantly negative and causes values of actions to update towards zero.

In the 2000s, evidence started appearing of model-based influences on dopamine function [65], and has since been further investigated and become part of the standard account [63, 69, 143, 148, 297]. This line of research also started to highlight overlap in the systems/brain areas used for model-free vs model-based reasoning [69, 244].

In the last few years, this picture of dopamine function has become increasingly complex, partly stemming from difficulties distinguishing between motivation and reward [23, 295] and partly because the functions of MF and MB learning seem to be intermixed in the dopaminergic system(s) [69]. This has thus caused calls for a new view of "multiplexing" (integrating) model-based and model-free signalling [143], for example by using a variation of RL called successor representation [88] or adding local control of dopamine, switching between MF and MB by way of other neurotransmitters [23].

The latter point is important, and should be stressed, that dopamine alone is not the entire story about reward processing. There are many other neurotransmitters that either have indirect effects by regulating dopamine (including acetylcholine [264]; substance P [30]; glutamate [44]); while other neurotransmitters having separate but

complementary functions such as serotonin [238]. To further add to this complexity, prediction errors for rewards have been found in areas outside the “main” areas we describe below (basal ganglia/midbrain and frontal cortex), for example the cerebellum [283].

However, we will base the current work mainly on findings in the dopamine system as it is the most studied system with the clearest connections to RL.

2.1.2 REWARDS AND DOPAMINE

Schultz [231] defines three main functions of rewards. First, they act as positive reinforcers to induce learning. Second, they elicit movements towards desired objects, meaning they act as factors for decision making. This function is sometimes defined as motivation [23], but can also be described as a subjective value formalized as economic utility [234]. Third, rewards have a role in emotions like pleasure and desire [231].

The processing of rewards is sequential and starts out with the sensory components of object detection and identification, followed by valuation of the objects which leads to decisions, actions and reinforcement [231]. The main brain areas involved are, as previously mentioned, the midbrain dopamine cells, in the substantia nigra and ventral tegmental area (VTA), and the structures to which these midbrain dopamine cells projects to such as striatum, orbitofrontal cortex, the amygdala [231, 232] and the anterior cingulate cortex [107]. The midbrain areas and striatum are part of the basal ganglia, long implicated in action selection [105, 209]. Neuronal signals in these different areas code for different aspects of reward such as amount, probability, uncertainty, subjective value, utility and risk [231].

The dopamine neuron response works in three main ways on three different time scales [231]. On the sub second timescale, *phasic* responses code for reward value in the form of reward prediction errors and this is the response described with RL algorithms. The *slow* or *intermediate* response acts on the timescale of seconds to minutes, and is related to behavioural activation, forced deactivation, stress, attention, reward-related behaviour, punishment and movement. On the longest timescale of minutes and more, the *tonic* response is related to the level or amount of dopamine transmitter and receptors available. The tonic response is involved in many varied and general

functions such as movement, cognition, attention and motivation and is the main factor in psychiatric illness such as Parkinson's, ADHD and schizophrenia. Below, whenever we refer to dopamine response it is assumed we speak of the phasic responses. But as just described, we also see there is no clear distinction in mechanism between these timescales. Furthermore, as will also be discussed later, functions like attention plays an important role in RL related learning as well.

As touched on above, the distinction between rewards, learning and motivation can also be unclear. Berke [23] defines motivation as a forward model that uses predictions of future rewards to energize current behaviour, whereas learning looks backwards to update values of states and actions leading to rewards. In order to differentiate between learning and motivation, [23] proposes a model where the value coding (learning) happens through synaptic plasticity (long term potentiation [26]). The use of those values is mediated through dopamine neuron firing. By relying on cholinergic interneurons to switch between learning and motivation process, dopamine can thus work as a modulator of resource allocation decisions, both in the sense of deciding if energy use is worth expending to work for a reward and also whether to engage model-based decision making. In this sense, the argument is similar to [137] where it is argued for a mechanism of cost-benefit arbitration between model-free and model-based decision making. Unfortunately, the study [23] bases their argument on, [108], uses micro dialysis which can sample only as quickly as around a second or longer so it is difficult to relate the findings to the very precise sub second timescale that [231] counts as phasic. The implication of the model proposed by [23] is that one needs to look at local terminal receptor density. This is important because both pre- and postsynaptic dopamine can be affected by other neurotransmitters. This confusion of timescales is unfortunately common in the literature on dopamine, which often creates difficulties in comparing results to the underlying models.

Another long-standing puzzle is the role of aversive reinforcers (punishments). [231] argues the dopamine response is only about the physical impact of aversive reinforcers, but there are also indications that dopamine response can explain the effects of punishments if the different kinds of dopamine receptors are taken into account [31]. Another line of research suggests that punishments may instead be mediated by partly different circuits than rewards, for example

involving the lateral habenula [166, 249]. As punishments are related to fear, the amygdala and endogenous opioids also appear to be involved [7, 299]. In short, the story for punishments is more unclear than that of positive reinforcers and we will thus mostly focus on the latter – rewards.

2.1.3 RL ALGORITHMS AND DOPAMINE FUNCTION

Here we will briefly explain the mathematical concepts of RL, present details of three algorithms and then discuss these from the perspective of neurobiological plausibility.

2.1.3.1 Short primer on Markov Decision Processes

As we have discussed, the RPE hypothesis tells us that learning is mediated through errors in reward prediction. The basis of this was formalized by Rescorla & Wagner [211] in the following way, here adapted for simplicity (Q is used for easier comparison with algorithms presented below):

$$\Delta Q = \alpha(R - Q) \quad 2.1$$

where ΔQ is the change in associative strength (i.e., learning) between a stimulus and reward and is proportional to the prediction error ($R - Q$). R is the actual value of the reward and Q is the predicted value. α is the so-called learning rate and modifies how much the error should influence learning at each learning opportunity, for example a trial in an experiment. A reward that is fully predicted does not contribute to learning, since the error would be zero and associative strength can also decrease if reward is withheld. This way of removing an association is called extinction in psychological literature [162].

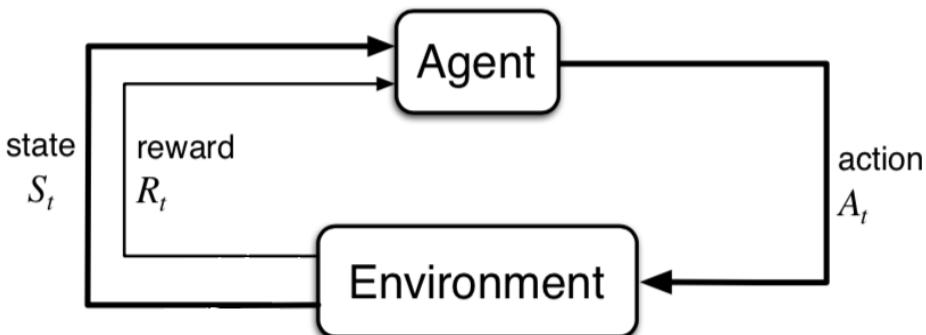


Figure 2.1 Markov Decision Process. On each timestep t , an agent interacts with the environment by observing state S and reward R , then taking an action A . The environment responds by transitioning into the next timestep, where a new state and reward are observed. Adapted from [259].

RL algorithms build on the concept of Markov Decision Processes (MDPs) [258], as seen in Figure 2.1. In the discrete case, an agent - artificial or animal - interacts with an environment by observing the world at time t , thus receiving two signals, the state S and the reward R . The reward value may be negative (interpreted as punishment), zero (no reward/absence of reward) or positive. The agent then takes a step (or action, A) in the environment. The action taken will cause the environment to transition to a new state according to some transition function $P_a(s, s_{t+1})$, providing the agent with a new observation. Importantly, an MDP satisfies the Markov property, meaning that all information to predict the future is contained within the current state, regardless of what has happened in the past. As we will see below, this information can be summarised by for example storing values of each state visited. One consequence of MDPs is that the mathematical proofs of RL algorithms converging to optimal solutions depends on the agent visiting all states an infinite number of times [258]. However, the solution can often be approximated to a useful degree much earlier than infinity, which is encouraging.

In RL, the goal is for the agent to maximize the total reward received. We thus want to find a behavioural *policy*, most often denoted π , for what action to take in each state to find said maximum. In order to do so we need a measure of the value of each state, the value function $V(s)$. The value defines how good a given state is based on the expected return of this state, under a specific policy, i.e., what future sum of rewards is expected moving on from this state [258]:

$$V_\pi(s) = E_\pi\{R_t | s_t = s\} = E_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s \right\} \quad 2.2$$

This means that the value of state s , given we follow the policy π , is the discounted sum of rewards of all states k following that policy. The γ here is the discount parameter, controlling how much influence rewards far into the future will have. The discount parameter can help solve the so called “credit assignment problem”, meaning the problem of assigning values to preceding states when rewards are many steps into the future. In other words, as we have noted, the value V is the discounted sum of rewards. Equation 2.2 (and variations of it) is sometimes called “expected utility” and is also the basis for prospect theory, which aims to account for human biases [128]. Here we simply call it the state-value function.

Later in this chapter, and in the following chapters, we often use state-action values. In each state, we can have multiple actions, where each state-action pair has a value which depends on future states and actions in similar fashion to the previous equation:

$$\begin{aligned} Q_\pi(s, a) &= E_\pi\{R_t | s_t = s, a_t = a\} \\ &= E_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a \right\} \end{aligned} \quad 2.3$$

Above we mentioned the state transition function. If we know this function – what state follows given an action – we have a model of the world. Except for constrained experimental cases, it is unusual to have perfect knowledge of the world, but if we do, the state transition function can be used to calculate values for each state and action combination. By iterating over all different such combinations, following different policies, we can find the optimal policy, denoted π^* .

If there is no model of the world, as is most often the case in the RL problems we are discussing in this work, there are two options; model-based and model-free. Important to note here is that the kind of world model we just mentioned above is *not* the same as the model in model-based RL. Model-based RL means approximating the world model (state transition function) and using that approximation we can then apply the same methods to find V and π as if we knew the real model as per above. Model-free RL instead either approximates

V (value-based methods) and/or π (policy-based methods) directly without also approximating the world model. In both cases of RL, we thus learn values and policies through experience. The trade-off between MF and MB RL is that model-free methods are data-inefficient (much experience is needed to approximate values well) and thus learn relatively slowly, while model-based methods can learn faster but require more computational power (because they have to calculate action values for each choice) [126]. However, these are only general guidelines and the full state of affairs will depend on specific algorithms and implementations. For example, it may require a lot of computational power to provide enough experience to a model-free agent to perform well.

2.1.3.2 Three RL algorithms

The above equations and concepts are quite general. So, when learning a task trial-by-trial, we need more specific implementations. There are, as one might imagine, many different ways to do this. So, in the name of brevity we shall here present three discrete examples of model-free algorithms. They are chosen partly because they are all fairly common in the literature, as well as demonstrating how their details may impact neurobiological interpretations.

ACTOR-CRITIC is composed of an actor that selects actions based on a policy function $\pi(s)$, while the critic handles state values and uses them to give feedback on the actor's choices [258, 261]. The core part is the TD prediction error:

$$\delta_t = r(s_t) + \gamma V(s_t) - V(s_{t-1}) \quad 2.4$$

Where r is the reward of the next state, V is a value function for states and γ is the discount parameter controlling the influence of future state-values. Through experience, the critic can learn the true values of states (or at least an approximation) by minimising the prediction error and updating the value of the last visited state:

$$V(s_{t-1})_{new} = V(s_{t-1})_{old} + \alpha \delta_t \quad 2.5$$

The actor's policy function is then updated in the same way¹:

¹ Different learning rates are sometimes used for the actor and critic, see for example [55]

$$\pi(a|s_{t-1})_{new} = \pi(a|s_{t-1})_{old} + \alpha\delta_t \quad 2.6$$

The consequence being that for positive prediction errors, $\delta > 0$, the policy/actor increases its tendency to pick the chosen action in that state. For negative prediction errors, $\delta < 0$, the actor will lower its tendency to pick that action in the future.

SARSA combines state and action values into Q-values. It is called SARSA because on every update it uses the quintuple $s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1}$ to update the value for the state-action pair chosen at each time step:

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha[r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t)] \quad 2.7$$

The γ parameter here controls how much impact the values of state-action pairs in the next step will impact the new value on the current step. If the next state, s_{t+1} , happens to be the last (terminal) step, then $Q(s_{t+1}, a_{t+1})$ is defined as zero, otherwise it is selected from the currently stored Q-values based on the current policy. This makes SARSA a so called “on-policy” algorithm. It has been shown that SARSA converges to the optimal policy if all state-action pairs are visited an infinite amount of times [258].

Q-LEARNING is very similar to SARSA, but instead of choosing a specific action a_{t+1} in the γ multiplied term, the maximum value of the available actions in state s_{t+1} is used:

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha[r_t + \gamma \max_a Q(s_{t+1}, a) - Q_t(s_t, a_t)] \quad 2.8$$

Because the maximum value is picked, regardless of the current policy, this algorithm is an “off policy” TD algorithm [258]. Again, with the assumption that all state-action pairs are visited an infinite number of times, Q-learning also converges to the optimal policy. Most of the algorithms we use in later chapters will be based on Q-learning, so it is one we shall come back to. There we also discuss and demonstrate in more detail how these algorithms work.

Readers may have noticed that for neither Q-learning nor SARSA, we have specified *how* actions are selected. We come back to this question of how to follow a policy in the section on action selection below, as there are multiple options for how this is approached.

2.1.3.3 Neurobiological plausibility

Because of the division between policy and values in actor-critic, it can easier explain findings in imaging studies where multiple areas are seen to be involved in RL [124, 155]. For example, [261] shows actor-critic maps well onto ventral striatum (critic) and dorsolateral striatum (actor). The reward prediction error is shared between the actor and critic, and correlates with dopamine neuron response [28, 124, 155].

However, [191] comes to the conclusion that SARSA or Q-learning may explain dopamine response results better than actor-critic. This because the dopamine firing patterns recorded by [180] were better explained with a Q-learning model, whereas the dopamine neuron firings found by [217] fit better with SARSA. In other words, these two studies differed in whether dopamine neuron firing patterns showed evidence of “off-policy” [180] or “on-policy” [217] (Q-learning and SARSA respectively, as described above) when estimating future rewards from the current state.

Widening our perspective from specific neurobiological correlates of model-free RL algorithms, there is the question of where and if we can find evidence of model-based RL? It turns out such evidence is claimed to be “ubiquitous” [69] and, thus, there has been an increased focus among authors to study the interaction between and potential separation of MB and MF RL [52, 72, 155, 219]. We return to the topic of model-based and model-free in section 2.2 below.

2.1.4 ACTION SELECTION

The basal ganglia has long been believed to have an important role in action selection [82, 105, 209], a problem intimately tied to the coding of reward values as those are used to select optimal actions. Above we used the terminology of following policies for selecting actions. So, how do we actually follow a policy? How does an animal select between exploiting its current knowledge about state values and exploring options that may in future be better than what the current knowledge implies? This is known as the exploration/exploitation problem and is relevant for both artificial systems [259] and human learning [90].

When using RL algorithms to model behaviour, the methods of action selection are not based on as solid ground as the RL theoretical

framework/RPE hypothesis itself (but see [83]). The two most commonly used algorithms for action selection are SoftMax and epsilon greedy (ϵ -greedy) [259].

SoftMax creates a probability distribution over the available actions based on action values and uses an inverse temperature parameter to control the stochasticity of choice; lower values of this parameter make the SoftMax distribution more random or in other words decreases the chance of selecting the action with the highest value. In the limit when the inverse temperature is at its minimum (zero), then action selection is completely random: all actions have an equal probability of selection irrespective of their expected value.

There is no theoretical framework (that we know of) for how to select the temperature parameter value. In simulations it is therefore chosen so as to increase performance and in modelling animal behaviour it is fitted to best describe the data. Of course, one can rely on previous research to select appropriate values for the temperature, but as we shall see in later chapters, such values may not be reliable.

ϵ -greedy uses a fixed value between 0 and 1, say 0.2, and if a randomly generated number is lower, then a random action is chosen. Otherwise, the action with the highest value is chosen. ϵ can also be tied to time or number of trials, so that over a simulation or experiment the parameter value is lowered and thus less and less exploration happens [150].

When fitting RL models to experimental data, the arbitrary nature of these two methods can be seen as an issue. Goodness of fit measures are of course reliant on not only estimating the SoftMax or epsilon parameter themselves, but how can we be sure either of these are appropriate in the first place? Furthermore, [90] points out that neither SoftMax or epsilon takes uncertainty of the value estimate itself into account. The author therefore investigates the so-called Upper Confidence Bound algorithm (which favours exploration of actions with high uncertainty in their value estimations) and Thompson sampling (select action based on the probability that it is optimal). [90] then suggests humans use a combination of directed (bias towards information seeking) and random exploration (exploration that is not informed in any form), which is also supported by [292]. Another way to incorporate uncertainty would be to tie epsilon directly to the prediction error [267, 268].

The exploration/exploitation problem is an important one, and thus an active research area. Without a theory of an integrated way of tying exploration to the prediction error, as in [267], we will have to keep testing action selection algorithms until one sticks. Contributing to such testing is unfortunately out of scope for this thesis. Luckily, SoftMax is as close as we get to something that has stuck with researchers and it is therefore the method used in our investigations below.

2.2 MULTIPLE SYSTEMS AND HIERARCHIES

In daily speech we often speak of habits or automatic behaviour versus deliberate or intentional actions. This idea can be traced back at least as far as Aristotle [183], but potentially is as old as human thinking itself. Humans have always told stories of the duality between the rational and irrational, “human” reason overcoming animal instincts, the soul and the flesh.

In the modern western scientific tradition, the idea of two main forms of thinking – dual process theory – is commonly seen as introduced by William James in the late 19th century. James [122] called the two forms associative and true reasoning where the first is based on experience and the second a form of planning or reasoning about the future. Similar thoughts can later be found in Freud’s theories of the conscious and the unconscious, along with the Gestalt psychologists who more appropriately for our purposes proposed two distinct learning mechanisms [6]. According to Gestaltists, associative learning occurs gradually through the repeated co-occurrence of external stimuli or memories. Insight learning occurs suddenly when people discover new relationships within their prior knowledge as a result of reasoning or problem solving processes that reorganize or restructure that knowledge [6]. Later contributions by Evans [77] framed the two systems as System 1 and System 2, the latter of which being unique to humans. The most famous version of dual systems theory is probably that of Kahneman [127], framing them as the fast but habitual System 1 and the slow but flexible System 2. How these systems may interact and what neural foundation they might have is a current topic of debate [104].

One way of viewing potential interactions was presented by Lashley [144]. Lashley used a number of examples from language

understanding and production, motor generation, rhythm and sense of space to point out that there likely is some higher or more abstract structure that is imposed on lower, more practical behaviours. He emphasized that this higher structure is more than direct associative connections between those practical behaviours. This fits well with later findings in the visual system, where two main visual streams [102] are also hierarchically integrated [276]. The matter of hierarchies at play when performing a specific task is something we discuss later in this chapter.

Instrumental learning (i.e., RL for our purposes) in humans and other animals is likely to involve one or more higher level functions like working memory and executive functioning, in addition to learning incrementally from prediction errors [28, 54, 65, 69]. The different theories of dual processing briefly presented here have different levels of relevance for our work, but the point is to demonstrate that the distinction between model-free (MF) and model-based (MB) RL is related to other discussions within the cognitive sciences. The advantage of framing the discussion as one of MB and MF is, as we saw in previous sections, that we here have close connections between mathematical models and neurobiological findings. In other words, William James' and Lashley's ideas still have merit [12, 65, 73] but it is still not clear how precisely they might apply [88, 143].

In this section, we look more closely at different proposals of dual systems processing in relation to learning. Starting from work on distinguishing between MF and MB RL in behavioural and neurobiological studies, we find they are perhaps more integrated than separated. We then discuss examples of integration and/or separation of MF and MB from both computer science artificial intelligence research and from the cognitive sciences. We then consider hierarchical proposals that may connect RL to more general theories of cognition and ask what role memory and other mechanisms play. This journey will lead us to consider how and why we may want to investigate the nature of representations.

2.2.1 MODEL-FREE AND MODEL-BASED SIGNALS AND BEHAVIOUR

In the RL literature, the MF and MB systems are sometimes also called habitual and goal-directed, respectively [12, 13]. Interaction between

two decision making systems for habits and goal directed planning has long been theorized, as discussed above, but in terms of conditioning and rewards, early work by [12] suggested habits and goal-directed behaviour are underpinned by separate systems. The formalization into model-free versus model-based learning was then introduced by [65]. As briefly mentioned in the history section above, others have since confirmed that both kinds of signals exist in the dopamine system, once called the “ubiquity of model-based reinforcement learning” [69].

Evidence of for separate systems has been found in neuron recordings in monkeys performing bandit tasks [145], suggesting a role for frontal cortex in model-based predictions and the basal ganglia being primarily responsible for model-free RL. Likewise in humans performing spatial navigation encouraging model-based behaviour, fMRI results indicate the medial temporal lobe and frontal cortex for model-based planning [244], and suggest striatum as the location of values and action selection. Later research such as [273] showed that prefrontal areas of cortex also can code values of reward objects, leading the authors to suggest dorsolateral prefrontal cortex might play a role in coding higher-order aspects of the task.

Other areas such as the hippocampus also has an important role; [60] and [172] show that model-based planning behaviour is causally dependent on the hippocampus (and see below section on memory for more on the hippocampus). However, they found little evidence for model-free RL in their experiment on rats, whereas in the same task with humans, evidence for model-free RL had been found. This led the authors to suggest that perhaps model-based control is a default mode and model-free is engaged only when habits have formed, a thought supported by [54, 65]. From a slightly different viewpoint, it has been suggested that the two systems run in parallel, operating in a segregated way and then integrated in prefrontal cortex [297]. Others would argue they are more integrated than that [63, 88, 94, 97, 143].

Assuming they are somewhat segregated, there would be required some arbitration between them, as now classically shown by [65]. The same point of arbitration is further argued by Kool and colleagues [137]. They found that model-based control is engaged when greater accuracy on a task is required in order to receive greater long-term

rewards, because model-free learning is less expensive but also less accurate because of decreased flexibility. In other words, MF learning is less computationally taxing because action values are already computed, so only a comparison is needed to find the best action, while MB learning has to compute the values each time [65, 97]. An important difference between these proposals is that [137] considers a cost-benefit evaluation function to select between the systems, while [65] base the evaluation on uncertainties. The latter approach is also explored in [54] but they used direct value estimates instead of uncertainty.

This confused state of affairs has led some to propose that we need to rethink what model-based actually means [143], others to suggest that dopamine may help encode more than just prediction error [100], and yet others suggest alternate accounts of the MB/MF dichotomy may explain results better [88, 175]. The rest of this chapter section takes a closer look at these and other proposals for how to resolve the dopamine and MB/MF conundrum.

2.2.2 DUAL SYSTEMS IN ARTIFICIAL INTELLIGENCE

In 2016, the company Deepmind held an exhibition match between their Go² playing system AlphaGo and multiple world champion Lee Sedol [242]. AlphaGo won 4-1 and this event is already of historical significance on par with Garry Kasparov's loss at chess to Deep Blue in 1997.

What is significant about AlphaGo is that it successfully combines MF and MB in a way that can be applied to such a difficult problem as Go. The MF part of this system is an RL algorithm and the MB a tree based search algorithm called Monte Carlo Tree Search [242]. The AlphaGo system was trained on a dataset of Go games played by humans. It was improved upon a year later to learn only from self-play and thus dubbed AlphaGo Zero [243]. Another year later and the researchers managed to use the same principles to create AlphaZero [241], a system with less pre-defined information specific to Go that could therefore play not only Go, but also Chess and Shogi (Japanese version of Chess).

² Go is a board game commonly played on a 19x19 grid, where two players take turns placing stones. The goal is to conquer as much territory as possible. Depending on the rules, the number of possible positions may be as high as 10^{170}

Another interesting system is that of “world models” [106]. Here, a recurrent neural network is trained to learn a model of the agent’s world which is then used to teach a controller how to navigate that world model. During play of a car racing game, the model’s predictions of sensory inputs are used together with the actual sensory inputs to inform the controller of the best action to take. This implementation is not so completely a dual system as AlphaGo but leans rather more towards model-based learning. What is really interesting here though is that the trained world model can be used to “dream” the learned environment and actually improve the policy of the controller offline.

The combination of online and offline learning is also the idea behind Dyna, an early combination of MB and MF [257]. Its policy and value functions are updated both directly from experience in an MF way, while the same experiences are used to update a model. From this model simulated experiences are generated and used to update the same behaviour policy as the MF system updates. Variations of Dyna have been used successfully for example to play Atari games [129] and in proposals for how separate MB and MF systems can collaborate in simulations [219].

[2.2.3 INTEGRATIONS OF MODEL-BASED AND MODEL-FREE](#)

Langdon and colleagues [143] present multiple findings that indicate dopamine neurons respond to other aspects than just scalar reward value, for example reward identity (juice instead of a puff of air, say). This suggests that dopamine neurons have access to richer representations of states than traditionally thought, perhaps as some sort of more general prediction error for states. The authors therefore propose that model free predictions of scalar reward values are multiplexed (integrated) with model-based vectors containing information about multiple reward dimensions such as type and timing, as well as other relevant aspects of the state, perhaps even inferred information not directly observed.

This line of reasoning is similar to the “sensory prediction error” hypothesis of dopamine introduced by Gardner and colleagues [88]. They adapt the “successor representation” (SR) algorithm – a middle ground between MF and MB that will be discussed in detail in chapter five (and also in the next section on state representations). SR may be adapted to predict not just rewards but also states, and such an

account they claim might explain many of the controversies of both the model-based versus model-free debate, as well as details about dopamine functioning itself. Though not arguing for sensory prediction errors per se, multiple additional authors have proposed SR as a way of unifying the debate on model-free and model-based [89, 161, 175, 177, 219].

Other promising attempts have been made towards integrating previous research, coming from the computer science “learning to learn” or “meta learning” field which has origins going back to the late 1940s with work by Harlow [110]. Briefly, Harlow showed how monkeys trained on a two-object discrimination task (essentially a two-armed bandit where one arm always is rewarded and the other never, see chapter three) initially required multiple trials to consistently pick the correct object. But after enough experience (multiple blocks of different objects) they had learned they could find the correct choice based on only the feedback in the first trial. They had learned to form “learning sets” as Harlow put it.

Recently, recurrent neural networks have been used to model parts of the brain known to be involved in learning – the basal ganglia and frontal cortex – by using a reinforcement learning algorithm to train new learning algorithms [27, 71, 284, 302]. In other words, a task-specific RL system emerges from the dynamics of the recurrent network.

Specifically, [284] model PFC, basal ganglia and thalamic nuclei as a recurrent neural network. This is trained with DA phasic signals as reward prediction errors and using state, reward and action as inputs, on six different categories of tasks. Results indicate the recurrent network exhibits an emergent RL system that can handle six different experimental tasks in line with behavioural findings and neuronal recordings from humans and animals in previous research. Interestingly, although the system is trained separately for each type of task, the emergent RL system learns to handle variations of the same type of task. For example, if the task type is two-armed bandits, the system will slowly learn this type of task when training starts. But after a point it will “know” the task and when encountering a new two-armed bandit, it will learn the optimal choice in just a few trials. In other words, it has “learned to learn”, just like Harlow’s monkeys.

This combination of “fast and slow” reinforcement learning [27, 71] therefore provides a possible explanation of previous inconsistencies in the model-based/free dual systems view, where a hierarchy emerges dynamically through learning.

2.2.4 HIERARCHIES

Considering the curious findings on MB and MF RL, together with the research presented above, it could be the case that what we are seeing is the effects the brain organisation into a hierarchical system. If we were to assume the simplest case of two hierarchical layers, habits or model-free functionality would be on the lower level and planning or higher order reasoning, as in model-based functionality would be on the upper level. If we further suppose these levels as the basal ganglia/dopaminergic midbrain projection system and frontal cortex, respectively, then what we see in the “puzzling” model-based responses in the basal ganglia might actually be traces of the signals coming from the upper level.

A classic example of hierarchy is that of Brooks’ subsumption architecture for robots [32]. Behavioural modules are stacked on top of each other, from low-level behaviours like “avoid obstacles” and “wander aimlessly” to top-level ones like “formulate and execute plans” and “reason about objects in the world”. All modules run in parallel and separately receive sensory input and output behaviour. The core idea is that lower-level modules are unaware of higher levels, but higher levels can control lower-level modules if needed. The effect of this hierarchy is that if “avoid obstacles” is the lowest level 0, and the “wander aimlessly” level 1 activates, then the robot will indeed wander around aimlessly. It will do so, while simultaneously avoiding obstacles, because level 0 activates as needed – without level 1 having any need to be “aware” of this functionality.

Hierarchical Reinforcement Learning (HRL) is an umbrella term for organising the learning of actions [28, 256] and/or states [53, 73] in hierarchies. One may frame these approaches as attempts to solve the “curse of dimensionality” (or scaling problem) – i.e., that states and actions can be very high dimensional and therefore exponentially difficult to solve [28]. We come back to the question of high dimensionality in the next section about state representations.

For now, HRL for actions can be done by abstracting action sequences over time into “options” [256]. In other words, the number of decision points for an agent is decreased, since an option can take the agent through many actions in one go. For example, “make coffee” or “make tea” may be options, each choice consisting of multiple actions to get your preferred beverage. Work by [28] shows how such options can improve the performance of an agent navigating a maze, as long as the options are chosen appropriately. This highlights the problem of how to learn the options; should they be built-in or learned dynamically? For a software system aimed at solving a particular problem, building them in may suffice, but if trying to understand and explain the brain’s learning system we would be very likely to need a mechanism to learn them dynamically. Recent work [161] has proposed the successor representation may be used to learn action options more dynamically (and see below on episodic RL for a possible mechanism, and further proposals in the next section about state representations).

The scaling problem can also be solved by abstracting over states instead of actions [154], a process that is likely supported by (pre)frontal cortex [218]. One good example is work by Collins and colleagues [49, 53, 73] where states are organised hierarchically as task sets. We will come back to Collins’ HRL in the next section on state representations and it will also be described in detail in chapter five.

The two ways of abstracting over time - actions and states - may seem disparate, but if we frame this in the terminology of model-based RL, it is likely that task structure - the world model - contains actions as well as states and the two are thus closely connected. This idea is supported by [28] who state that temporal abstraction can be incorporated in model-based algorithms if there is an option model where “primitive” actions (the basic actions that are put into sequences to create options) can be skipped. Furthermore, they also state that execution of subtasks is highly context sensitive. For example, if one picks up a pen with eraser on one end then depending on the intent (i.e., context/task structure), the available actions are different. Grab the pen as usual if the intent is to write or grab it closer to the eraser end if the intent is to erase. In other words, task structure and actions are closely interlinked.

Huys et al. [120] build on this and show how subjects performing a planning task reduce computational costs by “combining partial searches with greedy local steps to solve subtasks”. In other words, their subjects exploit the structure of the task to create subgoals that allow them to improve performance. For example, in a task where subjects had to choose actions in a certain sequence to receive the highest reward (visualised as a search tree), it was found that subjects had no problem with shorter sequences. But with longer sequences subjects started to group sequences together, creating subgoals that effectively left parts of the search tree unexplored and some optimal paths were missed.

That task structure and actions are interlinked is quite obvious if seen from the perspective of embodied cognition [45, 46, 202]. There are many interpretations of what this term means [290], but common for them all is that cognition does not simply happen exclusively in the brain. Cognition is a process that happens through the interaction of brain, body and environment – including other animals and objects. For example, Kirsch & Maglio [134] showed how good Tetris players do not rotate pieces in their mind to see where they will fit. It is simply too slow. Instead, players rotate pieces in the game – using their hands to press buttons – and use visual matching to identify placement positions. The cognition happens in the linked system that is body, brain and video game.

The principle of embodied cognition has been an active theory within robotics for many years, in fact the system described above by Brooks is now seen as a foundational example [159]. Embodied cognition is also an important part of theories framing cognition as one of “predictive processing”, where the principle of prediction errors is seen as a core part of interacting with the world [10, 45, 84, 115]. Animals actively explore the world in order to minimise prediction errors, a process formalised as the free energy principle [84, 85, 159], from which RL itself may be derived [83, 85].

Through the lens of predictive processing, neural structure, function and behaviour arise from a hierarchy of mechanisms where incoming sensory information is compared to predictions on every level [10, 115]. For example, if you pick up your cup to take a sip of coffee and you taste tea, you are surprised, given you expected coffee. Just like RL algorithms learn from reward prediction errors, your

surprise was also a prediction error. If you both expected tea and tasted tea, there is no prediction error, and you would go about your business without need for explicit thoughts about your drink.

[2.2.5 MEMORY SYSTEMS AND BEYOND](#)

In Minsky's Society of Mind [173], it is argued that the brain is made up of a vast number of small functional entities called agents. Thinking and behaviour emerges from the interaction between these agents and therefore does not rely on any single principle. This view is directly opposed to the one taken by previously mentioned work such as AlphaGo [242] and meta-RL [284]. Especially in the latter work the working hypothesis is the idea that maximizing reward can explain all behaviour. Sutton [262] calls it the "reward hypothesis" and describes it thus: "That all of what we mean by goals and purposes can be well thought of as maximization of the expected value of the cumulative sum of a received scalar signal (reward)." Since we are framing this work in terms of reinforcement learning, we lean more towards the reward hypothesis than society of mind, but there is value in, and perhaps necessity, in considering more systems than just MB and MF.

We have briefly touched on this idea previously, mostly when discussing brain areas such as the hippocampus being involved in RL. The hippocampus (and closely connected structures in the medial temporal lobes) since it is widely considered one of the main brain areas subserving episodic and spatial memory [24, 37, 181]. We have also discussed how the prefrontal cortex is involved in model based learning, an area that is also important for working memory [15, 16].

The hippocampus has been suggested as a candidate for shaping the world models that are needed for MB RL, partly due to the strong evidence of hippocampal involvement in spatial navigation [181]. So-called place cells code for an animal's position within a spatial map that corresponds to the current environment and there are some indications that these place cells show predictive signalling by simulating paths towards goals [203]. Place cells in the hippocampus receive signals from grid cells found in entorhinal cortex, which are thought to represent more general maps of the environment from which the place cells can construct a map that is more task-specific both in time and space [181, 272]. Intriguingly, recent studies on the successor representation (SR) have found potential correlates

between aspects of the SR and place and grid cells [175, 177, 248, 288]. We come back to SR in the below section on state representations, as well as in chapter five.

Regardless of whether SR is the most appropriate model to account for these findings, it seems likely the hippocampus is important for the formation and/or storage of models. Adding to this idea of hippocampus as a “model storage” region is the claim linking it to more general prospective imaging of the future [223] and more abstract relationships that seem similar to state transition functions [97, 200]. It may thus be the case that hippocampus is not just storing models but also involved in generating predictions based on them.

This may lead us to ask, what is the difference between models and memory? In our view, perhaps the simplest way to differentiate them is that a model is a summary of experience whereas (episodic) memory consists of a record of the experiences themselves. Therefore, it is likely the two are closely intertwined since some form of memory is needed to store learned models, and models can in turn be an efficient way to utilize limited storage capacity [301]. In fact, it seems this is how human memory works; it does not generally have precise recordings of past experiences but, rather, recalling events is made in a generative or constructive way [59, 222, 223]. For example, this is thought to be why eye-witness accounts are prone to influence by interviewers [87]. Gershman & Daw [97] notes that the replay of episodic memories may be influenced by a combination of cooperation and competition in the interaction between striatum (a main site for MF RL) and hippocampus, due to their functional connectivity.

One important type of memory is working memory (WM), which can briefly be described as a form of active short-term memory, able to both store and manipulate information [301]. WM is associated with activity in prefrontal cortex and multiple studies have shown how there is a close connection between WM and RL, and in some cases one can be mistaken for the other [51, 54, 57, 168, 301].

An example of the interaction between RL and WM is that of [54] where the authors showed that an RL+WM model provided better fit to participant behaviour than RL alone. The task presented participants with trial blocks containing 2-6 different stimuli (pictures from a certain category like fruits, places, sports, etc.) and in each trial

one of these stimuli was presented. Participants had to learn which of three possible responses (actions) led to reward. Assignment of the correct response to each stimulus was random so the correct response could, in the two stimuli case for example, be the same for both stimuli. In this way, the load on WM was varied between blocks, since remembering correct responses for two stimuli is easier than for six stimuli, especially as in the latter case it may be many trials before seeing the same stimuli again.

The behavioural results showed that learning was slower in problems with greater load (blocks with more stimuli) but there were minimal differences asymptotically, meaning that, as training progressed, performance reached similar levels regardless of load. Similarly, for delay since a stimulus was last presented, longer delay initially degraded performance but this effect also disappeared over learning. The authors interpreted this thus: with increasing experience the RL system accumulates sufficient evidence and eventually supersedes the WM system.

The RL+WM model used in [54] is fairly simple and does not identify what specific events are stored in memory. But a more elaborate proposal of memory integration with the RL system called episodic RL [97] (see [27] for a machine learning perspective) does keep track of specific events. This theory builds on a method called episodic control, introduced by [151] and uses episodic memories to construct nonparametric approximations of the state or state-action value function [97]. More specifically, both MB and MF RL uses a parametric approach; MF stores action values and MB stores model parameters to generate trajectories, but once these parameters have been estimated, the raw data - experiences - are discarded. Episodic RL keeps the entire set of experiences in memory and can compare the current situation to previous ones. Such comparisons thus depend on how states are represented, in order to compute some similarity measure. The comparisons can additionally enable episodic RL to handle new situations by finding a previous experience that is a good combination of similarity to the current situation and also is associated with reward. In artificial intelligence research, episodic RL has been successfully used to increase data efficiency for playing Atari games [27].

Saving experiences is in some way similar to experience replay [157] which was an important component in the success of the Atari playing system DQN [174]. But where experience replay usually takes place offline between training sessions and it randomly samples stored sequences of state-action-reward (while “sleeping” if you will), episodic RL works online. So, when a familiar state is encountered, an action is suggested by the episodic system based on previous trajectories from that state. This is similar to the action options hierarchical RL (HRL) [28] mentioned earlier and, indeed, [97] suggested episodic RL could be a way to create the options or action sequences proposed in HRL. Since memories of experienced state trajectories would work similarly to MB planning, for some tasks it may be the case that what has previously been classified as MB is actually episodic influences. Furthermore, and as discussed earlier, this would of course complicate the picture as episodic RL would also have to solve the question of whether and when it communicates, or integrates, with the MF and MB systems.

One way for episodic RL to do so could be based on findings that hippocampal involvement commonly dominates early in training and gradually gives way to striatal systems [97]. This fits with previously mentioned results by [54], where WM dominated early and MF took over after training. This also ties back to arbitration between these systems as in [214]; thus, any meta controller would then also have to include the episodic system in its arbitration. In fact, Ritter and colleagues [215] introduce a system combining the above-mentioned learning-to-learn (or meta-RL) principle with episodic memory. This enabled the system to reactivate behaviour from memory, if the task encountered was found similar to a previous experience.

This brings us to important questions. To compare the current situation – the state – to previously encountered ones in memory, we need to compute some form of similarity measure. The success of artificial intelligence agents using episodic RL thus critically depends on defining appropriate state representations [27, 215]. As mentioned above, this works for “simple” problems like playing Atari games. But how would this scale to high-dimensional, continuous, partially observable state spaces where data are sparse, and observations have dependencies over long temporal distances [241, 256]?

In real world situations, observations are extremely rich in informational content and thus suffer from the so-called curse of dimensionality (high dimensional state spaces) – how do we focus on what is important? Observations are also continuous, meaning animals need some mechanism to partition the world into suitable states – in both time and space. At the same time, these observations do not contain *enough* information because some information is hidden and needs to be inferred – like understanding someone's feelings. Further, some information is only partially observable, requiring past information to be maintained in order to make correct choices, which challenges the memory-less Markov property that RL relies on – that all necessary information is contained in the current state.

In the next section we shall see how this leads us to consider not only memory but also attention and inference – beliefs about hidden information.

2.2.6 SUMMARY AND CONSOLIDATION

Instrumental learning in humans and other animals have traditionally been a story of model-based (MB) and model-free (MF) learning and how these may interact. There are differences in approach to this problem, with some integrating MF and MB [242, 257], some looking at how states can be more fully represented [88, 143] and others embracing the hierarchical organisation, leading to model-based behaviour emerging from model-free interaction with the world [284, 302].

There is also the matter of computational or cognitive cost associated with arbitrating or switching between these systems [137]. The idea of using a less costly system (MF) for easier tasks and deploying the costlier (MB) only when required for more difficult tasks may make intuitive sense. However, several studies have shown that MB (or memory-based processing) dominates early in training and gradually gives way to MF [54, 65, 94], coinciding with hippocampal involvement early and striatum getting more involved later in training [283, 301]. This is inverse of the approach taken by [284, 302] where the MF system is used to train the MB system. This latter viewpoint is against the common folk understanding of habits, which is that they need time to form. One way of looking at it is through the hierarchical lens, where MB lies above MF in the

hierarchy. MF would then be used to train MB, as in [284, 302] and as that model is improved, the action values are made available for the MF system to use directly, accounting for the potential discrepancy.

Dual systems make intuitive sense, exemplified above with multiple examples of such dualities from different perspectives; but maybe it is not so easy. Perhaps, it is the case that multiple systems are interconnected through hierarchies that build more and more complex structures and representations. We can see such structure in the visual system, where low level features are encoded from sensory information in the occipital cortex and higher level information builds in hierarchical layers, splitting into two main visual streams [102, 276]. Hierarchical convolutional neural network models have proved to predict signals of the (ventral) visual system both structurally in the layout of layers, and in neuron population activity for each such layer [212, 298]. Similar findings have been shown for auditory cortex [132].

It would make sense, then, if models of the world work in a similar fashion – low level features building into higher-order structure. This line of reasoning naturally leads us to the predictive processing hypothesis [46, 115], where top-down predictions from a world model are compared with bottom-up incoming sensory information. On each level of the hierarchy, prediction errors arise, informing the generative model above how to adjust its predictions for the future.

This view would appear to fit with the established framework of prediction errors driving learning in RL, and provides a way to resolve the recent confusion of finding evidence of both model-free and model-based predictions at multiple levels of the dopaminergic projection system [56, 69]. There is also synergy with proposals that add richer state representations to RL [143] and with the argument for a generalized “sensory prediction error” hypothesis of dopamine [88]. Adding to this is the claim that RL “naturally falls out of” the free energy equations of statistical physics [264].

Speculating further, for an animal undertaking a task, parallel systems like working memory would enable active manipulation of the current model under consideration, supported by episodic memory to compare and retrieve models from previous experience. For the specific task, generated predictions for the entire mental model of the task structure would be tested against the consequences

of actions. When the predictions fit, perhaps one can view the absence of prediction errors, when one has found the right mental model, as what is commonly called the aha experience [265].

In short, regardless of how these systems interact, it looks clear that there is more to this story than just a tale of two systems. To us, it seems more likely the story is one of hierarchies where the RL principle of prediction errors is ubiquitous. Our viewpoint, going forward, will be one where model-free RL is supported by higher-order system(s) providing RL with useful state coding information. So, the question then is, how do animals find and create states, to combine into a representation of task structure and/or world models from their chaotic sensory observations?

2.3 REPRESENTING THE WORLD AS STATES AND TASKS

Hoffman [113] describes how the Australian Jewel beetle almost went extinct. The evolutionary process has made the males of the Australian Jewel beetle very proficient in finding mates; big brown objects with pimples on their butts. This worked great for millions of years until members of the *Homo* species came along and dumped beer bottles into the beetle habitats. These beer bottles happened to be big, brown and have pimply bottoms – such perfect specimens of beetle beauty that the male beetles found them preferable to the real thing. Those males that would not die of starvation or exhaustion, were eaten alive by opportunistic ants. In the end, the beetles were saved when beer companies changed the design of their bottles, removing the pimples from the bottoms. Apparently, this was a more efficient solution than making *Homo sapiens* stop polluting the environment.

This story is interesting for a number of reasons. First, it tells us that vision – and other senses – are interfaces that do not necessarily reflect the world objectively, because they are optimized for utility [113, 114, 136]. Hoffman compares it to computer user interfaces, like that of a mobile phone. We do not need to know how the hardware works, and it would be inconvenient to manually shuffle electrons around to show cat pictures. The icons, glyphs and text on the screen comprise a *useful* interface.

Some would argue “perception as an interface” is a problem for the cognitive sciences in general [80], as the research field(s) rely on

the assumption of animal and human perception as a view of objective reality. Instead, reality does not actually exist and is continuously constructed by the senses interacting with brains and bodies.

The question of whether objective reality exists is slightly out of scope for this thesis. But similar questions arise from the view of predictive processing, where top-down predictions receive feedback from bottom-up sensing. Where in this hierarchy does our experience lie? If it is anywhere near the “top” then whatever we experience is mainly a prediction, not objective reality³.

The predictive nature of our minds is often exploited by magicians [244] who also rely on phenomena like inattentional blindness [245]. Even more fitting for our investigations of state representations is perhaps boundary extension [221], the tendency to recall information that was not present in a previously seen photograph. The reconstructive nature of memory [222] fills in the blanks of partly seen objects.

The second reason for the beetle story being of interest is how it nicely exemplifies the difficulty of categorisation. As touched upon in the previous section, and will be discussed further shortly, animals have to generalise between states that look different but are actually similar. While at the same time they must differentiate between states that look the same but are actually different. This leads to the third reason the beetle story is interesting.

Obviously, the beetles have strong innate priors for the generalisation/differentiation process, whereas humans are more flexible. There is considerable debate as to how much of human behaviour is innate [75, 164, 246, 247] and it is not our focus, but there is no question today that behaviour is a construct of nature *and* nurture. Framed as learning, one could see innate behaviour as adaptation over evolutionary time.

Humans do have innate priors, like the ability to recognise faces at birth [209] but we can also learn representations to perform activities like dancing, climbing, rocket science, painting or playing games like chess. In fact, expert chess players have learned to see the board in a way that they essentially only see legal moves and positions, unlike

³ Of course, if we assume a materialistic world view, then our brains are part of reality, and thus mental predictions are by definition also part of objective reality.

beginners who have to exert effort to filter illegal ones [210]. Across the animal kingdom we can see examples of different degrees of innate abilities, for example mountain goats being able to climb steep cliffs only hours after birth [164].

In artificial intelligence research, a focus has been on so called end-to-end learning [141, 174], meaning systems that learn to, for example, play games from scratch where the vision system is being trained simultaneously as action selection. In a way, then, that approach is more akin to training of a new-born rather than an adult. Yet, since the same studies often have to pre-define what a basic state consists of (for example a still frame of pixels), the approach is somewhat similar to experiments with adults where states are often pre-defined as single trials consisting of stimulus and action options. Artificial intelligence research has, however, recently started to take innateness into account, through an idea often termed inductive biases [52, 58, 188].

So, what does all this mean for our question of how animals create states? There is an obvious difference between the evolved hard code for male beetles' perception of objects that look like female beetles and the weaknesses in humans' predictive inference perception, but it is fascinating that we understand little of *how* they are different. We also need to disentangle states and task structure, which compels us to look into (1) what are states and how are they created; (2) how states are informed by and combined into task structure; and (3) how task structure is summarized into a model, and how that model interacts with other processes like learning, attention and memory.

[2.3.1 FROM SINGLE STATES TO TASK SEQUENCES](#)

States are one of the core parts of RL, as we saw above when describing Markov Decision Processes (MDPs). In this mathematical formulation, states are inputs received from – and observed in – the environment. Traditionally, states, actions and reward functions have been taken for granted in the RL literature [58, 189], meaning they are commonly pre-defined as required by the experimental task.

In such experimental tasks, a state is thus the idealized world participants see on the computer monitor and in machine learning the state is most often a vector of pixels. But as we mentioned above, in real world situations, observations are high-dimensional and continuous. This means, as already noted, there is need to both

generalize between states that look different but are actually similar while at the same time differentiate between states that look the same but are actually different. Perhaps it is counter-intuitive, but real-world observations also do not have enough information to adhere to the Markov property⁴ since many situations require memory of previous events or inference of hidden causes. Furthermore, most RL algorithms converge for certain only when all states and actions have been visited an infinite number of times. Biological organisms cannot possibly try every possible sequence of actions.

We will address the just mentioned points in turn, as much as is possible, as they are inherently connected. It is understandable that states have been taken for granted, given the incredible complexity involved in how additional systems would interact. But in the past few years, as the RL story has solidified, the field has started to move towards investigating the matter of states and task structure, with task structure standing in for models of the world [52, 58, 189].

Even disentangling the difference and relation between states and task structure is not straightforward. For example, Wilson and colleagues [293] define a state as “an abstract representation of the current location in a task”. Unless speaking of a specific task, and thereby coming back to the issue of pre-defining states, we can offer no further clarity to their definition. State and task structure are certainly intertwined, as what constitutes a single state may need exploration of the environment before solidifying, meaning in experimental terms being informed of states by the structure of the task. This is perhaps similar to “chunking” [171] in order to remember longer sequences. For example, remembering a nine-digit phone number is easier if one memorizes them in groups of three. Indeed there are such investigations related to learning, [101] provides several examples such as Chess masters only remembering useful board configurations, and the creation of subgoals by [188] mentioned above in the section on hierarchies. Lashley [144] phrased it as “all skilled acts seem to involve the same problems of serial ordering”.

⁴ Future states depend only on the current state and action, not the past states.
See section 2.1.3.1

2.3.2 STATE REPRESENTATION IN MACHINE LEARNING

The problem of serial ordering – task structure – is sometimes also called representation learning [189]. Incidentally, the same term was long used in the machine learning community⁵, for what is today called deep learning. Although coming from different viewpoints, the combination of RL and deep learning have proved to work surprisingly well when scaled up, being able to play board games [241] and video games [4, 196] at expert human level and beyond. But such systems rely on massive amounts of data to learn useful policies and therefore consume massive amounts of computing power [297].

It is not yet clear how informative such systems will prove to be with regards to underlying mechanisms of animal brains [183, 220] (although deep learning shows promise predicting signals of the visual system [298]). This is especially so seeing how the complexity of deep learning systems may itself need interpretation [184, 221]. Furthermore, the above-mentioned systems also have problems handling new tasks and/or new situations [189].

However, an interesting new kind of representational structure called “transformer” [277] shows promise to perhaps handle new situations better. Briefly, transformers are a method for modelling sequential input combined with “self-attention” that provides importance weights to each part of the sequence and outputs a new sequence (a.k.a. sequence-to-sequence) [158]. The original paper [277] applied transformers to language translation and the method is now the standard in the natural language processing field, with recent models like GPT-3 [33] being so powerful it can generate text which is difficult to distinguish from that of humans. Even more intriguing is that transformers can be used not only for language, but also vision [192], music [193], chess [193], and mathematics [194]. Most relevant for our purposes is the decision transformer [42], which applies the method for RL problems by using state-action-reward sequences as input and outputs a sequence of optimal actions. So far, this only works off-line with an existing dataset of experiences, but research is ongoing to adapt it for on-line use [207].

Although inspired by the cognitive concept of attention, transformers are more like the psychological concept of priming

⁵ One of the most prestigious conferences in the machine learning field is ICLR – the International Conference on Learning Representations.

[158], influencing subsequent attention to stimuli. However, cognitive attention is an important process in how humans create states from high-dimensional input.

2.3.3 ATTENTION TO DETAILS

Attention in psychological terms is a complicated topic in itself [158], but a good example of the kind of attention we describe below, selective attention, is the “cocktail party effect”. Even in a crowded, noisy room we are able to focus on a specific conversation (at least to a certain point).

In a series of studies [152, 190, 294], Niv and colleagues show how such attention guides (state) representation learning, by finding task relevant features. They used the dimensions task, in which participants are shown three different stimuli on screen and have to pick one. The stimuli varied across three features (shape, colour and pattern) with three variations for each feature, for example one stimulus could be a circle with red borders and dots inside of it. The most rewarding feature was decided based on a single feature, and thus subjects had to find this relevant feature by trial-and-error, while simultaneously learning the reward values across all stimuli. Computational models that included parameters for biasing state values with attention proved a better fit for participant behaviour in both the just described version of the task [190, 294] and one where features were replaced by object categories (faces, tools, buildings) [152]. In other words, attentional mechanisms influenced what feature participants were focused on.

Intriguingly, the magnitude and focus of the attentional bias was itself influenced by the learned values and their prediction errors [152, 190]. This indicates that the state representation develops dynamically during learning in collaboration with RL mechanisms, concurring with results from other authors [39, 78].

Neuroimaging in the above studies showed correlations with prefrontal cortex activations for the attentional parts of the model when using the dimensions task. Based on these results, [206] propose a model where hypotheses about task structure direct top-down attention to task-relevant features. These features compose the state space over which RL learns values, and those values are used in the process of deciding which hypotheses to consider. It has been suggested that working memory, due to its limited capacity, helps to

focus attention and learning in this process [235]. Perhaps, under this viewpoint, the role of working memory is that it further helps to narrow down the number of hypotheses under active consideration.

So, humans can find states through the interaction of task structure and attention. In other words, we overcome the “curse of dimensionality” by iteratively finding the most relevant features for the current task. But we need to be careful, so we do not overgeneralise across features and run into the beetle problem. We also have the need to differentiate. This functionality is tightly coupled with hidden info, as we shall see.

2.3.4 CLUSTERING STATES INTO BELIEFS

Both generalisation and differentiation between states rely on mechanisms to compare the existing observation with memory traces [50, 189], a process highly related to category learning [236]. In RL-related studies this process also additionally involves inferring hidden causes and potential relations between such causes, in order to find appropriate states.

For example, [99] demonstrates how participants group stimuli based on similarity. Participants were asked to provide an estimate of how many circles they saw on the screen. The circles shown on a specific trial were all either blue or green, but the colours varied throughout the task. In one experimental condition, the number of circles shown were drawn from two normal distributions with means 65 and 35, for blue and green circles respectively. In a second condition, the distributions instead had means 65 and 55, for blue and green circles respectively. The average estimate for each colour was then calculated for each condition. The results showed that in the first condition, participants’ estimate for the blue circles was close to 65, the true mean. But in the second condition, participants’ estimate for blue circles averaged closer to 60, which is the combined mean of the blue (mean 65) and green (mean 55) distributions in the second condition. The authors therefore draw the conclusion that in the first conditions, participants were able to separate the two distributions into separate states, because the difference in means was large enough. But in the second condition, the means between the blue and green circle distributions were so close that participants clustered blue and green into a combined state.

Another study, [93] shows how state inference can explain the phenomena of spontaneous recovery in fear extinction. More specifically, when rats are trained to receive shock after a bell sound they will eventually display fear when hearing only the bell. To extinguish the behaviour, the bell sound is followed by nothing for a number of trials. In rats where behaviour is extinguished in that way, the fear behaviour may in many cases reappear after some time has passed. However, if using gradual extinction, meaning the shock following the bell happens less and less often (rather than going from continuous reinforcement to zero), then there is a much lower chance of the behaviour reappearing.

This behaviour can also be explained by the concept of states, where in the regular extinction, the rat will infer that bell-no shock is a new type of state. Spontaneous recovery thus happens if the rat believes it is back to the bell-shock state (which might occur after a delay). But with gradual extinction, the rat infers only one state. In other words, for both humans and rats in the extinction studies above, there is a battle between learning and memory, where the first new kind of trial can either be put in the same cluster (RL) or a new cluster (structural learning).

This clustering process can more formally be described as latent cause models [53, 91]. By associating multiple specific states, or trials, to either the same latent cause or separate ones the animal is able to generalise or differentiate. The state space can thus be built up iteratively, often modelled with Bayesian non-parametric models [91]. Despite the name, these models do have parameters, but they allow for potentially infinite number of parameters added dynamically through two generative processes⁶ called Chinese restaurant (single cause for each observation) or Indian buffet (multiple causes may generate an observation).

When the state space is built up, there is still the question of where in the state space the animal might be. There is bound to be some uncertainty involved in this process, so we can assign a probability

⁶ The processes have these names through analogy. In the Chinese restaurant, tables are latent causes and states are guests arriving. Each guest is assigned a single table probabilistically, as in a single cause for each state. In the Indian buffet, latent causes are different dishes. Here each arriving guest can sample multiple dishes to create a combined meal (state with multiple latent causes).

distribution over the state space. This is commonly referred to as belief states [9, 100] in the neuroscience literature and partially observable MDPs (POMDPs) in computer science [205]. Sometimes Hidden Markov Models (HMM), closely related to POMDPs, may also be used. For example, if task representations are known in advance, [109] show how action values are updated all at once when the task structure changes, which is better explained by modelling participants' state beliefs as an HMM, rather than RL which updates only the value for the action that was actually chosen.

We can now reconnect these theories on how states and state spaces are formed with the attentional mechanisms noted earlier. It may be the case that multiple sets of Bayesian belief states – each set representing a task structure – are considered as hypotheses of task structure [206]. This is similar to hierarchical theories on task set selection [53, 73], where the task set selection process has neurobiological correlates in prefrontal cortex [49, 70].

More specifically, work by [74, 226, 227, 292, 301] found that a full posterior distribution over latent causes better explained orbitofrontal cortex (OFC) activity than other models used. The authors drew on theories of episodic memory implying that memories are organized according to inferred schemas that specify situations and store previously learned relationships. Schemas thus require inference about the underlying situation or latent cause that generates observations, similar to states in RL [292].

2.3.5 COGNITIVE MAPS

OFC indeed seems to be critical to representation of state spaces [74, 227, 228, 293, 302], with Wilson and colleagues [293] calling OFC a “cognitive map of task space”. They show how OFC, due to its connections to multiple brain areas, is unique in its ability to disambiguate task states that are perceptually similar but conceptually different. OFC can do this, for example, by using information from working memory.

By replicating behavioural results from four kinds of experiments (reversal learning, delayed alternation, extinction and devaluation), Wilson and colleagues [293] show that behavioural consequences of OFC lesions can be explained by impairment in the state space underlying performance on the task. By using reversal learning in a two-armed bandit task (see chapter 4 for more information on

reversal learning in two-armed bandits) as an example, their model can be explained by lesioned animals always being in one state that has two actions. When the reversal happens, the lesioned animal has to update both action values step by step to their new values. Healthy animals instead have two different states, each with two actions. So, when the reversal happens, these healthy animals switch state and can more quickly reach the correct action values (assuming action values are initialized to zero for example). This model generalises to probabilistic reversal learning tasks, and explains extinction and delayed alternation where animals need to integrate outcomes from multiple trials to infer what state or context they are in [293]. Furthermore, they show how spontaneous recovery, which we discussed above, is also affected in OFC lesioned animals. With a one state model, for example, the original association is actually erased during the extinction phase; but with two states, only one of them is affected. So, with passage of time the healthy animal becomes unsure if it is in state one or two, allowing for spontaneous recovery.

Niv [189] proposes that OFC works as an abstract link to representations in other areas, providing a route to the integration of representations. Many of the mechanisms mentioned so far rely on memory in some form, and the concept of cognitive maps especially, are traditionally associated with hippocampus. Indeed, the hippocampus has been suggested to play a role in the process of clustering [189] mentioned above. Additionally, the hippocampus looks to be important for cognitive maps of state space in both spatial and non-spatial [229, 288, 303] tasks, as well as for state transitions [175, 227]. In short, it seems that Tolman [269] was right; animals do have internal models of the world – “cognitive maps”.

The hippocampus has long been implied in memory formation and retrieval, as in the famous case of patient H.M. who due to epilepsy had most of their hippocampi and surrounding areas removed bilaterally. H.M. was subsequently unable to form new memories [181]. Other evidence includes the observations that London taxi drivers, who undergo extensive learning of all of London’s streets, differ in hippocampal structure [163] from controls. Furthermore, the hippocampus is important for imagining the future [181].

An important finding is that of place cells, which are hippocampal neurons that fire when an animal is in specific locations; for example,

within a maze [181]. Place cells not only represent the animal's current location, but also those locations it has visited in the past. Place cells are supported by grid cells in the entorhinal cortex (EC) [182] as well as a multitude of other function-specific cells whose activity correlate with head direction, goal direction and reward vicinity [14, 89, 187, 199, 228].

More recently, it has been suggested this hippocampal-entorhinal system is not only involved in spatial cognitive maps, but may play an important role in structural abstraction for non-spatial relational knowledge [288]. This would predict a role for the hippocampus-EC system in the formation of latent states, as discussed above. This proposal appears to fit neatly with other studies showing a role for the hippocampal system in non-spatial tasks [14, 89, 188, 200, 229], time integration [275] and insight [170].

Although some findings indicate OFC having a larger role for future state representation than hippocampus [302], others suggest the hippocampal complex is highly important as a predictive map for long term reward prediction [248]. It seems that the more likely answer that both are important. Work by [303] suggests OFC and hippocampus play complementary roles, with the hippocampus being engaged especially during high memory load. Work by [176, 218] builds on this and shows how future predictions work across multiple scales, with: anterior prefrontal cortex planning furthest into the future; orbitofrontal cortex and anterior hippocampus predicting medium time horizons; and the posterior hippocampus having the shortest future predictions.

[2.3.6 THE SUCCESSOR REPRESENTATION](#)

One promising specific model for cognitive maps is the successor representation (SR) [96, 175]. We have mentioned it a few times above, and it will be described in detail in chapter five. Briefly, SR can be seen as a combination of model-free and model-based RL, where both future and past states can be encoded as rows and columns, respectively, in a table (in a limited, discrete state space scenario). This is intriguing because the SR can thus learn to approximate the state transition function. This therefore allows for a combination, or compromise, between MB and MF RL. SR-based algorithms has been shown to display and explain behaviour that otherwise would need either MF or MB RL [177, 219] as well as correlating with neuronal

activity in neuroimaging studies on spatial and non-spatial tasks [21, 34, 89, 175, 178, 200, 248].

To go into further detail on some of those studies, work by Stachenfeld and colleagues [248] show how SR simulations display patterns that correlate with place and grid cell activity in rodent studies. Furthermore, SR combined with offline replay better explains human behaviour in revaluation tasks (where reward values and/or state transitions are changed between training and test) than either MF or MB RL [177]. In maze navigation simulations, using latent learning (see chapter five), revaluation and detour tasks – where previous paths to reward are blocked – can similarly be solved flexibly by SR with offline learning [219]. Garvert and colleagues [89] demonstrate SR equations are similar to graph theory and that SR correlates with neural activity for abstract relationships between objects. The same point is argued by Peer and colleagues [200] who show the similarity between spatial cognitive maps and graphs for structuring knowledge. Furthermore, Gardner and colleagues [178] demonstrate that using SR, it may be possible to recast dopamine prediction errors as more general sensory prediction errors, of which reward is but one such prediction error.

2.3.7 HIERARCHICAL STATE-ACTION STRUCTURE

It was mentioned above how actions can be seen as part of the state. The research on discovery and selection of actions is not as rich as that of “states” but has focused on options – sets or sequences of actions – and task sets [58]. The options framework was described above in the section on hierarchies, and work by [179] show that the generation and selection of such options seem to rely on partly different processes. One way of discovering options is by using the successor representation to find both states and action-options concurrently [161].

As noted above, task sets have their origin with Harlow’s learning sets [205]. In a two-choice bandit task, where the best option switches between experimental blocks, monkeys and children “learn to learn”, meaning that after many blocks of the task they are quicker to adapt to the change than they are in early blocks. In other words, when you have experience of a task, as soon as you can identify the current task and fit it onto the structure of a previously learned task, you already know a lot about how to perform. Collins and colleagues [53, 70] show

how such task sets can be created and selected between based on the current context. These proposals are similar in mechanism to the above mentioned attention based systems where frontal cortex maintains and selects between hypotheses for task structure [206].

The nice aspect of such task set models is that they are naturally hierarchical in nature (hence they can be called HRL, see chapter five). They can define context, state and actions together in different configurations. Such HRL models were more recently used in [73], where they show HRL better accounted for human behaviour than either model-free RL or a hierarchical Bayesian model.

2.4 SUMMARY AND DISCUSSION

We started this chapter by looking at the history of RL and its mathematical foundations. We then discussed the two main families of RL algorithms, model-free and model-based, with the first seen as quick and habitual behaviour and the second as slower but more flexible planning behaviour. We contrasted this duality with that of dualities in other fields of cognitive science and saw how it is more likely a question of multiple systems, and not just two.

Based on what we found about RL and multiple systems, we could then ask how animals create states. From that question we clarified three more pointed questions: (1) what are states and how are they created? (2) how are states informed by and combined into task structure? and (3) how is task structure summarized into a model, and how does that model interact with learning and memory? As we have seen these three questions are inherently entangled and are not easy or even possible to separate.

However, the current research points towards a view that attention guides perception towards relevant features that can be used as states. Memory and inferential processes then collaborate to find (probable) causes of observations, and then categorize these observations into task representations consisting of multiple states. Animals track their current “location” in such task representations probabilistically, holding beliefs about the current state. These beliefs are further supported by predictive cognitive maps that help track state transitions and locations based on previous experience. Through feedback loops, attention and working memory can then utilise (reward) prediction errors to adjust and evolve hypotheses about task

structure, potentially changing what environmental details are used for the state representations.

Episodic⁷ memories would provide priors for categorization and task structure hypotheses and these priors, combined with the current observations, result in the posterior belief state distribution of task structure. Creatures like the Australian Jewel beetle would have extremely fixed priors (evolutionary memories) for belief states whereas more flexible creatures like mammals have more flexible priors, provided via both learning and memory. The idea of priors being provided by evolution is something often taken for granted in free energy formulations [10] but it would make sense that some of these priors can be learned, especially in humans.

More general frameworks like predictive coding and free energy fit well with the above research findings, where the processes involved are continuously updated based on reward and sensory prediction errors. These frameworks are obviously similar to the sensory prediction hypothesis [72, 184, 196, 290].

For our purposes, we are interested in investigating how humans can quickly and flexibly adapt their representations of a task in order to find structure. We will do so with a new type of task, presented in chapter five. Thus, we are interested in computational cognitive models that build on the principle of RL and extend this principle with alternative state structures. Based on the existing research presented above, we chose versions of SR and HRL to model behaviour in our new task, details of which are also presented in chapter five.

In order to fit computational models to behaviour, we need model fitting and selection methods. There are many such methods, all with different advantages and drawbacks. Neural network models, for example, are powerful but computationally demanding and difficult to interpret neurobiologically. We will instead generally focus on analytically tractable models; despite this the fitting of such models still has considerable complexities. We will demonstrate these complexities in chapter three and four and use our findings to select the specific methods to use for our investigations in chapter five.

⁷ Most likely other types of memory like procedural memory also play a role, but the literature mainly focuses on episodic memory in this regard. Perhaps a future research venue?

3 METHODOLOGICAL INTRODUCTION BY ANALYSING THE BANDIT TASK

"With four parameters I can fit an elephant, and with five I can make him wiggle his trunk."
— John von Neumann [167]

In this chapter, we will introduce our methodology by demonstration, using the so-called Bandit task. This task is simple enough to make our analyses straight forward, yet able to highlight methodological complexities that have not been exhaustively explored in the existing literature. There is a recent trend however with authors making some of the points we will highlight below [72, 185, 197, 291].

The Bandit task gets its name from being conceptually similar to the "one armed bandits" - slot machines - often seen in casinos. In the case of a bandit with one arm we can make a choice between two actions; to pull or not to pull the arm. In the casino setting, one must deposit a coin for each pull, meaning there's a cost associated with that decision. But the potential reward can be many times more than the cost. Of course, that hope in the mind of the player is what the casino is betting on.

In human reinforcement learning research the option of "no pull" or inaction is rarely considered⁸, and the same will be true for our purposes. What that means is that we need at least two arms for our bandit task to make the task interesting from an experimental perspective. Personally, I imagine this as one machine with two arms, but we could also see it as two machines with one arm each.

The two arms are each separately connected to a certain probability of reward, usually with one arm having higher probability than the other. That means some exploration is needed to find the best option, demonstrating the exploration/exploitation problem mentioned in chapter two. Do we exploit the option we have so far learned to be the more rewarding or do we explore the other option for potentially greater rewards should the reward contingencies change?

Furthermore, whether a cost is associated with a choice can differ between experiments. Sometimes, before each choice one makes a bet

⁸ This would be an interesting research line in itself

on the payoff, but commonly that is not the case in the Bandit task. A more common variant when it comes to rewards is that wrong choices are associated with punishments. As mentioned in chapter two, the evidence for punishments being associated with the same circuitry as rewards are conflicting. Thus, we do not use punishments in our investigations, but rather see the absence of reward as being a simpler place to start.

Importantly, the Bandit task does not have states. Or, from another viewpoint, it has only one state. One state, and two actions. This distinguishes it from the tasks we will investigate in later chapters, where different states will come into play. This simplicity of the Bandit task therefore makes it useful for us to explore first. We are mainly using it here to set the ground rules of our methodology for the subsequent studies and simulations.

We could of course add more arms to increase the complexity of the task. This can be referred to as the n -bandit task or multi-armed bandit process, where the number of arms is represented by n .

The bandit task can also be represented by playing cards, where, for example, we have two or several decks of cards to draw from. This version and its variants are often referred to as the Iowa Gambling Task [20, 35, 296] and has been used to study many psychological phenomena such as autism [300], depression [38] and schizophrenia [252].

There are myriad examples of how such a seemingly simple task can be useful as a model for real life events outside the lab. Just as statistical textbooks often use coin flips to demonstrate a probability distribution that can represent real situations such as whether someone has a certain disease, or if a treatment works or not; the Bandit task is the decision research equivalent. Any situation where an animal must explore two options in order to discover which one provides the bigger payoff can be modelled as a bandit task. The most obvious one being trying different foods to find out the one which provides the most sustenance or the nicest taste. For example, bumblebees searching for food in a newly discovered flower patch where flowers are of different colours [68, 140, 179, 234]. Usually, however, there are more factors at play like the look and smell of the options – states, as we shall revisit in later chapters – but at a basic

conceptual level many choice tasks can be described as bandit tasks [253].

Here, we are not claiming any major new findings with regards to the bandit task itself. As mentioned above this task has been explored previously in many studies [68, 140, 180, 235]. However, as noted, it is useful to establish baseline properties of RL and to present our methodology. With regards to the methodology, there are some novel features, and we are going to present different modelling approaches and contrast their advantages and disadvantages. This has partly been done in previous research [64, 204, 252, 253, 291] but to our knowledge not as exhaustively as we attempt here. As mentioned above, there is a recent trend showing some of the basics of model fitting are not as simple as a reading of the literature would make you believe [72, 187, 197], and our investigations below further adds to this point.

Furthermore, the methods established in this chapter are going to lay the foundation for investigations in subsequent chapters. Since those subsequent chapters will use more complex tasks and models, it's helpful to see how they work in a simpler case first.

3.1 WHAT ARE MODELS?

Before getting into the details of our modelling methodology, we should discuss what models are and what they can tell us. Webb [287] discusses – in short – what robots can tell us about animal behaviour, a discussion that is appropriate here as well. Though we do not construct physical robots, our simulations can be seen as virtual robots demonstrating (limited) behaviour change via some sort of learning. What [287] says is we can see robots as models, but we should keep in mind how and where in the investigative process these fit, as per Figure 3.1.

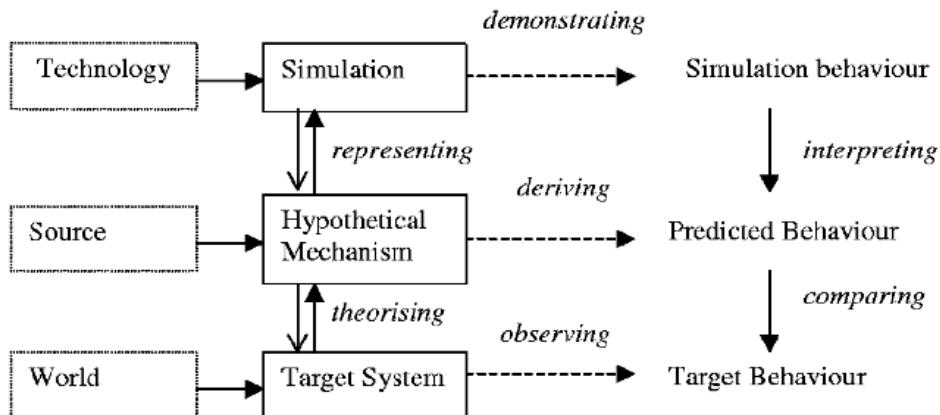


Figure 3.1 Overview of simulations (robots) as models of animal behaviour in the world. Our investigations draw from theories sitting on the middle row – source – and are implemented by computers (technology, top row) as simulations. These simulations can provide predictions of behaviour that we can compare to behaviour in the target system in the world (bottom row). Adapted from [287]

A complementary view comes from the “three levels of Marr” [192], where we have the computational level (what is the problem), the algorithmic level (what is the solution), and the implementational level (what is the physical structure that runs the algorithm). In our case – reinforcement learning – our problem is optimal decision making (maximising reward and in some cases minimising punishment), the algorithm is reinforcement learning and as we have discussed previously the implementation is (mainly thought to be) dependent upon the dopaminergic system projecting to the mammalian basal ganglia. According to Marr, if we have support on all the three levels then we can say with greater certainty that the robot’s behaviour supports our hypothesis.

However, it’s appropriate to heed Webb’s warning that “a model that behaves like its target is not necessarily an explanation of the target’s behaviour.” [287].

3.1.1 INDIVIDUAL DIFFERENCES

A common question in psychological research is to investigate how model parameters may correlate with other characteristics, such as personality as measured by a personality scale (e.g., Big5/OCEAN; [216]) or the presence/absence/degree of symptoms in some clinical population such as depression or schizophrenia [112, 204]. Perhaps one learns slower in rewarding contexts if one is depressed, but there is no effect for learning in non-rewarding contexts? This could

potentially be reflected in a lower average learning rate in modelled behaviour (e.g., the alpha parameter in Q-learning models, below) for a group of patients with depression versus that of a healthy control group.

These questions have in recent years become known collectively as computational psychiatry [119], rising in popularity together with the increased availability of computational power, and tools for fitting models becoming easier to use and more widely available.

The common way to conduct experiments in the literature is to test models at the group level. Either for different populations, as in asking the question “is there a difference in learning rate between individuals with ADHD and those without?”. Or other times it can be about asking “in this experimental task do people use model free RL, model-based RL or a combination of the two?”.

An important aspect to keep in mind with many studies within the cognitive sciences is they disproportionately rely on Western, Educated, Industrialised, Rich, and Democratic (WEIRD) experimental subjects [112, 205]. For example, it is most common to recruit participants among university undergraduates, who receive course credits for participation. What is problematic with this is not only are these subjects a small subset of the world population, but they have also been found to be quite different from the average person [36, 198].

In recent years, online recruitment of participants has become more widely available with services such as Amazon Mechanical Turk (www.mturk.com) and Prolific (www.prolific.co). These services have been shown to provide both high quality data [36] and increased diversity compared with the typical university student samples [36, 199]. In other words, using online recruitment services for gathering data may alleviate some of the diversity issues with traditional experiment participants. However, the trade-off of using online testing is we sacrifice control in being unable to check what participants are doing and so may have to discard more data (see examples of this in chapter five).

Issues with diversity do not necessarily invalidate existing results or approaches, but they highlight the question of individual differences. Perhaps it is the case that a majority of people use model-based RL in some choice task, but other individuals actually use

model-free RL. Some published work [64, 72, 290] shows very nicely how individuals might differ; see Figure 3.2. These group-based approaches are not necessarily invalid, but they do gloss over the question of individual differences.

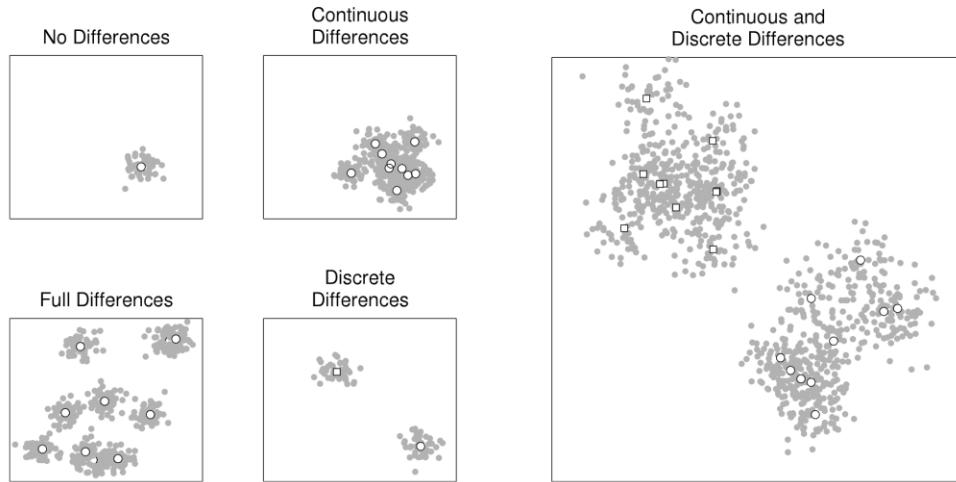


Figure 3.2 Different scenarios of individual differences in some experimental task. The white dots represent a specific parameterization common for one or several participants. The grey area show the range of inferences possible to make about the white dot, based on behavioural data. Top left: There are no individual differences, and the true value is said to be the population mean of the measure of interest. Top middle: Individuals vary continuously around some common distribution, which is the case for hierarchical models discussed below. Bottom left: Each individual is different and there is no common structure between them, which is the case when we fit models individually below. Bottom middle: Individuals vary in what strategy (model) they use, which we will also see examples of below when some individuals are better explained by one model over another. Right: All the types of differences are combined, and we have differences in strategy (model) used and within those strategies people vary in how they are applied. Adapted from [18]

“Fitting a model” is the process by which we can get a statistical measure of how well a certain model fits some dataset. For example, is it the case that a particular reinforcement learning model can accurately describe the behaviour of a human performing the two-armed bandit task? One should however keep in mind that even if we contrast several models and one of them fits “best”, this is only the best-fitting among the ones we have tested. There will always be, due to practical constraints, a larger space of models not considered relative to the small number of those actually considered.

Considering how difficult model fitting is to get to work well at all [64, 72, 291] (also see below), it is perhaps no wonder that the question of individual differences is not commonly brought up. But it is still an

important aspect to keep in mind and has important implications for what we can infer from model fitting [64, 146, 290] and how generalisable the findings are [185].

Exhaustive demonstrations of the model fitting methods we present below have occasionally been done separately before [64, 147, 291], but to our knowledge have rarely (perhaps never) been presented without making the case for one method over the other. So, here we focus more on contrasting them to prepare for later investigations with more complex tasks.

Nevertheless, despite the difficulties of model fitting, it can still be a useful tool, if one remains aware of the classic adage that “all models are wrong, but some are useful” [62, 184].

3.2 MODELS AND TASK PARAMETERS

We will now take a closer look at the details of the task and the algorithms and models used with this task. As mentioned above, the bandit task we will be using is a two-armed bandit task, henceforth referred to as the Bandit task. This task has three task parameters: the reward probabilities for each of the two arms and the number of trials (the number of pulls on the arms). We will refer to these parameters as $p_{arm1}(r)$, $p_{arm2}(r)$ and T in equations when presented in text. In code⁹ they are referred to as `arm1`, `arm2` and `trial_count`.

For our first simulations, and later model fits, we are going to use Q-learning [258, 286] and an algorithm making random choices, which has a possible bias for picking one arm more often than the other. We call the latter algorithm simply RandomBias.

Q-learning commonly has three model parameters: learning rate α , discount parameter γ (which determines the influence of expected future rewards) and, for choosing actions, uses either the SoftMax β parameter or ε -greedy choices with the ε parameter. However, an issue with ε -greedy is that likelihoods may become intractable and therefore difficult to recover [62, 185], and we thus focus on SoftMax. Here we will also leave out the discount parameter, because in a task such as the bandit task we are looking at one-step rewards (i.e., the action on each trial is followed by a reward or non-reward event). The

⁹ https://github.com/fohria/phd_thesis

discount parameter is useful in tasks where there are multiple steps - multiple action choices - needed before a reward is found. In the computer science literature this may be called "sparse rewards" [259]. In a later chapter we are going to contrast Q-learning with two parameters and Q-learning with three parameters to demonstrate this difference (section 5.2).

In this chapter then, we use Q-learning with two parameters: learning rate α and the inverse temperature β . The latter is sometimes called τ in the literature. Henceforth this version of Q-learning will be called " QL_2 ". QL_2 is thus the equation:

$$Q_{t+1}(a_t) = Q_t(a_t) + \alpha(r_t - Q_t(a_t)) \quad 3.1$$

where $a_t \in \{1,2\}$ denotes the arm chosen (or action)¹⁰, r_t is the reward on trial t , and Q_t holds trial to trial values representing the values of the different actions. Note that only the selected action's value is updated.

Because the Q values vary independently, we use SoftMax to create a choice probability vector from the Q values:

$$p_t(\mathbf{a}) = \frac{e^{Q_t(\mathbf{a})}}{\sum e^{Q_t(\mathbf{a})}} \quad 3.2$$

where $\mathbf{a} = \{a_1, a_2\}$.

Together, equations 3.1 and 3.2 can be combined into an algorithm as seen in Code Snippet 3.1.

