Computational modelling of Human Behavioural Data

Peggy Seriès, IANC
Informatics, University of Edinburgh, UK

pseries@inf.ed.ac.uk

CCN Lecture 12

What is comp. modelling of behavioural data ?

Goal: use precise mathematical models to make better sense of behavioural data.

- **Why behaviour?** Modelling behaviour offers the possibility for linking neurobiological hypotheses with cognitive function or dysfunction.

- **What Data?** choices, reaction times, but can also be eye movements or other observable data

- **Why models?** Precisely specify assumptions about mechanisms, “algorithmic hypotheses”. Models are used for simulations, parameter estimation, model comparison, latent variable inference.

A Road Map of Good Practice

(This lecture is based on this paper)

eLIFE elifesciences.org

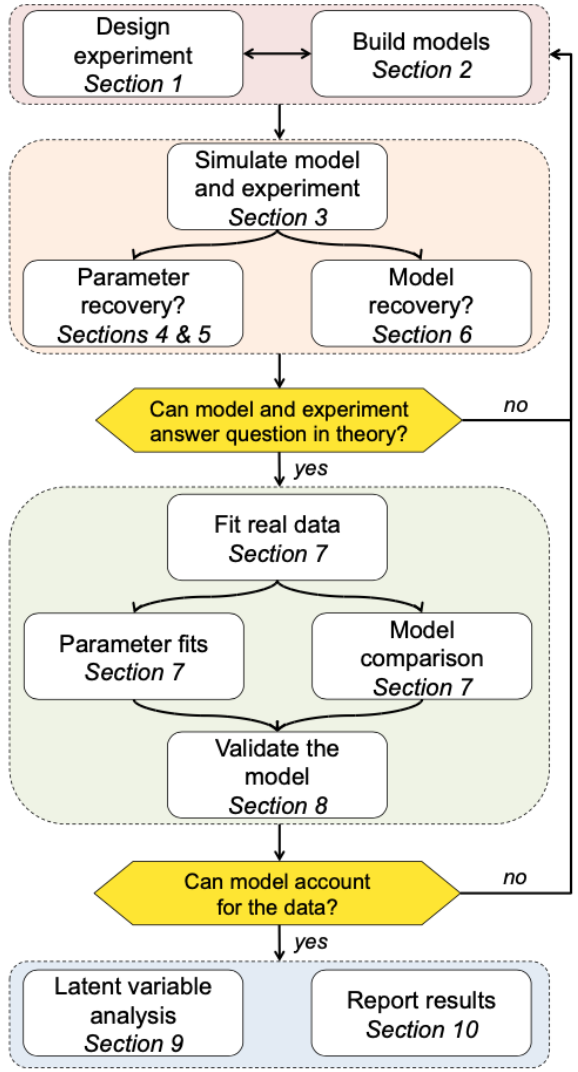
REVIEW ARTICLE

Ten simple rules for the computational modeling of behavioral data

Robert C Wilson[1,2†*], Anne GE Collins[3,4†*]

[1]Department of Psychology, University of Arizona, Tucson, United States; [2]Cognitive Science Program, University of Arizona, Tucson, United States; [3]Department of Psychology, University of California, Berkeley, Berkeley, United States; [4]Helen Wills Neuroscience Institute, University of California, Berkeley, Berkeley, United States

Abstract Computational modeling of behavior has revolutionized psychology and neuroscience. By fitting models to experimental data we can probe the algorithms underlying behavior, find neural correlates of computational variables and better understand the effects of drugs, illness and interventions. But with great power comes great responsibility. Here, we offer ten simple rules to ensure that computational modeling is used with care and yields meaningful insights. In particular,

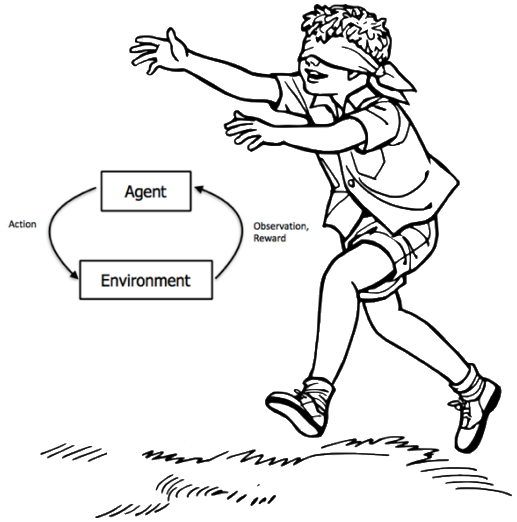


Wilson and Collins. eLife 2019;8:e49547.

1. Design a good experiment

- What are the **questions** we want to answer?

  (e.g. How do individuals form an internal model of the world? Are depressed individuals impaired in reward learning?)



- Formalise the question.
- **Do we need models** to help answer these questions?
- Do you expect signature of the targeted process to be evident from simple statistics of the data? (else modelling might be pointless)
- Even the best models cannot salvage bad experiments

Example: multi-armed bandit

Question: Are depressed individuals impaired in reward learning?

How do they learn to maximize their rewards in a case where the most rewarding choice is initially unknown?

e.g. K=2 slot machines, binary reward. T=1000 choices for each, P(r|A)=0.2, P(r|B)=0.8.

2 groups: 1 healthy controls vs Major Depressive Disorder patients.

