



Bayesian Statistics and Bayesian Cognitive Modeling (BayesCog)

Dr. Lei Zhang

Adaptive Learning Psychology & Neuroscience (ALPN) Lab
Centre for Human Brain Health, School of Psychology
University of Birmingham



lei-zhang.net
@lei_zhang_lz



https://github.com/lei-zhang/BayesCog_Part2

UKE Hamburg, 12/2023



Schedule

Oct. 2023

Statistical modeling

- Introduction
- R Basics
- Probability Basics
- Bayes' theorem
- MCMC and Stan
- Single-Parameter Model – Binomial Model
- Multiple-Parameter Model – Linear Regression
- Inference, Posterior Predictive Check

Dec. 2023

Cognitive modeling

- Reinforcement Learning Model
- Hierarchical Models
- Optimizing Stan Codes
- Model Comparison
- Stan Style Tip and Debugging
- Model-Based fMRI
- Capstone Project: RLDDM modeling (?)

About me

- Current: Associate Professor
- Postdoc @ [SCAN-Unit](#), w/ [Claus Lamm](#)
- Ph.D. Cognitive computational neuroscience, w/ [Jan Gläscher](#)
- M.Sc. Cognitive neuroscience
- B.Sc. Psychology



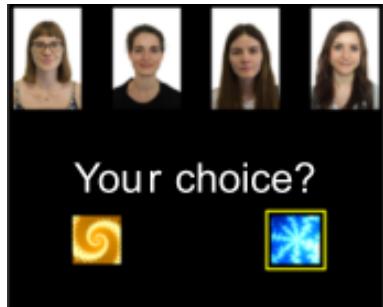
BASQUE CENTER
ON COGNITION, BRAIN
AND LANGUAGE



My research:

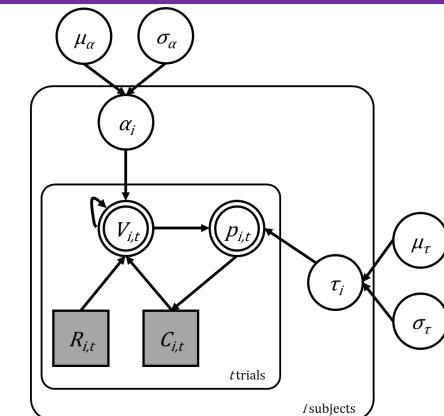
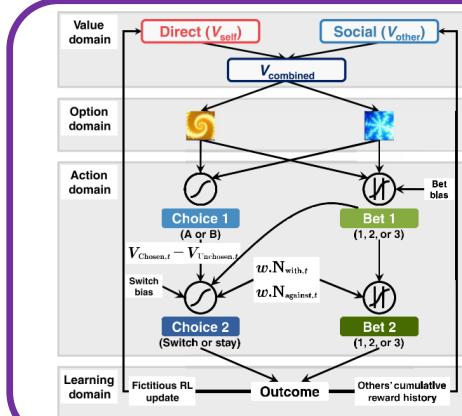
- I ask people to make decisions

Computation



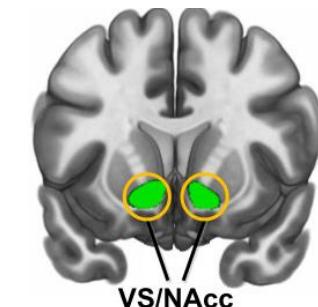
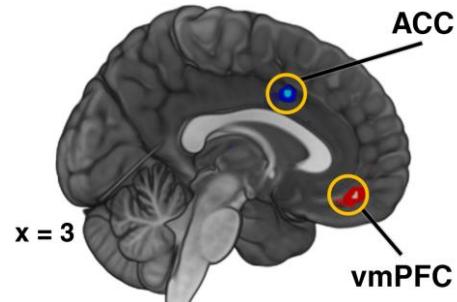
- I build computational models

Algorithm

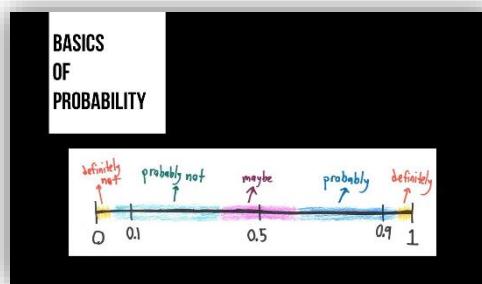


- I examine neural mechanisms

Implementation

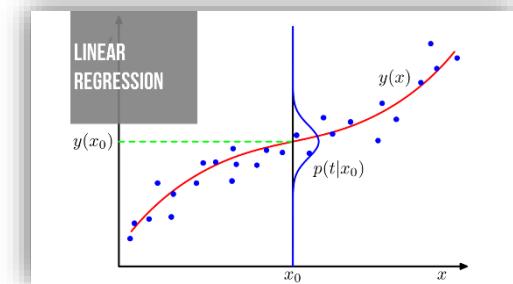
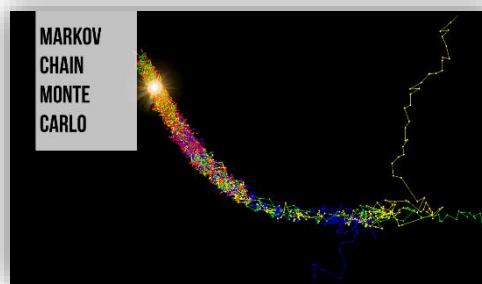
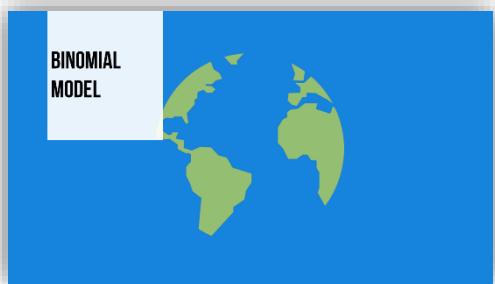
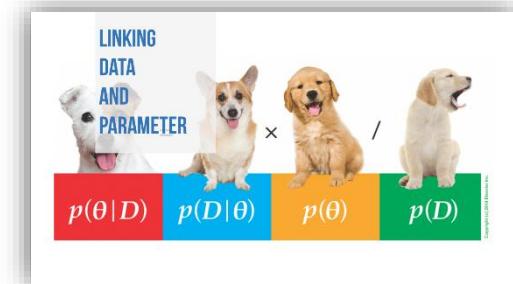


Recap of Topics from Part I

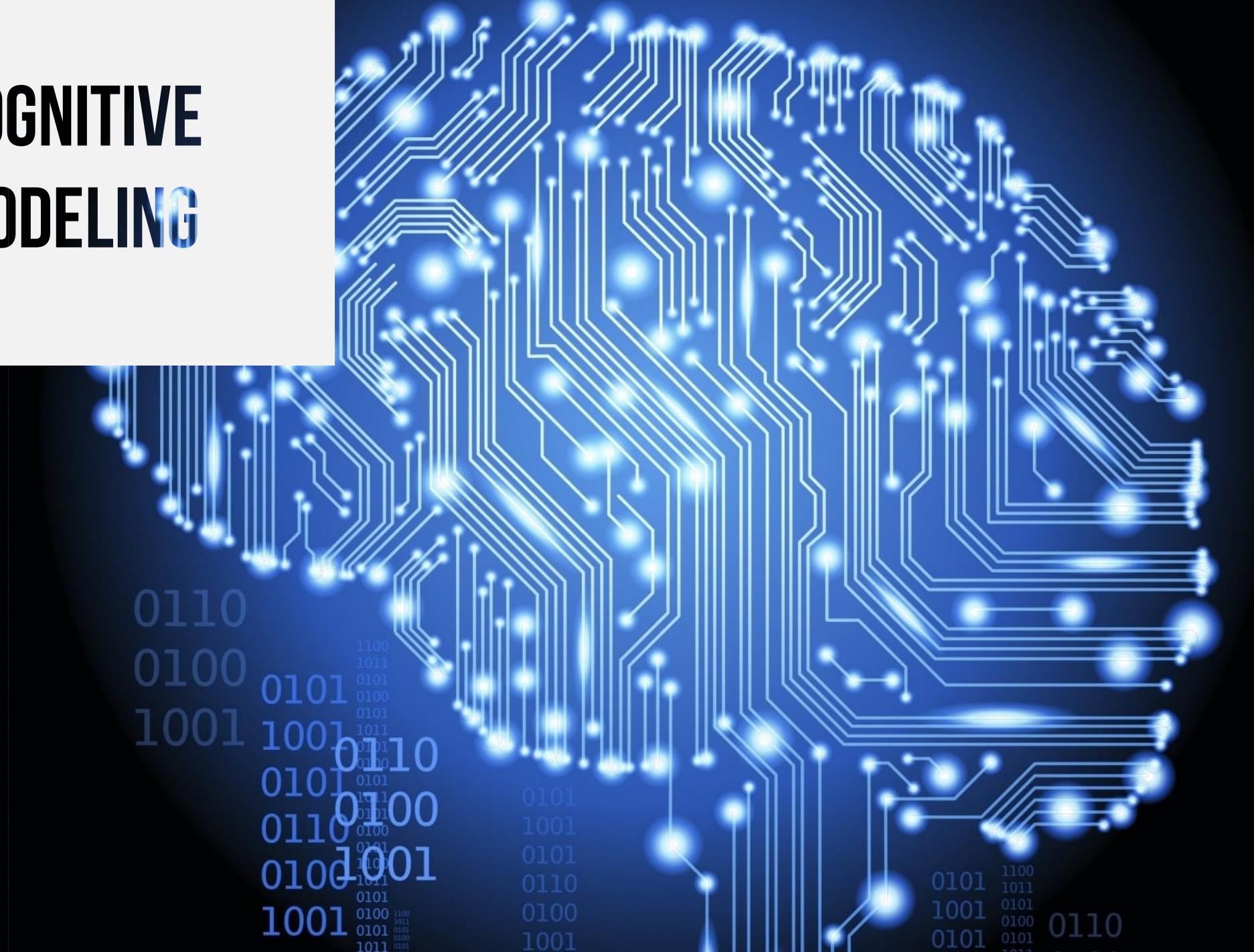


BAYES' THEOREM

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$



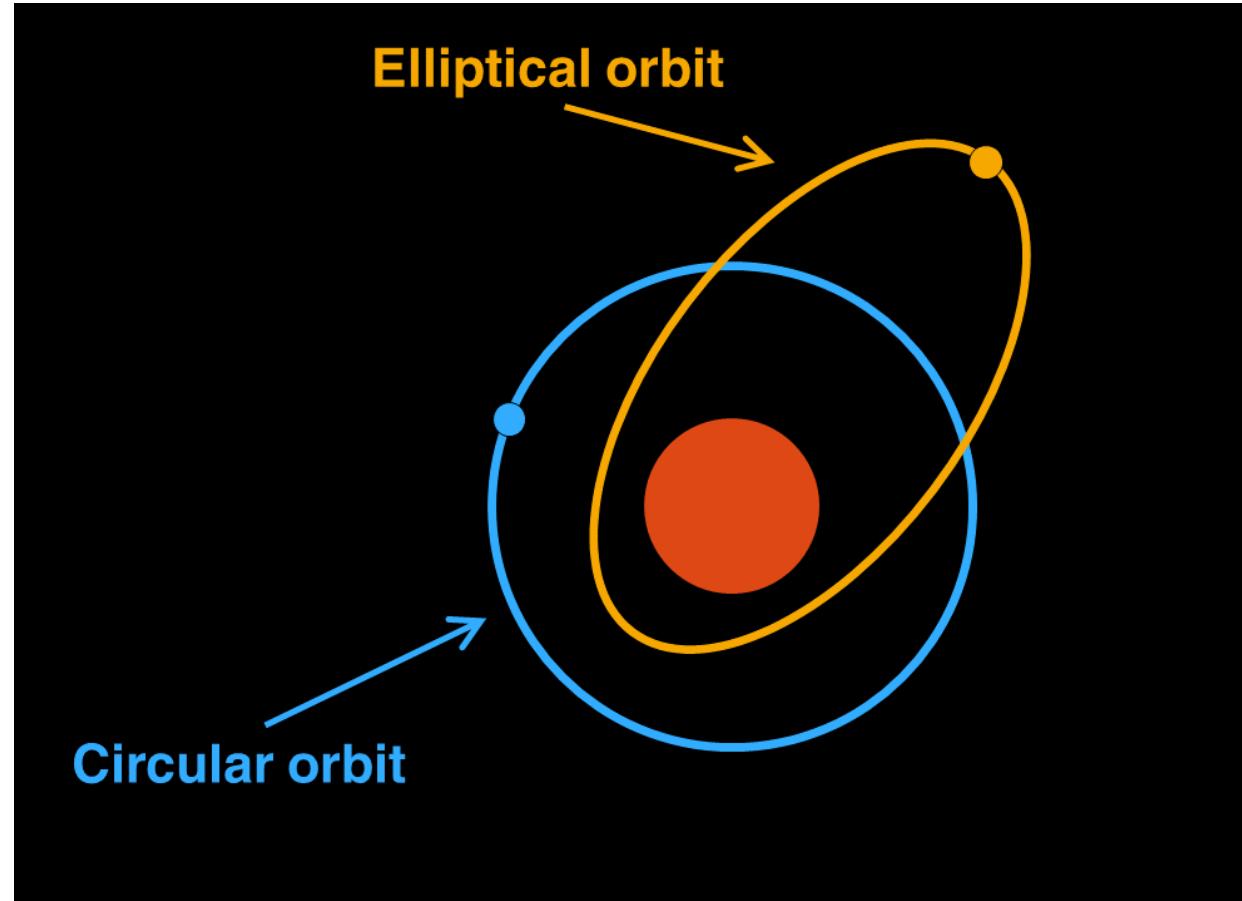
COGNITIVE MODELING



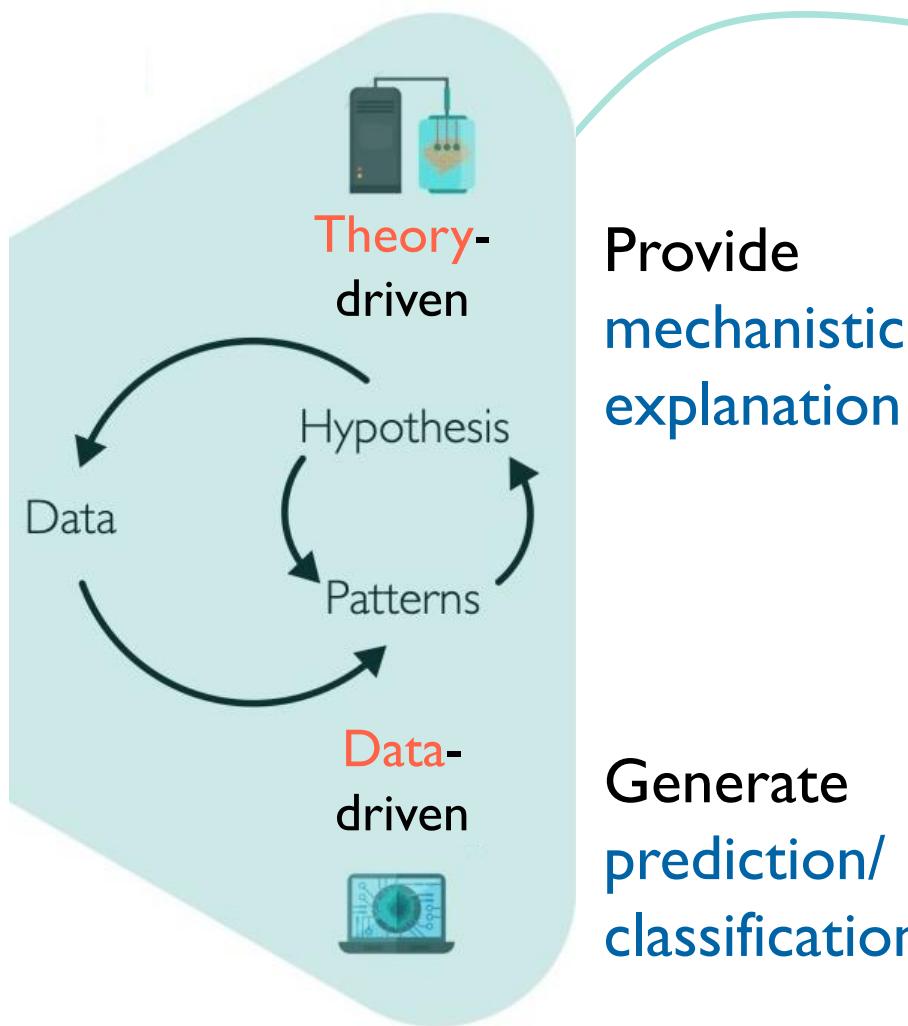
The idea of computational modeling is never new

cognitive model
statistics
computing

Scientists use mathematical models to approximate certain processes (physical or mental), in order to explain and to predict.



Two flavors of computational modeling



Construct



- Reward,
- Learning,
- Self v. Other,
- etc.