¹⁰ In Python code, index starts from 0, so there arms would be 0,1 instead

```

Q = np.array([0.5, 0.5]) # init with equal probabilities

for trial in range(trial_count):

    # compute choice probabilities using softmax
    q_soft = Q - np.max(Q)
    probabilities = np.exp(beta * q_soft) / np.sum(np.exp(beta * q_soft))

    # make choice based on choice probabilities
    actions[trial] = choose([0, 1], probabilities)

    # generate reward based on choice
    rewards[trial] = np.random.rand() < bandit[actions[trial]]

    # calculate prediction error and update action values
    delta = rewards[trial] - Q[actions[trial]]
    Q[actions[trial]] += alpha * delta

```

Code Snippet 3.1 *Q-learning algorithm in Python.* It uses two parameters: learning rate α and SoftMax temperature β . The “*np*” is short for *numpy*, a standard mathematical library for Python.

Technically, α may be larger than 1 but in practice this is rare and will lead to instability. So, we assume a parameter value range of $0 < \alpha < 1$. For β , its effect on the Q-values is that β values closer to zero decreases the difference between the Q-values, thus increasing explorative behaviour (or stated differently, increasing random choice of actions). Higher values of β will accentuate any difference between the Q-values and cause increasingly greedy behaviour, i.e., picking the action with the higher value. The possible parameter value range is thus $\beta > 0$, but in practice we will most often use $1 < \beta < 20$. Below we investigate how different values of α and β impact performance.

The randomly playing agent, from here on called “*RandomBias*” agent, picks a random action on each trial. But it may have a bias towards picking one arm over the other, which is controlled by the parameter *bias*. The probability of the *RandomBias* agent to pick each arm is thus expressed as:

$$p_{\text{arm}_1} = \text{bias}, \quad p_{\text{arm}_1} + p_{\text{arm}_2} = 1$$

The *RandomBias* agent therefore doesn't need to use a choice function such as SoftMax as the model specifies action probabilities directly. Thus, the parameter range for *bias* is $0 < \text{bias} < 1$.

3.2.1 NOTE ON TERMINOLOGY

Here we also introduce certain terminology to more easily distinguish between different parts of our investigations. In simulations, we will refer to artificial agents as either agents or subjects. These agents or subjects "play" a task, for example the bandit task. Often, we perform multiple simulations for a specific parameter value combination for an agent, where we refer to it as agent or subject repetitions. Later, when including human participants in our investigations, we may refer to them as subjects or participants depending on the context. Importantly, when we "fit models" we use the term model - even though technically an artificial Q-learning agent and a Q-learning model are the same thing.

When investigating how well the models can be fit to data, we will simulate data using known agents and parameter values. Then we fit models to this data and see how close we get to true parameter values (a process often referred to as testing "parameter recovery"), as well as if we can identify what agent type generated the data. Often we refer to the resulting datasets as "simfits", short for simulations and fits.

3.3 SIMULATING PERFORMANCE IN THE BANDIT TASK

To explore the task performance of our agents in the bandit task we will first stick to one specific variant of the Bandit task with task parameters:

$$p_{arm1}(r) = 0.2, p_{arm2}(r) = 0.8, T = 1000$$

For both arms, $r = 1$ if a randomly generated number is below $p(r)$, otherwise $r = 0$ (see above Code Snippet 3.1). It is not important, in this example, that we have arbitrarily chosen the task parameters such that $p_{arm1}(r) + p_{arm2}(r) = 1.0$.

For each agent type (*QL2* and *RandomBias*), we randomly generate 1000 combinations of its agent parameter values. These values were generated individually for each agent in the following way:

$$\alpha_{QL2} \sim U(0, 1), \quad \beta_{QL2} \sim U(1, 20), \quad bias_{RandomBias} \sim U(0, 1)$$

We also repeat simulations with each such agent parameter combination 100 times to get an average performance for that model

parameter combination. This is due to the inherent random variation across runs for the same parameter combination¹¹. We also calculate a score - on each trial - which is based on whether the agent picked the correct arm or not. Correct here means choosing the arm with higher reward probability.

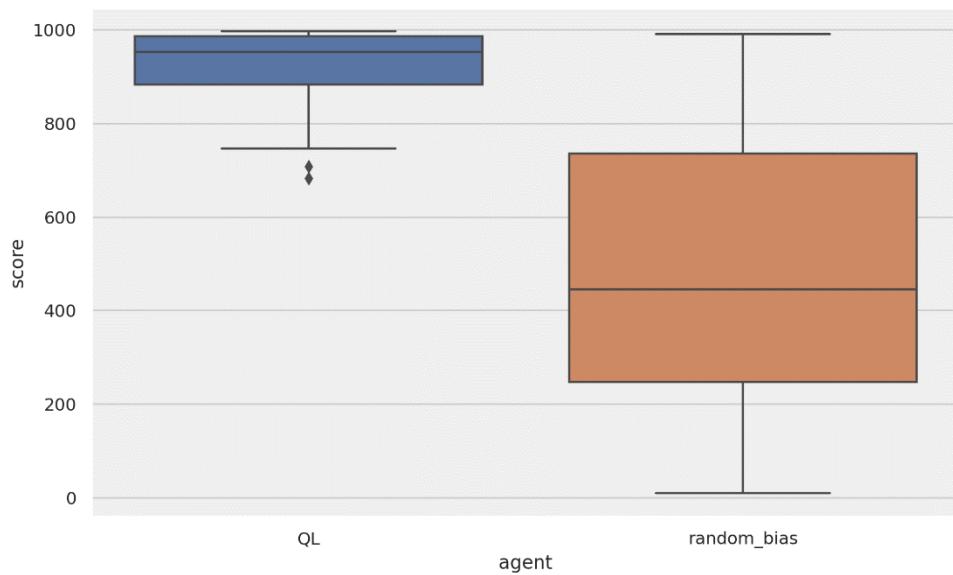


Figure 3.3 Performance summary for agents QL2 and RandomBias playing the Bandit task for 1000 trials. Score is calculated as the sum of correct choices, where correct choice is the arm with higher reward probability.

As seen in Figure 3.3, the score is the total sum of all correct choices, where a correct choice is picking the arm with the highest reward probability. In the plot we can see that the *QL2* agent learns this well enough to get an average score of 920 (SD 83), while the *RandomBias* agent nicely has an average of around the middle possible, 493 (SD 294).

3.3.1 EXPLORING EFFECTS OF VARYING MODEL PARAMETERS

Overall performance across the two agent types regardless of parameter values, as done above, is useful to make sure our agents perform in the way we expect. In this case it so happens that we see an overall pattern between the two agents, but it could have been the case that *QL2* performance was only good for specific parameter combinations. For simpler models this can be easier to predict, such

¹¹ If we run one agent with a specific parameter value combination, say $\alpha = 0.3$ and $\beta = 5$, 100 times, we will get an average sum of correct choices of 940.5 (SD 14.6).

as for the *RandomBias* agent where there will be a good correlation between agent parameter value and score. Depending on our goals we may also be interested in what parameter combinations – if any – are better than some others.

For those reasons just stated, we should also investigate performance across different parameter combinations. We call this a "parameter sweep", where we define a range of values for each parameter. For the *QL2* agent, which has multiple parameter values, we combine the ranges across all permutations. As before, we simulate each parameter combination 100 times to get an average for that specific parameter combination. The values used for these permutations for the *QL2* β parameter was 1, 2, 5, 10, 20. The remaining two parameters are as below, where the first number is the start, the second number is the end, and the third number is the step size.

$$\alpha_{QL2}: 0.01 \rightarrow 1.0; 0.02$$

$$bias_{RandomBias}: 0.01 \rightarrow 1.0; 0.01$$

For clarity, this means 50 α values and 100 $bias$ values. As an example, one *QL2* parameter combination is $\alpha=0.03$, $\beta=5$. For this parameter combination, 100 subjects are simulated, and their scores averaged. Since we have 5 β values and 50 α values, we have $5 * 50 = 250$ parameter value combinations.

In Figure 3.4 we have our *QL2* agent to the left and our *RandomBias* agent to the right. Note here we use $p(correct)$ - probability of picking the correct arm - as our measure on the y axis instead of the total sum of correct choices as we did above.

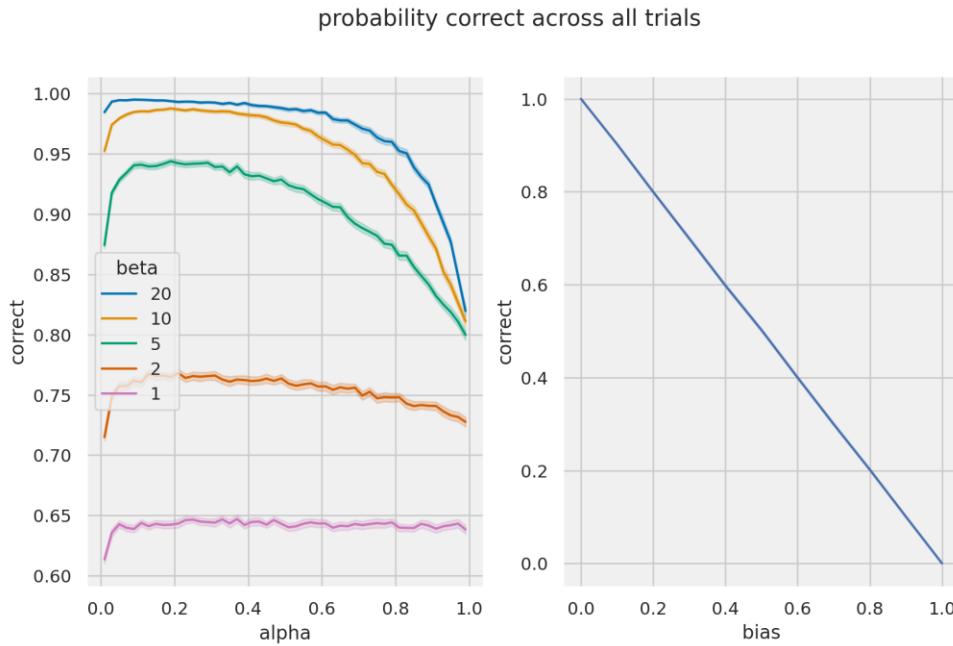


Figure 3.4 Probability of picking the correct arm – the arm with highest reward probability – for parameter sweeps with QL2 (left) and RandomBias (right). Left: QL2 agent performance. X-axis represents the α parameter space and the differently coloured lines are different values of β . Right: RandomBias performance. Note that y-axis differs between the left and right plot. Shaded areas around lines represent 95% confidence interval.

In the left part of Figure 3.4, we can see how the probability of correct choice (y-axis) depends on α (x-axis) and β (coloured lines). What can be gathered is that for high performance (high probability of correct), we'd like $\alpha < 0.4$ and $\beta > 5$. At higher α , performance decreases because with a high learning rate, what Q-value is currently biggest will be more likely to oscillate depending on the reward received on the last trial (as implied by Equation 3.1 above). Since higher β means greedier (i.e., exploitative) behaviour, the combination of high α, β together means behaviour becomes slightly more oscillatory and thus performance suffers overall.

In the right part of Figure 3.4, we see that our *RandomBias* agent behaves precisely as expected. If it has $bias = 0$ (left side of x-axis) for picking arm₁, that means conversely that it will always pick arm₂, which is the correct arm. As we follow the x-axis, we get higher $bias$ for arm₁ and thus the probability of picking the correct choice becomes lower.

3.4 EXPLORING TASK PARAMETERS

Another aspect to look at is how variations of the task itself can impact performance. Following on from the performance we just looked at above, we can easily look at how agent parameter combinations differ between different trial counts for the Bandit task. Since every trial is the same, we can use the same data as above and look at the first n trials and compare performance. Here we will only do this for the QL_2 agent, since we have already established that the *RandomBias* agent works as we expect, and it is not very interesting to look at this agent further here.

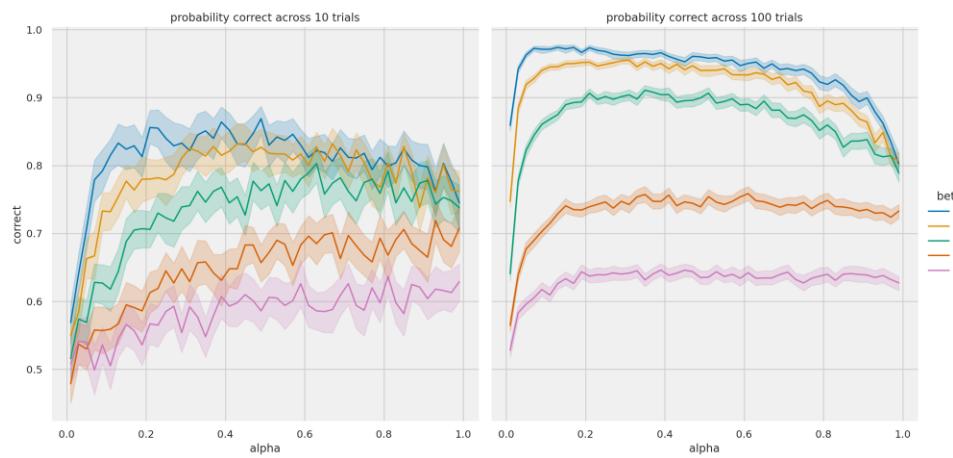


Figure 3.5 QL_2 performance across different number of task trials. Left; 10 trial long Bandit task. Right; 100 trial long Bandit task. Coloured lines in both plots represent different values for β parameter. Shaded areas around each line represents 95% confidence interval.

In the left part of Figure 3.5, we see performance for the first ten trials. With $\beta > 10$, the agent is quite greedy and so finds and sticks with the correct arm. It can thus reach eight or nine correct choices within these first ten trials, across a wide range of alpha values. Specifically, α needs to be somewhere between 0.2 and 0.6 in order to reach those eight or nine correct choices. Meanwhile in the right side of Figure 3.5, we see that within the first 100 trials the overall behaviour is already quite like that for the 1000 trials plotted in the left side of Figure 3.4. But here we have slightly wider confidence interval shadings, since we only have a tenth of the trials to average over.

3.4.1 VARYING BANDIT ARM REWARD PROBABILITIES

Finally, we can also investigate the effect of varying the arm reward probabilities for the bandit. We do this in two different ways. First, we assume $p_{\text{arm1}}(r) + p_{\text{arm2}}(r) = 1$. In other words, we vary the arm reward probabilities in a dependent manner. The range of values used are $0.1 \leq p_{\text{arm1}} \leq 0.9$ in steps of 0.1, totalling 9 values for p_{arm1} . Second, we let both arms vary independently in the range between 0 and 1; $p_{\text{arm1,arm2}}(r) \in (0, 1)$. We use the same range as just mentioned above, but now use this for both arms and create all permutations of these resulting in a total of 81 value combinations. In both cases we use a QL_2 agent with somewhat arbitrarily chosen (but guided by the results in the previous section) parameter values $\alpha = 0.3, \beta = 5$ to emphasise that in this part of the demonstration we keep the agent stable while varying the task parameters. As earlier, for each arm reward combination, we simulate 100 agents to get an average across runs.

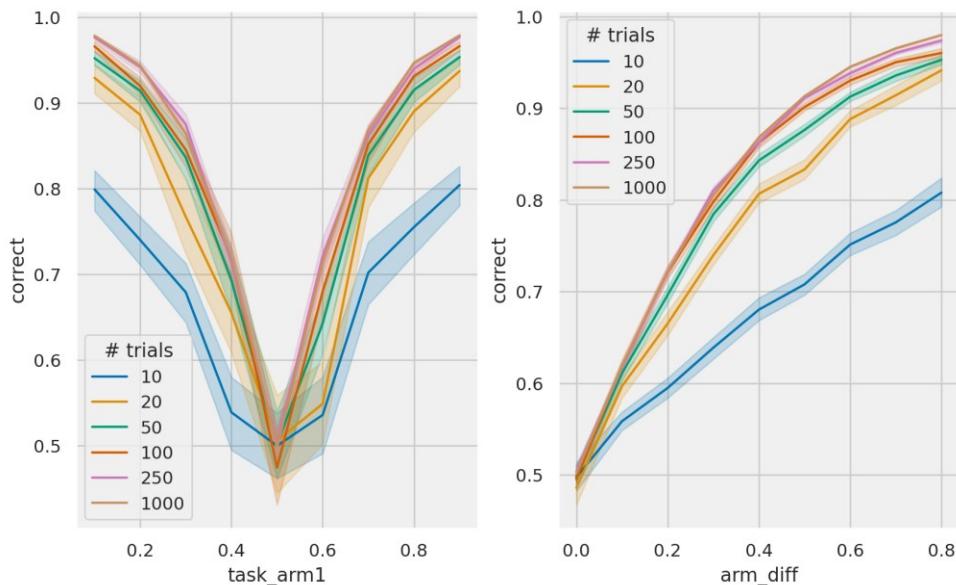


Figure 3.6 Impact of arm reward probability differences for the Bandit task. Left: Arms vary dependent on each other, so their individual reward probabilities sum to one. X-axis show probability for one of the arms. Right: Arms vary independently of each other between zero and one. X-axis shows the difference in reward probability between the arms. Both; shaded areas around each line represents 95% confidence interval.

In Figure 3.6, left, we have plotted the reward probability for arm1 on the x-axis. Since the arm reward probabilities are dependent on each other, the reward probability for arm2 is $1 - \text{arm1}$ for each point on the x-axis. We can see that this produces a nicely symmetrical

performance plot, where the probability of picking the correct arm is 50/50 when $p_{\text{arm}_1}(r) = p_{\text{arm}_2}(r) = 0.5$. Probability of correct choice then goes up towards the left or right as one arm gets higher probability of reward than the other. More interestingly, we can see how the blue line for 10 trials is much lower than the others, and how already at 20 trials, the difference to 50 or more trials is not huge, less than 0.1.

In the right plot of Figure 3.6, the arm reward probabilities vary independently. The x-axis shows the difference in arm reward probability, regardless of the absolute probability value. Here we see a similar pattern in that after around 20 trials, the difference in average performance is fairly small when increasing the number of trials in the task.

One overall conclusion we can draw here is that already around 100 trials, average aggregated behaviour for the *QL2* agent is quite like that for 1000 trials. This can be an important insight when planning experiments involving humans, as they are easily bored by such a simple task as our two-armed Bandit task. Putting a human through 100 trials is much more reasonable than having them do 1000.

At the same time, 100 trials may not yield enough data for more involved statistical analyses. In order to inform such subsequent analyses, it's important to understand the task variations and their interactions with agent types and parameters. We shall presently see how in the next sections.

3.5 RECOVERING PARAMETERS WITH MAXIMUM LIKELIHOOD ESTIMATION

The standard and straightforward method of fitting a model to data is maximum likelihood estimation (MLE). For a model m , we wish to find parameters θ_m such that they maximize the likelihood of our data D . This can be written as:

$$p(D|\theta_m, m)$$

What we would like to find then, is the probability of θ_m , in order to maximize it, i.e.,

$$p(\theta_m|D, m)$$

According to Bayes' Rule, we can write this as:

$$p(\boldsymbol{\theta}_m | D, m) \propto p(D | \boldsymbol{\theta}_m, m) \cdot p(\boldsymbol{\theta}_m | m) \quad 3.3$$

where $p(\boldsymbol{\theta}_m | m)$ is the prior probability of the parameters. The above can be described as that the posterior probability of $\boldsymbol{\theta}_m$, given data D , model m is proportional to the likelihood (first term on the right-hand side) of the data D , given parameters $\boldsymbol{\theta}_m$, model m multiplied by the prior probability of the parameters $\boldsymbol{\theta}_m$, given the model m . We will come back to discussing probability distributions below when we look at Bayesian parameter estimation methods, but for now we will be satisfied with point value estimates of $\boldsymbol{\theta}_m$, since that is what MLE provides.

Because the left- and right-hand sides in Equation 3.3 are proportional to each other, the relationship still holds if we treat the prior probability of $\boldsymbol{\theta}_m$ as flat (e.g., ignore it). Therefore, for the purposes of MLE, we are left with:

$$p(\boldsymbol{\theta}_m | D, m) \propto p(D | \boldsymbol{\theta}_m, m) \quad 3.4$$

If we define $\hat{\boldsymbol{\theta}}_m$ as the parameter values that maximize the likelihood (right-hand side of Equation 3.4), then that is our maximum likelihood estimate (MLE).

In our simulations and experiments, our data D consists of observed choices c at each trial or timestep, together with an observed or implied reward r . If $D = (c_t, r_t)$, then our likelihood looks like:

$$p(c_t | D, \boldsymbol{\theta}_m, m)$$

And thus, the probability of some sequence of choices and rewards is the product across all trials T :

$$\prod_T p(c_t | D, \boldsymbol{\theta}_m, m)$$

Furthermore, since the probabilities at each time step are small values less than 1, the above product will quickly become too small and close to 0 to be computationally tractable even for small numbers of T . Therefore, what is almost always maximized is not the above product but instead the log-likelihood sum, which is mathematically

equivalent. If we define the maximized log-likelihood sum as \widehat{LL} we then get the final form of the expression that we seek:

$$\widehat{LL} = \sum_{t=1}^T \log p(c_t | D, \boldsymbol{\theta}_m, m) \quad 3.5$$

To evaluate and find the maximum \widehat{LL} , we have two alternatives. One is the brute force approach – evaluate the likelihood function for as many parameter combinations as we can and pick out the one with the highest \widehat{LL} . This may work okay for simpler models like QL_2 where we only have two parameters but will quickly become infeasible as we increase the parameter count and number of trials.

The more efficient approach is to use optimization algorithms implemented in existing software packages, such as the `minimize` function in Python's SciPy [280]. The careful reader will notice that the function is meant to minimize a function, but this is easily overcome as we can simply have our log-likelihood function return a negative log-likelihood. With all our tools in place, we shall now see how they can be applied.

3.6 SIMULATE AND FIT QL_2

We start with an overall recovery analysis of our QL_2 model. We simulate 1000 artificial QL_2 subjects, where each subject receives random parameter values for α and β drawn from uniform distributions:

$$\alpha \sim U(0,1) \quad \beta \sim U(1, 20)$$

Our subjects will play the Bandit task for 1000 trials, where the arm probabilities are:

$$p_{arm1}(r) = 0.2 \quad p_{arm2}(r) = 0.8$$

For each subject, we record the actions and rewards received and feed those into SciPy's `minimize` function. The function requires us to specify maximum bounds for the parameter values to be tested. Technically, there are unbounded optimization algorithms to pick from in the same SciPy package, but these can often lead to unstable computations, leading to numerical errors or even program crashes.

We have chosen to bound α between zero and one, and β between one and 40. Note that the range of possible fitted β values is twice as large than the actual values used for simulations. We do this to investigate the scenario where we want to assume as little as possible about our data yet have reasonable boundaries for stable computations. Although it would be possible to imagine α being more than one, such large step sizes are uncommon and can also lead to huge Q-values for each action, in turn causing high chance of overflows [64]. We set a fairly high upper boundary for β for two reasons. One reason is to demonstrate that for real data, we do not know the real parameter values that were used to generate the data, even if we knew the exact model used to generate that data. Second, this large upper bound can be useful to identify cases where the likelihood surface (see Figure 3.8) is such that precise parameter value estimations are incredibly difficult.

These 1000 simulations and model fits can be done very fast, less than 30 seconds on a modern laptop and less than four minutes on a laptop from 2015. This is thanks to three points. First, the SciPy library `optimize` function as mentioned above. Second, we have optimized our simulation and likelihood functions using Numba [142], which compiles Python code into C code on the fly. Third, by using the Python built in library `multiprocessing` for easy use of multicore processors¹².

For each simulate-and-fit (`simfit`) pair, we save the $(\alpha, \beta)_{sim}, (\alpha, \beta)_{fit}$ point values and calculate the absolute distance between the values like so $|\alpha_{sim} - \alpha_{fit}|$, and equivalently for β . We call these “ α distance” and “ β distance”, respectively. Note that due to inherent randomness in action choices, even when using a fixed random seed for generating the parameter values used for simulations, we will still get variation in fitted parameter values as the observed actions can still vary, as exemplified earlier when we ran 100 subjects with the same parameter values (see footnote 11). Because of this variation, we do not get the same values each time we run the code. However, the general trends and types of outliers etc., still occur between each run, so the precise values reported below can be considered as representative examples.

¹² See the code repository for details, https://github.com/fohria/phd_thesis

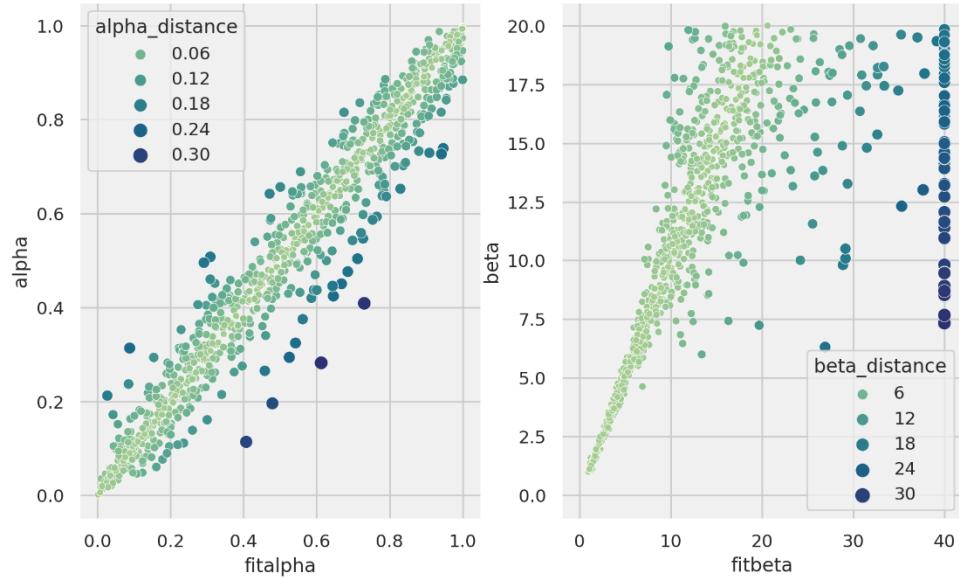


Figure 3.7 Correlation plots between parameter values used in simulations (y-axis) and recovered parameter values found by fitting the QL2 model (x-axis). Colour and size of dots indicate the absolute distance between simulated and fit value as per legends. Left: α parameter (learning rate). Right: β parameter (inverse temperature). Note that x-axis for β plot has double the range as its y-axis. Also note there are sizes smaller than the legends show but not bigger.

In Figure 3.7, the results from our simfits are plotted. For α , the correlation is quite good with $R^2 = 0.98$. But for β the story is slightly worse. We have $R^2 = 0.70$ which may sound all right, but as we can see in Figure 3.7, there are an additional two main patterns that complicate the picture. It is striking that these complications arise even when we have 1000 trials in each simulation. As already noted, this number of trials will far exceed the number of trials used in most human learning experiments. With fewer trials one would expect the quality of the parameter recovery to be poorer. We explore this issue more fully below.

First, by following the y-axis for β upwards, we see that below simulated β of around 6, the correlation is quite strong. Above that we start getting overestimations for fitted β values to a larger and larger degree as the simulated β increases. This is because, as seen in Figure 3.4, as we increase β , behaviour increasingly becomes greedier, always selecting the action with the higher Q-value. So, for larger β the difference in behaviour between agents is hard to distinguish, since random variation can cause an agent with $\beta = 10$ to pick actions in a way indistinguishable from one with $\beta = 20$.

Second, for β more than around 6 we have increasing number of cases where estimates “hit the wall” of our boundary of our fitted beta values ($\text{max}=40$). Sometimes, this might indicate implementational errors or that our model doesn’t explain the data [290] but, in this case, we know what model we used to generate the data. When we hit the wall here, it is due to the same issues mentioned in the previous paragraph, and MLE is simply not equipped to handle these cases. Since we know that we have simulated with β at a maximum of 20, and say we accept fitted values within a very generous error range of 10, we can check how many cases have a fitted $\beta > 30$. For the 1000 simfits we did, 98 of them meet this criterion.

Another way to view this issue is to look more directly at the likelihood surface, created by using a specific sequence of data and calculating the likelihood across a wide range of parameter values. This is essentially the brute force way of calculating the likelihood, mentioned above as impractical. But it is useful to demonstrate how the just mentioned issues arise.

We thus select values for a QL_2 agent that are informed by the above results to allow for the possibility of fits that hit the wall. The QL_2 agent thus uses $\alpha = 0.5, \beta = 12$ and we simfit it for the Bandit task 200 times. We save all actions and rewards, and then select one of the best fitting value pairs ($\alpha = 0.51, \beta = 11.97$) and one of the worst ($\alpha = 0.77, \beta = 40.0$). The resulting likelihood surfaces for the two cases are plotted in Figure 3.8.

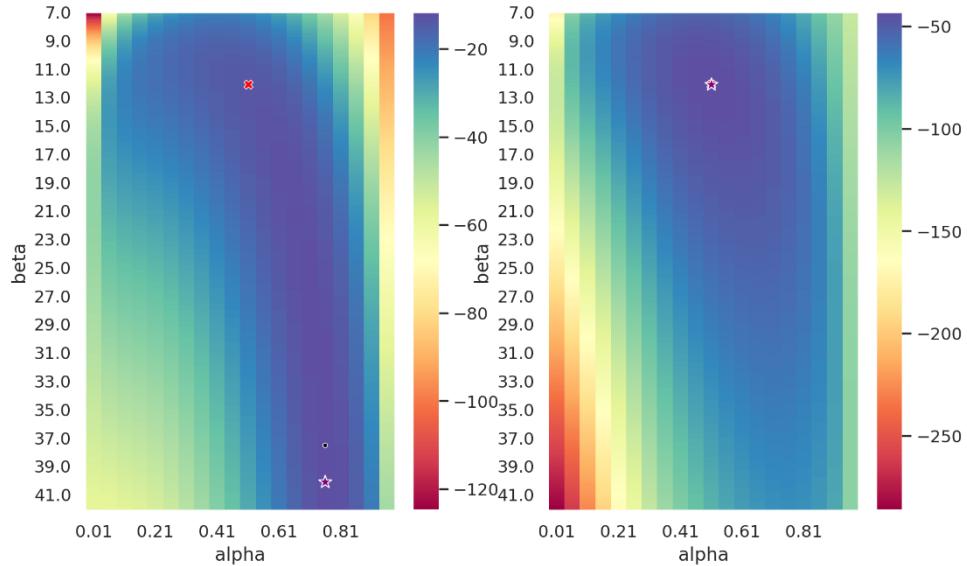


Figure 3.8 Likelihood surfaces for two separate choice sequences, both generated by the same QL2 agent. Red cross is the parameter combination for the agent. Purple star is the MLE fitted parameters. Black dot is the best likelihood found by brute force. Left: Likelihood surface where the best likelihoods form a long ridge, making it hard to find the right combination. Right: Likelihood surface where the best likelihoods are concentrated to a smaller area. Note here the star covers the other two symbols as they are all in essentially the same position.

As seen in the figure, the choice and consequent reward sequences result in likelihood surfaces that look quite different. In one case, the maximum likelihood sits along a ridge far from the value we are looking for (Figure 3.8, left), and in the other case the maximum likelihood is right where \times marks the spot (Figure 3.8, right). Remember, these sequences are generated by the same agent, and we can here clearly see how this method can be problematic, at least for some models. For Q-learning the issue is that β and α are not independent in their influence on performance (as seen in Figure 3.4). This characteristic can create the problematic likelihood surface that is illustrated above.

3.7 RECOVERY QUALITY WITH VARYING NUMBER OF TRIALS

As mentioned above, it is unlikely we would get humans to do 1000 trials of any task, especially such a boring task as the two-armed bandit. Then again, we would perhaps find that casinos hold enormous data stores on humans doing exactly that. Unfortunately, we do not have access to such data.

In experiments in the cognitive sciences, we do not have the monetary resources of casinos to encourage subjects to keep playing. Instead, we must make do with fewer trials in experiments. A very important question when recovering parameters then, is how many trials are needed to still get reasonably good estimates? This question is unfortunately one rarely mentioned in the literature [64, 291], where instead trial counts are raised to compensate, or the topic left open.

To investigate how the number of trials impact how well we can recover parameters, we repeat what we did above; we simulate and fit 1000 times. But this time we vary the number of trials for the task across 1000, 500, 250 and 100 trials. All these task variations use the same reward probabilities for each arm as stated above, and the same for agent parameter values:

$$\begin{aligned} p_{arm_1}(r) &= 0.2, \quad p_{arm_2}(r) = 0.8 \\ \alpha &\sim U(0,1), \quad \beta \sim U(1,20) \end{aligned}$$

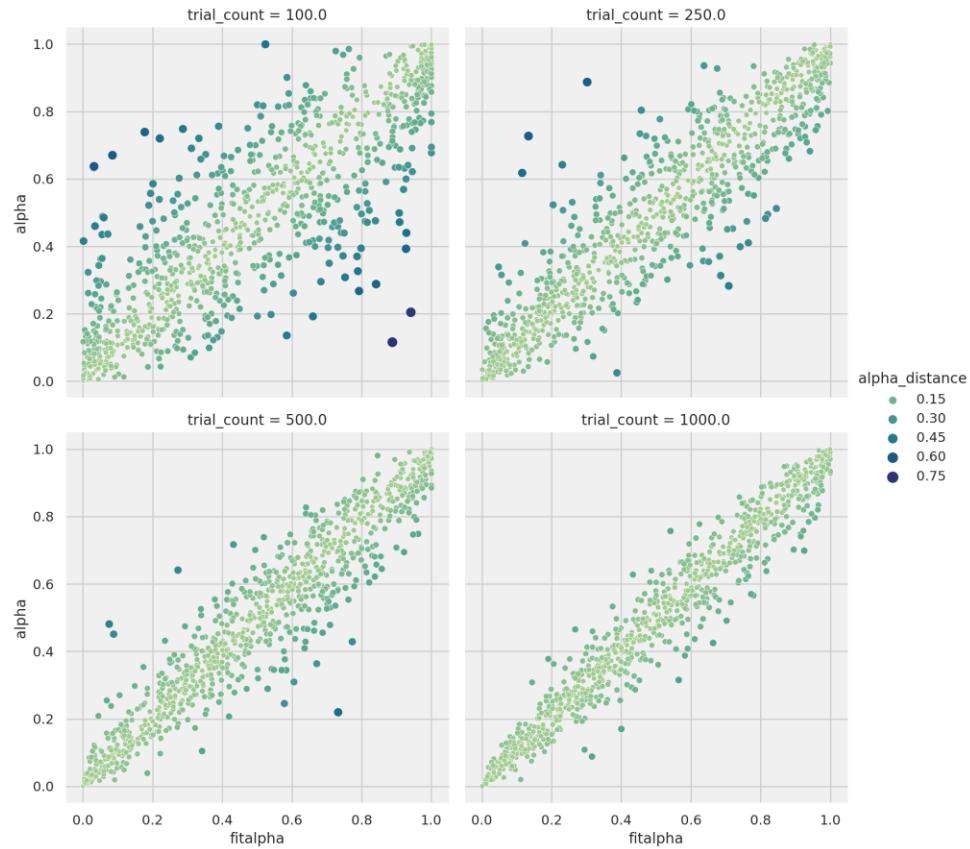


Figure 3.9 Correlation plots of α values, where simulation values are on the y-axis and fitted values on the x-axis. Colour and size of dots represent distance between fitted and simulation value. Top-left: 100 trials. Top-right: 250 trials. Bottom-left: 500 trials. Bottom-right: 1000 trials.

For α , R^2 values for 1000, 500, 250, 100 trials are, respectively: .98, .97, .94 and .87. Although we get a lower correlation coefficient for 100 trials, this number doesn't tell the whole story. As we can see in Figure 3.9, 100 trials clearly show a wider spread. Thus, perhaps a better visualisation of this is violin plots for the distance between α_{sim} and α_{fit} , shown in Figure 3.10.

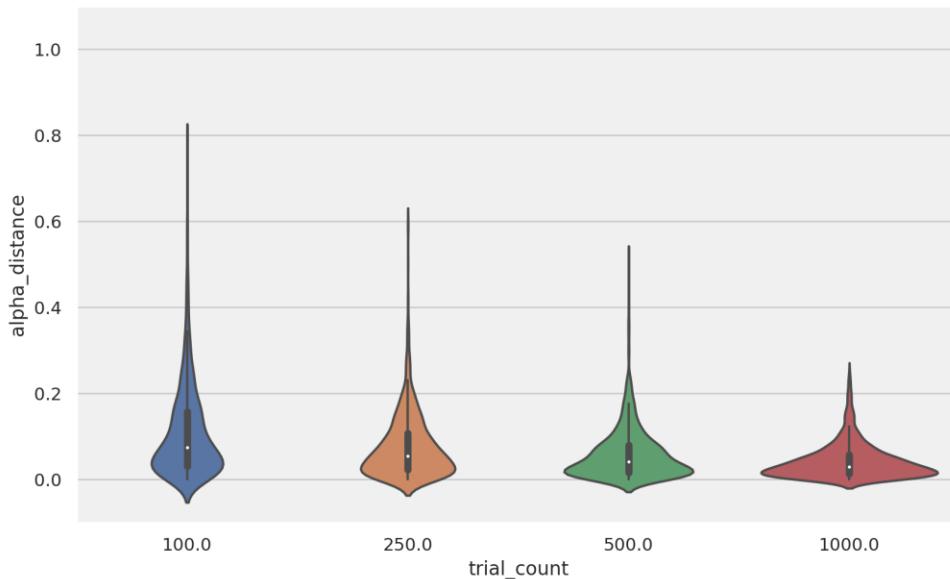


Figure 3.10 Violin plots showing distance between simulated α parameter and fitted α on y-axis and on the x-axis our categories for different number of trials is shown.

In Figure 3.10, it becomes clear that depending on our purpose, 500 trials may still be acceptable if we are okay with a risk of alpha estimates being up to 0.5 away from the target. For 250 trials, we risk being 0.6 away from the target, which is obviously quite a lot when alpha can only vary between 0 and 1. For 100 trials it is almost useless; with the risk of alpha being 0.8 away from its target value we are basically guessing.

Moving onto β parameter, we have R^2 values for 1000, 500, 250, 100 trials as .71, .68, .66 and .59, respectively. These correlation coefficients indicate there isn't that huge of a difference between 1000 and 250 trials, but it drops more when we get down to 100 trials. But again, we can see in Figure 3.11 how visualising the results is much more informative than looking at numbers. There, if we look at the large circles hitting the wall at the right bound, we can see that even at 250 trials, we can still be relatively certain of fit values that are below 5 or so. But at 100 trials this has broken down, and we risk hitting the wall at any value for the β used when simulating.

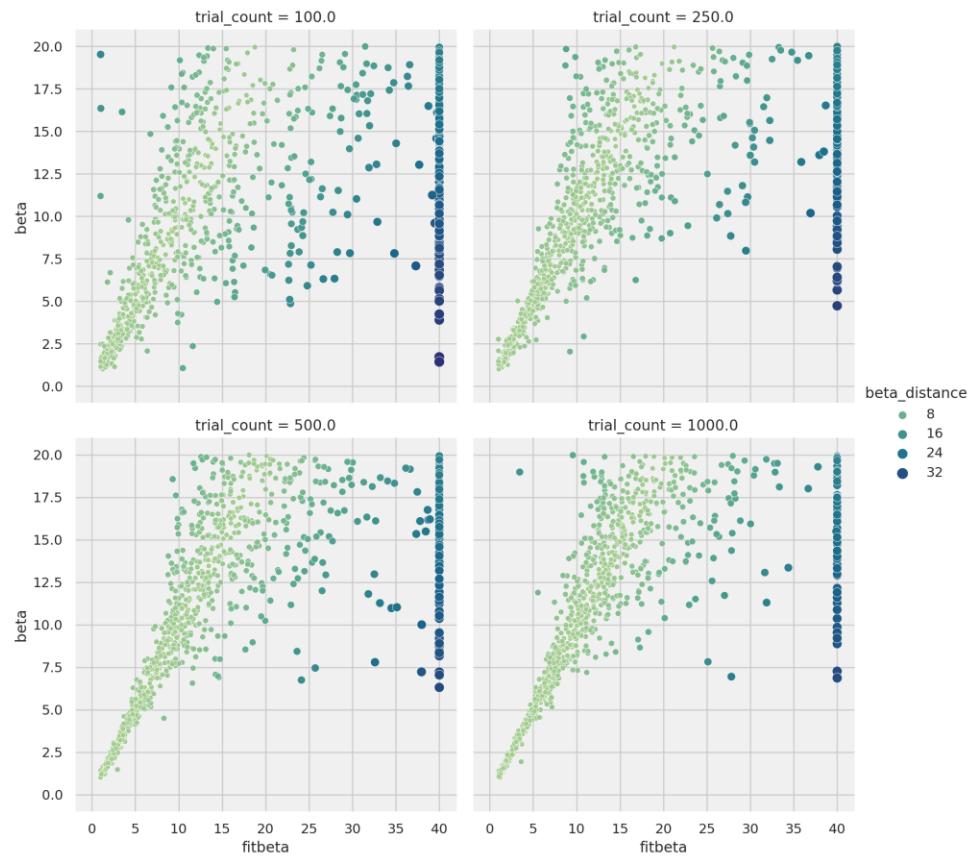


Figure 3.11 Correlation plots for β parameter where fitted values are on the x-axis and simulation values on the y-axis. Colour and size represent distance between simulated and fitted value. Note that x-axis is double the range than the y-axis. Top-left: 100 trials. Top-right: 250 trials. Bottom-left: 500 trials. Bottom-right: 1000 trials.

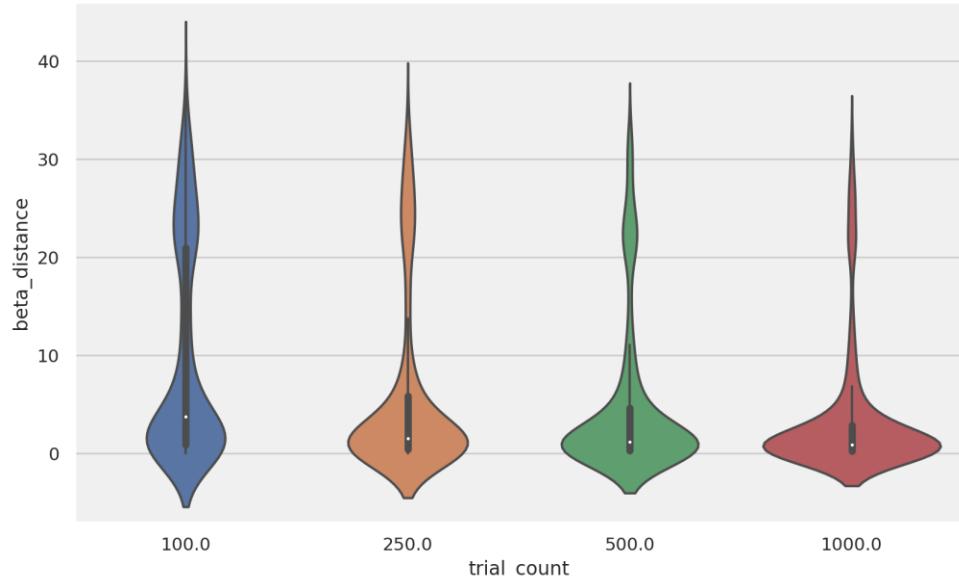


Figure 3.12 Violin plot showing distance between simulated and fitted β on y-axis and number of trials as categories on the x-axis.

This pattern is further emphasised by looking at the violin plot in Figure 3.12 where a big part of estimated β distances for 100 trials are 20 away from the target, which is the entire range of values used for simulations. Just as for α , 100 trials are simply not enough.

We should not forget that our parameters α and β are not isolated, they are used in combination. There are multiple ways we could visualise this, but difficult to include distances in reasonable ways since the two parameters are on such different scales. We could normalise β to be between 0 and 1 but this tends to distort the differences in our experience and makes it more difficult to distinguish results between different number of trials. One way, inspired by [291], is to select a certain criterion for α , and categorise results based on that. For example, we can say that all $\alpha_{distance} > 0.2$ are "bad" and mark them as such when plotting β .

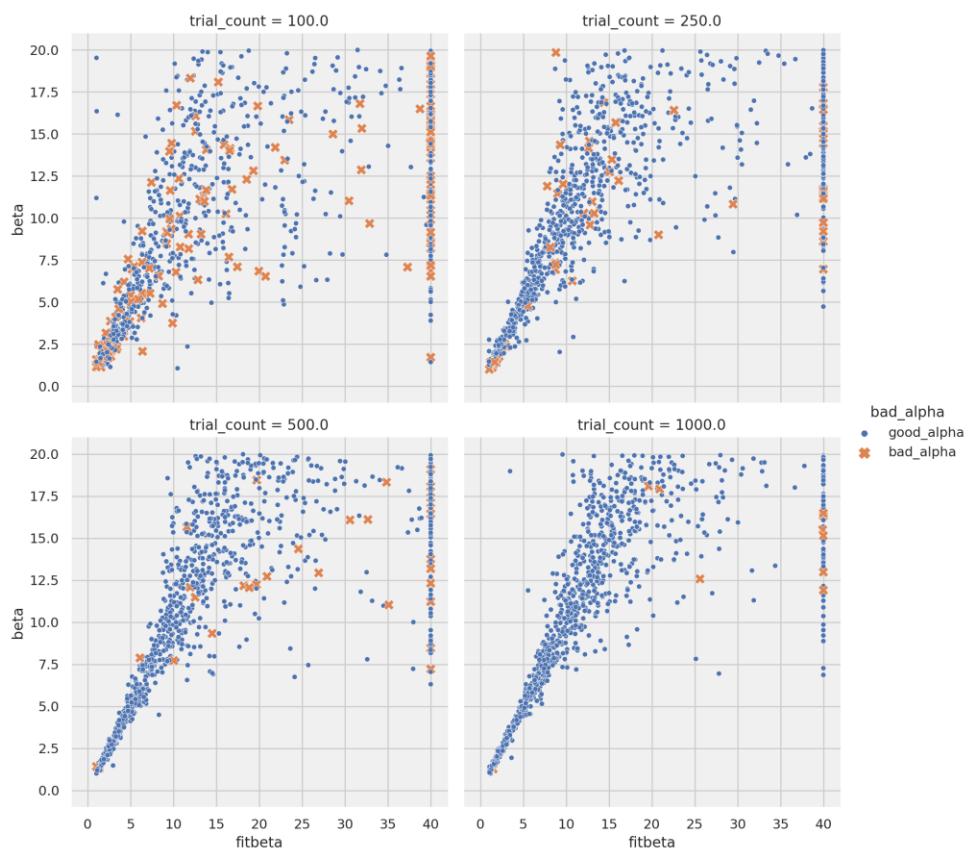


Figure 3.13 Correlation plots for β parameter with fitted values on x-axis and simulation values on y-axis. Orange X marks cases where the distance between fitted α and simulated α is more than 0.2. Top-left: 100 trials. Top-right: 250 trials. Bottom-left: 500 trials. Bottom-right: 1000 trials.

Looking at Figure 3.13, we can see that estimates for both parameters get increasingly worse as we lower the number of trials, not just the two parameters individually.

Drawing some conclusions from the results just presented, we can nicely see how fewer trials make it more and more difficult to recover parameter values. This is perhaps not a surprise, and completely obvious, and therefore rarely mentioned in the literature.

But even with 1000 trials, β recovery can still "hit the wall" when we have $\beta_{sim} > 6$ or so. This is because, as mentioned earlier, with higher β , SoftMax will assign higher probability of picking the action with slightly higher Q-value, causing the agent to act more monotonously. So, after a certain point of increasing β value, there's not much difference between, say, $\beta = 7$ and $\beta = 40$, because in both cases the agent acts similarly – always picking the action with slightly higher Q-value. And with fewer trials, we can see that we risk hitting the wall for pretty much any β value.

Furthermore, it looks like 250 trials is at the lower end of where we can still be reasonably certain of our result, if we get $\beta_{fit} < 5$. Then again, we can see in Figure 3.13 that already at 500 trials we get some bad α s among the low and well fitted β s.

[3.7.1 CONFIDENCE FOR INDIVIDUALLY FITTED PARAMETER VALUES](#)

We have already touched upon this in the above investigations, but another important question here is: given a fit result for an individual simulation, how certain can we be of the results?

To answer this question, we will use the same data as in the previous section. Here we add a distance calculation that is not the absolute value, in order to see a fuller picture of how our estimates are distributed. In other words, we are subtracting the real parameter value from the estimated one, to allow our histograms to have overestimated values on the right and underestimated values to the left. We will first present results for α followed by those for β .

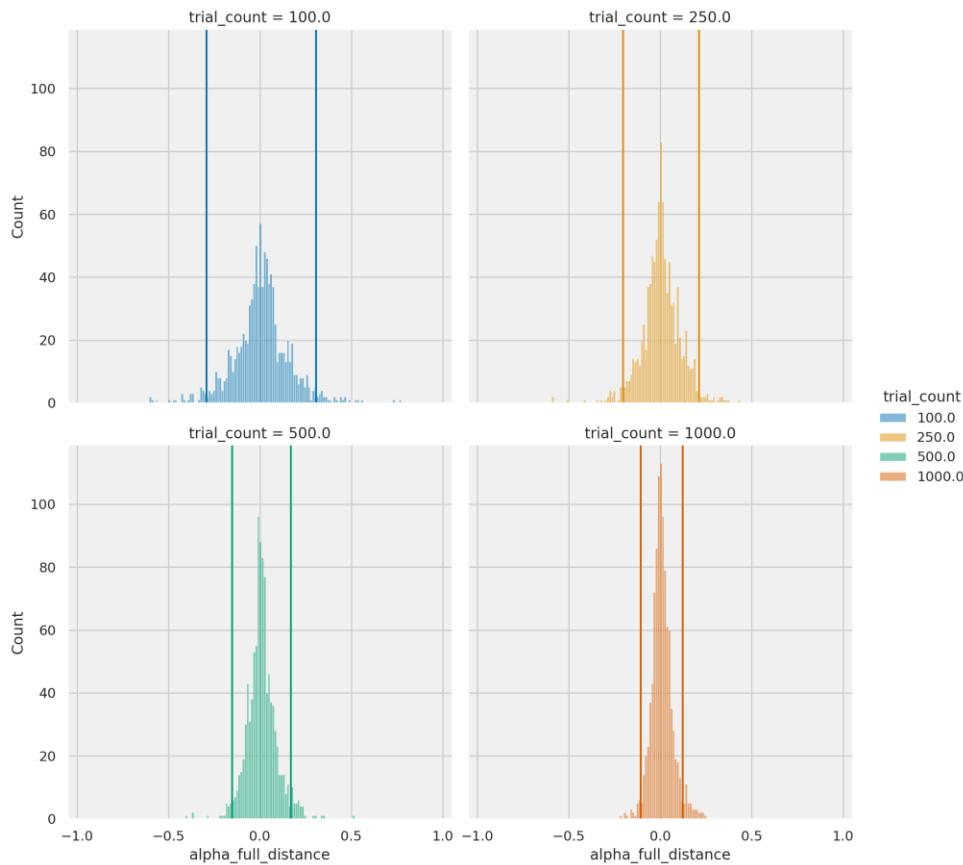


Figure 3.14 Histograms showing the distribution of distance between fitted α values and the α values used for simulations. Vertical lines show the 95% confidence interval. Top-left: 100 trials. Top-right: 250 trials. Bottom-left: 500 trials. Bottom-right: 1000 trials.

For α we find 95% confidence intervals for 1000, 500, 250 and 100 trials to be $(-0.11, 0.12)$, $(-0.15, 0.17)$, $(-0.20, 0.21)$ and $(-0.29, 0.31)$ respectively. The distributions for α are shown in Figure 3.14.

For β , the 95% confidence intervals for 1000, 500, 250, and 100 trials are $(-11.5, 16.3)$, $(-13.1, 20.8)$, $(-14.2, 23.8)$, and $(-14.8, 31.9)$ respectively. The distributions for β are shown in Figure 3.15.

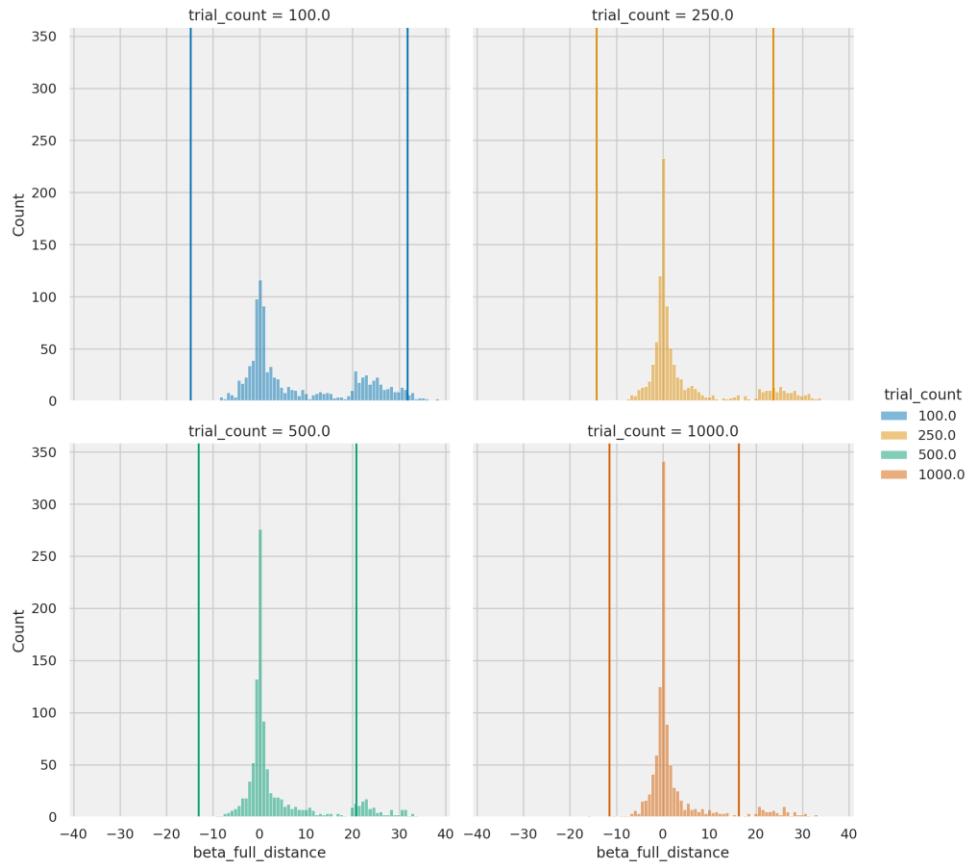


Figure 3.15 Histograms showing the distribution of distance between fitted β values and the β values used for simulation. Vertical lines show 95% confidence intervals. Top-left: 100 trials. Top-right: 250 trials. Bottom-left: 500 trials. Bottom-right: 1000 trials.

For α the story is straightforward; we generally get decent fits for this parameter even at lower trial counts. For β however, the story is more complicated. Overall, it's common that β is overestimated, and at lower trial counts this becomes very apparent. However, we could also see in Figure 3.11 that below certain fitted values for β , there's a much better correlation with the real value. What if we filter our results based on this knowledge? We could for example only look at cases where we have $\beta_{fit} < 10$.

Doing so gives us 95% confidence intervals for 1000, 500, 250 and 100 trials that are $(-2.6, 2.2)$, $(-2.6, 2.1)$, $(-3.6, 3.0)$, and $(-5.0, 4.1)$. These look much better, but the downside is that we have now thrown away more than 50% of our data, even in the 1000 trial case. This ratio is somewhat misleading as it simply looks at how many cases we have of $\frac{\beta_{fit} < 10}{1000}$ for each trial count category. It could be less in the case of a real data set. But it could also be more. We have no idea when it comes

to real data, and that is assuming we can be completely certain the model we are using is correct in the first place!

3.7.2 STANDARD ERRORS WITH THE HESSIAN

There is another possible way to get a measure of uncertainty around our parameter estimates by utilizing the Hessian [64]. This is a square matrix with a row and column for each parameter representing the second order derivative of the likelihood function with respect to the (here, two) parameters. In other words, it's a measure of how steep the hill or valley of our likelihood function is around the parameter coordinates, as seen in Figure 3.8. The square roots of the Hessian's diagonal terms are the standard errors for the parameter estimates, α_{se}, β_{se} in the case of QL_2 .

Unfortunately, this seemingly excellent tool doesn't work well in practice. One issue is that since the Hessian measures the slope, this slope gets truncated for parameter estimates at the bounds of our search. Without those bounds, we run the risk of getting computational errors.

For the data shown in Figure 3.7, we get R^2 values for the correlation between $\beta_{distance}, \beta_{se}$ as 0.099 and for α we get 0.12. For β we also get many values for the standard error that are over 100 and one that is 3622. One would imagine the last example is an estimate that displays the issue mentioned above, where the Hessian gets truncated at the boundaries, but it's in fact a case where the estimate is very good, with $\alpha_{distance} = 0.0002, \beta_{distance} = 1.47$.

3.8 RECOVERY QUALITY FOR VARIOUS ARM REWARD PROBABILITIES

How does reward probability differences between the two arms impact MLE's ability to recover the parameters? We will investigate this similarly to how we did above for different trial counts; we simulate 1000 agents each for a set of "arm reward differences". For convenience, we assume $p_{arm1}(r) + p_{arm2}(r) = 1$. We will investigate four arm differences, starting with the one of .6 we have been using above, and decreasing to 0.4, 0.2 and 0.

For α , R^2 for 0.6, 0.4, 0.2 and 0.0 difference in arm reward probability is .98, .99, 0.99, and 0.99, respectively. Equivalently for β we have 0.71, .75, .74, and 0.70. However, as seen in Figure 3.16 to

Figure 3.19, scatterplot variance decreases together with arm reward probability difference. In other words, smaller arm difference leads to better recovery.

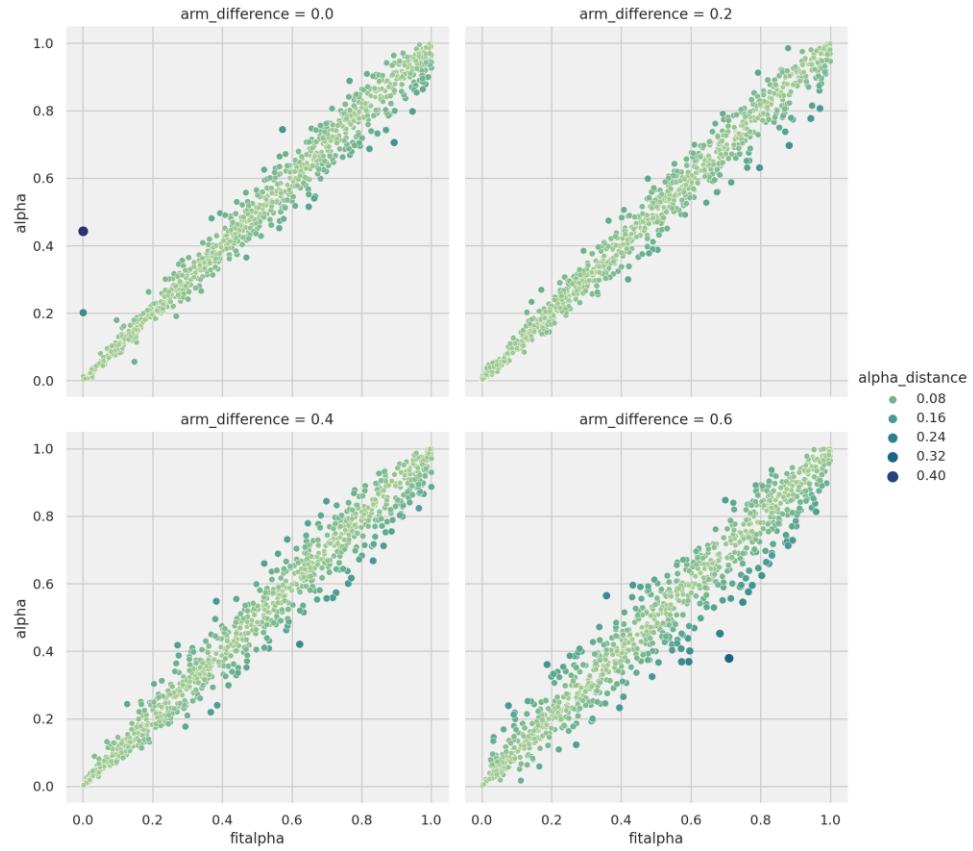


Figure 3.16 Correlation plots for fitted α on x-axis and simulation α on y-axis. Colour and size represent the distance between fitted and simulated value. Top-left: Arm difference 0. Top-right: Arm difference 0.2. Bottom-left: Arm difference 0.4. Bottom-right: Arm difference 0.6.

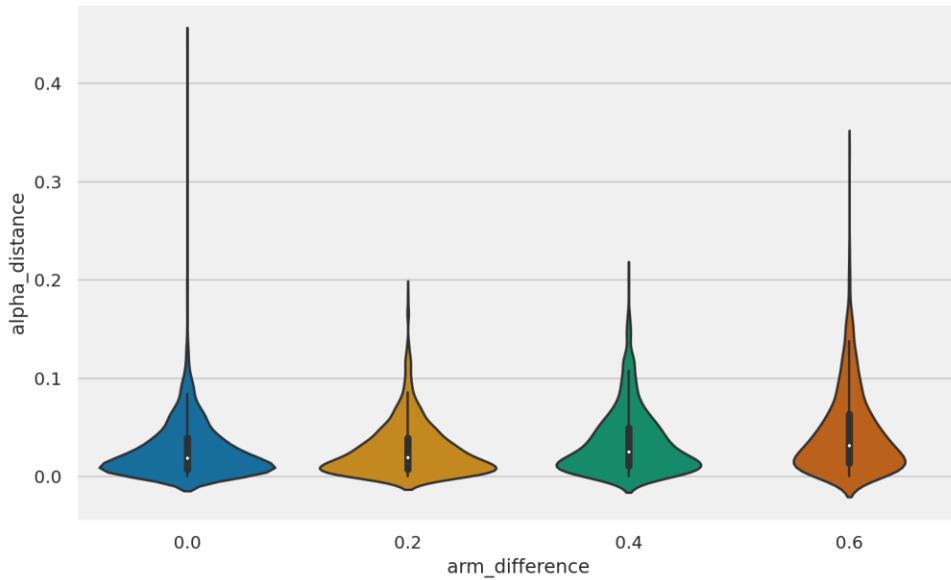


Figure 3.17 Violin plots showing distance between fitted and simulated α on y-axis for each arm reward probability difference on the x-axis.

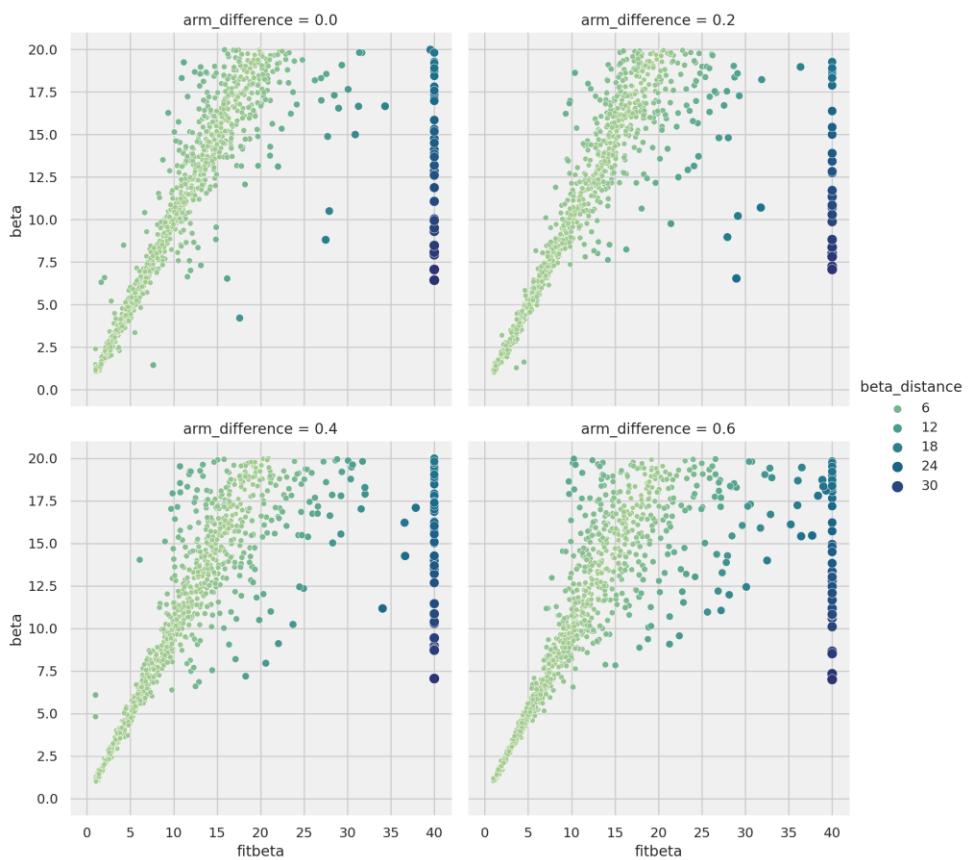


Figure 3.18 Correlation plots for simulated β on y-axis and fit β on x-axis. Note x-axis is double the range of y-axis. Colour and size represent distance between fitted and simulated value. Top-left: Arm difference 0. Top-right: Arm difference 0.2. Bottom-left: Arm difference 0.4. Bottom-right: Arm difference 0.6.

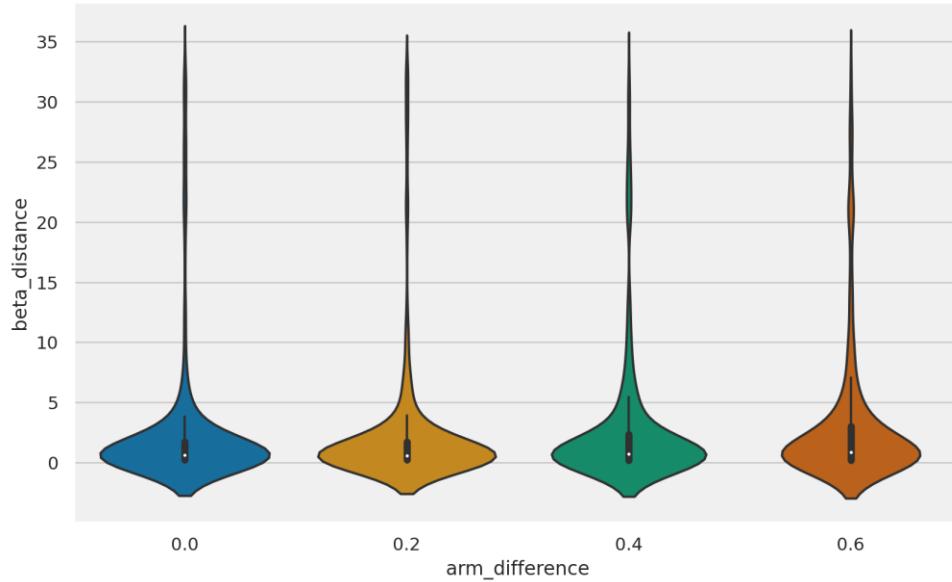


Figure 3.19 Violin plots showing distance between simulated and fit β on y-axis and arm difference categories on x-axis.

These results may seem counterintuitive, how can recovery be better with less difference between the arms? As mentioned in the previous section, as β increases, the agent behaviour becomes greedier; always picking the action with higher Q-value even if the Q-value difference is small. If, additionally, one arm is clearly rewarded more often than the other, the combination leads to one-sided behaviour where almost all actions are the same throughout all trials.

When the reward probability of the arms are both 50/50, the Q-values also become more equal, and we can have higher values of β before behaviour becomes one-sided. In other words, the more varied the actions are, the more granularity exists in the data, and more information can be extracted - thus leading to better parameter estimates overall.

3.9 COMPARING MLE TO BAYESIAN INFERENCE

As we have seen, MLE is very fast and often good enough, depending on what our goal is for parameter recovery and the details of our task such as number of trials. Unfortunately, our likelihood functions are often not well behaved, as demonstrated by visualising likelihood surfaces for different data in Figure 3.8.

Bayesian inference is often seen as better suited for this type of problem [2, 204, 226, 279, 289]. The main reason for this can be

explained by going back to Equation 3.3, where we explained how MLE assumes we ignore the prior probability of the parameter values. In Bayesian Inference (BI), the prior is explicitly used to impose a structural boundary for the posterior distribution of the estimated parameter values. In a way the prior is similar to how we use hard boundaries in MLE, but by using non-uniform continuous distributions for the prior these boundaries are less like walls and more like outskirts of probability in an actual statistical distribution. However, the more data we have, the more that data influences the shape of the posterior distribution and thus the less influence the prior has.

Mathematically, we should here provide the full form of Bayes' Rule, with m for model and d for data:

$$p(m|d) = \frac{p(d|m) p(m)}{p(d)} \quad 3.6$$

Above, the left-hand side is called the posterior distribution, the first term in the numerator on the right-hand side is called the likelihood, the second term is the prior and the denominator is called the marginal likelihood or evidence. All these are distributions, and this is the advantage of Bayesian inference – since we work with distributions, we automatically get a measure of the uncertainty of our calculations. In addition to the common measures of such uncertainty such as the distribution mean and standard deviation, a common measure is the highest density interval (HDI). It can be applied to any distribution and specifies an interval that spans a certain percentage of the values of the distribution, say 95%. For a regular normal distribution with mean μ , the 95% HDI would thus stretch across the 95/2% values to the left of the mean up and including the 95/2% to the right of the mean.

The formula in Equation 3.6 can be analytically calculated only for relatively simple distributions. The main reason being that the marginal likelihood will, for continuous distributions, become intractable. And even with modern computers, calculating it numerically also becomes infeasible for larger models with many parameters. There are two main approaches to overcoming this issue, Markov Chain Monte Carlo (MCMC) and variational inference (VI). We will not go into much detail on these methods here, as that is not

our focus and these methods, especially VI, are actively researched. For more in-depth explanations see [138] (MCMC) and [25] (VI).

MCMC exploits that the marginal distribution has a normalising effect and is thus not strictly needed to find the posterior distribution. Instead, a large number of random numbers are generated from the posterior distribution, i.e., sampled. These are then evaluated with the likelihood and prior and kept if they pass a test for acceptance, otherwise not. There are multiple algorithms for how the posterior is sampled and what test is made for accepting or rejecting the samples. Among the most used ones is the Metropolis-Hastings algorithm, where a new sample is generated based on the current sample (thus fulfilling the Markov property mentioned in a previous chapter). Samples are then accepted based on an acceptance ratio calculated from the likelihood and prior. Regardless of the specific MCMC algorithm, thanks to the magic of randomness, with a large enough sample we get a decent approximation of the posterior distribution.

The alternative, VI, involves selecting candidate distributions for the posterior and then minimising the distance between the candidate and the true posterior. This approach turns the problem into an optimisation problem, allowing the use of existing optimisation techniques to find the posterior.

Generally, MCMC is slower but more accurate while VI is less accurate but faster [249].

Another advantage of Bayesian methods is that we can create so called hierarchical models. In a regular Bayesian model we have, say, one parameter θ in our model and some prior pr on what that parameter value might be. In a hierarchical model, the values defining the pr distributions are themselves drawn from a distribution, at a “higher level” so to speak.

To provide a more practical example, consider our Q-learning model with two parameters α and β . We have conducted an experiment with several participants all doing the two-armed bandit task and we now have some data to analyse. We may have some priors for what the values of α and β may be for individual participants, but would it not also be reasonable to believe that all the participants share some similarity? Well, not necessarily, as explained above with regards to individual differences and strategies. But for the sake of argument, maybe it is the case that all our participants’ α

value are distributed normally around some mean? We can include that belief in our model, by having all participants share a common hierarchical prior for the normal distribution of α . Every single participant's α is then drawn from the common distribution.

The advantage this grants us is not only the ability to test the hypothesis that participants do share some commonality, but in a way, we also gain access to much more data for each single participant. The hierarchical model uses the data from all participants to find the common α prior, which can then inform the estimates for individual subjects. This is a huge advantage when we are dealing with experiments where we often have limited data sets.

A potential downside of all participants sharing information is so called "shrinkage" [138]. Because the α of our individual participants are drawn from a common distribution, this means their individual estimates will be "pulled" closer together. This is not necessarily a problem but depends on how specific or narrow our prior distribution is and how much data is available. With few datapoints the prior will have larger impact on the posterior and final parameter value estimates may be misleading. We will see examples of this below.

3.9.1 BAYESIAN MODELLING TOOLS

There are many tools available for different programming languages to make it easier to construct and analyse Bayesian models. One such tool is Stan [250], which allows for defining models in .stan files that can either be run directly from the command line or used in conjunction with interfaces from several different languages such as Python (CmdStanPy¹³) and R (RStan¹⁴). On top of these tools, more fully featured toolboxes have been built, one of which is hBayesDM [2], an R package that comes with several models and tasks for decision making built in. The models in hBayesDM are written in Stan. In our investigations below, we will specify when we have used models from hBayesDM, otherwise we have made our own using Stan and CmdStanPy.

A methodological note here is that when sampling using MCMC, we can sample using several "chains". This is, simply stated, doing

¹³ <https://github.com/stan-dev/cmdstanpy>

¹⁴ <https://github.com/stan-dev/rstan>

the sampling process multiple times separately. We could run one sampling process for a very long time and be somewhat certain we reach a good approximation. But by using several chains that are separately initiated with random starting points, we can use fewer samples for each and afterwards check if all our chains have converged to approximately the same answer.

3.9.2 FITTING BAYESIAN MODELS

The downside of MCMC, especially for hierarchical models, is that for larger sets of data, computation can be time consuming. In the dataset we will use for comparisons below, we have 50 artificial subjects playing the two-armed bandit task, each doing 1000 trials.

Ideally, we would use 10-20 times more subjects for an investigation such as this, like we did for the MLE investigations. But this dataset takes our hierarchical MCMC model around two hours to complete the fitting process. Technically, Stan allows for splitting up data “inside” a model so that one chain can be run across multiple CPU cores, but that requires fiddly, and thus error-prone, re-coding of the Stan models themselves. Thus, since four chains are generally enough to identify divergences and more chains do not decrease computation time nor increase accuracy, there is unfortunately no direct way to improve computational time other than having faster single core CPU performance. Besides, in many older psychological experiments, or neuro-imaging studies, 50 subjects or fewer are not unusual so it's not an unreasonable number. Having 50 humans do 1000 trials in a two-armed bandit task is arguably less reasonable.

Our dataset thus consists of 50 artificial subjects all using the QL2 algorithm to play the two-armed bandit task. Each subject's parameters were drawn randomly:

$$\alpha \sim U(0, 1), \beta \sim U(1, 20)$$

The task used 1000 trials and had arm reward probabilities:

$$p_{arm1}(r) = 0.2, p_{arm2}(r) = 0.8$$

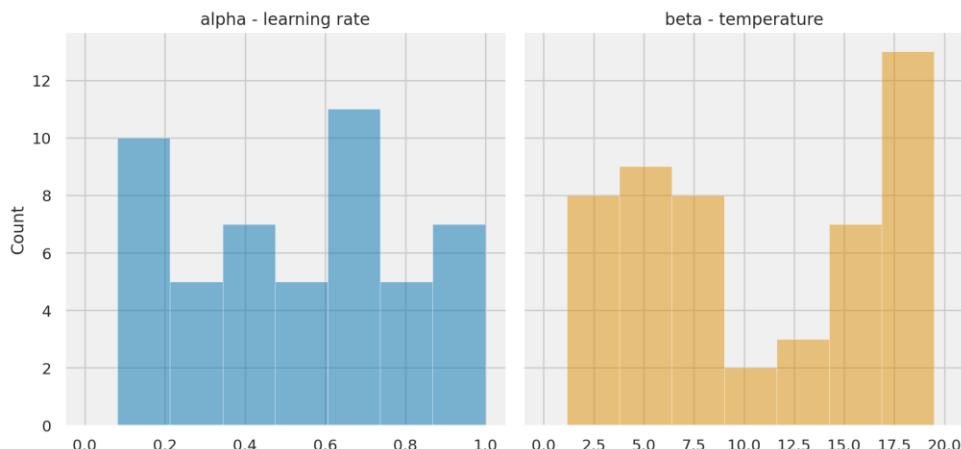


Figure 3.20 Histograms showing the parameter value distributions for the 50 subjects in our dataset. Left: α parameter. Right: β parameter.

In Figure 3.20 we have plotted the parameter value distributions for the simulated participants of our dataset. We reran the 50 simulations a few times to get somewhat even distributions across parameter values¹⁵.

In the following analyses, different trial counts use the same dataset. Instead of doing new simulations for each trial count, we use the first x trials of the data for each of 1000, 500, 250 and 100 trials. The three methods compared are:

- MLE (same likelihood as used above) fitted to individual subjects (called MLE below). This model uses boundaries for fitted parameters as $0 < \alpha < 1$ and $1 < \beta < 40$.
- Bayesian MCMC model fitted to individual subjects (called IND-BI below, adapted from the hBayesDM hierarchical model below). This model uses priors for fitted parameters as $\alpha \sim U(0, 1)$ and $\beta \sim U(0, 50)$.
- Bayesian MCMC hierarchical model fitted to entire dataset (called HIER-BI below, model from hBayesDM, modified for wider β). This model uses normal priors for fitted parameters $N(0, 1)$ that are then transformed to the ranges $0 < \alpha < 1$ and $0 < \beta < 20$.

The observant reader will notice there are inconsistencies in our use of priors and boundaries for the three methods. This is done to

¹⁵ This particular dataset can be found in CSV format in the thesis code repository

emphasize the differences in the model fitting results and discuss the strengths and weaknesses of each method.

The HIER-BI model is included in the hBayesDM package and called “2 arm bandit delta”. As mentioned above, we have done a slight adjustment to their model, which is to increase the maximum possible β (called τ in the hBayesDM model) to 20 from an initial value of 5. This low limit has most likely been set due to the reasons we saw in previous sections, where increasing β s are increasingly difficult to fit. But it is our opinion this is not ideal. We do not know what distribution underlies our data, and it is one thing to assume a normal distribution but quite another to assume bounds for that distribution. The data should guide our posteriors, not the bounds of our priors. Nevertheless, the hBayesDM model is equivalent to our *QL2* model, except for the former’s hierarchical nature. IND-BI is partly adapted from this hierarchical model into one suitable for individual fits. The code for these investigations can be found in the thesis repository¹⁶.

¹⁶ https://github.com/fohria/phd_thesis

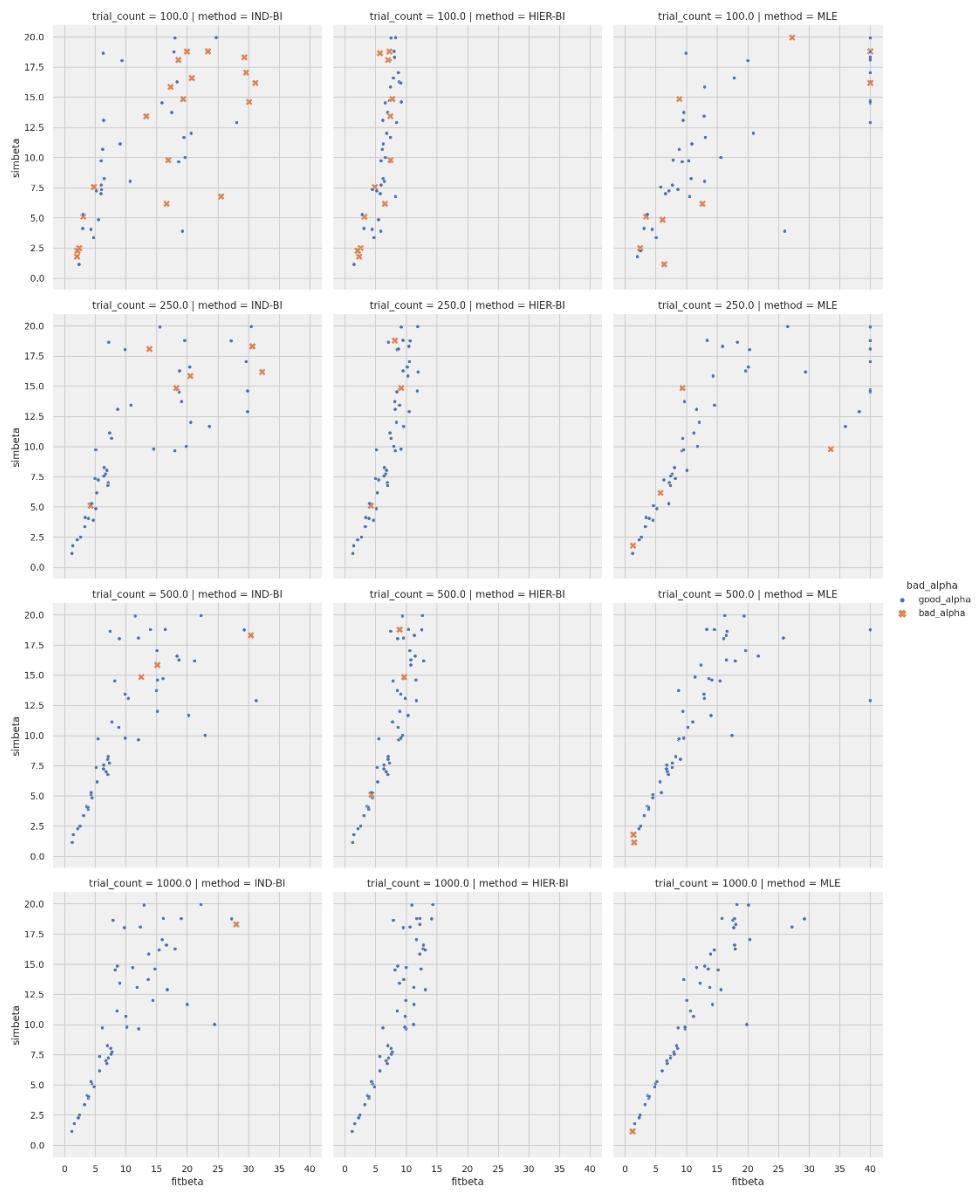


Figure 3.21 Correlation plots for fitted β on x-axis and simulation β on y-axis. Bad α are marked with orange crosses and such as have a distance of more than 0.2 between fitted and simulated α . Rows are, from top to bottom, 100 trials, 250 trials, 500 trials, and 1000 trials. Columns show different methods, where left column is IND-BI, middle column is HIER-BI and right column is MLE. See text for further explanation.