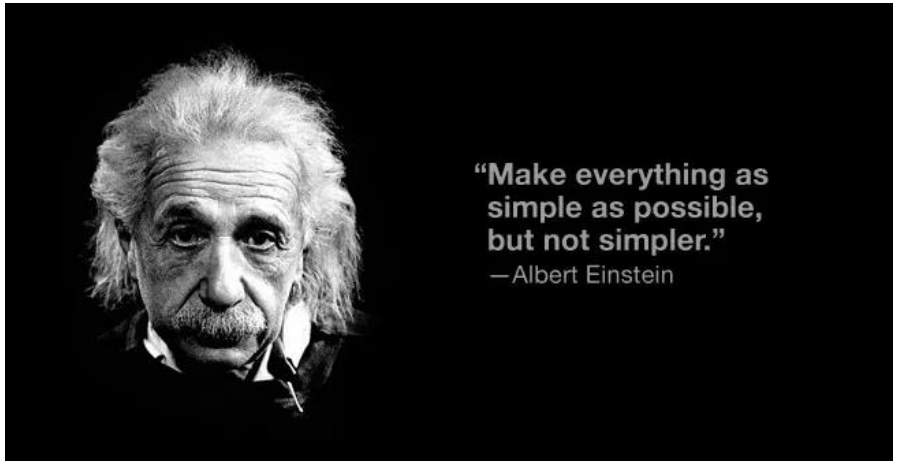
Are patients slower to learn? worse? if so, can we dissect what makes them so?

2. Design Good Models/ Define Model Space

- Is the model's aim to be **descriptive**/ a summary of the data? **explanatory**/ mechanistic, **optimal**/normative?

- A computational model should be **interpretable** (as much as possible)
- Models should capture all the hypotheses you want to test.

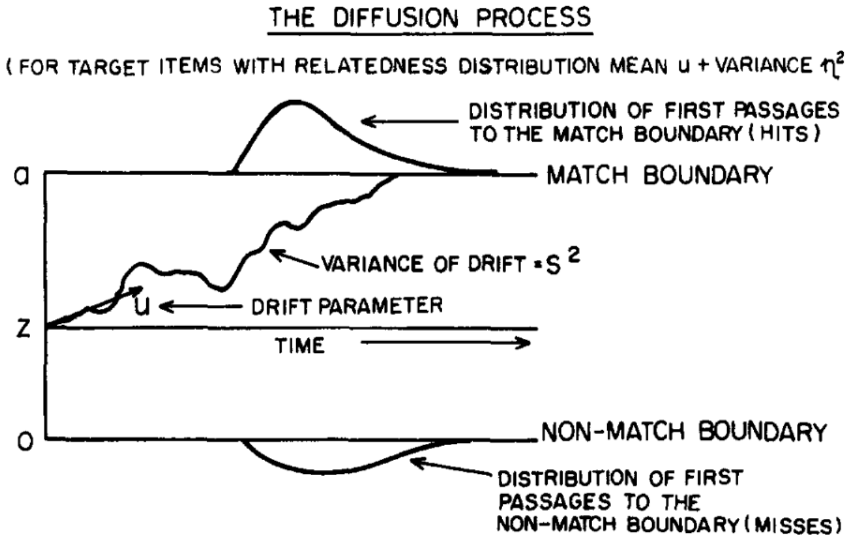
  Inference is conditional on model space!
- Beware of **bias** in how you choose/assess your model.



Reactions times are of interest:

Drift diffusion model (DDM)

Allows joint modelling of reaction times and accuracy

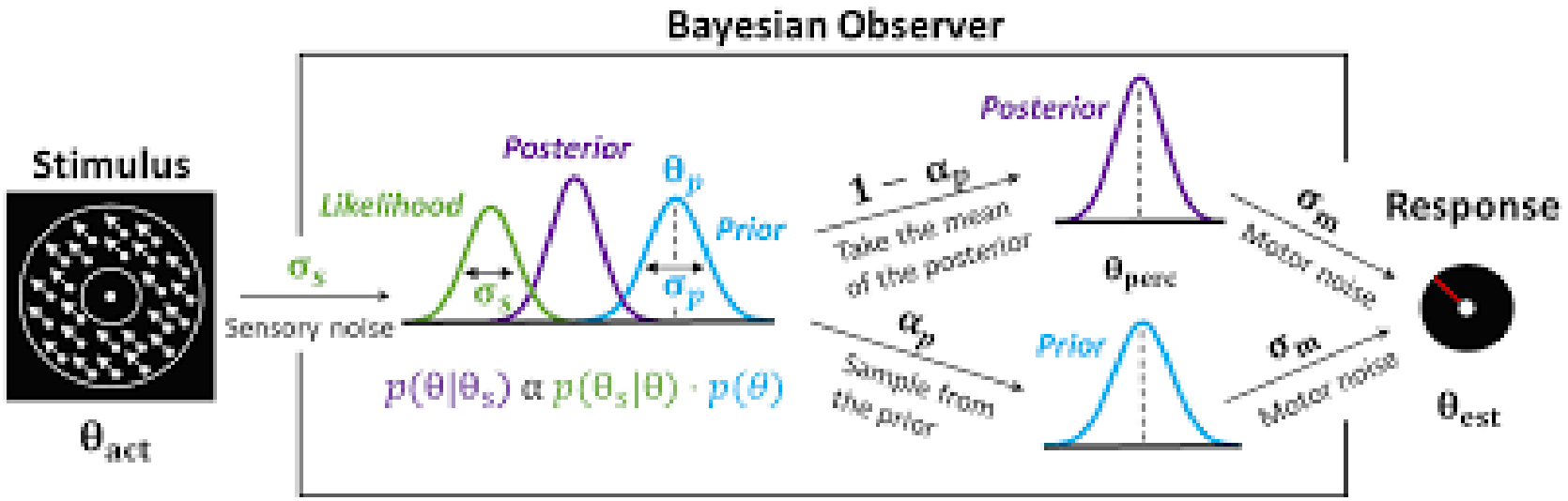


Ratcliff 1978, Psy. Rev.