Computational modeling

On the Role of Theory and Modeling in Neuroscience

Daniel Levenstein,¹ Veronica A. Alvarez,² Asohan Amarasingham,³ Habiba Azab,⁴ Zhe S. Chen,⁵ Richard C. Gerkin,⁶ Andrea Hasenstaub,⁷ Ramakrishnan Iyer,⁸ Renaud B. Jolivet,⁹ Sarah Marzen,¹⁰ Joseph D. Monaco,¹¹ Astrid A. Prinz,¹² Salma Quraishi,¹³ Fidel Santamaria,¹³ Sabyasachi Shivkumar,¹⁴ Matthew F. Singh,¹⁵ Roger Traub,¹⁶ Farzan Nadim,^{1,7*} Horacio G. Rotstein,^{1,7*} and A. David Redish^{18*}

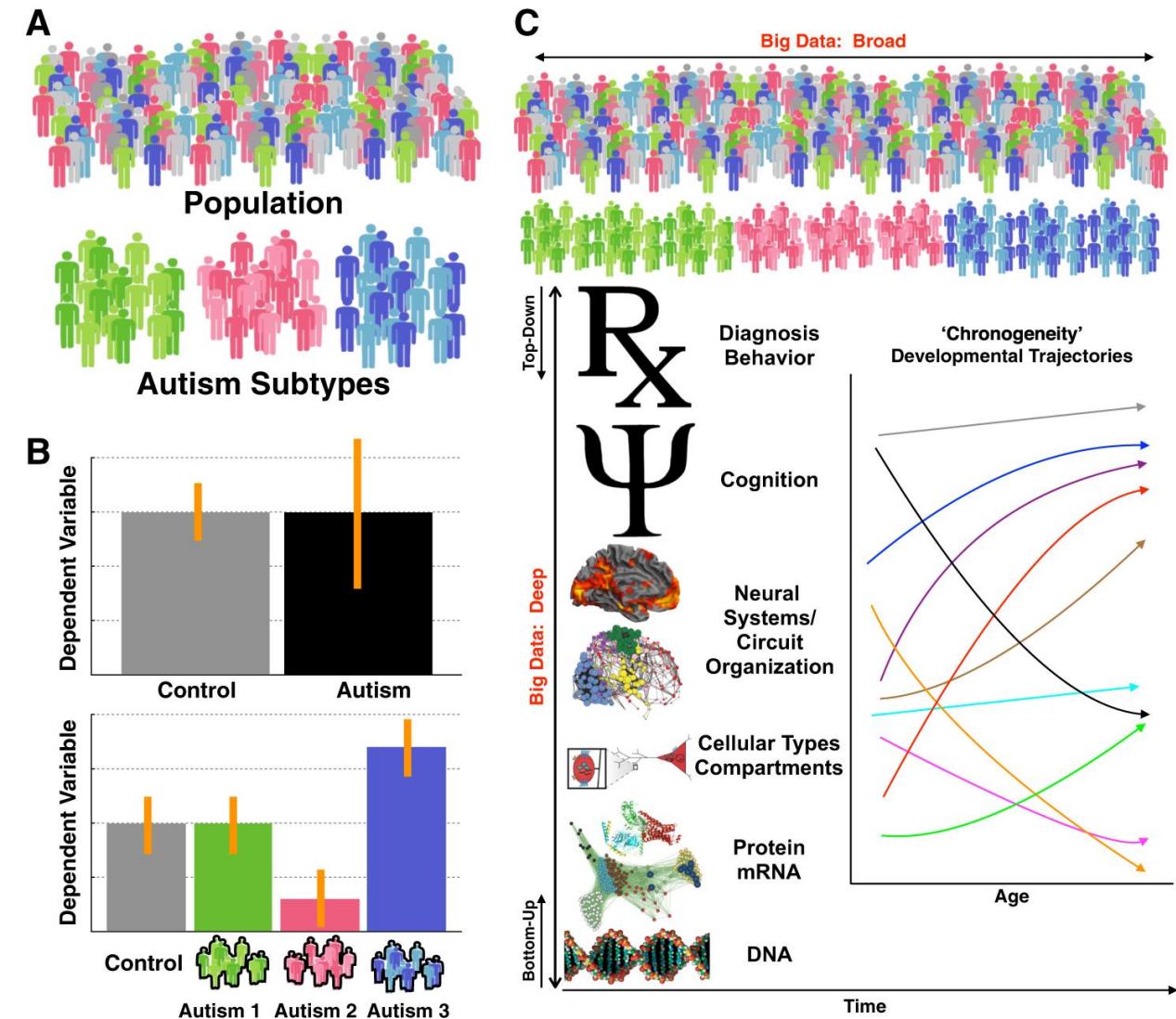
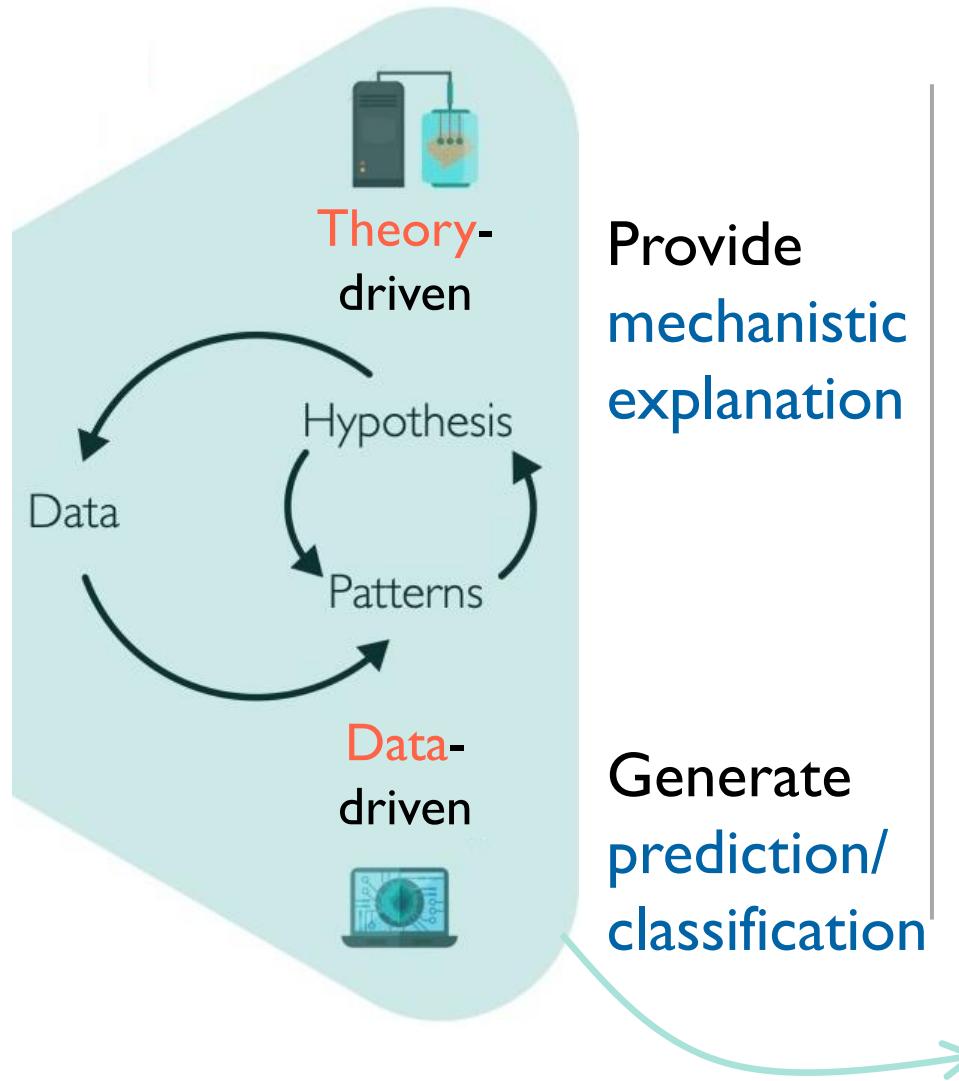
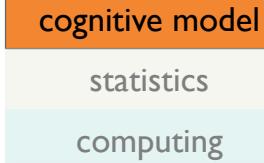
How Computational Modeling Can Force Theory Building in Psychological Science