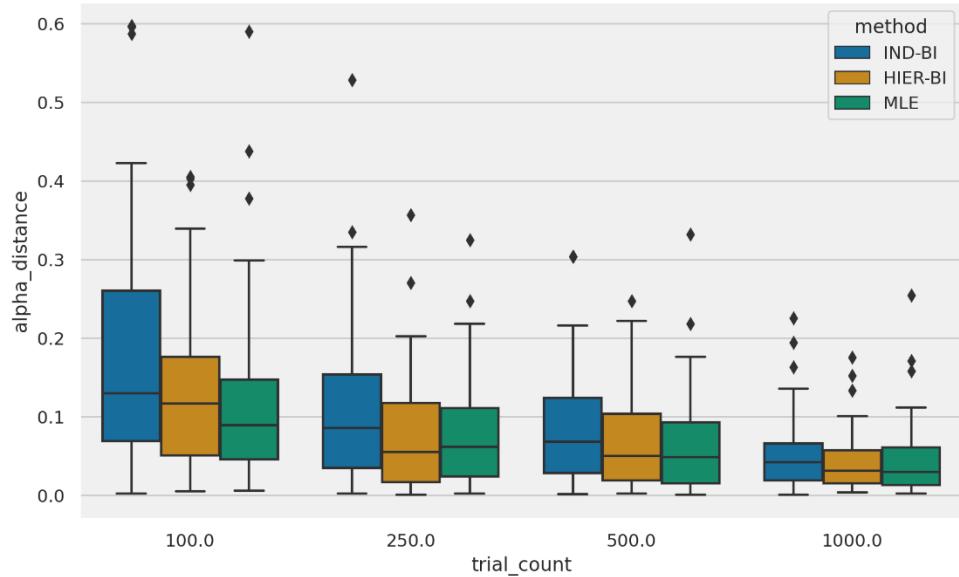


Figure 3.22 Boxplots showing distance between simulated α and fitted α on y-axis, with number of trials on x-axis. Colours indicate method, as seen in legend.

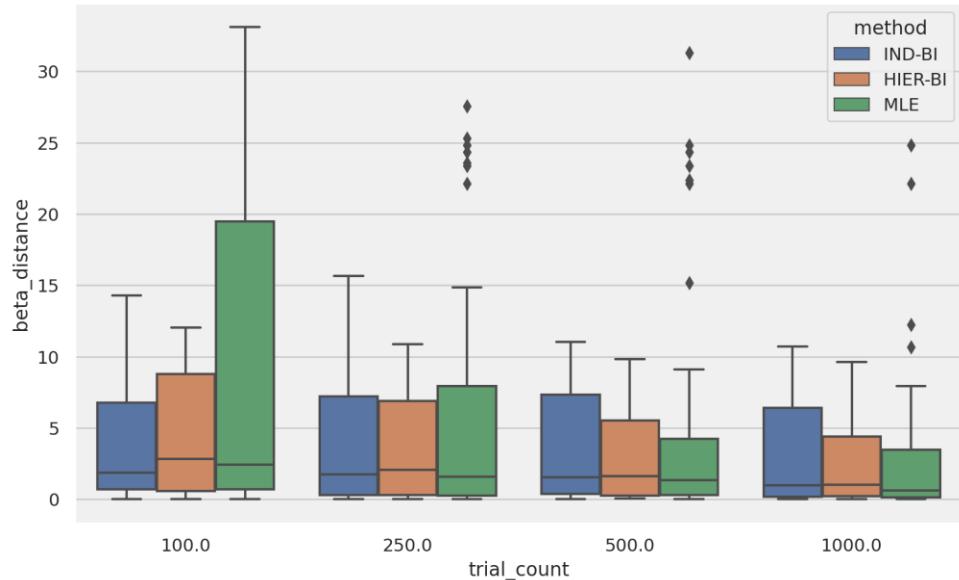


Figure 3.23 Boxplots showing distance between simulated β and fitted β on y-axis, with number of trials on x-axis. Colours indicate method, as seen in legend.

Results are shown in Figure 3.21 to Figure 3.23. What may be surprising here is how well MLE holds up, until we get to 100 trials where MLE cannot keep up with the average β distance in Figure 3.23. This is not surprising in the case of HIER-BI as its prior is set to a maximum of 20 for β , but IND-BI has a wider range of possible range values than MLE yet is able to maintain a lower average distance. Also, and as is further discussed below, HIER-BI has lower values in

general due to shrinkage. However, had we used a boundary of 20 for the MLE, its results would look much better in comparison. This could be reasonable, if we have some knowledge about possible ranges for the data we are fitting. Values for β higher than 20 also have diminishing returns, as we saw in the behavioural studies above, where even between $\beta=10$ and $\beta=20$, behaviour was close.

Looking further, again we find that our summary boxplots (statistics) are not showing the bigger picture. If we look at the middle column of Figure 3.21, we notice how fitted values for β seem to get stuck at an upper limit for HIER-BI. This is because the model assumes a normal distribution for the group-wise distribution from which individual β values are drawn. We mentioned this shrinkage phenomenon above; the advantage of hierarchical models is that we essentially combine the data from all subjects, but the downside is that if the group of subjects is heterogenous, our results may be misleading.

In the same figure and column, we can also see how increasing amounts of datapoints (going from a low number of trials to large), extends the limit for fitted β values towards 15 for 1000 trials and towards around 10 for 100 trials. This exemplifies how priors are much more impactful with little data. At higher trial counts, we get a wider spread of values, which happens because the more data we have the less impactful is the prior we have set.

Importantly, the constriction of individual parameter values towards the group mean is – as alluded to above – what is behind the seemingly low distances for β estimates for HIER-BI seen in Figure 3.23. The estimated mean for the group level normal distribution was around 8.2, so estimates automatically become decent (compared to the other two methods). This is not necessarily a problem, and quite a reasonable assumption when looking at data occurring in nature, which is often normally distributed. Ideally the mean would have been around 10 instead, to allow for more values towards 20.

But crucially, this assumes that all subjects we include when fitting hierarchically used the same model for making decisions. If not, the hierarchical model may not be very useful at all, and potentially be more difficult to interpret. We will come back to that topic in a later section.

3.9.3 HIERARCHICAL MCMC COMPARED TO VBI

As mentioned above, VBI is faster than MCMC but generally seen as less accurate [25]. Using the same data and methods as in the previous section we here compare MCMC and VBI. Again, as mentioned above, MCMC with all 1000 trials takes around two hours on a modern laptop, whereas VBI takes just a few minutes. In both cases we use the same modified 2-armed bandit Stan model in hBayesDM as described in the previous section.

For both α , β estimates, the two methods are overall comparable, with value estimates correlating between the methods with an R^2 of .99 for all trial count categories. T-tests show no significant difference between the methods for any trial count category. In Figure 3.24 we can see that, especially for β , MCMC has slightly better estimates overall, but if we take worst case scenarios into account (top T part of the box plots) then there is indeed not much of a difference between the methods.

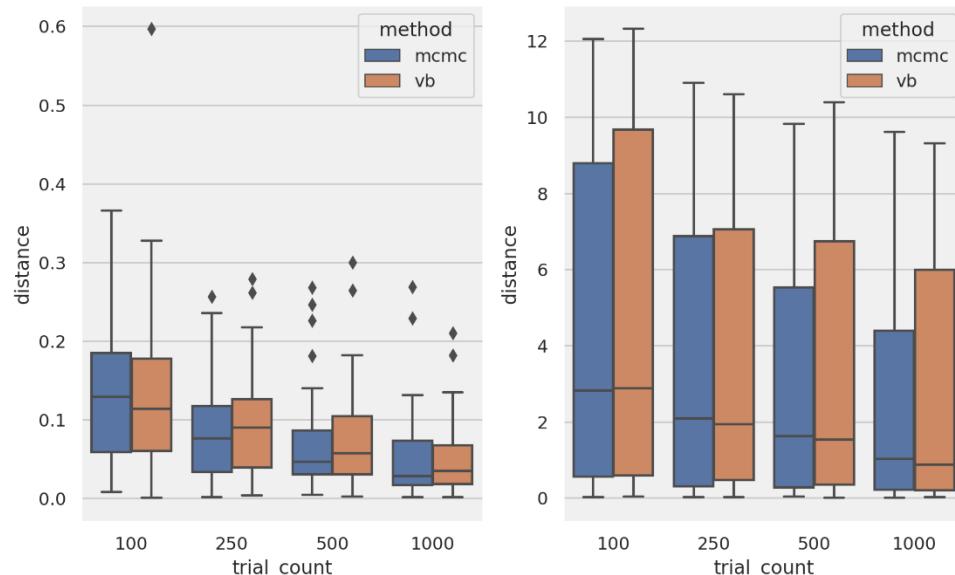


Figure 3.24 Box plots showing the distance between simulated parameter and fitted parameter on y-axis with each trial count category on the x-axis. Colours indicate method as per legend. Left: α parameter. Right: β parameter.

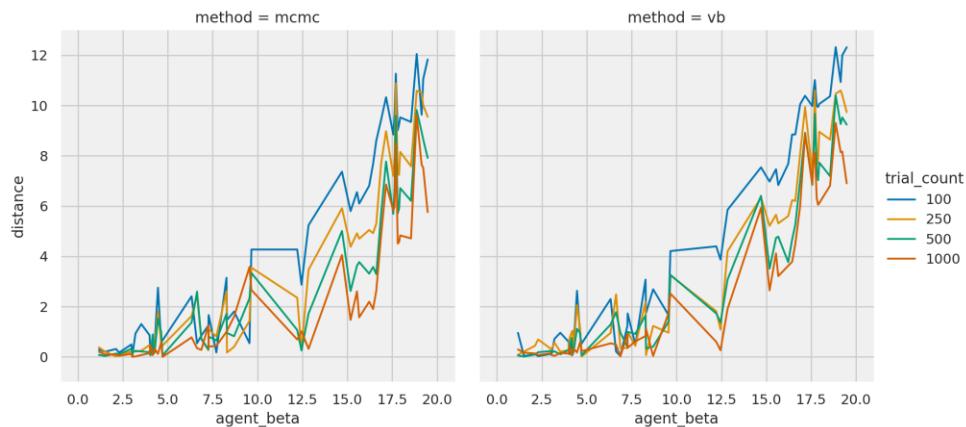


Figure 3.25 Line plots showing distance between simulated and fit β value on the y-axis and simulated β value on the x-axis. Different coloured lines indicate trial count categories, as per legend. Left: MCMC. Right: VBI

Looking into further detail at β estimates – since we know from our earlier investigations these are more problematic than those for α – we can see in Figure 3.25 that the two methods also show similar patterns for increasing simulation β versus distance to fitted value. Here there is somewhat more noticeable differences between trial count categories for MCMC method, whereas VBI holds up well even at 250 trials. Another way to look at it is that it looks like MCMC is better helped with increasing number of trials than VBI.

Both Bayesian methods include standard deviations of the parameter estimates in their results. This comes naturally from calculating distributions instead of point estimates as for MLE. Do the standard deviations provide useful information?

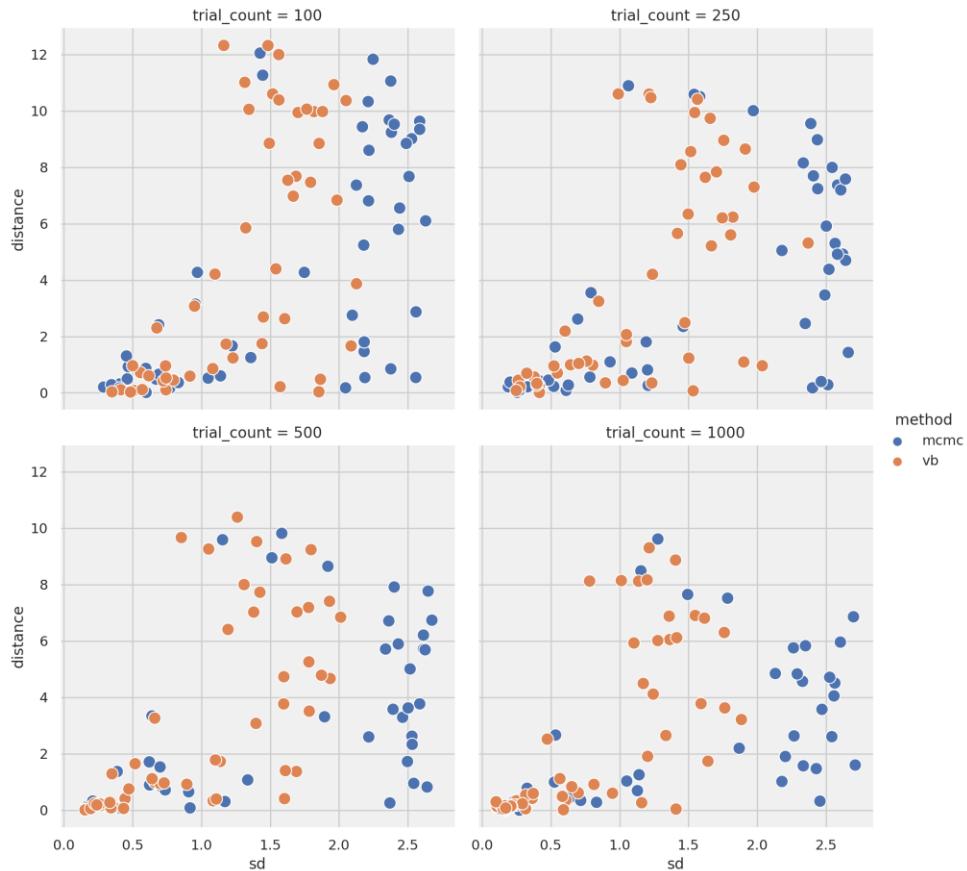


Figure 3.26 Scatterplots of distance between fitted and simulated β on y-axis and SD estimate on x-axis. Top-left: 100 trials. Top-right: 250 trials. Bottom-left: 500 trials. Bottom-right: 1000 trials.

In Figure 3.26 we have plotted β SD values (x-axes) against distance between fitted and simulated β values (y-axes). Unfortunately, SD estimates for either method are somewhat unreliable. For both methods, and across all trial counts, we see that if $SD < 1$, there is – except for a few outliers – a decent chance the distance is below four or so. But for $SD > 1$, there is no longer any connection between SD size and distance.

Finally, we can also check how wide the HDI's are and what the probability is that the real β value falls within this interval, which can be seen in Figure 3.27. There we can see that VBI generally has shorter HDI (left part of the figure), but this is coupled with lower probability of the simulated value being inside the interval. MCMC method generally has higher probability of the real value being inside the HDI, but even at 1000 trials we don't reach even 80%. At 100 trials, both methods are below chance level on whether the HDI captures the real parameter value.

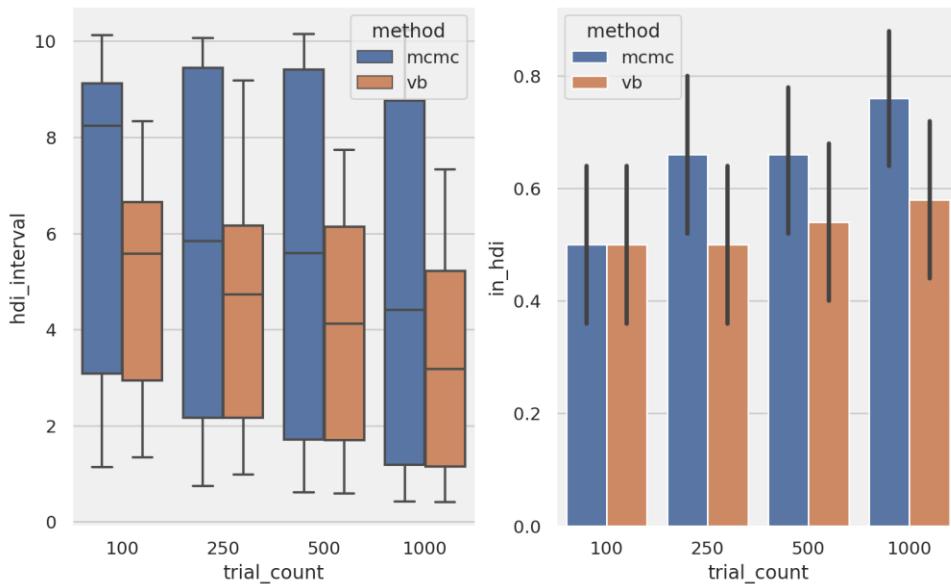


Figure 3.27 HDI check for β estimates for MCMC and VBI methods. Left: HDI length on y-axis and number of trial categories on x-axis. Right: Probability of simulated β being within HDI for each method and trial count category.

What conclusions can be drawn regarding MCMC and VBI? Overall, it looks like the methods are quite comparable. Since MCMC takes roughly 60 times as long to compute than VBI there's little reason to use MCMC. Perhaps MCMC is preferable as a final analysis if correctness of parameter estimates is central to one's goal, as there's a higher chance the parameter value is inside the HDI.

3.10 MODEL COMPARISONS

Until now we have focused on recovering parameters for a specific model. But we rarely know what model best fits our data. The more common scenario – at least for our purposes – is the case where we have several possible models to explain a set of data. Which model best fits our data?

One standard way of comparing models is using the Bayes Factor [130, 147], which compares the evidence for a model m_1 relative to that of a model m_2 , given data d :

$$BF_{12} = \frac{p(m_1|d)/p(m_2|d)}{p(m_1)/p(m_2)} \quad 3.7$$

, where $P(m_1), P(m_2)$ are the priors. If the priors are uniform the expression simplifies to:

$$BF_{12} = \frac{p(m_1|d)}{p(m_2|d)} \quad 3.8$$

Values of $BF_{12} > 1$ provides evidence for m_1 whereas values < 1 provides evidence for m_2 . To decide what BF is “good” evidence, different authors propose varying limits. For example, [147] consider values > 10 or $< 1/10$, respectively, as strong evidence. In cases where we use BF (mainly in later chapters), we follow [64, 290] as per Table 3.1.

| $2 * \log_e BF$ | BF | <i>Evidence</i> |
|-----------------|---------|------------------------------------|
| 0-2 | 1-3 | Not worth more than a bare mention |
| 2-6 | 3-20 | Positive |
| 6-10 | 20-150 | Strong |
| > 10 | > 150 | Very Strong |

Table 3.1 Bayes Factor interpretation for values on log scale (left most column), raw ratio (middle column). Adapted from [64, 130, 277, 284].

Important when speaking of BF, is that in the above equations, the entire parameter space is taken into consideration. If we instead use estimated maximum likelihood estimates, we get a special case of the Bayes factor called the likelihood ratio.

An additional factor to take into consideration is the number of parameters of our models. The more parameters, the more we risk overfitting [64, 291], meaning that with enough parameters we can fit any kind of data perfectly, but it will not say much about the world in general. Compare the statement “everybody likes ice cream” (a gross generalisation) to asking every single person in the world if they like ice cream or not. The latter “model” will fit the world perfectly, but it may not be very useful to describe groups in general.

There are several proposed techniques to get around this, like the BIC, AIC, WAIC and LOO [64, 130, 278, 285]. What most of these measures have in common is that they penalise models based on the number of parameters. We will use the Bayesian Information Criterion (BIC) here to demonstrate how this can work in practice. The BIC is calculated as:

$$BIC = k \ln(n) - 2 \ln(\hat{L}) \quad 3.9$$

Where k is the number of parameters in the model, n is the number of trials (datapoints) and \hat{L} is the maximised likelihood. Lower BIC indicates better fit.

3.10.1 COMPARING MODELS WITH BIC

We will now test and demonstrate model comparisons using the BIC with the two models from above, *QL2* and *RandomBias*. We generate and simulate 10000 random agents with *QL2* playing Bandit task, then fit the resulting data with both *QL2* and *RandomBias* models. Then we simulate 10000 *RandomBias* agents playing the same Bandit task and fit both of our models to that data. For each such simfit case, we calculate the BIC and record which of the models had the best fit according to this information criterion. When they are all done, we summarise the “scores” in a confusion matrix. What we would like to see here, then, is that in the vast majority of cases, the best fitting model is also the model that generated the data.

We are here using MLE for this demonstration, as we will investigate more methods below.

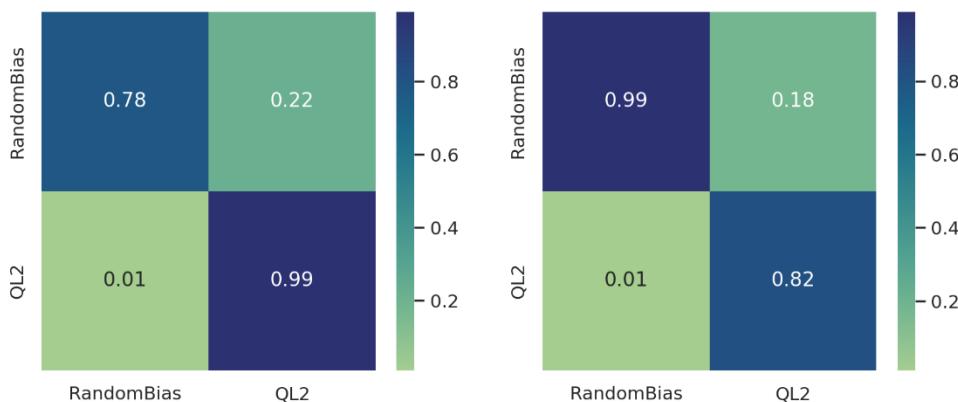


Figure 3.28 Left: Confusion matrix. Row names are the agent used for simulation. Column names are the model fitted. Numbers inside squares are the ratios of the model fits to each simulated model. Right: Inverse confusion matrix. Row names are the agent names used for simulations, while column names are the model names used for fitting. Numbers inside each square indicate the probability that the model fitted generated the data

As we can see in Figure 3.28, left, the *RandomBias* model “fits itself” around 80% of the time and in the other 20% of cases the *QL2* model fits the *RandomBias* data. This is not as curious as it may seem on first appearance, as it is quite likely that a randomly playing agent

that happens to have a certain bias towards one arm fits the behaviour of a greedy QL2 agent. When it comes to QL2, it fits itself in 98% of the cases. This is quite good and promising for future investigations.

When fitting “real” data, i.e., data we do not know how it was generated, we are interested in the question of how likely it is that the best fitting model is the correct one. We can answer this by “inverting” the confusion matrix (Figure 3.28, right). In other words, the confusion matrix shows us $p(\text{fit model}|\text{simulated model})$ and the inverted confusion matrix shows us $p(\text{simulated model}|\text{fit model})$ [291].

[3.10.2 MORE ADVANCED MODEL COMPARISON METHODS](#)

As we saw previously, MLE runs the risk of hitting the parameter value boundaries when fitting. Does this impact the accuracy of BIC calculations and model comparison/selection?

According to [204] it does, and they propose using Bayesian hierarchical modelling to alleviate the situation. Similar arguments are made by [213, 252]. Common for these methods are that the hierarchical thinking is extended to more levels, including a distribution at the top for what model is most likely for the entire population. On the next level down is what model is most likely for each subject, and then for each model what parameter values are most likely for that subject. These levels include shared distributions between subjects for parameter values. The resulting “meta models” are thus quite big, especially if there are many models under consideration. MCMC sampling is thus infeasible and instead VI is used.

This type of model comparison is quite powerful, as we can include Bayesian uncertainty at every level of our investigations. Methodology wise these methods also become quite complex. Technically, these types of hierarchical models can be implemented in Stan, but it would be time consuming and error prone as one would have to code the entire model structure manually. Also, Stan’s VI algorithm is still labelled “experimental”.

Luckily [204] provides a toolbox called Cognitive Bayesian Modelling (CBM) where one only needs to provide the likelihood function for any custom models and then compose the comparison together with few lines of code. Building on the theories of [213, 252],

[203] provides the Variational Bayesian Analysis toolbox (VBAT), a huge collection of tools for designing, performing and analysing experiments. But here it's possible to input measures like the BIC and the toolbox performs the "upper" part of the hierarchical analysis, providing results comparable to those of CBM. Worth noting is that CBM uses a maximum a priori (MAP, essentially MLE with a prior) method called Laplace fitting and uses estimated parameter values from there to initialise its fully hierarchical Bayesian inference (HBI). Both toolboxes are available exclusively for MATLAB.

What these tools can tell us is two main measures. One is the so-called model frequency, measured with Probability Exceedance Probability (PXP), which is a measure of how likely it is that one model is more common among the participants investigated than other models. And second, we get probability measures for each individual subject, how likely it is that each model is the best fit.

This is very important for our purposes, especially in later chapters. Earlier we mentioned heterogeneity of data and how it can be the case that different subjects use different strategies (see Figure 3.2). These toolboxes allow us to investigate such individual differences. The model frequency is also interesting but can be seen as a symptom of how experiments in the cognitive sciences are usually conducted. There are two or more groups in different conditions, and there is an interest to investigate if a certain group exhibits a certain type of behaviour (a certain model is the better fit). If the majority of subjects in a group exhibit this behaviour, the experiment is a success. But what about the subjects of that group that did not behave as the model would predict?

3.10.3 COLLECTED METHODS FOR MODEL COMPARISONS

Armed with these toolboxes, we now have a collection of fitting and model comparison/selection methods. Below we list the methods under consideration and in parentheses after their description indicate what they will be called when reporting results later.

1. BIC value for model fits to individual subjects using MLE (mle)
2. BIC value for model fits to individual subjects using variational Bayesian inference (vbstan)
3. VBAT comparison using BIC value from 1, mle (vbat_mle)
4. VBAT comparison using BIC value from 2, vbstan (vbat_vbstan)

5. BIC value for each subject for model fits with hBayesDM hierarchical models using VI (hbayes)
6. VBAT comparison using BIC value from 5, hbayes (vbat_hbayes)
7. CBM toolbox hierarchical comparison (cbm-hbi)
8. VBAT comparison using log evidence from CBM Laplace fits from 7, cbm-hbi (vbat_laplace)

Because MCMC is time consuming and our investigations above showed that VI provides competitive results, we have dropped the MCMC method from consideration.

[3.10.4 COMPARING MODEL COMPARISON METHODS](#)

We shall now compare the performance of the above-described methods for model comparison and selection. We continue the use of our two models *QL2* and *RandomBias* and create the following datasets we will then fit and contrast using the different methods. All use the Bandit task, with arm reward probabilities as 0.2 and 0.8 respectively.

- 1000 subjects, 1000 trials. 50/50 *QL2*/*RandomBias* (A)
- 1000 subjects, 100 trials. 50/50 *QL2*/*RandomBias* (B)
- 1000 subjects, 1000 trials. 25/75 *QL2*/*RandomBias* (C)
- 1000 subjects, 100 trials. 25/75 *QL2*/*RandomBias* (D)

In all cases, we draw parameter values for *QL2* as per these distributions:

$$\alpha \sim U(0, 1), \quad \beta \sim U(0, 20)$$

and for *RandomBias* as per:

$$bias \sim U(0, 1)$$

Also note that in the below plots for parameter value distances, the methods shown differ from the methods shown in the model selection plots. This is because the VBAT method, as explained above, does not estimate parameters, it uses the BIC value calculated from another method to select models. Thus, only the non-VBAT methods are shown in the parameter value distance plots.

We have now also adjusted MLE boundaries and priors for the Stan-based models (*vbstan* and *hbayes*) to be more comparable. For MLE boundaries are $0 < \alpha < 1$ and $0 < \beta < 50$. For *vbstan* $\alpha \sim U(0, 1)$

and $\beta \sim U(0, 50)$. For hbayes the transformation for β now allows values up to 50. For CBM, we use the recommended settings where all parameters have the same value for the prior variance (6.25).

3.10.4.1 Dataset A

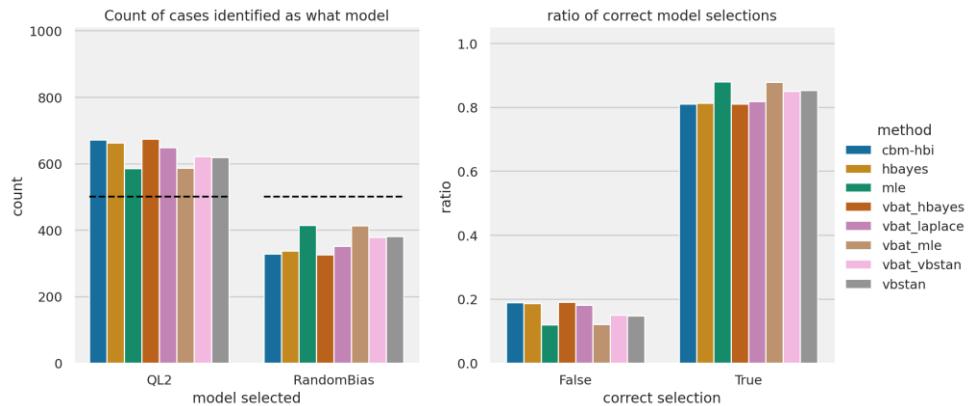


Figure 3.29 Plots showing what model was selected (left) and whether the model selected was correct (right). Each individual bar represents a specific method as per the legend. Left: The black dotted lines show the true number of cases for each model. The closer to the line the better. Right: Higher value for “True” is better. Note this is a ratio measure and not absolute case count.

In Figure 3.29 we can see that no method can correctly identify all cases (left plot in figure). This is due to cases of *RandomBias* with high bias for one arm can behaviourally look very similar to a greedy *QL2* agent, as has been discussed above. What is surprising here is that the MLE method is numerically better than other methods. This despite the fact that individual parameter estimations, as seen below, are commonly less accurate for MLE.

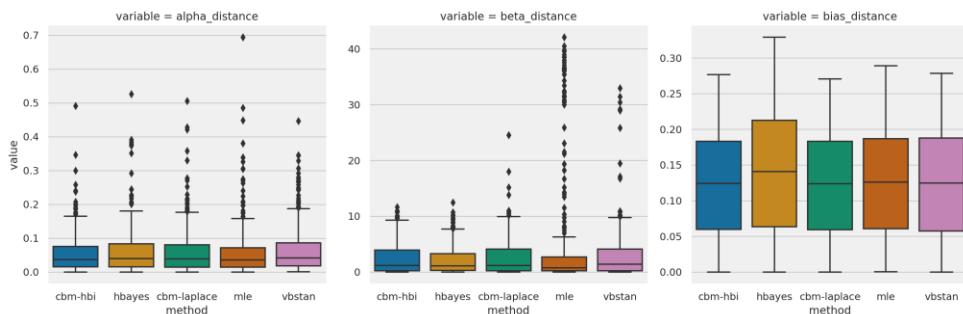


Figure 3.30 Parameter value distances for each method and agent/model. Left: QL2 α parameter. Middle: QL2 β parameter. Right: RandomBias bias parameter. Note each plot has different scales on y-axis.

In Figure 3.30 we can see that the different methods are comparable for the $QL2 \alpha$ parameter and the *RandomBias* parameter. But for the $QL2 \beta$ parameter we have the two hierarchical methods CBM-HBI and hBayesDM having lower value outliers than the other three methods that are fitted individually to subjects.

3.10.4.2 Dataset B

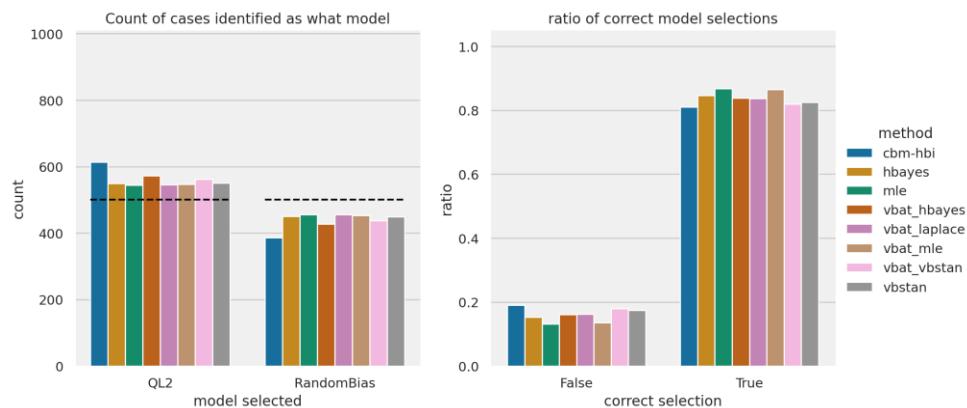


Figure 3.31 Plots showing model selection results for dataset B across different model selection methods. Note that y-axis scale differs between the two plots. Left: Number of cases identified as $QL2$ or *RandomBias*. The dotted black line indicates the true number of cases for each model. Right: Ratio of correct model selections for each method.

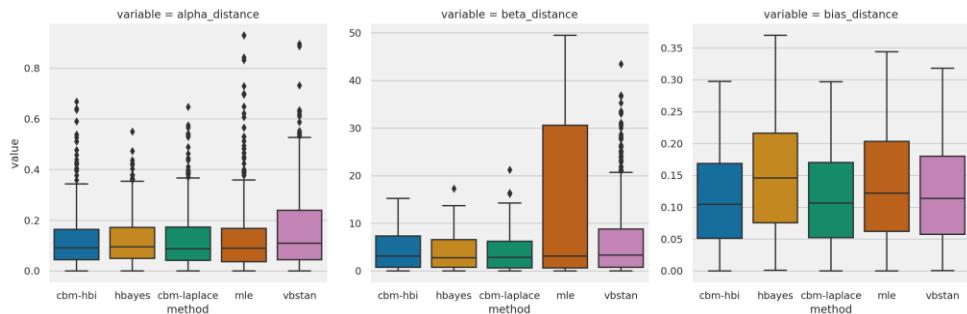


Figure 3.32 Parameter value distances between simulated parameter value and fitted parameter value for each method considered. Note that the scale on y-axis differs between the plots. Left: $QL2 \alpha$ parameter. Middle: $QL2 \beta$ parameter. Right: *RandomBias* bias parameter.

At 100 trials we see that all methods are closer to the correct number of cases for each category in the left plot of Figure 3.31, compared to the 1000 trials in dataset A. This may seem counter intuitive as we have fewer data points here in dataset B. But it is likely due to that here there are not enough trials to have long sequences of a greedy choice of one arm, making it less easy for the models to

confuse a *RandomBias* case for a *QL2* case. Interestingly, here we see in Figure 3.32 that for the *QL2* β parameter (middle plot), MLE performs badly compared to the other methods. And yet, MLE and VBAT MLE are again the best performers when it comes to model selection. There seems to be no direct correlation between parameter value distance and model selection performance, as claimed by e.g. [204].

3.10.4.3 Dataset C

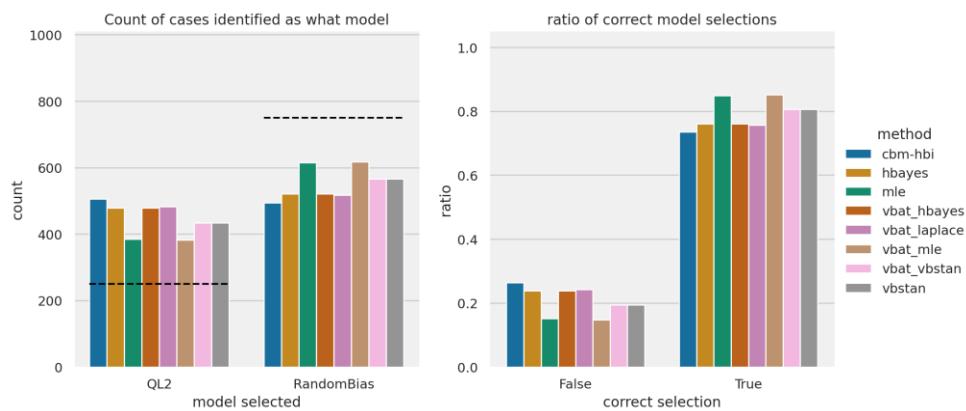


Figure 3.33 Model selection plots comparing different model selection methods for dataset C. Note the different scales on y-axis in the two plots. Left: Number of cases selected as either *QL2* or *RandomBias*. Black dotted lines indicate true number of cases. Right: Ratio of correct model selections for each method type.

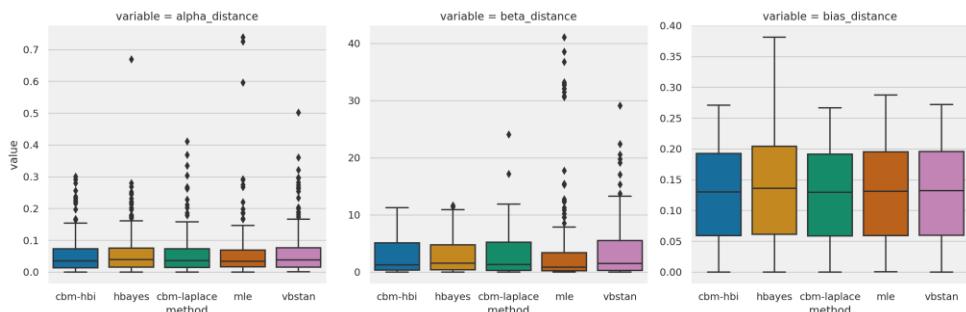


Figure 3.34 Parameter value distances between simulated and fitted values for *QL2* (left, middle) and *RandomBias* (right). Left: *QL2* α parameter distance for each method. Middle: *QL2* β parameter distance. Right: *RandomBias* bias parameter distance.

With only 25% cases of *QL2* agent simulations, we can see in Figure 3.33 that all methods overestimate the number of *QL2* cases (left). This is likely to be due to the same reasons – greedy behaviour – as discussed above. We also see in the same figure that MLE and MLE

VBAT are both closer to the correct number of cases (left) and the only methods reaching more than 80% correctly identified model cases.

In the parameter distance plots in Figure 3.34 we see similar patterns as in previous datasets. The methods are reasonably similar for $QL2 \alpha$ and *RandomBias* bias parameters, and for $QL2 \beta$ the two hierarchical methods hBayesDM and CBM-HBI have much lower incidence of outliers than the individually fitted methods.

3.10.4.4 Dataset D

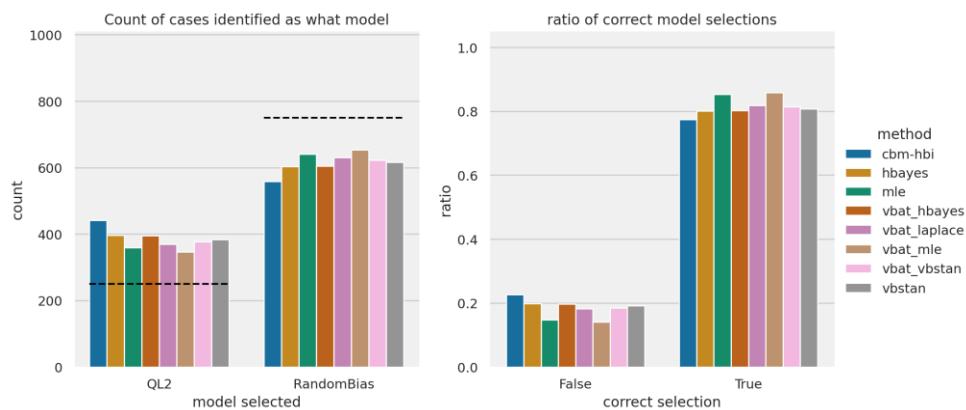


Figure 3.35 Model selection performance for the different methods. Note the difference in scale on y-axis. Left: Number of cases classified as either $QL2$ or *RandomBias* model. Black dotted lines indicate the true number of cases. Right: Ratio of correctly identified model cases.

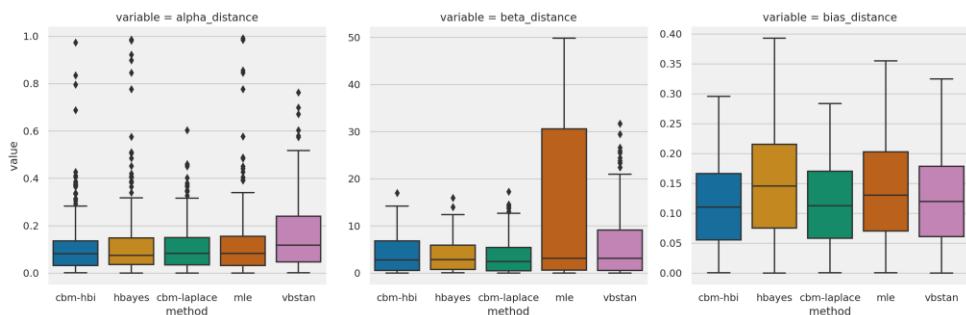


Figure 3.36 Parameter value distances between simulated parameter value and fitted value. Note the plots have different scales on y-axis. Left: $QL2 \alpha$ parameter distances. Middle: $QL2 \beta$ parameter distances. Right: *RandomBias* bias parameter distances.

Model selection performance (Figure 3.35) is slightly improved across all the methods for the 100 trials here in dataset D as compared with the 1000 trials of dataset C. Again, the MLE based methods have noticeably better performance.

For the parameter value distances in Figure 3.36, we also see a familiar pattern, namely that estimates for $QL2 \alpha$ and β (left, middle) are quite bad in general. Remember, since α only ranges between 0,1 and β between 0,20, these distances that reach close to 1 and 20 respectively are more or less uninformative. The story is better for the bias parameter (right). Despite this issue, we still have decent model selection performance as just mentioned.

3.10.5 METHODOLOGICAL CONSIDERATIONS AND CONCLUSIONS

These results are somewhat surprising. Theoretically, the HBI method of CBM should perform much better according to [204], where a point is made that better parameter estimations should provide better model selections. This does not seem to be the case. There is also in recent years a growing literature proposing Bayesian methods and Bayesian model selection as the superior option [18, 146, 204, 213, 226, 282]. Our results here show that is indeed true if the goal is decent parameter value estimates. One should however keep in mind that even then, that is only true when we have enough trials. With few trials, around 100 or less, even fully hierarchical Bayesian models provide completely unreliable results for the $QL2$ learning rate, α . Furthermore, if the goal is to estimate parameter values, it may be worth the time investment to use MCMC, at least as a final way to get results. During model development and testing, sticking to VI should provide a much quicker and more efficient feedback loop. Unfortunately, in practice it is not always so easy. VI can fail in strange ways where one needs considerable experience to know if it fails because the model is mis-specified or because there is not enough data. Or that the model does not fit that data well. To be fair, VI is as mentioned above not as mature as MCMC. In Stan (RStan, hBayesDM), there's an explicit warning on start-up of the fitting process that the algorithm is experimental.

When it comes to model selection, our results show that MLE alone is almost always numerically the best performer, even without adding VBAT analysis on top. But it could be the case this is only thanks to our relatively simple models and task. Perhaps CBM-HBI will show its strengths when we have more models in the running and/or if those models are more complex. Combined with more complex tasks, it could be that is when MLE falls apart and becomes more unreliable in comparison. A nice finding here, when it comes to

model selection, is that toolboxes such as VBAT and CBM provides individual uncertainty measures that are very useful when investigating potentially heterogenous data.

As mentioned above, many papers and frameworks (like hBayesDM, CBM) argue for the use of Bayesian inference over other methods, but it seems from our investigations that, in order to use them as done in this chapter, there are always details left out or that have not been considered. Frameworks like hBayesDM, for example, only has one available model for the two-armed bandit task, and it is hard bounded at 5 for its temperature (β) parameter. It's difficult to see any reason for choosing this setting other than to (artificially) reduce the number of errors one gets when fitting datasets with it. Furthermore, the authors do not include a random bias guessing model to sanity test their own models against generated data. Although they do seem to have different models for other tasks, it seems they do not have "control" models like for example the RandomBias model in the case of the two-armed Bandit task. As shown above, we thus added our RandomBias model to hBayesDM by customising the existing "delta" (QL2) model.

Another consideration, coming back to computational time, is that for larger datasets, for example with more complex tasks than the two-armed bandit, Bayesian inference methods would become intractable. For example, if we have a maze task with a 10x10 grid, a Q-learning agent may require (tens of) thousands of steps to learn to find a reward. If MCMC takes 2h for 50 subjects and 1000 trials, one can imagine the time it would take for just a few agents and tens of thousands of steps. We would have to use VI if we absolutely wanted to use Bayesian inference. Meanwhile, MLE fitting is extremely fast. The 20000 (10000 for each agent) simfits used to produce Figure 3.28 took 170 seconds on a modern laptop.

For the purposes of this thesis, we can conclude that MLE in conjunction with BIC and VBAT will be our preferred method for its speed and ease of use. Additionally, we shall keep using CBM when possible, as it is simple to add models to it and, as mentioned previously, it may prove better for more complex scenarios. It will also be good to have a sanity check around so as to not blindly trust the results from a single method.

3.11 APPLYING METHODOLOGY TO HUMAN SUBJECTS

We have access to a small unpublished dataset where human participants perform the Bandit task, and we shall here apply what we have learned above. The dataset is from Stolz, Pickering, & Meuller (submitted), and consists of 23 subjects. The task the subjects performed was a so-called Reversal Bandit (which will be further discussed in the next chapter), where the arm reward probabilities switch at certain points during the experiment run. Here we will thus only use the trials up until the trial before the first switch point.

Let us call this dataset the Reversal Bandit dataset or “RB dataset” for short. In this chapter, since we are only investigating a regular Bandit task and thus only use the first 80 trials, we call this subset of the data “Bandit dataset”.

Before analysing this data, it’s important that we get an understanding of how our model recovery performs for this particular task configuration. As we saw above, arm reward probability differences as well as the number of trials impact these measures. Furthermore, so far, we have mainly been using *QL2* when looking at parameter recovery so we should also investigate *RandomBias* parameter recovery with our two methods we will be using: MLE and CBM.

3.11.1 MODEL RECOVERY CHECK

We will simulate 1000 subjects performing 80 trials on this task, half with *QL2* and half with *RandomBias*. In the Bandit dataset, *arm1* is the “best” arm with arm reward probability of 0.7, and *arm2* has reward probability of 0.3. So, these 1000 subjects play a Bandit task with the same reward contingencies. Parameter values for the agents are randomly drawn individually for each agent as follows:

$$\alpha_{QL2} \sim U(0, 1), \quad \beta_{QL2} \sim U(0, 20), \quad bias_{RandomBias} \sim U(0, 1)$$

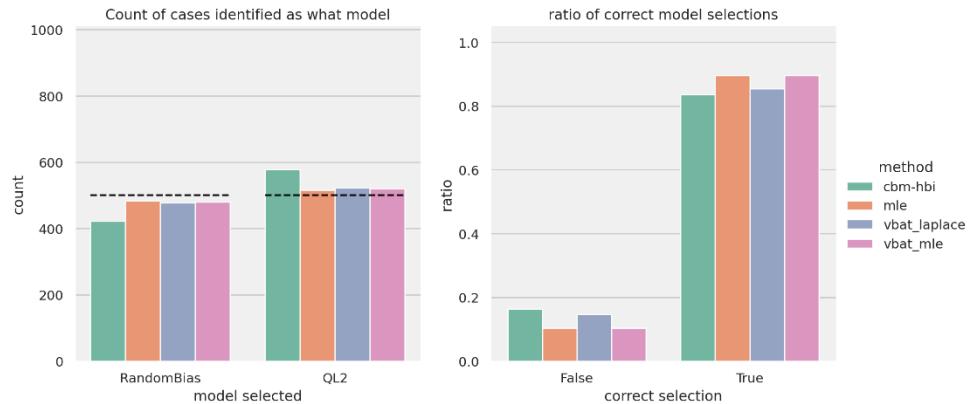


Figure 3.37 Model selection plots for 1000 simulated subjects where half were QL2 agents and half were RandomBias. Left: Count of cases selected as being RandomBias or QL2. Black dotted lines indicate true number of cases. Right: Proportion of correctly identified models.

The pattern seen in Figure 3.37 is similar to the earlier method comparisons where MLE and VBAT MLE show overall numerically better performance. Here we can also see the importance of checking the ratio of correct model selections (right), since in the left plot it looks like VBAT Laplace method is comparably good, being close to the black dotted lines, but in the right plot we see it mislabels subjects to a larger degree.

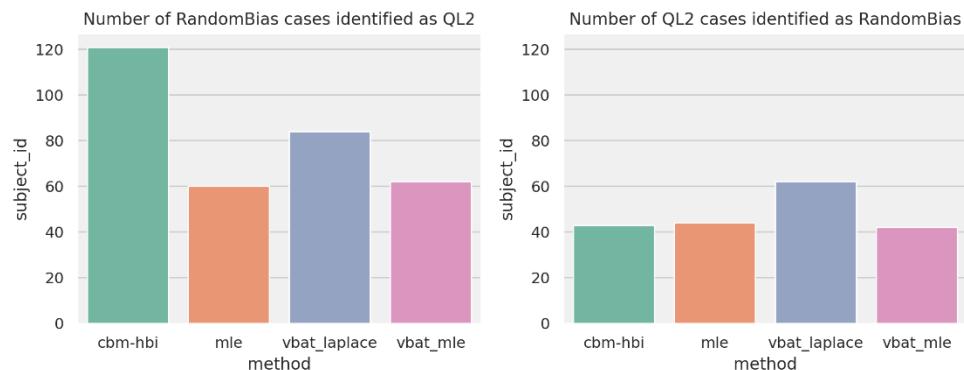


Figure 3.38 Barplots showing details on how many of the misidentified cases were RandomBias misidentified as QL2 (left), or QL2 identified as RandomBias (right)

We can further see in Figure 3.38 that CBM-HBI method tends to misidentify *RandomBias* cases as *QL2* which again is the same pattern as we have seen previously. Nonetheless, it is important that we have confirmed that this is the case also in this task configuration.

3.11.2 PARAMETER RECOVERY CHECK

Let us also look at the parameter value recoveries for our methods in this task configuration. Note that for each parameter mentioned

below we are looking at the subset of either the QL2 or RandomBias simulations. This is because we only have simulated parameter values for those subsets to compare with the fitted values.

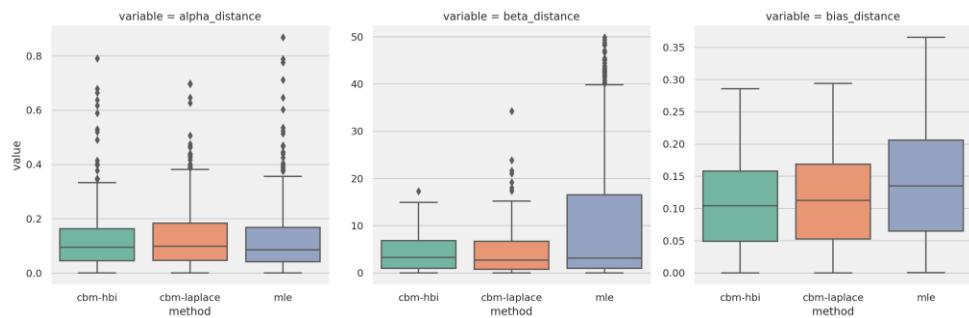


Figure 3.39 Boxplots showing average distance between simulated parameter and fitted parameter across the methods. Left: QL2 α parameter distance. Middle: QL2 β parameter distance. Right: RandomBias bias parameter distance.

Not surprisingly, CBM-HBI is the better performer overall as seen in Figure 3.39. With so few trials, QL2 α recovery is not good with either method (left), but thanks to shrinkage in the hierarchical CBM-HBI we get decent fits for QL2 β (middle). MLE performs the worst for the *RandomBias* parameter (right).

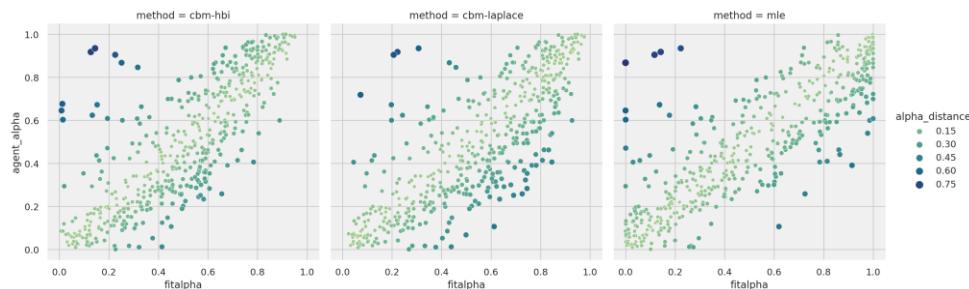


Figure 3.40 Plots for QL2 α parameter for each method. The simulated parameter value on y-axis and the fitted parameter value on x-axis. Size and colour indicate the distance between simulated and fitted value. Left: CBM-HBI method. Middle: CBM-Laplace method. Right: MLE method

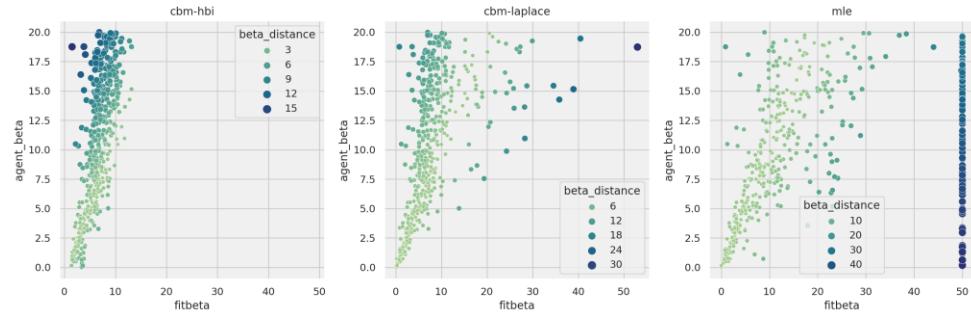


Figure 3.41 Plots for QL2 β parameter for each method. The simulated parameter value is on y-axis and the fitted value is on the x-axis. Size and colour indicate the distance between simulated and fitted value. Note that each plot varies in their distance scales. Left: CBM-HBI method. Middle: CBM-Laplace method. Right: MLE method.

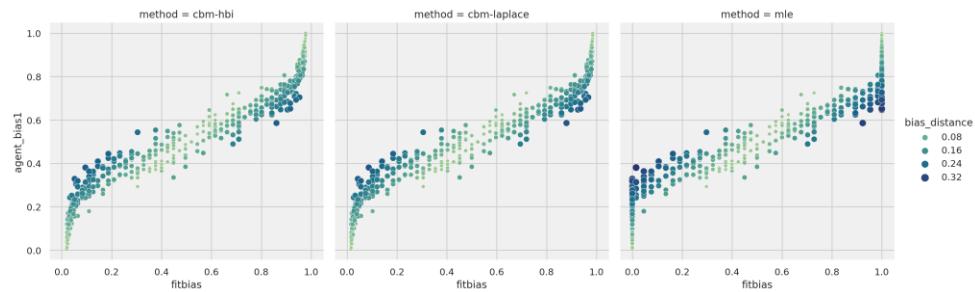


Figure 3.42 Plots for RandomBias bias parameter for each method. The simulated parameter value is on y-axis and the fitted value is on the x-axis. Size and colour indicate the distance between simulated and fitted value. Left: CBM-HBI method. Middle: CBM-Laplace method. Right: MLE method

In Figure 3.40, Figure 3.41, Figure 3.42 we have plotted the simulated and fitted parameter values in scatterplots, with colour and size of the markers to indicate distances. Overall, there is a similar pattern as we have seen in previous investigations with such plots, for example that with so few trials, the MLE method is highly unreliable for QL2 β and can hit the boundary wall for any simulation β value (Figure 3.41, right). Also noteworthy is in the same figure (left) how CBM-HBI method underfits the higher QL2 β values due to shrinkage.

Interestingly, in Figure 3.42, we see that all three methods tend to fit extreme values for the *bias* parameter when the simulated value is below 0.2 or above 0.8. The CBM methods are smoother, whereas MLE tends to look more discrete.

3.11.3 MODEL SELECTIONS FOR BANDIT DATASET

With the above sanity checks we can now analyse the actual human Bandit dataset.

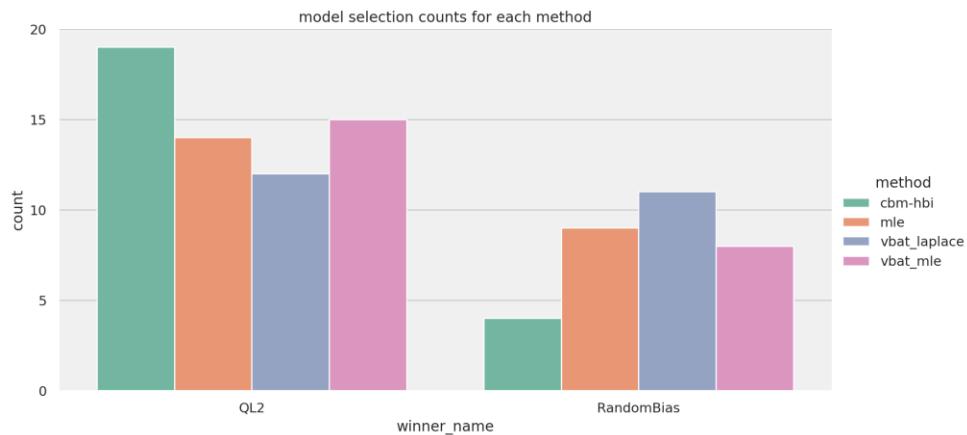


Figure 3.43 Model selection counts for the Bandit dataset. Number of cases on y-axis and on x-axis we have what model was selected, contrasted by method.

As seen in Figure 3.43, there appears to be some subjects better explained by random behaviour than reinforcement learning. Or rather, some subjects are better explained by *RandomBias* than *QL2*. We know from our checks above that CBM-HBI generally overfits *QL2* cases and that VBAT-MLE is correct to a larger degree. We can understand better how this happens by looking at the raw data (Figure 3.44) where we see that there looks to be a difference between the subjects who are doing this task for the first time (group 1, 11 subjects) and those who have already done a similar task before (group 2, 12 subjects; previously they had done a punishment only version of the task). Group 1 has $M=0.28$ ($SD=0.45$) for what arm was chosen (where 0 is arm1 and 1 is arm2), while group2 has $M=0.19$ ($SD=0.39$). We can confirm that there is in fact a difference between the groups with an independent T-test; $t(22)=4.59$, $p=4.7E-6$. The fact that the task they have done before uses punishment/no punishment instead of reward/no reward is probably irrelevant, as the structure of the task is otherwise identical to all intents and purposes.

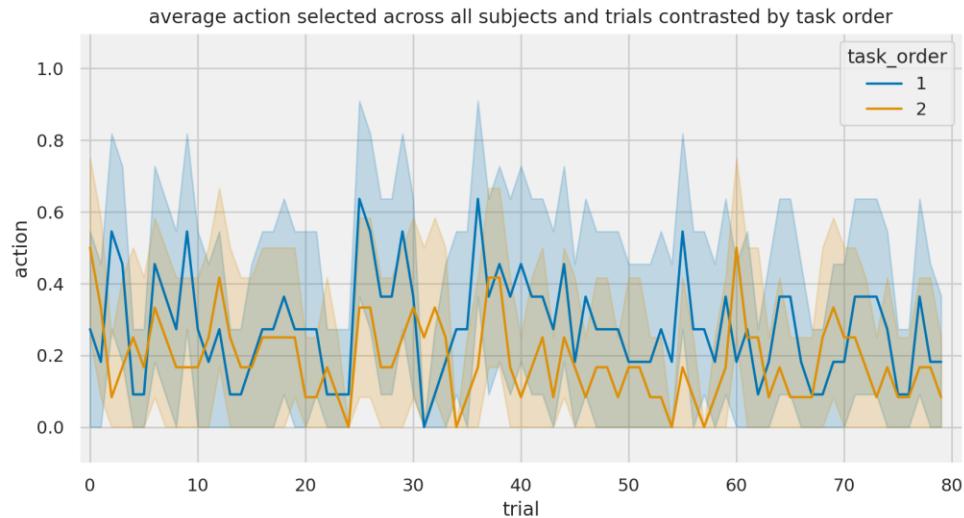


Figure 3.44 Plot showing what action was selected on the y-axis - arm1 (0) or arm2(1) – across all 80 trials shown on the x-axis. Differently coloured lines indicate if participants did this task first (task order 1) or if they had done a similar task earlier (task order 2)

Since greedy behaviour can be explained well by a *RandomBias* model with a strong bias for one arm, it could be the case here that the subjects of group2 – since they show greedier behaviour – more commonly are classified as *RandomBias*. In Figure 3.45, where we have only used VBAT-MLE, we can confirm that is indeed the case.

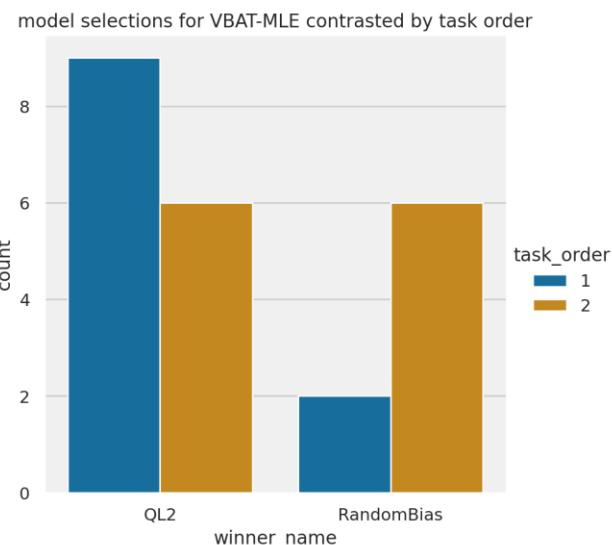


Figure 3.45 VBAT-MLE model selections contrasted by task order. Subjects with task order 1 did the task for the first time while subjects with task order 2 did the task following prior experience with a very similar task.

As mentioned above, the hierarchical methods of CBM and VBAT also provide subject-level model probabilities. We can thus also look at the probabilities for each subject being explained by one model

over the other across the methods as seen in Figure 3.46. There we can see how there are a few subjects where the methods agree on their behaviour being better explained by *RandomBias*.

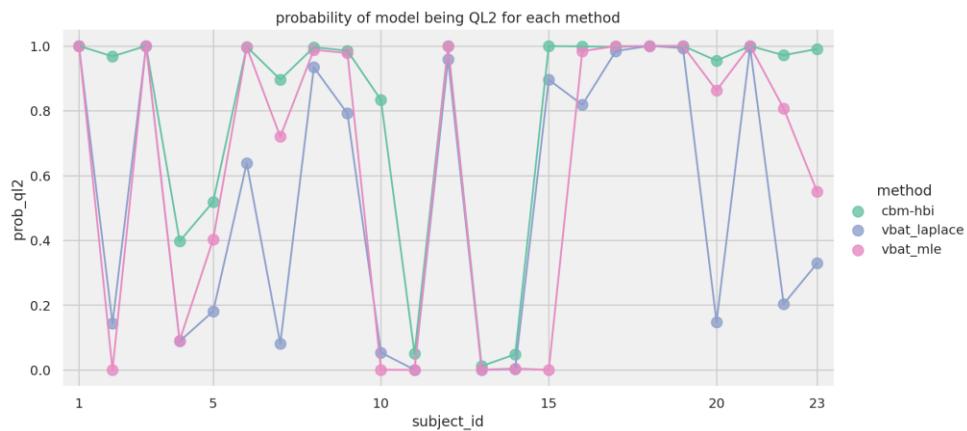


Figure 3.46 Probability of model being QL2 across all subjects. Because we are only comparing two models, this also shows probability of being RandomBias (towards 0 on y-axis).

3.11.4 PARAMETER VALUE PLOTS FOR BANDIT DATASET

As an additional check we can also look at how the methods agree on parameter value estimates. This is not going to be our focus going forward, as we are more interested in model selections overall, but it is good practice to do this check in order to find potential discrepancies¹⁷.

¹⁷ Anecdote: thanks to doing this check we found an error in our code that did not impact model selections but did effect parameter value estimates for the MLE method.

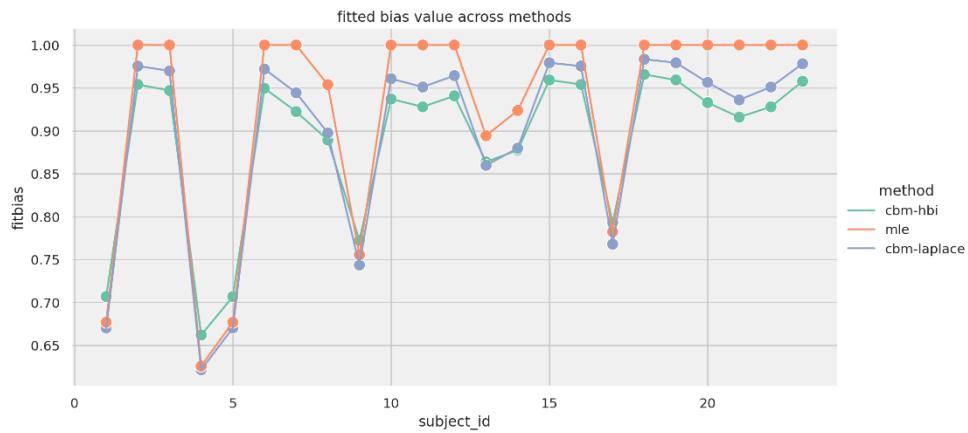


Figure 3.47 Plot showing fitted RandomBias parameter value for bias on y-axis, across all subjects (x-axis). Colours indicate method used.

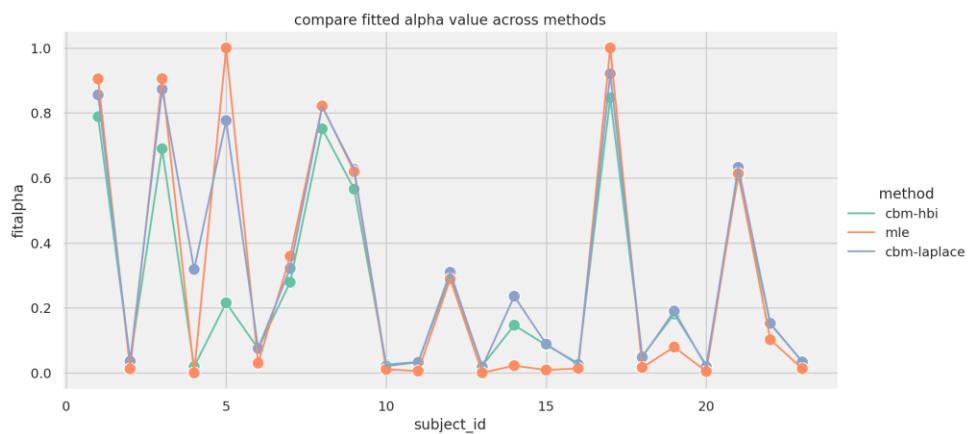


Figure 3.48 Plot showing fitted QL2 α parameter value on y-axis for each subject on x-axis. Colours indicate what method was used.

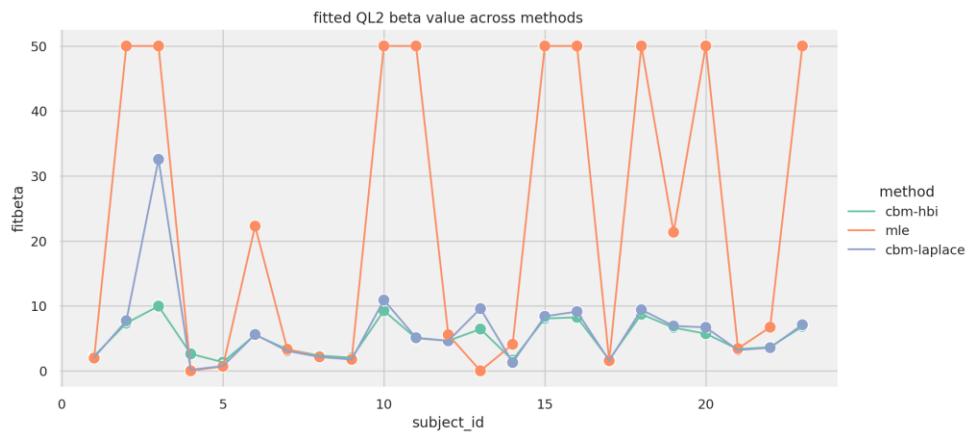


Figure 3.49 Plot showing fitted QL2 β parameter value on y-axis for each subject on x-axis. Colours indicate what method was used.

Overall, we can see in Figure 3.47, Figure 3.48, Figure 3.49 that the methods largely agree but MLE hits its bounds more often than the others. This is especially noticeable in Figure 3.49 showing β parameter for QL_2 model. This could indicate that we will have issues with model selection, but as we have seen in our exhaustive testing of these methods, this does not impair MLE to correctly identify models, especially when combined with VBAT.

3.12 CHAPTER SUMMARY

In this chapter, we have investigated methods for parameter estimation and model selection. We did so using a simple decision-making task in the form of a two-armed bandit task. We simulated behaviour in this task using the basic RL algorithm Q-learning and a control algorithm that picks choices at random with a certain bias towards one option. Behavioural data from the simulations was then fitted using these two models to recover parameter values and select what model most likely generated the data.

What we show, in short, is that recovering parameter values is difficult. The main reason for it being difficult, is because to get better estimates we need large numbers of data points. But when our goal is to fit data from human subjects, we often do not have enough data points since it is not easy to get humans to perform in such long experiments that would ideally be needed.

We also show that this problem remains even when using modern approaches in the form of Bayesian inference, methods that are commonly considered much more performant. Bayesian methods are indeed better at estimating parameter values – and provide value uncertainty measures of these estimates – but they are much slower than the classic MLE method and the uncertainty measures do not necessarily provide valuable information.

The most interesting finding here, however, is when it comes to model selection. One would expect that better parameter estimates leads to improved model selection accuracy. But we show that MLE fitting – especially when combined with a tool like VBAT – often has superior accuracy when selecting the model most likely to have generated the data.

4 STEPPING INTO STATES WITH REVERSAL LEARNING

In the previous chapter, we investigated how learning may happen in the simple case of a two-armed bandit task. We will now extend that task by allowing the arm reward probabilities to change during the experiment.

We will first investigate how our agents from the previous chapter handle this extension of the task and show how they do not always explain animal behaviour well. We then discuss how the agent algorithms themselves can be extended by adding the concept of states, along with alternate agent algorithms from the literature. As in the previous chapter, we then contrast and compare these behavioural models considering parameter recovery and model selection. Using the knowledge thus gained, we apply the models on two sets of data with human participants and discuss the outcome.

As mentioned in the previous chapter, we can allow the arm reward probabilities in the bandit task to change throughout the experiment, either once or several times. Most common, and simplest, is the case when we have a two-arm bandit, and the good and bad arms are switched at some point. Hence, the name “reversal” bandit, or reversal learning. This type of task has been used in different forms for decades, for example to investigate learning in monkeys [169, 293], in bumblebees [43, 208], and humans [109, 165].

Moving beyond simple reversals, the task can be made more complex for example by increasing the number of options [295] or using a range of reward values for each option so that the ranges for each overlap slightly and thus several trials are needed to find the best [73, 296]. The more variation in the reward schedule, the more difficult it will be to discern that a reversal switch has occurred.

Another way to complicate the task is to have not a switch per se, but to make rewards depend on how many times a specific option has been picked. For example, in [296], there are two decks of cards, each with 80 cards. Participants are told to maximize their points over a total of 80 card draws, where each card gave a point between 1 and 10. One deck – the advantageous one – had an average of 3 points per card over the first 20 cards drawn, an average of 7 points over the next 50 cards drawn, and an average of 3 points over the last 10 cards

drawn. The other deck – the disadvantageous one – gave an average of 8 points per card for the first 30 cards drawn, an average of 5 points for the next 20 cards drawn and an average of 2 points per card for the final 30 cards drawn.

Such a task, however, is not strictly about reward learning as it is also or perhaps more, probing the exploration/exploitation question. But as mentioned in the background chapter, it is very difficult to separate these questions as they are so intimately tied.

Another example of a reversal task, is one where we also have two options and which one is the best switches multiple times during the experiment as in [224]. We have options A and B, where A starts out having a 70% probability of reward and B has 40%. After the participant has selected the correct option A consecutively four times, there's a 25% probability on each following trial that the reward probabilities switch. After such a switch, the participant again must pick the correct option – which is now B – four consecutive times to allow another switch to occur.

One may think that such a “simple” extension of the bandit task – arm reward reversals – doesn't impact the complexity of the task very much. But as we shall see, it is sufficient to require additions to our existing algorithms, perhaps even quite different algorithms.

4.1 SIMULATING PERFORMANCE IN THE REVERSAL BANDIT TASK

We start out by investigating a straight-forward reversal task, where we have 280 trials in total and what arm is best switches three times throughout the experiment. The switch points are on trials 83, 151 and 226. The arm reward probabilities are 70% for the good arm and 30% for the bad arm, with arm1 starting out as good. This task version has been chosen as it is the same as the full version of the human dataset introduced at the end of the previous chapter. There, we only used the first 80 trials of this dataset but below we use the full dataset.

It should be mentioned here that in the human data, there are two groups which differ slightly in the switch points. The other group used 87, 161, 224. This was done to counterbalance the unlikely yet possible case of subjects doing the experiment their second time would be counting trials. For our purposes and convenience, this detail is not replicated in our simulations.

Initially, we only use the QL_2 agent from the previous chapter to explore this task. We investigate the performance of QL_2 by first exploring the parameter space. 1000 agents are simulated, each with randomly drawn parameter values as per:

$$\alpha \sim U(0, 1), \quad \beta \sim U(0, 20)$$

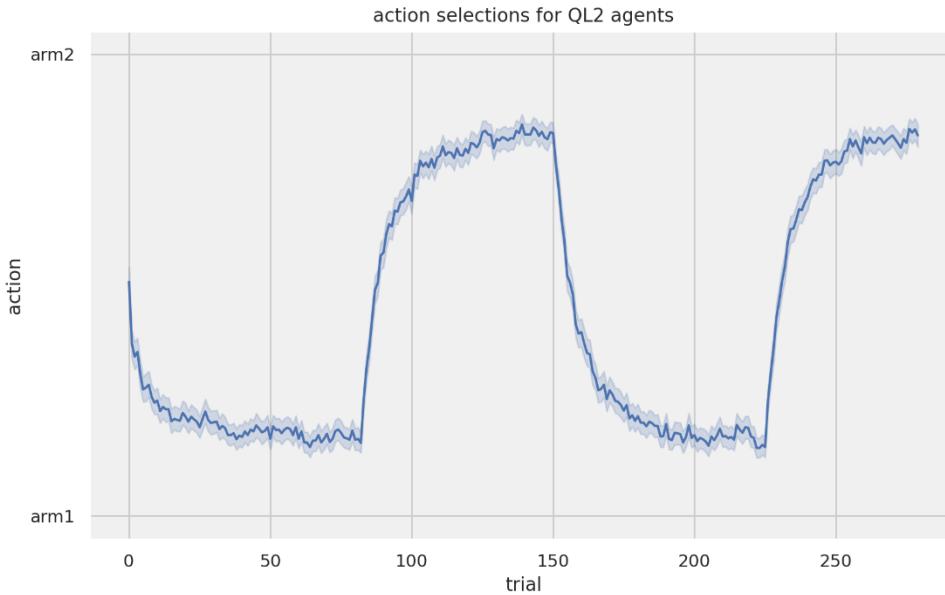


Figure 4.1 Performance averaged across QL_2 agents using parameter values from the entire parameter space. On y-axis we have the selected action and x-axis shows each individual trial. The shaded area indicates 95% confidence interval.

As seen in Figure 4.1, the QL_2 agent can manage the Reversal Bandit. The results are, as mentioned above, the average across 1000 agents with individually randomly generated parameter values so what we learn here is that most parameter value combinations can handle the task. What is noticeable is that the QL_2 agent requires many trials after a switch (most easily seen after trial 150) to learn the new best action and performance is still improving almost up until the next switch point.

We also conduct a “parameter sweep” where we investigate performance for all permutations of parameter value ranges for the QL_2 agent. Performance is measured as proportion of actions picking the correct best arm and the parameter value ranges are:

$$\alpha \in (0.01, 1), \text{ stepsize} = 0.02; \quad \beta \in (1, 2, 5, 10, 20)$$

In total this gives us 50 α values, and thus 250 parameter value combinations. Each parameter value combination is simulated 100

times to account for random variation in the performance of single simulation runs. The results are presented in Figure 4.2.

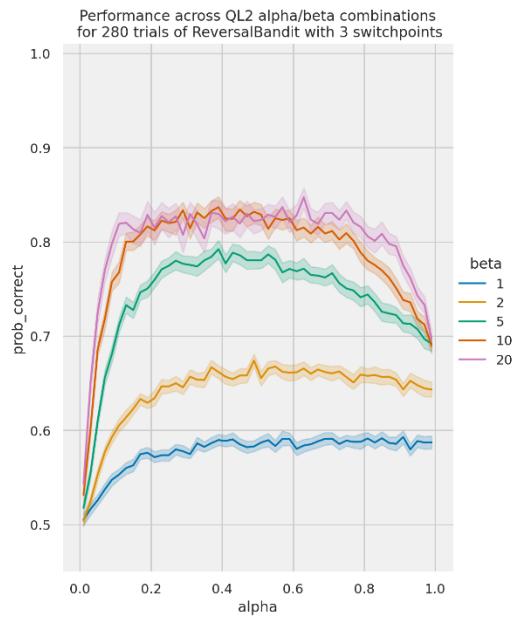


Figure 4.2 Performance for parameter value combinations of QL2 α (x-axis) and β (coloured lines). Probability of choosing the correct arm is shown on y-axis. Shaded areas indicate 95% confidence interval.

As seen in Figure 4.2, higher β is generally correlated with higher performance. One may think that higher α would be good to quickly learn when a switch occurs, but that would also mean “accidental” rewards from the bad arm would influence behaviour too much. Interestingly, there is no discernible difference between $\beta = 10$ and $\beta = 20$ except for higher α values. We can see why, if we look at individual action choices for each trial. Based on the results in Figure 4.2, we select $\alpha = 0.4$ and then run 1000 simulations for each of the β values seen in the plot and mentioned above. There is no other significance to doing 1000 simulations here instead of 100 as before, other than the plots being easier to read. These simulations are shown in Figure 4.3. There we can see how the greediest agent ($\beta=20$) reaches higher average on correct arm selection, but the agent with $\beta=10$ is slightly faster at switching, due to not being “stuck” selecting the action with higher value. This pattern repeats for the lower β values, where the lower it is, the faster the agent can switch. But higher β is needed to reach higher performance between each switch point. We can also see here that even for our best performing parameter

combinations, more than 50 trials are needed after a switch to get almost exclusive selection of the best arm.

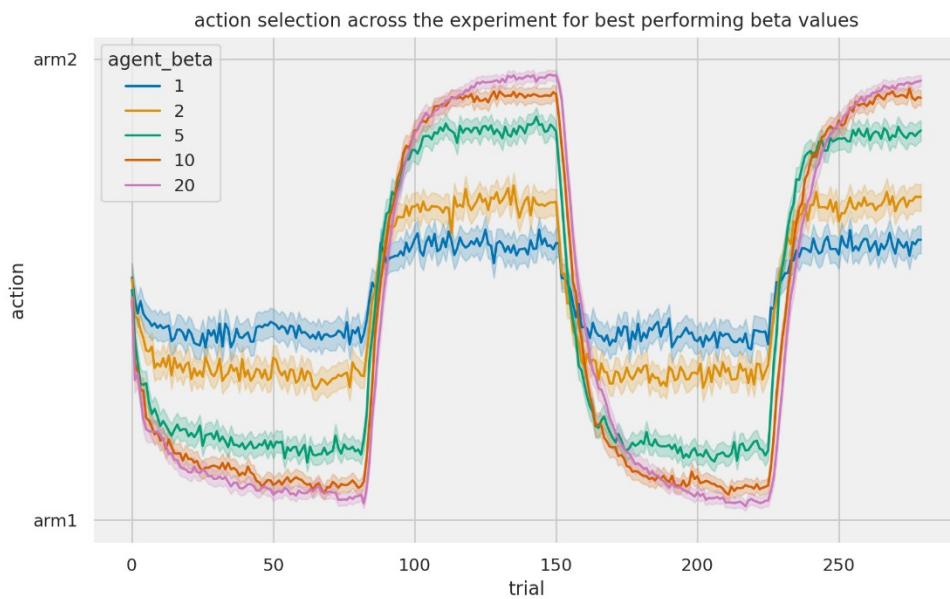


Figure 4.3 Action selections for QL2 agents using $\alpha=0.4$ and β as per the plot legend. The selected action is shown on the y-axis for each individual trial (x-axis). Each line is the average for 1000 simulations.