(a family of models)

Comparison with benchmark performance or notion of prior is of interest:

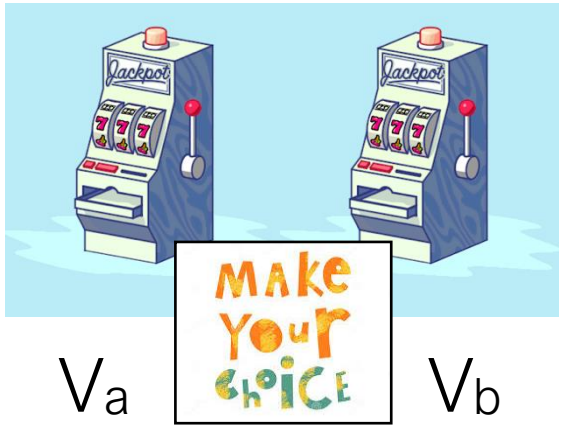
Bayesian Models



(a family of models)

Reinforcement Learning Models

- Make a choice
  - Based on "internal values"
- Observe an outcome (reinforcer)
  - Often probabilistic
  - Generates prediction error
- Update internal values



$$V(t+1) = V(t) + \varepsilon \times (r(t) - V(t))$$

[Rescorla-Wagner]

- Decide between two options

$$p(a \mid V, \theta) = \frac{1}{1 + \exp(-\beta \times (V_a - V_b))}$$

Probability of choosing machine a [Softmax]

(a family of models)

2. Define Model Space (Example)

- **Model 1**: random responding with some bias for one or the other (one (bias) parameter);

2. Define Model Space (Example)

- **Model 1**: random responding with some bias for one or the other (one (bias) parameter);
- **Model 2**: win-stay lose-shift (one (noise) parameter);

2. Define Model Space (Example)

- **Model 1**: random responding with some bias for one or the other (one (bias) parameter);
- **Model 2**: win-stay lose-shift (one (noise) parameter);
- **Model 3:** “vanilla” Rescorla-Wagner model:

$$Q_{t+1}^k = Q_t^k + \alpha(r_t - Q_t^k)$$

learn the values for each machine based on reward observed at each trial

[Rescorla-Wagner]

$$p_t^k = \frac{\exp(\beta Q_t^k)}{\sum_{i=1}^{K} \exp(\beta Q_t^i)}$$

compute probability of making a choice for one machine at each trial, based on the learned values; beta controls how deterministic/noisy this choice is.

2 free parameters, learning rate alpha and softmax parameter beta

2. Define Model Space (Example)

- **Model 1**: random responding with some bias for one or the other (one (bias) parameter);
- **Model 2**: win-stay lose-shift (one (noise) parameter);
- **Model 3:** “vanilla” Rescorla-Wagner model:

$$Q_{t+1}^k = Q_t^k + \alpha(r_t - Q_t^k)$$

learn the values for each machine based on reward observed at each trial

$$p_t^k = \frac{\exp(\beta Q_t^k)}{\sum_{i=1}^{K} \exp(\beta Q_t^i)}$$

compute probability of making a choice for one machine at each trial, based on the learned values; beta controls how deterministic/noisy this choice is.

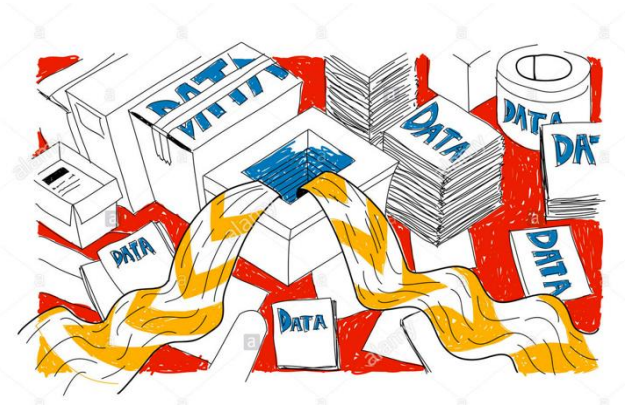
2 free parameters, learning rate alpha and softmax parameter beta

- **Model 4+:** variants on Rescorla-Wagner model that account to test for different hypotheses, e.g. role of sensitivity to reward? memory? different learning rates for reward vs punishment etc..

3. Simulate, Simulate, Simulate

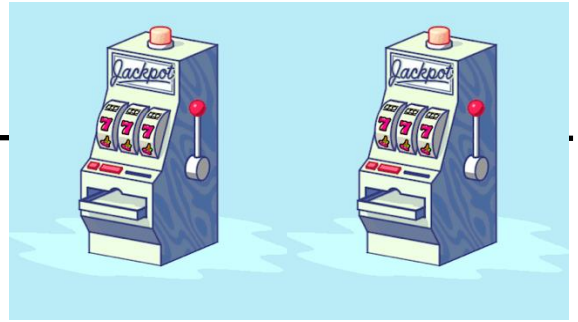
- Once you have a model,

  you can **create fake/surrogate/artificial data** and set up all your fitting pipeline.