Olivia Guest and Andrea E. Martin

Building better theories

Clare Press ^{1, 2} Daniel Yon ¹, Cecilia Heyes ^{3, 4}

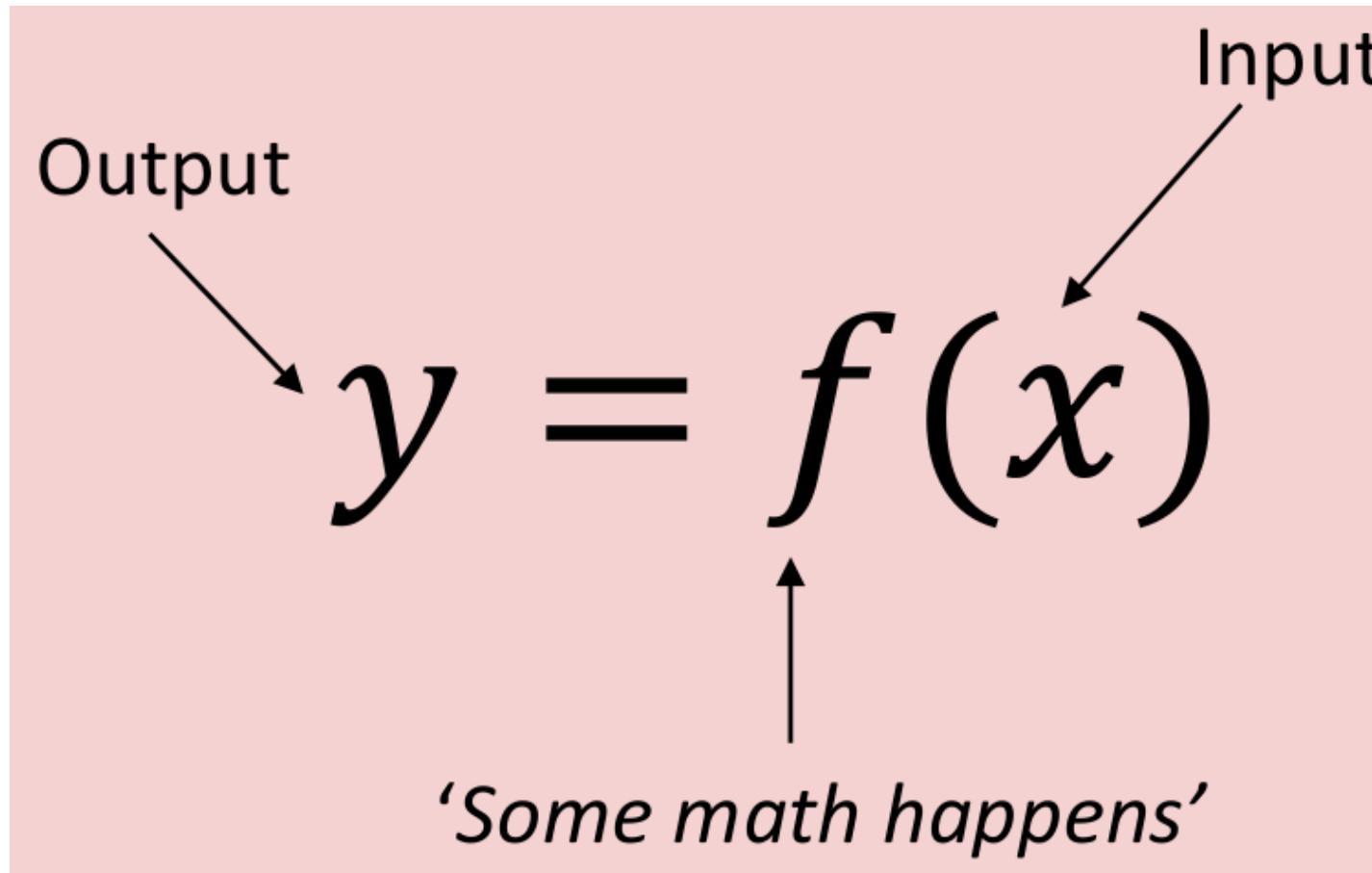
Two flavours of computational modeling



Computational modeling of cognition

cognitive model
statistics
computing

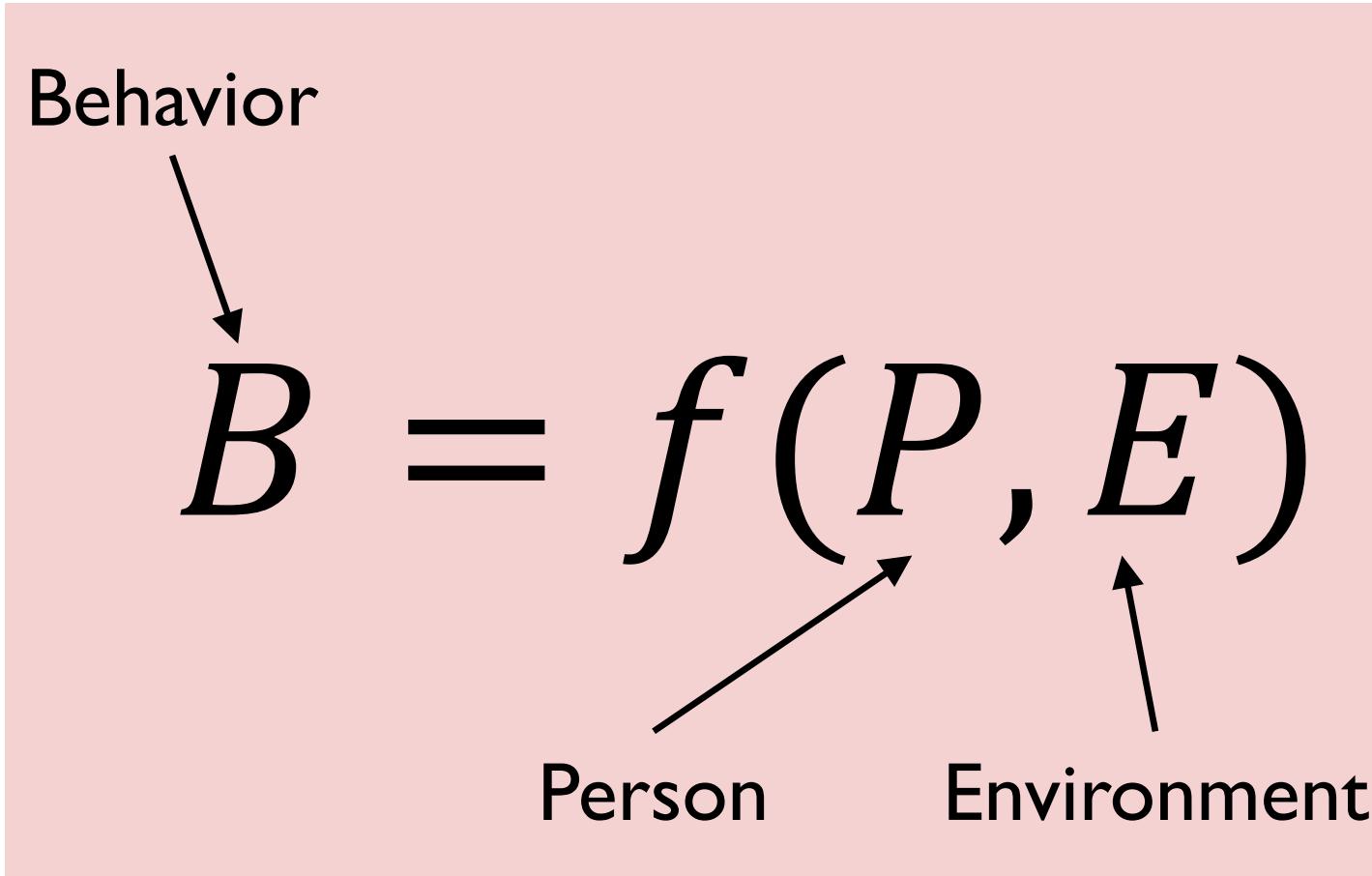
Cognition as information processing



Computational modeling of cognition

cognitive model
statistics
computing

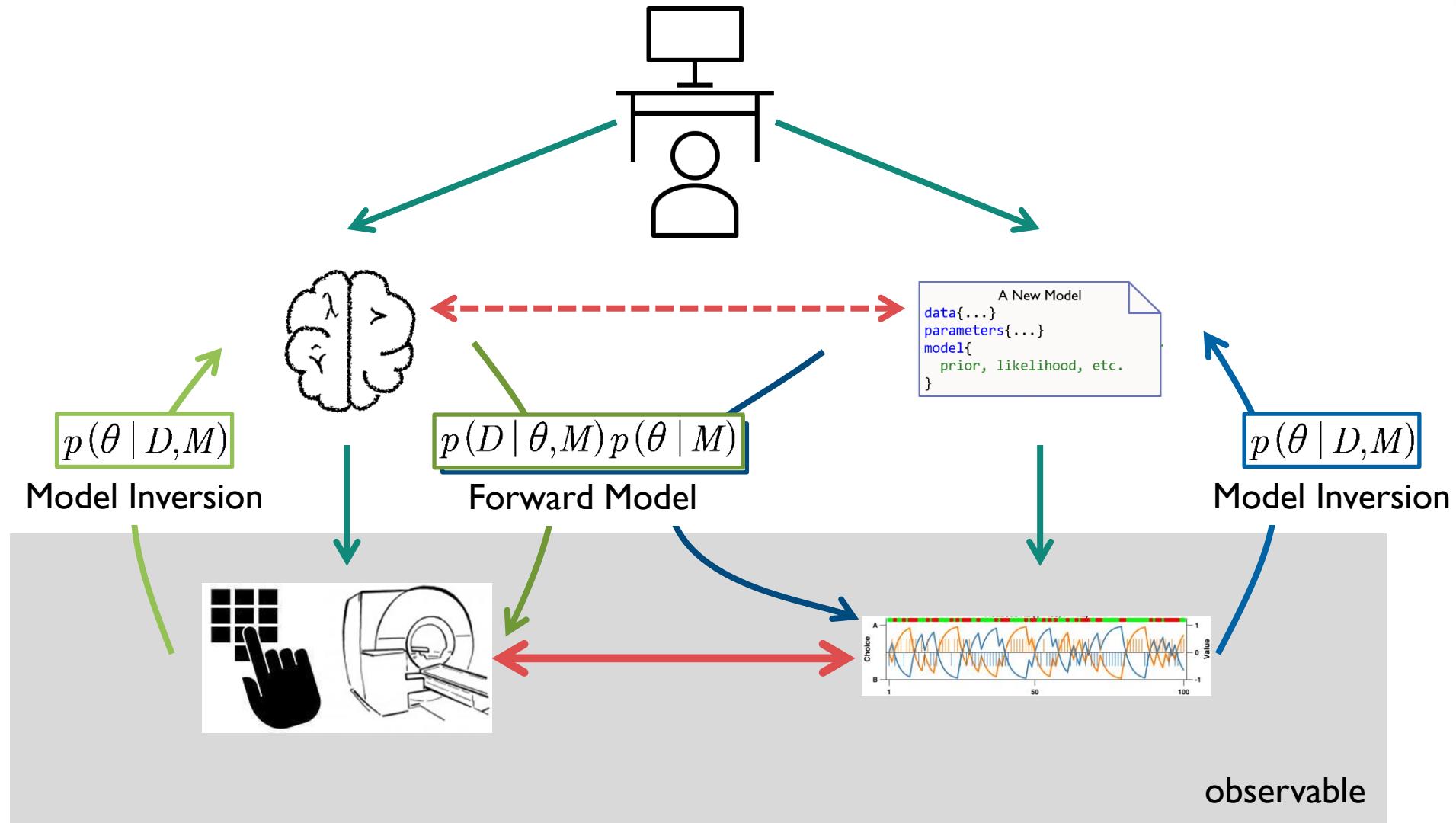
Cognition as information processing



Kurt Lewin, (1936)

What is Cognitive Modeling?

cognitive model
statistics
computing



Essentially, all the models are wrong, but some are useful.



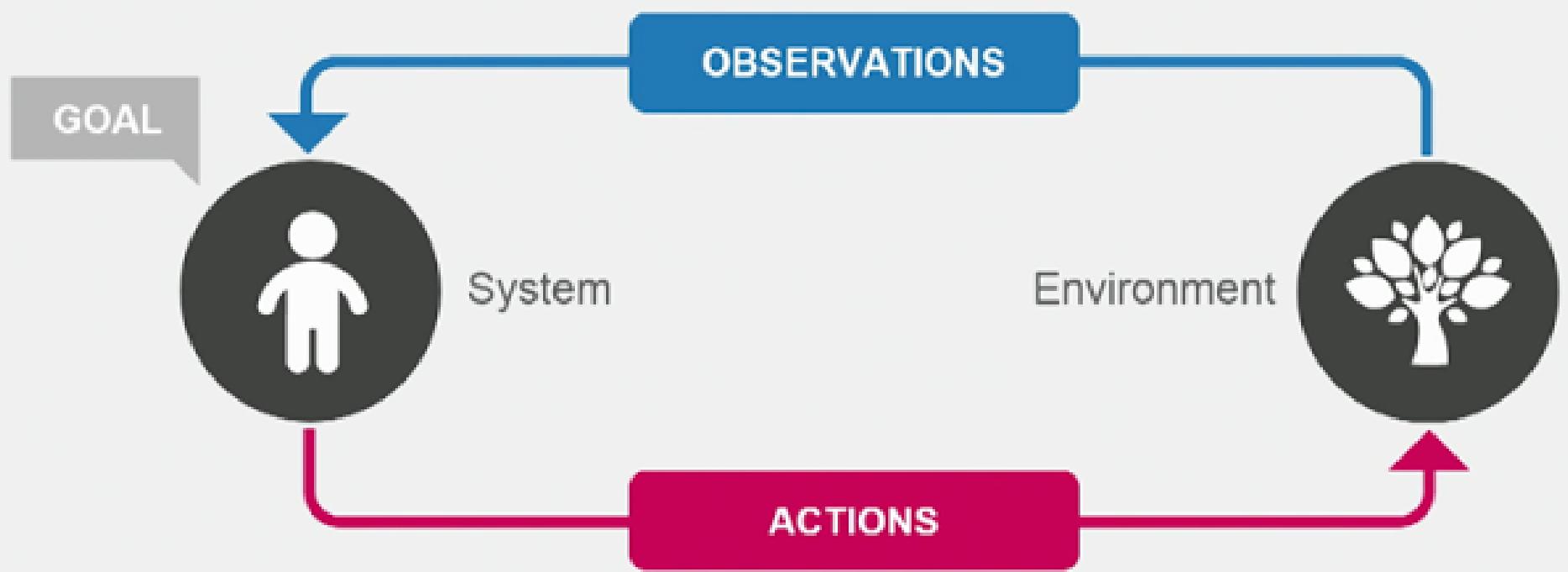
– George E. P. Box

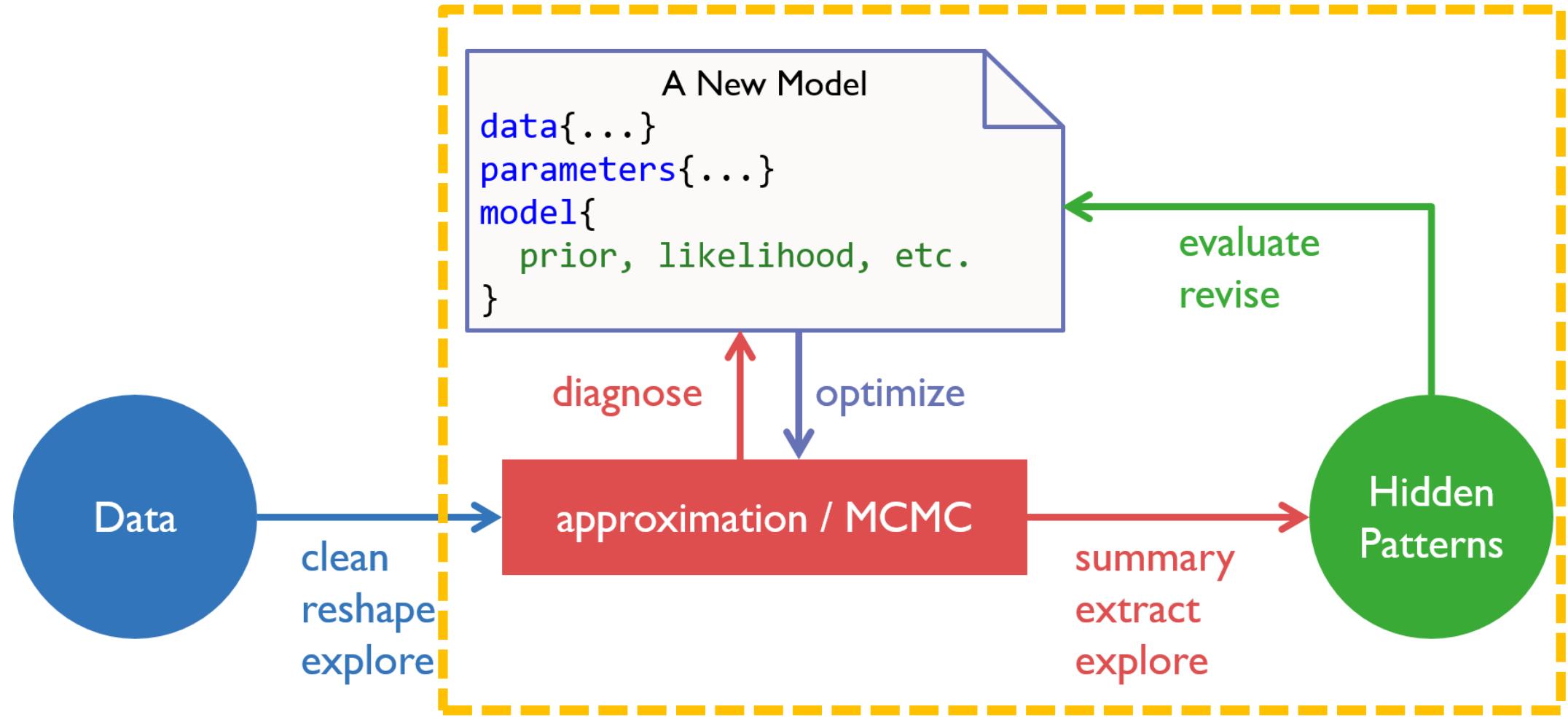
Essentially, all the models are ~~wrong~~ imperfect, but some are useful.

Common cognitive models in decision-making

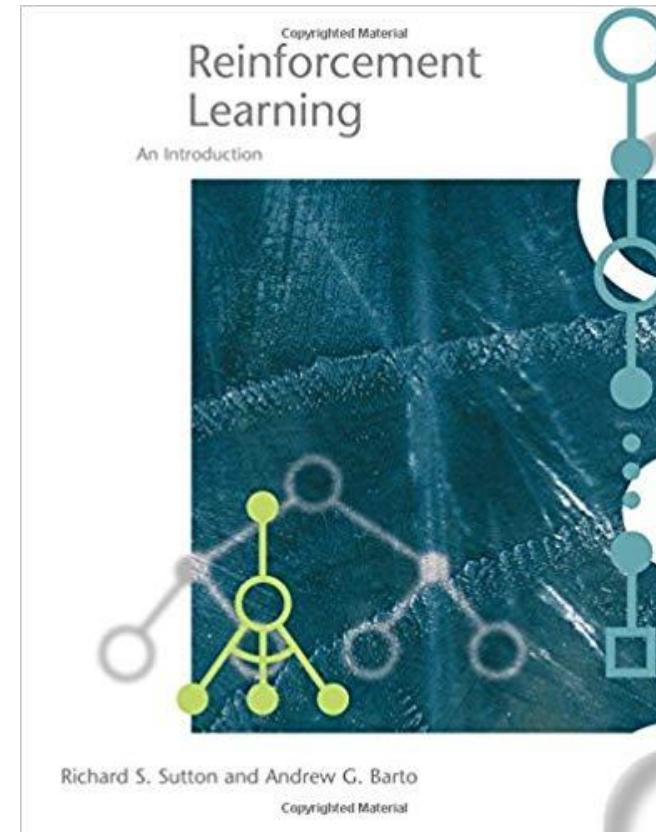
- Reinforcement learning model
- Bayesian learning model
- Risk-aversion model
- Hyperbolic delay discounting model
- Fehr-Schmidt inequity aversion model
- Sequential sampling model
- Experience-weighted attraction model
- ...

REINFORCEMENT LEARNING FRAMEWORK

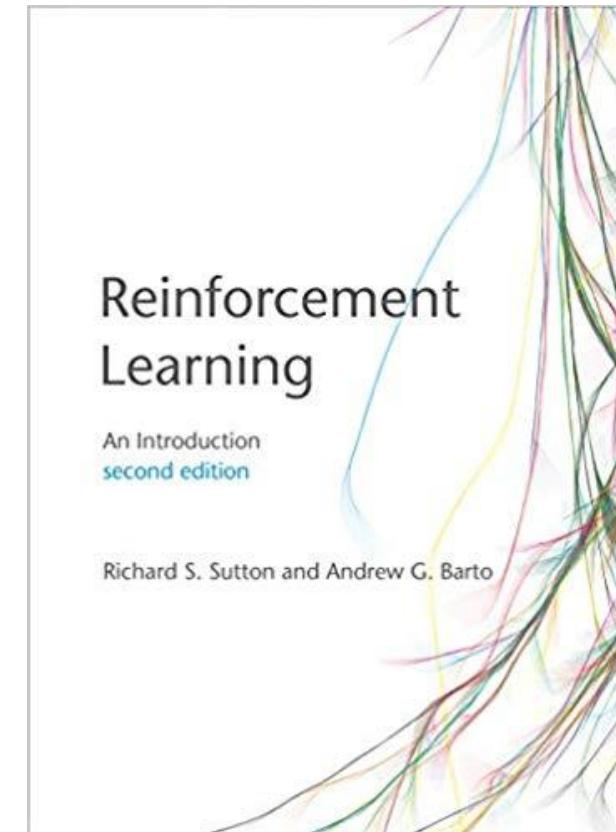




The very short history



1998

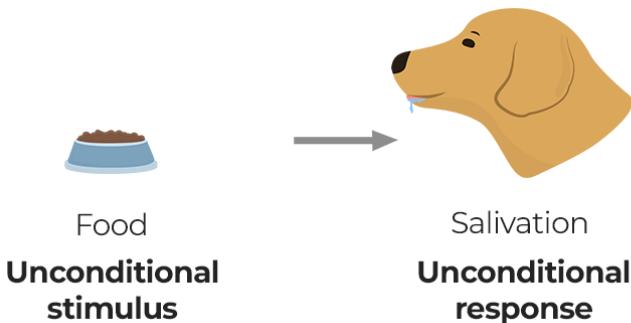


2018

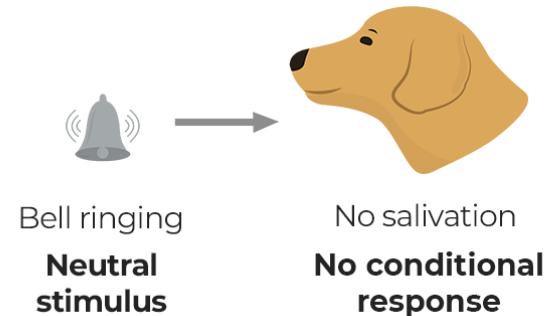
why is it relevant?

cognitive model
statistics
computing

1. Before conditioning



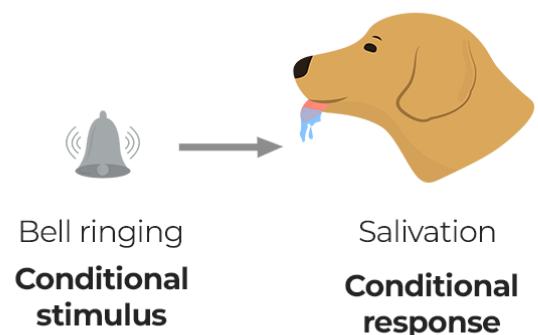
2. Before conditioning



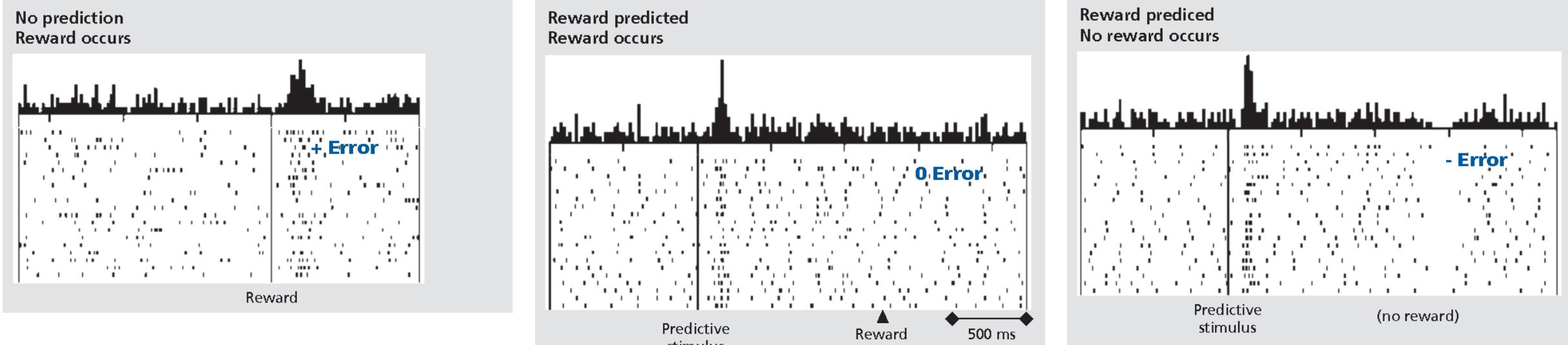
3. During conditioning



4. After conditioning



Reward Prediction error



Predicted reward trials



A neural substrate of prediction and reward

[W Schultz](#), [P Dayan](#), [PR Montague](#) - Science, 1997 - science.org

The capacity to predict future events permits a creature to detect, model, and manipulate the causal structure of its interactions with its environment. Behavioral experiments suggest that ...