We can further illustrate this in the form of Figure 4.4, where we have plotted the average proportion of best arm choices after a certain number of trials following each switch. The overall pattern is that performance still has an upwards trend at 50 trials after a switch point.

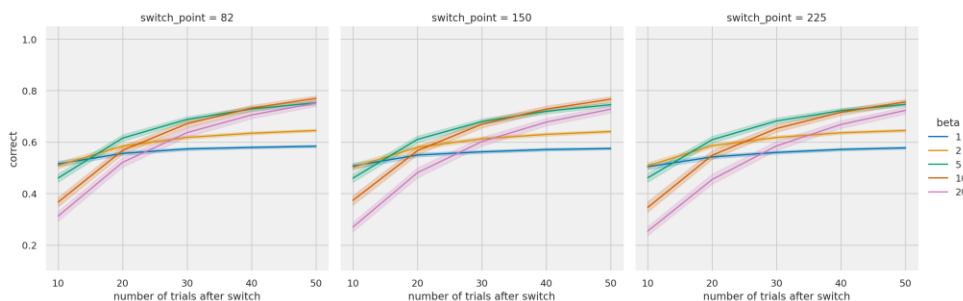


Figure 4.4 Proportion of correct choices (y-axis) for QL2 agents using $\alpha=0.4$ and β as per the plot legend. The x-axis shows number of trials after each of the three switch points. Left: The first switch point at trial 82. Middle: The second switch point at trial 150. Right: Third switch point at trial 225.

4.2 CONTRASTING QL2 BEHAVIOUR WITH HUMAN BEHAVIOUR

We now turn our attention to the dataset with human data from the same task. As mentioned in the previous chapter (and see there for more details), this dataset consists of 23 subjects. We are again only looking at the case with reward/no reward, and around half of the subjects had, when performing this task, already done the same task with punishment/no punishment. In Figure 4.5, we can see how this reflects in the performance of the subjects who did the task for the second time. They are much quicker at switching, especially at switch point two (middle). Strangely both groups perform worse after the third switch which may be a sign of subjects getting tired and/or bored. Note that the two groups' switch points were slightly different, but the plot shows number of trials after switch.

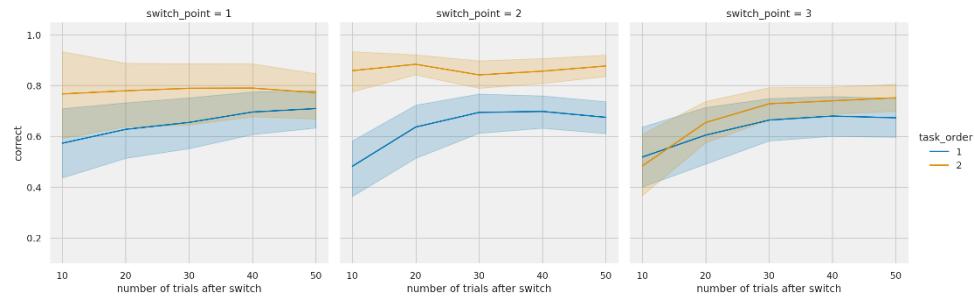


Figure 4.5 Average proportion of correct arm choices (y-axis) for human subjects performing the reversal bandit task. Colours indicate if subjects did the task for the first time (task order 1) or if it was their second time (task order 2). X-axis indicates number of trials after a switch in arm reward contingencies. Shaded areas indicate 95% confidence interval. Left: First switch point. Middle: Second switch point. Right: Third switch point.

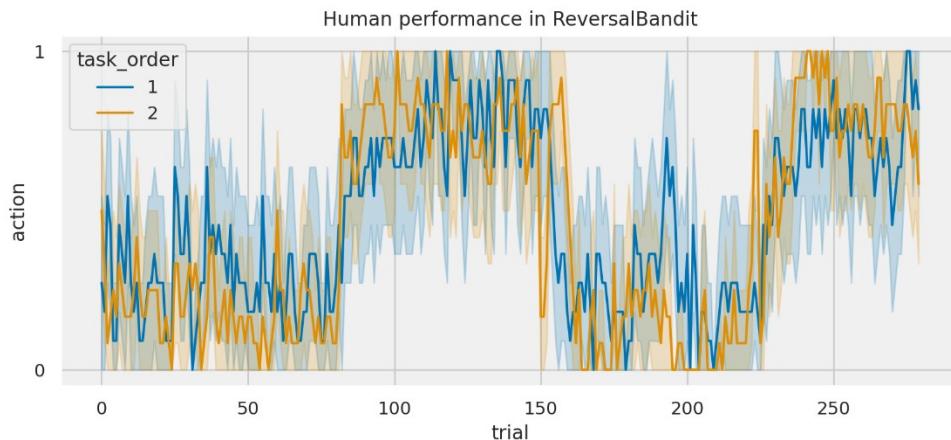


Figure 4.6 Average arm (action) selection across subjects (y-axis) for each individual trial (x-axis). Coloured lines indicate subject group as shown in legend and described in the text. Shaded areas indicate 95% confidence interval.

We can also look at the action choices across all the trials, as plotted in Figure 4.6. Because of the low number of participants, the confidence intervals are quite big, but the overall patterns are still distinguishable. Although it's difficult to say for sure, we can for example see that it looks like many subjects in group one believed there was a switch around trial 180-190 (blue line, Figure 4.6). It may also be the case that many subjects in group two remembered there was a switch after approximately 150 trials the first time they played, hence the dip and rebound on and following trial 150 (gold line, Figure 4.6). This could also indicate that all subjects in a group saw the same sequence and thus the noticeable average changes in behaviour.

More important than such details is that the human switch behaviour is quite different from that of QL2. It's clear by contrasting Figure 4.3 and Figure 4.6 that humans seem to switch faster. How come?

As discussed in the background chapter, humans and other animals can internally reason about states of the world. Subjects in the reversal task dataset were told there would be reversals of the arm rewards. Thus, they were primed for such state switches which could account for the faster switch behaviour compared to QL2.

4.3 ALTERNATIVE ALGORITHMS WITHOUT STATES

Before we discuss states further, we should consider if there are alternative RL algorithms that may work better than our existing QL2.

There are myriad ways to modify Q-learning, and even more algorithms if we consider other types of RL, as mentioned in the background chapter. Here we have chosen two that are commonly used in the field of decision making in the cognitive sciences.

4.3.1 DUAL A QL

One way to increase the complexity of QL is to distinguish between positive and negative reward prediction errors. We can do this by adding another learning rate parameter, so we have α_{pos} and α_{neg} . In mathematical terms this means:

$$RPE = r_t - Q(a_t)$$

$$Q_{t+1}(a_t) = Q_t(a_t) + \alpha * RPE \begin{cases} \alpha = \alpha_{pos} & \text{if } RPE \geq 0 \\ \alpha = \alpha_{neg} & \text{if } RPE < 0 \end{cases}$$

Note that it is somewhat arbitrary if we choose $RPE \geq 0, RPE < 0$ or $RPE > 0, RPE \leq 0$. The reasoning behind this separation is that it's plausible that positive and negative prediction errors differ in how impactful they are on learning [92, 204]. Important to note is that our regular QL2 model is a special case of Dual- α QL, where $\alpha_{pos} = \alpha_{neg}$. Thus, QL2 is a *nested* case of Dual- α QL.

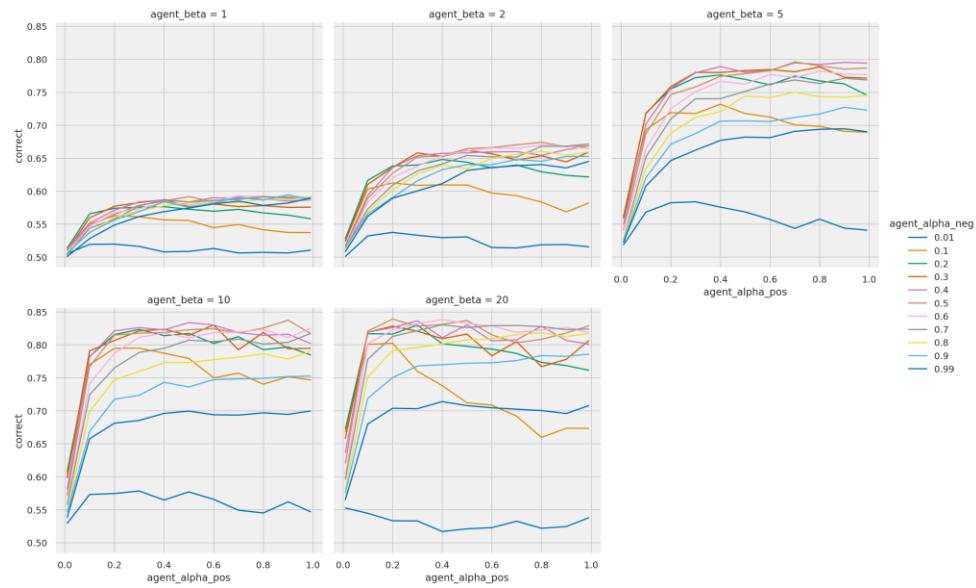


Figure 4.7 Proportion correct arm choices (y-axis) for Dual- α QL agents. Each line is the average of 100 agents with the same parameter combination. X-axis is the value for positive α , and each line is for a different value of negative α as per the legend. Upper left: $\beta=1$ Upper middle: $\beta=2$ Upper right: $\beta=5$ Lower left: $\beta=10$ Lower middle: $\beta=20$

In Figure 4.7 we can see the proportion of correct choices for a parameter sweep with Dual- α QL agent. The task is the same ReversalBandit as described above (280 trials and three switch points). The ranges used for parameter combinations were the same for both α and β with values as seen in the legend. Ranges for β were also as per the figure. These ranges resulted in 605 parameter combinations in total. Overall, we can see that larger β leads to higher performance. For positive α the optimum is somewhere between 0.2 and 0.5, together with a negative α between 0.3 and 0.7. This illustrates that the more parameters an agent/model has, the more difficult it is to get a good overview of its performance across the parameter space. Compare Figure 4.7 with Figure 4.2.

4.3.2 DUAL UPDATE QL

Another way to adapt QL is to not only update the action value for the chosen action but also update the value for the non-chosen action. This takes the structure of the task into account and may be a way of representing human participants' knowledge of that general structure, without introducing states in a more explicit way [225]. If we denote the non-chosen action as \tilde{a} , on each trial t we perform the following two updates:

$$\begin{aligned} Q_{t+1}(a_t) &= Q_t(a_t) + \alpha(r_t - Q_t(a_t)) \\ Q_{t+1}(\tilde{a}_t) &= Q_t(\tilde{a}_t) + \alpha(-r_t - Q_t(\tilde{a}_t)) \end{aligned}$$

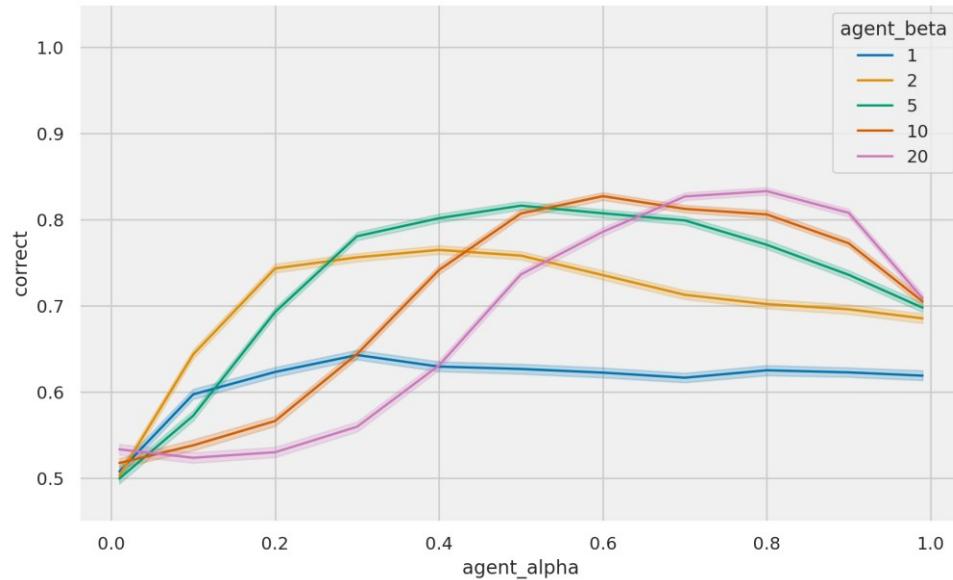


Figure 4.8 Proportion of correct choices (y-axis) for parameter combinations of Dual Update QL agent. Agent α value is on x-axis. Each line represents different values of β as per the legend. Shaded areas indicate 95% confidence interval across the 100 simulations for each parameter combination.

In Figure 4.8 we have plotted performance for the Dual Update QL agent as proportion of correct choices across the parameter space. The range for α was the same as that seen for negative α in Figure 4.7. Interestingly, compared to other QL agent types we here see that best performance is reached with high values of α , and it additionally looks like there's a slight correlation here between the higher the β , the higher is the optimal α .

4.4 ADDING STATES TO Q-LEARNING

The simplest way to add state information would be to adapt our existing QL2 algorithm, enhancing it with states. This is easy, since Q-learning in its original form already has states. The version we have been using is a simplification. However, the states meant in standard Q-learning are observable states. For example, if you are a law-abiding citizen and the crosswalk light is red, you stay. When the light turns green, you walk. Expressed differently, when the light is red there's a high risk of punishment (being hit by a car or charged for jaywalking if that crime exists where you live) if you walk. When the light is green, there's a low risk of punishment for walking, perhaps even a high chance of reward for walking because you're en route to

your favourite coffee shop. For a Q-learning agent in this situation, we would have two states – red and green.

In the case of our ReversalBandit task, there are no such observable states. As we mentioned above, this task and especially its non-reversal version the regular bandit task, are often seen as stateless. Another viewpoint is that there is only a single state of the world.

We could also view the ordinary Bandit task as one where there are two hidden states. Either arm1 is the best choice, or arm2 is. But since the state never changes, it never becomes relevant to consider. In the Reversal bandit however, these hidden states do become relevant because what arm is the good one switches.

Since we know the task contingencies, we can create a QL2 version with perfect information about these hidden states. When arm1 is best, we are in state 1 and when arm2 is best we are in state2. In our algorithm we already switch what arm is best at certain switch points, so we can set the new state for the QL2 agent as well. This is of course not very realistic for investigating our human data, but it can serve as a decent baseline complement to our existing baseline agent RandomBias. If this model proves to fit data better than others, it may indicate there is something amiss in our data. For reasons that will become clear in later chapters, we call this version State Enhanced Q-learning or SEQL2 for short.

In mathematical form¹⁸, SEQL2 updates state-action values on each timestep t like:

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha(r_t - Q_t(s_t, a_t)) \quad 4.1$$

¹⁸ For algorithmic version, see https://github.com/fohria/phd_thesis

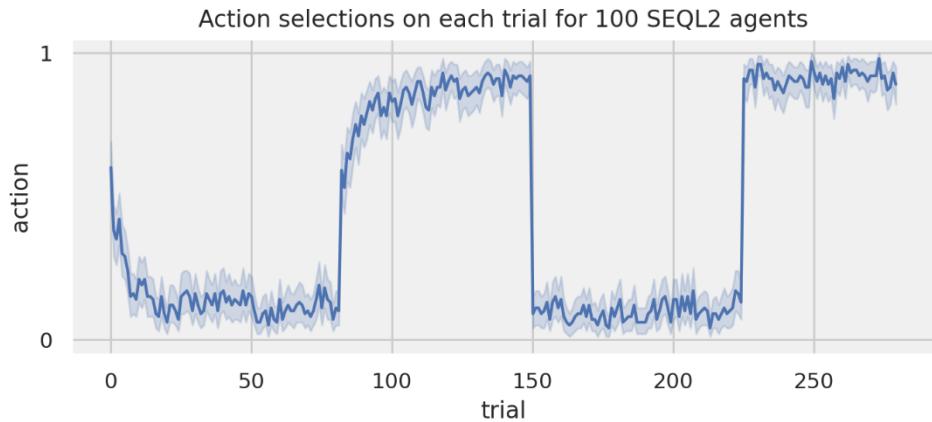


Figure 4.9 Action selections (y-axis, arm1 or arm2, 0 or 1 respectively) on each trial (x-axis) averaged across 100 SEQL2 agents, all using the parameter values $\alpha=0.3$ and $\beta=7$. Shaded areas indicate 95% confidence interval.

To demonstrate the behaviour of our SEQL2 agent, we run 100 simulations of the same ReversalBandit task described above for the human subjects. All 100 simulations used the same agent parameter values, where $\alpha = 0.3$ and $\beta = 7$. In Figure 4.9 we have summarised the behaviour as what arm was picked across all the trials. We can see that before and after the first switch, the behaviour is similar to the standard QL2 agent (Figure 4.1), but after the second and third switches we immediately go back to the previous value. This is because before the first switch, we are in state 1 and the agent must learn what arm is best, just as the regular QL2 agent does. After the switch, we are now in state 2, which has its own Q-values, and the agent must learn these Q-values from scratch. But after the next switch – the second switch – we are back to state 1 and the agent thus uses its already learnt Q-values.

4.5 ALGORITHMS THAT ARE MORE STATEFUL TO BEGIN WITH

As discussed in the background chapter, there is evidence that animals such as rats and humans can infer hidden states of the environment based on non-hidden observations. In the cognitive sciences literature this is commonly called belief states¹⁹ and the models use Bayes' rule to update a prior belief of what state the

¹⁹ In the computer science literature these are commonly referred to as Partially Hidden Markov Decision Processes, POMDPs.

animal is in, based on observations of their own actions and their consequences – rewards in our case.

The common way to model such belief states is using Hidden Markov Models (HMM) [109, 225]. These allow us to rely on the Markov property, i.e., that all the information needed to make a choice on the current timestep is included in the prior that we have with us. In other words, the prior used in the current trial is the posterior from the last trial.

The specific implementation we are using here is based on [225]. We start out with a prior belief over the two states, $prior(s_t)$, where $prior(s_t = 1) + prior(s_t = 2) = 1$. This belief state indicates if we think arm1 is currently the best choice ($s_t = 1$) or if arm2 is the best choice ($s_t = 2$). Based on this prior, we select an action using the prior as the probability of selecting each action. In other words, if we strongly believe that arm1 is the best choice ($p(s_t = 1) \gg p(s_t = 2)$) then it is more likely we pick arm1.

When the action is performed, we receive a reward value, and together they make an observation $o_t = \{a_t, r_t\}$. The posterior belief over the two states is then:

$$posterior(s_{t+1}|o_t) = \sum_{i=1}^2 p(s_{t+1}|s_t^i) \frac{p(o_t|s_t^i)prior(s_t^i)}{\sum_{j=1}^2 p(o_t|s_t^j)prior(s_t^j)} \quad 4.2$$

where:

$$p(o_t|s_t) = 0.5 + 0.5 * \begin{cases} c & \text{if } a_t = s_t \text{ and } r_t = 1 \\ -c & \text{if } a_t \neq s_t \text{ and } r_t = 1 \\ d & \text{if } a_t = s_t \text{ and } r_t = 0 \\ -d & \text{if } a_t \neq s_t \text{ and } r_t = 0 \end{cases} \quad 4.3$$

and:

$$p(s_{t+1}|s_t) = \begin{cases} \gamma & \text{if } s_{t+1} = s_t \\ 1 - \gamma & \text{if } s_{t+1} \neq s_t \end{cases} \quad 4.4$$

In Equation 4.3, c and d allow for differentiation between rewards and non-rewards, which we do not exploit here so in practice $c = d$ for our purposes. The γ parameter in Equation 4.4 is interpreted as the probability of staying in the current state. We have also extended the functionality of γ with an additional parameter δ so that the probability of staying in the current state decreases in probability for

each timestep. In order to not have γ go below zero, we change γ on each timestep like so:

$$\gamma_{t+1} = \max(\gamma_t - \delta, 0) \quad 4.5$$

When a switch in state happens – when the prior and posterior differ in what state has the highest probability – then γ is reset to its initial value. We will use two versions of the HMM agent. The original, named HMM, and the extended one named HMM- δ . As usual, the code implementation can be found in the thesis code repository²⁰.

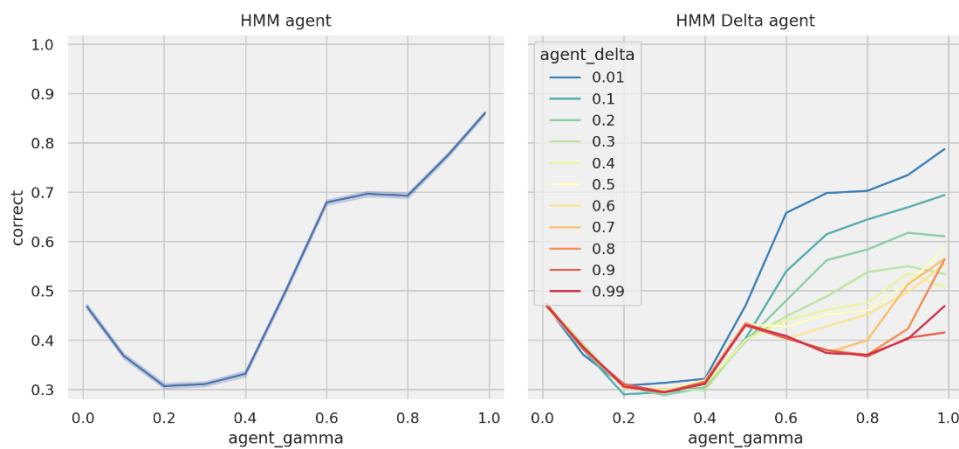


Figure 4.10 Proportion of correct choices (y-axis) in the ReversalBandit task for the HMM agent (left) and HMM- δ agent (right). Parameter γ is shown on the x-axis and each line in the right-hand plot is a different δ parameter value as per the legend.

In Figure 4.10 we have plotted the performance for the two HMM agents in our ReversalBandit task across their parameter spaces. The ranges used were the same for all parameters and are those seen in the right-hand figure legend. Each parameter combination has been simulated 100 times and the figure shows the average of these 100 runs. The HMM agent reaches its best performance at $\gamma=0.99$ with almost 90% proportion of correct choices. For HMM- δ we can see that the pattern is similar, the best combination is high γ , preferably combined with low δ . To study the behaviour in more detail, we select only the simulations of the best cases for each agent, which for both are $\gamma=0.99$ and for HMM- δ we have $\delta=0.01$. We plot these simsets in Figure 4.11, where we can see that the agents nicely follow the switch points through the experiment, but here the transitions are smoother

²⁰ https://github.com/fohria/phd_thesis

than in our “cheating” SEQL agent above. We also see that HMM- δ switches faster, thanks to the δ parameter influencing the probability to stay or switch action. The HMM agent is slightly slower to switch but reaches higher performance.

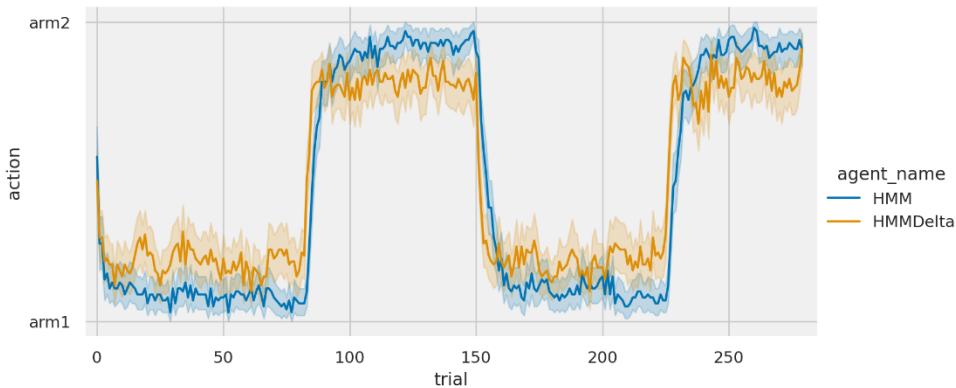


Figure 4.11 Proportion of action choices (y -axis; $arm1=0$, $arm2=1$) on each trial (x -axis) for 100 HMM agents all with parameter values $\gamma=0.99$ and $\delta=0.01$. Shaded areas indicate 95% confidence interval.

4.6 BEHAVIOURAL COMPARISON OF AGENTS

Before we simulate and fit all the models to see how well our model selection methods work, it would be good to compare the behaviour of our models. For each agent we select a “good” parameter combination – one that is among the best performing parameter combinations for that agent. Performance here means proportion of correct choices across the experiment. Then we compare agent performance in two ways. First, the average performance, based on proportion of correct choices during the task. Second, we compare switch behaviour after each switch as that seen for the human subjects in Figure 4.5. We also add the human subject data in these plots.

The parameter values chosen, guided by the above results, for each agent, were as follows.

QL2: $\alpha = 0.4, \beta = 10$

SEQL2: $\alpha = 0.4, \beta = 10$

Dual- α : $\alpha_{pos} = 0.4, \alpha_{neg} = 0.6, \beta = 20$

Dual-Update: $\alpha = 0.8, \beta = 20$

HMM: $\gamma = 0.99$

HMM- δ : $\gamma = 0.99, \delta = 0.01$

RandomBias: $bias = 0.5$. The reasoning for this value for RandomBias is because trial lengths are roughly the same for switch intervals. The best performance for the RandomBias agent would thus be to have equal probability for both arms.

For all the agents, we run each 100 times and average the results.

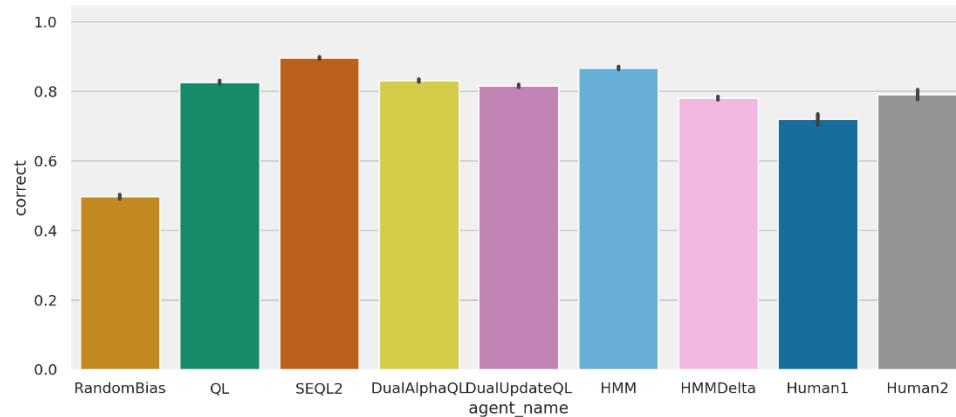


Figure 4.12 Proportion of correct choices (y-axis) in the ReversalBandit task for our agents and two human groups (x-axis). Error bars indicate 95% confidence interval.

In Figure 4.12 we can see that most of the agents perform similarly across the experiment. As a reminder, the Human1 group are those who did the task for the first time and the Human2 group did the task their second time. Unsurprisingly the SEQL2 agent performs best, since it has access to information that the other agents or humans do not. But the HMM agent is very close. The other noticeable outlier is the Human1 group, with slightly lower performance, but still higher than the RandomBias agent. It's quite understandable that humans learning the task for the first time will vary more in their behaviour.

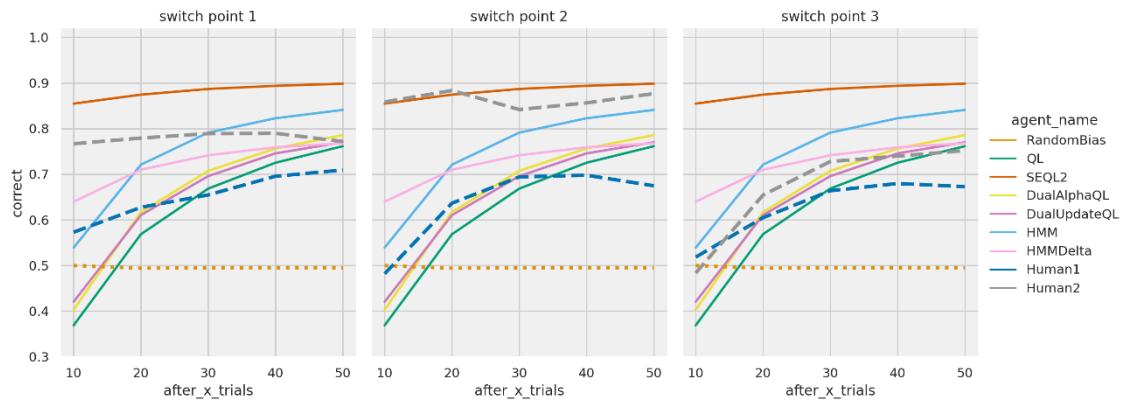


Figure 4.13 Proportion correct choices (y-axis) at different number of trials after each switch point (x-axis). Each line represents a different agent as per the legend. Human “agents” are differentiated by dashed lines and our baseline the RandomBias agent is dotted. Left: First switch point. Middle: Second switch point. Right: Third switch point.

In Figure 4.13 we have plotted the proportion of correct choices after each switch point for all our agents and the two human groups. As we saw earlier, the Human2 group performs very well at the second switch point, almost as well as our cheating SEQL2 agent. For switch points one and three we can see that the human groups are closer to the other agents. It will be quite interesting to see what our model selection algorithms determine is the better fit for these humans across the entire task.

4.7 PARAMETER RECOVERY PERFORMANCE

As discussed in the previous chapter, simulating behaviour algorithmically is slightly different from fitting models. The latter involves creating likelihood functions, and these may not always be well behaved. In the case of our ReversalBandit, we can use the same likelihood functions as in the previous chapter, since the QL likelihoods only use actions and rewards as the behavioural observations. SEQL2 being the exception, as it needs to be fed the stimuli as well. Overall, the QL agents are well-behaved and similar in parameter recovery to what was explored in the previous chapter so they will not be presented here²¹.

We will however look closer at the two HMM agents, as we are not yet familiar with the recovery performance of them. We perform the parameter recovery check as in the previous chapter. That is, for each

²¹ But see code repository at https://github.com/fohria/phd_thesis

agent we simulate 1000 agents, each with a randomly selected parameter combination. For each agent we then fit its corresponding likelihood function to itself to see if parameter values can be recovered. Parameter values for the agents were generated as:

$$\gamma_{HMM} \sim U(0, 1), \quad \gamma_{HMM\delta} \sim U(0, 1), \quad \delta_{HMM\delta} \sim U(0, 1)$$

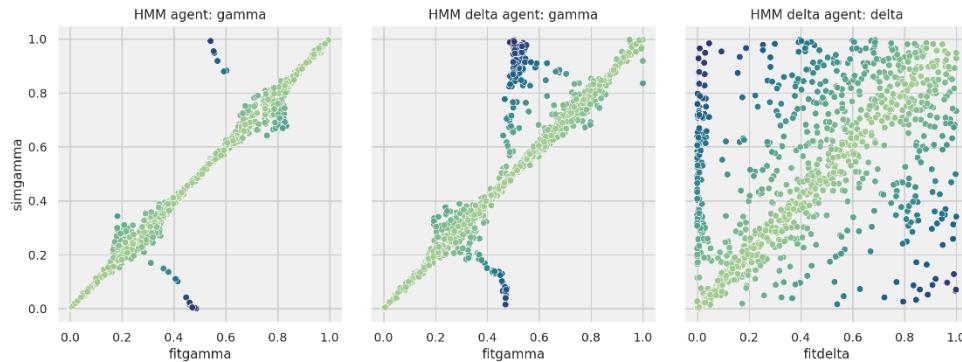


Figure 4.14 Simulated parameter value (y-axis) plotted against fitted parameter value (x-axis). Left: HMM agent γ parameter. Middle: HMM- δ agent, γ parameter. Right: HMM- δ agent, δ parameter. Colour shading indicate the absolute distance between the simulated and fitted value, where lighter green is low distance and darker blue is large distance.

In Figure 4.14 we see the recovery results. We see that for the HMM agent, parameter recovery is good. Only a few of the thousand cases are off, and not by extreme amounts. But for the HMM- δ agent the story is not quite as good. We have a similar pattern for the γ parameter, but it's worse looking for HMM- δ . The δ parameter itself is okay in many cases, but the wide spread indicates that any single fitted value has high uncertainty.

Why this happens is because our introduction of δ becomes problematic in the likelihood function. We reset γ after internal switch points, where – as described above – the posterior and prior beliefs differ in what state is more likely. We also “bottom out” γ (Equation 4.5) to make sure that γ cannot become negative. This causes the likelihood surface to have sharp drops or at least become less continuous. We saw an example of this in Figure 3.8. This may perhaps be accommodated by using more complex HMM models, as demonstrated by for example [165], further discussed below in future work.

The good news is that as we saw in the previous chapter, parameter recovery quality is not directly correlated with model selection performance. The less good news is that hierarchical

methods like the CBM toolkit will struggle to fit models with badly behaving likelihoods and require much longer computation time, making it disruptive to do larger scale model selection investigations. Simulating and fitting 1000 HMM- δ agents using MLE can be counted in seconds. CBM HBI requires around 10 minutes to fit 100 subjects, but around 2 hours to fit 200 subjects. Even then, results are not noticeably better – perhaps even worse – than MLE fitting, as seen in where CBM-HBI simfits for 200 subjects are shown.

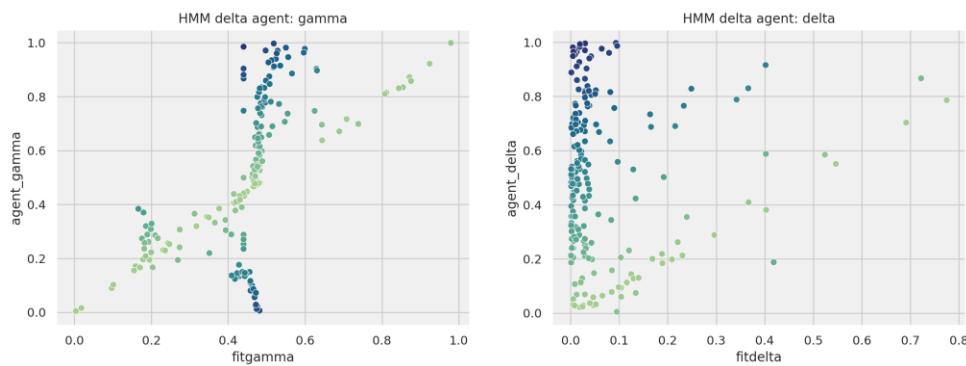


Figure 4.15 Simulated parameter value (y-axis) plotted against CBM-HBI fitted parameter value (x-axis) for subjects simulated with and fitted with HMM- δ . Left: γ parameter. Right: δ parameter. Colour shading indicate the absolute distance between the simulated and fitted value, where lighter green is low distance and darker blue is large distance.

The huge increase in computation time between 100 and 200 subjects is partly due to CBM HBI (like other Bayesian hierarchical models) by its nature uses data from all subjects instead of fitting them individually. And partly due to more subjects increases the risk of “troublesome” subjects, meaning data sequences that are difficult to fit. These issues arise when fitting the HMM- δ model to subjects simulated with the same algorithm. When fitting the model to subjects simulated with other algorithms, the problem is exacerbated, and even more time is required. In such troublesome cases, even the Laplace fitting part can fail and falls back to using estimates based directly on the priors. Meanwhile, the HMM agent with CBM for the same 100 cases takes less than a minute.

4.8 MODEL SELECTION PERFORMANCE

The final investigation we need to do before fitting models to our human data is to check how well our model selection methods can identify and differentiate different models. It is to be expected that

some cases of *QL2* may be identified as *Dual- α QL* and vice versa, since the former is a subset of the latter. On such occasions it should be the case that *QL2* is preferred, since we use BICs and thus prefer models with fewer parameters. The same reasoning is likely to hold for the *HMM* and *HMM- δ* agents as well.

To recap, we have seven models in total that we are going to simulate, fit and then identify based on the fits. These models are *QL2*, *RandomBias*, *SEQL2*, *Dual- α QL*, *Dual-Update QL*, *HMM* and *HMM- δ* . We simulate 200 subjects for each model, each simulation with separately randomly generated parameter value combinations. We thus get a dataset with 1400 subjects. All seven models are then fitted to all 1400 subjects using MLE, from which we get a likelihood that is converted to BIC values. These BIC values are then fed into VBAT for final model comparison and selection.

We also use CBM, but there we exclude *HMM- δ* from the fitting process, since the likelihood as described above is not well-behaved²². This is unfortunate as we would have ideally liked to test the CBM paper [204] claim that it is better than MLE at identifying nested models. We could however test this claim with the *QL2* and *Dual- α* models, which are the same two models tested in their paper and these are also included by default in CBM. We used 1000 simulated agents (paper uses 40), with the same ratio of 25% *QL2* and 75% *Dual- α* . Their results showed an almost perfect identification of 25/75, but we failed to replicate this finding and CBM reported a 54/46 split of cases²³.

To be fair, [204] uses a different reward structure for their two-armed bandit task called a binarized Gaussian random-walk. The precise details of their task are unfortunately not well described in the paper, or the accompanying code, and we thus chose to use the same ReversalBandit task described above. It is of course likely we could replicate their results if we used the same task. However, the relevant conclusion from this replication failure is that, as was thoroughly discussed in the previous chapter, model selection is notoriously difficult and it's imperative that multiple task and agent scenarios are

²² If we assume that fitting *HMM- δ* to data generated by other models takes the same amount of time as fitting 200 *HMM- δ* subjects, that would be 14h for 1400 subjects. More likely it would take longer for reasons stated in previous section.

²³ Code for results can be found in thesis code repository

explored when making any claims that one method is better than another.

Furthermore, even without *HMM*- δ in the mix, CBM is slower to compute results compared to MLE. All the 1400 subjects take a minute or so to simulate, fit and analyse with BIC and VBAT. For CBM, which fits 1200 cases as we removed *HMM*- δ , the Laplace phase of CBM fitting takes around three minutes and is done using multiple processes with MATLAB's parallel pool functionality. The hierarchical fitting phase of CBM is not possible to do in parallel and takes around 16 minutes, so CBM fitting is roughly 19 times slower than MLE/VBAT. This is not terribly slow, but it's worth noting as these tests have been done on a relatively recent and powerful laptop²⁴, so with weaker hardware and depending on the performance of CBM compared to BIC/VBAT, it is something worth taking into consideration when selecting one's method both timewise and energy efficiency/sustainability-wise.

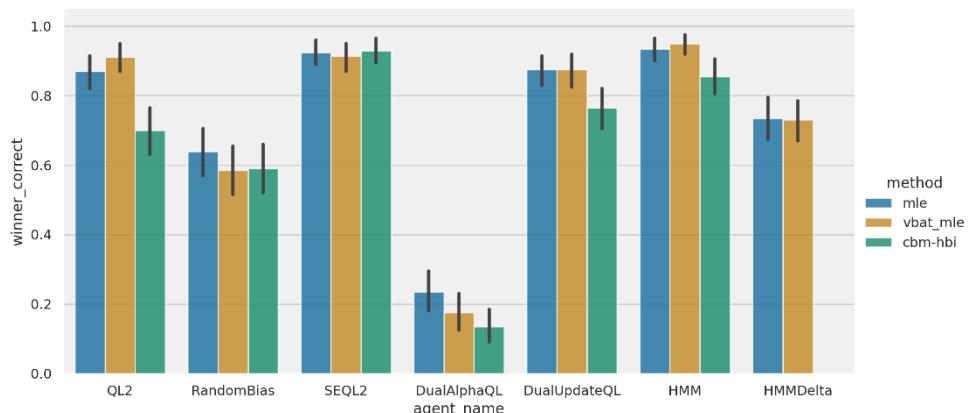


Figure 4.16 Proportion of correctly identified cases (y-axis) for each method (colours as per legend). X-axis labels represent the simulated agent name. Error bars are based on 95% confidence interval.

In Figure 4.16 we have plotted the model selection results for MLE/BIC, VBAT (using BIC from MLE fits) and CBM. For the MLE-based methods, most of the agents can correctly be identified more than 80% of the time, often upwards of 90%, which is a decent and promising result for fitting the human data. CBM generally performs worse and is at best on par with the other methods. More generally, we can see that *Dual-* α *QL* is difficult to identify, most likely due to

²⁴ CPU in laptop is a Ryzen 7 5800H, 45W

QL_2 model being in the mix as discussed above. Similar reasons are likely behind the poor identification for the $HMM-\delta$ agent, as the HMM agent is nested within $HMM-\delta$ agent, in addition to the issue with the troublesome likelihood function for the $HMM-\delta$ agent. Looking further at the comparatively low identification performance for RandomBias, it's likely due to some random cases can look like greedy QL agents. We can see these theories are true by looking more specifically at what model each subject has been identified as. This is shown in Figure 4.17, where we exclude the pure MLE method for readability.

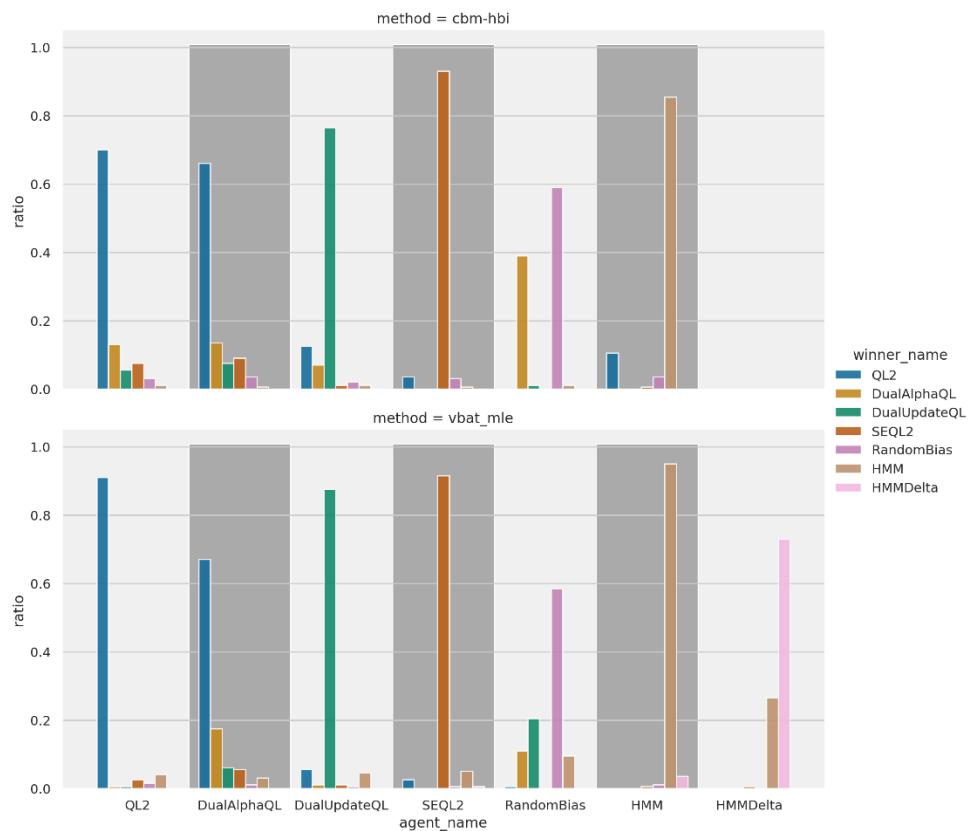


Figure 4.17 Proportion of cases (y-axis) for each simulated agent (x-axis) identified as what model (differently coloured bars as per legend). Shaded sections separate each x-axis category for readability. Top: CBM method. Bottom: VBAT method.

In Figure 4.17, we see that as we suspected, most of the $Dual-\alpha$ cases are identified as QL_2 . We can further drill down into this to confirm that many of these cases are such that the difference between the positive and negative α in $Dual-\alpha QL$ is < 0.5 on average²⁵. Due to the inherent randomness of action choices in these tasks it is

²⁵ Result not shown but plot available in code repository

understandable these misidentifications happen. Similarly, for the *HMM- δ* agent, we see that almost all cases not correctly identified as *HMM- δ* are instead identified as *HMM*.

For the comparison of VBAT and CBM methods, the biggest deviation is how simulated *RandomBias* agents are almost exclusively misidentified as *Dual- α* for CBM, but spread across *Dual- α* , *Dual-Update* and *HMM* for VBAT. The other deviation – and arguably more important – is that CBM misidentifies *HMM* as *QL2* for roughly 10% of the cases. Because what we are interested here is investigating whether participants use the concept of states or rely on “simpler” action-values, the VBAT result here is much more in line with what we would like to see and therefore will be more reliable for our human data with regards to this question.

Overall, these results are encouraging for fitting our human subjects. We can be reasonably sure that the model fitted describes the human behaviour well. It is especially encouraging that the two HMM models – which have the concept of states – can be reliably distinguished from the stateless models (QL family).

4.9 FITTING HUMAN DATA IN THE REVERSAL BANDIT TASK

We now have enough information to fit these six models to our human data and be able to interpret the results. To reiterate, the human data consists of 23 subjects playing the ReversalBandit task. Half the subjects had done a very similar task once before and half had not, which we indicate by referring to them as groups 2 and 1 respectively. The exact timesteps when switches occurred differed slightly between the two groups, as mentioned above.

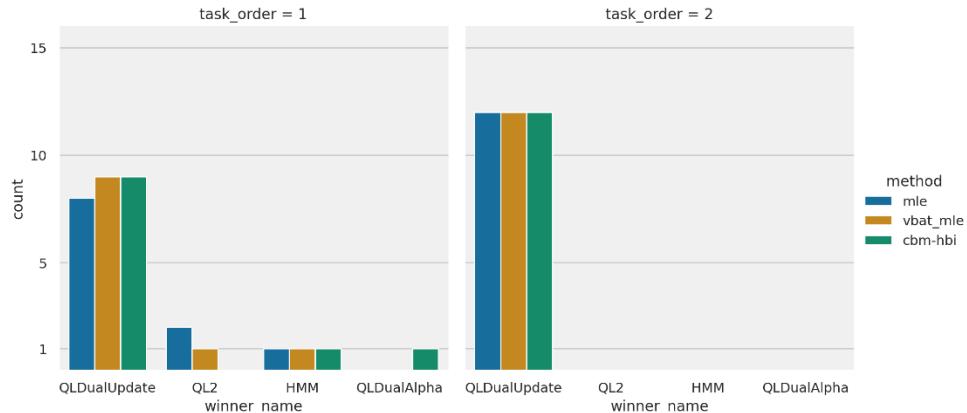


Figure 4.18 Number of subjects (y-axis) that were best fit with each model (x-axis). Coloured bars indicate method type as per legend. Left: Human group that did the task the first time. Right: Human group that did the task their second time.

In Figure 4.18 we can see that – somewhat surprising – most subjects appear to perform in line with the *Dual-Update QL* model. For the participants that did the task their first time (left hand side of the figure), their behaviour is more scattered and thus in a few cases better fit by other models. Moving on, we see that the CBM results, although somewhat more unreliable as we saw in our simfit model selection investigations, largely agree with the BIC/VBAT results, except for the one subject identified as *Dual- α* by CBM. This adds to the picture that the humans in group1 are more variable in their behaviour, but largely use the same strategy as the humans in group2.

As we have seen earlier, an advantage of VBAT and CBM is that we also get probability measures of each model's fit to each individual subject. This has been plotted per subject and method in Figure 4.19, where we see that overall, the methods agree on individual subject level what model fits best. An interesting case is subject 3 where CBM is convinced it's *Dual- α QL*, and VBAT is not fully certain this subject is *Dual-Update QL* but gives some chance to it being *QL2*. Subject 8 is a more clear-cut disagreement. But the most interesting case here is subject 5 where all three methods (MLE result not shown in figure) agree that *HMM* fits best. As we saw in Figure 4.16 and Figure 4.17, our methods are very good at identifying *HMM* cases. So, it could very well be that this particular subject thinks about the task in terms of states.

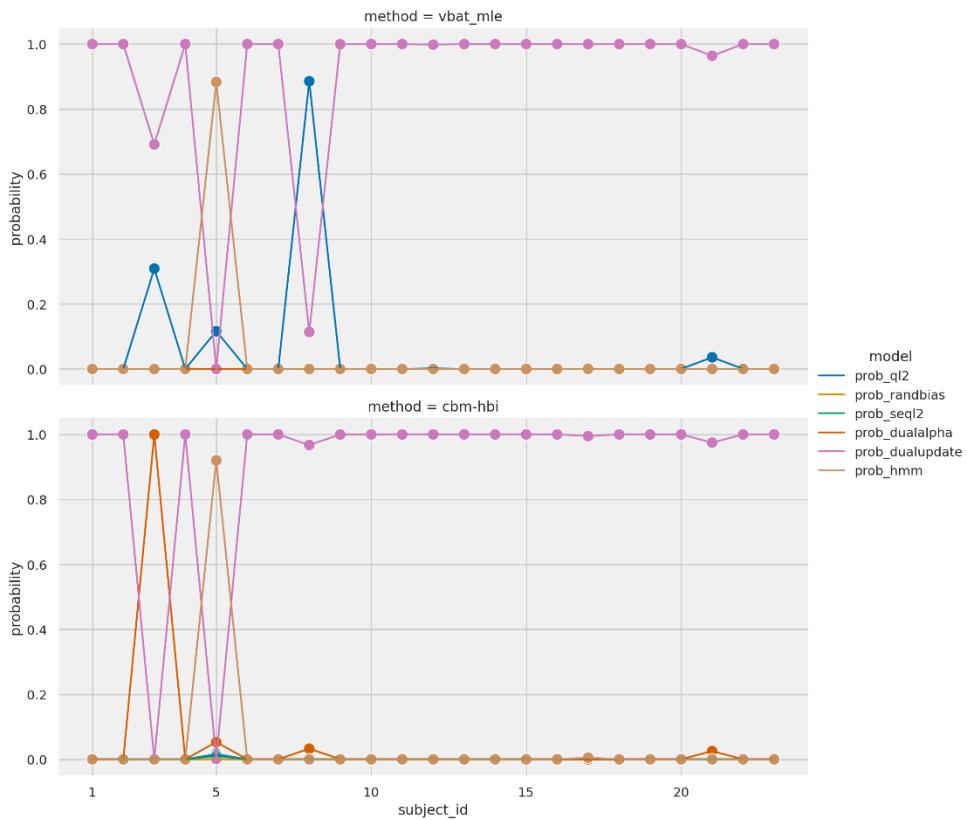


Figure 4.19 Model probability (y-axis) for each subject (x-axis). Separate lines and colours indicate model identity as per legend. Top: VBAT method. Bottom: CBM method.

What our investigations here show is that for this particular dataset and the particular models we have chosen to fit, there is overall no need for the concept of states to be introduced. Here, it seems more likely that most subjects keep track of the value of both arms at once. This is perhaps because they know that one arm is always better than the other, so if one arm is rewarded that automatically means the other one is bad. Hence, updating the values for both arms simultaneously make sense. Most published papers would stop there and conclude that *Dual-Update QL* is how humans approach this task. But humans are not all the same, individual differences do exist. So, if we are to believe our data, at least one individual could very well be approaching the task in terms of conceptualising it as states (subject 5 in the figures above). And for similar reasons it seems reasonable that when first encountering a task, humans attempt different strategies, which could explain how the fit results are more scattered for the subjects in group1.

4.10 INVESTIGATING THE WORTHY BANDIT

As mentioned in this chapter's introduction, there are many variations of the reversal bandit task. One such variation is one where reward magnitudes are varied together with what action option is the best one. The specific one we will focus on here is one we call WorthyBandit after the main author of the paper the task is taken from [296]. Subjects have two decks of cards to choose from, a Good Deck (GD) and a Bad Deck (BD). The BD provides high points for early cards drawn but then goes down-hill, while the GD starts out providing low points and then progressively get better. Subjects start out at zero points and the goal is to maximize their score. In most versions of the task, there is a goal criterion of 450 points to receive some additional reward like taking part in a raffle to win a prize. In order to reach this criterion, the subjects must resist exploiting what initially looks like a good deck and keep exploring the deceptively worse option. Note that the concept of card decks here is merely superficially different from the Bandit and ReversalBandit tasks above. They are all the same kind of tasks; two options for actions that each provide a reward.

In more detail, the points awarded for each card are between 1 and 10 and after being drawn, that card is discarded. Each subject draws a total of 80 cards (i.e., the task has 80 trials) and each deck has 80 cards (thus 160 cards in total across the two decks). The GD gives an average value of 3 points over the first 20 cards drawn, an average of 7 points over the next 50 cards drawn and an average of 3 points over the last 10 cards drawn from it. The BD provides an average value of 8 points for the first 30 cards drawn from it, an average of 5 points over the next 20 cards drawn and an average of 2 points over the last 30 cards drawn from it.

We have gained access to a dataset consisting of 166 human participants performing this task. Details of participants can be found in [239]. All participants had the same specific point sequence for the drawn cards. Thus, for consistency, we use the same sequence for all simulations below. The sequence has been plotted in Figure 4.20.

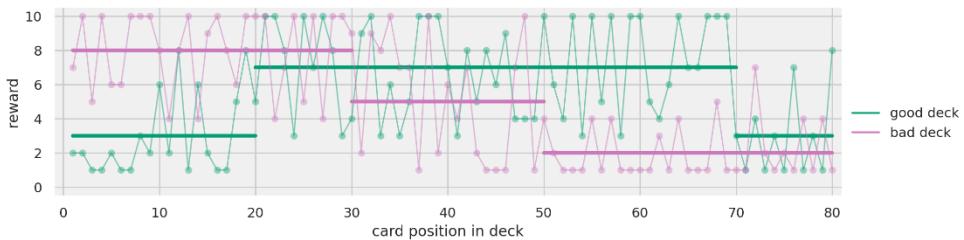


Figure 4.20 Reward magnitude (y-axis) for each card in the decks (x-axis). Colours differentiate decks as per legend. Horizontal lines represent mean reward for that card position range.

To better understand how deck choice impacts score in the task, we can plot the number of cards picked from each deck and the resulting score, as seen in Figure 4.21. There we can see that to maximise the points the participant needs to pick roughly an equal number of cards from each deck. Because we can only pick 80 cards in total, the two card counts are directly correlated, as seen in the figure's symmetry. The maximum score is 514 and the minimum score is 392. A minimum of 25 cards drawn from the good deck are needed to reach the goal criterion of 450 points. If more than 78 cards are drawn from the good deck, the final score drops below the goal.

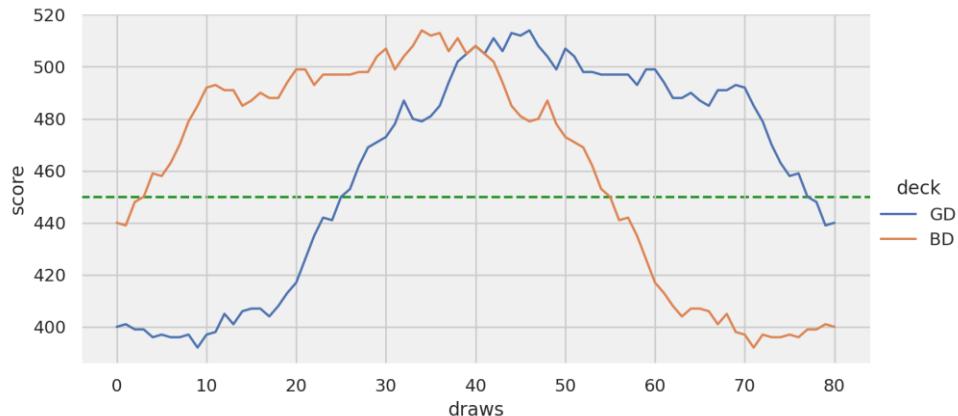


Figure 4.21 Total score in the task (y-axis) depending on how many cards are picked (x-axis) from each deck (coloured lines as per legend). The green dashed line indicates the goal criterion of 450 points.

4.10.1 AGENT PERFORMANCE IN THE WORTHYBANDIT

As mentioned above, there is a goal of having at least 450 points when the task ends. So, measuring performance in this task is quite natural – we measure the total points achieved. The other thing we would like to see is how the choice behaviour develops during the task run,

which we do by plotting action choices per trial averaged over subjects. For an additional measure, we also calculate the proportion of deck1 choices from the current trial to the end of the experiment. In other words, on the first trial, this measure is the proportion of deck1 choices across all the 80 trials. On the second trial, it's the proportion across the last 79 trials, and so on. On the last trial this measure can thus only be one or zero since it is only based on the final trial of the task. This measure allows us to see on an individual level how the choices change throughout the experiment.

We start out by looking at the behaviour of our *QL2* agent. We perform a parameter sweep, with 100 subjects for each parameter combination. The parameter ranges used are:

$$\begin{aligned}\alpha &\in (0.01, 0.1, 0.2 \dots 0.8, 0.9, 0.99) \\ \beta &\in (0.5, 1, 2, 5, 10, 20)\end{aligned}$$

Furthermore, we will also contrast three different reward functions for the agent. When rewards are between 1 and 10, this can lead to Q-values quickly inflating – especially if α is large – so that the action that received a reward of 10 will keep being the highest valued one until the other action has received a reward of 10 itself or enough lower rewards to compensate. But the latter may not happen with a greedy agent. Of course, one may argue that the β parameter then can be lower and exploration can still happen. However, with the inherent randomness of these algorithms it is very difficult to predict how they will perform, especially for a task like the WorthyBandit that has a quite complex reward schedule. Thus, our three reward schedules are standard, normalised and scaled, where standard are rewards between 1 and 10 as for the humans. The latter two are calculated as follows:

$$r_{\text{normalised}} = \frac{r - r_{\min}}{r_{\max} - r_{\min}}, \quad r_{\text{scaled}} = \frac{r}{10}$$

The difference between these two are that for normalised, the lowest reward value – 1 – will become 0, whereas for the scaled function the lowest reward value will be 0.1. This may seem strange, as 0 is usually considered as “no reward”, but since participants know the reward range it is possible that a reward of 1 is evaluated as being practically 0.

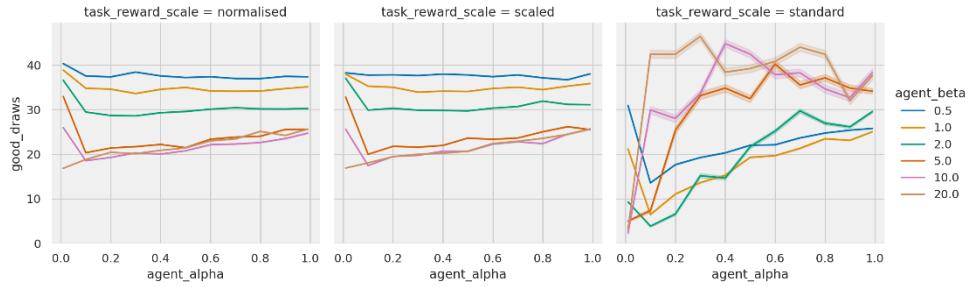


Figure 4.22 Number of cards drawn (y -axis) from the good deck for each α (x -axis) and β (coloured lines as per legend) parameter value combination. Left: Normalised rewards. Middle: Scaled rewards. Right: Standard rewards. Shaded areas indicate 95% confidence interval.

In Figure 4.22 we have plotted the number of cards drawn from the good deck for all the parameter combinations and the three reward value variants. The normalised and scaled variants (left, middle, respectively) perform very similarly, and in both cases the pattern is that lower β and very low α leads to more cards picked from the good deck. Meanwhile, for the standard variant (right plot of Figure 4.22) we seem to have the inverse relationship for β .

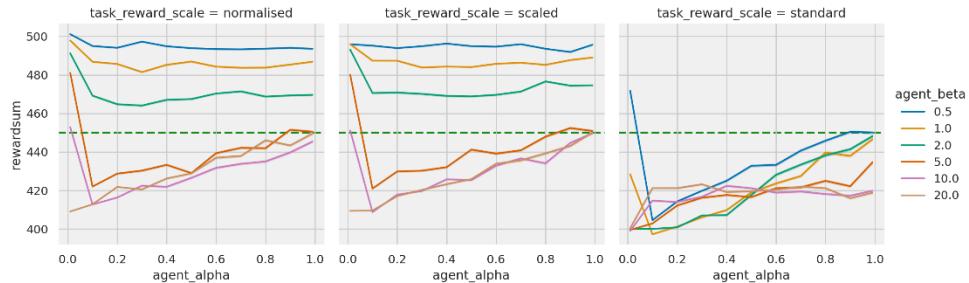


Figure 4.23 Total reward score (y -axis) for each α (x -axis) and β (coloured lines as per legend) parameter value combination. Left: Normalised rewards. Middle: Scaled rewards. Right: Standard rewards. Shaded areas indicate 95% confidence interval. Green dashed horizontal line indicates goal criterion.

Moving onto Figure 4.23, the standard variant (right plot) again stands out. How come almost no cases reach the goal criterion, when in the previous figure there were many cases reaching around 40 good draws? We can see why in the histogram in Figure 4.24, where we have selected the specific case of the simset (100 simulations with the same parameter values) with $\alpha=0.4$, $\beta=10$ for the standard reward scale. Practically all cases pick one deck, and then greedily pick the same deck throughout the entire experiment. This rarely happens with human participants (see below).

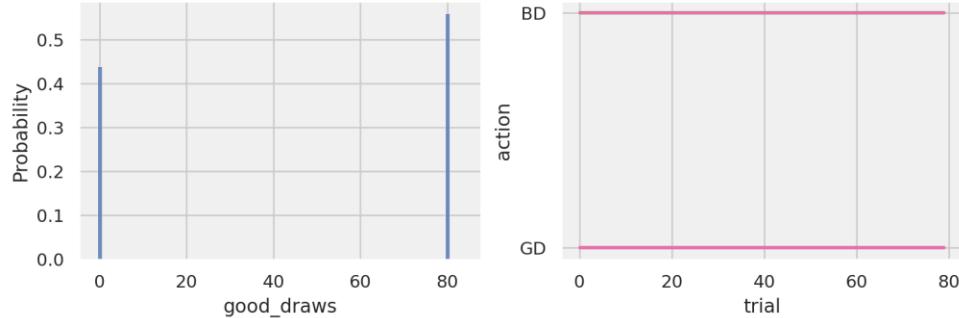


Figure 4.24 Closer look at the parameter combination $\alpha=0.4$, $\beta=10$ with standard rewards. Left: Histogram showing probability (y-axis) of picking a certain number of cards (x-axis) for all 100 subjects averaged. Right: Action selected (y-axis) across all trials (x-axis) for individual agents (separate lines). GD=good deck, BD=bad deck. Note there are 100 subjects, but since they all pick either BD or GD entire task length, the lines overlap.

Recall that β regulates exploration versus exploitation – higher β results in more greedy behaviour, always selecting the action with the highest Q-value. With lower β , exploratory actions are made more often which is an advantage in this task as mentioned above. A certain number of cards must be picked from the good deck before it becomes good. So, what happens here when using standard reward values, is that the QL2 agents with high β immediately get such a high Q-value for one deck compared to the other that it sticks to it throughout the entire experiment. Hence why the greedy agents average to around 40 draws from the good deck in Figure 4.22, because they either pick 80 from that deck, or 0. In other words, we must conclude that either normalised or scaled variant should be used here²⁶.

²⁶ We could theoretically introduce a reward scaling parameter into the algorithm, as demonstrated by e.g. [225]. That is not really the focus for us however, so we leave this for other authors.

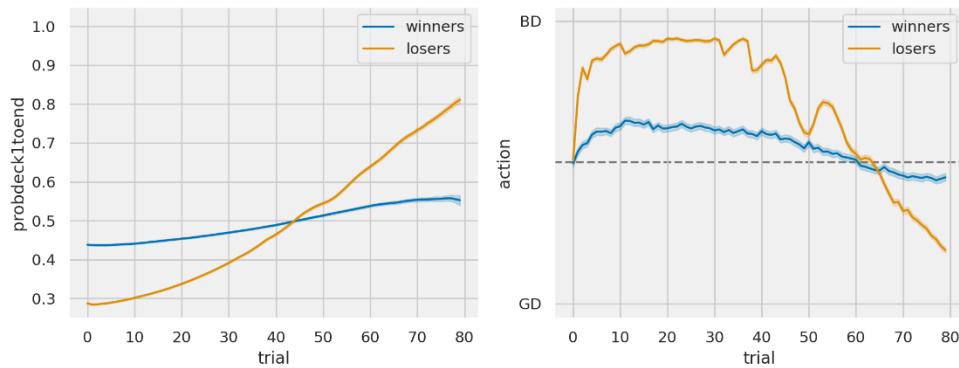


Figure 4.25 Behavioural differences between winners and losers. Left: Probability of selecting a card from deck1 (good deck, y-axis) relative to choices on successive trials (x-axis). Right: Proportion of choices being bad deck (BD) or good deck (GD) on y-axis across all trials on x-axis. Strictly it is the proportion of BD i.e., the line at BD is 1 and the line at GD is 0. Dashed line indicates the midline, i.e., equal probability of each option.

In Figure 4.25 we divided all the simsets (one simset is 100 simulations of the same parameter value combination) into two groups – those that reached at least 450 points (winners) and those that did not (losers). In the left part of the figure, we see the probability of picking the good deck – deck1 – in proportion to all the successive choices, as explained earlier. In both plots in the figure, we see how winners have a more balanced approach, while the losers aggressively pick the BD early on and then drastically switch to the GD as the BD runs out of big rewards. Note that the two plots in principle shows the same information, only presented from different viewpoints. For later plots of agent simulation performance, we thus stick to one of these, but see below for the human participants.

We now move onto performance for the *RandomBias* agent and do a parameter sweep. The parameter range used for this parameter sweep are as follows:

$$bias \in (0.01, 0.1, 0.2, \dots, 0.8, 0.9, 0.99)$$

100 subjects were simulated for each parameter combination, and results are seen in Figure 4.26.

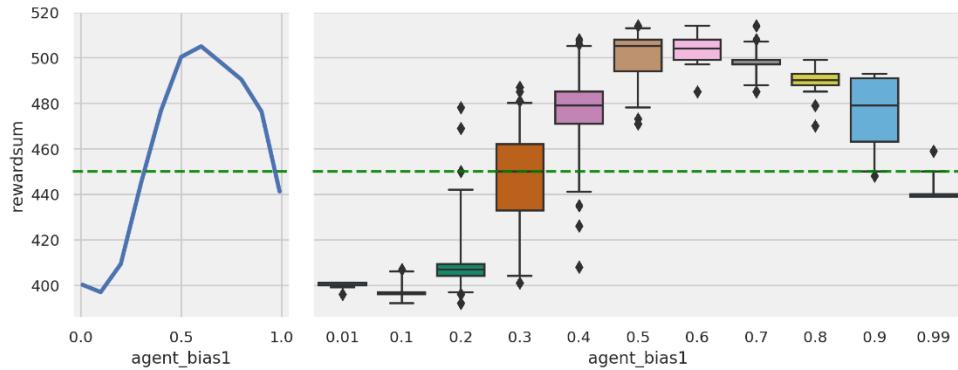


Figure 4.26 Performance for *RandomBias* agent (y-axis, total points scored) across the parameter value space (x-axis). Green dotted line indicates the goal criterion. Left: Average score for each parameter value. Right: Boxplots for each parameter value.

In the figure, we have combined results for all three types of reward scales as they show virtually identical results. We here see that a wide range of values for the bias parameter result in scores above the goal criterion. For *bias* 0.5 to 0.8, 100% of the simulations are above the criterion, as seen in the boxplots to the right in the figure. In total, 57% of all random cases are above the criterion. This is unfortunate, as it shows that although the task is complicated there is no clear measure for us to know if humans are learning anything in the task or picking actions at random. It is still possible that, depending on the behaviour of humans, some models we fit can better explain their behaviour than a random model. For example, if many humans struggle to get enough points, this may indicate that they are learning, not picking at random, but learning the wrong thing. However, if we rely on fitting models to know if humans have learned anything at all, that makes our investigations much more challenging, since as we have seen many times above that fitting models at all is not reliable.

If we compare this to the *ReversalBandit* from earlier, there it's easier to see that humans are doing something slightly different from strict Q-learning as they often switch much quicker than *QL2* can.

When it comes to *SEQL2*, things get very tricky. How, if at all, do we define states in this task? The most straightforward way would be to use Figure 4.20 as a guide, and then define that state 0 is when GD is best and state 1 is when BD is best, based on how many cards have been picked from each deck:

$$state = \begin{cases} 1 & \text{if } count_{BD} < 30 \\ 0 & \text{if } count_{BD} > 50 \\ 1 & \text{if } count_{BD} < 50 \text{ and } count_{GD} < 20 \\ 0 & \text{otherwise} \end{cases}$$

The simulation results for *SEQL2* are virtually identical to *QL2* results, as seen in Figure 4.27. Note this figure only shows the normalised and scaled reward variants. The other QL models, *Dual- α QL* and *DualUpdate QL* also show the same patterns and are thus not shown here but are available in the code repository.

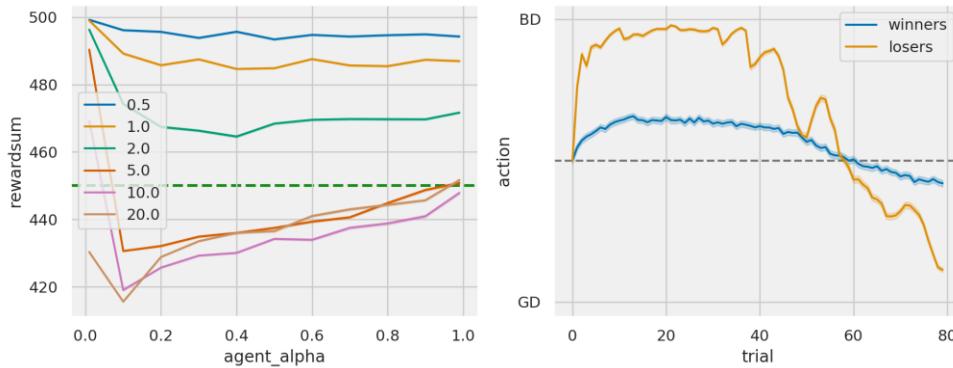


Figure 4.27 Simulations for *SEQL2* agent across the parameter space. Left: Total score received (*y*-axis) for different α (*x*-axis) and β (coloured lines as per legend). Green dashed line indicates goal score. Right: Proportion of cards picked (*y*-axis) from good deck (GD) and bad deck (BD) across trials (*x*-axis) averaged across all subjects but divided into groups as seen in legend.

The *HMM* and *HMM- δ* models have been altered slightly in order to accommodate the *WorthyBandit*. Instead of Equation 4.3, we are instead using a cumulative distribution function like so:

$$p(o_t|s_t) = \begin{cases} r \sim \Phi(\mu = 5.5, \sigma = 2) & \text{if } a_t = s_t \\ 1 - r \sim \Phi(\mu = 5.5, \sigma = 2) & \text{if } a_t \neq s_t \end{cases} \quad 4.6$$

Where the mean and standard deviations reflect the standard task reward variant. Since these parameters for the normal distribution are scaled accordingly with the scaled reward values, there is no difference between the task variants. Results for the parameter sweep for *HMM* can be seen in Figure 4.28. In the left part of this figure, we can see that similar to how optimal β for the QL agents switched between *ReversalBandit* and *WorthyBandit*, the γ parameter (probability of staying in the current state) is now optimal towards the lower end but can still be fairly high at 0.8 and still reach the goal criterion. On the right-hand side of the figure, we can see a similar pattern as for the other agents with regards to what deck winners and losers pick cards from. That is, losers greedily pick from the bad deck until around halfway into the task and only then gradually explore

the good deck, but fail to do this exploration enough before it's too late. While the winners have a more balanced approach throughout.

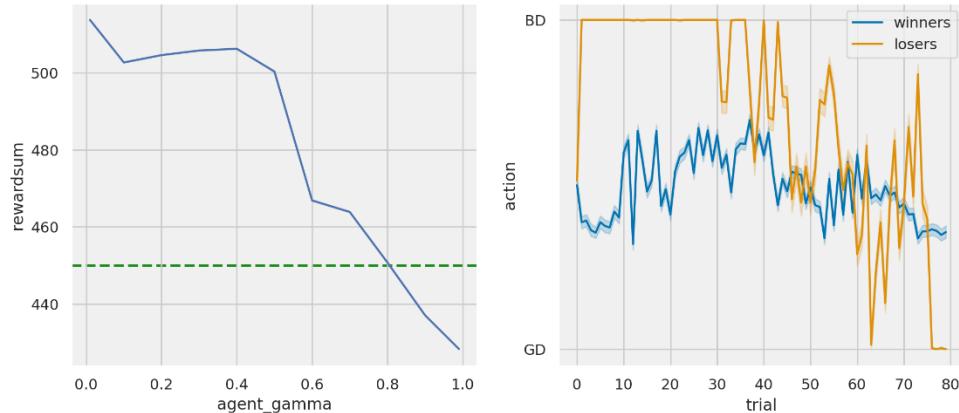


Figure 4.28 Performance and behaviour of the HMM agent in the WorthyBandit task. Left: Total score (y-axis) against γ value (x-axis). The green dashed line indicates the goal score. Right: Proportion of cards picked (y-axis) from the good deck (GD) or bad deck (BD) across trials (x-axis). The coloured lines indicate if the subjects averaged were winners or losers.

For the HMM- δ agent, results are seen in Figure 4.29. To the left, we see that with a very low γ , all combinations reach more than 510 score on average. Following those lines, we see that after $\gamma=0.5$ the variants start to diverge, where the line for $\delta=0.01$ has a similar trajectory to the regular HMM agent, which is also the only combination to ever go below the goal criterion. Only a few of these combinations reach the heights in score of the lower γ values, however. On the right-hand side of the figure, we again see the same pattern as we have seen for the previous agents, namely that winners have a balanced approach, whereas losers aggressively pick the bad deck for too long.

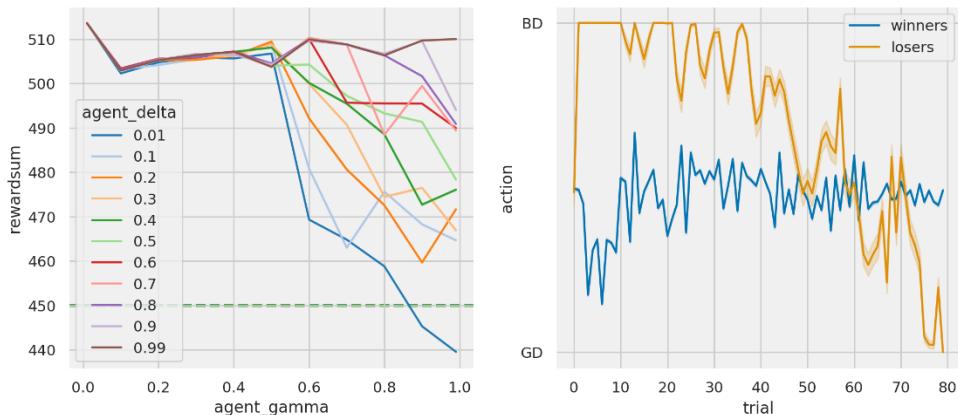


Figure 4.29 Performance and behaviour of the HMM- δ agent in the WorthyBandit task. Left: Total score (y-axis) against the γ value (x-axis) and δ value (coloured lines as per legend). The dashed green line indicates goal score. Right: Proportion of cards picked (y-axis) from the good deck (GD) or bad deck (BD) across trials (x-axis). The coloured lines indicate if the subjects averaged were winners or losers.

4.10.2 HUMAN BEHAVIOUR IN THE WORTHYBANDIT

For the human subjects we are presenting their scores and behaviour in similar ways as for the agents above. But here, since we have a low number of participants, we can plot their individual scores as separate points, as well as have a closer look at their behaviour on a subject level.

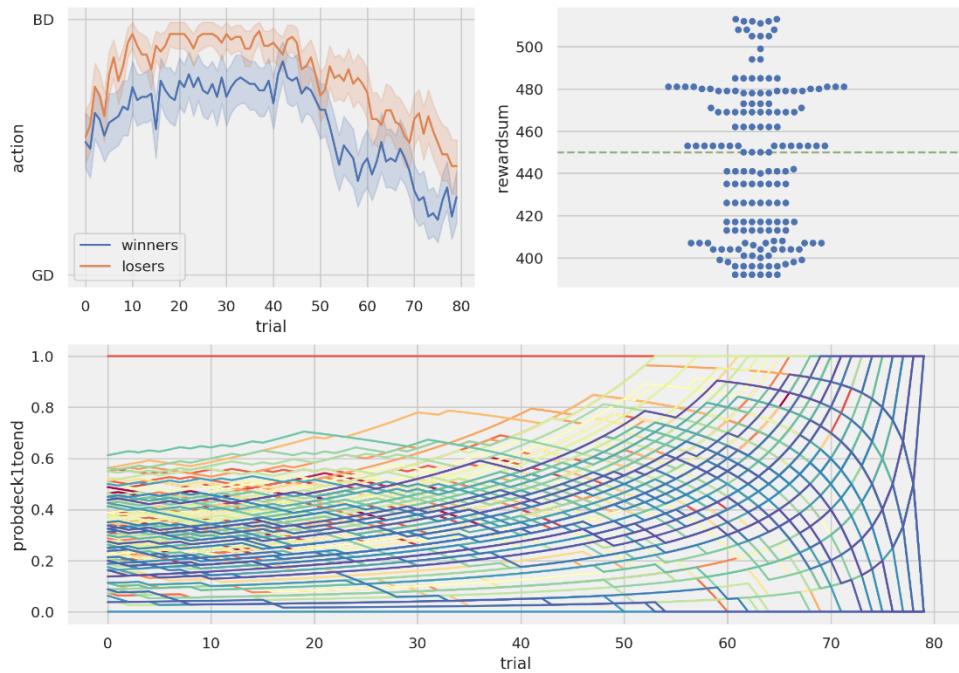


Figure 4.30 Human behaviour in the WorthyBandit task. Top-left: Average action choice (y-axis) for each trial (x-axis). Shaded area indicates 95% confidence interval. Top-right: Score (y-axis) for each participant. Each dot is a single participant. Green dashed line indicates goal criterion. Bottom: Proportion of choices that are deck1 (y-axis) for each participant (coloured lines) across all trials (x-axis).

In Figure 4.30 we have plotted the performance and behaviour of the human participants. In the top left we can see that overall, the winners tend to exhibit more exploratory behaviour than the losers. The winners also seem to react stronger to the shift to bigger difference between the decks after trial 50 (compare Figure 4.20). In the top right we see that only around half of the participants reach the goal criterion (47.6%, to be exact). In the bottom figure, which may look overwhelmingly busy at first, we can see the same overall pattern as in the top-left plot, namely that participants increase their picks of deck1 throughout the experiment overall, but here we also see that some participants notice the switch in the last 10 trials (lines slope downwards after a point) and some participants don't (lines continue upwards entire task). The observant reader can also spot that it looks like at least two participants pick the same deck throughout the entire task (one horizontal line at 1.0 and one line at 0.0). To doublecheck if there are any overlapping lines, we can check in the data, and indeed two participants pick the same deck the entire task. These subjects will most likely be best fitted by the RandomBias agent, or perhaps a very greedy QL agent.

Finally, to get an overview of all our agents together with the humans, we plot the total score for all agents (using the parameter sweeps described above) and the humans in Figure 4.31. We have not chosen specific, well performing, agent parameter combinations here. We also used a single reward type, normalised. Since there is a wide spread in performance for the human data, it is more appropriate to look at the spread for the agents to understand if there are parameter combinations that can account for all or some humans.

In Figure 4.31, we see that the human scores are spread in a uniform fashion across the possible scores. The HMM agents are better on average, with the *HMM- δ* agent performing very well. For the QL family of agents, *QL2* has the best average and smallest spread, but all those agents vary across the entire score spectrum, the same as the *RandomBias* and humans. What models may fit the human data best? Here it is possible that we may see a wider spread in model selection, for example that *HMM- δ* fits well performing humans well and *DualUpdateQL* fits the low-performing better.

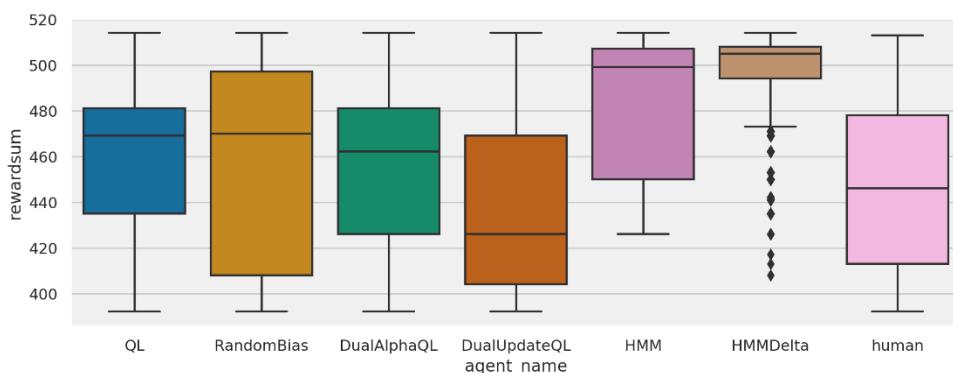


Figure 4.31 Score in the WorthyBandit task (y-axis) for each agent type (x-axis).