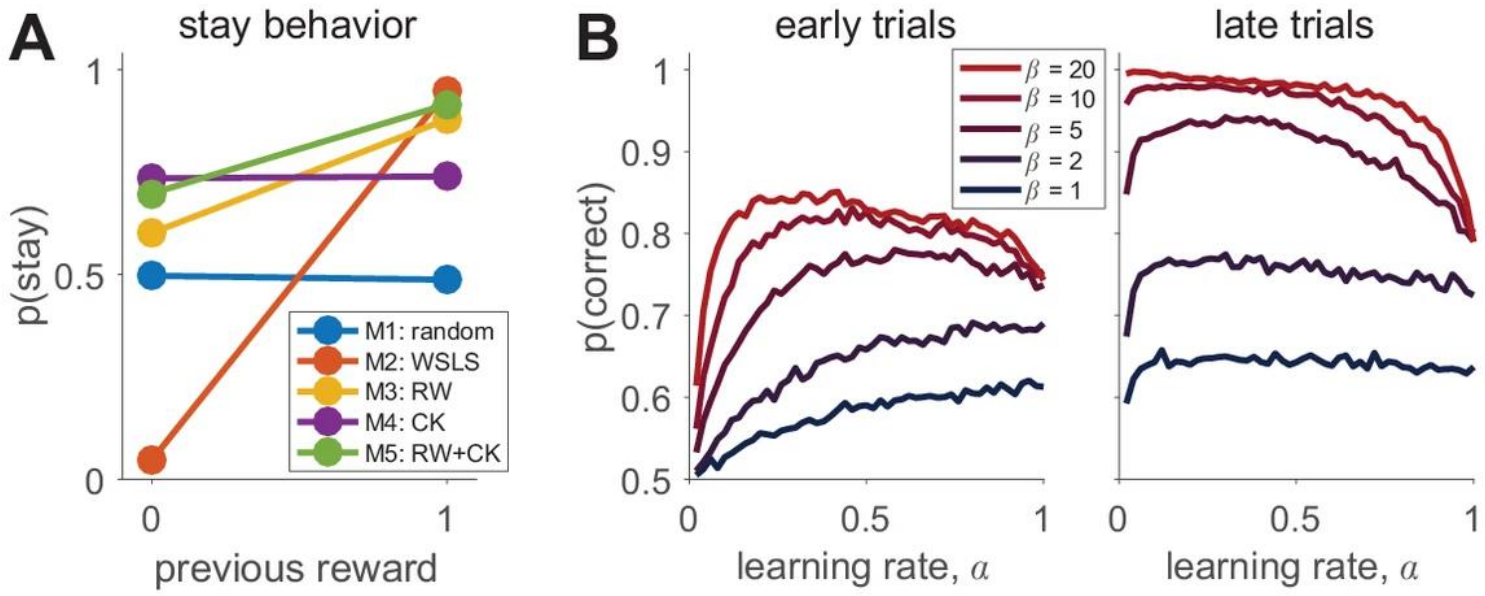


- Define model-independent **measures** that capture key aspects of the processes you are trying to model (e.g. % correct, % stay after reward)
- Simulate the model across a range of **parameter values;**
- Visualise the simulated behaviour of **different models**, hopefully they show different patterns/make different predictions.

3. Simulate, Simulate, Simulate (Example)



Example: visualising the **win-stay-lose shift behaviour** for the different models and **% correct**, as a function of learning rate.
—> different patterns of p(stay) for different models, and shows that learning rate affects both the speed of learning and asymptotic performance



4. What model parameters fit the data best?

The simple way to estimate the value of the parameters that best describe our data is using a **maximum-likelihood approach**.

Our goal: find the parameter values of model m, that maximize the likelihood of the observed data, d, given the parameters

$$LL = \log p(d_{1:T}|\theta_m, m) = \sum_{t=1}^{T} \log p(c_t|d_{1:t-1}, s_t, \theta_m, m)$$

4. What model parameters fit the data best?

The simple way to estimate the value of the parameters that best describe our data is using a **maximum-likelihood approach**.

Our goal: find the parameter values of model m, that maximize the likelihood of the observed data, d, given the parameters

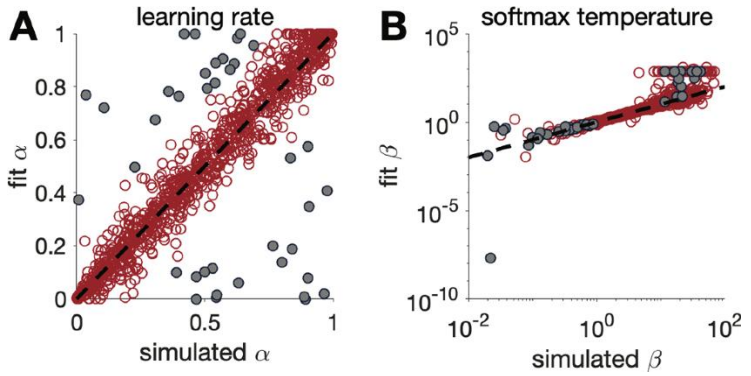
$$LL = \log p(d_{1:T}|\theta_m, m) = \sum_{t=1}^{T} \log p(c_t|d_{1:t-1}, s_t, \theta_m, m)$$

- In practice, find minimum of -LL using **gradient descent**, e.g. using Matlab `fmincon` or Python `scipy.optimize` package or the `optim function` in R
- Tips: Be sure your initial conditions give finite LogL. Beware rounding errors, zeros and infinities. Be careful with constraints on parameters.
- Beware local minima: run fitting procedure multiple times with random initial conditions, recording the best fitting log-likelihood for each run

5. Check that you can recover your parameters (parameter recovery)

Simulate Fake Data with fixed parameters → Fit models and parameters → Compare fitted parameters with values that generated the data

Parameter recovery for the Rescorla Wagner model (model 3) in the bandit task with 1000 trials. Grey dots in both panels correspond to points where parameter recovery for a is bad.