★ Save 99 Cite Cited by 10779 Related articles All 47 versions

Classic examples

REPORT

Dissociable Roles of Ventral and Dorsal Striatum in Instrumental Conditioning

John O'Doherty^{1,*}, Peter Dayan², Johannes Schultz¹, Ralf Deichmann¹, Karl Friston¹, Raymond J. Dolan¹

+ See all authors and affiliations

Science 16 Apr 2004:
Vol. 304, Issue 5669, pp. 452-454
DOI: 10.1126/science.1094285

Published: 06 November 2005

Uncertainty-based competition between prefrontal and dorsolateral striatal systems for behavioral control

Nathaniel D Daw , Yael Niv & Peter Dayan

Nature Neuroscience 8, 1704–1711 (2005) | Cite this article

Neuron
Volume 66, Issue 4, 27 May 2010, Pages 585-595

Article

States versus Rewards: Dissociable Neural Prediction Error Signals Underlying Model-Based and Model-Free Reinforcement Learning

Jan Gläscher ^{1, 3}  , Nathaniel Daw ⁴, Peter Dayan ⁵, John P. O'Doherty ^{1, 2, 6}



Open Access | Published: 07 June 2011

Empirical support for an involvement of the mesostriatal dopamine system in human fear extinction

K A Raczka, M-L Mechias, N Gartmann, A Reif, J Deckert, M Pessiglione & R Kalisch 

Translational Psychiatry 1, e12 (2011) | Cite this article

Very recent examples

cognitive model

statistics

computing

REPORT

Dopamine promotes cognitive effort by biasing the benefits versus costs of cognitive work

A. Westbrook^{1,2,3,*}, R. van den Bosch^{2,3}, J. I. Määttä^{2,3}, L. Hofmans^{2,3}, D. Papadopetraki^{2,3}, R. Cools^{2,3,†}, M. J. Frank^{1,4,†}

+ See all authors and affiliations

Science 20 Mar 2020:
Vol. 367, Issue 6484, pp. 1362-1366
DOI: 10.1126/science.aaz5891

Neuron

Available online 17 March 2020

In Press, Corrected Proof 

Article

A Neuro-computational Account of Arbitration between Choice Imitation and Goal Emulation during Human Observational Learning

Caroline J. Charpentier^{1, 2}  , Kiyohito ligaya¹, John P. O'Doherty¹



3 out of 4 focused on Reinforcement Learning models!

nature reviews
neuroscience

Review Article | Published: 12 March 2020

The neural and computational systems of social learning

Andreas Olsson , Ewelina Knapska & Björn Lindström

Nature Reviews Neuroscience 21, 197–212(2020) | Cite this article

Translational
Psychiatry

Article | Open Access | Published: 17 March 2020

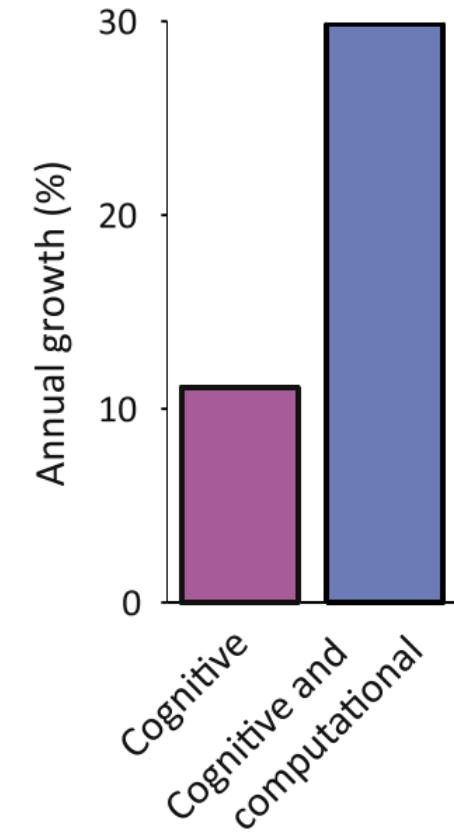
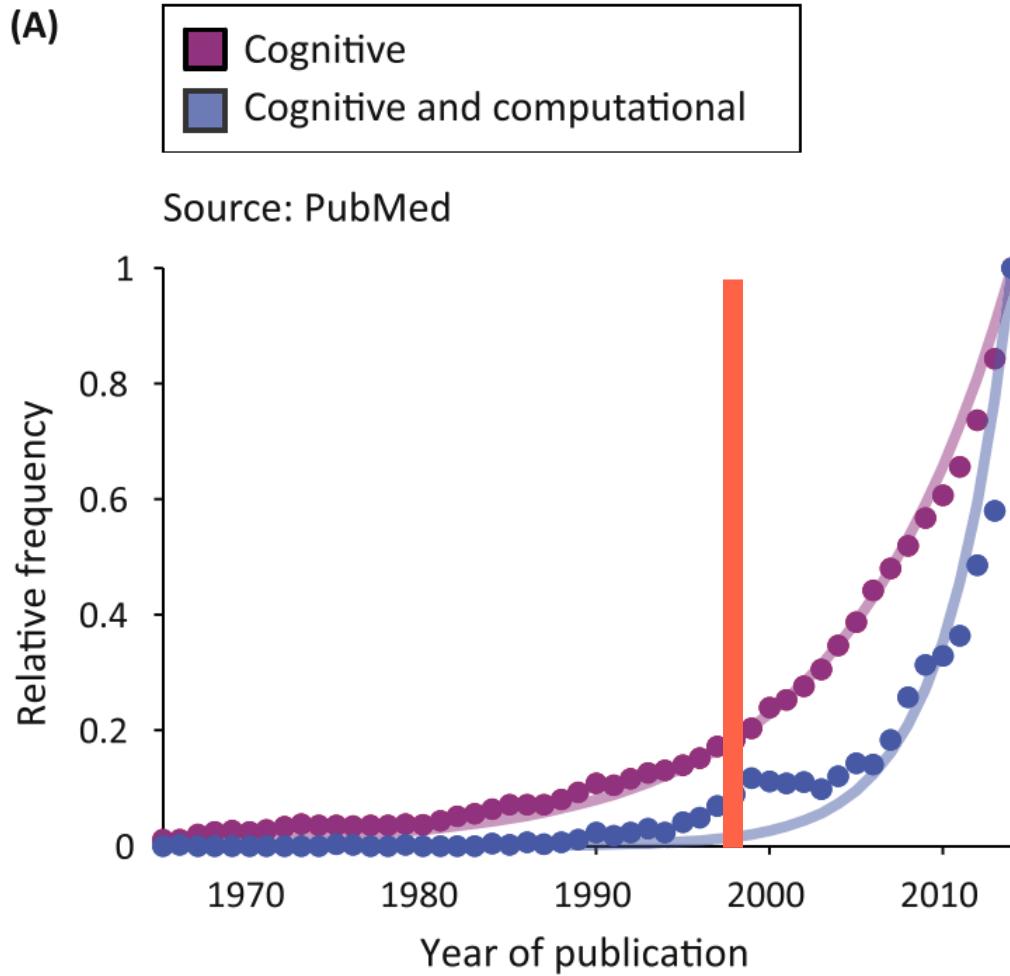
Neurocomputational mechanisms underpinning aberrant social learning in young adults with low self-esteem

Geert-Jan Will , Michael Moutoussis, Palee M. Womack, Edward T. Bullmore, Ian M. Goodyer, Peter Fonagy, Peter B. Jones, NSPN Consortium, Robb B. Rutledge & Raymond J. Dolan

Boom in Cognitive Modeling

cognitive model
statistics
computing

(A)



2-armed bandit task



a simple task often used in the laboratory:

- **repeated choice** between N options (**N-armed bandit**)
- ...whose properties (reward amounts, probabilities) are learned through **trial-and-error**
- ...with a **goal** in mind: maximize the overall reward

2-armed bandit task

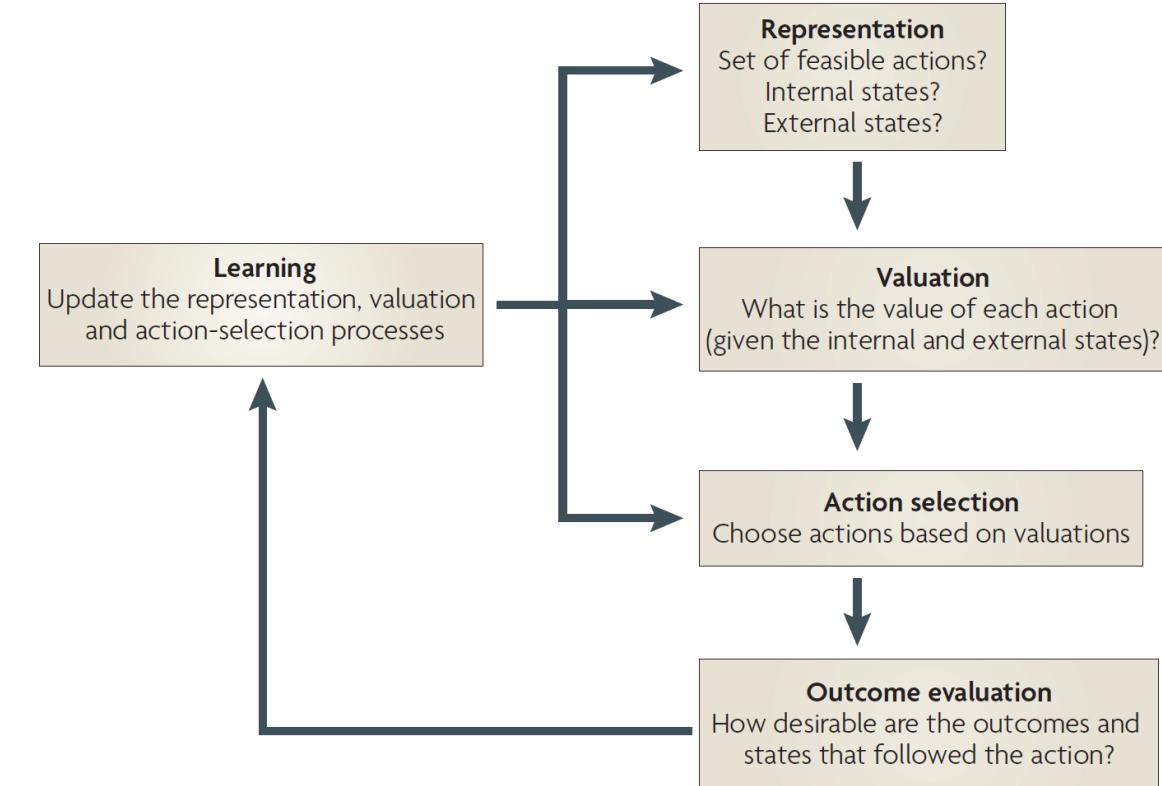
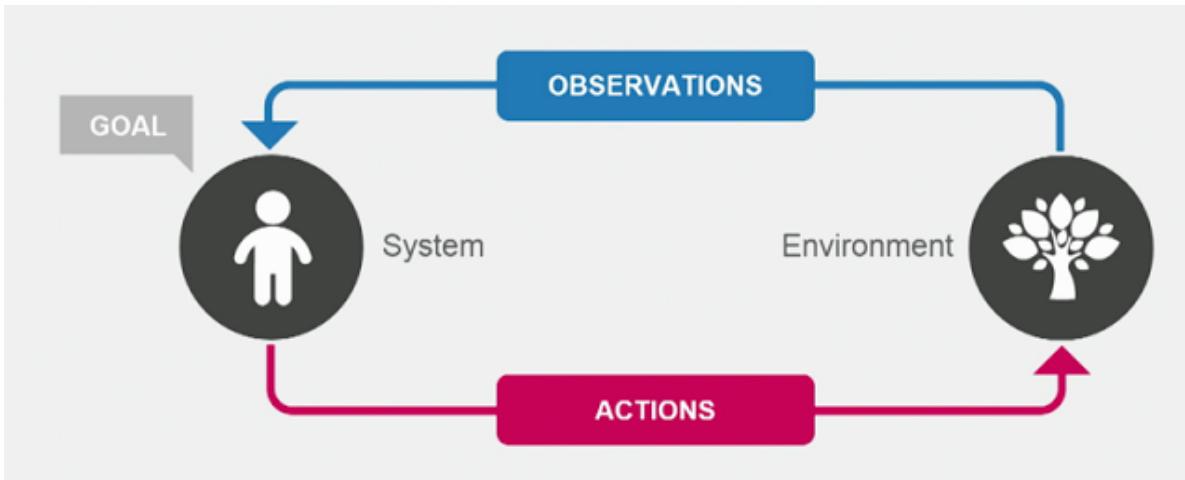


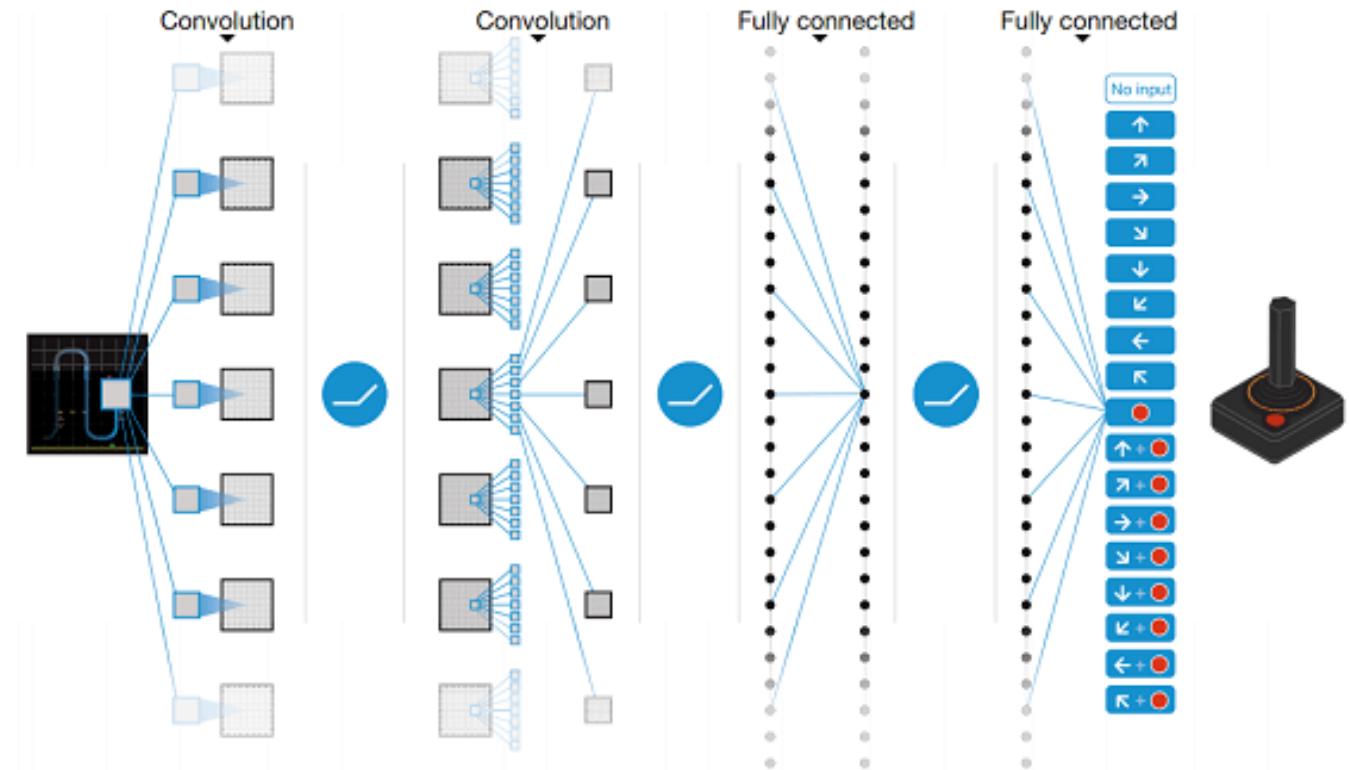
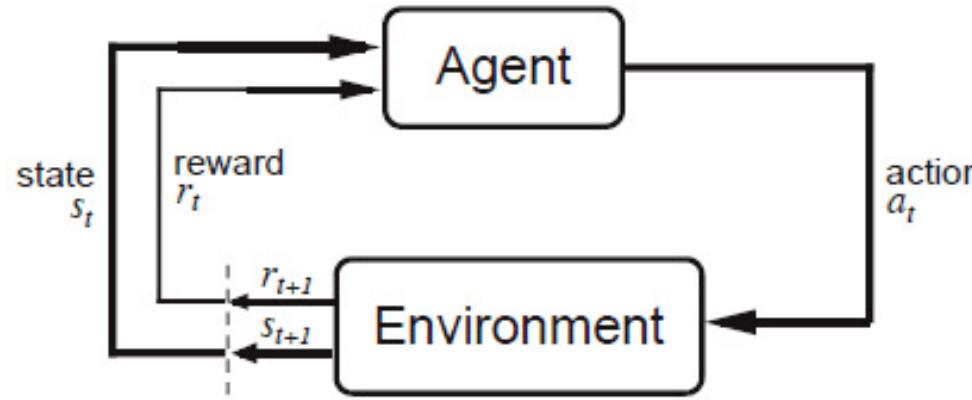
What can be your **strategies**:

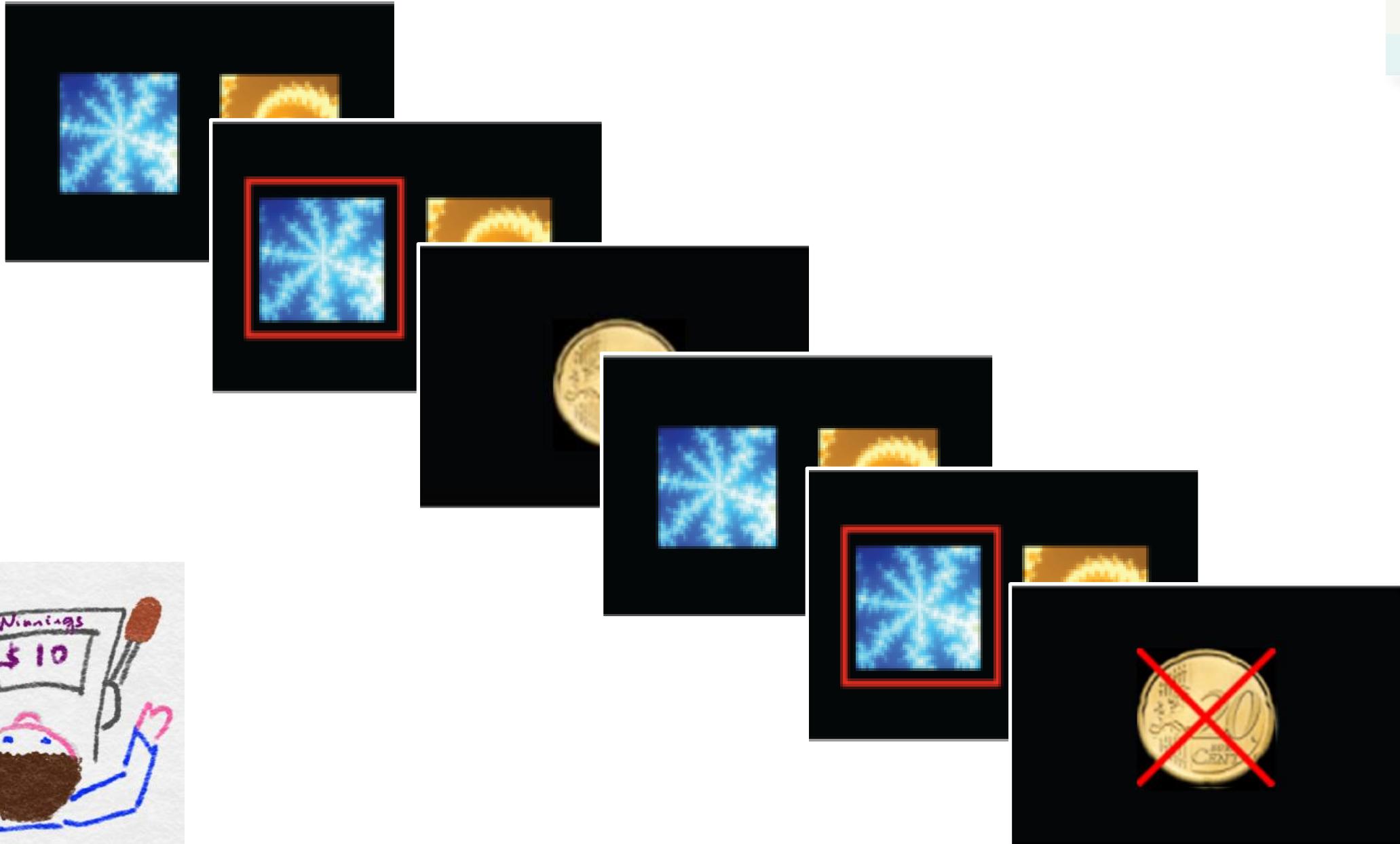
1. **predict** the value of each deck
2. **choose** the best
3. **learn** from outcome to update predictions
(repeat)

How prediction is shaped by learning?

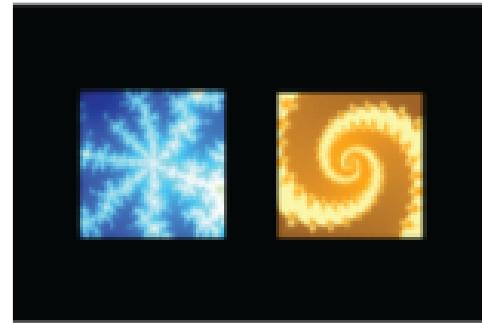
cognitive model
statistics
computing



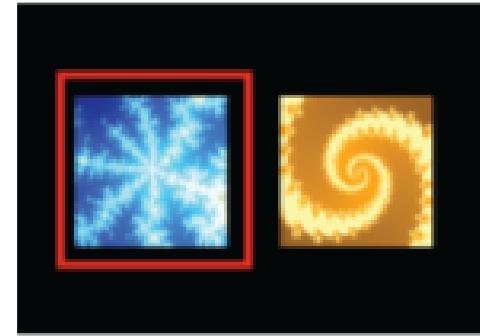




One simple experiment: two choice task



choice presentation



action selection



outcome

what do we know?

what can we measure?

what do we not know?

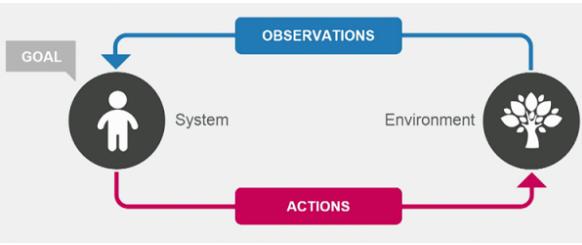
Data: choice & outcome

Summary stats: choice accuracy

Learning algorithm: RL update

$p(\text{choosing the better option})$

Rescorla-Wagner Value Update



Cognitive Model

- cognitive process
- using internal variables and free parameters

Observation Model (Data Model)

- relate model to observed data
- has to account for noise

Rescorla-Wagner (1972)

- The idea: **error-driven** learning
- Change in value is proportional to the difference between actual and predicted outcome



Robert A. Rescorla

Allan R. Wagner



Value update: $V_t = V_{t-1} + \alpha * PE_{t-1}$

Prediction error: $PE_{t-1} = R_{t-1} - V_{t-1}$

α - learning rate
PE - reward prediction error
V - value
R - reward

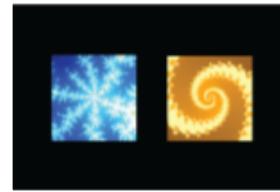
*Expectations on the next trial = the expectation on the current trial + learning rate *
prediction error (reward – current expectation)*

Understand the learning rate

cognitive model
statistics
computing

Value update: $V_t = V_{t-1} + \alpha * PE_{t-1}$

Prediction error: $PE_{t-1} = R_{t-1} - V_{t-1}$



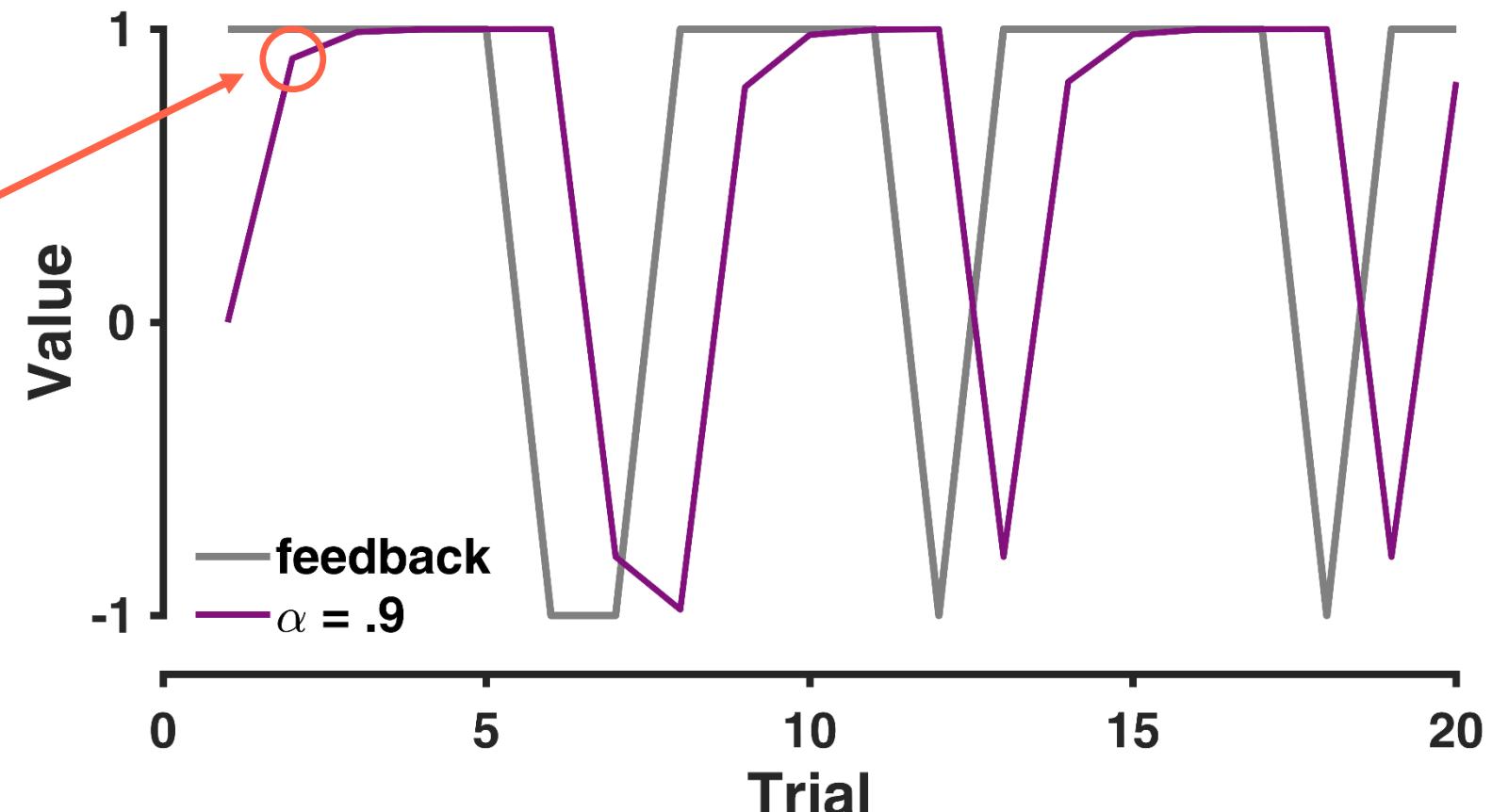
if $\alpha = 0.9$

$$V_1 = 0$$

$$V_2 = V_1 + 0.9 * (1 - V_1)$$

$$= 0 + 0.9 * (1 - 0)$$

$$= 0.9$$



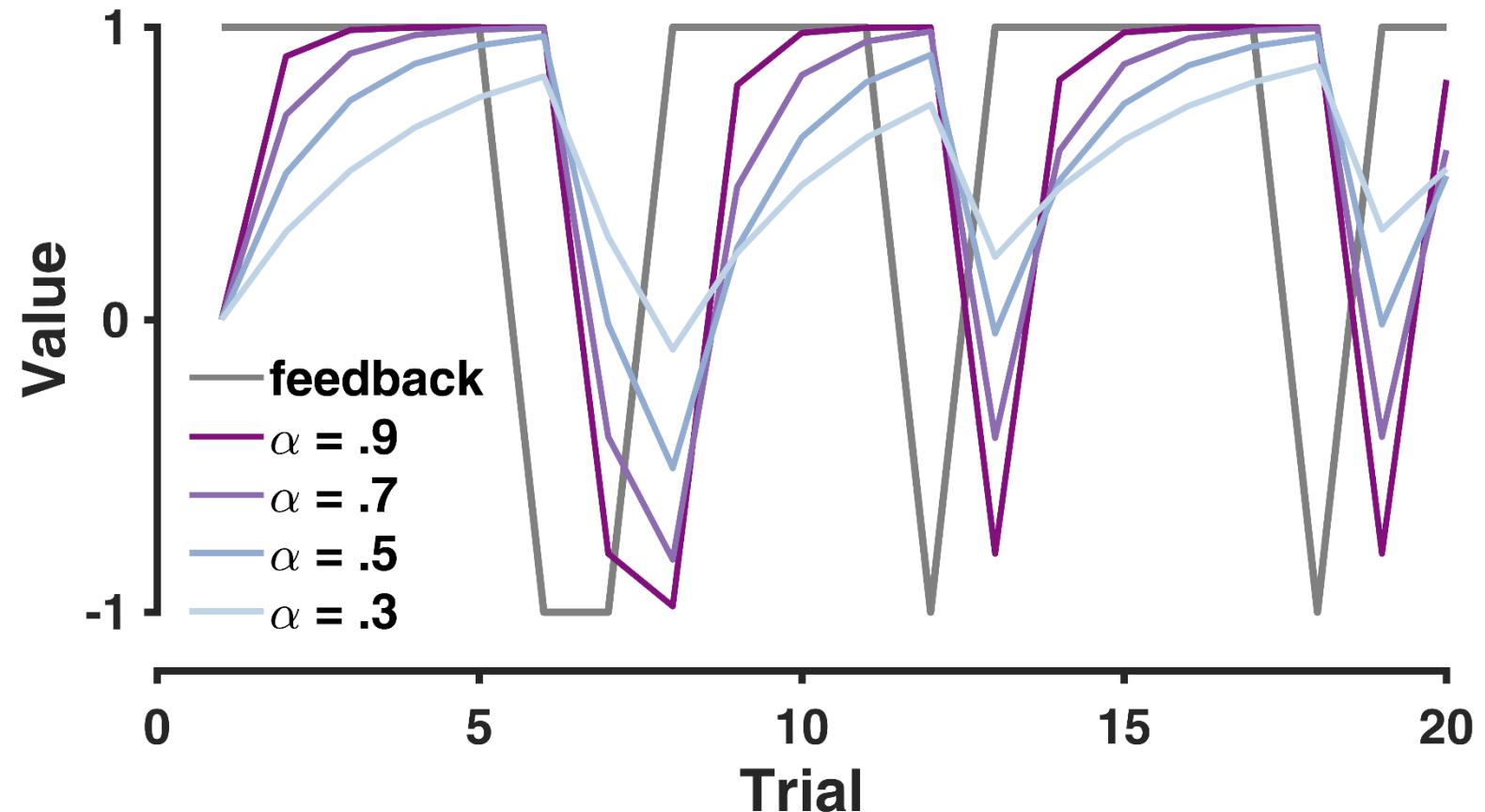
reward contingency – 80:20

Understand the learning rate

cognitive model
statistics
computing

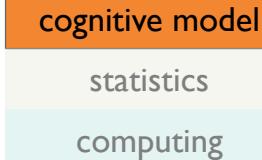
Value update: $V_t = V_{t-1} + \alpha * PE_{t-1}$