4.10.3 MODEL SELECTION PERFORMANCE IN THE WORTHYBANDIT TASK

With only 80 trials, fitting will be challenging. Parameter recovery will not be great, but luckily our focus here is model selection, which as we saw previously does not necessarily require good parameter recovery. Even this may prove challenging with so few trials. Additionally, the details of this task may impact how well model fits and selections work. This is the eternal challenge in experiments with humans in the cognitive sciences. Tasks that don't bore humans rarely

get us enough data, but if we extend the task, the data will be unreliable because humans are bored!

Nevertheless, like for the *ReversalBandit* above, we now simulate and fit 200 randomly generated agents of each type and see how well we can identify them. Because the QL agents do not work well with the standard reward scale, and the other agent types work well with any reward scale, we here only use the scaled reward scale (i.e., rewards are 0.1-1 instead of 1-10). We use this scale instead of the normalised one (which is 0-1) to have better correspondence with the reward values used in the human data. Except for the *SEQL2* agent, we can again use the same likelihoods as before, because the only data we have are rewards and actions. For fitting the *SEQL2* model to non-*SEQL2* simulations, we manually add states based on the card count, in the matter described above.

We unfortunately ran into issues and bugs with CBM when fitting this task. First, as with the *ReversalBandit* task, fitting the *HMM- δ* model is prohibitively slow so it is left out of this analysis. Second, the *Dual- α QL* model (which happens to be included by default in CBM) fits the data for some subjects so badly that the fitting algorithm reverts to using the prior value. This case of very bad fits seems to have some bug(s) in its implementation, causing cascading errors in the fitting process. The authors have been contacted but as of this writing there is no fix for this error, so *Dual- α QL* has also been left out of the analysis. Results for CBM model selections for the remaining models are still presented, but keep in mind the ratios cannot necessarily be directly compared to VBAT since the latter compares likelihoods across all the models.

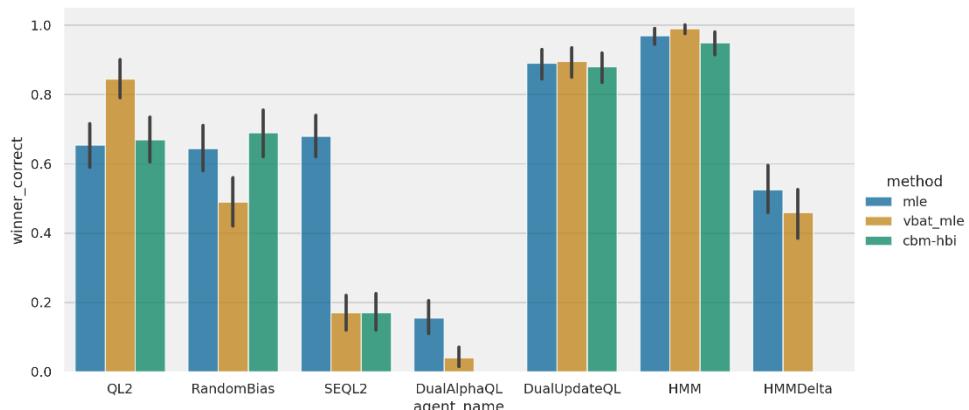


Figure 4.32 Model selection plots for the two methods (coloured bars as per legend), where the proportion of correctly identified cases is on y-axis, and the simulated agent type on x-axis.

In Figure 4.32 we have plotted the model selection results. What stands out is that *HMM* and *DualUpdate QL* are identified well by all methods, that MLE is multiple times better at identifying *SEQL2* than the other two and *Dual- α QL* is quite bad for all methods. CBM is, as has been seen before, not notably better than the other methods, except for the *RandomBias* subjects. It's possible that if CBM had been able to fit the additional two models its result would be closer to that of VBAT. To better understand what is going on here we need to move onto Figure 4.33.

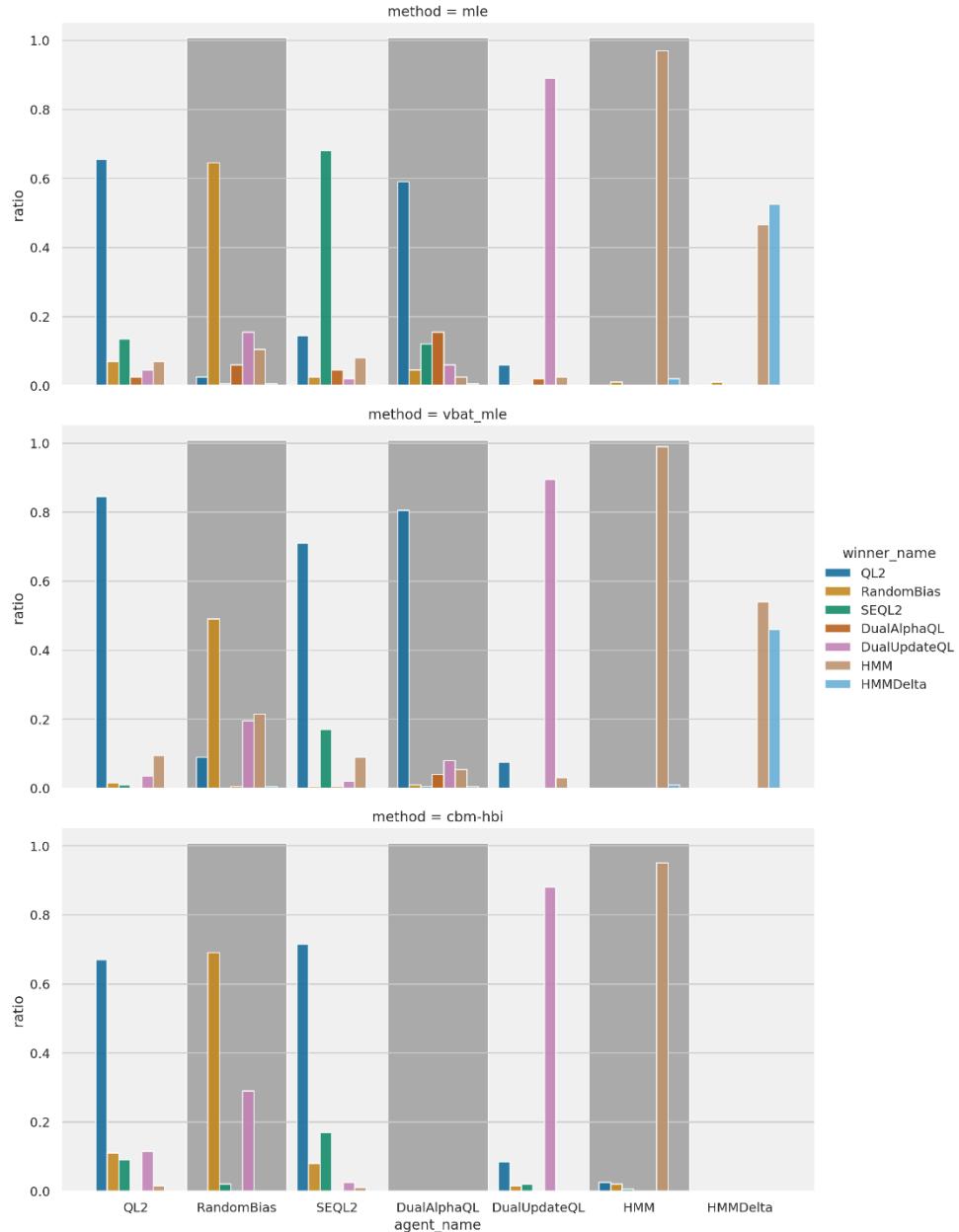


Figure 4.33 Proportion of cases (y-axis) for each agent type (x-axis) identified as what model type (different coloured bars as per legend). Shaded areas are simply for readability. Top: MLE method. Middle: VBAT-MLE method. Bottom: CBM method.

In Figure 4.33 we see the full model selection results, not just accuracy as in the previous figure. The most important finding here is that there is a clear separation between the HMM models and the QL family of models – practically no HMM cases are identified as QL and vice versa. Most HMM- δ cases that are misidentified are identified as HMM, which makes sense as they are nested. Unfortunately, it looks like between 20-40% of RandomBias cases can be identified as HMM, depending on the method. That means we cannot be fully certain a human best fitted with HMM actually uses

this strategy. For the QL family, we see that except for *DualUpdateQL*, which has high accuracy overall, VBAT tends to classify *Dual- α* and *SEQL2* as the standard *QL2*. *QL2* is nested in *Dual- α* so that part can be partly explained, and perhaps it is the case that the addition of states in *SEQL2* does not help or differentiate very much in the WorthyBandit for there to be much difference in their behaviour. Because of the difference in results for MLE and VBAT, for example that MLE is better at identifying *SEQL2* than VBAT, it looks like we can use both together to tease out the most likely model for individual subjects in the human data that is to be fitted. CBM unfortunately does not provide much additional information here.

One additional measure to check is the VBAT probabilities for each model, as seen in Figure 4.34. For *QL2* and *Dual- α QL*, VBAT is overall quite certain the best fit is *QL2*. For *RandomBias* there is some overlap between itself and *DualUpdateQL* and *HMM*. For *SEQL2* there is not much overlap between the correct model and the overall most fitted, *QL2*. For *HMM* and *DualUpdateQL*, the probabilities are all very close to 1 in the vast majority of cases, although it is not easy to see.

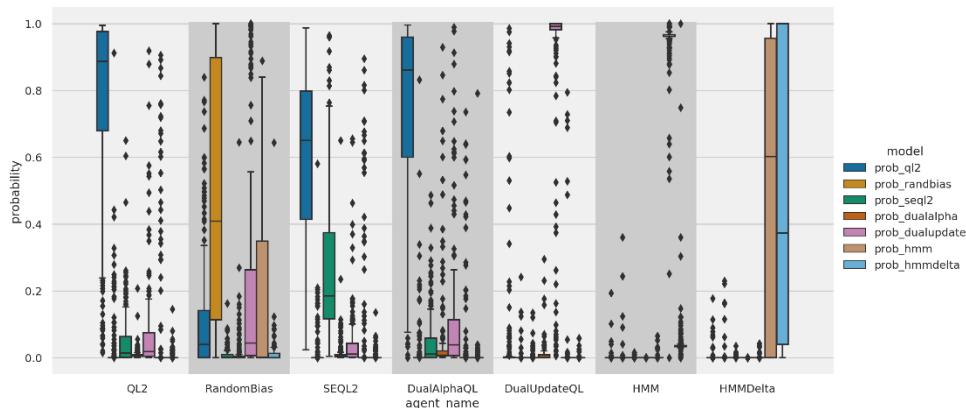


Figure 4.34 Probability (y-axis) for the model (coloured boxes as per legend) being the best fit for the group of simulated subjects of each agent type (x-axis). Shaded areas are simply for readability.

To summarise, if we fit *DualUpdateQL* or *HMM* to a subject, we can be fairly confident that is the strategy used, but the participant could also be using a random approach in the latter case. *Dual- α* is unlikely to be the best fit for any participant (and would thus be a surprise if that was the case for the human data). If we fit *QL2*, then we can be confident it is in the QL-family, at least. Somewhat surprisingly, it

also looks like MLE overall performs very well and in theory we would not really need to use the more advanced Bayesian methods.

4.10.4 FITTING HUMAN DATA IN THE WORTHYBANDIT TASK

With exhaustive investigations into the behaviour of our agents and performance of model selection, we can now fit the human data. Since we found CBM did not provide much useful additional information we exclude this method from the presentation here.

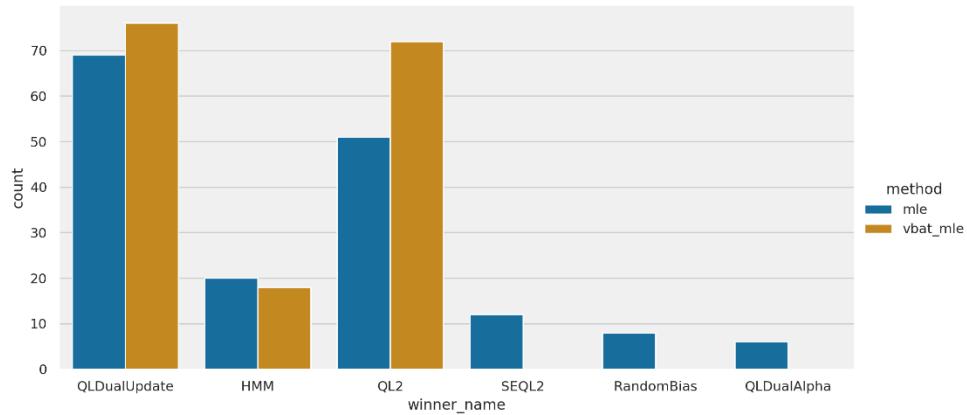


Figure 4.35 Count of model selections (y-axis) for each model (x-axis) for each method (coloured bars as per the legend)

We have plotted the model selection results for the human data in Figure 4.35. Only the MLE method found *SEQL2* (11 subjects), *RandomBias* (8 subjects) and *Dual- α QL* (6 subjects) to be the best fit for any of the subjects. They may look like an insignificant count in comparison, but we saw above that the MLE method is much better at fitting *SEQL2* than VBAT. We also see that VBAT has fitted more *QL2* subjects than MLE, which is also to be expected from the above investigation, as VBAT tended to fit most QL family models as *QL2*.

The most interesting findings here are the two models at the left, *DualUpdateQL* and *HMM*. Both were found to be reliable in model selection performance above. Are there any correlations between the model selections and score for the participants? To investigate this, we first focus on the subset of subjects where the two methods agree, as any conclusions we draw from that subset we can be more confident in. This turns out to be 136 of the 166 subjects.

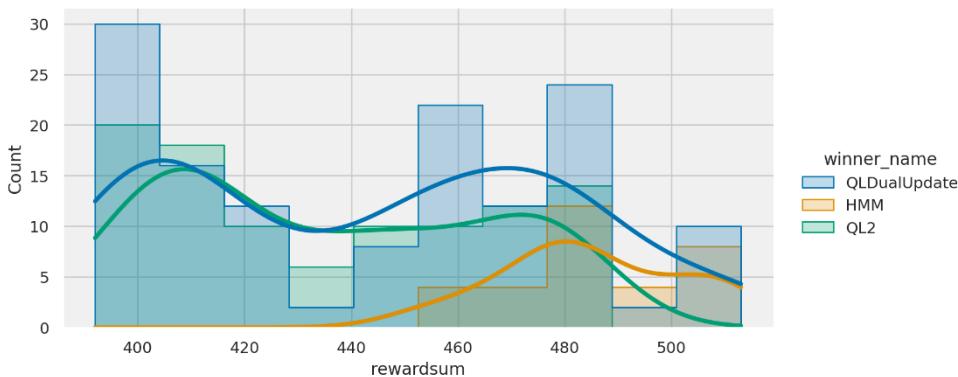


Figure 4.36 Histogram with number of subjects (y-axis) across the total score (x-axis) received in the experiment. Colours indicate model fitted as per the legend, and lines are calculated using kernel density estimate.

In Figure 4.36 we have the money shot. *QL2* and *DualUpdateQL* fit scores across the spectrum but falls off as we reach the very high scores towards the maximum possible. This while *HMM* increases in likelihood to be the best fit above 450, and after 470 or so, it is equally roughly equally likely for the rest of the range of possible scores. Perhaps it is the case that high-scoring individuals do use state inference to a larger degree than non-high scoring individuals?

We should, however, recall that in the agent performance investigations we saw that very few parameter value combinations for *HMM* go below 450 points (Figure 4.28). So, this result may simply be due to the inherent dynamics of the models, and not have anything to do with participants using state inference, *per se*. Additionally, there are several *DualUpdateQL* subjects in the right most bin as well.

Then again, the idea that one model supposedly fits all individuals is one we do not agree with, which has hopefully been clear since earlier in this thesis. It's entirely possible that *DualUpdateQL*, *QL2* and *HMM* are all possible and successful strategies in this task.

4.11 CHAPTER SUMMARY AND DISCUSSION

In this chapter, we further explored the limits and capabilities of the model fitting and selection methods under consideration – MLE, VBAT and CBM. We did so using variations of the two-armed bandit task where reward contingencies change throughout the experiment, so called reversal bandits. This allowed us to introduce the concept of states, both hidden and observable, and investigate how RL models

and HMM models may explain human behaviour in two datasets using different types of reversal bandits.

For the first task, ReversalBandit, we showed that most subjects did not need the concept of states to be successful. Instead, it was sufficient to keep track of action values for both arms simultaneously.

The second task, WorthyBandit, is much more complex and requires careful balance of exploitation and exploration. Here, many subjects found the approach of tracking hidden states more useful than that of simply tracking action values. However, most subjects managed fine with the same strategy used in ReversalBandit.

To summarise the results of both datasets, we show it is indeed likely that individual participants do use different strategies and adapt those to the task at hand.

Regarding the methodology, we found that MLE – supported by VBAT – was the better performer. CBM was at best on par, and at worst did not work at all for some models. This is unfortunate, because the comparison of the methods is left incomplete. More serious is the kind of bugs and issues with long computation times we ran into. The latter was an issue for methods used in the previous chapter as well. What these problems tell us is that – regrettably – authors do not fully test their tools before publishing papers telling the world how useful these methods are.

On that point, we show how important it is to cover as much of the parameter space as possible when testing methods, preferably with thousands of parameter combinations. Additional to this, it is equally important to consider how a single unique parameter value combination can vary across multiple simulation runs due to the inherent randomness of action selection methods such as SoftMax.

To conclude, the best performing method here is the non-Bayesian MLE. Granted, it is efficiently supported with VBAT which is a Bayesian based method. As with so many things in life, we thus show that a balanced combination of both worlds leads to the best result. Even better, this method is the most computationally efficient which is very good indeed for the environment.

5 THE SHAPE SEQUENCE TASK

We have so far seen how RL can often describe the behaviour of humans in simple bandit tasks. Even in these simple and apparently stateless tasks, the concept of states can be useful. It is thus time to introduce tasks with observable stimuli, on which states may be based directly or used to guide inference.

There are multitudes of such tasks, and as we saw with the reversal bandit in the previous chapter, just introducing the simple adjustment of varying what arm or deck is the best allows for greatly increased complexity of the task structure. Thus, adding observable stimuli allow for even more complex ways of constructing a task. The common base structure for typical state-based tasks however is that a stimulus such as a picture of an object is shown, and the participant has two or more action options to pick from. The simplest case would be a so called go-no go task where a picture is shown, and the participant decides if they push a button (go) or not (no-go).

The most common task variant (although we have not counted this formally) is likely to be one where there are a small number of different stimuli, and two response options of which one must be selected. Only one of the options is correct for each stimulus. For example, you may see pictures of different types of food and for each you must learn if I like or dislike it. Another alternative would be that of a bumblebee searching for nectar in a new-found flower patch. Perhaps there are several different kinds of flowers, each with a distinct colour (and most likely shape but let us keep it simple) – the flower stimulus defines the state. Through trial-and-error, flying from flower to flower, the bumblebee learns that overall, purple flowers yield more nectar than flowers of other colours (more formally we can say the purple flowers have the highest state-action value).

Another type of task that is quite different at surface, but the same concept, would be navigating through a house or a labyrinth of rooms. Each room has one or more doors that lead to other rooms, and you must learn the value of each action (passing through the door) in each state (room) to find a terminal reward such as the exit of the labyrinth or where the dog has hidden your slippers. In the latter case, if it is the millionth time the dog has stolen your slippers, you most likely already have a set of state-action values you can

follow for each room and door to get to your goal and hope that, this time, the slippers are not soaking wet from drool.

As set out in the introduction chapter, our aim is to investigate the intricate relationship between states, their representation and task structure. In the descriptions above, the state is easily defined as an image on a screen or a flower in a field but in the real world all observations are high-dimensional and continuous. Humans and other animals need to be able to generalise over states that look different but are functionally the same, while also differentiating between states that look the same but are functionally different. Furthermore, the information to make such generalisations and differentiations is not always directly observable, meaning additional information needs to be inferred.

Therefore, to study how humans discover task structure and make generalisations and differentiations, we came up with a new decision making task that we call the shape sequence task [240].

In this chapter, we will first describe the process of creating the shape sequence task (*Shapetask* for short). The process involved several iterations informed by experiments with human participants. Results from the final version of *Shapetask* show that some humans – but not all – are able to successfully solve the task.

We then investigate RL behaviour in *Shapetask* and show how manipulation of state representations can account for the different types of human behaviour. To improve the biological plausibility of these manipulations, we present models from the literature which have a degree of neurobiological support and that combine RL with complex state representations.

As one of these new models proves infeasible to fit using the methods presented so far, we then introduce a new model fitting method. We characterise model selection performance in *Shapetask* using the new method and can then finally fit the human data and discuss the results.

5.1 CREATING THE SHAPE SEQUENCE TASK

Developing a new task is a process of iteration over many months. We will thus present here first the general idea of the task and how it is supposed to help us investigate the questions we are interested in, as

stated above. Then we describe our task development and testing chronologically, ending up with the final version and variations that will later be the focus of our subsequent investigations.

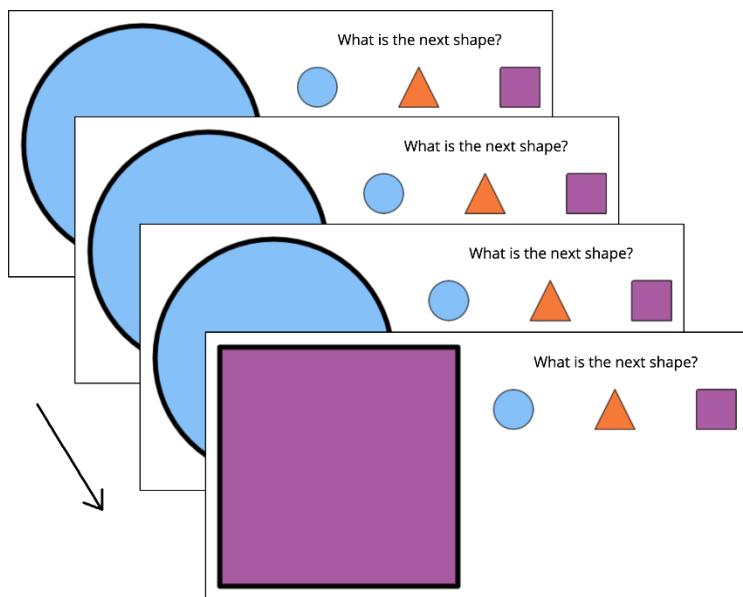


Figure 5.1 The basic structure of the shape sequence task. Each rectangular panel represents a single trial in the task. The participant sees a large shape on the left, and to the right they are asked what they think the next shape will be. In this example, the first shape they see is a blue circle. Then they see two more blue circles, followed by a purple square.

As seen in Figure 5.1, in each trial, participants see a large, coloured shape on the screen, together with three options on the right for what shape they think will appear on the next trial. The possible shapes are here blue circle, orange triangle, and purple square and the response option buttons are small versions of the same shapes. The options are always presented in the same spatial layout and participants indicate their choice by clicking one of the buttons, and the next shape in the sequence appears. Importantly, there is no explicit indication given to the participant if their choice was correct other than them seeing the next shape.

The sequence can be conceptualised by imagining three bags, where each bag has three shapes of the same kind inside. All the shapes in one bag are drawn before shapes from another bag are drawn. Thus, the underlying basic pattern is that each shape will be presented three times in a row. Of course, participants are only told their task is to find the pattern, the idea of bags is only used here in order to understand the task sequence better. There is a shortened

version of the task available online²⁷, to see the task in action as a participant would.

This simple task is not as easy as it looks. Participants may very well think that colour provides different information than the shape (especially if they are used to psychological experiments), which in this case is not true, but it may throw them off. In order to identify the complete pattern in the task, participants must identify that the last shape of the three presented requires a different response than the first two. They must also identify that there is similarity between the last presented shape for all three shapes. In other words, we can theoretically investigate how humans combine generalisation, differentiation and inference to find task structure which, as mentioned above, is exactly our aim. To use the concept of bags, the task can investigate the influence of higher-order hidden state properties – what bag are we currently in – as well as the process of going from states as single trials (one shape) to states as a set of several trials (bag).

The shape sequence task is similar to other tasks in the literature, for example the n-back task [54, 300] which is supposed to measure working memory (WM). It is therefore possible that shape sequence task partly or wholly measures WM and does not necessarily induce RL. But, even if that was the case, participants would still have to use some form of the generalisation and differentiation and inference mentioned above in order to find the task structure and apply working memory to that structure. Additionally – and as mentioned in background – RL is by necessity intertwined with memory²⁸, both episodic memory [293] and working memory [54, 301].

Another similar task – and more related – is the “dimensions” task [294]. The stimuli used differed on three dimensions: shape (square, triangle, circle); colour (green, red, yellow); and pattern (dots, grid, waves). For example, one stimulus could be a circle with red borders and dots inside, another a square with green borders and a grid pattern inside. Participants were shown three different stimuli simultaneously on screen and had to pick one of them. Rewards were

²⁷ <https://gamescapad.es/balltask>

²⁸ And most likely most other brain processes, as the idea that any brain function exists in isolation from others is only useful for humans doing research intended to illuminate such isolated processes

explicit and probabilistic, with one of the stimuli having 75% of reward and the others 25%. Which stimulus was the most rewarding was decided based on only one of the three dimensions (shape, colour, pattern), and the dimension-based contingencies changed every 15–25 trials. Participants thus had to find the relevant dimension among the multiple available.

The dimensions task is very similar to Shapetask both in appearance and aim – studying how humans find structure in the world. The difference lies in that in Shapetask, subjects have to find that sequential position is a feature of interest. This feature is not only across trials, but it also has to be constructed as a feature by the subjects since we do not signal sequential position explicitly.

Common for all varieties of our task, but dissimilar from most other tasks in the RL literature, is that we did not use explicit rewards. In chapter two we noted contradictory findings between model-based and model-free RL. We then suggested that these findings might arise because there is a more general kind of prediction error function delivered by dopaminergic projection systems in the brain. From this it is easy to imagine that a sensory (or stimulus event) prediction error would work as a sort of implicit reward prediction error and can be captured through algorithms similar to those used with explicit rewards and their associated prediction errors.

We have so far described one version of the task, but it is easy to imagine how the task can be varied in many ways. For example, we can vary the number of shapes in each bag or vary the number of shapes. The task could be simplified by removing colours. Different kinds of explicit feedback could be introduced, and so on. We chose to stick to three different shapes and three shapes in each bag. Even with these constraints, the task can be varied in multiple ways, as we will see presently. For all versions, we used jsPsych [149] to create the experiment as a web site.

5.1.1 SCORING THE SHAPETASK

Common for all task versions, we calculate four different scores on a per subject basis. For each trial, we calculate:

1. Correct – if the choice correctly predicts the next stimulus
2. Shift predict – if the choice for the next stimulus is different from the current stimulus

3. Win-stay – if the prediction on the previous trial was correct, and the same prediction is chosen again on the current trial
4. Lose-shift – if the prediction on the previous trial was incorrect, and a different prediction is chosen on the current trial

The first two, Correct and Shift predict, look at the current prediction in relation to the next stimulus. The latter two, Win-stay and Lose-shift, look at the current prediction in relation to that made on the previous trial. This means we can get trial-by-trial measures for each participant.

We can also sum up these scores to get summary scores for each participant. For each trial t , stimuli s and action²⁹ a , and where p is probability, we then get:

$$\begin{aligned} p_{\text{correct}} &= p(s_{t+1} = a_t) \\ p_{\text{shift predict}} &= p(a_t \neq s_t) \\ p_{\text{winstay}} &= p(a_t = a_{t-1} | s_t = a_{t-1}) \\ p_{\text{loseshift}} &= p(a_t \neq a_{t-1} | s_t \neq a_{t-1}) \end{aligned}$$

5.1.2 PILOT VERSION

In the first version of our task, we used only circle shapes differentiated by colour and was thus called “Balltask”. We were also interested in getting an estimate of participants’ uncertainty about their choices and so included a question after each ball prediction how certain the participant felt about their choice, indicated on a five option Likert scale. This Likert scale question showed only after the participant had made a prediction about the next ball colour. In Figure 5.2, we see how this would look for a single trial. On the right-hand side of this figure, we can see the Likert scale going from very unsure to very sure.

²⁹ In keeping with the general descriptions used in the RL literature we will refer to the stimulus predictions made on each trial as actions. Note that the action on trial t is the prediction for the stimulus on trial $t+1$.

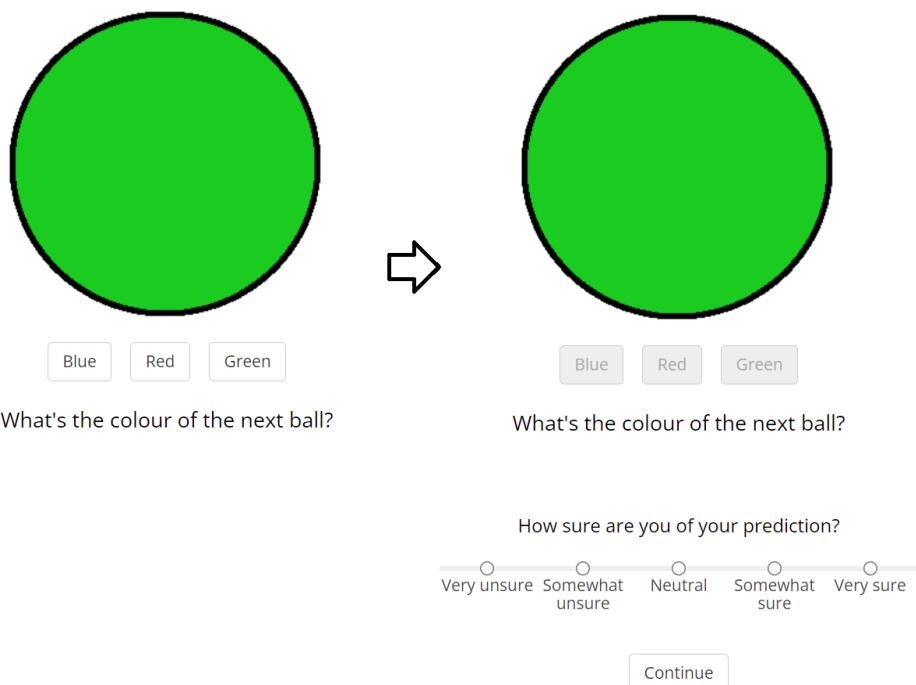


Figure 5.2 Shapetask version 1, a.k.a. Balltask. On each trial, the participant saw a big, coloured circle (green, red or blue) and three text buttons with the possible choices (left side of figure). After clicking one of the buttons, the question and scale showed up below (right side of figure).

The base pattern was, as described above, that there would always be three balls of the same colour in a row. In other words, we had three bags – one for each colour – and in each bag there were three balls. A bag was picked at random, and then all the balls in a bag were used up (i.e., without replacement) and shown one by one to the participant before (metaphorically) putting the balls back into the bag. Then a new bag was picked at random again. This meant that there was a chance that the same colour would be shown 6, 9 or 12 times in a row. Longer sequences were obviously theoretically possible, but they did not occur in the randomly generated sequences used for this task.

In this pilot version, the same sequence was used for each participant. In that sequence, there were a total of 270 trials, consisting of 45 cases of 3 identical balls in a row, 16 cases of 6 in a row, 3 of 9 in a row and 1 case of 12 in a row. We then recruited 27 people via Amazon Mechanical Turk, who were paid approximately £10/h for participating. In order to be reasonably certain that the participants would take the task seriously, we set a condition that participants had to have at least 100 previously approved submissions to Amazon

Mechanical Turk to participate. Additionally, after accepting the task, all participants were given a standard consent form and were free to decline if they so wished. After giving consent, they were given the following instruction:

*Welcome to the experiment!
You will be presented with a series of pictures. Each
picture is a coloured ball, either red, green or blue.
Your task is to predict what colour the next ball will be.
Good luck!*

What we hoped to see in this pilot was that at least some of the participants would spot the pattern, and therefore on every third choice (the last ball of a bag) pick another colour than the one shown on the screen at that time. The second expectation was that we could correlate their uncertainty indication with the choice behaviour.

5.1.2.1 Pilot results

To understand the results, recall that one “bag” consists of three balls of the same colour, and these balls are presented to the subject one at a time. Again, participants are unaware of the notion of the “bags”, but they make it easier to explain the procedure.

Because we are interested in seeing if there is a difference in how subjects respond to the first two balls of a bag, compared to the last (third) ball in a bag, we will show scores averaged across all balls in the same position within the sequence of draws from the hypothetical bag. That is, for example, the proportion of correct predictions across all balls in position one in a bag. This allows us to see if and how participants respond and change their behaviour during the experiment.

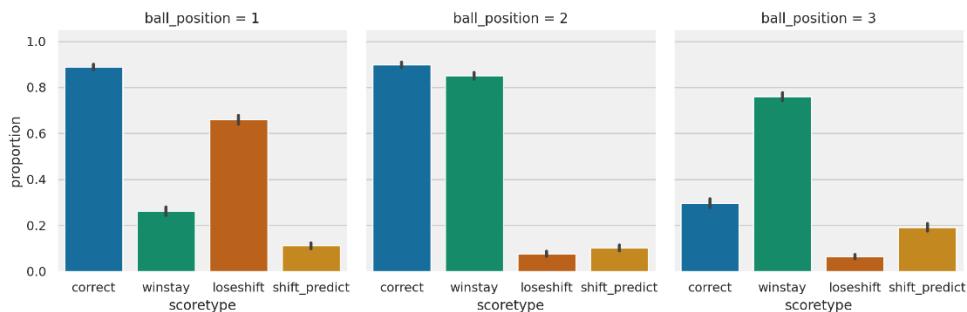


Figure 5.3 Summary scores for all subjects in the pilot version of Shapetask. Proportion of choices on y-axis and different coloured bars on x-axis are the different score types (see text for descriptions). Left: Ball position 1. Middle: Ball position 2. Right: Ball position 3.

In Figure 5.3 we can see that overall, participants have adapted a WinStay-LoseShift (WSLS) behaviour. That is, on positions one and two they predict that the next ball colour will be the same as the one they see (high correct score in left and middle plots in the figure), and they make the same prediction on position 3 (chance level for correct responses in righthand plot in the figure, together with high Win-stay proportion).

However, on average, we can also see a small increase in Shift predict on position 3 compared to positions 1 and 2. If we combine the positions 1 and 2 and compare them to position 3, this difference is significant on the group level (paired t-test, $t(26) = -2.53, p = 0.018$). It may be the case that a few subjects have been able to spot the pattern and this increase in Shift predict on position 3 is driven by these subjects. In Figure 5.4, we have used the last half of the experiment trials³⁰. On the right-hand side, we can see that it is indeed the case that a subset of our participants increases their proportion of shift predict choices in position 3 compared to positions 1 and 2.

³⁰ The pattern is similar when including all trials, but clearer with the last half. The rationalization here is that participants are still learning in the first half.

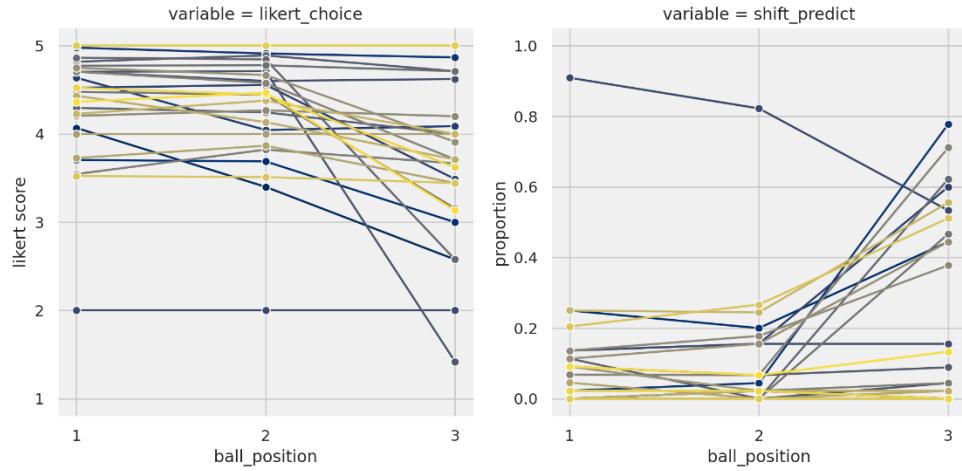


Figure 5.4 Averaged Likert choice (left) and Shift Predict score (right) for each pilot participant (coloured lines) and for each shape position (x-axis). Here only the last half of the experiment trials are included. Left: Likert choice, five-point scale from 1-5 (y-axis), with 1 being very unsure and 5 being very sure. Right: Shift predict proportion (y-axis).

Also in Figure 5.4, on the left-hand side, we have plotted the Likert choices. There we see that some participants have answered the same throughout the experiment (straight horizontal lines), possibly indicating low effort. But we also see some participants going from high certainty in positions 1 and 2 to low confidence for position 3. This is the pattern we expected to see, but is it necessarily the case that the same participants have this pattern for both the Likert choice and the Shift predict score?

For several subjects, yes, that is the case, but for at least one subject it is not, as seen in Figure 5.5. It is possible this participant is confident in their choice because they have spotted the pattern (recall that these data are for the second half of the trials). In which case the Likert choices may not provide much additional information.

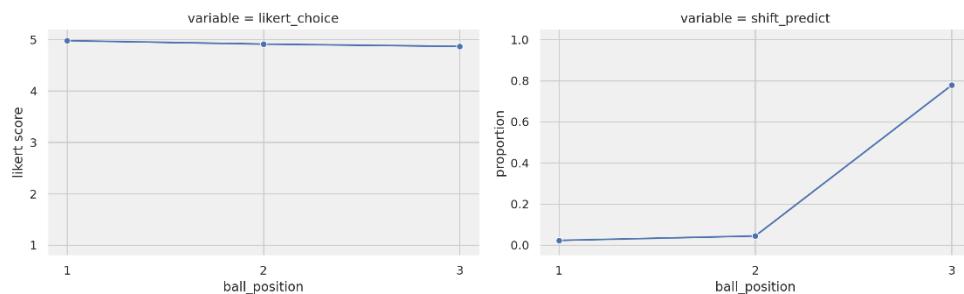


Figure 5.5 Likert choice (left) and proportion of shift predict choices (right) for subject two in the pilot data.

5.1.3 PILOT VERSION 2

The task in its pilot form seemed too difficult for many participants, and we also wanted to know how much – if any – of the results depended on the fact that we were using anonymous online participants. Did some participants fail to find the pattern because they did not understand the task? To find out the latter, we recruited 150 participants from a psychology undergraduate course at Goldsmiths, University of London, who did the task for course credit. We also introduced a new version of the task, to reduce the number of possible repeated sequences.

In the new version of the task, each colour had to appear before the same colour could repeat again. In other words, using the concept of bags containing balls again, we imagine there is a bigger bag containing the three bags of balls. We randomly pick the first bag out of the three (red, green and blue) in the big bag and empty that bag of its balls. Next, we pick one of the remaining two bags and finally there is only the last bag to pick from the large bag. When all bags and their balls have been shown, we refill the small bags and put all three back into the big bag and again start picking a random bag out of the three. This version is therefore called “bag of bags”. In this way, the maximum times a ball colour can repeat is six, which happens if the last bag picked in a big bag is the same colour as the first bag picked from the next big bag, when all small bags are full.

The 150 participants were randomly assigned to receive either version 1 of the experiment, or version 2. Version 1 was identical to the previous pilot, where bags are selected in random order and thus allows for longer sequences of the same colour than six. The only difference in this second pilot is that each subject received a uniquely randomized sequence. Version 2 is the just mentioned “bag of bags” alternative version. For easier identification we refer to version 1 and 2 as “random bags” and “bag of bags”, respectively. Both versions had 270 trials in total. Other than the sequences used in these two versions, the task was otherwise unchanged and so has the same look and functionality as described above for the first pilot.

5.1.3.1 Pilot version 2 results

Due to unfortunate bugs in the experiment code, Likert scale data was not recorded properly and is thus left out of analysis. Additionally, some subjects clicked “agree” on the consent form, closed the browser

window, then came back. These participants could not complete the task, since a cookie was set as soon as the participant agrees, to not allow duplicate entries. This left 106 subjects, of which we excluded additional subjects that left the experiment window for more than twenty seconds at some point during the experiment. This left 73 subjects, of which 34 were in the “random bags” version group and 39 in the “bag of bags” group.

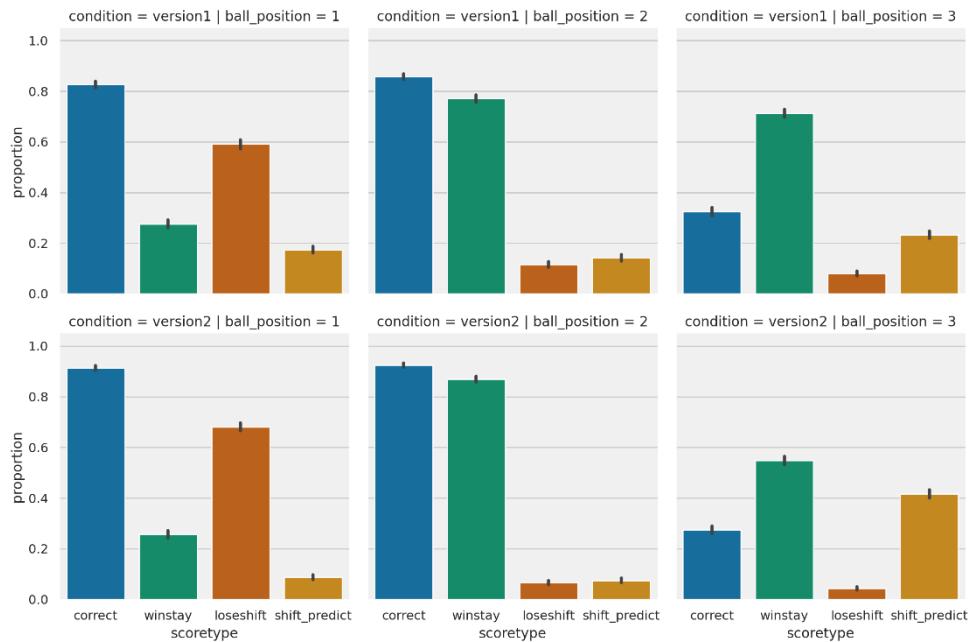


Figure 5.6 Proportions (y-axis) across all subjects and trials for each score type (x-axis), separated by experiment version (rows) and ball position (columns). Top: version1/random bags. Bottom: version2/bag of bags.

In Figure 5.6 we have plotted summary scores averaged across all the participants but separated by task version (rows) and ball position (columns). Overall, the results are similar to the first pilot, but looking at the rightmost column – shape position 3 – we see that those participants doing version 2, bag of bags, have much higher shift predict proportion than the group doing random bags. This is confirmed by a mixed between-within 2-way ANOVA which reveals a significant interaction between position (position 3 vs other) and version [$F(1, 71) = 29.2, p < 0.001$].

We can further investigate this pattern by looking at the line plots of Figure 5.7. There we can more clearly see this stronger pattern of higher shift predict for position 3, which for some subjects in the bag of bags version is very strong. This effect was significant in the group performing the bag of bags version: the shift predict probability for

shape position one and two combined was lower compared to position three ($t(38) = -8.4, p < 0.001$). Even for the random bags group, there was a significant effect for shape position one and two combined compared to position three ($t(33) = -3.2, p = 0.003$). However, we should keep in mind that since we do not have a static sequence for the random (version 1) version, the difficulty can vary between individuals as a function of their specific random task sequence.

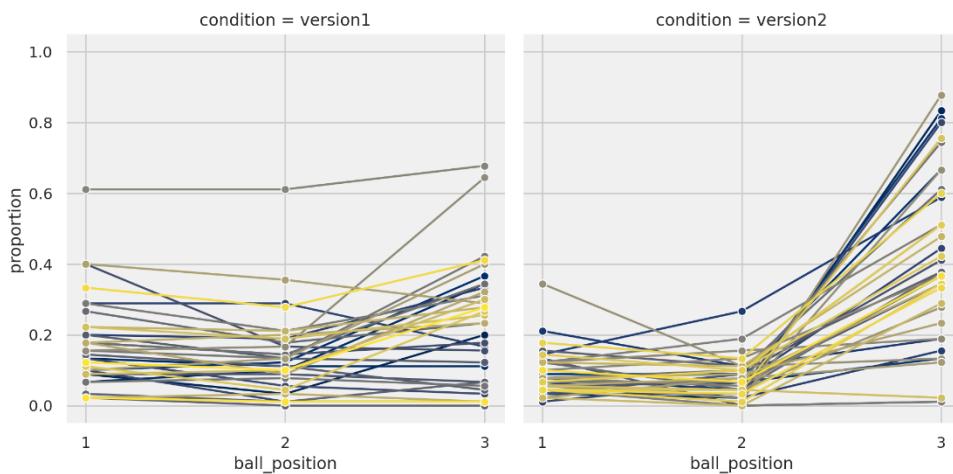


Figure 5.7 Proportion of choices being “shift predict” (y-axis) for each ball position (x-axis). Coloured lines separate each subject. Left: Version1/random bags. Right: Version2/bag of bags.

The pattern seen in the previous two figures arises already within the first 99 trials, something not shown here but can be confirmed in the code repository. We bore this observation in mind when developing the task further below. These results are overall promising, as it seems we have succeeded in making the task simpler.

One could perhaps object here that we are constructing the task in order to get to the results we would like to see, biasing the results. However, there is a difference between biasing a task to get the results you would like and balancing a task’s difficulty so that you can investigate the issue at hand. We believe it’s the latter that is being done here.

It is unfortunate the Likert scale results did not register properly, but it is not critical. As we saw in the previous pilot, the Likert scale was not adding much information. Additionally, thanks to informal feedback given by some participants in person, we realised the Likert scale makes the task very tedious, and feedback (the next stimulus

appearing) is delayed. The main feedback was that the task was generally boring and too long. Subjects took between 10-20 minutes to complete this task, and it is understandable it is experienced as boring since there is not much variety in what is happening.

5.1.4 BALLTASK BECOMES SHAPETASK

Based on the results and feedback from the two pilot tests, we made adjustments to the task, and added an additional task version. The main change was that the Likert scale asking about the participant's (un)certainty was removed. As explained above, it disrupted the flow of the experiment and as seen in Figure 5.5 did not necessarily provide useful additional information. It was thus decided that the Likert scale did not provide any real value.

The adjustments were mainly of a look-and-feel nature, where instead of coloured balls we differentiated the "bag contents" also by shape: circle, triangle and square. Each shape also has its own distinct colourblind-safe colour with the circle being blue, the triangle orange and the square purple. This may slightly increase the task difficulty as noted earlier, because participants could perhaps believe there was a difference in importance between the shapes and colours, when in fact there was not. The task appearance in Figure 5.1, presented earlier, is representative of this new look.

We also added animations and delays. When a new shape appeared on the screen, it bounced up and down before settling. The buttons (actions) were now the shapes themselves instead of text buttons, and these buttons did not appear until 1 second after the main shape had appeared. These additions were made for two main reasons; one to try and guide the participant's focus in a flow from stimulus to action option and the second in order to avoid participants clicking buttons without thought.

The two previous task versions, random bags and bag of bags remain but the number of trials was reduced to 99. The specific number 99 instead of, say, 100 is because we want a full set of bags of bags, and since each bag of bags contain $3^3=9$ shapes, we need to increase trial count in steps of 9.

An additional task version was introduced here, “bag of bags no repeat”³¹, abbreviated *BOB-NR*. This version works the same as the bag of bags (*BOB*) version, with the exception that the first small selected from the large bag cannot contain the same shape as the last small bag drawn from the previous bag of bags. In other words, in this version there will only ever be a maximum of three shapes in a row, never six shapes in a row which can happen occasionally in the bag of bags version.

Finally, we simplified the instruction text (after consent form but before the first shape was shown), putting the main instruction in bold:

*Can you spot the pattern?
Your task is to predict the next shape.
Good luck!*

We also included a text box at the end of the task, where the participant could freely type a response to what was asked:

*Did you notice any pattern to the sequence of shapes
which allowed you to predict the next one accurately?*

5.1.4.1 Shapetask results

For testing our revamped task and the three versions, we recruited participants online through Prolific (www.prolific.co). As mentioned in chapter three, Prolific is an online platform for academic research. It has been shown to provide good data quality and diverse participants [199]. Demographic information is available for all subjects, without us (the experimenters) having to ask for it. The platform allows to filter participants on, for example, fluency in English.

Recruitment was done in two phases. We first recruited 10-15 subjects for each version, partly to test our website and database setup, partly to test the Prolific platform. Confident in our setup and the platform, we then recruited an additional 25-30 subjects for each version. Participants were from across the world, with a few from

³¹ Note this version was called “bag of bags without repeat” in [240], but we decided that name in its shortened form “bob-wr” is too easily confused with “bag of bags with repeat”.

every continent (except Antarctica) and ages ranging from 18 to 74, with a mean of 28.7 (SD 9.8).

In these results, we excluded participants based on two criteria. If they had an average response time that was less than one second, and/or if they left the experiment window for more than 30 seconds in total during the task. This left 32 subjects in the bag of bags no repeat version (bob-nr), 39 subjects in the bag of bags version (bob) and 39 subjects in the random version (random) for a total of 110 subjects.

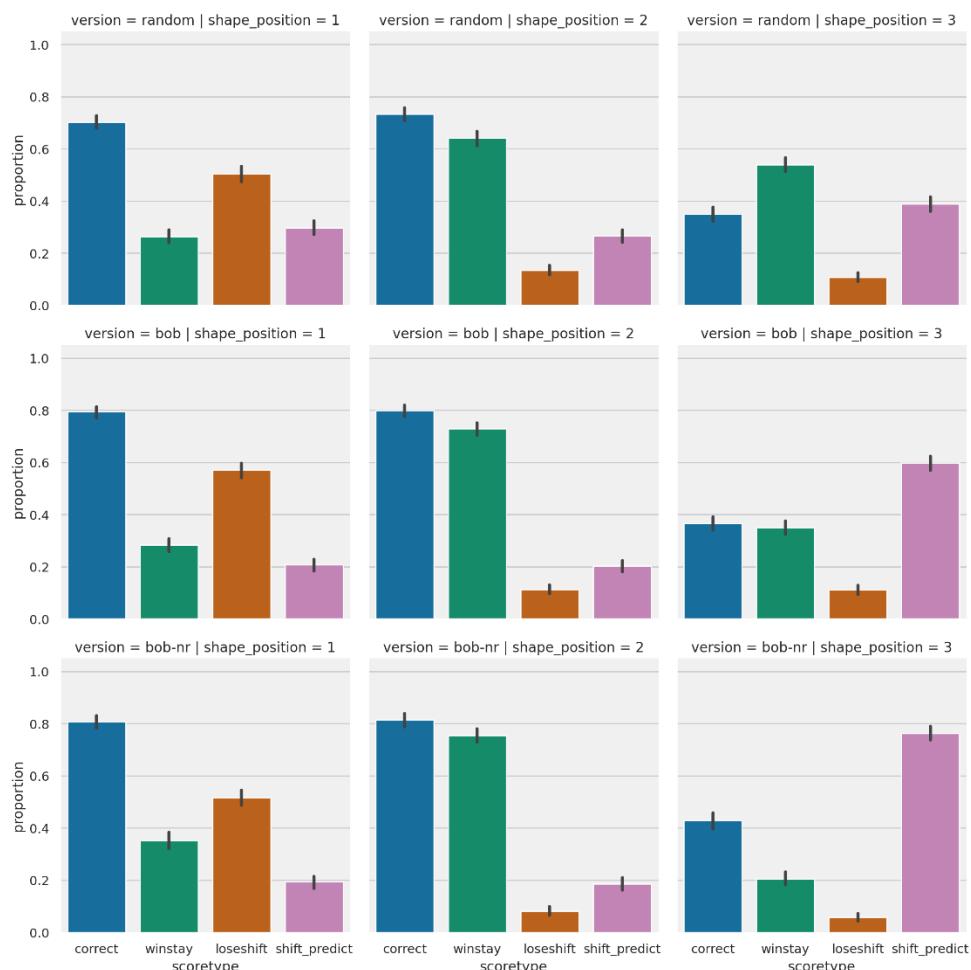


Figure 5.8 Proportion of choices (y-axis) for each score type (x-axis) averaged across all subjects and trials, separated by shape position (columns) and task version (rows).

In Figure 5.8 we have the summarised results for all subjects and task versions. If we start by looking at random (first row) and bob (middle row), we can see that compared to the previous pilot, the proportion of shift predict responses for shape position 3 is at least 0.2 more for the BOB condition. This is most likely thanks to our updated

and more streamlined version of the task making it easier for participants to understand the task and focus on the predictions without being distracted by the Likert scale answers. It could, of course, also be due to random variation, which is especially true for the random version, since difficulty there can vary as sequences may contain different amounts of six, nine or twelve of the same shapes in a row.

Furthermore, we can see on the bottom row of Figure 5.8 that in the *BOB-NR* version of the task, shift predict scores are now almost 0.8 across all subjects. This means most of the subjects have found the pattern, which to a lesser extent is the case in the *BOB* version (where there is an average of 0.6 shift predict responses in position three). In fact, it may be the case that the *BOB* version has advantages if the goal is to investigate potential differences in state representations for subjects that find or do not find the pattern.

Though the patterns just described in Figure 5.8 are quite clear, we can confirm there is indeed a significant interaction between task version and shift predict scores (positions one and two combined compared to position three). A mixed between-within two-way ANOVA gives a significant interaction effect $F(2, 107) = 26.2, p < 0.001$. For the individual versions, we also compare the group levels means for shift predict in positions one and two combined compared with position three. This difference is significant for all three versions (*random*: $t(38) = -3.6, p = 0.001$, *BOB*: $t(38) = -8.2, p < 0.001$, *BOB-NR*: $t(31) = -9.8, p < 0.001$).

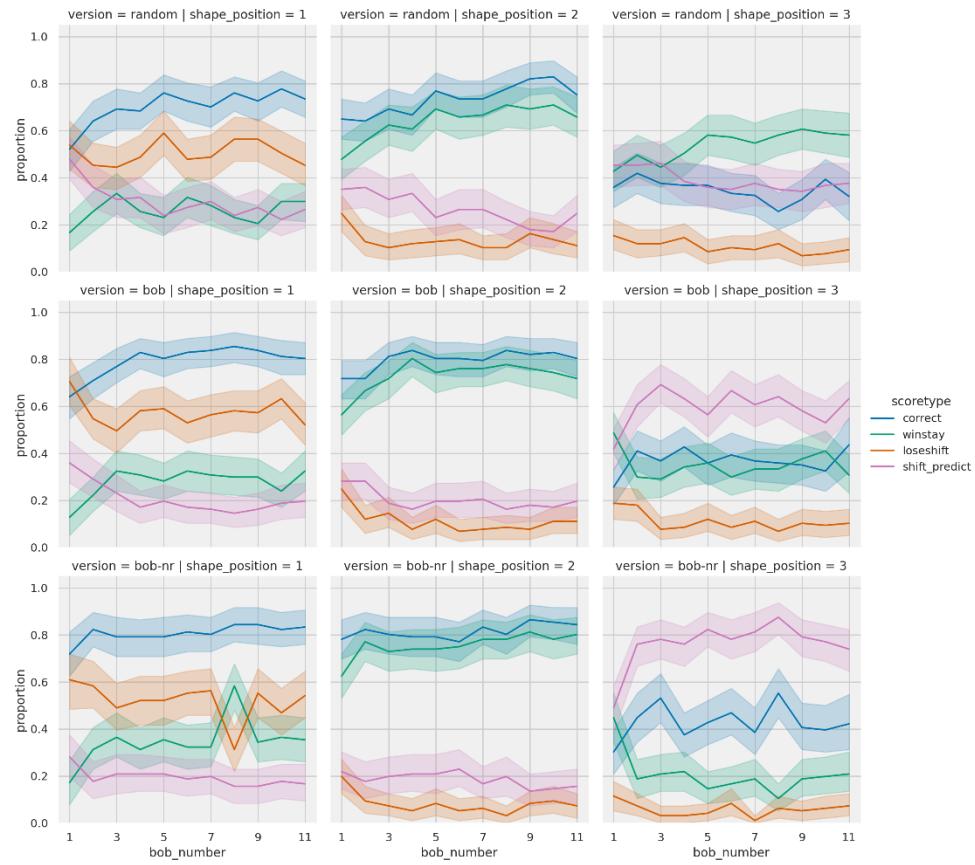


Figure 5.9 Proportion of choices (y-axis) for each score type (coloured lines as per legend) averaged across all subjects, separated by task version (rows) and shape position (columns). On x-axis we have task trials grouped as bag-of-bags, that is nine stimuli/trials per tick on x-axis.

Another way to look at the data is by groups of trials, in order to see when behavioural change was occurring across the experiment task. This is shown in Figure 5.9. Note that instead of trial number on x-axis we have grouped trials via a bag of bags (bob) number, meaning each tick on x-axis is averaged for a group of nine consecutive trials. These groups make the plots less busy and thus easier to read, while trends remain visible. We can see some rough patterns here, such as in the random version (top row) and shape position two (middle column), that correctness and shift predict continuously increase and decrease, respectively, throughout the experiment. In the other two versions, subjects catch on fairly quickly. We also see there is perhaps also a difference in shift predict for position 3, between bob and bob-nr, in that the latter version is around its height already at the second big bag (18 trials), while in the former – bob version – subjects require another big bag (27 trials) before the same can be said.

Thanks to the included free text response at the end of the experiment, we could in most cases confirm directly with the participant that they had found the underlying pattern in the sequence and understood the task. But not all participants did, and there is some variation in the results, especially in the random version.

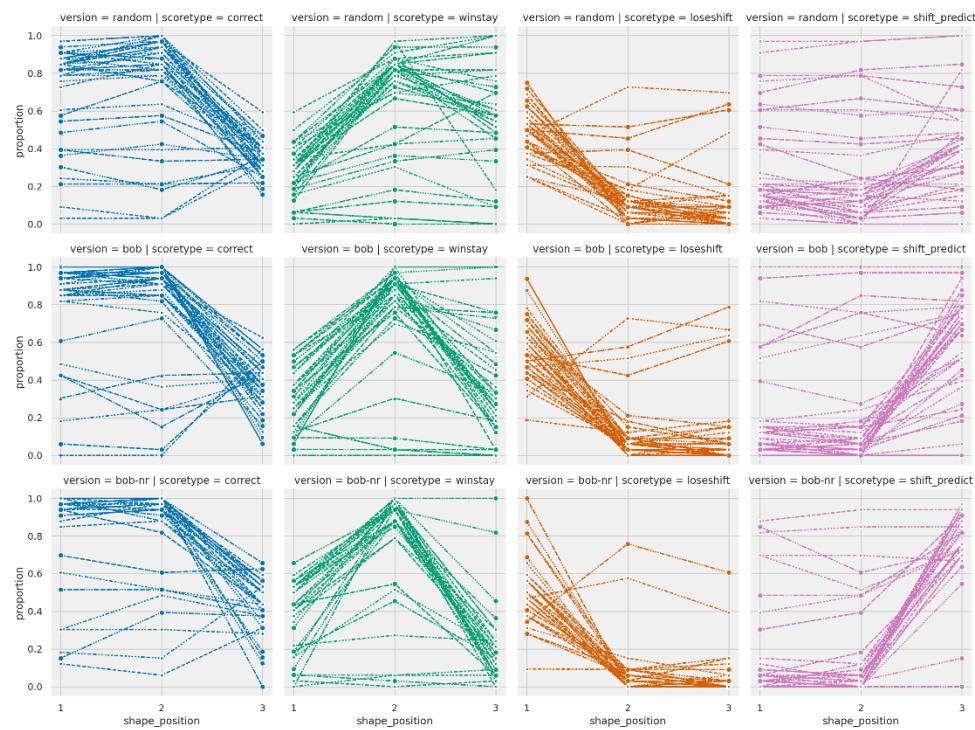


Figure 5.10 Proportion of choices (y-axes) for each score type (columns) and each individual subject (separate lines) across the three shape positions in each bag (x-axes). From left to right: Correct, Win-stay, Lose-shift, Shift Predict. Top to bottom: Random version, BOB version, BOB-NR version. Colours are here used as in Figure 5.9 for ease of comparison.

In Figure 5.10 we have plotted all subjects individually, for each score type averaged across the experiment for each subject. We can there see, especially for shift predict (right most column, pink colour) that a few subjects seem to always think there will be a shift (horizontal-looking lines towards the top of the plots). We also see for win-stay (second column from the left, green colour) that for the *BOB* and *BOB-NR* tasks, many more subjects have a downward slope from position 2 to position 3, compared with the random task version where more subjects stay with their choice for the third position choice (horizontal line or less slope from position 2 to 3).

To summarise, as we progressively reduce the randomness in the position of the shift (from 3, 6, 9 or 12 trials in the random version) to

mostly 3 and some 6 trial sequences (*BOB* version) to always after 3 trials (*BOB-NR*), subjects were able to shift their prediction on position three with increasing frequency. But not all subjects do this even in the *BOB-NR* task, at least not in the number of trials allowed.

5.1.5 DEFINING BEHAVIOURAL GROUPS IN SHAPETASK

Before looking into how RL algorithms perform, we would like to find a useful definition of whether the task is solved or not. From the figures above, especially Figure 5.10, we can see that “winners” have a good combination of high correct score in positions one and two, high win-stay score in position two and high shift predict in position three. Similarly, we have another subset of subjects that seem to fall into WSLS behaviour if they cannot spot the pattern, something especially noticeable in the more difficult random version. These subjects generally have high correct score in positions one and two – like the winners – but instead have high win-stay score in positions two and three. We thus specify these groups using the following cut-offs:

$$\text{winner} = \begin{cases} p_{\text{correct}} > 0.5 \text{ in position 1 and 2} \\ p_{\text{shift predict}} > 0.5 \text{ in position 3} \end{cases}$$

$$\text{wsls} = \begin{cases} p_{\text{correct}} > 0.5 \text{ in position 1 and 2} \\ p_{\text{winstay}} > 0.5 \text{ in position 2 and 3} \end{cases}$$

Subjects not matching these groups will be set as “other”. The best representatives of each group will have much higher scores than the above, but keeping limits lower hopefully allows slower learners to be included.

Note that these conditions are somewhat arbitrary, but they are relatively easy and can be used as a general guide both for human and algorithmic behaviour. It would have been preferable with a more straightforward way of classifying behaviour, but due to the inherent complexity of Shapetask we do need to look at patterns across the shape positions.

To test and demonstrate that these conditions give us decent groups, we plot the averages of these groups separated by score type for each shape position as in Figure 5.11.

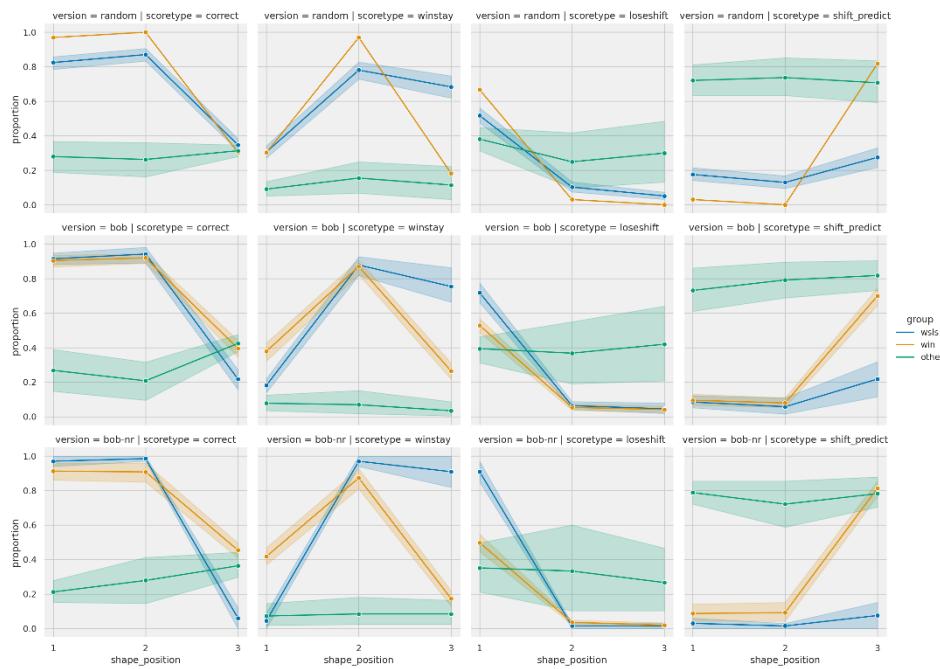


Figure 5.11 Shapetask groups. Proportion of choices (y-axes) for each score type (columns) across the three shape positions within a bag (x-axis). Groups as per legend. Top row: Random task. Middle row: BOB task. Bottom row: BOB-NR task. Left to right: Correct, Win-stay, Lose-shift, Shift Predict. Shaded areas indicate 95% confidence interval.

In Figure 5.11, we can see how across the three versions, these groupings provide similar shapes. This is to be expected since we have used the same conditions to group participants in each task. For the random version, there is only one subject in the win group (which is why there is no variance shading). This is not necessarily entirely due to differences in subject capability but could also be (partly) due to a consequence of randomness. In the random version of the task, some participants get a higher proportion of short sequences (3 or 6 in a row) like in *BOB* and *BOB-NR*. Other participants get a higher number of very long sequences of the same shape (9 or more in a row). If there are many bags of the same shape in a row, then WSLS behaviour becomes the overall best strategy. Hence these groups as they are presented here are not wholly applicable to the random task version, as will be further discussed below.

To get a better sense of how these groups relate to individual performance, we have plotted individual subjects in Figure 5.12. There are a few cases that are either miscategorised or have been slow to learn, but overall, these groups look to work well. We could potentially adjust our criteria to get more clearly defined groups, but how would we define these stricter conditions? Should we add that

winners must also have low scores for shift predict in positions one and two? What about lose-shift scores?

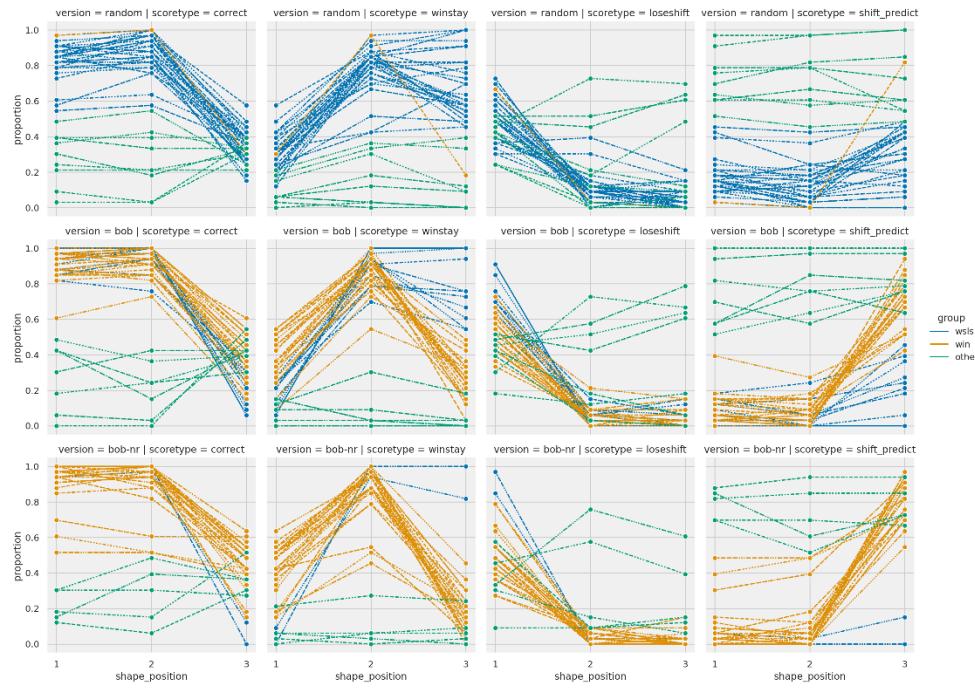


Figure 5.12 Shapetask groups for individual subjects. Proportion of choices (y-axes) for each score type (columns) across the three shape positions within a bag (x-axis). Each line represents a subject and colours represent groups as in Figure 5.11. Top: Random task. Middle: BOB task. Bottom: BOB-NR task. Left to right: Correct, Win-stay, Lose-shift, Shift Predict. Shaded areas indicate 95% confidence interval.

These conditional groupings are indeed somewhat arbitrary, but they work well enough to use as a guidance for our investigations of the algorithms' behaviour. There will be cases that fall outside these strict groupings that we may manually classify as WSLS/winner when looking at that case specifically.

To summarise our grouping results, for the *random* version there are 29 WSLS subjects, 1 win and 9 other. *BOB* has 10 WSLS, 22 win and 7 other, and *BOB-NR* has 25 win, 2 WSLS and 5 other. These changing proportions nicely match the increasing tendency (from *random* to *BOB* to *BOB-NR*) to learn the task better via appreciating the tendency to shift in position three.

We cannot say yet which algorithms will be able to handle the task, or if there are algorithms that can explain both winners and WSLS behaviour (but using different parameter values), for example. In the latter case, during model selection, we could find that the same model fits both subjects we believe understood the task and found the

pattern as well as subjects that did not understand the task and/or did not find the pattern.

5.1.6 SUMMARY OF SHAPETASK RESULTS

What is impressive here is the fact some humans can spot the pattern within the limit of 99 trials, and often much quicker. Some are even able to do so when there is randomness in when the shift will occur (as in the *random* and *BOB* conditions). How is it that humans are capable of finding task structure so quickly and efficiently? That is the question the rest of this chapter will focus on.

From these investigations, it looks like we have succeeded in creating a task with its difficulty so that at least around half of the subjects spot the pattern. The results are clear enough that we can confidently say the task works as intended, and by using the human results as a guide, we can now look at algorithmic behaviour in the same task to find potential candidates for explaining the human behaviour.

We should note here that due to inherent randomness in the random version of the task, it is difficult to compare behaviour between participants. We could use a static sequence like in the first pilot, but then we would have to do further tests to find a sequence that is easy enough that some participants can find the pattern. Since we have used alternative versions in *BOB* and *BOB-NR* that are quasi-random (random but with some constraints), the obvious route is to use one of these versions instead.

For subsequent investigations, it would streamline our process and presentation if we could focus on one of the two remaining versions. *BOB-NR* would be the more straight-forward option, as most of the humans successfully found the task structure in this task version. However, there is also a case to be made for *BOB*, because this version is more balanced in the proportions of winners vs the rest (as defined above). The majority of subjects were able to find the task structure, but also a fair many cases of WSLS and other. The reason behind the amount of WSLS cases in *BOB* compared to *BOB-NR* is probably due to some subjects being thrown off by the occasional six-shapes-in-a-

row but that was not exclusively the case³². Furthermore, since we have stressed in previous chapters the importance of individual behaviour in RL tasks, it seems appropriate to select the task version with a more balanced distribution of behaviours across the whole group. Going forward, we will therefore focus on *BOB* unless otherwise stated.

5.2 RL BEHAVIOUR IN SHAPETASK

Starting off our investigations of RL behaviour in Shapetask is the standard QL algorithm. Previously we have used its two-parameter version with learning rate α and SoftMax temperature β , where the latter is not technically part of the QL algorithm itself but used for action selection. Here we will extend QL to its full temporal difference (TD) form, with a future discount parameter $0 < \gamma < 1$:

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha[r_t + \gamma \max_a Q(s_{t+1}, a) - Q_t(s_t, a_t)] \quad 5.1$$

where $\max_a Q(s_{t+1}, a)$ means the maximum Q-value in the next state. This may be somewhat unintuitive, to look into the future before we get there. The straightforward reason is that Q-values represent the approximate value of state-action pairs, so in order to take the future into account, we use our current knowledge of that future, which are the Q-values. The more subtle answer is that the value update happens during a transition to the next state. We have picked an action in our current state and observe its consequences and update our knowledge based on this information. For that reason, we could also write the above equation with r_{t+1} , depending on if we conceptualise the task as the reward being received based on the action taken (r_t), or if the reward is what we find in the next state (r_{t+1}). Regardless of these nuances, we call this TD form of QL simply *QL3*.

Many common RL tasks in the literature are so called one-step reward tasks, where a stimulus is presented, an action is selected and then a reward is received. Hence, it is uncommon (and unnecessary)

³² Counting the number of six-in-a-rows for each group, the maximum was five six-in-a-rows and the minimum one. Four win-subjects and three WSLS-subjects encountered this maximum. Six winners and one WSLS subjects encountered the minimum. See bar plot in code repository.

to see anything more than QL_2 being used in the literature. In more complex tasks, there may be multiple stimulus-action (state-action) steps before a reward is received, and these tasks is where QL_3 is useful. In the computer science literature, these more complex tasks are commonly called tasks with sparse rewards [258].

To demonstrate, consider a simple 3×3 maze. Each square in this maze is a state, and thus has a value attached to it. Our agent can move horizontally or vertically and picks their action by getting the values of squares next to it. Note this means we are using state-values to represent the maze instead of using state-action-values [258], but the state-action-values are constructed each step by using the state values from neighbouring squares, enabling us to use Equation 5.1.

If an action is picked that moves outside the maze, the agent remains in the same square. The agent will start in the top left corner and the reward is in the bottom right, as seen in Figure 5.13.

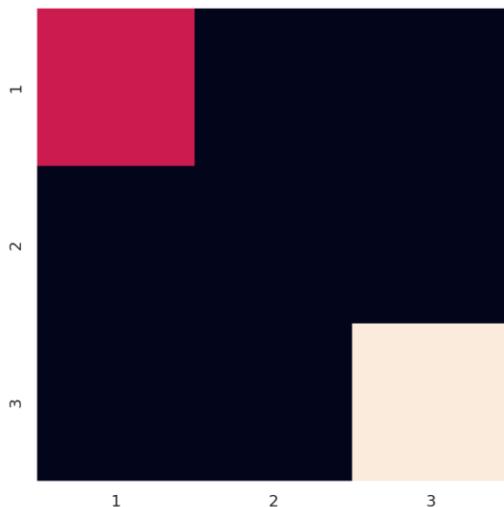


Figure 5.13 Simple maze task. The agent is the reddish square in the top left and the reward is placed in the bottom right, marked by creamy white.

If we simulate both QL_2 and QL_3 in this simple task, we can see the difference between their mechanisms. In the following simulations, we define steps as one square to square transition and one episode is the total sum of steps used to reach the goal square with the reward. The maximum steps allowed in total were set to 1000, resetting the agent to start position at the end of each episode. Both agents used $\alpha = 0.3$, $\beta = 5$ and QL_3 also used $\gamma = 0.9$. The results are shown in Figure 5.14. What this shows is how the QL_2 agent can update only the value for the state where the reward is found. Which means it will

forever be doomed to blindly take actions until it arrives in a square next to the reward. It will learn something only if it moves on to the reward square. But the QL_3 agent can successively update all state values back to the starting position and through experience create a lit path.

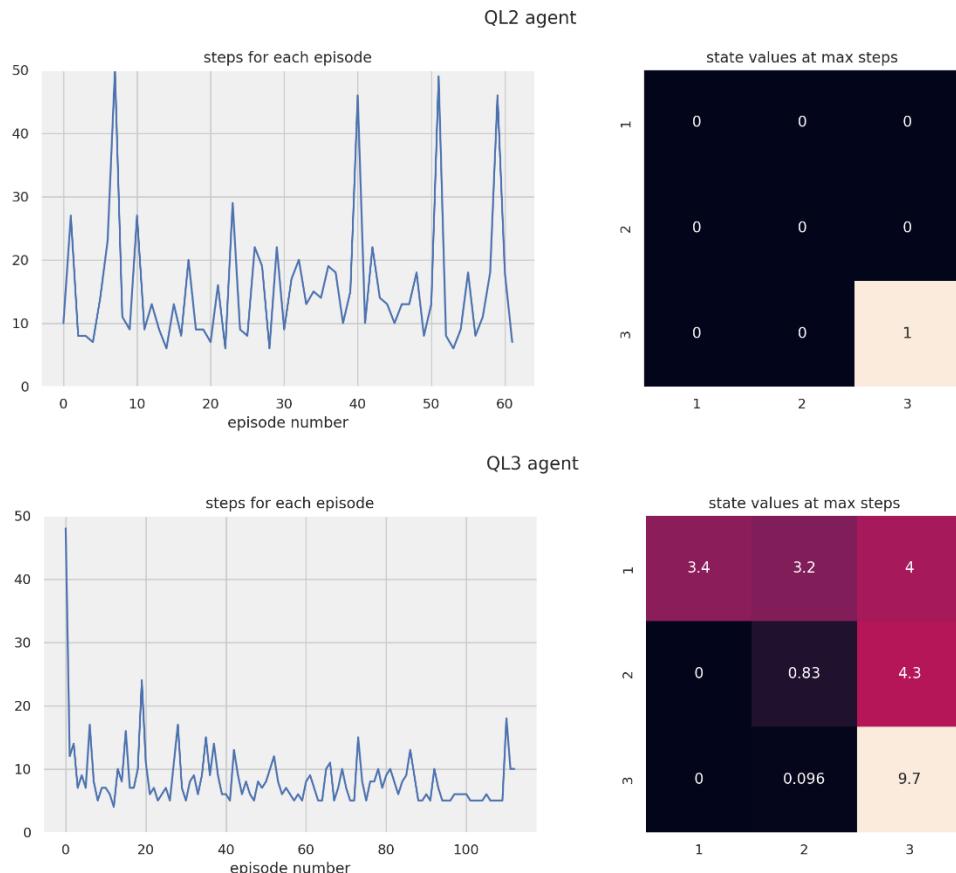


Figure 5.14 Comparison of QL_2 (top) and QL_3 (bottom) agents. The line plots show the number of steps needed for each episode and the heatmaps on the right show the final state values when experiment is over. Note these state values are the agents' internal state values representing the maze, not the maze itself.

To illustrate clearly what happens for the QL_3 agent, see Figure 5.15. The first time it encounters the reward, the value for that state is updated (left part of figure) and the episode ends. The next episode, when the agent arrives at a square next to the reward it will now, thanks to the future discount, update the value of that state, before continuing onto the state where the reward is³³.

³³ Unless it picks an exploratory action and goes up or left, but in this example it went right.

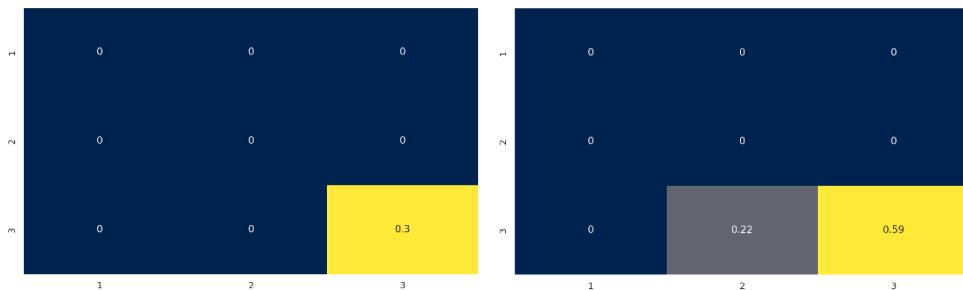


Figure 5.15 Q-Learning with future discount in simple maze example. The state values after first (left) and second (right) episodes compared.

Coming back to Shapetask, we can see it as a mix of one-step rewards and sparse rewards, as there are both immediate rewards (we have suggested that correctly predicting next shape equates to a reward) and longer-term rewards (correctly identifying the task structure). It is not straight forward to calculate if the addition of future discounted rewards would help in Shapetask, because our agents' behaviour is a combination of the three parameter values as well as inherent randomness in the action selections. But since there is structure across multiple steps, it's reasonable to use QL_3 here, and simulate behaviour in order to know what to expect.

5.2.1 QL_3 BEHAVIOUR IN SHAPETASK

Having QL_3 play Shapetask, we define the state as the stimulus being shown to the participant i.e., the shape³⁴. So, there are three possible states in the task. As explained above, we use the *BOB* version of the task with 99 trials. We then generate 1000 random parameter combinations and run 100 simulations (each of these also used the same task sequence) for each combination. The results presented below is the average over the 100 simulations, called a simulation set, or simset for short. We use these averages to overcome the inherent randomness of each individual run. Parameter values are generated as:

$$\alpha \sim U(0, 1), \quad \beta \sim U(0, 20), \quad \gamma \sim U(0, 1)$$

³⁴ We could also have used colour as the two are perfectly correlated stimulus features across this task. Human subjects may focus on colour/shape or the combination. We use shape as a shorthand for this more complex set of possibilities.

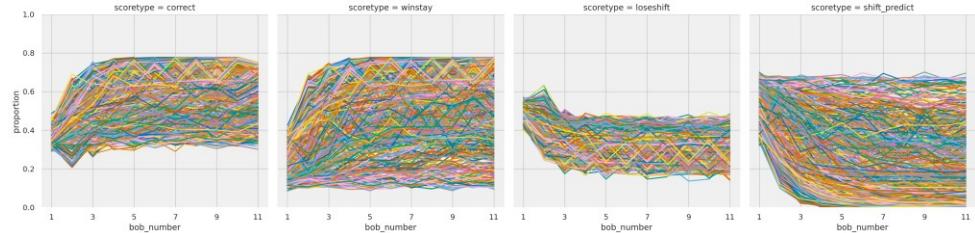


Figure 5.16 Overview of behaviour for all simsets. Proportion of choices on y-axis for each bag-of-bags (nine trials) on x-axis. Each line is one simset, and columns are score types. From left to right: Correct, Win-stay, Lose-shift, Shift-predict.