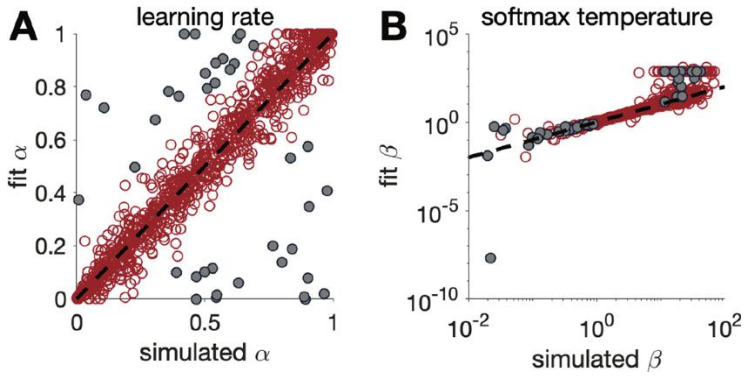


Parameter Recovery

5. Check that you can recover your parameters (parameter recovery)

Simulate Fake Data with fixed parameters → Fit models and parameters → Compare fitted parameters with values that generated the data

Parameter recovery for the Rescorla Wagner model (model 3) in the bandit task with 1000 trials. Grey dots in both panels correspond to points where parameter recovery for a is bad.



- Make sure your recovered parameters are in the right range
- Plot the **correlations** between simulated and recovered parameters
- Make sure the recovery process does not introduce correlations between parameters
- Remember that even successful parameter recovery represents a best-case scenario.

6. What Model fits the data best? (Model Comparison)

- Model comparison's goal is to determine which model, out of a set of possible models, is most likely to have generated the data.

- Comparing the **log-likelihoods of each model at the best fitting parameter settings** doesn't work, because models with more free parameters would be favoured (overfitting).

  Many other options (e.g. AIC) but a popular/simple is the **Bayes Information Criterion, BIC**, which has an explicit penalty for **number of free parameters** (k_m).

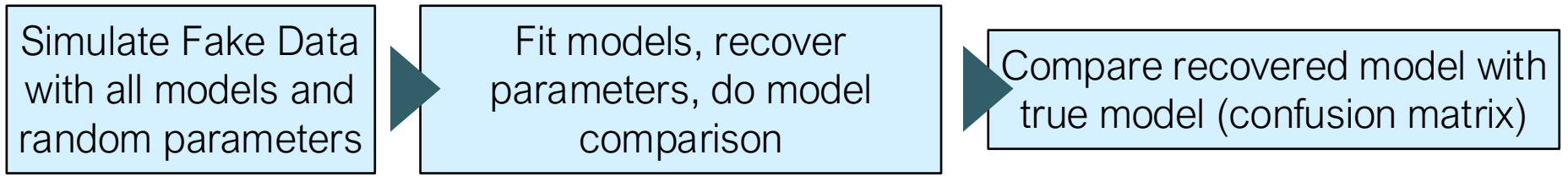
$$BIC = -2\log\hat{LL} + k_m \log(T)$$

T is the number of data points (trials).

- The model with the smallest BIC wins.

6. Model Recovery

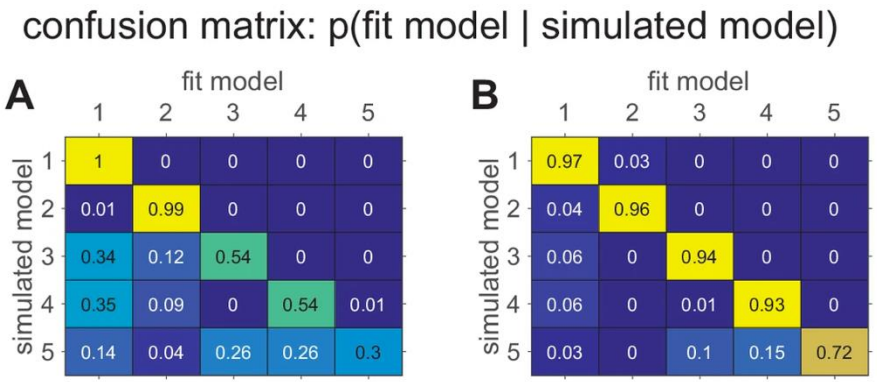
Check that your model comparison process gives sensible results for simulated data.



6. Model Recovery

Check that your model comparison process gives sensible results for simulated data.