Prediction error: $PE_{t-1} = R_{t-1} - V_{t-1}$



reward contingency – 80:20

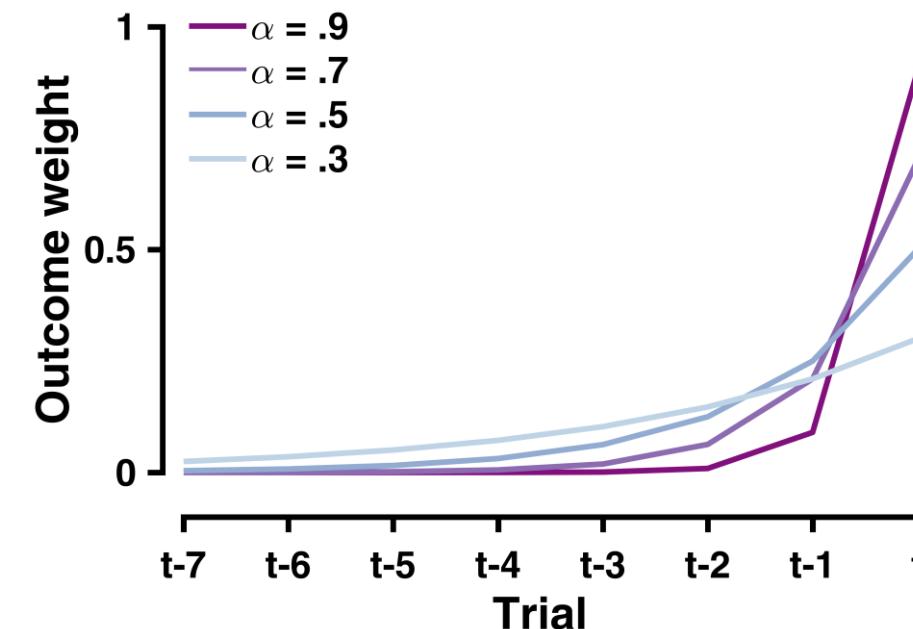
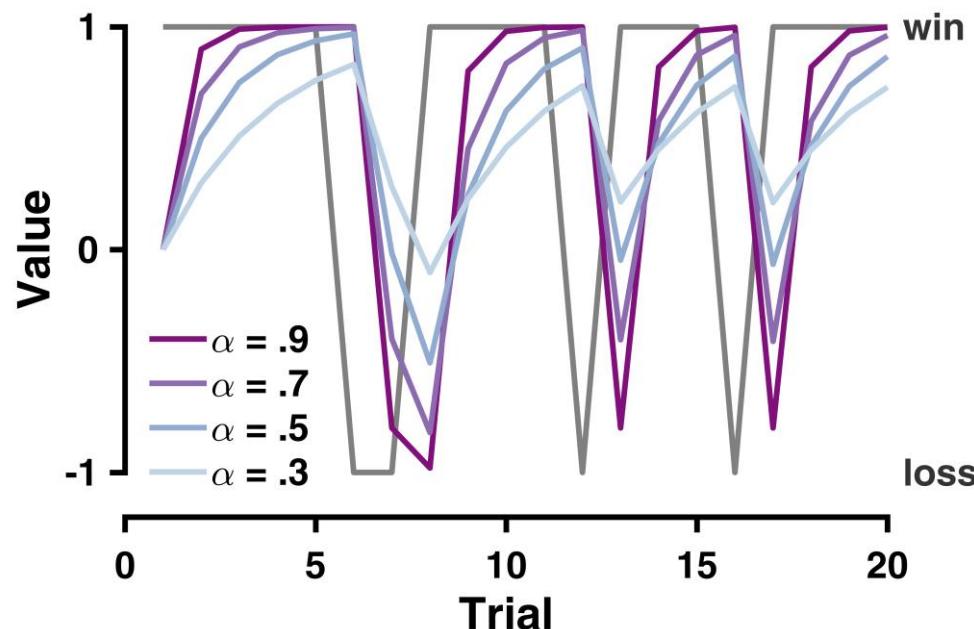
Understand the learning rate



Value update: $V_t = V_{t-1} + \alpha * PE_{t-1}$

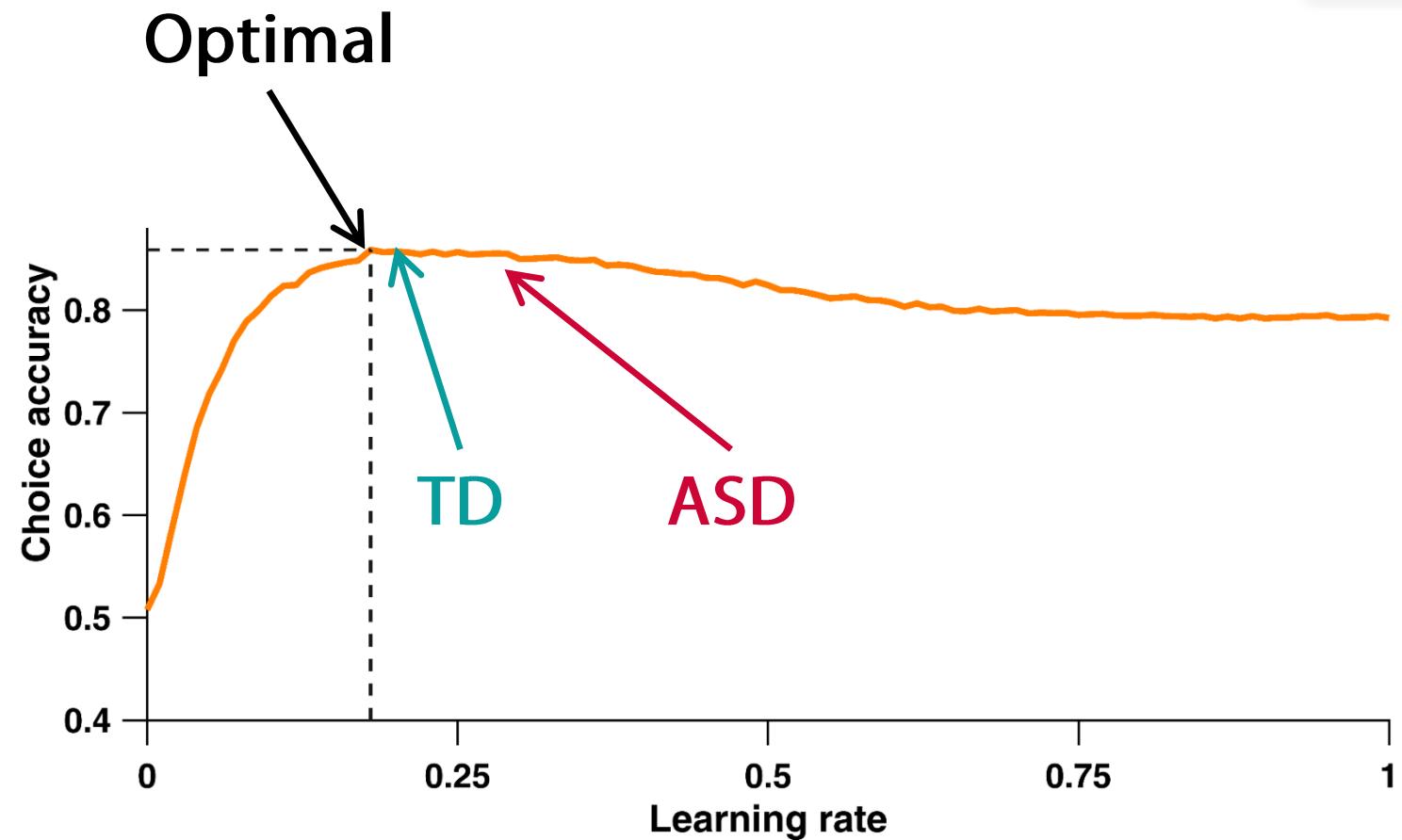
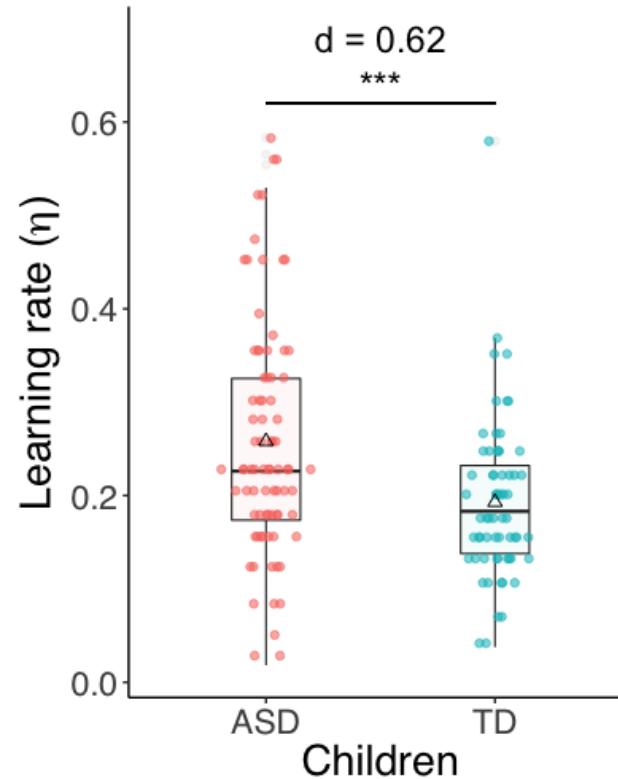
Prediction error: $PE_{t-1} = R_{t-1} - V_{t-1}$

$$\begin{aligned}
 V_t &= (1 - \alpha) V_{t-1} + \alpha R_{t-1} \\
 &= (1 - \alpha) (V_{t-2} + \alpha (R_{t-2} - V_{t-2})) + \alpha R_{t-1} \\
 &= (1 - \alpha)^{t-1} V_1 + \sum_{i=1}^{t-1} (1 - \alpha)^{t-i-1} \alpha R_i
 \end{aligned}$$

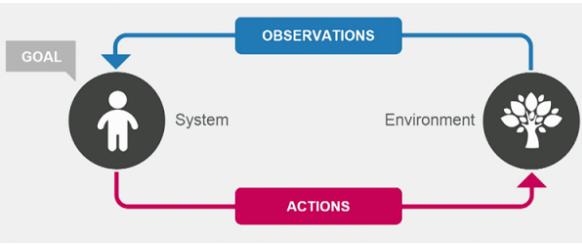


Optimal learning rate?

cognitive model
statistics
computing



Rescorla-Wagner Value Update



Value update:

$$V_{t+1} = V_t + \alpha * PE_t$$

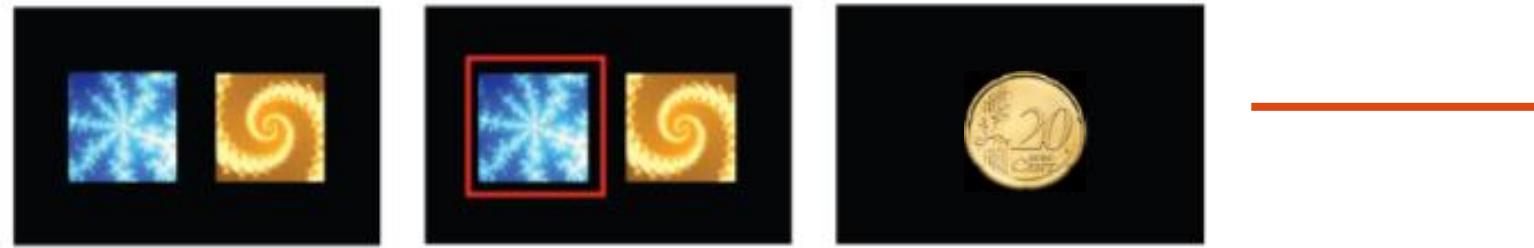
Prediction error:

$$PE_t = R_t - V_t$$

choice rule:

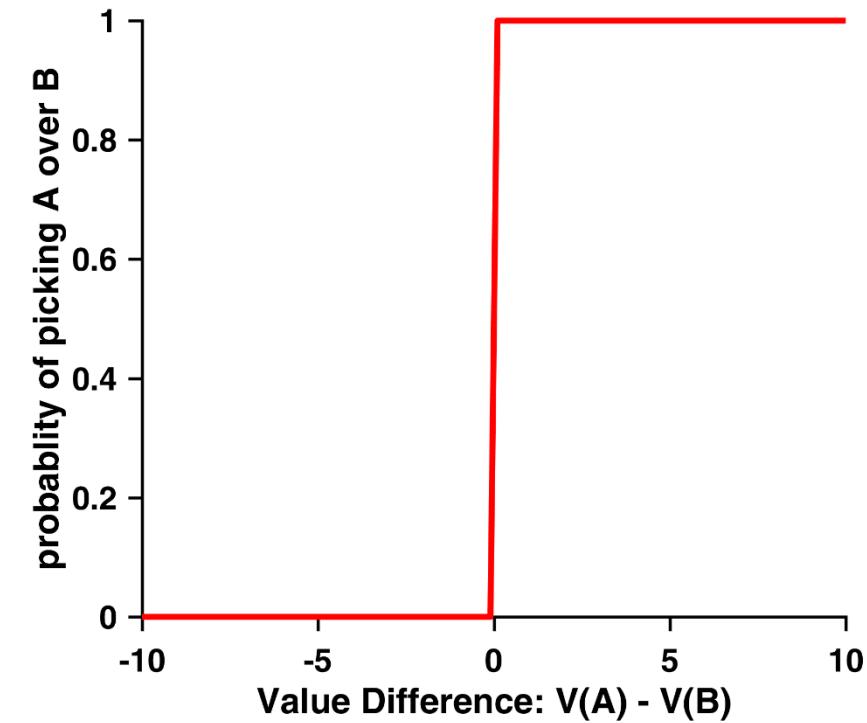
greedy / ϵ -greedy / softmax

Choice rule: greedy

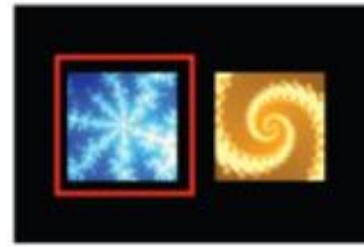
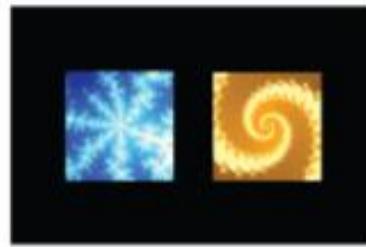


$$\begin{aligned} V(\text{orange spiral}) \\ V(\text{blue fractal}) \end{aligned}$$

$$p(C = a) = \begin{cases} 1, & V(a) > V(b) \\ 0, & V(a) < V(b) \end{cases}$$



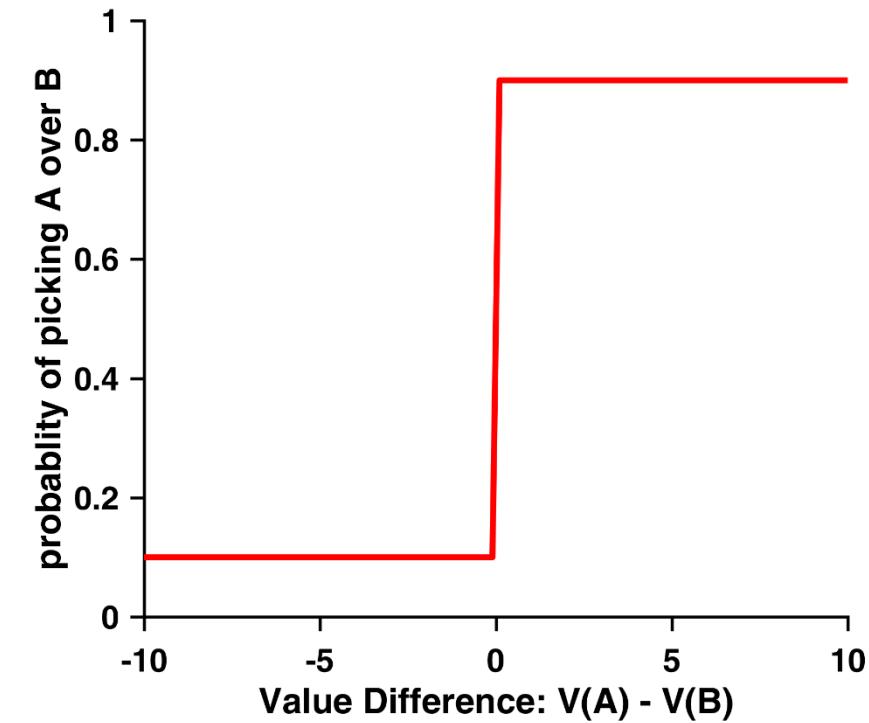
Choice rule: ϵ -greedy



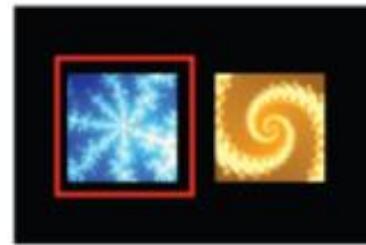
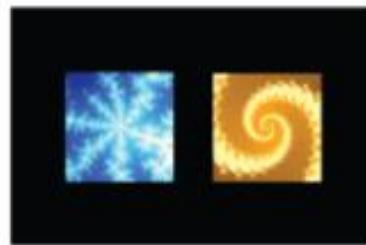
$$V(\text{swirl})$$

$$V(\text{snowflake})$$

$$p(C = a) = \begin{cases} 1 - \epsilon, & V(a) > V(b) \\ \epsilon, & V(a) < V(b) \end{cases}$$