In Figure 5.16 we have plotted the behaviour of each simset during the experiment task. Despite its busy appearance, the figure gives us an overview and a sense of the behavioural limits of QL_3 . It seems behaviour stabilises around bob number five, and from the trends in shift predict (right-most column) no simset appears to increase in shift predict as the task proceeds. No simsets reach higher than 80% correct or win-stay. To get a better sense of these behaviours for each shape position we have plotted the simsets again in Figure 5.17 but now using shape position on x-axis. As one could suspect from the previous figure, we can now say with more certainty that there are simsets which will exhibit WSLS behaviour (high win-stay in positions two and three, second column from the left), but few – if any – simsets that will solve the task (no lines with an upward trend for shift predict between positions two and three in right-most column).

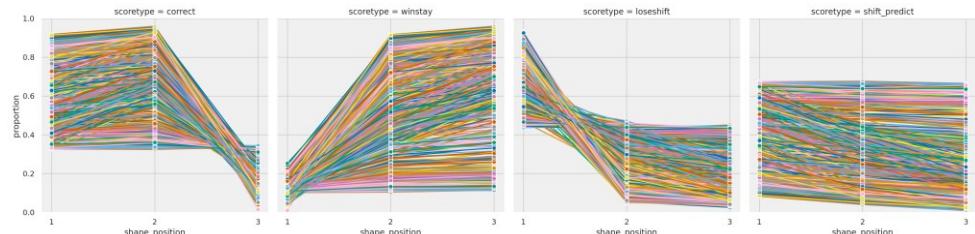


Figure 5.17 Average score (y-axis) for each simset (coloured lines) and shape position (x-axis), separated by score type (columns). From left to right: Correct, Win-stay, Lose-shift, Shift-predict.

To confirm our suspicions about the possible behaviours of QL_3 in Shapetask, we classify each simset according to the criteria in Section 5.1.5.

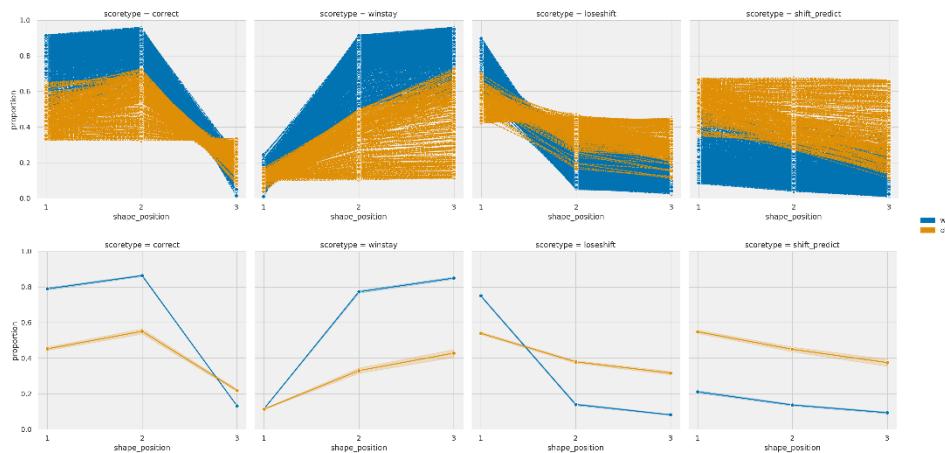


Figure 5.18 QL3 behaviour groups as per legend. Proportion of choices (y-axis) for each shape position (x-axis) and score type (columns). Top: All simsets plotted separately. Bottom: Average of all simsets for each group. Left to right: Correct, Win-stay, Lose-shift, Shift-predict.

The next step is looking at parameter values, and how they correlate with behaviour. This gets tricky with increasing number of parameters for the algorithm together with the complexity of Shapetask scoring, with multiple score types and the need for looking at shape positions. We could try to do something similar to Figure 3.4, but that would now require multiple rows and columns, combined with colour, style and size of the plot markers. It is simply not feasible. Luckily, we can use our groups to get a decent overview of how parameter values relate to performance. In Figure 5.19 each parameter value is plotted against itself (diagonal distribution plots) and against the other parameters (row, column combination). The overall patterns we see there is that for WSLS behaviour to occur, α should not be at the extreme ends of its range and β should be at least around >5 and above. For γ it's difficult to see a particular trend as both groups cover most of its parameter space.

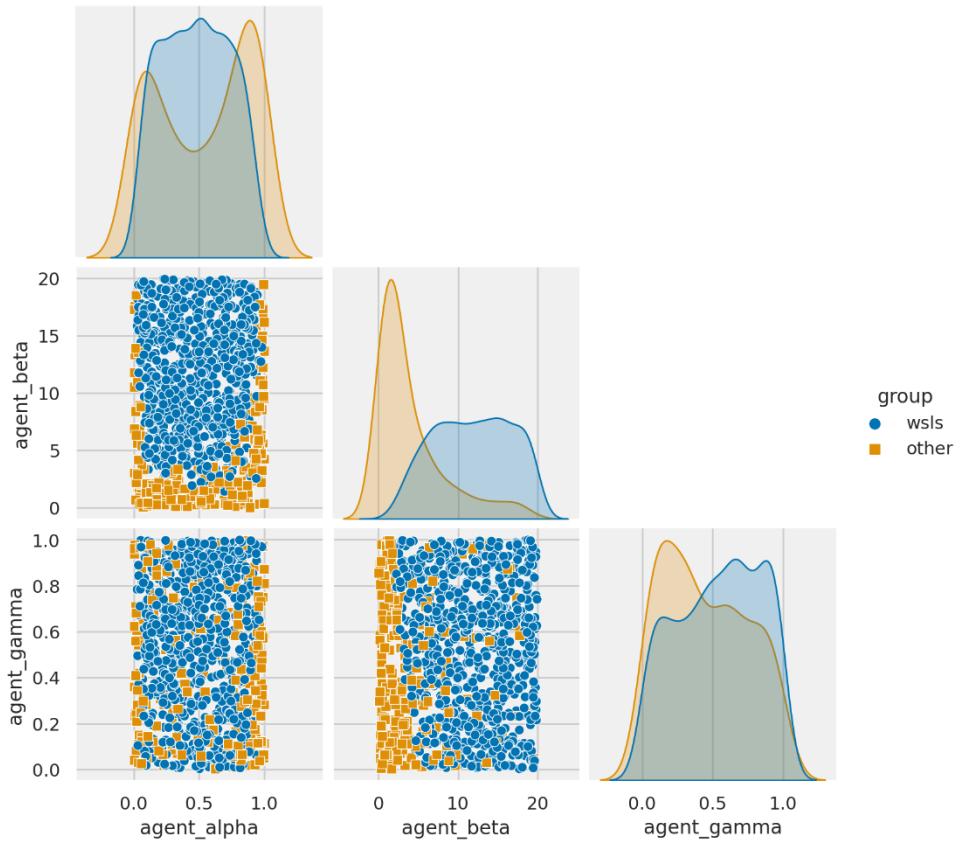


Figure 5.19 QL3 parameter values separated by behavioural group. For the scatterplots, rows and columns are parameter value type (α , β , γ), with the combination of row, column being used as y-axis and x-axis, respectively, for that specific parameter combination. For the diagonal distribution plots, they refer to the parameter in that column. Note that the information on the top right of the diagonal is equivalent to that of the lower left but inverted.

This visualisation is not ideal as we do not get the combination of all three parameters together in one plot. Technically we could plot them together here since there are three parameters, but that method would not work for algorithms with higher number of parameters. Instead, what we can do is use the knowledge from Figure 5.18 that there are simsets with much higher correct and win-stay scores than the 0.5 we use as condition to be grouped as WSLS. Adding a new group of “high WSLS” cases, where correct in shape position one and 2, as well as win-stay in position two and three, are all >0.8 , we get Figure 5.20, where KDE (Kernel Density Estimation) plots replace the scatterplots to increase clarity. With the additional reference point of the strong WSLS cases, we can now see trends in parameter values more easily. High β , low to medium α and γ is preferable somewhere between 0.5 and 0.9 or so. These are still not definitive answers to what parameter value combinations perform the best and illustrates

what was mentioned above that we do not know the exact behaviour of the algorithm before we have tested it.

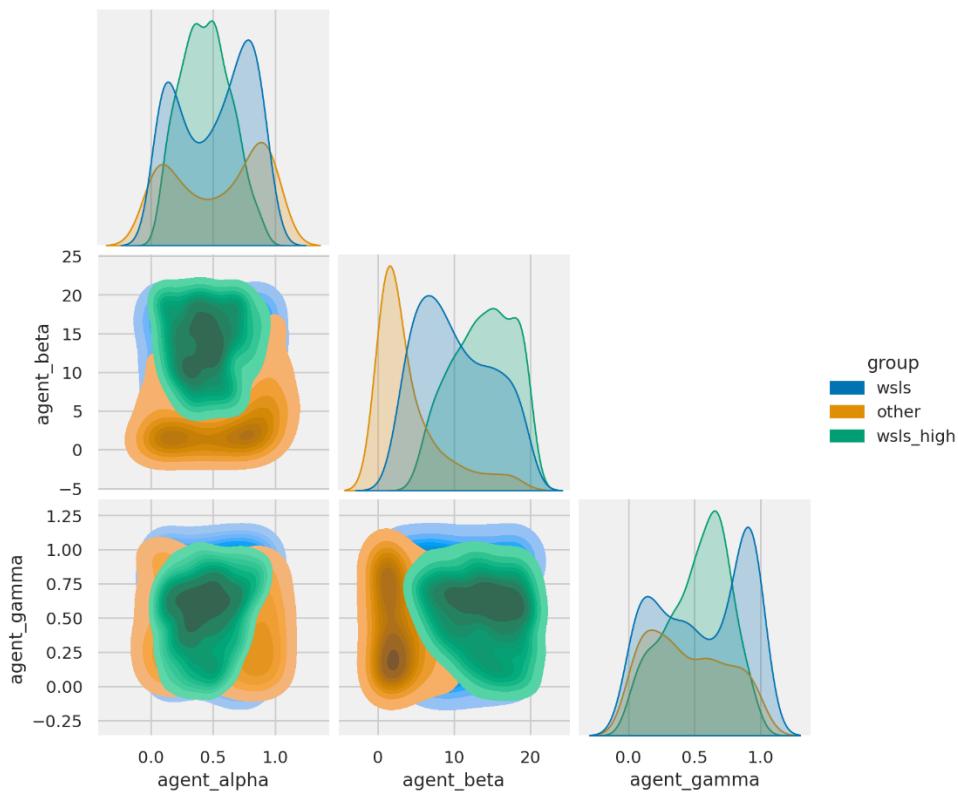


Figure 5.20 QL_3 parameter value plots for each group. The diagonal shows value distribution for each parameter value indicated on column bottom. Combination of parameter values for each group are shown in the three lower left plots, where darker shades indicate higher concentration.

Nevertheless, we do now have an idea of what to expect from QL_3 , namely that it cannot possibly explain the behaviour of those humans who find structure in Shapetask, namely those subjects who learn to predict a shift every third shape. But QL_3 may be able to explain behaviour of those humans applying a WSLS strategy, and we now have an idea of where in the parameter space of QL_3 such humans may fall.

5.2.2 MANIPULATING STATES TO CREATE SEQL₃

This is the point where, usually, alternate RL algorithms like for example model-based RL or other families of algorithms like HMMs would be considered. But we are working from the hypothesis stated in the chapter two that the basic RL system in the mammalian brain is supported by other areas, like for example pre-frontal cortex, by being fed state representations that are appropriate for the current

task. Working from that hypothesis, in what way can we manipulate the states used in QL_3 to make it solve *Shapetask*?

One way would be to use shape position (within a small bag) as the state instead of the shape itself. Another would be to have only two states: last shape of a bag (position three) or not (positions one and two). As researchers knowing the task structure, either of the just mentioned would make sense if we were trying to design a system to solve the task. But participants do not know this. What they see is a stimulus in the form of a coloured shape, that is their “base” state. Shape (and/or colour as noted earlier) should therefore be the basic form of the states we use in the algorithm. Alternately, we could have two separate QL_3 models, one with shape as states and one with shape positions. We then find a switch point where subjects switch from one representation to the other, where some subjects would never switch, and some would do so early. But how would this work if shapes had different bag sizes?

The most straightforward state representation would therefore be to use both shape and position. For our standard version of *Shapetask* with three shapes, each coming in bags of three, that is $3^*3=9$ states. Contrast this with QL_3 which only has three states and therefore has a less complex representation. We previously used the term *SEQL* – State Enhanced Q-learning – in an earlier chapter, and seeing this one is based on QL_3 , we dub it *SEQL₃*.

The state representation of *SEQL₃* is more complex than QL_3 and can be modified further to allow for alternate versions with more shape types and/or different number of shapes in each bag. From the perspective of participants in the *BOB* task this may indeed be the case. Subjects do not know the underlying structure and observe sequences of 3 or 6 same shapes in a row. Even if we only consider position, these subjects may be using 18 states (3 shapes \times 6 sequential positions). In *BOB-NR* there are only ever three same shapes in a row, so it may be the case this state enhancement is more appropriate for the *BOB-NR* task, than the *BOB* version.

5.2.3 SIMULATING *SEQL₃* BEHAVIOUR IN *SHAPETASK*

The simulation of simsets for *SEQL₃* is done with the same procedure and parameter value ranges as described above for QL_3 in Section 5.2.1. The only difference with *SEQL₃* is the state representation, where, if the shapes circle, triangle and square are represented by 1,

2, and 3 respectively³⁵, the stimulus sequence for a given BOB task is converted to include position information as:

$$\begin{aligned}[1, 1, 1] &\rightarrow [1, 2, 3] \\ [2, 2, 2] &\rightarrow [4, 5, 6] \\ [3, 3, 3] &\rightarrow [7, 8, 9]\end{aligned}$$

For each SEQL state 1-9, there are three actions, meaning in total we have 27 state-action values. Note we have condensed the presentation below to the most relevant plots, compared to what was presented for *QL3*. Additional plots are available in the code repository.

The results for all simsets are plotted in Figure 5.21, where a general pattern stands out. It seems few, if any, simsets exhibit WSLS behaviour, indicated by no lines with increasing trend from position two to three in the win-stay column, second from left. Rather, the win-stay scores, together with the trend for shift prediction (rightmost column) going upwards from position two to three, tells us that many *SEQL3* simsets can “win” the task.

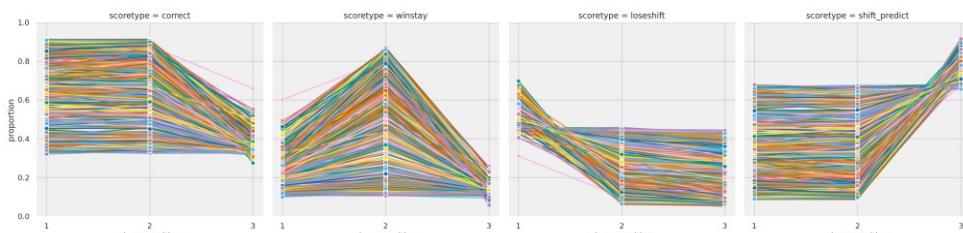


Figure 5.21 *SEQL3* performance in Shapetask for each simset (separate lines), averaged across all trials. Proportion of choices (y-axis) for each shape position (x-axis) and score type (columns). From left to right: Correct, Win-stay, Lose-shift, Shift-predict.

Grouping the simsets according to the conditions defined in Section 5.1.5, we can see in Figure 5.22 that our suspicions are correct. In fact, *SEQL3* does not exhibit WSLS behaviour at all, at least not as defined by the criteria.

³⁵ Assuming 1-based indexing. The values would be 0, 1, 2 in Python code.

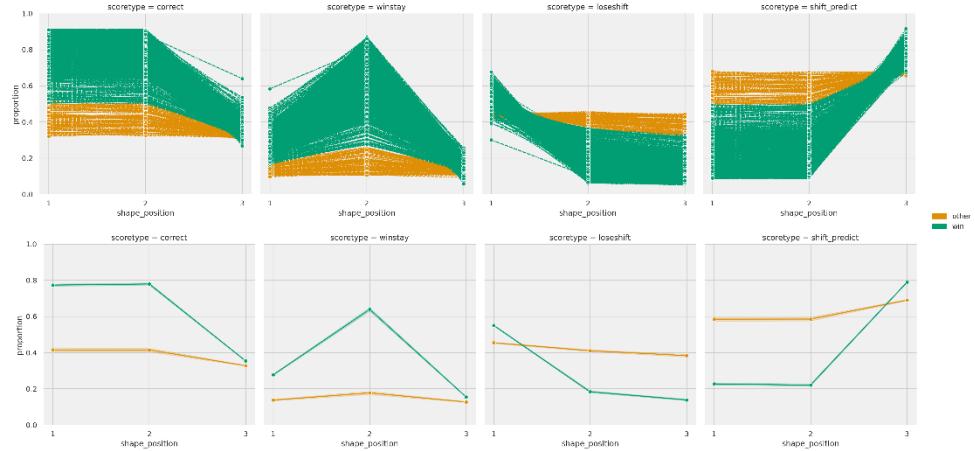


Figure 5.22 Behavioural groups for $SSEQ3$ in Shapetask. Proportion of choices (y-axis) for each shape position (x-axis) and score type (columns). Top: All simsets represented by individual lines, coloured by group as per legend. Bottom: Average of each group. From left to right: Correct, Win-stay, Lose-shift, Shift-predict.

To investigate parameter value combinations, we add a “high win” group for $SSEQ3$, in the same way that an additional high performance WSLS group was added for the $QL3$ agent. The conditions for the high win group use the same score type and positions as the win group, but the average scores are raised to being more than 0.8 instead of more than 0.5.

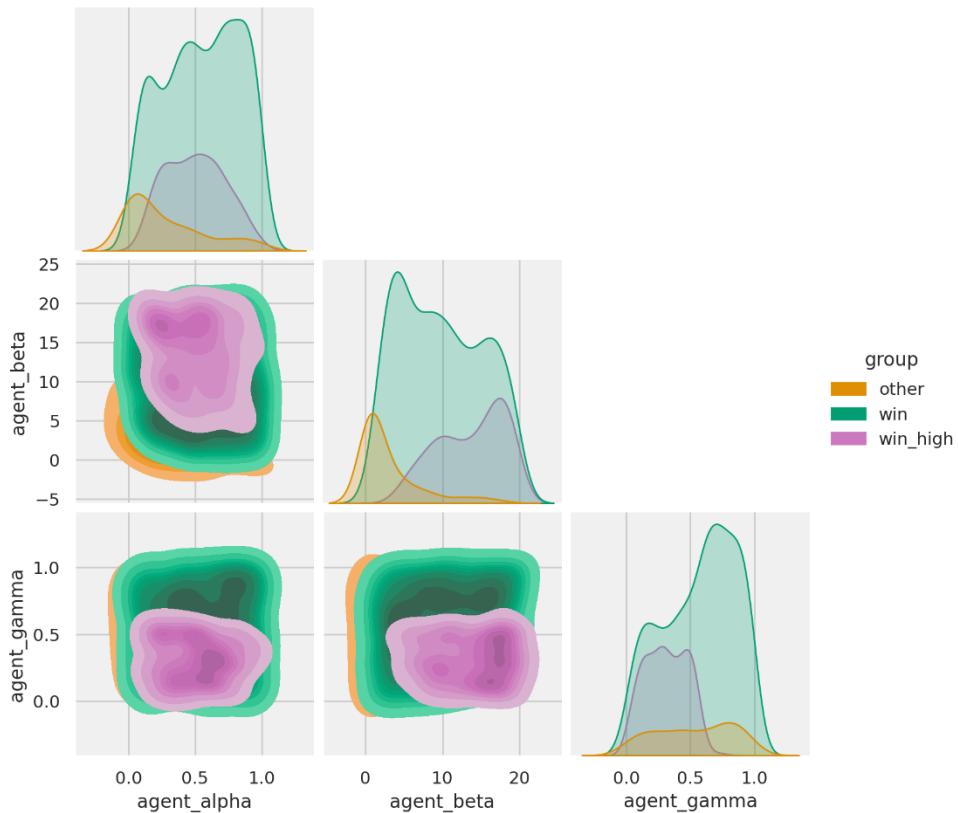


Figure 5.23 Parameter value correlations for SSEQL3 separated by groups (colours per legend). Diagonal distribution plots refer to the parameter indicated along bottom. KDE calculated surfaces are shown for parameter value combinations in the lower left, where darker shades indicate higher concentration.

In Figure 5.23, we can see that similar to QL_3 , $SSEQL3$ can win across most of the parameter space, except for combinations of low α and β (green win surface compared to orange other group surface and distributions). It should be noted that the large overlap between win and high win groups is due to the random variation for each parameter combination. Even though we run 100 simulations on the same task sequence for each such parameter combination, performance can vary across the selected limit for win high cases. As we have discussed above, our grouping criteria are chosen arbitrarily and meant as guidance for our investigations. From that standpoint, the important parameter value group differences to look at are between the high win group and the other group³⁶. For the high win group, we see that higher β is preferred compared with the “other”

³⁶ Independent T-tests between the two groups; $\alpha: t(325) = 7.6, p < 0.001$. $\beta: t(325) = 21.1, p < 0.001$. $\gamma: t(325) = -8.5, p < 0.001$.

group, combined with $0 < \gamma < 0.5$. The other group additionally has a higher degree of low α than the high win group.

Importantly, and as already mentioned above, we now know that $SEQL_3$ can account for the well performing subjects in the human dataset. But like QL_3 , it cannot account for all types of participants.

5.2.4 GENERAL DISCUSSION OF RL BEHAVIOUR

What does it mean that QL_3 cannot solve Shapetask, but $SEQL_3$ can? In essence, it means that our results – at least in this specific task and algorithm niche – are in favour of the hypothesis that state representation is critical, at least for some human participants. This is an important insight, because it tells us that perhaps low-level mechanisms like RL are indeed very similar across mammal species, and that what differentiates complex from less complex behaviour is how other brain areas are able to create, transform and provide suitable state representations to the RL system.

The fact that QL_3 and $SEQL_3$ exhibit WSLS and win group behaviour, respectively, across most of their respective parameter spaces adds to this line of reasoning that it is the state representation that makes the difference, not specific parameter value combinations.

These results agree then with other research discussed in the background chapter, that perhaps it is the case that the dopaminergic RL system works in tandem with (mainly) pre-frontal cortex and/or hippocampus, where state and task representations are manipulated/constructed in order to support learning.

However, we do not have any firm evidence to connect $SEQL_3$ with biological phenomena. We have selected the state representations used in $SEQL_3$ based on what makes logical sense, but we have not presented any mechanism or proof that $SEQL_3$ is how humans solve the task.

What we would like is a more principled way of reaching results like those of $SEQL_3$, based on algorithms and/or state representations that other research has shown has possible biological correlates. In other words, there are more models to consider before fitting our human data.

5.3 HIERARCHICAL RL

Hierarchical RL (HRL) is a broad term involving the concept that animals, in particular humans, can find and use environmental structure to guide and inform behaviour. This broad concept has multiple interpretations and implementations, each focussing on various aspects of algorithmic details. For example, [28, 256] frames the hierarchical notion as one about actions, where multiple “primitive” actions (heat water, put teabag into cup, pour water) are chunked together into “action options” (make tea). Since we are more interested in the state representation aspect³⁷, here we will focus on hierarchical RL as collections of strategies, or task sets [49, 53, 70, 73, 220].

The basic idea of task sets (TS) is that humans and other animals have different behavioural contexts, where the same stimulus can cause different (re)actions depending on the current context. New task sets can be created in new situations, selected between and reused across different contexts, allowing them to deal with generalisation, differentiation and hidden information [53, 73].

The implementation of HRL we focus on here is based on [73] and is essentially two QL algorithms stacked in hierarchical fashion, see Figure 5.24. Based on the context, which works as a higher-level stimulus, one of several task sets is selected using SoftMax, based on the Q-values of each task set. Every task set consists of its own set of stimulus-action values like the regular QL/QL₃ agents. Based on the reward received, separate prediction errors are calculated for the stimulus-action values and the context-task set values which are then used to update the values on the respective levels. Mathematically, we thus have, for each timestep t , context c :

$$\begin{aligned} Q_{t+1}(c, TS) &= Q_t(c, TS) + \alpha_{high}[r_t - Q_t(c, TS)] \\ Q_{t+1}(TS, s, a) &= Q_t(TS, s, a) + \alpha_{low}[r_t - Q_t(TS, s, a)] \end{aligned} \quad 5.2$$

where high and low indicate hierarchy level. As mentioned above, both levels use SoftMax to select task set and actions but use separate parameters $\beta_{high}, \beta_{low}$. When simulating and fitting their “alien task”,

³⁷ We should note that states and actions are sides of the same coin; both are part of the state representation. What actions are available are shaped by how one sees the world, and vice versa.

[73] used an additional forget parameter, used to reduce values on each trial to account for participants forgetting over time. We do not use that in our implementation (see below), as *Shapetask* with its 99 trials is relatively short in comparison to the hundreds of trials in the alien task.

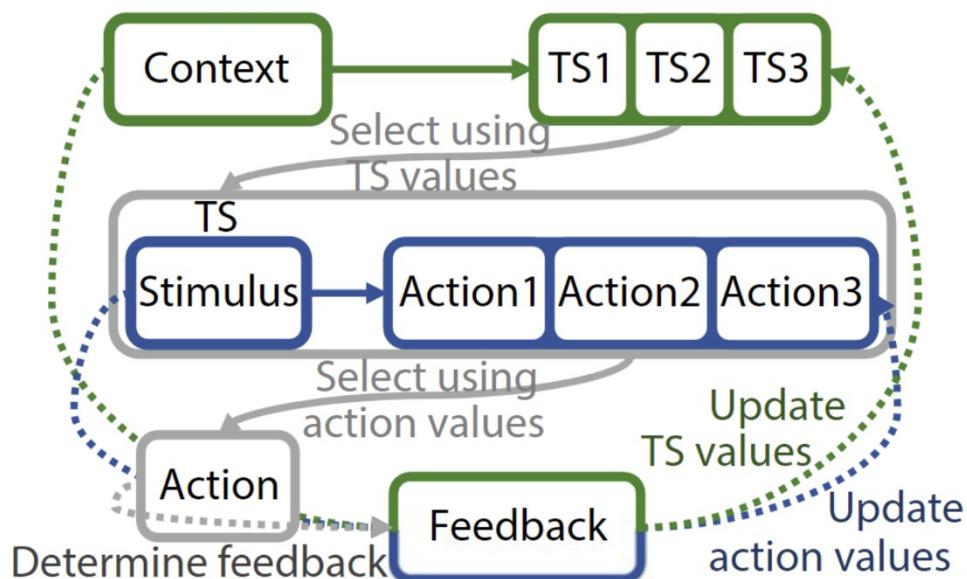


Figure 5.24 Overview of HRL. Context and task sets (TS) are at the top of the hierarchy and marked in green. Based on the context, here TS2 is selected. Based on the stimulus, an action is selected based on the action values of TS2, marked in blue. The selected action leads to feedback (reward) which is then used to update the values for the action and task set. Adapted from [73].

The HRL algorithm is used in [73] to model human behaviour in a task where four different aliens (stimuli) are given different objects like an umbrella (actions) which they appreciate differently depending on what season it is (context), for example winter or summer. The participant is then rewarded between 1-10 points, indicated by a measurement tape on screen. The reward size is based on if the context-stimulus-action was correct or not, with Gaussian noise added. The experiment was conducted with a practice training session of 40 trials – where one trial is one context-stimulus presentation – followed by a learning phase with 468 trials (3 blocks per context and 52 trials in each context). Four test phases followed; the first was similar to the learning phase but had the context hidden,

indicated by grey clouds, second a comparison phase where participants either saw two contexts or two stimuli and selected the one they preferred, third a novel context phase, and fourth and finally a mixed phase where context and stimuli were not blocked but changed trial-to-trial. Between each test phase there was a refresher of the learning phase, but in a shorter 120-trial version. The test phases, including the refreshers, added up to 1197 trials and thus 1705 trials in total when including the training and initial learning phase.

[73] shows nicely how HRL fits the human behaviour better in this task, across the learning phase and test phases, than both a “flat” RL model (with separate states for each context-stimulus combination, like our *SEQL3* above) and a hierarchical Bayesian inference model. Most importantly, participants successfully reactivated learned task sets in the hidden context phase and generalised knowledge of existing task sets to the novel context phase.

We can see how HRL can handle for example the hidden context phase, in that participants are informed about context switches but not about their identity. By manually equalising the context-TS values when the phase starts, [73] showed that HRL can relearn these values while the stimulus-action values in each TS remain intact.

In addition to these promising results, there is previous evidence of HRL being biologically plausible [48, 49, 53, 70], and as discussed in the background chapter there is ample evidence of hierarchical levels in brain function in general and for (reward) prediction errors in particular. In other words, HRL is a strong candidate for *Shapetask*, though it remains to be seen if it works with the low number of trials we used.

5.3.1 APPLYING HRL TO SHAPETASK

The key to applying HRL to *Shapetask* is to decide what the context would be and how many task sets there are. In the Alien task described above, the context and task sets are more obvious, as the experiment task has been developed with HRL in mind. In *Shapetask* it is not as straightforward, as only the shape types (and the sequence) are directly observable. One way to frame it would be to have the shapes (or colours) as stimuli and impose structure on top with positions as context/task sets. However, what if the participant frames it the other way around, with shape type as context and shape sequential position as stimulus (or, rather, what is on top and bottom

in the hierarchy)? Theoretically, the hierarchical order should not matter here as the number of combinations between shapes and positions remain the same. Furthermore, one might ask whether there should always be one task set per context? The latter question is discussed in [53], where it is shown that clustering of contexts to use the same task set is precisely how HRL allows for generalisation of knowledge across contexts. This would allow for an alternate interpretation of having two contexts/task sets, consisting of position being last in the bag or not. Work by [54, 70] discusses task sets in relation to working memory capacity and HRL, and suggest a maximum human capacity of 3-4 concurrent task sets but 2-3 are optimal.

These considerations highlight both the difficulty of applying algorithms to new tasks (i.e., ones that the tasks were not specifically designed for) as well as the importance of doing so, as questions arise that may not otherwise have been posed and investigated.

Nevertheless, following the fact that the shape types are directly observable and that contexts can be clustered across task sets, it would make the most sense to use shape position as context, with one task set per position, and each task set consisting of shapes as stimuli as well as actions.

5.3.2 HRL RESULTS IN SHAPETASK

Our HRL implementation is based upon Equation 5.2, and uses shape position (1, 2, 3) as context. For each context, there is one task set (TS₁, TS₂, TS₃) and each TS corresponds to 3x3 state-action values like for the QL/QL₃ agents. We generate 1000 parameter value combinations and simulate 100 subjects for each such combination (simset) and the BOB version, with parameter values drawn as:

$$\alpha_{high}, \alpha_{low} \sim U(0, 1), \quad \beta_{high}, \beta_{low} \sim U(0, 20)$$

Results will first be presented in an overview fashion as for the regular RL agents above. Because we have an additional and novel aspect here compared to those previous algorithms – the contexts and task set selections – we will additionally investigate task set selection with regards to performance group (wsls and win).

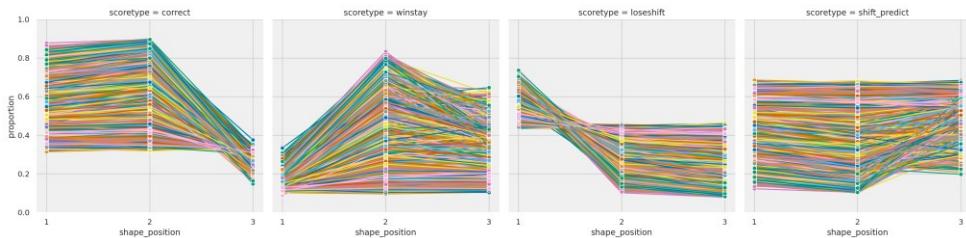


Figure 5.25 Overview of HRL performance in Shapetask. Proportion of choices on y-axis for each shape position on x-axis, across each score type in columns. Each line is a separate simset. From left to right: Correct, Win-stay, Lose-shift, Shift-predict.

In Figure 5.25 we have the overall performance of all simsets. We can glean that there are patterns of WSLS in that some lines have flat or increasing trend from position two to position three for win-stay (second column from left). There are also patterns of win in that there are increasing trends from position 2 to position three for shift predict (rightmost column). We confirm this in Figure 5.26 where simsets are divided into groups as per conditions defined above³⁸.

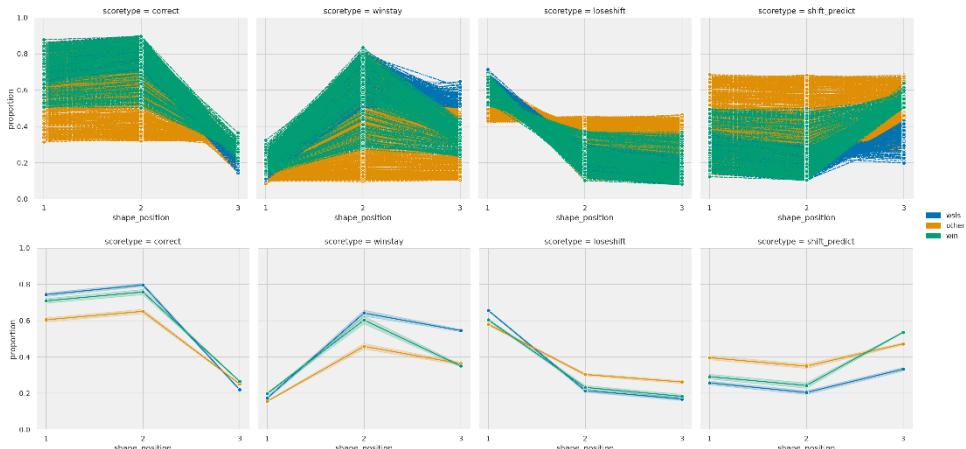


Figure 5.26 HRL performance in Shapetask by group. Proportion of choices on y-axis for each shape position on x-axis and score type (columns). Groups coloured as per legend. Top: All individual simsets as separate lines. Bottom: Group averages across simsets.

There are two main findings of interest in Figure 5.26. The first is that it appears HRL is capable of exhibiting both WSLS and win behaviour. This is potentially exciting, because an algorithm that can explain a wider range of behaviours is preferable to those that only explain a few. However, the second interesting find here is that these behaviours are not as distinctive as those of the strongest human examples of WSLS and winner subjects where the corresponding

³⁸ The proportions of groups were 69% other, 21% win and 10% wsls.

score proportions for win-stay and shift predict are much higher (compare middle row Figure 5.11). We will come back to this observation when investigating task set selection below. Overall, the win score levels are comparable to human BOB levels. The downside of these results is that most simsets are categorised as other, but perhaps this will prove a decent fit to some humans struggling to find a pattern in the task.

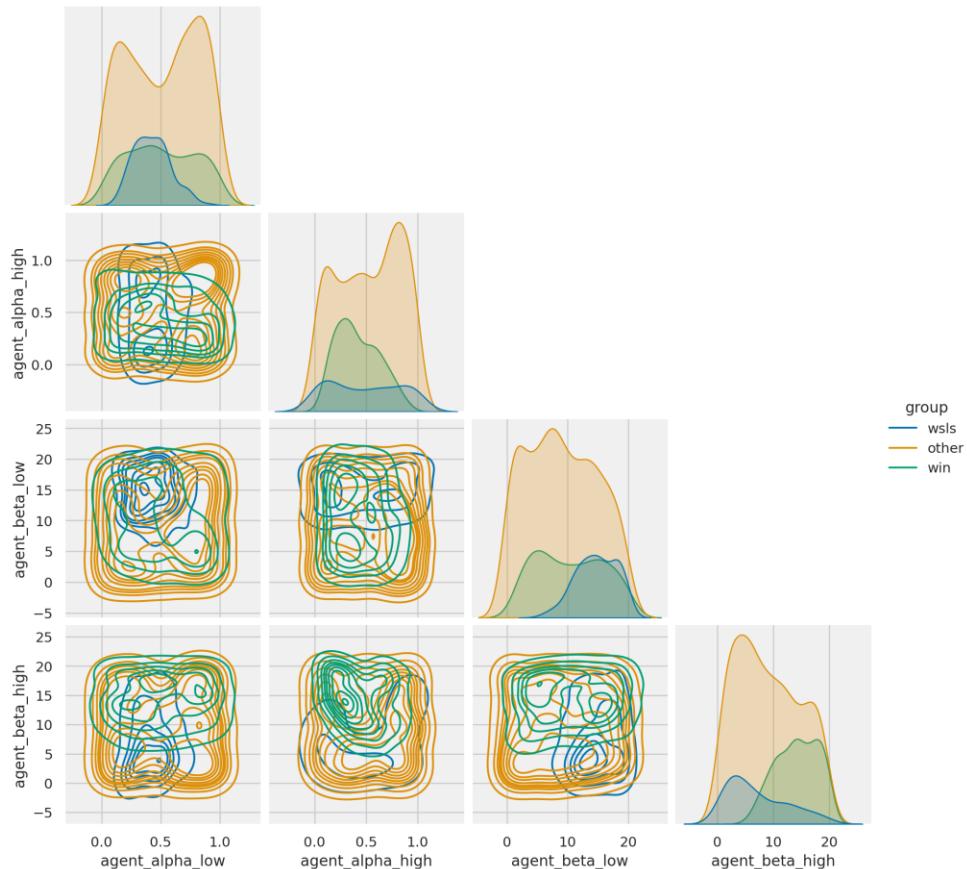


Figure 5.27 Parameter value pair plot for HRL in Shapetask. On diagonal, parameter value distributions as per labels at the bottom. The bottom left triangle of plots show joint distributions for each parameter value combination as per row and column labels combined. Colours indicate group as per label.

In Figure 5.27 we have plotted parameter value spaces for each parameter and group. To better see overlap, we here use “unfilled” KDE plots on the off-diagonals. Interestingly, the “other” group behaviour is possible across the entire parameter space. To win, what looks most important are the high-level parameters for selecting task set, with high value β_{high} , but values on the lower level do not seem to matter as much. For WSLS, the reverse is true, where a low value

β_{high} is preferred, and on the lower level a high β_{low} plus α_{low} between around 0.1 and 0.7.

Of the best performing simsets in the winner group, only a few reach above 0.6 for shift predict in position three. Similarly, only a few simsets reach above 0.6 for win-stay in position three in the WSLS group. We are interested to see if these simsets have plateaued in performance or if there is still an upwards trend towards the end of the experiment.

To investigate this, we chose to use the two last bag-of-bags of the task, meaning the last 18 trials, and averaged each score type across only those trials. From these scores we then classify “win high” as those cases that are >0.8 for correct in positions one and two, as well as >0.8 for shift predict in position three. Similarly, “WSLS high” are those that are >0.8 correct for positions one and two and >0.8 win-stay for positions two and three. This gives us Figure 5.28 where we see line shapes more similar to the levels humans achieved over the whole task (light blue and pink for high winner and high WSLS, respectively). Note that the figure shows behaviour for all trials, not just the last 18 trials used for grouping.

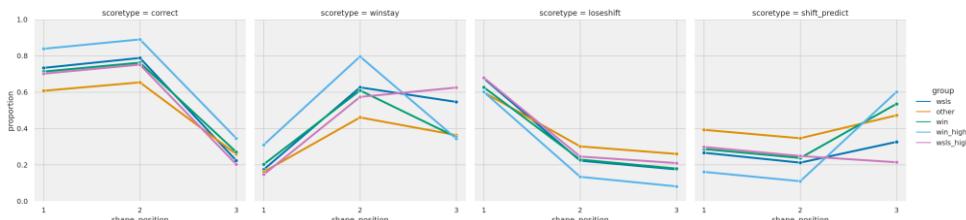


Figure 5.28 Proportion of choices (y-axis) for each shape position (x-axis) and score type (columns) for each group (coloured lines as per legend). From left to right: Correct, Win-stay, Lose-shift, Shift-predict.

This is encouraging, as it seems HRL is indeed able to qualitatively produce behaviour like humans. However, very few cases reach these criteria. Fewer than five (depending on random variation for parameter value combinations) out of the thousand simsets are “win high” or “wsls high”.

5.3.2.1 Taskset selection

We now turn our attention to the process of selecting task sets in the HRL algorithm for *Shapetask*. Since task set selection values have to be learned, there is stochasticity involved in whether good context-TS

values are learned. Overall, as seen in Figure 5.27, greedy values for β_{high} are thus preferred in order to win *Shapetask*.

However, because of the just mentioned stochasticity in selecting task sets, it is possible to see completely different behaviour types for different runs of the same parameter values. In other words, if we run HRL two times – both times with the same parameter values – we can get WSLS behaviour one time and winner behaviour the other.

To highlight the dynamics involved, we repeat the same process as in the previous section of simulating 1000 simsets. We then find a candidate simset that has the highest value for Shift-predict in position three, averaged over the last 18 trials. The example simset shown below thus have 100 subjects, all using the same parameter values $\alpha_{low} = 0.79$, $\alpha_{high} = 0.51$, $\beta_{low} = 14.85$, $\beta_{high} = 19.92$.

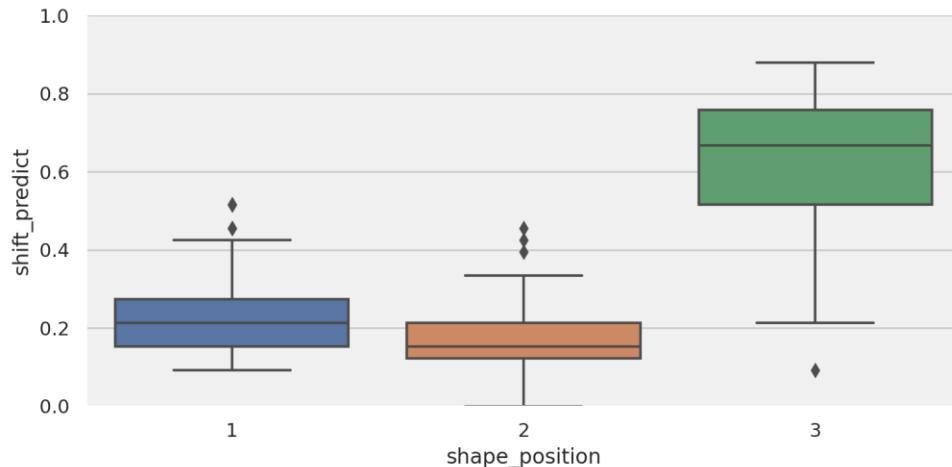


Figure 5.29 Proportion of Shift-predict (y-axis) for each shape position (x-axis) for example simset found as explained in text. Boxplots show the variance across subjects within the simset.

In Figure 5.29, the subjects within the example simset have been plotted, showing only the Shift-predict score across all trials for each shape position. The variance for position three is quite large, spreading across almost the entire value range. To investigate further, we find the best and worst subjects within this simset, defined by comparing the mean of Correct score for positions one and two together with the Shift-predict score for position three. The best subject is thus the one with the highest such combined score, and the worst subject that with the lowest. Because we store what task set is being selected on each trial, we can see what task set was selected on each trial.

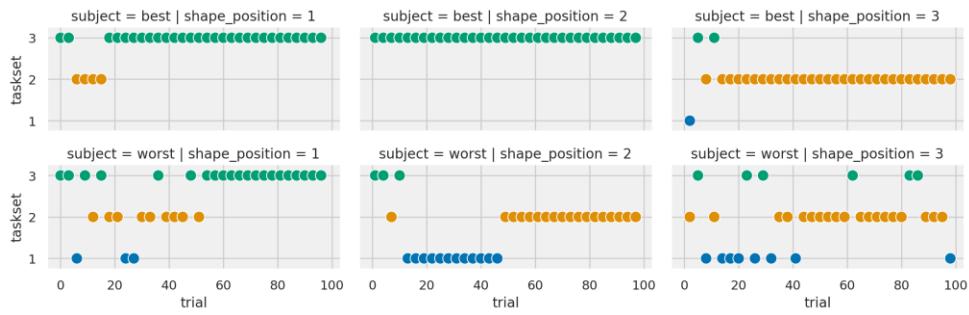


Figure 5.30 Taskset selections (y -axis) for each trial (x -axis) and subject type (rows). Colours refer to the same value as y -axis to make figure easier to read. Columns show the trials in that shape position. From left to right: Shape position 1, Shape position 2, Shape position 3.

In Figure 5.30, task set selections for every trial have been plotted for both the best (top row) and worst (bottom row) subjects. The best subject has early on successfully generalised across shape positions one and two and use the same task set for both positions (task set 3 in green circles, left and middle columns), while using a separate task set for shape position three (task set 2, yellow circles, right column). This allows these two task sets to be optimized for the task.

Meanwhile, the worst subject is more varied in the first half of the task. Around halfway, it settles on task set 3 in green circles for position one (left column) and task set 2 in yellow circles for position two (middle column). For shape position 3 (right-most column), the subject has in the last half of the task picked mostly task set 2 in yellow circles (right column).

What different kinds of score behaviour might our best and worst subject examples show?

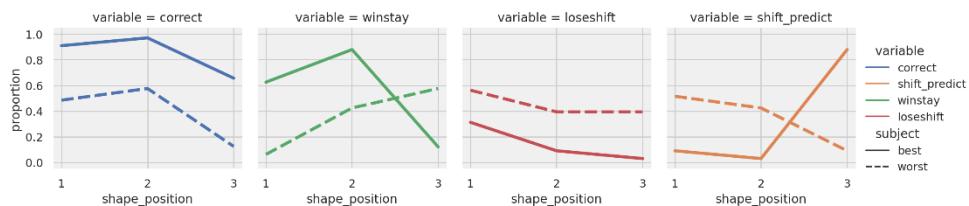


Figure 5.31 Behaviour of two example subjects from the same simset. Proportion of choices (y -axis) for each shape position (x -axis) and subject type (line styles as per legend). From left to right: Correct, Win-stay, Lose-shift, Shift-Predict.

In Figure 5.31 we can see the consequences of the different task set selections of the two subjects. The best subject (solid line) shows the same pattern as human winners, namely high correct score in the two first shape positions and high Shift-predict for position three. The

worst subject (dashed line) instead shows a pattern similar to human WSLS subjects, where Win-stay increases from shape position two to three.

In other words, two subjects that use the same parameter values exhibit qualitatively different behaviours. We should note that we can also find examples that will behave similarly to the best subject above, but that select different task sets for each position. What is crucial, then, is that the agent selects a unique task set for position three. This is what the worst subject struggles with as seen above.

This issue of large variance in task set selection can also explain why the “other” group in for example Figure 5.26 has a trend towards the behaviour of winners but does not quite get there³⁹.

5.3.3 CONSTRAINING STRUCTURE

As discussed above in Section 5.3.1, there are more constrained variants of the context-task set structure. We could keep the three positions as context and reduce the number of tasksets to two. Further reduction can be made by using two contexts: last shape in the bag, or not, together with two task sets. The latter mentioned is the smallest possible setup, and thus what the following results will be based upon. For convenience, this version of HRL is called *HRL-22*⁴⁰.

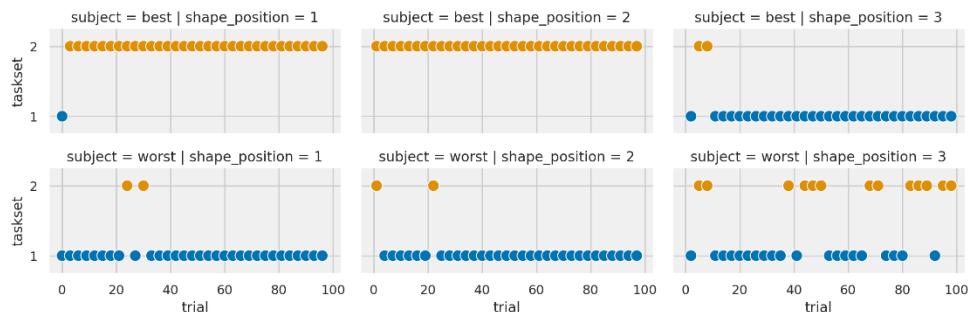


Figure 5.32 Taskset selections for best and worst subjects in best performing simset using HRL-22. Taskset selected is y-axis, across all trials (x-axis) for each shape position (columns). Top: Best subject. Bottom: Worst subject.

Unfortunately, as seen in Figure 5.32, the same issue remains even when constraining the context-TS structure to a minimum. If we increase the number of trials in the BOB Shapetask to 270, however,

³⁹ See plot in code repository, https://github.com/fohria/phd_thesis

⁴⁰ In code we call this “lastinbag”

then as seen in Figure 5.33, after around 160 trials even the worst performing subject (bottom row) can separate the task sets properly.

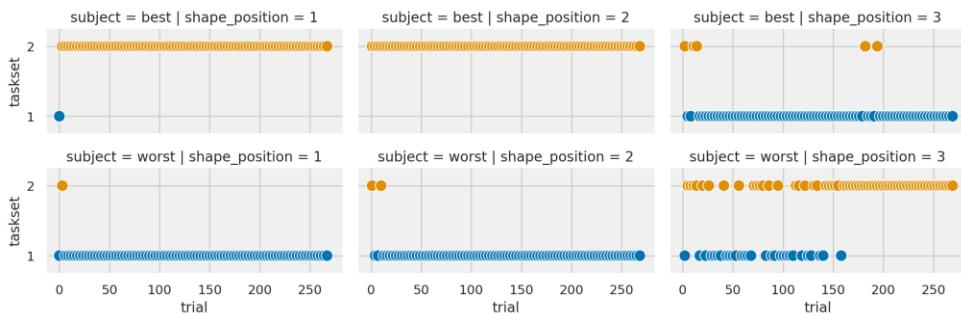


Figure 5.33 Taskset selections for best and worst subjects in best performing simset using HRL-22 and 270 trials in Shapetask. Taskset selected is y-axis, across all trials (x-axis) for each shape position (columns). Top: Best subject. Bottom: Worst subject.

That this switch in taskset usage translates to “winning” behaviour can be confirmed in Figure 5.34.

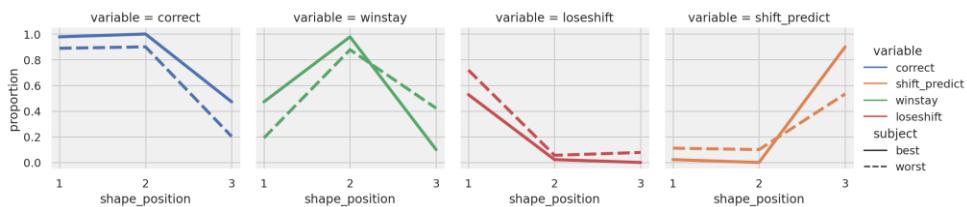


Figure 5.34 Behaviour of the worst and best subjects in the best performing simset using HRL-22 and 270 trials in Shapetask. Proportion of choices (y-axis) for each shape position (x-axis) and subject type (line styles as per legend). From left to right: Correct, Win-stay, Lose-shift, Shift-Predict.

Can we find that the same pattern holds for the original HRL implementation when extending the task to 270 trials? Theoretically, it should require more trials than for HRL-22, since HRL has one more context and task set hence there are more ways the stochasticity can play tricks.

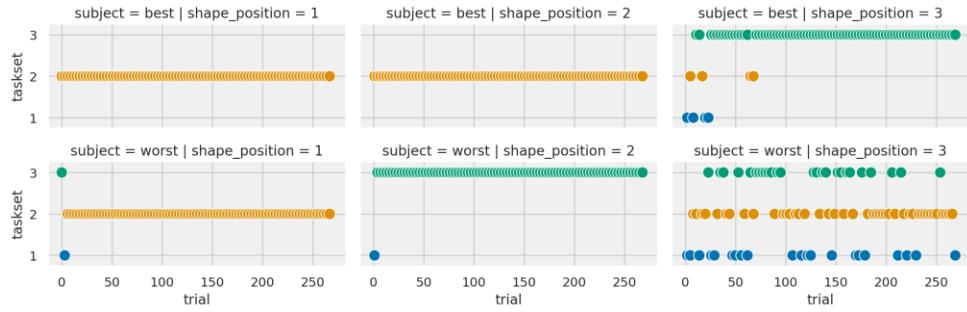


Figure 5.35 Taskset selections for the best and worst subjects in best performing simset using HRL and 270 trials for Shapetask. Taskset selected is y-axis, across all trials (x-axis) for each shape position (columns). Top: Best subject. Bottom: Worst subject.

As seen in Figure 5.35, 270 trials do not appear to be enough for the original HRL implementation to optimize the taskset selections. This is further evidenced by the behavioural plot of Figure 5.36.

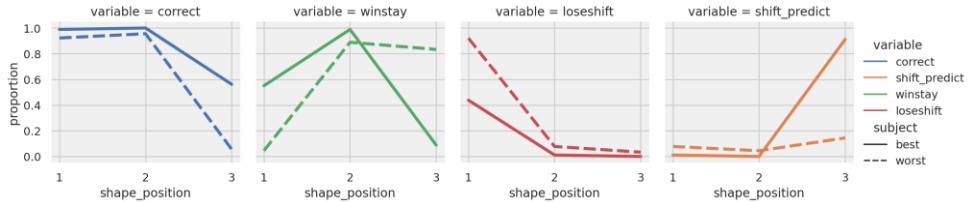


Figure 5.36 Behaviour of the best and worst subjects in best performing simset using HRL and 270 trials for Shapetask. Proportion of choices (y-axis) for each shape position (x-axis) and subject type (line styles as per legend). From left to right: Correct, Win-stay, Lose-shift, Shift-Predict.

5.3.4 OVERALL BEHAVIOUR OF HRL-22

In the previous section we found that our alternate implementation called *HRL-22* may provide better results when it comes to finding winners. However, a positive aspect of the original HRL results were that both WSLS and winner behaviour was possible, which is a good thing when it comes to fitting human data, if the goal is to explain something about the human behaviour. If one type of algorithm or state representation can explain more types of behaviour, that is preferable as it explains a larger number of observations.

Therefore, we run 1000 simset simulations for *HRL-22*, as we did for *HRL* above, group them as per our existing conditions and check potential parameter value correlations between these groups. We will use 99 trials for *Shapetask BOB* variant, since our human data uses 99 trials.

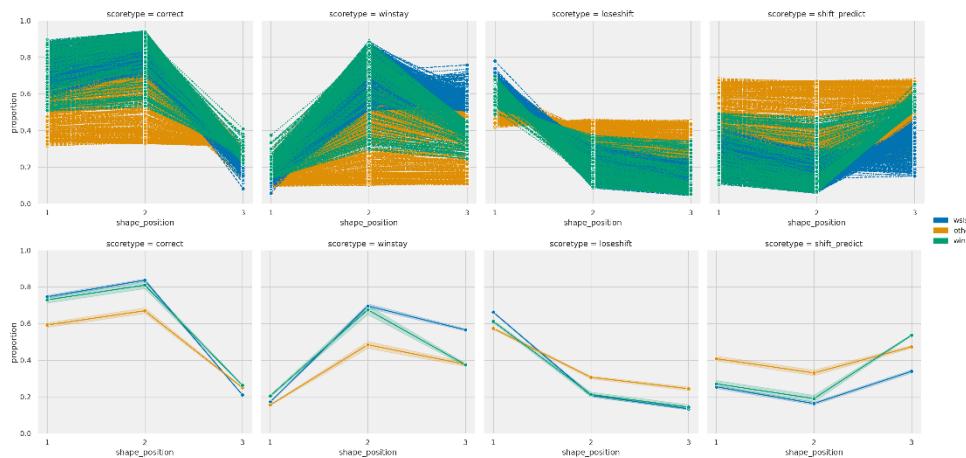


Figure 5.37 HRL-22 performance in Shapetask, separated by groups as per legend. Proportion of choices (y-axis) for each shape position (x-axis) and score type (columns). Top: Individual simsets separated by lines. Bottom: Average for each group. From left to right: Correct, Win-stay, Lose-shift, Shift-predict.

Interestingly, the constrained state representation of HRL-22 does not noticeably increase general performance compared with HRL. This is perhaps not surprising, following our previous finding that HRL-22 requires more than 150 trials to stabilise and separate task sets. What is positive, is that even the constrained context-TS representation of HRL-22 can still produce WSLS cases. Furthermore, the group proportions are now more balanced. For HRL-22 we get 53% other, 30% WSLS and 17% winners.

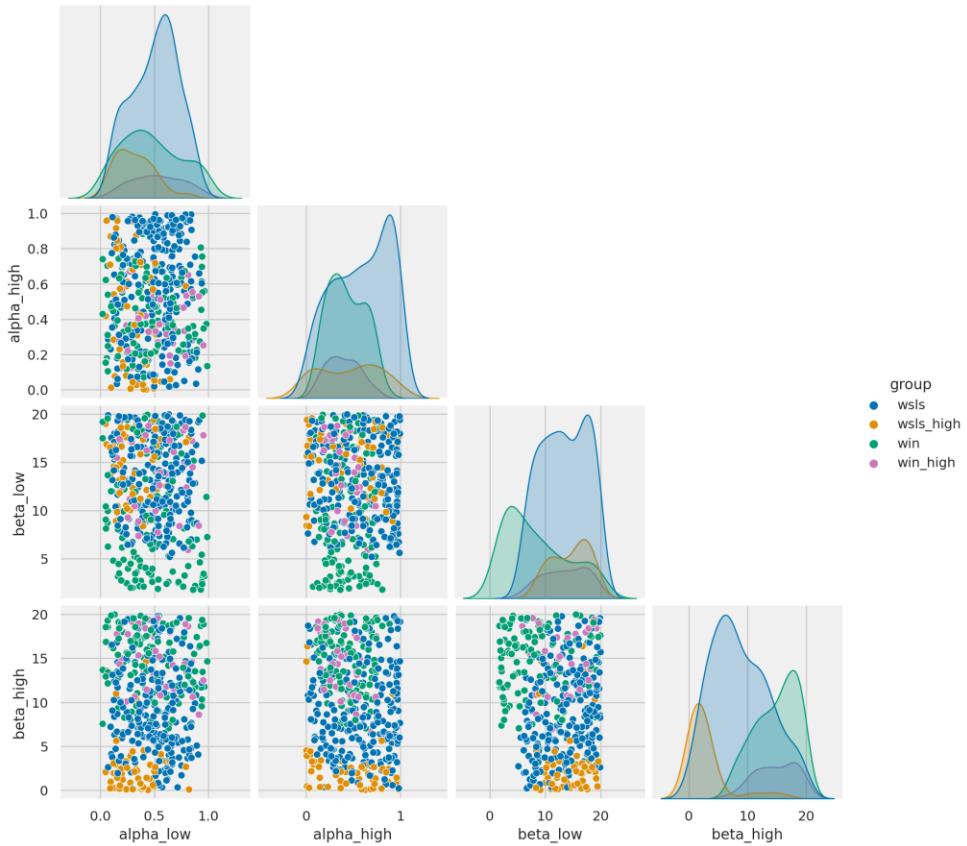


Figure 5.38 Pair plot comparing parameter value spaces for each group. Diagonal shows distribution for the parameter labelled at the bottom. Scatterplots show combination of parameter values as per row, column combination. Colours indicate group as per legend.

In Figure 5.38 we have plotted the parameter space of HRL-22. The “other” group has been excluded for clarity, as it still covers the majority of the combined parameter space. Two groups have been added in addition to the regular Win and WSLS groups, indicating cases with >0.7 shift predict in position 3 (Win-high group) or >0.7 Win-stay score in positions two and three (WSLS-high group). We chose 0.7 here instead of 0.8 as above in order to have more cases in the “high” groups to easier distinguish parameter space distributions.

With regards to the parameter space itself, it is clear that for $\alpha_{low}, \alpha_{high}, \beta_{low}$ there is much overlap between the groups. The main difference is found by comparing the Win-high and WSLS-high groups, in that the latter prefers low β_{high} while the former prefers high β_{high} . This makes sense from what we saw above when investigating taskset selections. With low β_{high} , the WSLS-high group takes many more explorative selections – random behaviour if you prefer – and there is less chance that distinct task sets for shape position three will appear and continue to be chosen. Without any

proper discrimination between task sets, the agent becomes essentially context-less and thus falls back on the WSLS behaviour we saw for the regular QL3 agent.

But with high values for β_{high} , the agent is equipped to distinctively create and select separate task sets for the appropriate context. In the best cases, as we saw above, the agent will select separate task sets for position three and positions one and two, enabling it to learn how to win at Shapetask.

5.3.5 HRL DISCUSSION

HRL is capable of exhibiting both WSLS and Winner behaviour, which makes it a fitting candidate to explain a majority of human behaviour in Shapetask. Unfortunately, due to inherent randomness in selection of task sets, even for a single combination of parameter values, HRL can exhibit this duality of behaviour. This issue is alleviated by increasing the number of trials, and is likely the reason why [73], which the HRL implementation is based on, uses around four times as many trials in its learning phase for both humans and algorithm, as we do in Shapetask.

In the human results for Shapetask, however, we show that some humans can find the task structure within 99 trials, and often in the early parts of the task. HRL is therefore not as good a model as *SEQL3*, at least in the case of explaining those well-performing humans.

To be fair, humans do come into a task with a great deal of pre-existing knowledge. We can account for that in two ways. The first way would be to pre-train our model in some fashion, an approach common in the deep learning approach to cognitive science. The crux there is figuring out how much training accounts for evolution, how much for life experience, and how much is training/learning in the task at hand?

The other approach is the one used here, to manipulate the state representation in ways that may account for how humans may come into a task with pre-existing knowledge. *SEQL3* provides a better account for well-performing humans, but HRL can explain a broader spectrum of behaviours. Furthermore, in the *BOB* task, some human winners do have performance scores on similar levels as the average best HRL simsets.

The interesting part of HRL is how to decide and construct the context and task sets. [174, 218] show how this can be done in a more dynamic way, using non-parametric methods to add new contexts and task sets. It is not immediately clear how this would be applied to Shapetask, as one would still need to define what aspects of the task are used as building blocks.

To summarise, HRL may be able to explain a wide section of the human data. The above behavioural demonstrations have also shown how the application of alternate algorithms on Shapetask illuminates aspects of those algorithms that may not be apparent in their original task environments.

5.4 THE SUCCESSOR REPRESENTATION

The Successor Representation (SR) was introduced by [174] and can be seen as a middle ground between model-free (MF) and model-based (MB) RL [175, 219]. SR does not learn the full state transition function, like MB RL. Instead, SR approximates it through experience, storing how often future states will be visited from the current state. The resulting so called “future state occupancy” values can then be combined with separately stored reward values for each state in order to support action selection at decision time. In other words, although SR requires more memory than regular MF RL algorithms like QL, SR does not need to compute future state values anew at every step, which MB RL does.

There are several lines of research supporting biological correlates for SR. It can represent activation patterns in prefrontal and hippocampal areas such as those measured by fMRI [175] – some of which correlate with place and grid cells [248] – which bridges the roles of OFC and hippocampus for task structure and state representations [288, 293, 303]. Together with support for SR in human behavioural tasks [177] and computational studies showing that SR (depending on its specific implementation) can explain both MF and MB RL phenomena [219], some authors have even suggested SR can help frame the RL dopaminergic system as one about prediction errors in general and not just reward prediction errors [88, 95, 176].

The specific implementation of SR we will focus on here is based on [219] and combines SR with temporal difference learning (TD) and

is thus called SRTD. However, we will first introduce the SR in more general mathematical terms, based on [88, 95, 177], to then focus on the specific version from [219].

Mathematically, SR has two main components in the matrix M and vector R , where M holds the state transition approximation (future state occupancy values), and R stores the immediate reward expected upon encountering state s . Thus, R is a vector with the same length as the number of states.

More formally, [66] shows that state values in the SR can be calculated as the sum of rewards for all states s' following the current state s :

$$V(s_t) = \sum_{s'} M(s_t, s') R(s') \quad 5.3$$

where it should be noted that s' includes the current state, as it is possible (depending on the task) to stay in the current state, and M is the SR, the expected discounted future state occupancy [88, 96, 218]:

$$M(s_t, s') = \mathbb{E} \left[\sum_{k=0}^{\infty} \gamma^k \mathbb{I}(s_{t+k} = s') \right] \quad 5.4$$

where \mathbb{E} is expected value, γ a future discount parameter and $\mathbb{I}(\cdot) = 1$ if its argument is true, and 0 otherwise. The \mathbb{I} is sometimes called the Kronecker delta [175], but also has other names like “one-hot encoding” depending on the field⁴¹.

Based on Equation 5.4 and the Bellman equation, a temporal difference error for M can then be derived [88, 96, 219]:

$$\Delta M(s_t, s') \propto \delta_t^M(s') = \mathbb{I}(s_t = s') + \gamma \hat{M}(s_{t+1}, s') - \hat{M}(s_t, s') \quad 5.5$$

where \hat{M} denotes the approximation of M . We can now see that the right-hand side of Equation 5.5 is indeed very similar to the reward prediction error in, for example, Equation 5.1. In other words, through experience, an SR-based agent can learn to approximate the state transition function, just like a QL3 agent can learn to approximate state-action reward values.

⁴¹ <https://stats.stackexchange.com/q/308916>

As noted, the matrix M is the SR itself and represents the state space of size S with S rows and S columns. Each row represents the state r and each column value M_{rc} is the probability⁴² of being in that state in the future – how many times we can expect to be in the state c , starting from state r . Conversely, each column represents the state c and each row value M_{rc} is the probability of that state having been visited in the past. In other words, rows represent the future and columns represent the past.

As an example, imagine a small gridworld maze of size 3×3 rows and columns, which thus have 9 states. M would then be of size 9×9 , as seen in Table 5.1. It is possible to initialise M as an identity matrix as done here, meaning there is some initial belief that the agent can transition into the same state. For example, in a gridworld maze where the agent comes back to the same state if it walks into a wall. But the diagonal can also be initialised to zero and this transition then also needs to be learned [176].

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 5 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 7 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

Table 5.1 SR example using a gridworld maze. Left: 3×3 gridworld where each state is numbered 1–9. Right: Corresponding SR representation before starting to explore the maze. Rows and columns are numbered as per their state in the maze. Each row represents the future state occupancy values, if one is currently in state=row. Similarly, columns represent the past.

⁴² It is technically not a probability but a value for expected future state occupancy – how many times we can expect to be in the future state c when starting in state r . But we will use probability and likelihood as shorthand to make the text easier to read.

We now turn to the specific implementation by [219], which we call SRTD. We still have the matrix M , but the vector R has been replaced with a weights vector w , which like R has the same length as the number of states. In principle this SRTD version works the same as SR described above, meaning if we replace R with w in Equation 5.3 we get the values for each state. By extension, we can get the values for all states through matrix multiplication:

$$V = M \times w \quad 5.6$$

Indeed, [219] state that the above equation will be correct when the items of w correspond to each state's one-step reward. The difference to R then, is that w here works as a linear function approximation and [219] show (in supplementary materials) that this approach yields improved performance early in training, before M has enough experience to have converged.

More specifically, M and w are learned in parallel, where on each transition from state s to s' , as indicated by the colon statement, every element of row s in M is updated as:

$$M_{t+1}(s, :) = M_t(s, :) + \alpha_{SR}[\mathbf{1}_s + \gamma M_t(s', :) - M_t(s, :)] \quad 5.7$$

where $\mathbf{1}_s$ is a vector of all zeros except in position s where it is 1, α_{SR} is the learning rate, γ the future discount and s' is the next state.

The second component w is then updated using the new M , together with the reward prediction error. All items i of w are updated according to:

$$w_{t+1}(i) = w_t(i) + \alpha_w * RPE * M_t(s, i) \quad 5.8$$

where RPE is the reward prediction error as in regular QL:

$$RPE = R(s, a) + \gamma V_t(s') - V_t(s) \quad 5.9$$

where $R(s, a)$ is the reward for taking action a in state s . Note that M and w share discount parameter γ but have separate learning rates α .

Thanks to M , SR(TD) has some of the features of MB algorithms in that there is information about state transition probabilities. The multiplication of M and w (Equation 5.6) is one operation, which is cheap computationally, and thus by influencing V grants aspects of MF algorithms. We do not have to calculate all future state values

based on the state transition function, as we must with MB RL algorithms.

5.4.1 LATENT EXPLORATION OF GRIDWORLD MAZES

In their recent paper [219] demonstrates how SR can be implemented to exhibit aspects of MB and MF behaviour. We shall here focus on their SRTD implementation, and the task used (based on [269]), latent learning in a gridworld maze as seen in Figure 5.39. The maze consists of 10x10 squares, where some are walls (black squares in figure) and some are corridors (grey squares in figure). Additionally, and further described below, it has an additional state “outside” the maze.

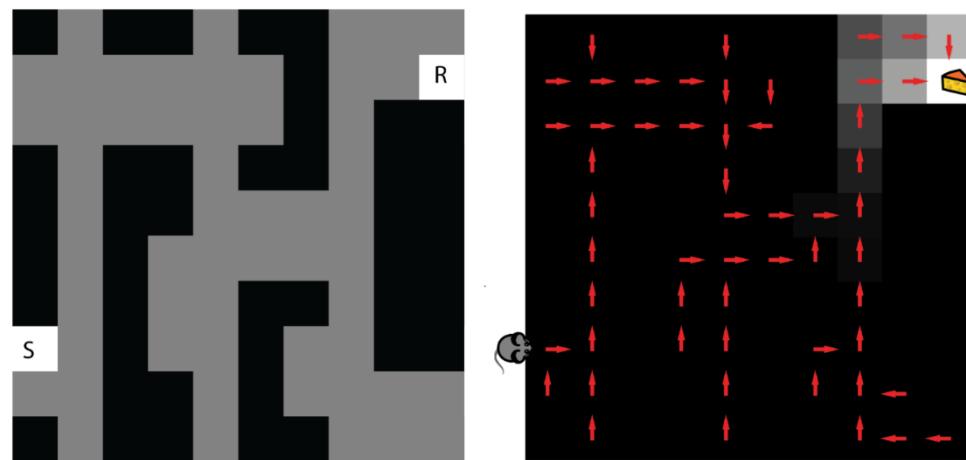


Figure 5.39 Gridworld maze. Left: Grey areas are traversable, and the black squares are walls. In the latent exploration phase, the agent is put in start position S and gets to explore the maze until it reaches reward position R. When square R is reached, the agent enters a consummation state and is then “picked up” and put back in start position S. In the reward phase, the agent is placed twenty times directly on R, which allows it to enter the consummatory state and receive a reward. Right: Test phase, where the M learned during exploration phase and w learned during reward phase are combined into state values. Red arrows point towards the state with highest value. Adapted from [219]

During the latent learning phase of the experiment, no reward is present, and the agent is allowed 25000 random exploration steps⁴³ in the maze as seen to the left in Figure 5.39. The agent starts this

⁴³ Of course, in a real-world task with actual rats, this number of steps may or may not reflect reality. Even if we divide a rat labyrinth into discrete squares, it is questionable if the rat would require so many steps. Lowering the number of steps in the code for the exploration phase decreases stability of results. But this is a general problem when relating RL to animal behaviour, as discussed in the background chapter.

exploration phase in the square marked S and whenever square R is reached, the agent only has the option to enter a separate consummation state⁴⁴ (not shown in the figure as it is technically outside of the maze, but see Figure 5.40) before it is then “picked up” and put back in start position S. The agent may thus journey from S to R multiple times until the maximum 25000 steps have been used up and the exploration phase ends. Only M (Equation 5.7) will change in value during this phase, since w (Equation 5.8) requires a reward to be present for the RPE (Equation 5.9) to differ from zero.

In the next phase, a reward is introduced to the consummation state. The agent is now placed directly on the square marked R in Figure 5.39, from which it moves into the consummatory state where it consumes the reward. In other words, the reward is not directly associated with the square R inside the maze. When the agent is placed at square/state R, there are no other actions available in this state than to go into the consummatory state, and so the agent steps from R to the consummation state and receives a reward. This is repeated twenty times, and since a reward is now present, w can now change when updated.

The test phase consists of placing the agent in the starting state S (corresponding to the square inside the maze to the right of the mouse in right-hand illustration of Figure 5.39) and calculating the values for each state in the maze (Equation 5.6). In other words, neither M nor w are updated here, instead [219] demonstrates the resulting behavioural policy with drawing arrows on each state. The arrows are pointing towards the neighbouring state with the highest value, (V , Equation 5.6) as seen in the right-hand side of Figure 5.39. Thus, the arrows show the path that a fully greedy agent would take. In other words, the agent can navigate to the reward location R through the combination of its state transition approximation M , learned during the exploration phase, and its experience of rewards stored in w from the reward phase.

This contrasts with a regular MF model like QL₃, which would require to explore the maze from the start multiple times with the reward present. This because QL₃ only has state-values, and they can

⁴⁴ It is not technically a consummation state in the exploration phase, as there is nothing to consume. But we call it consummation state in both phases for consistency.

be updated only if there is a reward present. We saw this in Figure 5.15, where the rewarded state was updated the first time the agent encountered it, then the states next to the reward state could be updated the second episode, and so on.

5.4.1.1 The Hotel California problem

As described above, during the latent exploration phase, the agent is put back into the starting state S after it has reached the maze position R. More specifically, in state R there is only one action allowed for the agent, which is to move into the consummation state, which is the state where the reward is later placed during the reward phase. Just like in the old song Hotel California, where you can check-out but never leave, the agent cannot escape R when it has arrived there. It can only “check-out” the consummation state.

In Figure 5.40, right-hand side, the original task illustration from Figure 5.39 has been altered to illustrate that the consummatory state is outside of the maze and only reached from the Hotel California (HC) state. In other words, there are in fact additional states in this maze than the 10x10 grid (also matching better how [219] simulated). The Hotel California effect is crucial, as it causes the state values V to cleanly end up pointing towards R. Without check-in, the simulation doesn't work the same.

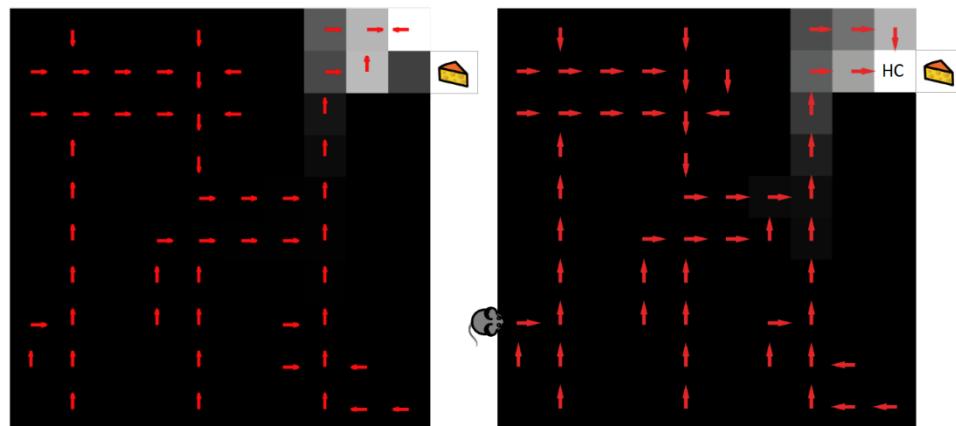


Figure 5.40 Illustration of the Hotel California problem. Left: Resulting policy after reward phase in modified latent learning task where agent is free to leave the Reward state during exploration phase. Right: Original version where agent is picked up from Reward state (marked HC) and put back to starting state (right of mouse) during exploration phase. Adapted from [219]

To demonstrate the importance of the specifics of the implementation used by [219], we modified their provided code and tested the case of letting the SRTD agent explore for 25000 steps without picking the agent up when it reaches R and return it to S. Instead, we let the agent move into and out of R just like all other states. We call this “free” latent exploration. Then we did the same reward phase procedure as in the original experiment, where we added the reward in R (or, rather, the consummatory state) and put the agent down into R twenty times. To be clear, in this phase the agent can only move to the consummatory state, like the original version. When we then calculate the policy by multiplying M and w we get the result seen on the left-hand side of Figure 5.40.

As can be seen, the SRTD algorithm can no longer solve this task as well as the original version with pickups during exploration⁴⁵. Why? What happens is that since we multiply M with w , and M now has quite high probability of moving from state R (the dark grey square without arrow, to the left of the cheese) to the neighbouring states, we get the curious case of those neighbouring states having a higher average return than the actual goal state. Note that state R does have value, it is dark grey and not black, but it has lower value than its neighbours.

We can understand this through Equations 5.6-5.9. In the HC (original) versions exploration phase, when the agent reaches state R, it can move only into the state outside the maze, state CS (consummation state). That means $M(R, \neg CS) = 0$, so in the reward phase when w is updated, $w(\neg R) = 0$, $w(R) \neq 0$. The consequence is that when V is calculated by combining M and w , we do get values for all states because $M(\neg R, :) \neq 0$, and they all point towards state R since that is the only state with a w value that is not zero.

In the free exploration version, because the agent is free to move in any direction from state R, $M(R, :) \neq 0$, so all w values are updated in the reward phase. Since the entire matrix M contributes to the state values V , the states surrounding state R also contribute. Notice that values in w cannot decrease, so with the repeated reward trials, the neighbours of R increase in value and can get higher values than state R itself.

⁴⁵ It will, of course, come very close to the reward so will still be able to learn how to find the reward quicker than a naïve agent

5.4.1.2 Tolman approves

The paper [219] cites Tolman's classic paper [269] for their latent exploration task. Experiments mentioned there were run similarly to the HC version of latent exploration, namely that rats were picked up when they reach the goal box, even when there was no reward. We will disregard that Toman's mazes were much simpler than the gridworld presented above and say only there is an argument to be made that [219] is simulating only a specific type of experiments; they do not claim to have done anything else. However, there have been a few experiments done since 1948, and even without looking at those in detail, it is not far-fetched to imagine that a rat may be able to explore a maze in such a way as the one used above, the free exploration version, and yet be able to quickly find its way to the goal box after having been exposed to rewards there. It seems curious to not have fully explored this case.

Furthermore, we do not have to move past the 1940s to find just such an experiment. Seward [237] let rats freely explore a maze for 30 minutes, including the goal box which had an empty food bowl, and they were free to go back from the goal box through the maze during this time. Seward then introduced a reward into the goal box and presented this to rats by lowering them down into the goal box through the roof. At this point the goal box was closed to the rest of the maze, to give the rats some secluded feeding time. Worth mentioning is that there were two goal boxes in each maze, and they differentiated from each other as well as the rest of the maze by different floor colours and/or materials. When later tested from the start of the maze, 87.5% of the rats that got to freely explore the maze went directly for the food. Directly meaning they did not go down blind alleys or alleys leading to another goal box.

The experiment just described is thus similar to our free exploration, shown in the left part of Figure 5.40. In this version of the task, the rat would quickly arrive at squares neighbouring the reward, from where finding the reward is easy. Perhaps this "fuzzy" recollection of the reward location describes animal navigation and cognitive maps better than the more exact HC version. Speculations aside, and more importantly, the free exploration version and subsequent test still proves the point that SRTD can account for latent learning in mazes. This because an SRTD agent without exploration,

and only exposed to the reward phase of the task (where the rat is placed directly in the reward square), would only have resulting state values V for the state R and its two neighbours. The rest of the maze has not been visited, so M is zero for most squares. This also makes sense, as a rat that has only seen a reward box and is then tested from start, will have no idea there is a connection between the reward square and the maze it suddenly finds itself in.

It is unfortunate these scenarios are not fully explored by [219], but to be fair, this does not take away from their grander points about what type of tasks SRTD is capable of. It is understandable that tricks like these are needed to get algorithms to work nicely, and sometimes to work at all, and those implementation details do not necessarily matter. We would have liked to see this issue at least mentioned however, preferably discussed, since SRTD works in our free exploration, although not as cleanly.

5.4.2 SHAPETASK AS A MAZE

Our interest in SR lies in applying it to Shapetask, in order to investigate how well SR might explain human behaviour in this task, if at all. Studies on SR, like the one described above, often focus on spatial knowledge [177, 219] but other research points to cognitive maps being applicable to more abstract knowledge, with links to SR [14, 21, 89, 248]. Therefore, Shapetask may prove useful in adding to this literature.

We approached the application of SRTD to Shapetask by extending the code from [219] to add Shapetask to the repertoire of tasks simulated with variants of their code. This posed the interesting and illuminating question of how to transform Shapetask into a maze problem?

No matter what kind of maze we construct – in our heads or in the code we will use for simulations of this task – we impose and assume structure that don't necessarily match what experiment participants do. What we do know is that our human participants were instructed to “find the pattern” which grants us some assumptions for the implementation. Therefore, translating Shapetask into maze form is an excellent example of the overarching problem we are trying to investigate; state creation and task structure and how these are closely intertwined.

Shapetask is not fully deterministic in its state transitions, in any of its versions (rather as if the maze had a non-zero chance of being changed between trials). We also have the situation that regardless of chosen action, the next state will follow the predetermined sequence and not rely on what action was chosen. This should not pose a problem directly, since there is nothing inherently stopping the matrix M in SRTD from having multiple states being possible to follow a certain previous state. But it may be the case that more experience (more trials) is needed for M to handle a non-deterministic task compared with a deterministic one.