Simulate Fake Data with all models and random parameters → Fit models, recover parameters, do model comparison → Compare recovered model with true model (confusion matrix)

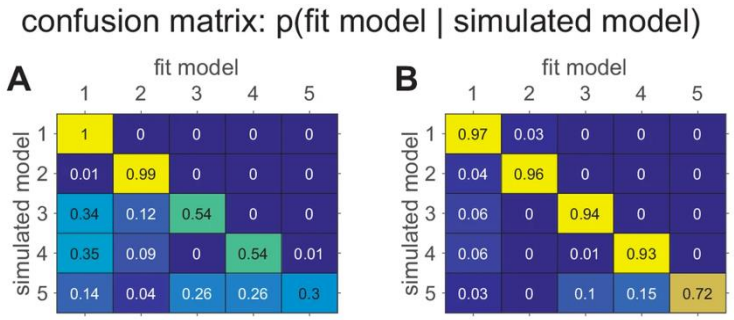


The confusion matrix can depend on the parameters. E.g. In panel B, all of the softmax parameters β and βc were increased by 1.

6. Model Recovery

Check that your model comparison process gives sensible results for simulated data.

Simulate Fake Data with all models and random parameters → Fit models, recover parameters, do model comparison → Compare recovered model with true model (confusion matrix)



- In a perfect world, the confusion matrix will be the identity matrix, but in practice, not always the case.
- compare different measures of model comparison (e.g. AIC vs BIC)
- careful w/ choice of parameters when plotting confusion matrix
- Elephant in the room: all those models you never considered ..

7. Run the experiment; collect data and analyse

- now ready to collect data!

7. Run the experiment; collect data and analyse

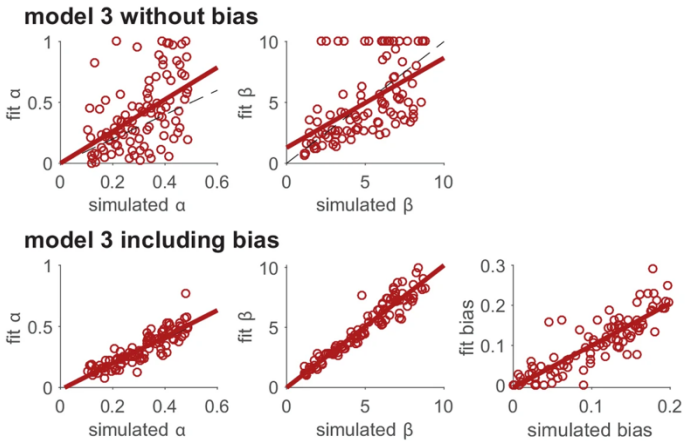
- now ready to collect data!

- **model-independent analysis**, comparison with expected patterns under different scenarios. Are the processes of interest engaged?
- **Model fitting**. How do the models fit the data?
- Might be necessary to improve/ extend model space to account for some features of the data that maybe unimportant theoretically, e.g. a systematic bias that affects fitting

8-9. Validate (at least) the winning model and Analyse

- Model comparison is **relative: the best fitting model might still be a bad model for the data!**
- Need to assess **absolute** quality of fit
- Simulate the winning model with best fitting parameters and see if it accounts for main features of the data.
- If this is not the case: back to drawing board! Possibly come up with a better model, or (disaster!) redesign the task..

E.g. we find that there is a systematic bias in the data to preferring the left slot machine, independently of current p(reward)



e.g. there's a side bias in the (artificial) data but we try to fit without accounting for it

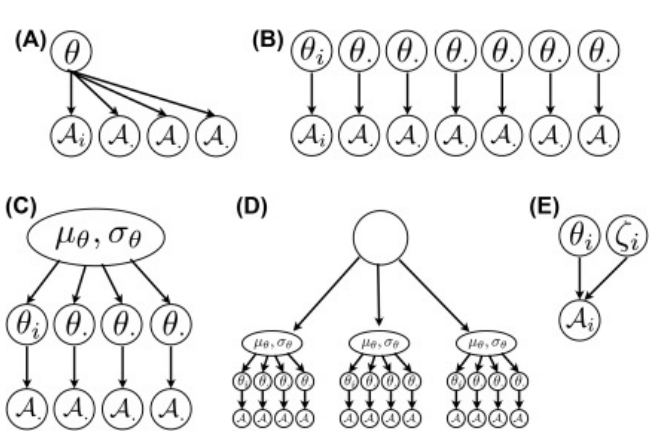
8-9. Validate (at least) the winning model and Analyse

- Model comparison is **relative: the best fitting model might still be a bad model for the data!**
- Need to assess **absolute** quality of fit
- Simulate the winning model with best fitting parameters and see if it accounts for main features of the data.
- If this is not the case: back to drawing board! Possibly come up with a better model, or (disaster!) redesign the task..

Model-dependent analyses should only be performed on the winning model, after researchers are satisfied that the model captures the behaviour.

A Rapidly Developing Field ..

- The standards are improving..
- More sophisticated methods are being developed, e.g.
- Use MAP instead of ML: include prior information on parameter values
- Hyper-parameters that are estimated at the same time as individual parameters (Hierarchical estimation)