Choice rule: softmax

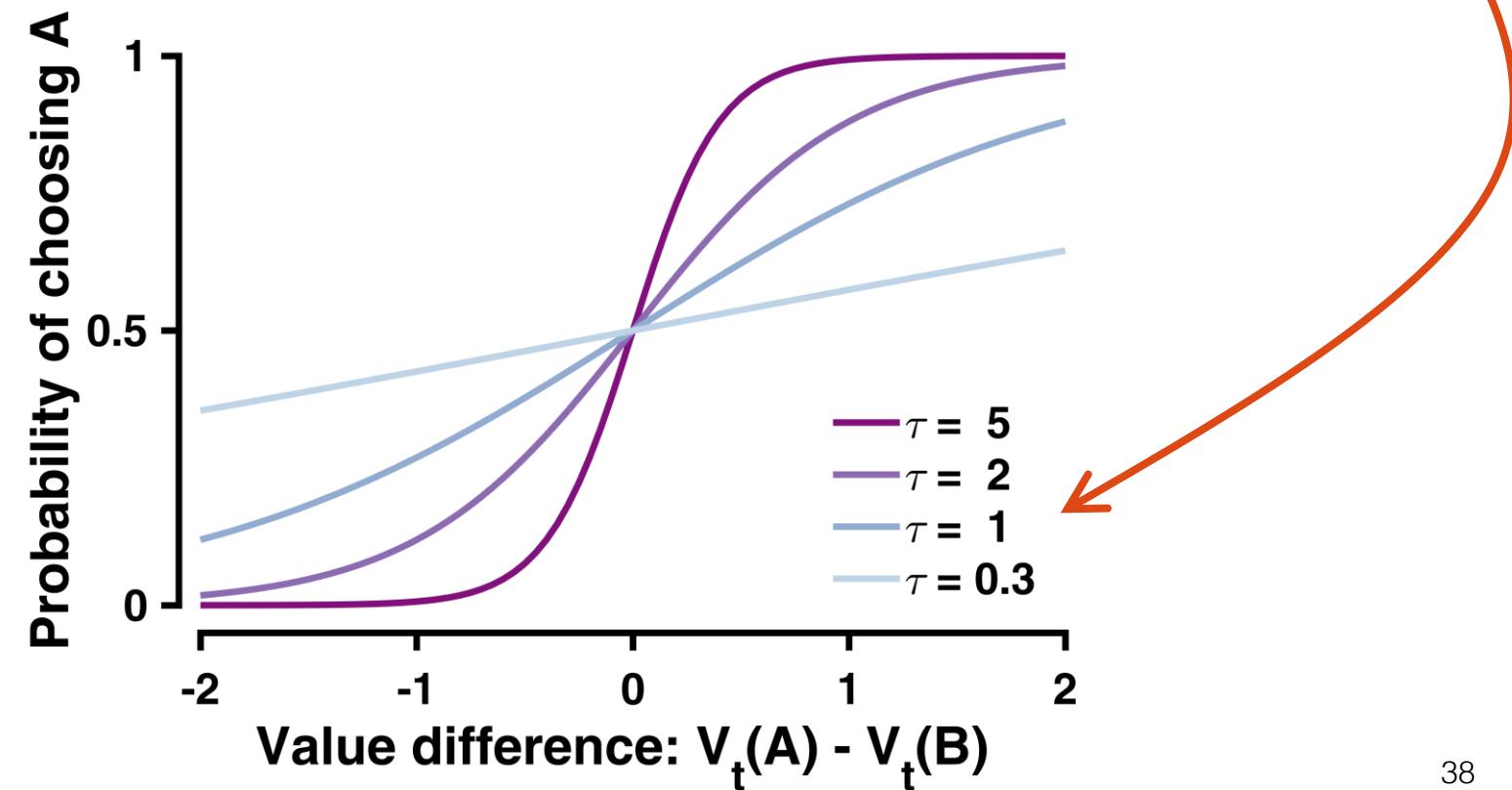


$$V(\text{orange})$$

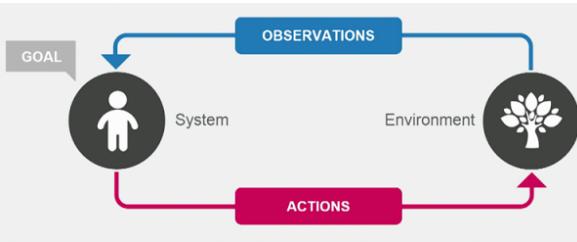
$$V(\text{blue})$$

$$p_t(A) = \frac{e^{\tau * V_t(A)}}{e^{\tau * V_t(A)} + e^{\tau * V_t(B)}}$$

$$= \frac{1}{1 + e^{-\tau * (V_t(A) - V_t(B))}}$$



Rescorla-Wagner Value Update



Value update:

$$V_{t+1} = V_t + \alpha * PE_t$$

Prediction error:

$$PE_t = R_t - V_t$$

choice rule (sigmoid /softmax):

$$p(C=a) = \frac{1}{1+e^{\tau*(v(b)-v(a))}}$$

α - learning rate

PE - reward prediction error

V - value

R - reward

τ - softmax temperature

Summarizing terminologies

simple Q-Learning

$$V_{t+1} = V_t + \alpha (R_t - V_t)$$

$$p_t(A) = \frac{1}{1+e^{-\tau*(v(A)-v(B))}}$$

$$\text{LL} = \text{categorical_lpmf}(C_t | p_t)$$

simple linear regression

$$\mu = a + b X$$

$$\text{LL} = \text{normal_lpdf}(Y | \mu, \delta)$$

Summarizing terminologies

simple Q-Learning

$$V_{t+1} = V_t + \alpha (R_t - V_t)$$

$$p_t(A) = \frac{1}{1+e^{-\tau*(v(A)-v(B))}}$$

$$\text{LL} = \text{categorical_lpmf}(C_t | p_t)$$

simple linear regression

$$\mu = a + b X$$

$$\text{LL} = \text{normal_lpdf}(Y | \mu, \delta)$$

data / observation

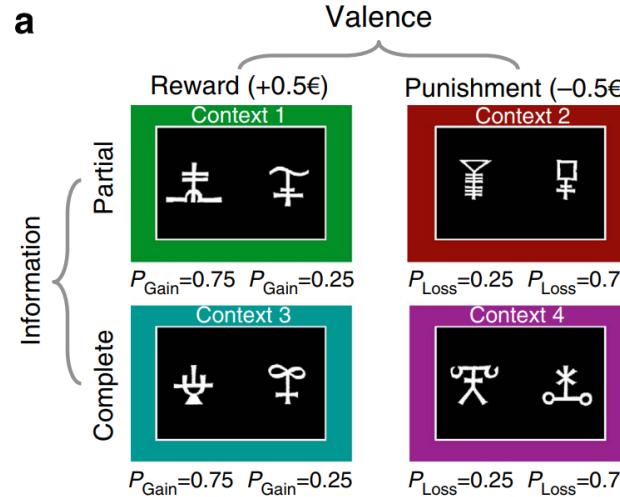
free parameter / model parameter

internal/latent (model) variable

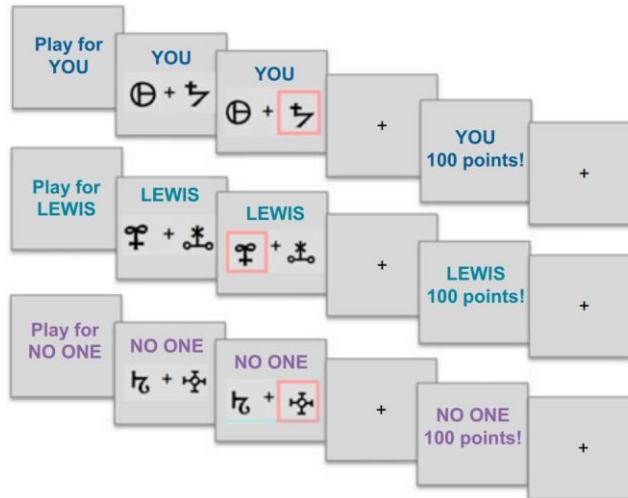
likelihood function

Generalizing RL framework

cognitive model
statistics
computing

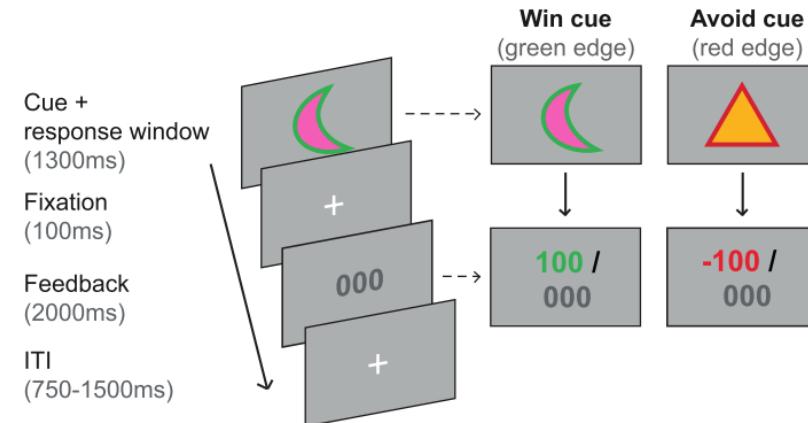


[Palminteri et al. \(2015\)](#)

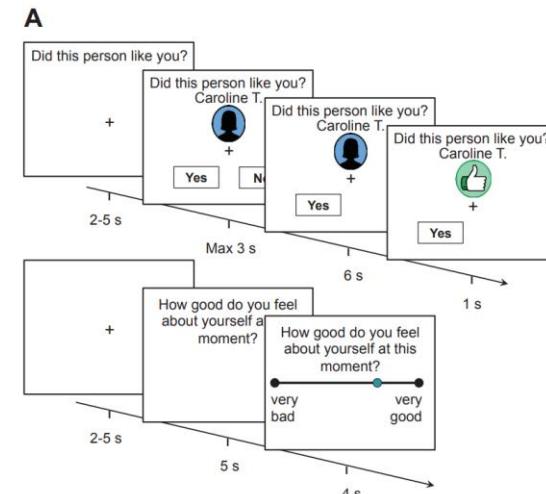


[Lockwood et al. \(2016\)](#)

A. Trial details



[Swart et al. \(2017\)](#)



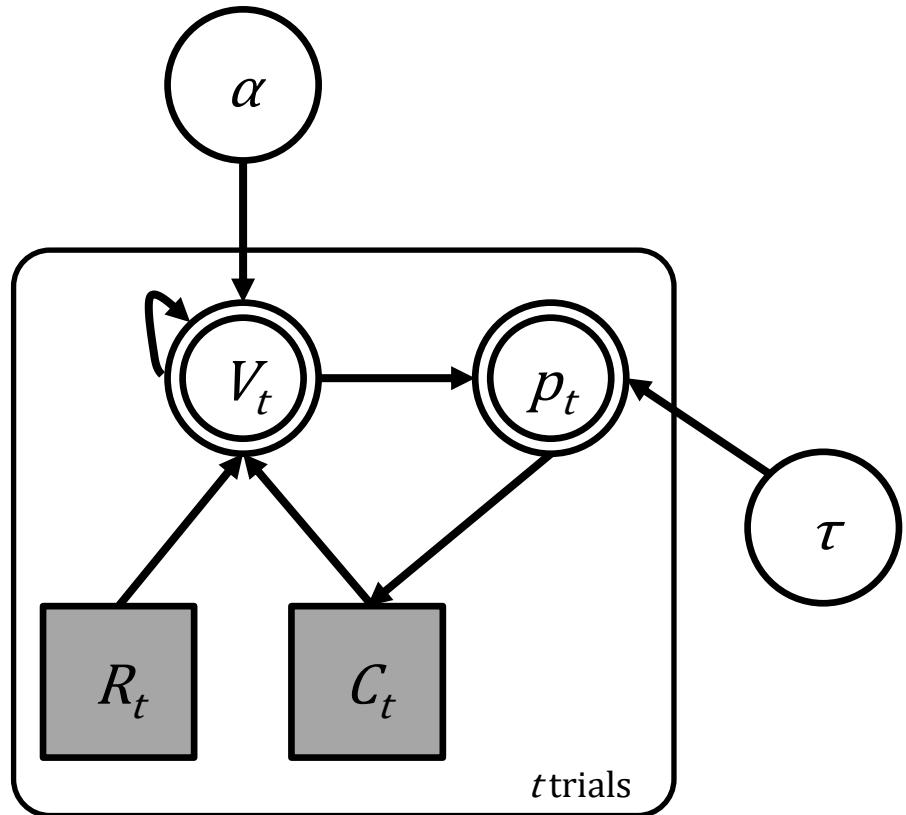
[Will et al. \(2017\)](#)

B

	Like (%)	Dislike (%)
1	85%	15%
2	70%	30%
3	30%	70%
4	15%	85%

RL – Implementation

cognitive model
statistics
computing



$$\alpha \sim Uniform(0, 1)$$

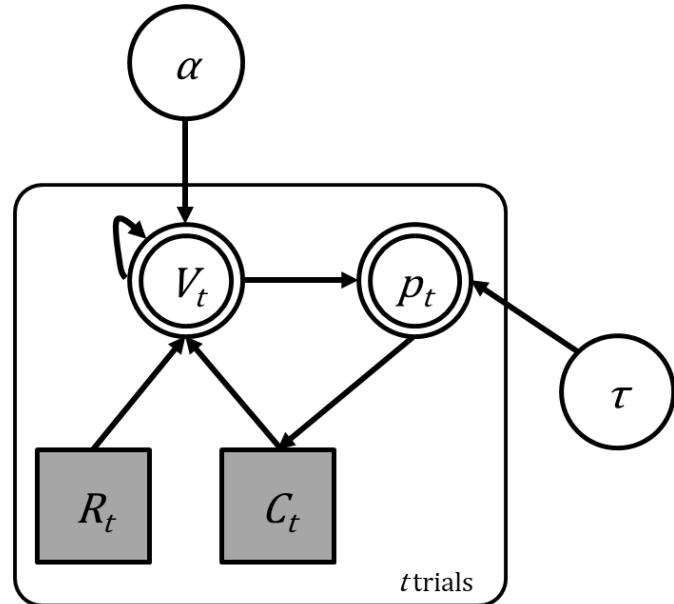
$$\tau \sim Uniform(0, 3)$$

$$p_t(C = A) = \frac{1}{1 + e^{\tau(V_t(B) - V_t(A))}}$$

$$V_{t+1}^c = V_t^C + \alpha (R_t - V_t^C)$$

RL – Implementation

cognitive model
statistics
computing



$$\alpha \sim Uniform(0, 1)$$

$$\tau \sim Uniform(0, 3)$$

$$p_t(C = A) = \frac{1}{1 + e^{\tau(V_t(B) - V_t(A))}}$$

$$V_{t+1}^c = V_t^c + \alpha (R_t - V_t^c)$$

```

transformed data {
  vector[2] initV;
  initV = rep_vector(0.0, 2);
}

model {
  vector[2] v[nTrials+1];
  real pe[nTrials];

  v[1] = initV;

  for (t in 1:nTrials) {
    choice[t] ~ categorical_logit( tau * v[t] );

    pe[t] = reward[t] - v[t,choice[t]];

    v[t+1] = v[t];
    v[t+1, choice[t]] = v[t, choice[t]] + lr * pe[t];
  }
}

```

RL - Implementation

cognitive model
statistics
computing

```
model {  
    vector[2] v[nTrials+1];  
    real pe[nTrials];  
  
    v[1] = initV;  
  
    for (t in 1:nTrials) {  
        choice[t] ~ categorical_logit( tau * v[t] );  
        pe[t] = reward[t] - v[t,choice[t]];  
  
        v[t+1] = v[t];  
        v[t+1, choice[t]] = v[t, choice[t]] + lr * pe[t];  
    }  
}
```

```
model {  
    vector[2] v;  
    real pe;  
  
    v = initV;  
  
    for (t in 1:nTrials) {  
        choice[t] ~ categorical_logit( tau * v );  
        pe = reward[t] - v[choice[t]];  
  
        v[choice[t]] = v[choice[t]] + lr * pe;  
    }  
}
```

RL – Fitting with Stan

cognitive model
statistics
computing

.../06.reinforcement_learning/_scripts/reinforcement_learning_single_parm_main.R

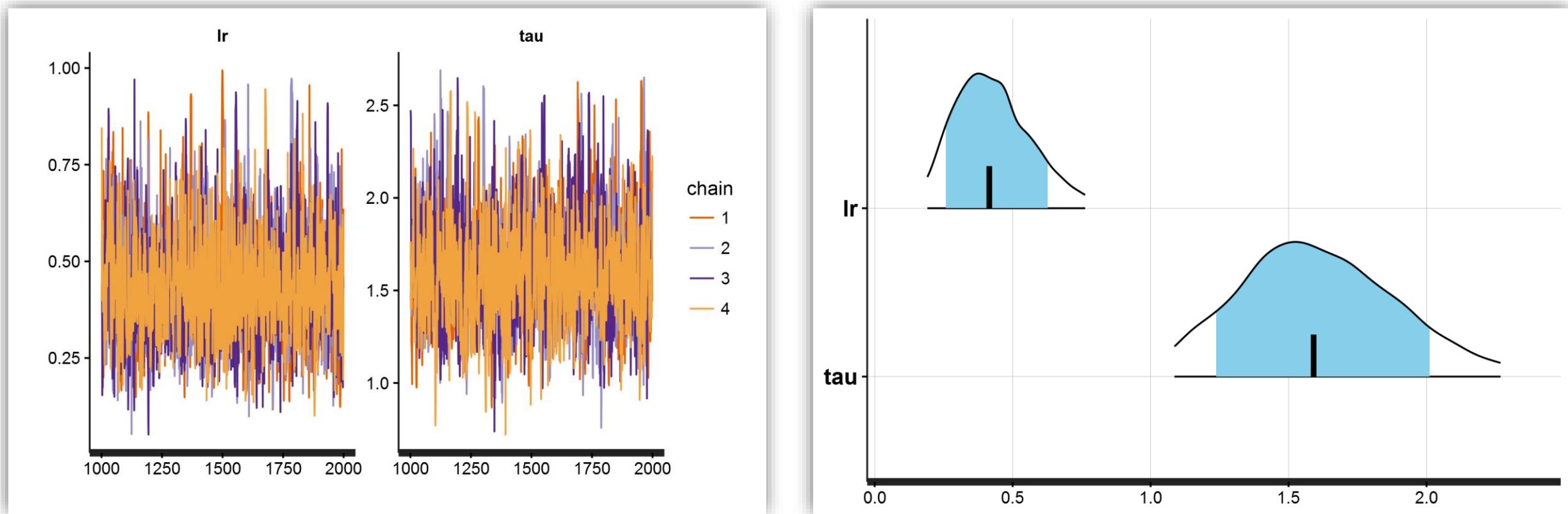
TASK: fit the model for single participants

```
> source('_scripts/reinforcement_learning_single_parm_main.R') # a function  
  
> fit_rl1 <- run_rl_sp(multiSubj = FALSE)
```

```
> load('_data/rl_sp_ss.RData')  
> head(rl_ss)  
     [,1] [,2]  
[1,]    2   -1  
[2,]    1    1  
[3,]    1    1  
[4,]    1    1  
[5,]    2   -1  
[6,]    1    1
```