The most straightforward approach to create a maze requires at least nine states for each shape and position, as seen in Table 5.2 (left). If the first shape in a bag-of-bags is square, then we have a sequence of states for that bag as 2, 5, 8. In the BOB version of Shapetask, we then go to either state 1 or 3. An example sequence for one bag-of-bags can thus be 2, 5, 8, 1, 4, 7, 3, 6, 9. The maze is somewhat magical, in the sense that if we imagine it as a physical maze with rooms connected by doors – the last door for each shape (7, 8 or 9) will teleport to either 1, 2 or 3.

| | 1 | 2 | 3 | | 1 | 2 | 3 | |
|---|---|---|---|--|---|----|----|----|
| 1 | 1 | 4 | 7 | | 1 | 1 | 2 | |
| 2 | 2 | 5 | 8 | | 2 | 4 | 5 | |
| 3 | 3 | 6 | 9 | | 3 | 7 | 8 | |
| | | | | | 4 | 10 | 11 | 12 |
| | | | | | 5 | 13 | 14 | 15 |
| | | | | | 6 | 16 | 17 | 18 |
| | | | | | 7 | 19 | 20 | 21 |
| | | | | | 8 | 22 | 23 | 24 |
| | | | | | 9 | 25 | 26 | 27 |

Table 5.2 Overview of Shapetask as a maze. Left: Each shape and position combination creates a uniquely numbered combined state. Right: Numbered rows corresponds to the numbered state in the table to the left. Values 1-27 are the numbered compound states interpreted as being in state=row, having selected shape=column on the previous trial.

In a regular maze like that described in the previous section introducing SRTD, we use state values and get action-values by looking at the values of neighbouring states. Through the Bellman

equation we can convert state-action values $Q(s, a)$ to state values $V(s)$ [258], but in order to understand how to create structures that work with the M , w and V of SRTD we have to realise the integrated nature of states and actions.

Recall our SEQL3 model, which uses nine states like in Table 5.2 (left), and for each state there are three action values and thus 27 values in total. In SEQL3, those action values mean “the value of selecting action a , in state s ”. By reframing the same value structure as “the value of being in state s , having chosen action a ”, it means we can get action values by checking the neighbours of the current state, and select actions based on this context.

For example, consider Table 5.2 (right). Each row corresponds to the state number in the maze in Table 5.2 (left). If the current state is position two in the square bag – state 5 in the maze – then depending on what action was picked in the last trial – triangle, square, or circle – we are in “compound states” 13, 14 or 15, respectively, as seen on row 5 of Table 5.2 (right). Here, being in compound state 13 means seeing the second square of the bag (i.e., sequential position 2), having picked triangle as the prediction on the previous trial. Compound states 14 and 15 also mean we are seeing the second square of the bag, but we picked square or circle, respectively, on the previous trial. In other words, the stimulus is the same for all three compound states, but the context is different.

This state structure as described allows us to reason about state transitions in Shapetask and use it as our M matrix, which becomes 27×27 in size, with weights w of size 27. The resulting $V = M \times w$ thus also becomes 27 values and is very similar – in fact identical – to the structure of SEQL3.

5.4.3 PLAYING SHAPETASK WITH SRTD

To simulate SRTD behaviour in the BOB version with 99 trials of Shapetask, we started by adding Shapetask as a playable task with minimal changes to the model code from [219]. This step was taken to make it easier to confirm our approach worked correctly. We then reimplemented the SRTD algorithm in Python (and confirmed it produced identical behavioural profile), which is the code used in the presented results below. It is worth noting that in both the original code and our implementation, M is updated in a different form than

Equation 5.7 (allowing cleaner separation of s and s'). The form used in code can be derived from the just mentioned equation thus:

$$\begin{aligned} M_{t+1}(s, :) &= M_t(s, :) + \alpha_{SR}[\mathbf{1}' + \gamma M_t(s', :) - M_t(s, :)] \\ &= M_t(s, :) + \alpha_{SR} \mathbf{1}' + \alpha_{SR}\gamma M_t(s', :) - \alpha_{SR}M_t(s, :) \\ &= (1 - \alpha_{SR})M_t(s, :) + \alpha_{SR} \mathbf{1}' + \alpha_{SR}\gamma M_t(s', :) \\ &= (1 - \alpha_{SR})M_t(s, :) + \alpha_{SR}[\mathbf{1}' + \gamma M_t(s', :)] \end{aligned}$$

For the Python implementation, we additionally changed to SoftMax from the original code's use of ϵ -greedy for action selection. This was done so action selection is consistent across all our algorithms. As with previous algorithms, we generate 1000 random combinations of parameter values and for each such combination we simulate 100 subjects. Parameter values were drawn as:

$$\alpha_{SR} \sim U(0, 1), \quad \alpha_w \sim U(0, 1), \quad \beta \sim U(0, 20), \quad \gamma \sim U(0, 1)$$

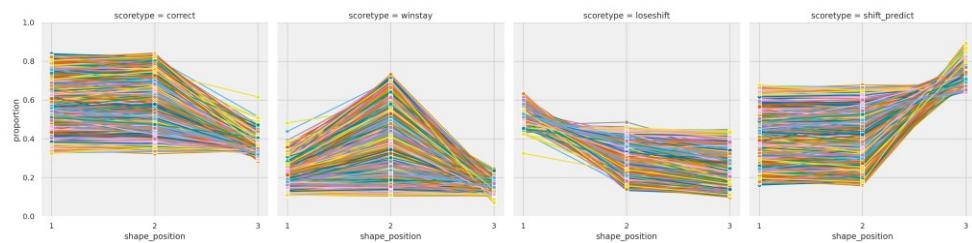


Figure 5.41 Overall SRTD behaviour in Shapetask. Proportion of choices (y-axis) averaged for each simset (separated by lines) and shape position (x-axis) and score type (columns). From left to right: Correct, Win-stay, Lose-shift, Shift-predict.

In Figure 5.41 the averaged results for each simset are shown, and it looks like we have a general pattern of winners without WSLS cases. We confirm this by grouping the simsets as we have done above for previous algorithms (defined in section 5.1.5) and we do indeed have only the winners and others groups here, as seen in Figure 5.42.

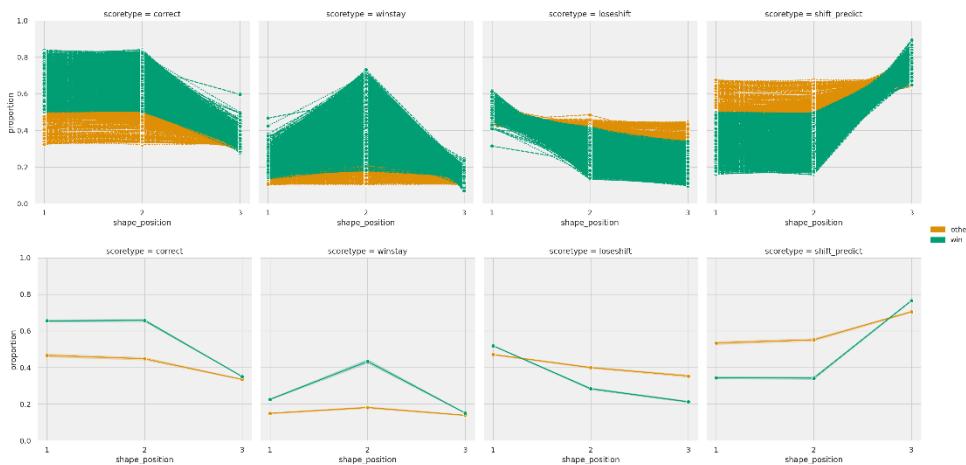


Figure 5.42 SRTD behaviour grouped by winners and others. Proportion of choices (y-axis) for each shape position (x-axis) and score type (columns). Group colours as per legend. Top: All simsets shown as individual lines. Bottom: Averages for each group. From left to right: Correct, Win-stay, Lose-shift, Shift-predict.

What can also be gleaned from Figure 5.42 is that there is some variation in the winners group. Therefore, before looking at parameter value correlations for the groups, we find the higher performing winners, using conditions as high being >0.8 Correct score for shape positions one and two, and >0.8 Shift-predict score in shape position three. To allow for potential slower learning in some cases, we use the last 18 trials for these groupings. In the specific simulation run shown in these figures there were 783 winner simsets, of which 231 met the criteria of “high winners”.

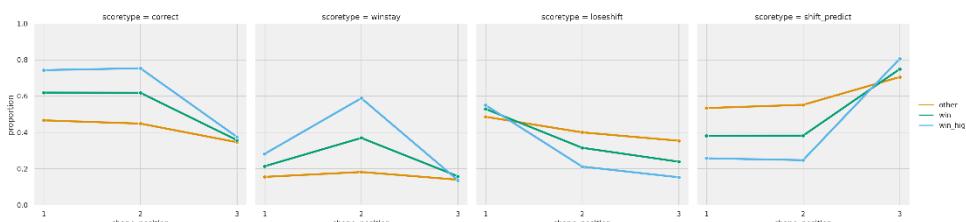


Figure 5.43 SRTD behaviour grouped by high winners, winners and others. Proportion of choices (y-axis) for each shape position (x-axis) and score type (columns). Group colours as per legend. From left to right: Correct, Win-stay, Lose-shift, Shift-predict.

In Figure 5.43 we can see that the high winner’s group indeed show a more distinct pattern across the score types and shape positions. Thanks to this subgroup, we can see in the parameter space plot in Figure 5.44 that both winners and others share most of the parameter space, but the high winners distinguish themselves by having low γ values. Because SRTD has information about shape position it is likely

not hugely beneficial with a high γ value, i.e., looking farther into the future. Instead, one-step rewards for each shape position are sufficient and often – as seen – better for performance.

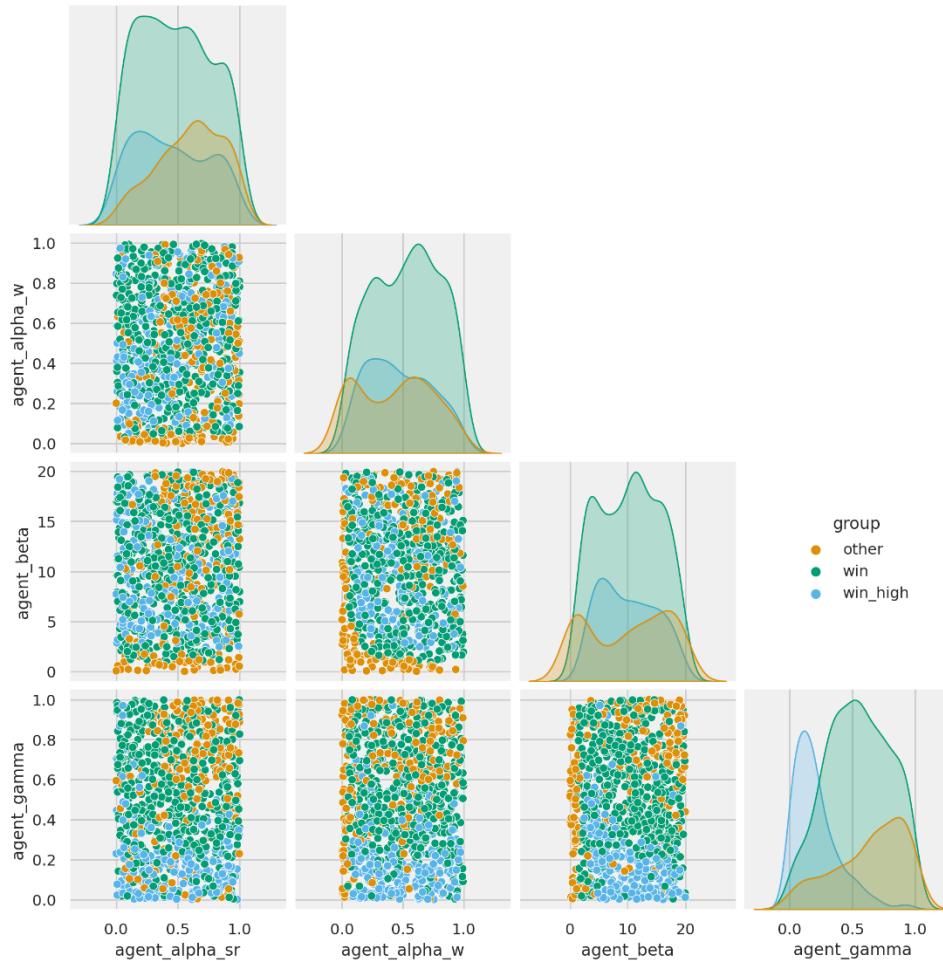


Figure 5.44 SRTD parameter values for each group. Diagonal distribution plots refer to the parameter value labelled at the bottom of each column. Group colours as per the legend. The scatter plots show the combined parameter space for the two parameters labelled on the row and column.

5.4.4 DISCUSSION OF SRTD

SRTD can solve the Shapetask well, and this within the 99 trials. The range of results seen look promising for potentially fitting human winners. This performance is perhaps not surprising, seeing as the state representation structure is in principle identical to that of SEQL3. This similarity is interesting in that it confirms the utility of SRTD for state representation and task structure, but also that perhaps some humans are able to chunk and generalise into more efficient structures like SEQL3.

The alternate framing of Shapetask as a maze together with the state transition matrix M , provides the convenient side effect in that it quite naturally allows us to speak of contexts. If we are on, say, position 1 of the circle bag, then no matter what action we pick – what shape we predict will appear next – the state we arrive in is the same, but the context will depend on our choice. This is similar to arguments made for episodic RL [95, 97], suggested to account for cases where the Markov property may not be enough, where not only the values themselves matter but also the recent history of steps.

The downside of SRTD is that it is unlikely to fit the human subjects using WSLS behaviour. For this reason, HRL looks like the overall more promising approach, if one is looking for a model that can account for as many human subjects as possible.

To summarise, the results shown for SRTD adds to the growing literature showing how the concept of cognitive maps is applicable to more abstract reasoning and knowledge, and not only explicitly spatial (navigation) problems.

5.5 CONTROL GROUP MODELS

In addition to the already mentioned models, we would also like models we can use as baseline control. One such is a model making random choices, allowing for some bias for one or more actions. We call this model RandomBias, and it has two parameters, $bias_1$ and $bias_2$. The probability this model picks any of the three actions can be described as:

$$\begin{aligned} p_t(a_1) &= bias_1, p_t(a_2) = bias_2, p_t(a_3) \\ &= 1 - (bias_1 + bias_2) \end{aligned} \quad 5.10$$

We generate 1000 random parameter combinations with 100 subjects for each combination, with parameter values drawn as:

$$bias_1 \sim U(0, 1), \quad bias_2 \sim U(0, 1 - bias_1)$$

In Figure 5.45 we visualise the results as the average score for each shape position and simset. Because we pick actions at random and have three actions and stimuli, we see that overall, there is roughly $1/3$ correct score throughout the experiment for each shape position. Similarly for Shift-predict score we see that roughly $2/3$ of the time a

shift is predicted for each shape position. None of these simsets are grouped as WSLS or winners and thus group plots are not shown.

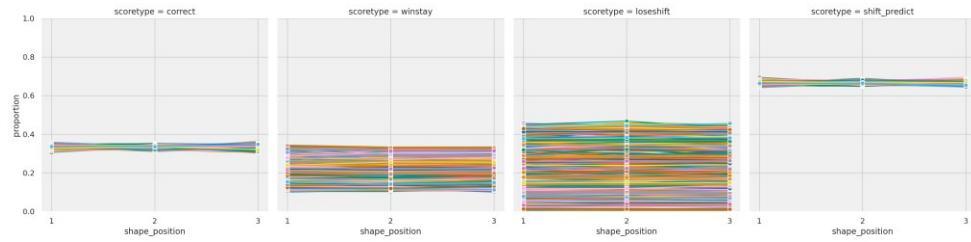


Figure 5.45 RandomBias behaviour in Shapetask. Proportion of choices (y-axis) for each shape position (x-axis) and score type (columns). Individual lines are the averages for each simset (parameter value combination). From left to right: Correct, Win-stay, Lose-shift, Shift-predict.

5.6 MODEL SELECTION FOR SHAPETASK

Having introduced the models of interest for Shapetask, we now turn to fitting those models to data. Of the five models – QL₃, SSQL₃, HRL, SRTD and RandomBias – we already know that only one of them can possibly explain the full variety of human behaviour at group level (at least for the wide range of parameter combinations that we used in our simulations). That model is HRL, because it can show both WSLS and winner behaviour.

Unfortunately, HRL cannot be fitted using likelihood-based methods. The main reason is that we do not have access to the taskset selections in our observed behavioural data. All we have are the observed actions, rewards and stimuli. Even in the HRL-22 version this means the only way to select tasksets in the likelihood function is using SoftMax for taskset selection. But this causes the likelihood to become non-deterministic and MLE fitting mostly returns the same values used to initialise the fitting function (random guesses within the boundaries). Another possibility would be to generate all possible taskset selections, and find the best fit based on the data we do have. However, even for HRL-22 with two tasksets, that would be 2^{99} possible combinations for 99 trials of Shapetask and thus computationally intractable.

Because of this issue with HRL, we will first exclude it from analysis to investigate how well our existing methods for model

fitting work for Shapetask. We then present an alternate method with which HRL can be included and show model recovery performance.

5.6.1 MODEL SELECTION PERFORMANCE WITH MLE FITTING

We use the same simulation functions and parameter ranges as described above (for each separate algorithm) to generate 200 agents for each of the five algorithms, resulting in 1000 simulated subjects in total. In this step we do include HRL, more specifically HRL-22. We then fit all models except HRL to all 1000 subjects. Even though HRL is excluded in the fitting step, we will get an idea of what other algorithms are the common best fits for the HRL subjects and use this information in subsequent analyses.

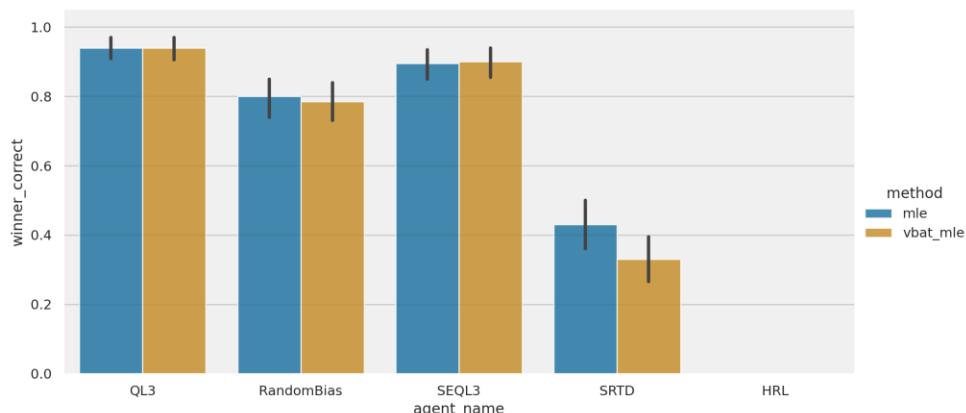


Figure 5.46 Overall model selection performance for Shapetask. Proportion of correctly identified subjects on y-axis for each simulated model on x-axis. Coloured bars indicate fitting method as per legend.

In Figure 5.46 we have plotted the overall results for the simulation and fitting process. First off, HRL cannot be correctly identified since we do not fit using that model which is why there is no result for this agent. It is included in the plot anyway, for easier comparison with subsequent plots. Further, we can see that for QL3, RandomBias and SEQL3 the performance is quite good, with >80% correctly identified cases. For the SRTD subjects the result does not look as good, it does not reach even half correctly identified subjects. However, this result makes sense by looking at Figure 5.47 where we have plotted all model selections. There we see that in the majority of cases, the model selected instead of SRTD is SEQL3. As we noted above in the behavioural studies, SEQL3 is very similar in behaviour and structure to SRTD. Because the BIC measure penalises models with more

parameters, SEQL3 is often selected instead. It should be noted that due to randomness, the exact proportions of SRTD subjects identified as SEQL3/SRTD may vary.

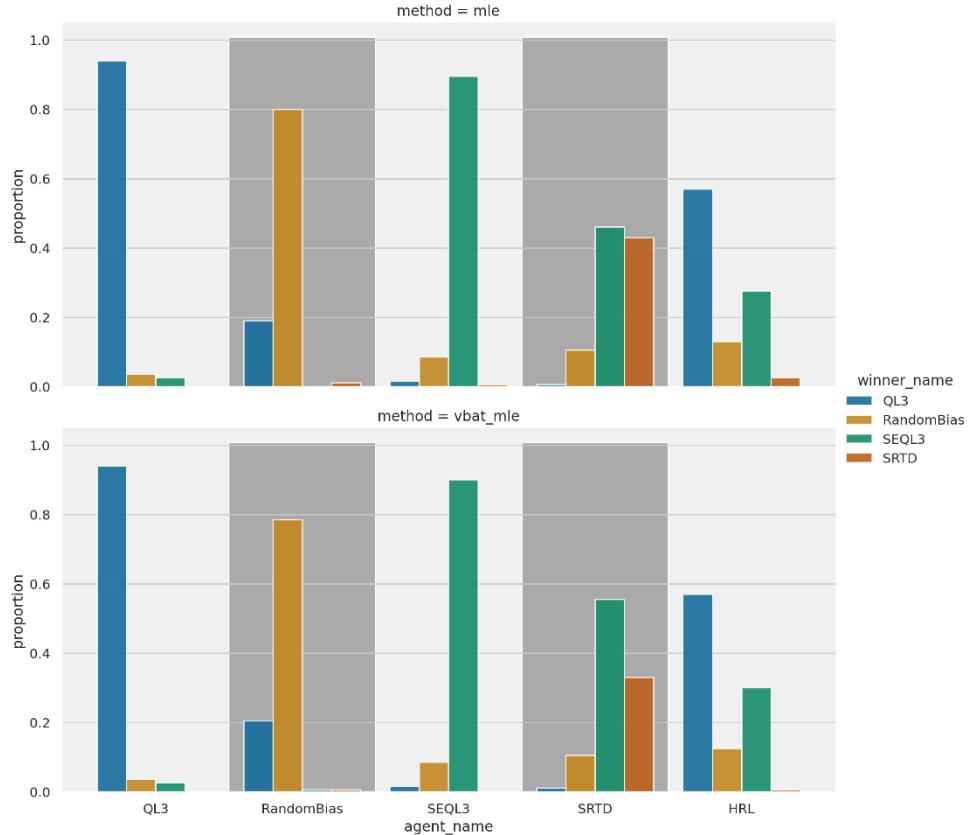


Figure 5.47 Model selection performance in Shapetask. Proportion of identified cases (y-axis) for each simulated algorithm (x-axis). The fitted model used as per colours in legend. Shaded/non-shaded areas are only for readability. Top: MLE/BIC method. Bottom: VBAT method.

For the HRL subjects in Figure 5.47, results are in line with expectations, given that the HRL model could not be selected. That is, the two main models being selected are QL3 and SEQL3. As we saw above in the behavioural studies, QL3 is capable only of WSLS behaviour, while SEQL3 is capable only of Winner behaviour. Since HRL is capable of both these behaviours it makes sense these two models are the most selected ones.

5.6.2 MODEL SELECTION WITH APPROXIMATE BAYESIAN COMPUTATION

In order to fit models with intractable likelihoods such as HRL, we can use an alternate family of methods called Approximate Bayesian Computation (ABC) [255, 270]. ABC methods are fairly common in

fields such as systems biology [156] and astronomy [269, 273] but not widely used in research on learning and decision making (but see [251, 274] for related work). ABC can also be called “likelihood free” or “simulation based”. The latter term is closest to how ABC works, as in essence it entails simulating the model and comparing the results with observations.

The simplest form of ABC is rejection sampling [270, 274]. We have some observed data D , and we would like to find the model M parameters $M(\theta)$ that most likely produced D . We draw a random value θ^* from a suitable probability distribution, simulate $M(\theta^*)$ and get some data D^* . The observed and simulated data are then compared using a distance function d , and tolerance $\varepsilon \geq 0$. If $d(D, D^*) < \varepsilon$ we accept the drawn θ^* , otherwise we reject it.

In other words, the principle of ABC is in statistical terms a Monte Carlo approach – simulate random cases enough times and the approximation will be close to the true value or distribution. Of course, this builds on carefully selecting the distance function and tolerance. Additionally, if the prior distribution we use to draw samples of θ from is very different from the true posterior, we may have to simulate millions of times before approaching decent convergence. This makes ABC rejection sampling – and ABC methods in general – very computationally inefficient. We will come back to a brief comparison between MLE and ABC below.

Because of this computational inefficiency there are more advanced methods such as Markov Chain Monte Carlo (MCMC, also used in Bayesian inference with likelihoods as described in chapter three) and Sequential Monte Carlo (SMC) [270, 274]. We will here focus on the latter, ABC-SMC, as it is the one used in the Python package pyABC [135, 224] which we use for our analyses.

The advantage of ABC-SMC is that instead of sampling values of θ one at a time, it generates many values at once, called a particle population. Through importance sampling [270], new particle populations are generated from accepted particles in the previous generation. This allows the algorithm to gradually reduce the tolerance ε for each generation while still accepting a sufficient number of particles (parameter values) for the resulting posterior distribution [135, 270, 274].

Several Python packages exist that implement ABC-SMC, such as astroABC [123], ABC-SysBio [156] and ELFI [160]. We settled on pyABC [135, 224] as it has a straight-forward interface we could integrate with our existing simulation functions, as well as automatic tolerance tuning and built-in support for model selection.

Model selection in ABC works by comparing the marginal likelihoods of the posterior distributions [270]. In other words, if we generate a distribution O of all possible data outcomes for model $M(\theta)$, we can get the probability of a specific outcome o as $p(o|M)$. This aspect is used by ABC-SMC to approximate the marginal posterior distributions $P(M_i|D)$ where D is some observed data and i the model index. Given these posterior distributions for models we can compare models with the Bayes Factor (BF), which was introduced in an earlier chapter. Below we refer to BF evidence intervals as presented in Table 3.1.

An important note is that the way model selection works in pyABC (based on [270]) means that models with higher number of parameters are penalised, because the more parameter dimensions, the smaller the chance that parameters are accepted.

Knowing the basics of ABC, we should mention that in [269, 273] – which our HRL implementation is based on – the authors used a variation of simulation based model fitting. They mention ABC methods were unsuitable for their use case, though what they did could be called a group-based ABC approach. They simulated the entire group of subjects thousands of times and selected those simulations that best described the entire group, based on means and standard deviations for their chosen distance metric.

We have chosen not to apply this approach here because we already know, as discussed above, that HRL is the only model that could possibly describe all or most humans in Shapetask. The task in [73] apparently did not result in behaviour as heterogenous as that in Shapetask, where we have multiple distinct kinds of behaviour. Furthermore, as also discussed in previous chapters, we are interested in potential individual differences in what model best describes behaviour.

5.6.3 IMPLEMENTING ABC FOR SHAPETASK

As mentioned above, ABC requires a distance function to compare observed data with simulated proposals. Since our data consists of sequences of actions, rewards and stimuli, where subsequent entries depend on former data points, it is infeasible to compare the outcomes directly. Instead, we can use summary statistics for the distance function [270, 274]. Luckily, we already have suitable summary statistics we know well from our behavioural studies above, namely the four score types Correct, Win-stay, Lose-shift and Shift predict. For a simulated outcome of actions, rewards and stimuli we calculate the mean of the four score types for each shape sequential position and thus get $4 * 3 = 12$ summary statistic values. If we treat these values as coordinates in a 12-dimensional space, we can calculate the Euclidean distance between the observed data from a subject and the simulated data from models.

The downside of using summary statistics is that we lose trial-to-trial dynamics of the data such as learning rates. For example, if one algorithm reaches stable performance on trial 80 and another at trial 40 yet may have the same mean summary statistic across all trials. The consequence is that it will be more difficult to distinguish between, for example, SSQL3, SRTD and HRL as they are all capable of producing winners. This could in theory be alleviated by adding complexity to the distance function, like also calculating learning curves for each score type. But the more complex the distance function, the more computationally heavy the fitting process becomes, so again our decision not to do this is a trade-off.

5.6.4 ABC MODEL SELECTION PERFORMANCE

To investigate model selection performance with ABC-SMC, we simulate and fit in two steps. First, we do simulate and fit like we did for MLE above. That is, we simulate all five models but exclude HRL from the fitting process. Second, we simulate and fit using all five models. This approach allows us to compare our results with MLE in the first step and thus get an apples-to-apples contrast of performance. The insights thus gained can then be used to make better interpretations of the second step.

We simulate 20 subjects per model (see below for why such a low number) and fit all models to each subject. As we have five models to simulate in both steps, this gives us $5 * 20 = 100$ subjects in total. Because

SoftMax β parameter values close to zero causes the algorithms that use it to behave more randomly, we here made a slight adjustment to parameter ranges used, compared to MLE. All β for simulations and fits were drawn from $\beta \sim U(1, 20)$. This still provides a wide range of behaviours, and is the same range for β values used in e.g. [190, 231].

As mentioned above, ABC-SMC evolves particle generations with each generation being a better approximation of the true posterior. For this performance check, we use five generations as the maximum. With these settings, fitting for all 100 subjects takes around 2 hours 40 minutes, meaning 1.6 minutes per subject. pyABC takes full advantage of multicore systems, and the simulation and fitting process was done on a laptop⁴⁶ released in 2021. In comparison, on the same laptop, the above MLE simulation and fitting of 1000 subjects took less than one minute. This is why a relatively small number of simulated subjects was used here.

It is possible the above-mentioned alternatives to pyABC are faster for this method, but none would be as fast as MLE. As was discussed in previous chapters, then with regards to Bayesian MCMC and Variational Bayesian Inference, such long computation times is a hindrance to iteratively finding good solutions. In the case of ABC-SMC one would like to be able to experiment with, for example, different distance functions of varying complexity. But such long computation times for confirming results understandably becomes restrictive.

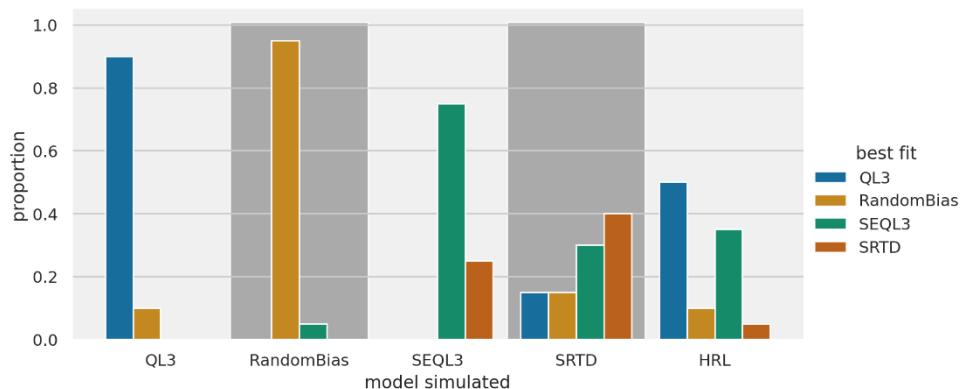


Figure 5.48 Model selection performance when HRL is excluded from models fitted. Proportion of subjects (y-axis) for each simulated model (x-axis) that was best fitted with what model (coloured bars as per legend). Shaded areas are only for readability.

⁴⁶ Laptop CPU: AMD Ryzen 7 5800H, 45W TDP

In Figure 5.48 the results have been plotted for fitting four models (excluding HRL) using ABC to 100 subjects, 20 for each algorithm. Keeping in mind this dataset is small, the results are nonetheless quite encouraging. If we compare to Figure 5.47, we see the results largely concur. We have very good performance for QL₃ and RandomBias, in the latter case performance is better than for MLE, where some RandomBias cases are selected as QL₃. For SEQL₃, MLE is clearly better, but ABC only selects SRTD instead of SEQL₃ which as has been discussed above can be explained in that their performance overlaps. We can see the same phenomena for the simulated SRTD cases where again SRTD and SEQL₃ are the two most selected models for these subjects. In contrast to MLE fitting, here ABC fits some SRTD subjects as QL₃ which tells us that the summary statistics used in the distance function mean that some SRTD subjects on a performance level are similar to QL₃ cases. For the HRL subjects, ABC fitting concurs with MLE fitting on a qualitative sense, in that we get a large amount of QL₃ and SEQL₃ fitted subjects, with QL₃ being the most common one.

To summarise, ABC fitting largely agrees with MLE fitting and we can now include HRL in the fitting process and with some more confidence be able to analyse the results.

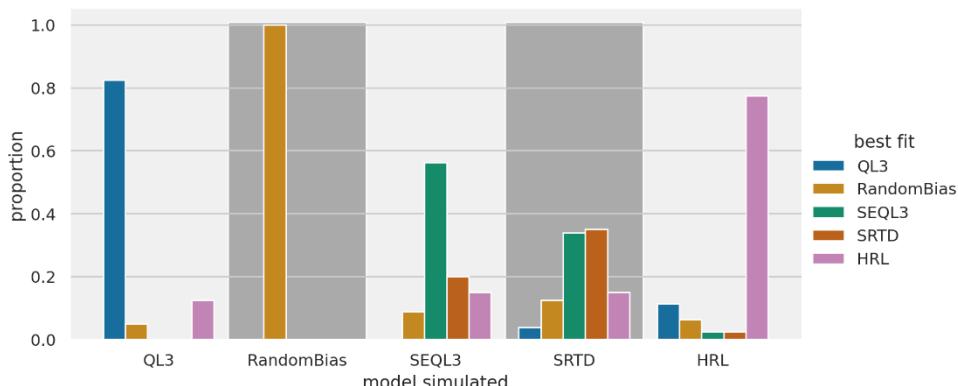


Figure 5.49 Model selection performance in Shapetask for ABC method. Proportion of subjects (y-axis) for each simulated model (x-axis) that was best fitted with what model (coloured bars as per legend). Shaded areas are only for readability.

For the second step of simulation and fitting of all five models, we generated a larger dataset with more subjects for better statistical certainty. We got 400 simulated and fitted subjects, 80 subjects per simulated agent/algorithm type. Because we now had an additional

model to fit on each iteration, the entire process took around 15h in total, so around 2.25 minutes per subject.

In Figure 5.49 we have plotted the results for the simfit process when HRL is included as one of the models fitted. We have very good, >75%, performance for the QL₃, RandomBias and HRL models. It is somewhat surprising that HRL performs so well, as the expectation would be that HRL was more mixed like SEQL₃ and SRTD. Again, we should remember that we only have 80 subjects for each model. But seeing how HRL is also the best fit for a decent amount of QL₃, SEQL₃ and SRTD subjects, another way to look at these results is they show the flexibility of HRL. Because HRL can display a wider variety of behaviours than the other models, and since the ABC method only samples part of the parameter space for each model, it's overall more likely HRL samples will have a low enough distance to be accepted by the ABC-SMC algorithm. In other words, the non-HRL cases fitted as HRL are most likely subjects with less distinct behaviour than what QL₃, SEQL₃ and SRTD are capable of. We will get back to such behavioural connections below.

We also see that SRTD and SEQL₃ cases overlap quite a bit, again due to their similarity as we saw in the behavioural studies and discussed above with both MLE and ABC fitting of four models (when HRL was excluded). The better performance for SEQL₃ here is likely due to it being able to produce stronger winners than SRTD or HRL. Again, see below for these speculations on behavioural connections.

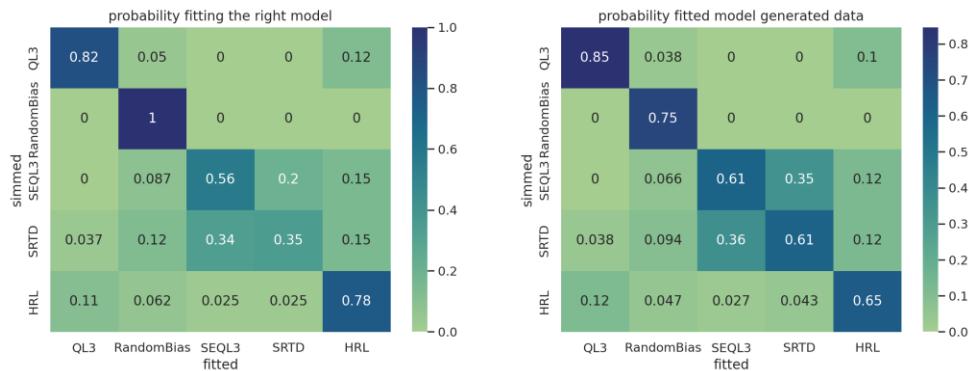


Figure 5.50 Confusion matrices for model fitting of Shapetask. Left: Regular confusion matrix, where rows sum to one, with name of simulated model on y-axis and name of fitted model on x-axis. For a given simulated agent row, each number on the row represents the proportion of cases where the best fitted model was the model with name as per x-axis. Right: Inverted confusion matrix, where columns sum to one, with name of simulated model on y-axis and name of fitted model on x-axis. For a given fitted model column, the numbers in that column represent the likelihood that the best fitted model generated the data.

We can summarise the results differently, in the form of confusion matrices as seen in Figure 5.50. The regular confusion matrix (left) shows the same information as in Figure 5.49, only in a more succinct format. The inverse confusion matrix (right), however, sheds a different light on the data. Values along the diagonal in the inverse confusion matrix indicate how likely it is that the model with the best fit was in fact the model that generated the data. For example, RandomBias seemed to have perfect behaviour in the previous figure (and the regular confusion matrix). But since RandomBias is also the best fit for some subjects simulated with other models, then if we assume these results are somewhat reliable, and RandomBias is the best fit for a human subject, we will only be correct in 75% of cases. That is, of course, also under the assumption that one of these models must be the true model of human behaviour.

The inverse confusion matrix provides a slightly rosier view of the performance when fitting SEQL₃ and SRTD, with around 60% probability for both. That is still not very high, but certainly better than what the regular confusion matrix provides. More importantly, if either SEQL₃ or SRTD are found to be the best fit, there is very little chance of any other models than those two being the correct one. As previously discussed, these two models are similar in structure and behaviour and that connection is kept even when only using summary statistics in ABC fitting.

Also as previously discussed, if HRL is found to be the best fit, there is around $1/3$ chance of either QL₃, SEQL₃ or SRTD being the true model. This also reinforces the previously made points that HRL is more flexible in the kinds of behaviours it is capable of.

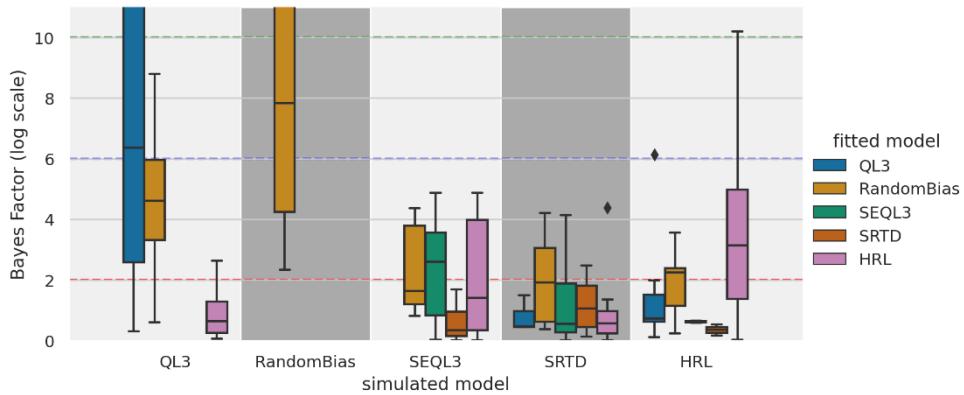


Figure 5.51 Bayes Factor (y-axis) for the best (colours as per legend) and second best fitted models for each simulated model (x-axis). Dashed horizontal lines indicate Bayes Factor evidence categories as per Table 3.1. Note that the y-axis has been cut so two categories are not shown in full, in order to better view the other categories. Shaded areas are for readability.

Another way to analyse the results is with the Bayes Factor (BF), as shown in Figure 5.51. The BF shown is using the natural log scale transformation described above in Table 3.1, and is here calculated between the winning model and the model with the next highest probability. Note that for two categories – QL₃ fitted to QL₃ simulations and the RandomBias simulations – are not shown in full. Their upper quartiles (75%) are 72.3 and 18.5, respectively. In cases where the next highest probability was 0, we calculated the BF as $1/10^{-50}$.

Recall from Table 3.1 that $\log(BF)$ between 2 and 6 are considered “positive” evidence and 6-10 are “strong”. What we can understand from the BF plot above is that for QL₃, RandomBias and HRL simulated cases, BF values above 2 are common. For correctly fitted QL₃ cases, the majority are above BF 6 and many even above 10. We have a similar pattern for correctly fitted RandomBias cases, but if we look at QL₃ cases identified as RandomBias, these can stretch up to BF 9. This means that some fits suggest the evidence for the RandomBias model would conventionally be regarded as strong, even when the data were actually generated by QL₃. Thus, we preferably need a BF of more than around 9 to be certain of a true

RandomBias case. For HRL fitted cases, there are those with values above 6, but we can also see that many cases of HRL wrongly fitted to SRTD have BF values between 2 and 5, so to be certain an HRL case is truly HRL we would prefer BF values more than 5.

For SRTD fitted wrongly to SRTD, we can see that the top whisker is around 4, which is fairly close to the top whisker of correctly fitted SRTD cases. Similarly, for SRTD fitted cases, they rarely get BF above 2 even for correctly fitted cases. Thus, it seems BF will be unreliable to correctly determine whether a subject, who was best fitted with SRTD or SRTD, gave data that were actually generated by either of these models.

To summarise, Figure 5.51 tells us that BF will probably only be useful to confirm QL₃, RandomBias and HRL cases. For those, we can only be certain with BF values above 6 for QL₃ and HRL, and above 9 for RandomBias.

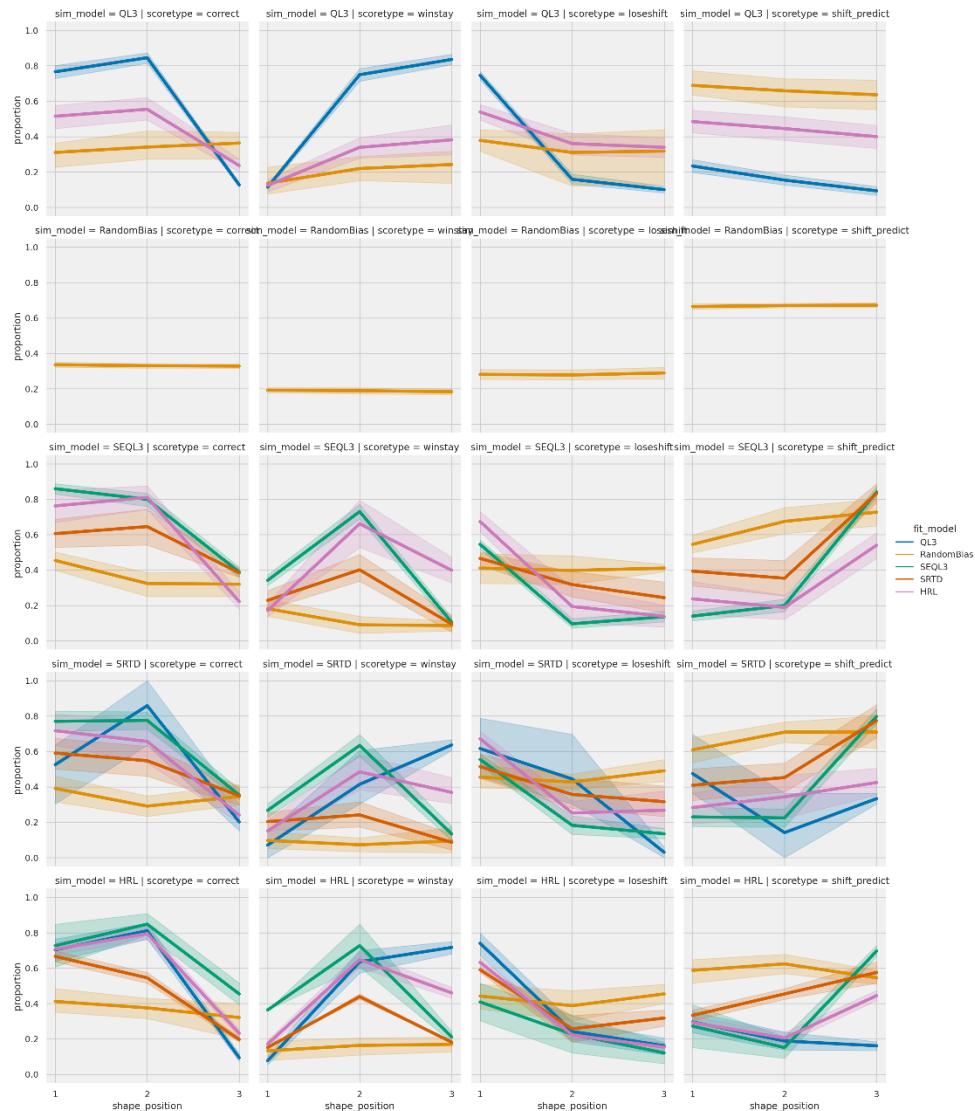


Figure 5.52 Behavioural curves for each simulated agent type (rows) showing proportion of choices (y-axis) for each shape position (x-axis) and score type (columns). Colours indicate what model was the best fit as per legend. Transparent areas around each line indicate 95% confidence interval. Simulated agent types, top to bottom: QL3, RandomBias, SEQL3, SRTD, HRL.

It may also be informative to look at behavioural plots of each simulated subject, based on what the best fitted model was. In Figure 5.52 we plot each simulated agent type on separate rows, with coloured lines in each subplot indicating what model was best fit to that group of subjects.

The most outstanding case here are the RandomBias simulations (second row from the top), with almost entirely flat lines for each score type. This is also reflected in the subjects simulated with other agent types where RandomBias was the best fit – they are the group of cases with score lines closer to flat lines than for all the other cases.

We also see that subjects best fit with QL₃ has a distinct WSLS type behaviour, although in the simulated SRTD cases best fit with QL₃ show a weaker pattern of WSLS behaviour. This latter category is new to us – from the above behavioural investigations we did not catch this “weak” form of WSLS cases, as the mean for win-stay in shape position 2 is not above 0.5. But the same cases have a slight increase in shift predict on shape position 3, which the QL₃ fitted cases for HRL and QL₃ simulations do not have.

When it comes to SEQL₃, as expected from the behavioural investigations, this is the best fitting model for very strong winners. It additionally has only three parameters, whereas both SRTD and HRL have four, thus giving it an edge in the model selection process. We can also see that when SRTD is the best fit, it often has high shift predict in position 3, often as high as SEQL₃, while HRL fitted cases have lower shift predict in position 3. Simultaneously, HRL has higher overall correct in positions one and two, and higher win-stay in positions 2 and 3, compared to SRTD, in most of the fitted cases.

In short, there are clear behavioural categories depending on what model is found to be the best fit. This should not come as a surprise, given we use these scores as our distance function in ABC fitting. Yet it is good to confirm our methods make sense.

5.6.5 SUMMARISING MODEL SELECTION PERFORMANCE

As expected from the behavioural studies, ABC model selection shows that HRL is the only model capable of being the best fit for all the models under consideration (except RandomBias). It is also the only model among the three models with alternate state representations – SEQL₃, SRTD and HRL – that has good recovery performance, as shown in Figure 5.49.

As we have discussed, the state representation structure for SEQL₃ and SRTD are close. That SEQL₃ has better recovery performance than SRTD can partly be seen as an effect of SEQL₃ having fewer parameters, and our model selection methods are favour models with few parameters. Because of their shared state representation structure, if we see SEQL₃ and SRTD as a group, then their recovery is in fact quite good, as demonstrated by the inverse confusion matrix in Figure 5.50.

It is also encouraging that RandomBias and QL₃ show distinct and good performance, despite HRL being able to display behaviour like that of QL₃. This means if we do find QL₃ or RandomBias being the best fit to human subjects, we can accept the result with good confidence, and especially so if the Bayes Factor in favour of these models is large.

However, it remains something of an issue that HRL, SRTD and SEQL₃ can overlap. The Bayes Factor plots in Figure 5.51 showed us that common values for accepting one model over another ($\log(BF)$ in the range 2-6; Table 3.1) are not strong enough to tease the considered models apart. Of these three models, HRL was the only one showing the possibility of BF values more than 6.

Despite the restricted usefulness of BF values, the behavioural plots of Figure 5.52 show why HRL, SRTD and SEQL₃ are difficult to tease apart. The best example being the SRTD simulations (third row from the top), where the three models' behaviours can be described as nested. SEQL₃ being the best fit for strong winners, SRTD the best fit for some winners and HRL being the best fit for behaviour that falls between winners and RandomBias behaviour. Thanks to our extensive behavioural studies before model fitting and selection, we can use these behavioural plots to help confirm the model selection results.

5.7 FITTING MODELS TO HUMAN BEHAVIOUR IN SHAPETASK

Armed with knowledge of how well the fitting process can be expected to work, we can now fit our human data. We have fitted the five models (HRL, QL₃, RandomBias, SEQL₃ and SRTD) using pyABC and the same settings as described above for the simfit process. We will present overall results for all three Shapetask versions – BOB, BOB-NR and Random – and then dig deeper into the results for BOB since that is the version we focus on as discussed earlier in this chapter. We have results for 110 human subjects, of which 39 are BOB, 32 are BOB-NR and 39 are for the Random version.

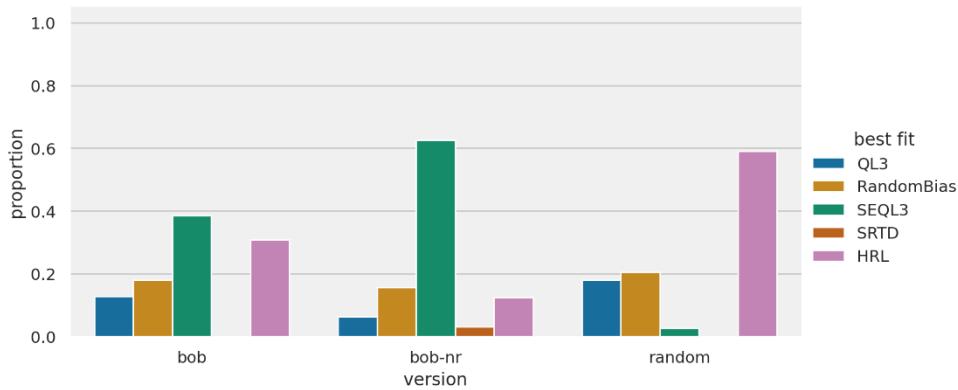


Figure 5.53 Model selections for human subjects in Shapetask. Proportion of subjects (y-axis) that were best fit with the model indicated by coloured legend, for each Shapetask version (x-axis).

The results from fitting and model selection can be seen in Figure 5.53. Starting with the result for the Random version of Shapetask, HRL is the best fitting model for most subjects, followed by RandomBias and QL3. As mentioned earlier, it is difficult to draw general conclusions from this version of the task, as the shape sequence varies a lot between subjects. It is nevertheless interesting that HRL can account for behaviour for so many subjects, as this implies the subjects tried to learn something about the task, and again drives home the point that the HRL model is quite flexible.

For the BOB and BOB-NR versions of Shapetask, there is an interesting contrast in the amount of SEQL3 fitted subjects. Recall that SEQL3 can reach higher overall performance than the other models. Because BOB-NR is less variable in the maximum length of shape repeats (3 vs 3 or 6) than BOB, a larger proportion of subjects had very good performance in the BOB-NR task. As seen in the behavioural studies of the human data above, strong winner subjects figured out the task quickly and thus reached high performance best explained by SEQL3.

We have also speculated that perhaps SEQL3 and SRTD are close enough in state representation structure that we can group them. One way to check if this speculation holds is to consider only those subjects that were best fit with SEQL3, and see what model was the next best fit. For BOB-NR, 20 cases were best fit by SEQL3 and of these 12/20 had SRTD as the runner-up, and 8/20 subjects had HRL as runner-up. For BOB, 15 cases were best fit by SEQL3 and of these there were 5/15 cases with SRTD as the runner-up and 10/15 had HRL as runner-up. Keeping in mind that sample sizes are small, these

numbers are somewhat consistent with our speculation. SRTD generally is the better fit in cases where shift predict on shape position is as high as SEQL3 but where correct score is not as high, as seen in for example Figure 5.52.

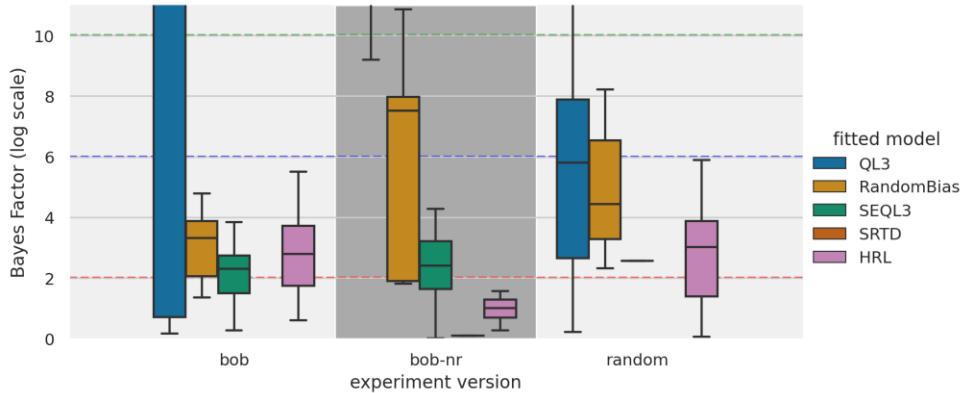


Figure 5.54 Boxplot showing Bayes Factor (y-axis) for each fitted model (colours as per legend) and Shapetask version (x-axis). Note y-axis has been cut off and does not show full range of a few categories (see text). Dashed horizontal lines indicate Bayes Factor evidence levels as shown in Table 3.1.

In Figure 5.54 we show the distribution of Bayes Factors (BF) for each task version. As previously, the BF is calculated between the best fitted model and the next best fitted model. The figure has been cut off on the y-axis as a few categories have very high values. For QL3 fits in BOB version, the mean $\log(BF)$ is 138 (SD 126) and for BOB-NR mean QL3 $\log(BF)$ is 120 (SD 156). QL3, and some RandomBias cases, are the only two models where the $\log(BF)$ is high enough to be certain it is the best fit. Even though many cases have $\log(BF)$ between 2-6 (positive evidence; Table 3.1), we found in Figure 5.51 that values should preferably be above 6 to confidently discern between the five models under consideration.

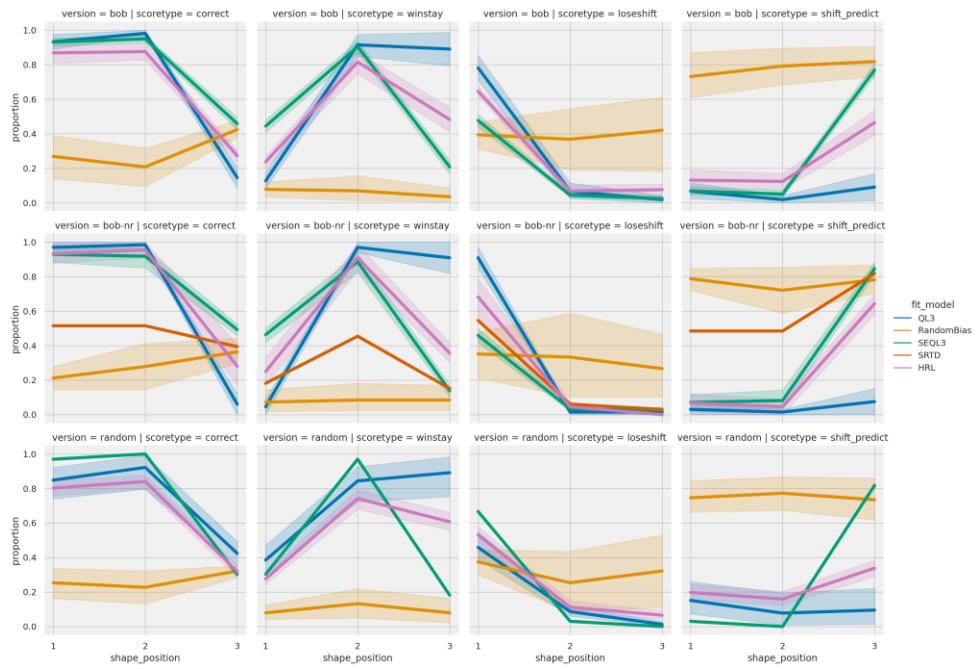


Figure 5.55 Behavioural plots for human subjects in Shapetask, grouped by fitted model as per colours in legend, with score type in columns and task version on rows. Each individual plot shows proportion of choices on y-axis across each shape position on x-axis. Transparent areas around lines indicate 95% CI. From left to right: Correct, Win-stay, Lose-shift, Shift predict. From top to bottom: BOB, BOB-NR, Random.

In the behavioural plots of Figure 5.55, we can mostly confirm our previous speculations. SEQL3 is the best fit for strong winners, QL3 is the best fit for strong WSLS subjects and HRL the best fit for weaker winners or mixes of WSLS and winner behaviour (Random version, bottom row). The RandomBias model covers a larger spectrum of scores, as seen by the large CI, but common for all the cases best fit by the RandomBias model is the relative non-varying behaviour across the shape positions, indicating that subjects did not pick up on any sequential patterns. SRTD is capable of strong winners like SEQL3, but except for a single BOB-NR case, it is absent in the results. This is most likely due to the parameter count being larger for SRTD than SEQL3, so the model selection method prefers the latter model. In the one SRTD case the shift predict score for positions one and two are almost 0.5. See below for a closer look at this subject.

Focusing on the BOB version, there are comparable proportions of HRL and SEQL3 cases – 12 and 15 cases, respectively. Before drawing conclusions about SEQL3 being the subjects with best performance, we should stop and ask if there is any correlation between number of 6-in-a-row shape runs and being best fit by HRL or SEQL3 models.

Perhaps it is simply the case that subjects who got more 6-in-a-row in BOB are the ones best fit with HRL? We can answer this question with an independent T-test between the number of 6-in-a-row for the two groups. It shows no significance, $t(25) = 0.33, p = 0.75$ so that is most likely not the reason for the difference in fits. This comparison is not significant between SEQL3 and QL3 either: $t(25) = -0.88, p = 0.39$.

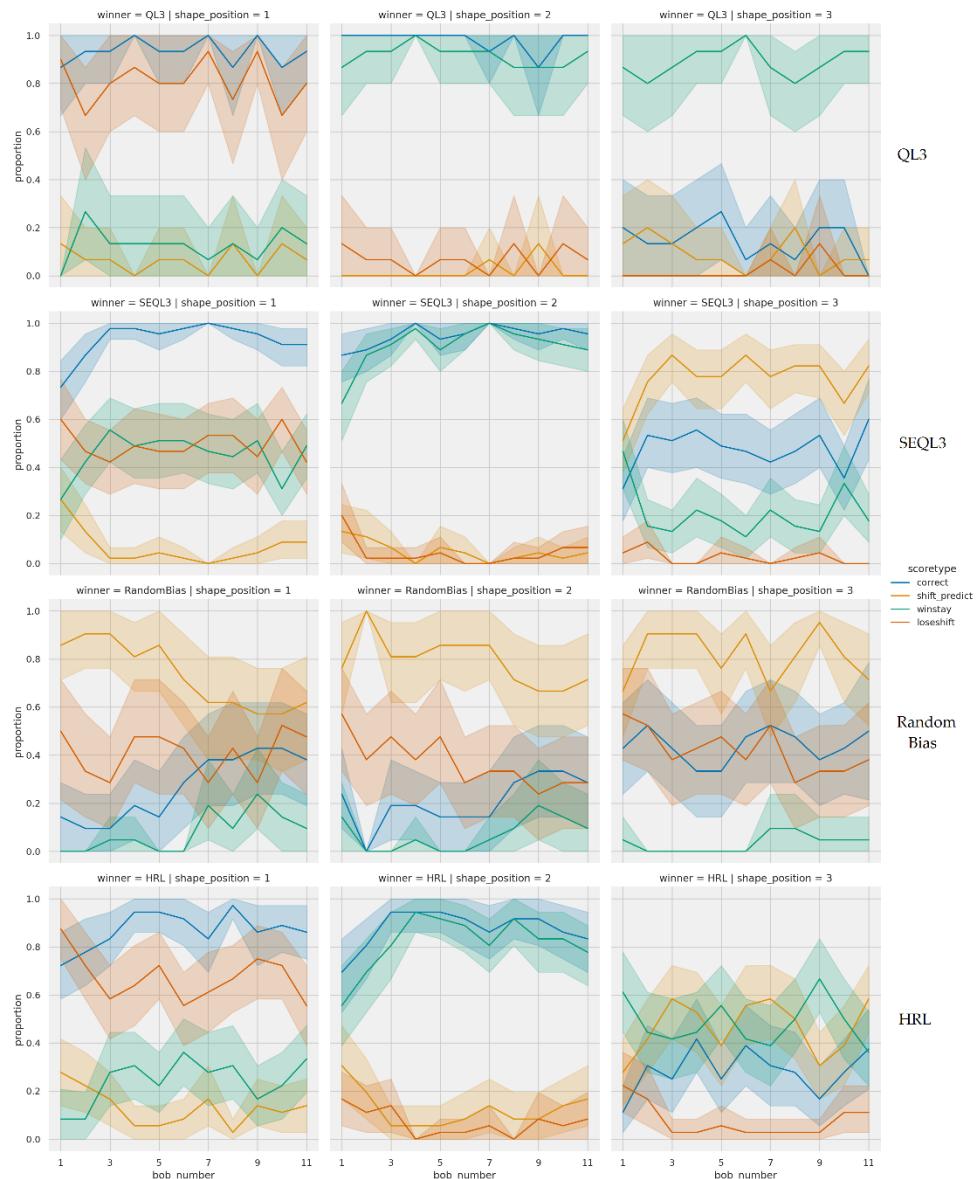


Figure 5.56 Learning curves for human subjects in BOB task. Proportion of choices (y-axis) for each bag-of-bags (x-axis, one bag-of-bags is 9 trials). Coloured lines indicate score type as per the legend, with transparent areas representing 95% CI. From left to right: Shape position 1, 2 and 3. From top to bottom: Cases best fit by QL3, SEQL3, RandomBias, HRL, respectively.

What may give us a hint about the reason for the difference between HRL and SEQL3 cases in the BOB version of the task are the

learning curves plotted in Figure 5.56. Note that the x-axis corresponds not to individual trials but bag-of-bags (i.e., each tick on x-axis is the average for 9 trials), to get smoother curves (less noise) with patterns that are more distinguishable.

Starting with QL₃ on the top row, we see very distinct WSLS behaviour. This is expected since it is what we also saw from QL₃ in behavioural investigations. What is interesting to note here though is that the behaviour starts very early in the experiment, meaning that the curves start out at around the same levels which they end up at. This makes it difficult to say if subjects are indeed learning, or if they are just selecting the same shape as whatever the current stimulus happens to be.

This is in contrast with the SSEQL₃ subjects (second row from the top), where we do see an increasing trend for the first 3 bag-of-bags (27 trials), after which behaviour stabilises. This pattern is especially apparent in the Shift predict score line for shape position three (right most column).

The RandomBias subjects (third row from the top) are quite interesting, because we can see that the average scores shown previously in Figure 5.55 do not tell the whole story. Towards the second half of the experiment, subjects' Correct score increases for shape positions one and two, while Shift predict decreases. Win-stay also has an increasing trend for the first two shape positions, even slightly in the third shape position. This tells us it may be the case these subjects (or at least a subset) are slower at picking up the pattern and could perhaps have figured it out with more trials.

For the HRL subjects, we can see that for shape positions one and two they look like a mix between QL₃ and SSEQL₃ behaviour. As for SSEQL₃, we do see that for the first 3 to 4 bag-of-bags their curves increase/decrease, so there is some learning going on. In the third shape position the Shift predict oscillates throughout the experiment, indicating that perhaps these subjects have spotted separate patterns for different shapes, and get confused whenever a particular shape suddenly appears in a set of six or three when before it was in three/six, respectively. Or perhaps they think the pattern changes throughout the experiment. In a way one could perhaps say these subjects do not see the forest for the trees – they focus on the details and fail to see the grander pattern.

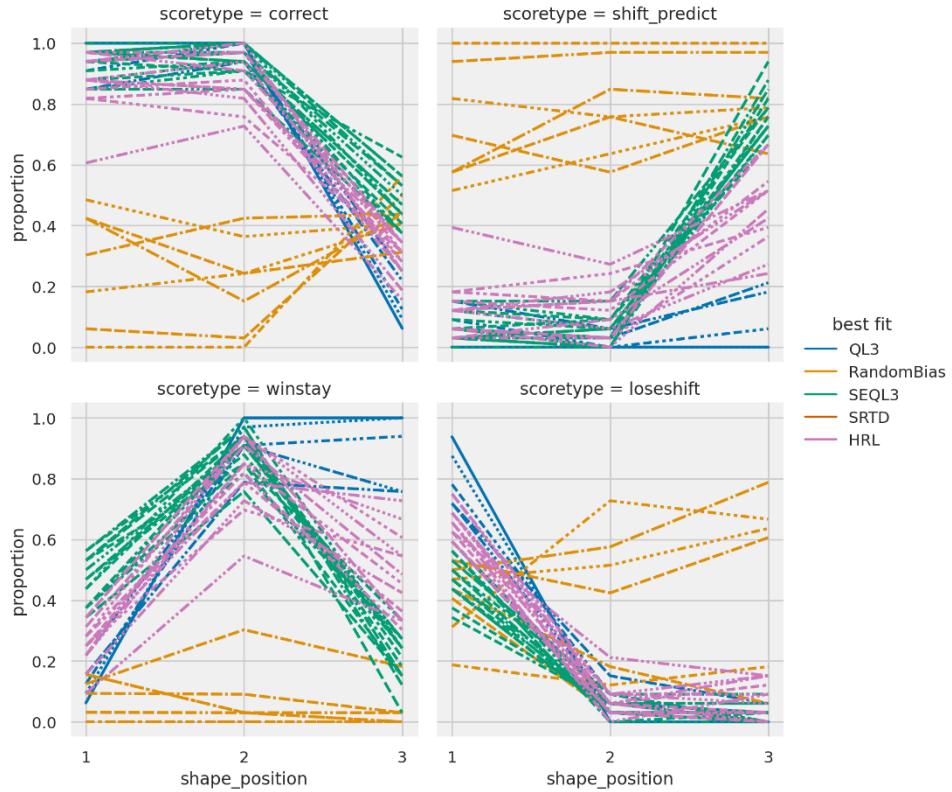


Figure 5.57 Individual summary scores for subjects playing Shapetask BOB. Proportion of choices (y-axis) for each shape position (x-axis) and score type (columns). Colours indicate best fitted model as per legend.

We should also take a look at individual subjects, in case there are any outliers we cannot catch in the group plots. This has been done in Figure 5.57. First, in the upper right plot, there are two RandomBias subjects that have shift predicted on almost every trial. It seems likely they did not make much effort in this task.

More interestingly, we can here see more clearly that there is a continuum in summary scores for subjects best fitted with QL₃ to HRL to SEQL₃. QL₃ subjects show quite clear WSLS behaviour, with one of these subjects having sloping to lower win-stay (lower left) in position 3 from 2. Staying in the win-stay plot, HRL subjects pick up where the just mentioned QL₃ subject left off. As we increase the slope between positions 2 and 3 we get to the SEQL₃ fitted subjects.

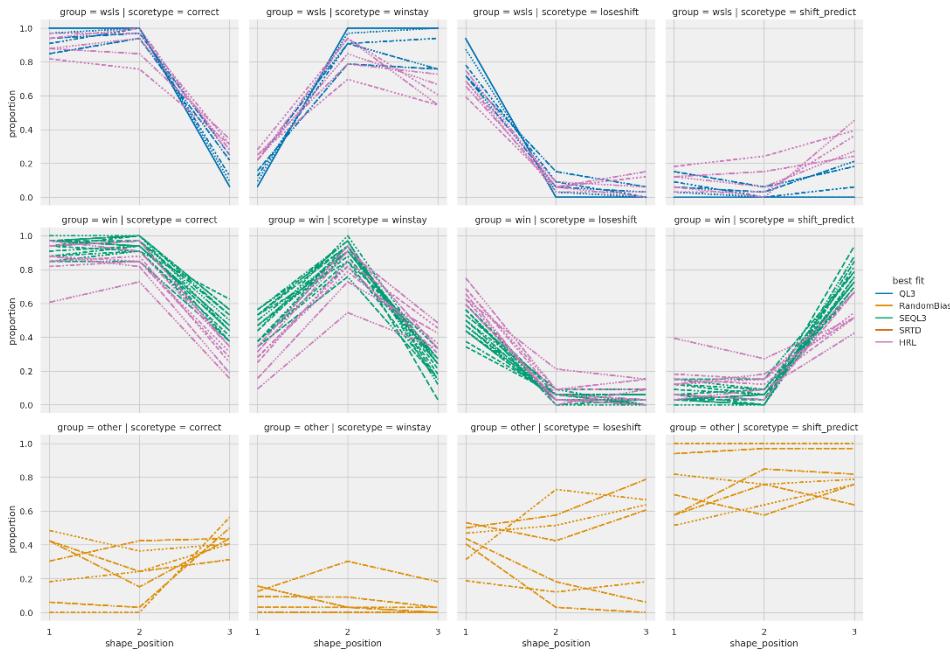


Figure 5.58 Shapetask BOB subjects, coloured by fitted model (see legend) and separated by behavioural groups (rows). Proportion of choices on y-axis for each score type (columns) and shape position (x-axis). From top to bottom: WSLS, Win, Other. From left to right: Correct, Win-stay, Lose-shift, Shift predict.

We can also connect back to the behavioural groups we defined in section 5.1.5. In Figure 5.58 we have separated our BOB subjects by those behavioural groups on rows, and coloured subjects by the model that was the best fit. Again, the behavioural groups are arbitrarily chosen (but of course based on logic), but they nicely highlight the same patterns just mentioned above with regard to the previous figure. Namely, that HRL straddles winners and WSLS subjects. Also noticeable is how the RandomBias model captures all the “other” cases exclusively. This is also the case for the BOB-NR subjects (see code repository for these plots), as well as for the subjects doing the random version of the task – except for one HRL fit.

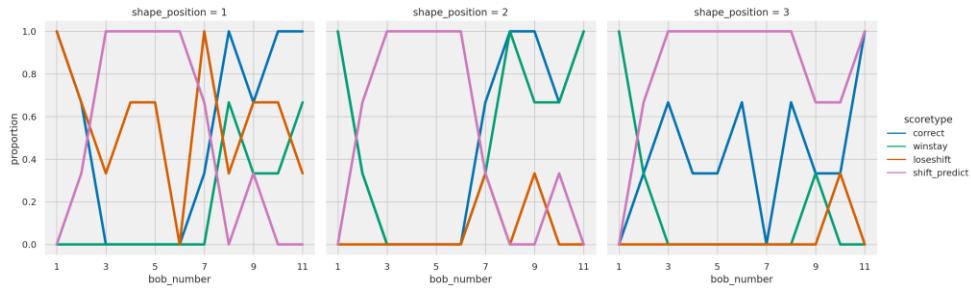


Figure 5.59 Learning curves for individual subject 38 in BOB-NR task, fitted best with SRTD model. Proportion of choices (y-axis) across bag-of-bags (x-axis; 9 trials averaged) for each score type (coloured lines as per legend) and shape position (columns). From left to right: Shape position 1, Shape position 2, Shape position 3.

Finally, we highlight the one individual fitted with SRTD. This subject – 38 – did the BOB-NR version of the task and as noted above, they had a peculiar score summary of 0.5 shift predict for the first two shape positions. In Figure 5.59 we can see that on early trials, subject 38 predicted a shift (pink line) on almost every trial, regardless of shape position. Then around halfway through – bob number 6 on x-axis – there is quite a dramatic shift in behaviour. Now subject 38 starts to win-stay/lose-shift on shape position 1, win-stay on position 2 and shift predict on position 3. In other words, subject 38 did almost solve the task, and found the pattern around halfway through the experiment. Perhaps some of the RandomBias fitted subjects mentioned above had a similar approach, and therefore showed a large proportion of shift predict choices for all shape positions. But for those subjects, the pattern never clicked.

5.8 CHAPTER SUMMARY AND DISCUSSION

In this chapter we introduced a novel decision-making task we call the shape sequence task – or Shapetask for short. Participants are shown a pre-determined sequence of shapes one at a time and need to predict what shape comes next. Participants are instructed to “find the pattern” but get no explicit feedback on their choices. We show that in versions of the task with reduced randomness in the sequence generation (i.e., with fewer sequences of the same shape that were longer than three), a majority of human subjects can find the underlying pattern within just 99 trials, often quite early on in the training sequence.

We then showed how standard Q-learning (QL_3) can account for those human subjects in Shapetask who show win-stay lose-shift (WSLS) behaviour, but *not* those subjects that solve the task by learning to shift their prediction every third trial in the sequence. Equipping Q-learning with a different state representation, however, the resulting $SEQL_3$ algorithm can show similar behaviour to those humans who find the key sequential pattern.

We argue these results show how Shapetask 1) can theoretically investigate how humans combine generalisation, differentiation and inference to find task structure; and 2) reveals that standard RL, equipped with appropriate states, can account for human behaviour. By not using explicit rewards we also argue the results 3) support theories proposing that RL may account for a more general “sensory” prediction error rather than one that is just reward based.

To improve support for these arguments, we introduced two models from the literature with plausible neurobiological connections. Hierarchical RL (HRL) imposes structure by essentially stacking two QL algorithms with the upper level selecting task sets – contexts – and the lower level selecting actions. We show this model can account for multiple types of behaviour in Shapetask, both “winners” (those who learn the sequential pattern) and subjects showing WSLS behaviour.

SRTD (Successor Representation, Temporal Difference) combines aspects of model-free and model-based RL by approximating the state transition function through experience. We show this model can better account for strong human winners in Shapetask than HRL but does not explain the behaviour of WSLS subjects. Furthermore, we demonstrate how SRTD is structurally equivalent to $SEQL_3$.

Finally, we demonstrate model selection in Shapetask using the ABC (Approximate Bayesian Computation) method. The four models QL_3 , $SEQL_3$, HRL, SRTD and a baseline RandomBias model were then fitted to the human data. The results of this model fitting and selection shows that strong WSLS subjects are indeed best fit with QL_3 , strong winners are best with $SEQL_3$, and those subjects showing weaker WSLS or winner behaviour are best fitted with HRL.

5.8.1 TECHNICAL DISCUSSION

It is unfortunate that HRL has an intractable likelihood, forcing us to use more crude methods for model fitting and selection. Using ABC we lose the dynamics of the task, meaning the changing behaviour across trials. Likelihood based methods naturally include this aspect.

We could compensate by selecting only the last x trials and perform ABC fitting on those data. But at what point does that become cherry picking our data, and/or how do we choose those last x trials? Perhaps a better option would be to include more aspects in the distance function. For example, one could find the slope of scores across all trials or even fit curves and use the found parameter values for distance.

But as discussed earlier, this would increase computation time enormously, as it takes many hours already. Calculating the slope or, even worse, fitting a curve, are much more complex calculations than just summing up the scores. However, if one were to attempt this, it would be appropriate to investigate if there is a faster Python library than pyABC (if one indeed exists). Another option would be to look beyond Python and find a fast ABC library in another programming language, for example C++ or Julia. Also, since each simulation is separate, the process is easily able to take advantage of parallelisation. Thus, it would most likely gain huge speed improvements to implement ABC with support for using GPUs (Graphics Processing Unit).

5.8.2 GENERAL DISCUSSION

No matter what specific model from HRL, SRTD or SSQL₃ is the best fit for individual subjects, we have shown that state representations do indeed matter for successfully solving Shapetask. These models update behavioural values based on implicit rewards in the same way as QL₃ does, but they differ in the composition of the structure these updates are applied to. We have thus showed that some manipulation of state representation change is needed, even if it would be neither of the specific models tested here.

Because subjects show such distinct behavioural types in Shapetask, it is unlikely that one single model would explain all

subjects. Instead, we would argue that one size does not fit all⁴⁷, and humans as a group are capable of multiple strategies. Our results would then also indicate that even among winner subjects, some humans use a strategy like SEQL3, and some a strategy like HRL.

Trying to tease HRL apart, it is possible that some subjects indeed represented the task as HRL would predict. This allowed these subjects to flexibly select an appropriate task set depending on the position and shape/colour. But this representation came at the cost of not showing strong distinct behavioural patterns: either WSLS or solving the task. This might have been because subjects had to expend effort exploring how to calibrate task set selection. This is suggested because of our observation that a larger number of trials was likely to be needed to be certain that HRL stabilises.

Similarly, those subjects best fit with SEQL3 successfully found a very efficient state representation, often fairly quickly. One could perhaps object here and say maybe subjects simply counted the number of shapes in a row and selected actions like “one-two-shift”. But without an appropriate representation of the states and task structure, how would subjects form the notion about what to count?

What is more important to consider, is how far we can take the equivalency between SEQL3 and SRTD. We showed above that structurally, they are in practice equivalent. We also showed in model selection performance that when SEQL3 or SRTD had generated the data, especially so if SRTD had generated the data, then the likelihood was very small that any other model than these two would be the best fit. Since SEQL3 additionally has fewer parameters than SRTD, and assuming the two models are nested, it would then make sense we only found a single case where SRTD was the best fit, rather than SEQL3.

Assuming then this equivalency between SEQL3 and SRTD hold, most winning human subjects were best fit with SRTD rather than HRL. Perhaps, and this is highly speculative, it is the case that SRTD is the fuzzy start of humans representing a task and as they gain information, this representation solidifies and solidifies into either QL3 or SEQL3 like representations. For most subjects, this happens quickly, but for subject 38 the solidification took longer. For some

⁴⁷ If one is so inclined to claim one size should fit all, then at best our results are inconclusive as to what size (model) that would be.

subjects – the ones best fit by a RandomBias model – this solidification does not happen until late in the task or perhaps never at all.

However, this is not quite the whole story, as our results show that those subjects using a QL₃ (WSLS) strategy, did so from the very start and showed little if any exploration. Then again this may be a question of effort and motivation, as there was no extra payment for solving the task. So QL₃/WSLS participants made their prediction by clicking the same shape that they saw on the screen and thus were able to get the task over with quickly. Additionally, as there was no explicit feedback on participants' choices, this behaviour was not "punished". Perhaps they expected a message to appear if they did the wrong thing and so carried on unchanged when such a message did not appear. It is also possible QL₃/WSLS subjects settled on the immediately found strategy and genuinely thought they had found the correct pattern, or at least a "good enough" pattern.

The obvious next step to develop this task further would be to add explicit rewards. As discussed above, part of our goal was to investigate how and if implicit rewards, in the form of sensory prediction of shapes (and/or colours), could be captured by standard RL algorithms that commonly assume explicit rewards. We successfully show this is indeed the case, adding to previous theories [52, 69, 194] on a more general prediction error story for midbrain dopamine than that of just rewards. But adding explicit rewards would complement our findings and would possibly enable us to answer some questions that are not clear from our current results.

On that point, using explicit rewards could allow for discerning between cases of low effort and those believing they found the correct pattern. The same can be said for RandomBias cases, where we do not know if subjects genuinely were confused or did not care. It is also possible that using explicit rewards would allow more subjects to successfully solve the task, which could, if desirable, be compensated in difficulty level by varying the number of shapes in each bag.

In short, we show Shapetask is viable for investigating the topic of state representation and task structure. With the possibilities for new variations of Shapetask, we believe it shows great promise for use in future work investigating these topics.

6 DISCUSSION AND FUTURE WORK

We started our journey by asking how humans and other animals find and create structure in the world. We approached this question from the perspective of reinforcement learning (RL) – learning from rewards. Our hypothesis was that RL is supported by other brain areas, where appropriate state representations for the task at hand are created, manipulated and/or retrieved. These state representations are provided to the RL system, where they are used as fundamental units for the learning process.

We based this hypothesis on two pillars, as detailed in chapter two. The first pillar is the reward prediction error hypothesis of dopamine, which is well established on a neurobiological level [191, 232]. The second pillar is derived from previous contradictory findings concerning what type of RL may occur in brains [52, 69, 195]. This contradiction may be explained by taking a more integrated view of dopamine as a more general stimulus prediction error [88, 175, 219] supported by (orbito-)frontal cortex and hippocampus providing state representations [14, 53, 227, 293].

6.1 SUMMARY OF KEY FINDINGS

6.1.1 CHAPTER THREE

In order to study how humans find task structure, we wanted to test models found in the literature on existing and new data gathered from human experiments. Testing such mathematical models of brain function, we need methods for parameter estimation and model selection. In chapter three, we introduced and investigated two main classes of methods for fitting models to data: maximum likelihood estimation (MLE) and Bayesian inference. Both methods use likelihoods – in essence inverted simulation functions – to find the most probable parameters used to generate the data being fitted. Bayesian methods have traditionally been difficult to use due to their computational requirements but have gained popularity in recent years in part due to increased availability of fast computers and partly thanks to improved algorithms for Bayesian inference. Most importantly they have gained popularity due to claims they are better

able to both estimate parameter values and find the best fitting models [2, 204, 226, 289].