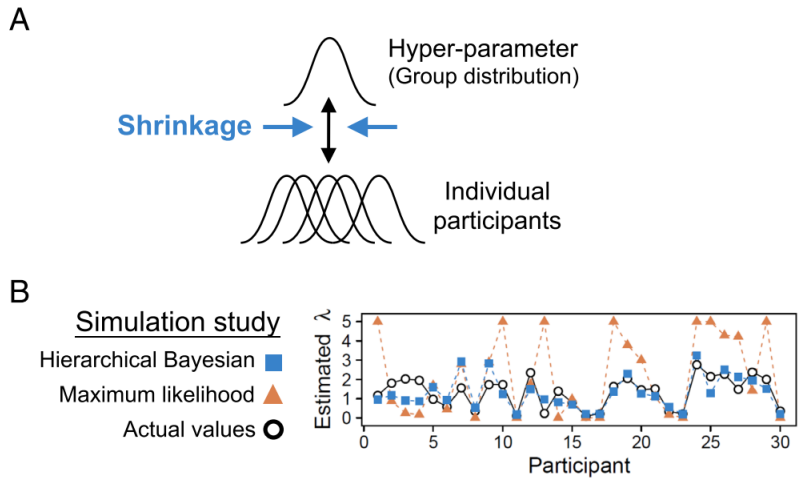
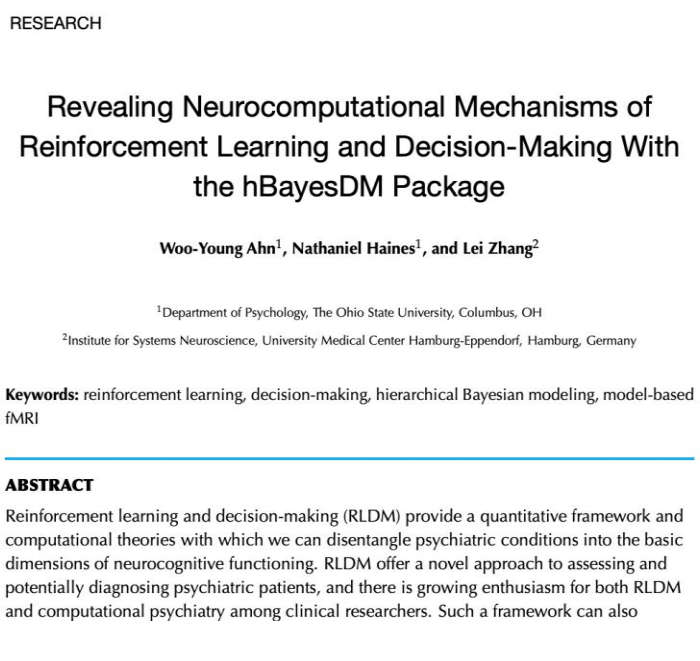


Figure 2. (A) A schematic illustration of hierarchical Bayesian analysis (HBA). In this example, the individual parameters are assumed to come from a group (hyper)parameter. (B) Results of a parameter recovery study (Ahn et al., 2011) between HBA and maximum likelihood estimation.

A Rapidly Developing Field

$$\log p\,(\theta_m \mid d_{1:T}, m),$$

- nowadays Hierarchical Bayesian estimation is preferred
- approximate full posterior using sampling approaches (such as MCMC);
- Packages are being offered like hBayesDM (Ahn et al CP, 2017)
- Combine (latent variables of) models with other measures (e.g. fMRI).



RESEARCH

Revealing Neurocomputational Mechanisms of Reinforcement Learning and Decision-Making With the hBayesDM Package

Woo-Young Ahn[1], Nathaniel Haines[1], and Lei Zhang[2]

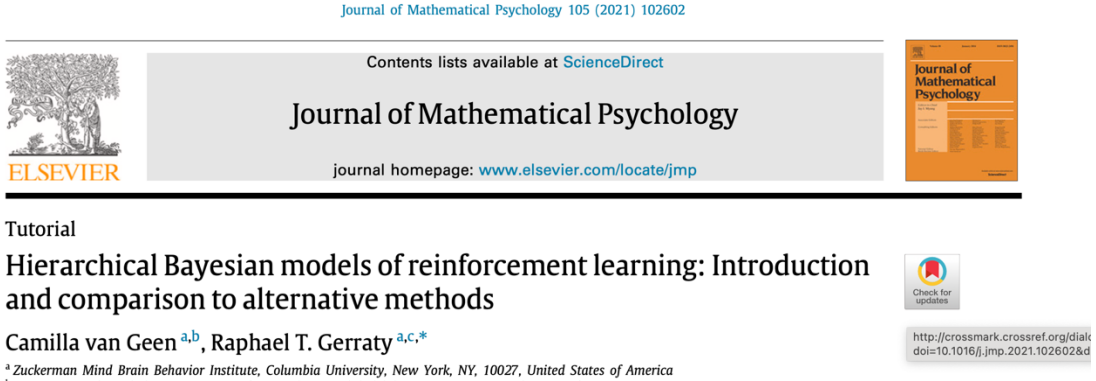
[1]Department of Psychology, The Ohio State University, Columbus, OH

[2]Institute for Systems Neuroscience, University Medical Center Hamburg-Eppendorf, Hamburg, Germany

Keywords: reinforcement learning, decision-making, hierarchical Bayesian modeling, model-based fMRI

ABSTRACT

Reinforcement learning and decision-making (RLDM) provide a quantitative framework and computational theories with which we can disentangle psychiatric conditions into the basic dimensions of neurocognitive functioning. RLDM offer a novel approach to assessing and potentially diagnosing psychiatric patients, and there is growing enthusiasm for both RLDM and computational psychiatry among clinical researchers. Such a framework can also



Journal of Mathematical Psychology 105 (2021) 102602

Contents lists available at ScienceDirect

Journal of Mathematical Psychology

journal homepage: www.elsevier.com/locate/jmp

Tutorial

Hierarchical Bayesian models of reinforcement learning: Introduction and comparison to alternative methods

Camilla van Geen [a,b], Raphael T. Gerraty [a,c,*]