We used a standard learning and decision-making task – the two-armed bandit – and simulated artificial datasets with an RL based algorithm playing this task (section 3.3), as well as an agent playing randomly (with potential for bias towards one option). We then compared (in sections 3.9-3.10) both parameter recovery and model selection performance with MLE and several kinds of Bayesian inference-based methods, some of which used toolkits published by other authors.

Among these toolkits, we found the Computational and Brain/Behaviour Modelling (CBM) [204] to be the easiest to use with a straight-forward approach to adding new models. It uses Bayesian hierarchical model fitting and selection, and thus naturally provides uncertainty measures for fits on both subject and group level. Unfortunately, CBM does not seem to be actively maintained and is only available in MATLAB (both issues will be further discussed below).

As for results, we found that some of the Bayesian based methods were indeed generally slightly better at parameter recovery than MLE. However, there was no direct correlation between parameter recovery performance and that of model selection. In fact, MLE was often noticeably better at selecting the correct model than Bayes' based methods (section 3.10.4). This was true even with few data points (100 trials per subject), which is supposedly where Bayesian inference shines [67, 226].

We found this surprising, and very important for two main reasons. One is that in experiments with human subjects, we often have to make do with relatively few datapoints because humans very easily get bored and so tasks are often kept quite short. The second is that MLE is multiple times faster – on the scale of minutes versus hours in worst case scenarios – than the Bayesian methods. This is important because if MLE is just as good or even better for model selection, there is little point in wasting energy on a more computationally demanding method.

It is also important because constructing likelihood functions is complicated. It is very easy to make mistakes, and so one should always test one's likelihood functions work by simulating data and

then see if the parameters used for generating said data can be recovered [197, 291]. This is often an iterative process – especially for mathematical non-natives. Hence, if each time you test a change to your likelihood function you have to wait hours to see the results, you will be satisfied when it seems to work. However, you might be deterred from a fuller exploration by the lengthy time commitment involved, and so may miss something that makes it not work so well for datasets other than your own.

The above findings conflict with some published studies that show Bayesian methods to be superior [2, 204, 289]. We suggest this may be due to that authors often use few artificial subjects, on the level of tens of simulated subjects, instead of thousands. Larger numbers are needed in our experience in order to cover most of the parameter space, which is important as recovery ability can vary across said parameter space. As just mentioned above, this is most likely due to the authors simply not having (or taking) the time for the entire fitting process for thousands of simulated subjects. Such pressure of time is unfortunately – in this author’s opinion – likely to be a direct consequence of the publish or perish culture in today’s academic world [86].

However, we should perhaps hold this judgment before testing the methods on more agents and task types. Our results may not necessarily be representative for more complex types of models and tasks. Therefore, we addressed this in chapter 4.

6.1.2 CHAPTER FOUR

In chapter four, we kept the best performing methods from the previous chapter – MLE and CBM – to investigate further. We now introduced more complexity to the task in the form of two types of reversal bandits. We also added a wider array of models into the mix, some of the RL family and two from the family of Hidden Markov models (HMM), models which have hidden – or belief – states that are inferred based on observations.

For each task and model, we simulated subjects and fitted all models. This confirmed our results from the previous chapter that MLE was the most reliable method for selecting the model that most likely generated the data (section 4.8). Furthermore, despite being easy to use in general, we found the CBM toolkit exhibited bugs that caused us having to leave this method out of a few of the

comparisons. This unfortunately made the comparison incomplete but would unlikely have changed the picture on what model selection method was best, considering the previous results from chapter three.

The specific issue encountered was one where fitting a model to data that was very different from what the model expected would cause the CBM program to crash. We reported the bug and have yet to receive a reply. From what we understand of the crash, it is caused by the fitting process being unable to find a good fit and thus it is then supposed to fall back on the priors. We bring this up here, because it further illustrates our point made about the results in the previous chapter, where authors do not fully test their methods. Because we ran into this error, not just with one of the models we added, but also one of the models that was included in the CBM toolkit. Again, we do not judge the authors. Oversights are easily made under pressure to publish.

We also fitted two human datasets with the two types of reversal bandit. In the first task, called simply ReversalBandit, all subjects except one were best fit by some form of RL based model without states. The one other subject was best fit with an HMM (section 4.9). The second task, known as WorthyBandit, was more complex, in that rewards varied in value rather than just being present or absent. Once again most subjects still were best fit by RL based models but here more subjects were best fit with the HMM (section 4.10.4).

We would argue, based on the assumption most of the best fits were indeed correct, that the result from the human data shows the value of considering individual behaviour within the sample. In many studies found in the literature, authors describe what the best fitting model is, but only at the group level [73, 76, 79, 155, 254, 271, 281]. We believe this approach can be misguided, depending on the kind of models one is fitting. If the model describes what happens on a biological level – how neurons activate for example – then this might be appropriate at group level as we all share the same underlying neurobiological principles. But the kind of models we are studying here describes learning and decision-making at the behavioural level. Considering variations in life experience and the creativity of humans, a group of them can deploy a multitude of strategies and solutions to one and the same problem. For example, for the free response question at the end of Shapetask (see section 5.1.4),

participants presented several alternate theories on what pattern they believed they found. So why would one – and only one – learning model describe all individuals' performance on a task?

As previously noted, even though Bayesian MCMC (Markov Chain Monte Carlo, see section 3.9) indeed has better overall parameter recovery performance than MLE, it is not fully reliable. Additionally, one should keep in mind that one reason for hierarchical Bayesian models overall having less noisy parameter estimates is due to shrinkage – individual values being drawn closer together as they all share a group level distribution. This is often cited as a major advantage [203], but only will be of value if all participants use the same strategy, i.e., can be described by the same model. If participants cannot be described by the same model – which we have argued is often the case – then shrinkage could actually lead to misleading results.

The above issues on heterogenous groups and shrinkage are taken into account in the Variational Bayesian Analysis Toolbox [72, 184] (VBAT, part of which we used in conjunction with MLE fitting as described in section 3.10.2) and CBM [204], where model selection and fitting are done concurrently. That means subjects that are not well fit with a specific model should also contribute little, if anything, to the group level parameter distribution(s) for that specific model. However, as discussed above, our results indicate MLE is still the better choice for model selection. Hopefully, Bayesian methods can improve here so we can get the good model selection performance of MLE as well as uncertainty measures of Bayesian inference.

For the reasons discussed, we would therefore recommend caution about studies claiming correlations between model parameter values and other measures such as personality, intelligence scales or mental health, unless the study includes proof their method can reliably recover parameters of simulations with known parameter values. Even then, it is important to consider how such results can be interpreted in relation to other tasks and populations [72, 185].

6.1.3 CHAPTER FIVE

In chapter five, we introduced the Shapetask, a novel learning and decision-making task. Participants are shown a sequence of coloured shapes and must predict what the next coloured shape will be. There are three different shapes, and the sequence of trials is generated so

that the same shape will always be presented at least three times in a row (although variation in number of repeats – sometimes six repeats, or even nine and twelve repeats in the hardest version of the task – occurred to differing extents across versions). Participants were not explicitly rewarded or punished for their choices. We argued that when a participant’s prediction for the next shape is correct, this acts pretty much like an explicit reward and likewise when their prediction is wrong this acts like a non-reward outcome. Subjects thus have to figure out that sequential position (how many times a shape has occurred in a row) is important, and that the context when the third repetition of a shape occurs is different from the first two repetitions, thus potentially requiring different choices. We showed that in easier versions of the task, with a lower chance of getting repeating sequences longer than three, a majority of the participants are able to correctly find the underlying pattern. Not only do they find the pattern, but most do so within 100 trials, often earlier (section 5.1.4.1).

We then showed how standard RL in the form of Q-learning (called QL3) can account for the behaviour of participants choosing a win-stay, lose-shift (WSLS) strategy (section 5.2.1). But in order to account for the participants that solve the task by correctly identifying the pattern, we had to modify the state representation of the task used by the same Q-learning algorithm (creating a novel version we called State Enhanced QL3, SEQL3; section 5.2.3).

We believe this shows support for our initial broad hypothesis that RL is supported by appropriate state representations. In chapter two (section 2.2) we noted how there are conflicting accounts in the literature as to whether model-free or model-based RL – or both – can explain signals in the dopaminergic system. We also described recent proposals that what has been viewed as model-based may perhaps be better explained by model-free systems being fed state representations from frontal cortex areas and/or hippocampus [206, 239, 288, 293]. If Q-learning can display different behaviours depending on the state representations on which computations are made – then perhaps the view of model-free versus model-based RL is misguided. Perhaps, what has been interpreted as model-based RL reflects the influences of the state representations being fed in from other brain areas.

These speculations are, as discussed in chapter two (section 2.3), in line with recent research. But in our case, it was mere speculation, as we manipulated the state representations into a form that was suitable for the task; and, at this point, we had shown only qualitative results with simulations. So, to more firmly establish a connection with other research, we fitted models proposed and developed in that research. These models (see next paragraphs) take an integrated view of RL and state and task structure.

We selected two of the most promising models from the literature. First, SRTD (Successor Representation, Temporal Difference) which combines aspects of model-free and model-based RL by approximating the state transition function through experience (section 5.4). We demonstrated how Shapetask can be construed as a spatial maze-like task fitting for the SRTD implementation we used (which had been developed to simulate performance in complex spatial mazes). We also demonstrated how SRTD is structurally equivalent to SEQL₃, and thus will have a similar behavioural profile in simulated behaviour. That is, like SEQL₃, SRTD can account for human participants who learn the sequential aspects of Shapetask (in chapter five we referred to these subjects as “winners”). We also showed that SRTD does not account for subjects who display a persistent WSLS strategy. This result adds to existing literature [188, 200] suggesting that cognitive maps for spatial and abstract knowledge may share mechanisms.

The second model, Hierarchical RL (HRL), imposes structure by essentially stacking two QL algorithms one above the other (section 5.3). The upper level is used to select task sets – or contexts – and the lower level selecting the actions deployed in each context. We showed that this model could account for multiple types of behaviour in Shapetask, both winners and subjects showing WSLS behaviour.

One notable detail found in our behavioural investigation of HRL was that two simulated subjects that use the same parameter values can exhibit qualitatively different behaviours (section 5.3.2.1). We showed this was due to randomness in task set selection, and that with a larger number of trials the issue could be alleviated. This effect appeared even with only two task sets. The consequences of this are that HRL may be highly dependent on task contingencies such as, for example, number of trials and reward magnitudes. For example, if

high reward magnitudes are possible, it may require a larger number of trials before task set values appropriate for the task are found. This effect seems to be additionally exacerbated with more than two task sets, as we showed in the same section. Additional study of this effect would be needed, but if researchers would attempt to, for example, correlate HRL parameters with other measures, this effect should be kept in mind.

Although HRL has shown promising results in previous research, it unfortunately has an intractable likelihood function disallowing the use of likelihood-based methods, such as MLE, for model fitting. We therefore used a simulation-based model fitting method – Approximate Bayesian Computation (ABC) – to fit our selected models to the Shapetask human data (section 5.6.2). We showed this method had a similar model selection profile to MLE when fitting the models with tractable likelihoods, which made us more comfortable with interpreting the ABC fitting results.

The model fitting and selection results for our human data (section 5.7) showed that strong WSLS subjects were best fit with QL_3 , strong winners best fit with $SEQL_3$, and subjects showing weaker WSLS or winner behaviour are best fitted with HRL. Some subjects were best fit with a randomly acting agent that had specific probabilistic biases for picking each of the three actions.

These results are interesting, because regardless of whether HRL, SRTD or $SEQL_3$ is the best fit for individual subjects, each of them updates behavioural values based on implicit rewards in the same way as QL_3 does. The difference between them lies in the composition of the structure these updates are applied to. We have thus showed that some manipulation of state representation change is needed to capture performance in Shapetask, even if none of the specific models tested here do a perfect job of capturing the full range of observed human performance.

Because subjects show such distinct behavioural types in Shapetask performance, it is unlikely that one single model would explain the behaviour of all subjects. Instead, we would argue against a one size fits all viewpoint, and stress that humans are capable of multiple strategies. If one were to argue for one size fits all, perhaps in pursuit of scientific parsimony, then it is very unclear, based on our results, what sort of model could capture all the diverse behaviours

that humans show in Shapetask. Our results in the model fitting and selection of Shapetask (section 5.7) would then indicate that even among winner subjects, some humans use a strategy like SEQL₃, and some a strategy like HRL.

The above arguments make intuitive sense. If it is the case that state representations matter – that the neural systems of humans and potentially other animals form mental structures that are useful for learning – then why would it be the case that everyone uses the same sort of representation? In other words, we are suggesting that although the basic physiology of the underlying neural system(s) may be the same, through the combination of life experience and genes, some individuals may use one strategy instead of another when faced with the same task. Since we do not really investigate how HRL/SRTD arises – only that these are cognitive structures that may have been used by participants – it is entirely plausible that some subjects used what looks like HRL and others used SRTD/SEQL₃.

An additional finding here to note is that as we did not use explicit rewards in Shapetask and ran our simulations treating outcomes that met participants' predictions as implicitly rewarding. The success of our simulations, using this assumption, therefore adds to the existing literature suggesting a larger role for dopamine than that of reward prediction errors [88]. This literature argues that dopamine accounts not only for reward prediction errors, but sensory prediction errors more generally. This would appear to fit with more general theories proposing that prediction errors of both the world, our body and our thoughts is how the mind is made up [253].

Overall, we believe the results just summarised show that Shapetask is a valid tool for investigating how humans find structure in the world. Our results are preliminary and thus naturally incomplete, and some aspects of our findings may be explained by lack of effort, confusion, or both, in our human participants. In order to test the ideas further, we would need, among other things, to compare learning using explicit rewards in the same task.

6.2 LIMITATIONS

The first limitation relates to Shapetask, and the assumptions we make about what “solving” the task means. We define solving it as changing one's prediction (with high probability) every third trial in

the sequence. We did find that those participants who solved the task by our definition were often able to report on this strategy in their debriefing comments, as well as informal feedback from pilot participants. So, we believe our assumption holds, but adding more variations of the task such as explicit rewards (further discussed below), would further ground this assumption. Furthermore, in the development of the task itself, we should perhaps have started Shapetask from a single shape with multiple colours, or a single colour and multiple shapes as a baseline and work up from there.

Another limitation is that we did not include any measure of working memory capacity, or used a model with aspects of working memory built in. This could be appropriate since not only is it sometimes argued that working memory is tightly integrated with (reinforcement) learning [54] – as discussed in chapter two – but there may also be correlations between working memory capacity and success in Shapetask. It is possible that there is no difference *per se* in the state representations a group of participants apply, but rather they vary in the level of difficulty they have remembering how many shapes in a row they have seen previously.

On the topic of additional models, we also could have tested more types of RL agents without manipulating the state structure. For example, we could have tested Dual- α and Dual-Update chapter four on Shapetask. Since these models use state representations like those of QL₃, they would not be able to differentiate between the last stimulus of a bag and the preceding ones, and thus are likely to have similar behavioural profiles to that of QL₃. In the case of Dual- α , it may also be the case that perseverance effects should be taken into account [254].

We also could have run studies in which participants performed both Shapetask and another task. This would have allowed us to “cross compare” algorithms on these multiple tasks. For example, if we had run these further studies, we could have compared the fits of the Alien task from [73] and Shapetask using SRTD, and similarly tested the fit of HRL on the maze task from [219] as well as Shapetask. It is not immediately clear how either implementation combination would work, so this would require additional tinkering with the models and/or tasks in order to do this.

There was not sufficient time during this PhD to fit additional models and run further studies with task combinations. This is always an issue with PhDs, but the issue was exacerbated here because much of the model fitting turned out to be unexpectedly time consuming. There were two aspects to this.

First, the time taken to simulate and fit successfully implemented models was often considerable. We noted earlier how important it is to ensure broad parameter coverage in model testing. So, in this thesis, we used sets of 1000 randomly selected parameter combinations where each such combination was run 100 times to average across stochastic variations of the tasks and agents (see for example section 3.3). Furthermore, we saw with ABC fitting for Shapetask, that it was roughly comparable (in terms of results) to MLE fitting. But it took around two minutes per subject with the five models included. Adding further models would naturally increase the time taken, and thus the time needed to test performance, for two reasons. The first is quite simply that more models mean more time. The second is that more models will inevitably lead to similar performance profiles. For example, adding more RL variants would very likely mean more models would show WSLS behaviour like QL3. In this case, our distance function may prove insufficient to properly distinguish between them. This would need to be alleviated by incorporating more nuances in the distance function used for ABC fitting; such additional nuances add further to the run times of the code.

We proposed a few more options on how to deal with ABC fitting due to intractable likelihoods in the chapter five discussion. Yet another option would be to simply exclude models with intractable likelihoods. But that would of course not be ideal. We have not encountered many papers using HRL, and its intractable likelihood may be partly responsible for this. However, see [213] for an interesting approach of using neural networks to approximate likelihoods.

A further reason for the unexpected time taken to test and fit the models was that, based on the literature, we did not expect Bayesian inference to be so little better than MLE. We therefore assumed our implementations of Bayesian inference were wrong, because we expected Bayesian inference – MCMC in particular – to be much more

accurate in parameter recovery and model selection. We switched from the Python specific library pyMC₃ to Stan as the latter has a larger community when it comes to the kind of models we are interested in, with existing packages like hBayesDM. This was overall a good move, as Stan has a clean separation of models from other code, interfaces for multiple programming languages, and built-in support for regular loops. Whereas in pyMC₃, specific operators are needed to loop through data where the next trial depends on the previous one as in our models⁴⁸.

Despite eventually finding the best packages to deploy, as discussed previously, it gradually became clear that many of these libraries (and associated papers) have undergone relatively limited testing. In hindsight this is not entirely unexpected as discussed above. Some packages have arbitrarily chosen boundaries for parameter values, some do not include more basic control models, and some appear to have tried their method on data from a relatively limited range of tasks. We encountered a case where the paper claims they have both code and data available, and the code was indeed available openly. But it would not run without the data. And the data was, in practice, not open. It was required to register at a government data repository and fill in a large number of forms to even be considered for access. Another case we encountered was one where the code repository linked from the published paper was empty, except for a message saying, “code available soon”. The paper was five years old.

We mention these examples, not to criticise anyone specifically, but rather to point to a problem within the entire field, and academia in general. Publish or perish is a known cultural issue within academia [293], and it causes authors to not be able to take the time needed to produce thoroughly tested work. This is especially a problem when it comes to work like that of this thesis, where code is as much a part of the thesis (or paper) as the text itself. In recent years, it has become more common for journals to ask, or even require, that code be published together with the paper. That is a step in the right direction, but it is extremely rare for anyone to (be required to) peer review the

⁴⁸ As of this writing in July 2022, pyMC was released with some major changes. It is possible the interface is easier to use now (it would be called 4, but they dropped the numbering)

code in the same way that they do the text. We suggest that this – peer review of code – would be a highly desirable change to the publication process in this field. Similar calls have been made by [214].

This issue is not only important in terms of the time required. It is also important because it can lead to misleading results. Papers may get published where authors have used existing tools and claim their model fitting implies that participants used a certain model with certain parameter value ranges. If the tool they used has not been thoroughly tested, the results may be inaccurate, as discussed above. Creating tools that makes it easier for others is obviously a very good thing for the scientific community. But it is important to provide users with the information they need to make informed decisions about the tool which has been provided.

Nevertheless, as we have shown, MLE is as good as, if not better, than Bayesian inference in terms of model selection for the models and tasks used in this thesis (as demonstrated in chapters three and four). We hope that future publications of this work may allow other researchers a more informed choice regarding the modelling and fitting tools they use.

In this section, we have addressed limitations, some of which naturally suggest further work that might remedy said limitations. Next we turn to suggestions for future work more generally.

6.3 FUTURE WORK

The most straight-forward future work here will be releasing the Python framework we developed for the thesis. It would involve some additional work in documentation, packaging and testing but would perhaps be valuable for other researchers to use. We may call it “simfitRLy”. Given our comments above we would naturally try to ensure this package has as few bugs as possible, good documentation and is easy to use and understand.

On that note, it would also be beneficial to port the part of VBAT we used into Python. Currently, VBAT is only available for MATLAB, which is prohibitively expensive for many researchers and interested individuals around the world. There is a free and open-source variant of MATLAB called Octave, but it is not as fully featured and does not

run VBAT. Many of the optimization functions, for example, that are used by VBAT and other model fitting programs, do not exist for Octave. We believe science should be as open as possible, and for that reason, journals might consider avoiding accepting code written in MATLAB that will not run in Octave. The code itself may be openly published, but it is in practice not fully open if one has to purchase the MATLAB program. We are quite privileged to have been able to access MATLAB through our university. We realise this is quite the radical suggestion, seeing as much of neuroscience and neuroimaging research uses MATLAB.

As mentioned above, there are two main avenues for future work. Task variations of Shapetask and adding more models.

6.3.1 TASK VARIATIONS

The downside of our assumption that implicit rewards are at play in Shapetask is that it is possible some subjects do believe they found the correct pattern when behaving in WSLS fashion. This would be possible if a participant felt they were gaining the maximum level of successful predictions possible from the task. With explicit rewards (and/or instructions that make it clear that almost 100% successful prediction can be attained) we would be able to more easily distinguish between such participants and those who were simply expending low levels of effort. However, it is likely that general performance will then increase, since subjects can calibrate their beliefs based on the feedback received. If desirable, one could possibly offset this improvement in average success levels by, for example, varying the number of shapes in each bag (e.g., 4 triangles, 3 squares, 2 circles).

Explicit rewards are not without their downsides, however. With implicit rewards we can partly investigate theories of sensory prediction error. This would not be as easy, or even possible, with explicit rewards. The existing implicit rewards version of the task could be used in imaging studies to investigate what overlap between brain areas there might be between explicit and implicit rewards, interrogating our speculations about sensory prediction.

On the note of imaging studies and task variations, as mentioned in chapter five, there are similarities between Shapetask and the “dimensions” task presented in [294]. In their task, participants had to find what aspect of the stimuli was relevant for receiving rewards,

and imaging results [190] showed that attentional mechanisms were involved with reducing the complexity down to the relevant features. The dimensions task thus requires representation learning, like Shapetask, but the difference between them is that in Shapetask the feature of interest is found across trials. There is bound to be overlap between relevant brain areas and networks used in both tasks, as they both involve representation learning, but we imagine Shapetask could activate working memory areas to a larger degree. If so, seeing as Shapetask and the dimensions task are similar, one could also imagine a spectrum of task versions between them, to test potential similarities and differences in attention versus working memory for representation learning.

Another, more intriguing, idea for improving motivation and effort when adding explicit rewards would be to make the task more interesting. As it is, the task is quite boring. One possible way of doing so would be to make the task into Rock-Paper-Scissors. Most people (at least in the western world, to our knowledge) are familiar with this game, and since it is a game, it may more naturally engage participants' motivation. It can be the same principle as Shapetask, if we imagine a computer opponent on screen, that has been coded to select Rock/Paper/Scissors in specific sequences. If participants can find the pattern that the opponent, for example, always picks three rocks and then either paper or scissors, then they can win most of the time. This would, of course, be more of explicit reward than implicit, as there is a clear win/lose condition. It would also potentially involve more social cognition than Shapetask, depending on how the opponent would be presented.

6.3.2 MODELS

When it comes to model fitting, there is research showing that the inclusion of response time data may improve parameter recovery [152, 300], which would be interesting to contrast with our results. Response times may also help distinguishing between the contributions of RL and working memory (WM) [168]. We have briefly mentioned WM above but have not included models in our analysis that incorporate WM directly. This would be an important addition, as they are closely connected [153, 301] and some processes which researchers think of as RL may in fact be WM [54, 301].

There are multiple aspects we could have added like variable learning rate, noise parameters, more action selection types (UCB, Thompson sampling) etc. They would all be valid and interesting to investigate but would of course be time consuming and not directly relevant to our interest in state representations. However, as discussed in chapter two, neural processes of state representation and action selection are interconnected and some studies indicate alternatives to SoftMax might be preferable [177].

More specific to that point of interest would be to include HMM models for Shapetask. Especially interesting would be more advanced HMM models, for example one such as [165], that more naturally include dynamic beliefs on switch points in reversal tasks. However, it is not immediately apparent how to apply hidden state models on Shapetask. This because the states that participants need to find in Shapetask – positions – are not hidden, per se. They are rather meta patterns across trials. One approach would be to adapt HRL taskset selection to use HMMs, which could potentially also allow a useful likelihood function to be created for HRL.

It would perhaps also be possible to combine SRTD with HRL, or at least the hierarchical concept, as shown with multi-scale SR [178]. SR also has multiple other implementation variations that would be valuable to add for comparison, such as SR-Dyna [177, 219]. However, SR-Dyna uses offline replay, which has been implemented in behavioural tasks as inter-trial delays. In Shapetask such delays would perhaps put larger load on potential WM influences.

So far we have mainly discussed “static” state representation models, meaning the representational structures are pre-defined. But how do animals create new and change our existing structures? There is research on non-parametric models [53, 91, 99, 189], in our context meaning models that can add categories based on data. So, instead of pre-defining for Shapetask that we have three shapes and three positions, the model would be able to dynamically add the shapes and positions as they are experienced through the task. This is, of course, not very straight-forward to implement, rather incredibly complex. Because the model would also need to be able to, if needed, reduce the number of states if such generalisation would be beneficial. In Shapetask, that would entail generalising that the first two positions are in principle the same state, but different from the

last position, where a different response is required. Of course, the model would need to be able to detect other patterns of repetition (over two, four, five trials etc.), subject to working memory capacity constraints [264] in particular for larger repeating sequences.

6.4 FINAL REMARKS

It is quite interesting to see how the bottom-up theories of RL and the top-down theories of the predictive mind [45, 84, 115] share the concept of prediction errors and are starting to converge in the research of recent years. We suspect that phenomena like the aha experience [265] will prove to be symptoms of internal (implicit) reward mechanisms that trigger when the predictions of our internal model – the state representation – matches what happens in the world.

Our thesis is just a small piece in this puzzle of unknown size, and we hope our findings will prove fitting for future pieces.

7 REFERENCES

- [1] Addyman, C. 2020. *The Laughing Baby: The extraordinary science behind what makes babies happy*. Unbound Publishing.
- [2] Ahn, W.-Y. et al. 2017. Revealing Neurocomputational Mechanisms of Reinforcement Learning and Decision-Making With the hBayesDM Package. *Computational psychiatry* (Cambridge, Mass.). 1, (Oct. 2017), 24–57.
- [3] AI and Compute: 2018. <https://blog.openai.com/ai-and-compute/>. Accessed: 2018-10-18.
- [4] AlphaStar: Mastering the Real-Time Strategy Game StarCraft II: 2019. <https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/>.
- [5] Armus, H.L. et al. 2006. Discrimination learning and extinction in paramecia (*P. caudatum*). *Psychological reports*. 98, 3 (Jun. 2006), 705–711.
- [6] Ash, I.K. et al. 2012. Investigating Insight as Sudden Learning. *The Journal of Problem Solving*. 4, 2 (2012), 2.
- [7] Atlas, Lauren Y 2019. How instructions shape aversive learning: higher order knowledge, reversal learning, and the role of the amygdala. *Current Opinion in Behavioral Sciences*. 26, (Apr. 2019), 121–129.
- [8] Austerweil, J.L. et al. 2019. Learning How to Generalize. *Cognitive science*. 43, 8 (Aug. 2019), e12777.
- [9] Babayan, B.M. et al. 2018. Belief state representation in the dopamine system. *Nature communications*. 9, 1 (May 2018), 1891.
- [10] Badcock, P.B. et al. 2019. The hierarchically mechanistic mind: A free-energy formulation of the human psyche. *Physics of life reviews*. 31, (Dec. 2019), 104–121.
- [11] Ballard, I.C. and McClure, S.M. 2019. Joint modeling of reaction times and choice improves parameter identifiability in reinforcement learning models. *Journal of neuroscience methods*. 317, (Apr. 2019), 37–44.
- [12] Balleine, B.W. and Dickinson, A. 1998. Goal-directed instrumental action: contingency and incentive learning and their cortical substrates. *Neuropharmacology*. 37, 4–5 (Apr. 1998), 407–419.
- [13] Balleine, B.W. and O'Doherty, J.P. 2010. Human and rodent homologies in action control: corticostriatal determinants of goal-directed and habitual action. *Neuropsychopharmacology: official publication of the American College of*

- Neuropsychopharmacology.* 35, 1 (Jan. 2010), 48–69.
- [14] Baram, A.B. et al. 2021. Entorhinal and ventromedial prefrontal cortices abstract and generalize the structure of reinforcement learning problems. *Neuron.* 109, 4 (Feb. 2021), 713–723.e7.
 - [15] Barbey, A.K. et al. 2013. Dorsolateral prefrontal contributions to human working memory. *Cortex; a journal devoted to the study of the nervous system and behavior.* 49, 5 (May 2013), 1195–1205.
 - [16] Barbey, A.K. et al. 2011. Orbitofrontal contributions to human working memory. *Cerebral cortex.* 21, 4 (Apr. 2011), 789–795.
 - [17] Barron, A.B. et al. 2010. The roles of dopamine and related compounds in reward-seeking behavior across animal phyla. *Frontiers in behavioral neuroscience.* 4, (Oct. 2010), 163.
 - [18] Bartlema, A. et al. 2014. A Bayesian hierarchical mixture approach to individual differences: Case studies in selective attention and representation in category learning. *Journal of mathematical psychology.* 59, (Apr. 2014), 132–150.
 - [19] Barto, A.G. 1995. Adaptive critics and the basal ganglia. *Models of information processing in the basal ganglia.* (1995), 215.
 - [20] Bechara, A. et al. 1994. Insensitivity to future consequences following damage to human prefrontal cortex. *Cognition.* 50, 1–3 (Apr. 1994), 7–15.
 - [21] Behrens, T.E.J. et al. 2018. What Is a Cognitive Map? Organizing Knowledge for Flexible Behavior. *Neuron.* 100, 2 (Oct. 2018), 490–509.
 - [22] Bennett, J.E.M. et al. 2021. Learning with reinforcement prediction errors in a model of the *Drosophila* mushroom body. *Nature communications.* 12, 1 (May 2021), 2569.
 - [23] Berke, J.D. 2018. What does dopamine mean? *Nature neuroscience.* 21, 6 (May 2018), 787–793.
 - [24] Bird, C.M. and Burgess, N. 2008. The hippocampus and memory: insights from spatial processing. *Nature reviews. Neuroscience.* 9, 3 (Mar. 2008), 182–194.
 - [25] Blei, D.M. et al. 2016. Variational Inference: A Review for Statisticians. *arXiv [stat.CO].*
 - [26] Bliss, T.V. and Collingridge, G.L. 1993. A synaptic model of memory: long-term potentiation in the hippocampus. *Nature.* 361, 6407 (Jan. 1993), 31–39.
 - [27] Botvinick, M. et al. 2019. Reinforcement Learning, Fast and Slow. *Trends in cognitive sciences.* 23, 5 (May 2019), 408–422.
 - [28] Botvinick, M.M. et al. 2009. Hierarchically organized behavior and its neural foundations: a reinforcement learning perspective. *Cognition.* 113, 3 (Dec. 2009), 262–280.
 - [29] Box, G.E.P. 1979. Robustness in the Strategy of Scientific Model

- Building. *Robustness in Statistics*. R.L. Launer and G.N. Wilkinson, eds. Academic Press. 201–236.
- [30] Brimblecombe, K.R. and Cragg, S.J. 2015. Substance P Weights Striatal Dopamine Transmission Differently within the Striosome-Matrix Axis. *The Journal of neuroscience: the official journal of the Society for Neuroscience*. 35, 24 (Jun. 2015), 9017–9023.
 - [31] Bromberg-Martin, E.S. et al. 2010. Dopamine in motivational control: rewarding, aversive, and alerting. *Neuron*. 68, 5 (Dec. 2010), 815–834.
 - [32] Brooks, R. 1986. A robust layered control system for a mobile robot. *IEEE Journal on Robotics and Automation*. 2, 1 (Mar. 1986), 14–23.
 - [33] Brown, T.B. et al. 2020. Language Models are Few-Shot Learners. *arXiv [cs.CL]*.
 - [34] Brunec, I.K. and Momennejad, I. 2022. Predictive Representations in Hippocampal and Prefrontal Hierarchies. *The Journal of neuroscience: the official journal of the Society for Neuroscience*. 42, 2 (Jan. 2022), 299–312.
 - [35] Buelow, M.T. and Suhr, J.A. 2009. Construct validity of the Iowa Gambling Task. *Neuropsychology review*. 19, 1 (Mar. 2009), 102–114.
 - [36] Buhrmester, M. et al. 2011. Amazon's Mechanical Turk: A New Source of Inexpensive, Yet High-Quality, Data? *Perspectives on psychological science: a journal of the Association for Psychological Science*. 6, 1 (Jan. 2011), 3–5.
 - [37] Burgess, N. et al. 2002. The human hippocampus and spatial and episodic memory. *Neuron*. 35, 4 (Aug. 2002), 625–641.
 - [38] Byrne, K.A. et al. 2016. Dopamine, depressive symptoms, and decision-making: the relationship between spontaneous eye blink rate and depressive symptoms predicts Iowa Gambling Task performance. *Cognitive, affective & behavioral neuroscience*. 16, 1 (Feb. 2016), 23–36.
 - [39] Canas and Jones 2010. Attention and reinforcement learning: constructing representations from indirect feedback. *AIB ... proceedings*. (2010).
 - [40] Cellia, M. et al. 2009. Impairment in flexible emotion-based learning in hallucination- and delusion-prone individuals. *Psychiatry research*. 170, 1 (Nov. 2009), 70–74.
 - [41] Chan, S.C.Y. et al. 2016. A Probability Distribution over Latent Causes, in the Orbitofrontal Cortex. *The Journal of neuroscience: the official journal of the Society for Neuroscience*. 36, 30 (Jul. 2016), 7817–7828.
 - [42] Chen, L. et al. 2021. Decision Transformer: Reinforcement

- Learning via sequence modeling. *arXiv [cs.LG]*.
- [43] Chittka, B. 1998. Sensorimotor learning in bumblebees: long-term retention and reversal training. *The Journal of experimental biology*. 201, 4 (Feb. 1998), 515–524.
 - [44] Chuhma, N. et al. 2014. Dopamine neurons control striatal cholinergic neurons via regionally heterogeneous dopamine and glutamate signaling. *Neuron*. 81, 4 (Feb. 2014), 901–912.
 - [45] Clark, A. 2015. *Surfing Uncertainty: Prediction, Action, and the Embodied Mind*. Oxford University Press.
 - [46] Clark, A. 2013. Whatever next? Predictive brains, situated agents, and the future of cognitive science. *The Behavioral and brain sciences*. 36, 3 (Jun. 2013), 181–204.
 - [47] Cohen, J.D. et al. 1994. Activation of the prefrontal cortex in a nonspatial working memory task with functional MRI. *Human brain mapping*. 1, 4 (1994), 293–304.
 - [48] Collins, A. and Koechlin, E. 2012. Reasoning, learning, and creativity: frontal lobe function and human decision-making. *PLoS biology*. 10, 3 (Mar. 2012), e1001293.
 - [49] Collins, A.G.E. et al. 2014. Human EEG uncovers latent generalizable rule structure during learning. *The Journal of neuroscience: the official journal of the Society for Neuroscience*. 34, 13 (Mar. 2014), 4677–4685.
 - [50] Collins, A.G.E. 2019. Reinforcement learning: bringing together computation and cognition. *Current Opinion in Behavioral Sciences*. 29, (Oct. 2019), 63–68.
 - [51] Collins, A.G.E. et al. 2014. Working memory contributions to reinforcement learning impairments in schizophrenia. *The Journal of neuroscience: the official journal of the Society for Neuroscience*. 34, 41 (Oct. 2014), 13747–13756.
 - [52] Collins, A.G.E. and Cockburn, J. 2020. Beyond dichotomies in reinforcement learning. *Nature reviews. Neuroscience*. 21, 10 (Oct. 2020), 576–586.
 - [53] Collins, A.G.E. and Frank, M.J. 2013. Cognitive control over learning: creating, clustering, and generalizing task-set structure. *Psychological review*. 120, 1 (Jan. 2013), 190–229.
 - [54] Collins, A.G.E. and Frank, M.J. 2012. How much of reinforcement learning is working memory, not reinforcement learning? A behavioral, computational, and neurogenetic analysis: Working memory in reinforcement learning. *The European journal of neuroscience*. 35, 7 (Apr. 2012), 1024–1035.
 - [55] Collins, A.G.E. and Frank, M.J. 2014. Opponent actor learning (OpAL): modeling interactive effects of striatal dopamine on reinforcement learning and choice incentive. *Psychological review*.

- 121, 3 (Jul. 2014), 337–366.
- [56] Collins, A.G.E. and Frank, M.J. 2016. Surprise! Dopamine signals mix action, value and error. *Nature neuroscience*. 19, 1 (Jan. 2016), 3–5.
- [57] Collins, A.G.E. and Frank, M.J. 2018. Within- and across-trial dynamics of human EEG reveal cooperative interplay between reinforcement learning and working memory. *Proceedings of the National Academy of Sciences of the United States of America*. 115, 10 (Mar. 2018), 2502–2507.
- [58] Collins, A.G.E. and Shenhav, A. 2021. Advances in modeling learning and decision-making in neuroscience. *Neuropsychopharmacology: official publication of the American College of Neuropsychopharmacology*. (Aug. 2021). DOI:<https://doi.org/10.1038/s41386-021-01126-y>.
- [59] Conway, M.A. and Pleydell-Pearce, C.W. 2000. The construction of autobiographical memories in the self-memory system. *Psychological review*. 107, 2 (Apr. 2000), 261–288.
- [60] Corbit, L.H. and Balleine, B.W. 2000. The role of the hippocampus in instrumental conditioning. *The Journal of neuroscience: the official journal of the Society for Neuroscience*. 20, 11 (Jun. 2000), 4233–4239.
- [61] Daunizeau, J. et al. 2014. VBA: a probabilistic treatment of nonlinear models for neurobiological and behavioural data. *PLoS computational biology*. 10, 1 (Jan. 2014), e1003441.
- [62] Daw, N.D. et al. 2006. Cortical substrates for exploratory decisions in humans. *Nature*. 441, 7095 (Jun. 2006), 876–879.
- [63] Daw, N.D. et al. 2011. Model-based influences on humans' choices and striatal prediction errors. *Neuron*. 69, 6 (Mar. 2011), 1204–1215.
- [64] Daw, N.D. 2011. Trial-by-trial data analysis using computational models. *Decision making, affect, and learning: Attention and performance XXIII*. 23, (2011), 3–38.
- [65] Daw, N.D. et al. 2005. Uncertainty-based competition between prefrontal and dorsolateral striatal systems for behavioral control. *Nature neuroscience*. 8, 12 (Dec. 2005), 1704–1711.
- [66] Dayan, P. 1993. Improving Generalization for Temporal Difference Learning: The Successor Representation. *Neural computation*. 5, 4 (Jul. 1993), 613–624.
- [67] Depaoli, S. et al. 2017. An introduction to Bayesian statistics in health psychology. *Health psychology review*. 11, 3 (Sep. 2017), 248–264.
- [68] Dezfouli, A. et al. 2019. Models that learn how humans learn: The case of decision-making and its disorders. *PLoS computational*

- biology.* 15, 6 (Jun. 2019), e1006903.
- [69] Doll, B.B. et al. 2012. The ubiquity of model-based reinforcement learning. *Current opinion in neurobiology.* 22, 6 (Dec. 2012), 1075–1081.
- [70] Donoso, M. et al. 2014. Human cognition. Foundations of human reasoning in the prefrontal cortex. *Science.* 344, 6191 (Jun. 2014), 1481–1486.
- [71] Duan, Y. et al. 2016. RL²: Fast Reinforcement Learning via Slow Reinforcement Learning. *arXiv [cs.AI].*
- [72] Eckstein, M.K. et al. 2021. What do reinforcement learning models measure? Interpreting model parameters in cognition and neuroscience. *Current Opinion in Behavioral Sciences.* 41, (Oct. 2021), 128–137.
- [73] Eckstein, M.K. and Collins, A.G.E. 2020. Computational evidence for hierarchically structured reinforcement learning in humans. *Proceedings of the National Academy of Sciences of the United States of America.* 117, 47 (Nov. 2020), 29381–29389.
- [74] Elliott Wimmer, G. and Büchel, C. 2019. Learning of distant state predictions by the orbitofrontal cortex in humans. *Nature communications.* 10, 1 (Jun. 2019), 2554.
- [75] Elman, J.L. et al. 1996. *Rethinking Innateness: A Connectionist Perspective on Development.* MIT Press.
- [76] Eppinger, B. et al. 2013. Of goals and habits: age-related and individual differences in goal-directed decision-making. *Frontiers in neuroscience.* 7, (Dec. 2013), 253.
- [77] Evans, J.S.B.T. 2003. In two minds: dual-process accounts of reasoning. *Trends in cognitive sciences.* 7, 10 (Oct. 2003), 454–459.
- [78] Farashahi, S. et al. 2020. Learning arbitrary stimulus-reward associations for naturalistic stimuli involves transition from learning about features to learning about objects. *Cognition.* 205, (Dec. 2020), 104425.
- [79] Feher da Silva, C. and Hare, T.A. 2020. Humans primarily use model-based inference in the two-stage task. *Nature human behaviour.* 4, 10 (Oct. 2020), 1053–1066.
- [80] Felin, T. et al. 2017. Rationality, perception, and the all-seeing eye. *Psychonomic bulletin & review.* 24, 4 (Aug. 2017), 1040–1059.
- [81] Fengler, A. et al. 2021. Likelihood approximation networks (LANs) for fast inference of simulation models in cognitive neuroscience. *eLife.* 10, (Apr. 2021). DOI:<https://doi.org/10.7554/eLife.65074>.
- [82] Frank, M.J. 2011. Computational models of motivated action selection in corticostriatal circuits. *Current opinion in neurobiology.* 21, 3 (Jun. 2011), 381–386.

- [83] Friston, K. et al. 2014. The anatomy of choice: dopamine and decision-making. *Philosophical transactions of the Royal Society of London. Series B, Biological sciences.* 369, 1655 (Nov. 2014). DOI:<https://doi.org/10.1098/rstb.2013.0481>.
- [84] Friston, K. 2010. The free-energy principle: a unified brain theory? *Nature reviews. Neuroscience.* 11, 2 (Feb. 2010), 127–138.
- [85] Friston, K.J. et al. 2009. Reinforcement learning or active inference? *PloS one.* 4, 7 (Jul. 2009), e6421.
- [86] Frith, U. 2020. Fast Lane to Slow Science. *Trends in cognitive sciences.* 24, 1 (Jan. 2020), 1–2.
- [87] Gabbert, F. et al. 2003. Memory conformity: can eyewitnesses influence each other's memories for an event? *Applied cognitive psychology.* 17, 5 (Jul. 2003), 533–543.
- [88] Gardner, M.P.H. et al. 2018. Rethinking dopamine as generalized prediction error. *Proceedings. Biological sciences / The Royal Society.* 285, 1891 (Nov. 2018). DOI:<https://doi.org/10.1098/rspb.2018.1645>.
- [89] Garvert, M.M. et al. 2017. A map of abstract relational knowledge in the human hippocampal–entorhinal cortex. *eLife.* 6, (Apr. 2017), e17086.
- [90] Gershman, S.J. 2018. Deconstructing the human algorithms for exploration. *Cognition.* 173, (Apr. 2018), 34–42.
- [91] Gershman, S.J. et al. 2015. Discovering latent causes in reinforcement learning. *Current Opinion in Behavioral Sciences.* 5, (Oct. 2015), 43–50.
- [92] Gershman, S.J. 2016. Empirical priors for reinforcement learning models. *Journal of mathematical psychology.* 71, (Apr. 2016), 1–6.
- [93] Gershman, S.J. et al. 2013. Gradual extinction prevents the return of fear: implications for the discovery of state. *Frontiers in behavioral neuroscience.* 7, (Nov. 2013), 164.
- [94] Gershman, S.J. et al. 2014. Retrospective revaluation in sequential decision making: a tale of two systems. *Journal of experimental psychology. General.* 143, 1 (Feb. 2014), 182–194.
- [95] Gershman, S.J. et al. 2012. The successor representation and temporal context. *Neural computation.* 24, 6 (Jun. 2012), 1553–1568.
- [96] Gershman, S.J. 2018. The Successor Representation: Its Computational Logic and Neural Substrates. *The Journal of neuroscience: the official journal of the Society for Neuroscience.* 38, 33 (Aug. 2018), 7193–7200.
- [97] Gershman, S.J. and Daw, N.D. 2017. Reinforcement Learning and Episodic Memory in Humans and Animals: An Integrative Framework. *Annual review of psychology.* 68, (Jan. 2017), 101–128.
- [98] Gershman, S.J. and Niv, Y. 2015. Novelty and Inductive

- Generalization in Human Reinforcement Learning. *Topics in cognitive science*. 7, 3 (Jul. 2015), 391–415.
- [99] Gershman, S.J. and Niv, Y. 2013. Perceptual estimation obeys Occam's razor. *Frontiers in psychology*. 4, (Sep. 2013), 623.
- [100] Gershman, S.J. and Uchida, N. 2019. Believing in dopamine. *Nature reviews. Neuroscience*. 20, 11 (Nov. 2019), 703–714.
- [101] Gobet, F. et al. 2001. Chunking mechanisms in human learning. *Trends in cognitive sciences*. 5, 6 (Jun. 2001), 236–243.
- [102] Goodale, M.A. and Milner, A.D. 1992. Separate visual pathways for perception and action. *Trends in neurosciences*. 15, 1 (Jan. 1992), 20–25.
- [103] Grillner, S. and Robertson, B. 2016. The Basal Ganglia Over 500 Million Years. *Current biology: CB*. 26, 20 (Oct. 2016), R1088–R1100.
- [104] Gronchi, G. and Giovannelli, F. 2018. Dual Process Theory of Thought and Default Mode Network: A Possible Neural Foundation of Fast Thinking. *Frontiers in psychology*. 9, (Jul. 2018), 1237.
- [105] Gurney, K. et al. 2001. A computational model of action selection in the basal ganglia. I. A new functional anatomy. *Biological cybernetics*. 84, 6 (Jun. 2001), 401–410.
- [106] Ha, D. and Schmidhuber, J. 2018. World Models. *arXiv [cs.LG]*.
- [107] Haber, S.N. and Knutson, B. 2010. The reward circuit: linking primate anatomy and human imaging. *Neuropsychopharmacology: official publication of the American College of Neuropsychopharmacology*. 35, 1 (Jan. 2010), 4–26.
- [108] Hamid, A.A. et al. 2016. Mesolimbic dopamine signals the value of work. *Nature neuroscience*. 19, 1 (Jan. 2016), 117–126.
- [109] Hampton, A.N. et al. 2006. The role of the ventromedial prefrontal cortex in abstract state-based inference during decision making in humans. *The Journal of neuroscience: the official journal of the Society for Neuroscience*. 26, 32 (Aug. 2006), 8360–8367.
- [110] Harlow, H.F. 1949. The formation of learning sets. *Psychological review*. 56, 1 (Jan. 1949), 51–65.
- [111] Hassabis, D. et al. 2007. Patients with hippocampal amnesia cannot imagine new experiences. *Proceedings of the National Academy of Sciences of the United States of America*. 104, 5 (Jan. 2007), 1726–1731.
- [112] Henrich, J. et al. 2010. The weirdest people in the world? *The Behavioral and brain sciences*. 33, 2–3 (Jun. 2010), 61–83; discussion 83–135.
- [113] Hoffman, D.D. 2009. The Interface Theory of Perception. *Object*

- Categorization: Computer and Human Vision Perspectives.* Sven Dickinson, Michael Tarr, Ales Leonardis, Bernt Schiele, ed. Oxford University Press. 148–265.
- [114] Hoffman, D.D. et al. 2015. The Interface Theory of Perception. *Psychonomic bulletin & review*. 22, 6 (Dec. 2015), 1480–1506.
 - [115] Hohwy, J. 2013. *The Predictive Mind*. Oxford University Press.
 - [116] Hollerman, J.R. and Schultz, W. 1998. Dopamine neurons report an error in the temporal prediction of reward during learning. *Nature neuroscience*. 1, 4 (Aug. 1998), 304–309.
 - [117] Huang, C.-Z.A. et al. 2018. Music Transformer. *arXiv [cs.LG]*.
 - [118] Hunt, D. 2020. *Modelling variations in human learning in probabilistic decision-making tasks*. Goldsmiths, University of London.
 - [119] Huys, Q.J.M. et al. 2016. Computational psychiatry as a bridge from neuroscience to clinical applications. *Nature neuroscience*. 19, 3 (Mar. 2016), 404–413.
 - [120] Huys, Q.J.M. et al. 2015. Interplay of approximate planning strategies. *Proceedings of the National Academy of Sciences of the United States of America*. 112, 10 (Mar. 2015), 3098–3103.
 - [121] Intraub, H. and Richardson, M. 1989. Wide-angle memories of close-up scenes. *Journal of experimental psychology. Learning, memory, and cognition*. 15, 2 (Mar. 1989), 179–187.
 - [122] James, W. 1890. *The Principles of Psychology* -. Henry Holt and Company.
 - [123] Jennings, E. and Madigan, M. 2016. astroABC: An Approximate Bayesian Computation Sequential Monte Carlo sampler for cosmological parameter estimation. *arXiv [astro-ph.IM]*.
 - [124] Joel, D. et al. 2002. Actor–critic models of the basal ganglia: new anatomical and computational perspectives. *Neural networks: the official journal of the International Neural Network Society*. 15, 4 (Jun. 2002), 535–547.
 - [125] Kaelbling, L.P. et al. 1998. Planning and acting in partially observable stochastic domains. *Artificial intelligence*. 101, 1 (May 1998), 99–134.
 - [126] Kaelbling, L.P. et al. 1996. Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research*. 4, (May 1996), 237–285.
 - [127] Kahneman, D. 2011. *Thinking, fast and slow*. Macmillan.
 - [128] Kahneman, D. and Tversky, A. 1979. Prospect theory: An analysis of decision under risk. *Econometrica: journal of the Econometric Society*. 47, 2 (Mar. 1979), 263.
 - [129] Kaiser, L. et al. 2019. Model-Based Reinforcement Learning for Atari. *arXiv [cs.LG]*.

- [130] Kass, R.E. and Raftery, A.E. 1995. Bayes Factors. *Journal of the American Statistical Association*. 90, 430 (Jun. 1995), 773–795.
- [131] Keasar, T. et al. 2002. Bees in two-armed bandit situations: foraging choices and possible decision mechanisms. *Behavioral ecology: official journal of the International Society for Behavioral Ecology*. 13, 6 (Nov. 2002), 757–765.
- [132] Kell, A.J.E. et al. 2018. A Task-Optimized Neural Network Replicates Human Auditory Behavior, Predicts Brain Responses, and Reveals a Cortical Processing Hierarchy. *Neuron*. 98, 3 (May 2018), 630-644.e16.
- [133] Khan, S. et al. 2021. Transformers in Vision: A Survey. *arXiv [cs.CV]*.
- [134] Kirsh, D. and Maglio, P. 1994. On distinguishing epistemic from pragmatic action. *Cognitive science*. 18, 4 (Oct. 1994), 513–549.
- [135] Klinger, E. et al. 2018. pyABC: distributed, likelihood-free inference. *Bioinformatics*. 34, 20 (Oct. 2018), 3591–3593.
- [136] Koenderink, J. 2011. Vision as a user interface. *Human Vision and Electronic Imaging XVI* (Feb. 2011), 786504.
- [137] Kool, W. et al. 2017. Cost-Benefit Arbitration Between Multiple Reinforcement-Learning Systems. *Psychological science*. 28, 9 (Sep. 2017), 1321–1333.
- [138] Kruschke, J. 2014. *Doing Bayesian Data Analysis: A Tutorial with R, JAGS, and Stan*. Academic Press.
- [139] Kuhn, G. and Rensink, R.A. 2016. The Vanishing Ball Illusion: A new perspective on the perception of dynamic events. *Cognition*. 148, (Mar. 2016), 64–70.
- [140] Kuleshov, V. and Precup, D. 2000. Algorithms for the multi-armed bandit problem. *Journal of machine learning research: JMLR*. 1, (2000), 1–48.
- [141] Lake, B.M. et al. 2017. Building machines that learn and think like people. *The Behavioral and brain sciences*. 40, (Jan. 2017), e253.
- [142] Lam, S.K. et al. 2015. Numba: a LLVM-based Python JIT compiler. *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC* (New York, NY, USA, Nov. 2015), 1–6.
- [143] Langdon, A.J. et al. 2018. Model-based predictions for dopamine. *Current opinion in neurobiology*. 49, (Apr. 2018), 1–7.
- [144] Lashley, K.S. 1951. *The problem of serial order in behavior*. Bobbs-Merrill.
- [145] Lau, B. and Glimcher, P.W. 2005. Dynamic response-by-response models of matching behavior in rhesus monkeys. *Journal of the experimental analysis of behavior*. 84, 3 (Nov. 2005), 555–579.

- [146] Lee, M.D. 2011. How cognitive modeling can benefit from hierarchical Bayesian models. *Journal of mathematical psychology*. 55, 1 (Feb. 2011), 1–7.
- [147] Lee, M.D. and Wagenmakers, E.-J. 2014. *Bayesian Cognitive Modeling: A Practical Course*. Cambridge University Press.
- [148] Lee, S.W. et al. 2014. Neural computations underlying arbitration between model-based and model-free learning. *Neuron*. 81, 3 (Feb. 2014), 687–699.
- [149] de Leeuw, J.R. 2015. jsPsych: a JavaScript library for creating behavioral experiments in a Web browser. *Behavior research methods*. 47, 1 (Mar. 2015), 1–12.
- [150] Leibo, J.Z. et al. 2017. Multi-agent Reinforcement Learning in Sequential Social Dilemmas. *arXiv [cs.MA]*.
- [151] Lengyel, M. and Dayan, P. 2008. Hippocampal Contributions to Control: The Third Way. *Advances in Neural Information Processing Systems* 20. J.C. Platt et al., eds. Curran Associates, Inc. 889–896.
- [152] Leong, Y.C. et al. 2017. Dynamic Interaction between Reinforcement Learning and Attention in Multidimensional Environments. *Neuron*. 93, 2 (Jan. 2017), 451–463.
- [153] Li, H.-H. et al. 2021. Joint representation of working memory and uncertainty in human cortex. *bioRxiv*.
- [154] Li, L. et al. 2006. Towards a unified theory of state abstraction for MDPs. *ISAIM* (2006).
- [155] Liakoni, V. et al. 2022. Brain signals of a Surprise-Actor-Critic model: Evidence for multiple learning modules in human decision making. *NeuroImage*. 246, (Feb. 2022), 118780.
- [156] Liepe, J. et al. 2014. A framework for parameter estimation and model selection from experimental data in systems biology using approximate Bayesian computation. *Nature protocols*. 9, 2 (Feb. 2014), 439–456.
- [157] Lin, L.-J. 1992. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*. 8, 3 (May 1992), 293–321.
- [158] Lindsay, G.W. 2020. Attention in Psychology, Neuroscience, and Machine Learning. *Frontiers in computational neuroscience*. 14, (Apr. 2020), 29.
- [159] Linson, A. et al. 2018. The Active Inference Approach to Ecological Perception: General Information Dynamics for Natural and Artificial Embodied Cognition. *Frontiers in robotics and AI*. 5, (Mar. 2018), 21.
- [160] Lintusaari, J. et al. 2018. ELFI: Engine for Likelihood-Free Inference. *Journal of machine learning research: JMLR*. 19, 16 (2018),

- 1–7.
- [161] Machado, M.C. et al. 2021. Temporal Abstraction in Reinforcement Learning with the Successor Representation. *arXiv [cs.LG]*.
 - [162] Mackintosh, N.J. 1974. The psychology of animal learning. 730, (1974).
 - [163] Maguire, E.A. et al. 2000. Navigation-related structural change in the hippocampi of taxi drivers. *Proceedings of the National Academy of Sciences of the United States of America*. 97, 8 (Apr. 2000), 4398–4403.
 - [164] Marcus, G. 2018. Innateness, AlphaZero, and Artificial Intelligence. *arXiv [cs.AI]*.
 - [165] Marković, D. et al. 2019. Predicting change: Approximate inference under explicit representation of temporal structure in changing environments. *PLoS computational biology*. 15, 1 (Jan. 2019), e1006707.
 - [166] Matsumoto, M. and Hikosaka, O. 2007. Lateral habenula as a source of negative reward signals in dopamine neurons. *Nature*. 447, 7148 (Jun. 2007), 1111–1115.
 - [167] Mayer, J. et al. 2010. Drawing an elephant with four complex parameters. *American journal of physics*. 78, 6 (Jun. 2010), 648–649.
 - [168] McDougle, S.D. and Collins, A.G.E. 2021. Modeling the influence of working memory, reinforcement, and action uncertainty on reaction time and choice during instrumental learning. *Psychonomic bulletin & review*. 28, 1 (Feb. 2021), 20–39.
 - [169] Meyer, D.R. 1951. Food deprivation and discrimination reversal learning by monkeys. *Journal of experimental psychology*. 41, 1 (Jan. 1951), 10–16.
 - [170] Milivojevic, B. et al. 2015. Insight reconfigures hippocampal-prefrontal memories. *Current biology: CB*. 25, 7 (Mar. 2015), 821–830.
 - [171] Miller, G.A. 1956. The magical number seven plus or minus two: some limits on our capacity for processing information. *Psychological review*. 63, 2 (Mar. 1956), 81–97.
 - [172] Miller, K.J. et al. 2017. Dorsal hippocampus contributes to model-based planning. *Nature neuroscience*. 20, 9 (Sep. 2017), 1269–1276.
 - [173] Minsky, M. 1988. *Society Of Mind*. Simon and Schuster.
 - [174] Mnih, V. et al. 2015. Human-level control through deep reinforcement learning. *Nature*. 518, 7540 (Feb. 2015), 529–533.
 - [175] Momennejad, I. 2020. Learning Structures: Predictive Representations, Replay, and Generalization. *Current Opinion in Behavioral Sciences*. 32, (Apr. 2020), 155–166.

- [176] Momennejad, I. 2021. Multi-scale Predictive Representations & Human-like RL. *Bernstein Conference* (2021).
- [177] Momennejad, I. et al. 2017. The successor representation in human reinforcement learning. *Nature human behaviour*. 1, 9 (Sep. 2017), 680–692.
- [178] Momennejad, I. and Howard, M.W. 2018. Predicting the Future with Multi-scale Successor Representations. *bioRxiv*.
- [179] Morris, A. et al. 2021. Generating Options and Choosing Between Them Depend on Distinct Forms of Value Representation. *Psychological science*. 32, 11 (Nov. 2021), 1731–1746.
- [180] Morris, G. et al. 2006. Midbrain dopamine neurons encode decisions for future action. *Nature neuroscience*. 9, 8 (Aug. 2006), 1057–1063.
- [181] Moser, E.I. et al. 2008. Place cells, grid cells, and the brain's spatial representation system. *Annual review of neuroscience*. 31, (2008), 69–89.
- [182] Moser, M.-B. et al. 2015. Place cells, grid cells, and memory. *Cold Spring Harbor perspectives in biology*. 7, 2 (Feb. 2015), a021808.
- [183] Moss 2022. Thought and Imagination. *Aristotle's On the Soul: A Critical Guide*. (2022).
- [184] Murdoch, W.J. et al. 2019. Definitions, methods, and applications in interpretable machine learning. *Proceedings of the National Academy of Sciences of the United States of America*. 116, 44 (Oct. 2019), 22071–22080.
- [185] Nassar, M.R. and Frank, M.J. 2016. Taming the beast: extracting generalizable knowledge from computational models of cognition. *Current opinion in behavioral sciences*. 11, (Oct. 2016), 49–54.
- [186] Natural intelligence in artificial creatures: 1995. <ftp://ftp.dca.fee.unicamp.br/pub/docs/gudwin/ia009/Balkenius.1995.Thesis.pdf>. Accessed: 2022-07-17.
- [187] Navarro, D.J. 2019. Between the Devil and the Deep Blue Sea: Tensions Between Scientific Judgement and Statistical Model Selection. *Computational Brain & Behavior*. 2, 1 (Mar. 2019), 28–34.
- [188] Nieh, E.H. et al. 2021. Geometry of abstract learned knowledge in the hippocampus. *Nature*. 595, 7865 (Jul. 2021), 80–84.
- [189] Niv, Y. 2019. Learning task-state representations. *Nature neuroscience*. 22, 10 (Oct. 2019), 1544–1553.
- [190] Niv, Y. et al. 2015. Reinforcement learning in multidimensional environments relies on attention mechanisms. *The Journal of neuroscience: the official journal of the Society for Neuroscience*. 35, 21 (May 2015), 8145–8157.

- [191] Niv, Y. 2009. Reinforcement learning in the brain. *Journal of mathematical psychology*. 53, 3 (Jun. 2009), 139–154.
- [192] Niv, Y. and Langdon, A. 2016. Reinforcement learning with Marr. *Current opinion in behavioral sciences*. 11, (Oct. 2016), 67–73.
- [193] Noever, D. et al. 2020. The Chess Transformer: Mastering Play using Generative Language Models. *arXiv [cs.AI]*.
- [194] Noorbakhsh, K. et al. 2021. Pretrained Language Models are Symbolic Mathematics Solvers too! *arXiv [stat.ML]*.
- [195] O'Doherty, J.P. et al. 2017. Learning, Reward, and Decision Making. *Annual review of psychology*. 68, (Jan. 2017), 73–100.
- [196] OpenAI et al. 2019. Dota 2 with Large Scale Deep Reinforcement Learning. *arXiv [cs.LG]*.
- [197] Palminteri, S. et al. 2017. The Importance of Falsification in Computational Cognitive Modeling. *Trends in cognitive sciences*. 21, 6 (Jun. 2017), 425–433.
- [198] Pavlov, I.P. 1927. *Conditional reflexes: an investigation of the physiological activity of the cerebral cortex*. Oxford Univ. Press.
- [199] Peer, E. et al. 2017. Beyond the Turk: Alternative platforms for crowdsourcing behavioral research. *Journal of experimental social psychology*. 70, (May 2017), 153–163.
- [200] Peer, M. et al. 2021. Structuring Knowledge with Cognitive Maps and Cognitive Graphs. *Trends in cognitive sciences*. 25, 1 (Jan. 2021), 37–54.
- [201] Perry, C.J. et al. 2016. Unexpected rewards induce dopamine-dependent positive emotion-like state changes in bumblebees. *Science*. 353, 6307 (Sep. 2016), 1529–1531.
- [202] Pfeifer, R. and Bongard, J. 2006. *How the Body Shapes the Way We Think: A New View of Intelligence*. MIT Press.
- [203] Pfeiffer, B.E. and Foster, D.J. 2013. Hippocampal place-cell sequences depict future paths to remembered goals. *Nature*. 497, 7447 (May 2013), 74–79.
- [204] Piray, P. et al. 2019. Hierarchical Bayesian inference for concurrent model fitting and comparison for group studies. *PLoS computational biology*. 15, 6 (Jun. 2019), e1007043.
- [205] Rad, M.S. et al. 2018. Toward a psychology of Homo sapiens: Making psychological science more representative of the human population. *Proceedings of the National Academy of Sciences of the United States of America*. 115, 45 (Nov. 2018), 11401–11405.
- [206] Radulescu, A. et al. 2019. Holistic Reinforcement Learning: The Role of Structure and Attention. *Trends in cognitive sciences*. 23, 4 (Apr. 2019), 278–292.
- [207] Rafiee, B. et al. 2022. From eye-blanks to state construction: Diagnostic benchmarks for online representation learning.

- Adaptive behavior.* (Apr. 2022), 10597123221085040.
- [208] Raine, N.E. and Chittka, L. 2012. No trade-off between learning speed and associative flexibility in bumblebees: a reversal learning test with multiple colonies. *PLoS one.* 7, 9 (Sep. 2012), e45096.
 - [209] Redgrave, P. et al. 1999. The basal ganglia: a vertebrate solution to the selection problem? *Neuroscience.* 89, 4 (1999), 1009–1023.
 - [210] Reingold, E.M. et al. 2001. Visual span in expert chess players: evidence from eye movements. *Psychological science.* 12, 1 (Jan. 2001), 48–55.
 - [211] Rescorla, R.A. and Wagner, A.R. 1972. A theory of Pavlovian conditioning: Variations in the effectiveness of reinforcement and nonreinforcement. *Classical conditioning II: Current research and theory.* 2, (1972), 64–99.
 - [212] Richards, B.A. et al. 2019. A deep learning framework for neuroscience. *Nature neuroscience.* 22, 11 (Nov. 2019), 1761–1770.
 - [213] Rigoux, L. et al. 2014. Bayesian model selection for group studies - revisited. *NeuroImage.* 84, (Jan. 2014), 971–985.
 - [214] Riquelme, J.L. and Gjorgjieva, J. 2021. Towards readable code in neuroscience. *Nature reviews. Neuroscience.* 22, 5 (May 2021), 257–258.
 - [215] Ritter, S. et al. 2018. Been There, Done That: Meta-Learning with Episodic Recall. *arXiv [stat.ML].*
 - [216] Rocca, S. et al. 2002. The Big Five Personality Factors and Personal Values. *Personality & social psychology bulletin.* 28, 6 (Jun. 2002), 789–801.
 - [217] Roesch, M.R. et al. 2007. Dopamine neurons encode the better option in rats deciding between differently delayed or sized rewards. *Nature neuroscience.* 10, 12 (Dec. 2007), 1615–1624.
 - [218] Rougier, N.P. et al. 2005. Prefrontal cortex and flexible cognitive control: rules without symbols. *Proceedings of the National Academy of Sciences of the United States of America.* 102, 20 (May 2005), 7338–7343.
 - [219] Russek, E.M. et al. 2017. Predictive representations can link model-based reinforcement learning to model-free mechanisms. *PLoS computational biology.* 13, 9 (Sep. 2017), e1005768.
 - [220] Sakai, K. 2008. Task set and prefrontal cortex. *Annual review of neuroscience.* 31, (2008), 219–245.
 - [221] Saxe, A. et al. 2021. If deep learning is the answer, what is the question? *Nature reviews. Neuroscience.* 22, 1 (Jan. 2021), 55–67.
 - [222] Schacter Daniel L and Addis Donna Rose 2007. The cognitive neuroscience of constructive memory: remembering the past and imagining the future. *Philosophical transactions of the Royal Society*

- of London. Series B, Biological sciences. 362, 1481 (May 2007), 773–786.
- [223] Schacter, D.L. et al. 2012. The future of memory: remembering, imagining, and the brain. *Neuron*. 76, 4 (Nov. 2012), 677–694.
 - [224] Schälte, Y. et al. 2022. pyABC: Efficient and robust easy-to-use approximate Bayesian computation. *arXiv [q-bio.QM]*.
 - [225] Schlagenhauf, F. et al. 2014. Striatal dysfunction during reversal learning in unmedicated schizophrenia patients. *NeuroImage*. 89, (Apr. 2014), 171–180.
 - [226] van de Schoot, R. et al. 2021. Bayesian statistics and modelling. *Nature Reviews Methods Primers*. 1, 1 (Jan. 2021), 1–26.
 - [227] Schuck, N.W. et al. 2018. Chapter 12 - A State Representation for Reinforcement Learning and Decision-Making in the Orbitofrontal Cortex. *Goal-Directed Decision Making*. R. Morris et al., eds. Academic Press. 259–278.
 - [228] Schuck, N.W. et al. 2016. Human Orbitofrontal Cortex Represents a Cognitive Map of State Space. *Neuron*. 91, 6 (Sep. 2016), 1402–1412.
 - [229] Schuck, N.W. and Niv, Y. 2019. Sequential replay of nonspatial task states in the human hippocampus. *Science*. 364, 6447 (Jun. 2019), 315978.
 - [230] Schultz, W. et al. 1997. A neural substrate of prediction and reward. *Science*. 275, 5306 (Mar. 1997), 1593–1599.
 - [231] Schultz, W. 2016. Dopamine reward prediction-error signalling: a two-component response. *Nature reviews Neuroscience*. 17, 3 (Mar. 2016), 183–195.
 - [232] Schultz, W. 2015. Neuronal Reward and Decision Signals: From Theories to Data. *Physiological reviews*. 95, 3 (Jul. 2015), 853–951.
 - [233] Schultz, W. et al. 1993. Responses of monkey dopamine neurons to reward and conditioned stimuli during successive steps of learning a delayed response task. *The Journal of neuroscience: the official journal of the Society for Neuroscience*. 13, 3 (Mar. 1993), 900–913.
 - [234] Schultz, W. et al. 2017. The phasic dopamine signal maturing: from reward via behavioural activation to formal economic utility. *Current opinion in neurobiology*. 43, (Apr. 2017), 139–148.
 - [235] Schulz, E. et al. 2020. Finding structure in multi-armed bandits. *Cognitive psychology*. 119, (Feb. 2020), 101261.
 - [236] Seger, C.A. and Miller, E.K. 2010. Category learning in the brain. *Annual review of neuroscience*. 33, (2010), 203–219.
 - [237] Seward, J.P. 1949. An experimental analysis of latent learning. *Journal of experimental psychology*. 39, 2 (Apr. 1949), 177–186.

- [238] Seymour, B. et al. 2012. Serotonin selectively modulates reward value in human decision-making. *The Journal of neuroscience: the official journal of the Society for Neuroscience*. 32, 17 (Apr. 2012), 5833–5842.
- [239] Sharpe, M.J. et al. 2019. An Integrated Model of Action Selection: Distinct Modes of Cortical Control of Striatal Decision Making. *Annual review of psychology*. 70, (Jan. 2019), 53–76.
- [240] Siljebråt, H. and Pickering, A. 2020. The Effect of State Representations in Sequential Sensory Prediction: Introducing the Shape Sequence Task. *CogSci* (2020).
- [241] Silver, D. et al. 2018. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*. 362, 6419 (Dec. 2018), 1140–1144.
- [242] Silver, D. et al. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature*. 529, 7587 (Jan. 2016), 484–489.
- [243] Silver, D. et al. 2017. Mastering the game of Go without human knowledge. *Nature*. 550, 7676 (Oct. 2017), 354–359.
- [244] Simon, D.A. and Daw, N.D. 2011. Neural correlates of forward planning in a spatial decision task in humans. *The Journal of neuroscience: the official journal of the Society for Neuroscience*. 31, 14 (Apr. 2011), 5526–5539.
- [245] Simons, D.J. and Chabris, C.F. 1999. Gorillas in our midst: sustained inattentional blindness for dynamic events. *Perception*. 28, 9 (1999), 1059–1074.
- [246] Sirois, S. et al. 2008. Précis of neuroconstructivism: how the brain constructs cognition. *The Behavioral and brain sciences*. 31, 3 (Jun. 2008), 321–31; discussion 331–56.
- [247] Spelke, E.S. and Kinzler, K.D. 2007. Core knowledge. *Developmental science*. 10, 1 (Jan. 2007), 89–96.
- [248] Stachenfeld, K.L. et al. 2017. The hippocampus as a predictive map. *Nature neuroscience*. 20, 11 (Nov. 2017), 1643–1653.
- [249] Stamatakis, A.M. and Stuber, G.D. 2012. Activation of lateral habenula inputs to the ventral midbrain promotes behavioral avoidance. *Nature neuroscience*. 15, 8 (Jun. 2012), 1105–1107.
- [250] Stan Development Team 2022. *Stan Modeling Language*.
- [251] Steingroever, H. et al. 2016. Bayes factors for reinforcement-learning models of the Iowa gambling task. *Decisions*. 3, 2 (Apr. 2016), 115–131.
- [252] Stephan, K.E. et al. 2009. Bayesian model selection for group studies. *NeuroImage*. 46, 4 (Jul. 2009), 1004–1017.
- [253] Steyvers, M. et al. 2009. A Bayesian analysis of human decision-making on bandit problems. *Journal of mathematical*

- psychology.* 53, 3 (Jun. 2009), 168–179.
- [254] Sugawara, M. and Katahira, K. 2021. Dissociation between asymmetric value updating and perseveration in human reinforcement learning. *Scientific reports.* 11, 1 (Feb. 2021), 3574.
- [255] Sunnåker, M. et al. 2013. Approximate Bayesian computation. *PLoS computational biology.* 9, 1 (Jan. 2013), e1002803.
- [256] Sutton, R.S. et al. 1999. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence.* 112, 1 (Aug. 1999), 181–211.
- [257] Sutton, R.S. 1991. Dyna, an Integrated Architecture for Learning, Planning, and Reacting. *SIGART Bull.* 2, 4 (Jul. 1991), 160–163.
- [258] Sutton, R.S. and Barto, A.G. 1998. *Reinforcement learning: An introduction.* MIT press Cambridge.
- [259] Sutton, R.S. and Barto, A.G. 2018. *Reinforcement Learning: An Introduction 2nd edition.* MIT Press, Cambridge, MA.
- [260] Sutton, R.S. and Barto, A.G. 1981. Toward a modern theory of adaptive networks: expectation and prediction. *Psychological review.* 88, 2 (Mar. 1981), 135–170.
- [261] Takahashi, Y. et al. 2008. Silencing the critics: understanding the effects of cocaine sensitization on dorsolateral and ventral striatum in the context of an actor/critic model. *Frontiers in neuroscience.* 2, 1 (Jul. 2008), 86–99.
- [262] The reward hypothesis: <http://incompleteideas.net/rrai.cs.ualberta.ca/RLAI/rewardhypothesis.html>. Accessed: 2019-02-23.
- [263] Thorndike, E.L. 1898. Animal intelligence: an experimental study of the associative processes in animals. *The Psychological Review: Monograph Supplements.* 2, 4 (1898), i.
- [264] Threlfell, S. et al. 2012. Striatal dopamine release is triggered by synchronized activity in cholinergic interneurons. *Neuron.* 75, 1 (Jul. 2012), 58–64.
- [265] Tik, M. et al. 2018. Ultra-high-field fMRI insights on insight: Neural correlates of the Aha!-moment. *Human brain mapping.* (Apr. 2018). DOI:<https://doi.org/10.1002/hbm.24073>.
- [266] Tobler, P.N. et al. 2003. Coding of predicted reward omission by dopamine neurons in a conditioned inhibition paradigm. *The Journal of neuroscience: the official journal of the Society for Neuroscience.* 23, 32 (Nov. 2003), 10402–10410.
- [267] Tokic, M. 2010. Adaptive ϵ -greedy Exploration in Reinforcement Learning Based on Value Differences. *KI 2010: Advances in Artificial Intelligence* (2010).
- [268] Tokic, M. and Palm, G. 2011. Value-Difference Based

- Exploration: Adaptive Control between Epsilon-Greedy and Softmax. *KI 2011: Advances in Artificial Intelligence*. J. Bach and S. Edelkamp, eds. Springer Berlin Heidelberg. 335–346.
- [269] Tolman, E.C. 1948. Cognitive maps in rats and men. *Psychological review*. 55, 4 (Jul. 1948), 189–208.
- [270] Toni, T. et al. 2009. Approximate Bayesian computation scheme for parameter inference and model selection in dynamical systems. *Journal of the Royal Society, Interface / the Royal Society*. 6, 31 (Feb. 2009), 187–202.
- [271] Trudel, N. et al. 2020. Polarity of uncertainty representation during exploration and exploitation in ventromedial prefrontal cortex. *Nature human behaviour*. (Aug. 2020). DOI:<https://doi.org/10.1038/s41562-020-0929-3>.
- [272] Tsao, A. et al. 2018. Integrating time from experience in the lateral entorhinal cortex. *Nature*. (Aug. 2018). DOI:<https://doi.org/10.1038/s41586-018-0459-6>.
- [273] Tsutsui, K.-I. et al. 2016. A dynamic code for economic object valuation in prefrontal cortex neurons. *Nature communications*. 7, (Sep. 2016), 12554.
- [274] Turner, B.M. and Van Zandt, T. 2012. A tutorial on approximate Bayesian computation. *Journal of mathematical psychology*. 56, 2 (Apr. 2012), 69–85.
- [275] Umbach, G. et al. 2020. Time cells in the human hippocampus and entorhinal cortex support episodic memory. *Proceedings of the National Academy of Sciences of the United States of America*. (Oct. 2020). DOI:<https://doi.org/10.1073/pnas.2013250117>.
- [276] Van Essen, D.C. et al. 1992. Information processing in the primate visual system: an integrated systems perspective. *Science*. 255, 5043 (Jan. 1992), 419–423.
- [277] Vaswani et al. 2017. Attention is all you need. *Advances in neural information processing systems*. (2017).
- [278] Vehtari, A. et al. 2015. Practical Bayesian model evaluation using leave-one-out cross-validation and WAIC. *arXiv [stat.CO]*.
- [279] Vincent, B.T. 2016. Hierarchical Bayesian estimation and hypothesis testing for delay discounting tasks. *Behavior research methods*. 48, 4 (Dec. 2016), 1608–1620.
- [280] Virtanen, P. et al. 2020. SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature methods*. 17, 3 (Mar. 2020), 261–272.
- [281] Voon, V. et al. 2015. Disorders of compulsivity: a common bias towards learning habits. *Molecular psychiatry*. 20, 3 (Mar. 2015), 345–352.
- [282] Wagenmakers, E.-J. et al. 2018. Bayesian inference for

- psychology. Part I: Theoretical advantages and practical ramifications. *Psychonomic bulletin & review*. 25, 1 (Feb. 2018), 35–57.
- [283] Wagner, M.J. et al. 2017. Cerebellar granule cells encode the expectation of reward. *Nature*. 544, 7648 (Apr. 2017), 96–100.
 - [284] Wang, J.X. et al. 2018. Prefrontal cortex as a meta-reinforcement learning system. *Nature neuroscience*. (May 2018). DOI:<https://doi.org/10.1038/s41593-018-0147-8>.
 - [285] Watanabe, S. 2013. A widely applicable Bayesian information criterion. *Journal of machine learning research: JMLR*. (2013).
 - [286] Watkins, C.J.C.H. and Dayan, P. 1992. Q-learning. *Machine learning*. 8, 3 (May 1992), 279–292.
 - [287] Webb, B. 2001. Can robots make good models of biological behaviour? *The Behavioral and brain sciences*. 24, 6 (Dec. 2001), 1033–50; discussion 1050–94.
 - [288] Whittington, J.C.R. et al. 2020. The Tolman-Eichenbaum Machine: Unifying Space and Relational Memory through Generalization in the Hippocampal Formation. *Cell*. 183, 5 (Nov. 2020), 1249–1263.e23.
 - [289] Wiecki, T.V. et al. 2013. HDDM: Hierarchical Bayesian estimation of the Drift-Diffusion Model in Python. *Frontiers in neuroinformatics*. 7, (Aug. 2013), 14.
 - [290] Wilson, M. 2002. Six views of embodied cognition. *Psychonomic bulletin & review*. 9, 4 (Dec. 2002), 625–636.
 - [291] Wilson, R. and Collins, A. 2019. Ten simple rules for the computational modeling of behavioral data. *PsyArxiv*. (2019).
 - [292] Wilson, R.C. et al. 2014. Humans use directed and random exploration to solve the explore-exploit dilemma. *Journal of experimental psychology. General*. 143, 6 (Dec. 2014), 2074–2081.
 - [293] Wilson, R.C. et al. 2014. Orbitofrontal cortex as a cognitive map of task space. *Neuron*. 81, 2 (Jan. 2014), 267–279.
 - [294] Wilson, R.C. and Niv, Y. 2011. Inferring relevance in a changing world. *Frontiers in human neuroscience*. 5, (2011), 189.
 - [295] Wise, R.A. 2004. Dopamine, learning and motivation. *Nature reviews. Neuroscience*. 5, 6 (Jun. 2004), 483–494.
 - [296] Worthy, D.A. et al. 2007. Regulatory fit effects in a choice task. *Psychonomic bulletin & review*. 14, 6 (Dec. 2007), 1125–1132.
 - [297] Wunderlich, K. et al. 2012. Mapping value based planning and extensively trained choice in the human brain. *Nature neuroscience*. 15, 5 (Mar. 2012), 786–791.
 - [298] Yamins, D.L.K. and DiCarlo, J.J. 2016. Using goal-driven deep learning models to understand sensory cortex. *Nature neuroscience*. 19, 3 (Mar. 2016), 356–365.

- [299] Yau, J.O.-Y. and McNally, G.P. 2019. Rules for aversive learning and decision-making. *Current Opinion in Behavioral Sciences*. 26, (Apr. 2019), 1–8.
- [300] Yechiam, E. et al. 2010. Adapted to explore: reinforcement learning in Autistic Spectrum Conditions. *Brain and cognition*. 72, 2 (Mar. 2010), 317–324.
- [301] Yoo, A.H. and Collins, A.G.E. 2022. How Working Memory and Reinforcement Learning Are Intertwined: A Cognitive, Neural, and Computational Perspective. *Journal of cognitive neuroscience*. 34, 4 (Mar. 2022), 551–568.
- [302] Zhang, Z. et al. 2018. A neural network model for the orbitofrontal cortex and task space acquisition during reinforcement learning. *PLoS computational biology*. 14, 1 (Jan. 2018), e1005925.
- [303] Zhou, J. et al. 2019. Complementary Task Structure Representations in Hippocampus and Orbitofrontal Cortex during an Odor Sequence Task. *Current biology: CB*. (Sep. 2019). DOI:<https://doi.org/10.1016/j.cub.2019.08.040>.