[a] Zuckerman Mind Brain Behavior Institute, Columbia University, New York, NY, 10027, United States of America
[b] Department of Psychology, University of Pennsylvania, Philadelphia, PA, 19104, United States of America
[c] Center for Science and Society, Columbia University, New York, NY, 10027, United States of America

http://crossmark.crossref.org/dialo
doi=10.1016/j.jmp.2021.102602&d

ARTICLE INFO

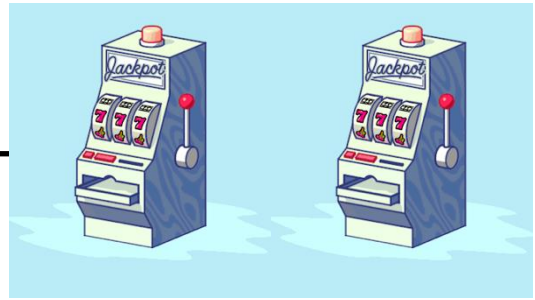
Article history:
Received 16 November 2020
Received in revised form 13 September 2021
Accepted 14 September 2021
Available online 15 October 2021

Keywords:
Reinforcement learning
Bayesian statistics
Model comparison

ABSTRACT

Reinforcement learning models have been used extensively to capture learning and decision-making processes in humans and other organisms. One essential goal of these computational models is the generalization to new sets of observations. Extracting parameters that can reliably predict out-of-sample data can be difficult, however. The use of prior distributions to regularize parameter estimates has been shown to help remedy this issue. While previous research has suggested that empirical priors estimated from a separate dataset improve predictive accuracy, this paper outlines an alternate method for the derivation of empirical priors: hierarchical Bayesian modeling. We provide a detailed introduction to this method, and show that using hierarchical models to simultaneously extract and impose empirical priors leads to better out-of-sample prediction while being more data efficient.

Example code in Python



Example: define a model

- Assume participants keep track of value of each stimulus, update values via prediction errors
  - Note that here we include a reward sensitivity parameter (rho)

$$V_{t+1}^{(s)} = V_t^{(s)} + \varepsilon(\rho \times r_t - V_t^{(s)})$$

$$p(a = A|V^{(A)}, V^{(B)}) = \frac{1}{1 + \exp(-(V^{(A)} - V^{(B)}))}$$

Simulate the model

```python
def model_simulate(theta):
    eps, rho = theta[0], theta[1]
    V = np.array([0.0, 0.0])  # Initialize values for two choices
    data = {"choices": [], "rewards": []}

    for t in range(100):
        p_choose_B = 1 / (1 + np.exp(-(V[1] - V[0])))  # Softmax-like probability
        c = 1 + (p_choose_B > np.random.rand())  # Choose 1 or 2

        if c == 1:
            r = 0.8 > np.random.rand()  # Reward probability for choice 1
        else:
            r = 0.2 > np.random.rand()  # Reward probability for choice 2

        V[c - 1] = V[c - 1] + eps * (rho * r - V[c - 1])  # Update value

        data["choices"].append(c)
        data["rewards"].append(r)

    return data
```

% simulate data for parameters theta

% Values initialised at 0

% for all trials

% compute prob choice given values

% draw a choice

% if choice 1 draw reward with prob 80%

% if choice 2 draw reward with prob 20%

% update values based on reward

% record choices and reward

Define the Log Likelihood

```
def model_neg_log_likelihood(data, theta):
    eps = theta[0]
    rho = np.exp(theta[1])  # Applying exponential transformation to rho
    V = np.array([0.0, 0.0])  # Initialize values for two choices
    num_trials = len(data["choices"])
    choice_probabilities = np.empty(num_trials)  # Initialize choice probabilities

    for t in range(num_trials):
        c = data["choices"][t]  # Get choice made
        c_alt = 1 if c == 2 else 2  # Alternate choice index (1-based indexing)

        # Compute probability of choosing c using softmax
        p_choose_c = 1 / (1 + np.exp(-(V[c - 1] - V[c_alt - 1])))
        choice_probabilities[t] = p_choose_c

        # Update value of chosen action
        V[c - 1] = V[c - 1] + eps * (rho * data["rewards"][t] - V[c - 1])

    # Compute negative log-likelihood
    nll = -np.sum(np.log(choice_probabilities))

    return nll
```

% for all trials

% compute probability of choosing c

% c= which target was chosen

% update V

% sum all the log of choice probability

Find best-fitting parameters

```
import numpy as np
from scipy.optimize import minimize

data = model_simulate([0.4, 5.5])
f = lambda theta: model_neg_log_likelihood(data, theta)

theta_start = np.array([0, 0])
result = minimize(f, theta_start, method="BFGS")

theta_opt = result.x
theta_opt[1] = np.exp(theta_opt[1])
```

%1 - simulate the data with fixed parameters
%2 - define NLL
%3 - initialise parameters to fit
%4 - find parameters (eps, rho) that best fit the data
%5 sometimes useful to transform some parameters by taking ln or exp

Conclusions

- Modelling of behavioural data is a way to **test hypotheses regarding deviations from optimal and individual differences,** e.g. in psychopathology

- To do things well, a number of steps are needed, including defining the **hypotheses**, simulating models before data collection and checking for **parameter and model recovery**. After data collection, check that the winning model provides a satisfactory description of the data.
  The best models can't salvage a bad experiment.

- In practise, many methodological issues.
- A field in evolution and standards increasing.