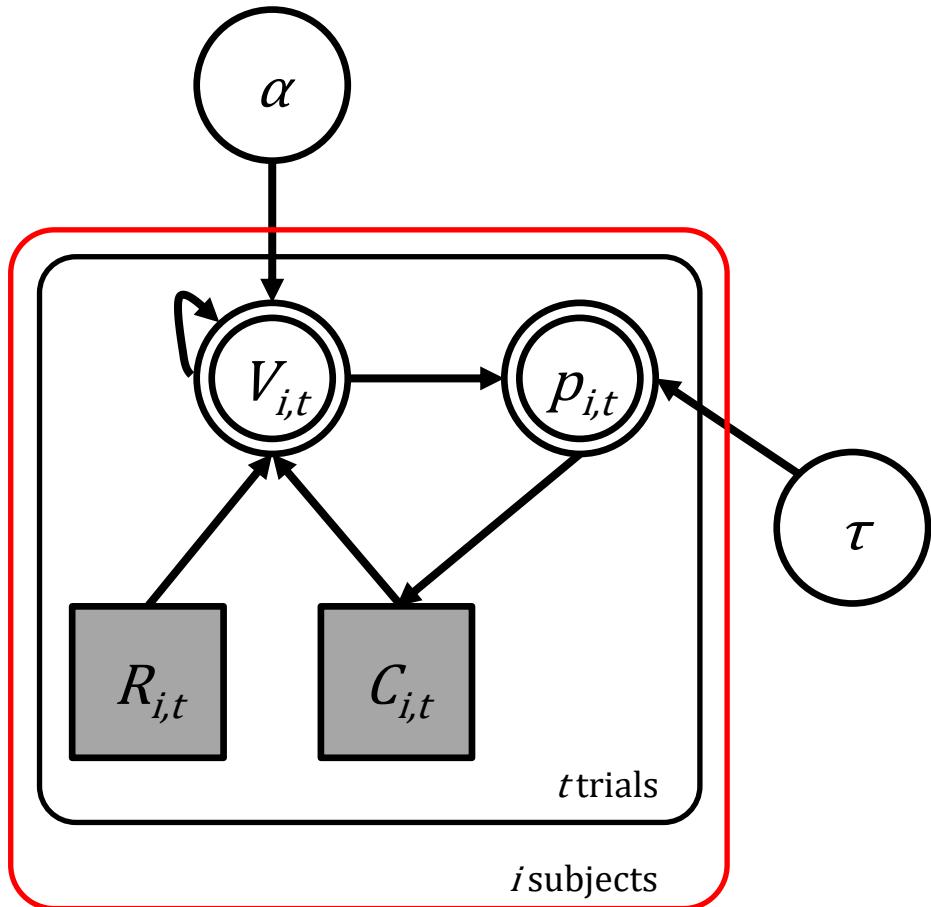
RL - MCMC Output

cognitive model
statistics
computing



Fitting Multiple Participants as ONE

cognitive model
statistics
computing



```
model {  
    for (s in 1:nSubjects) {  
        vector[2] v;  
        real pe;  
        v = initV;  
  
        for (t in 1:nTrials) {  
            choice[s,t] ~ categorical_logit( tau * v );  
            pe = reward[s,t] - v[choice[s,t]];  
            v[choice[s,t]] = v[choice[s,t]] + lr * pe;  
        }  
    }  
}
```

Exercise X

cognitive model
statistics
computing

.../06.reinforcement_learning/_scripts/reinforcement_learning_single_parm_main.R

TASK:

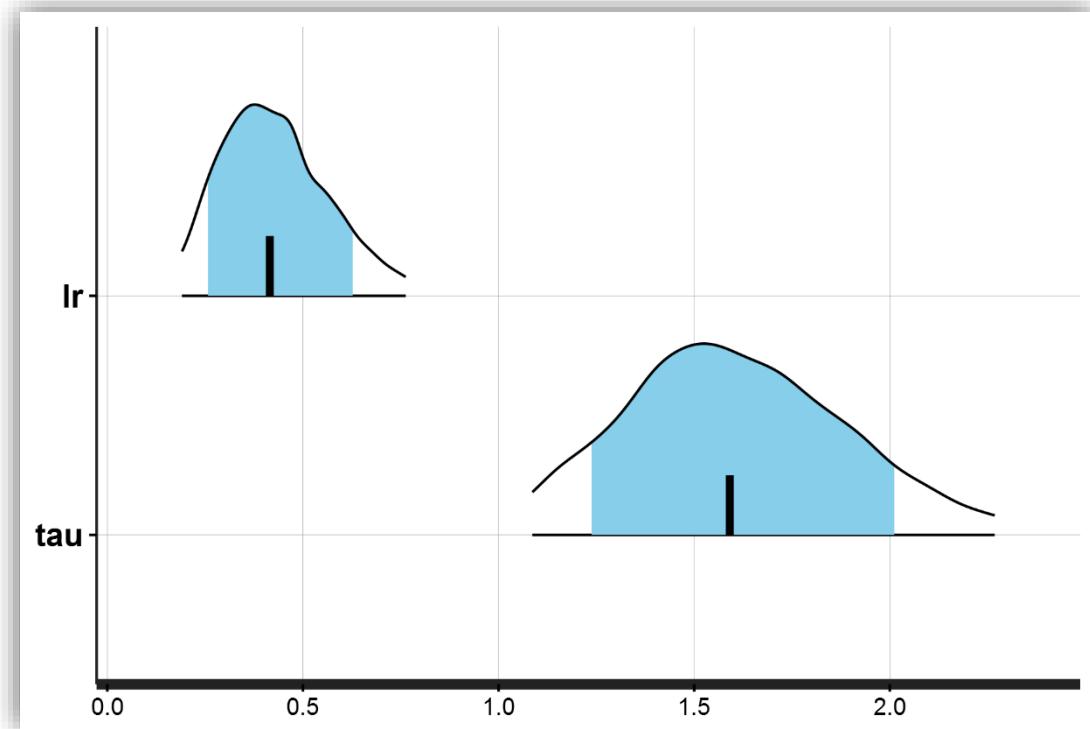
- (1) complete the model (Tip: the for-loop)
- (2) fit the model for multiple participants (assuming same parameters)

```
> source('_scripts/reinforcement_learning_single_parm_main.R')  
  
> fit_rl2 <- run_rl_sp(multiSubj = TRUE)
```

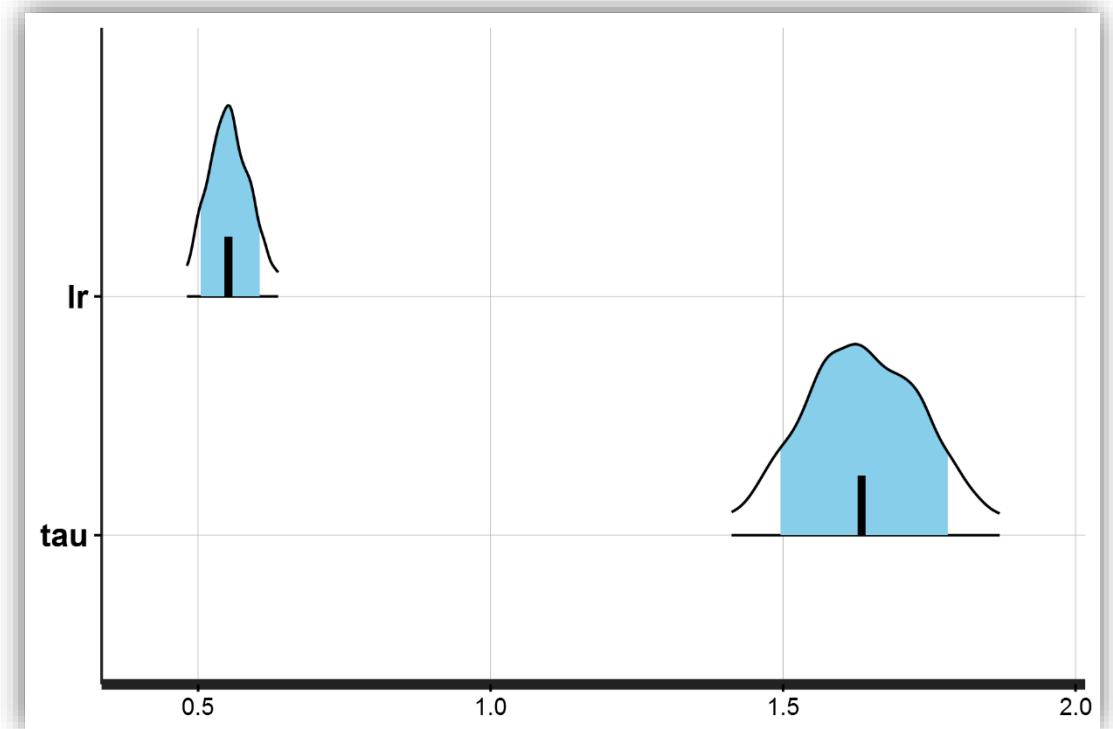
Exercise X

cognitive model
statistics
computing

$N = 1$

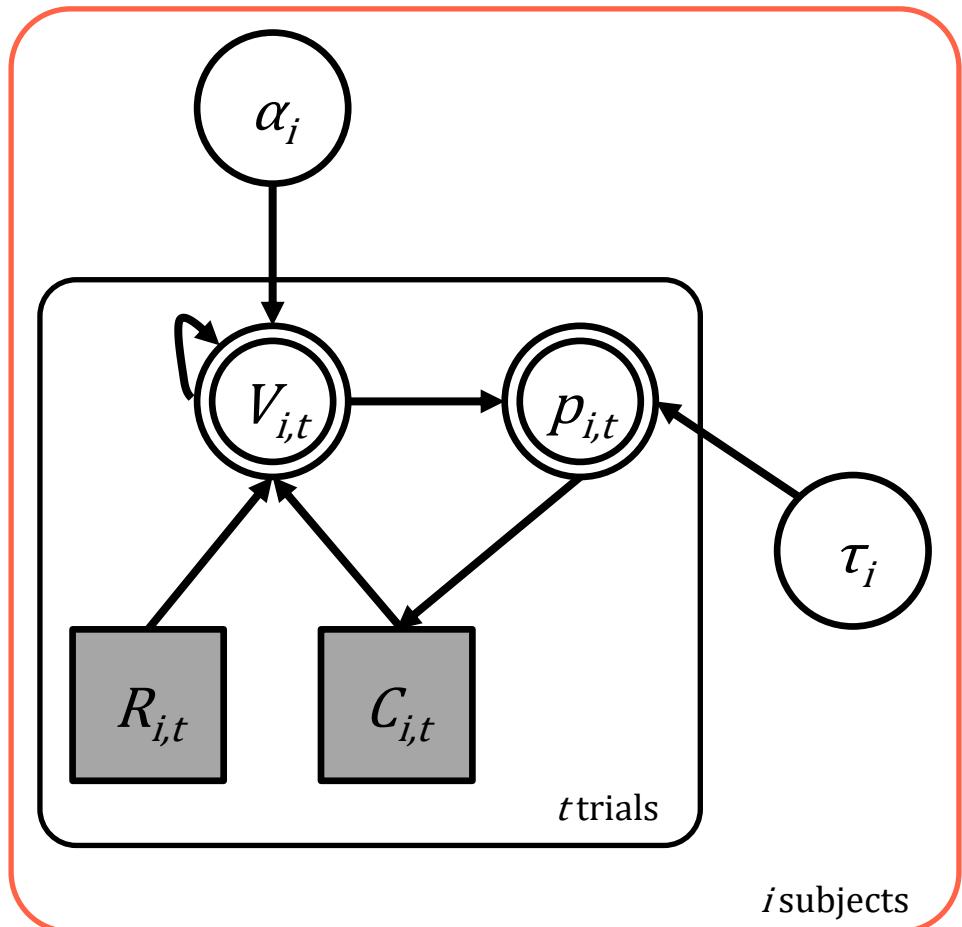


$N = 10$



Fitting Multiple Participants Independently

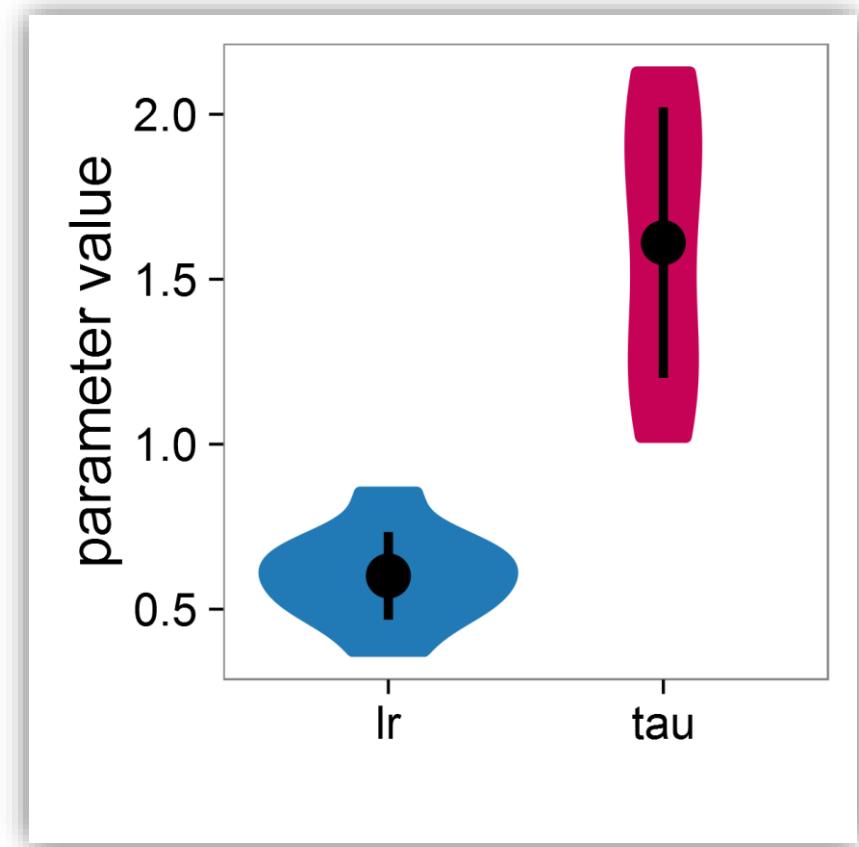
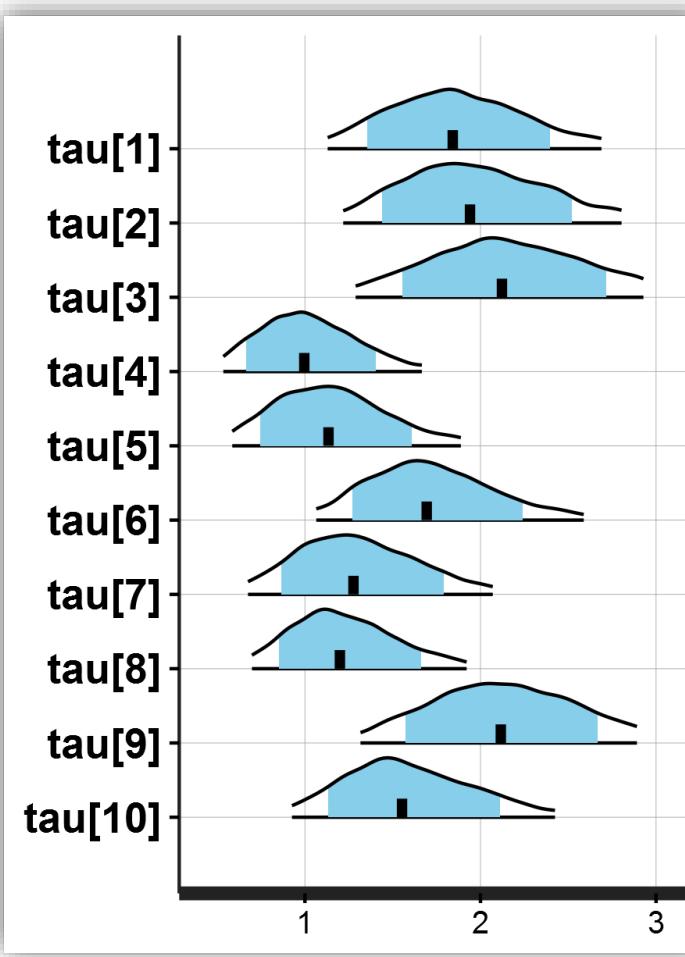
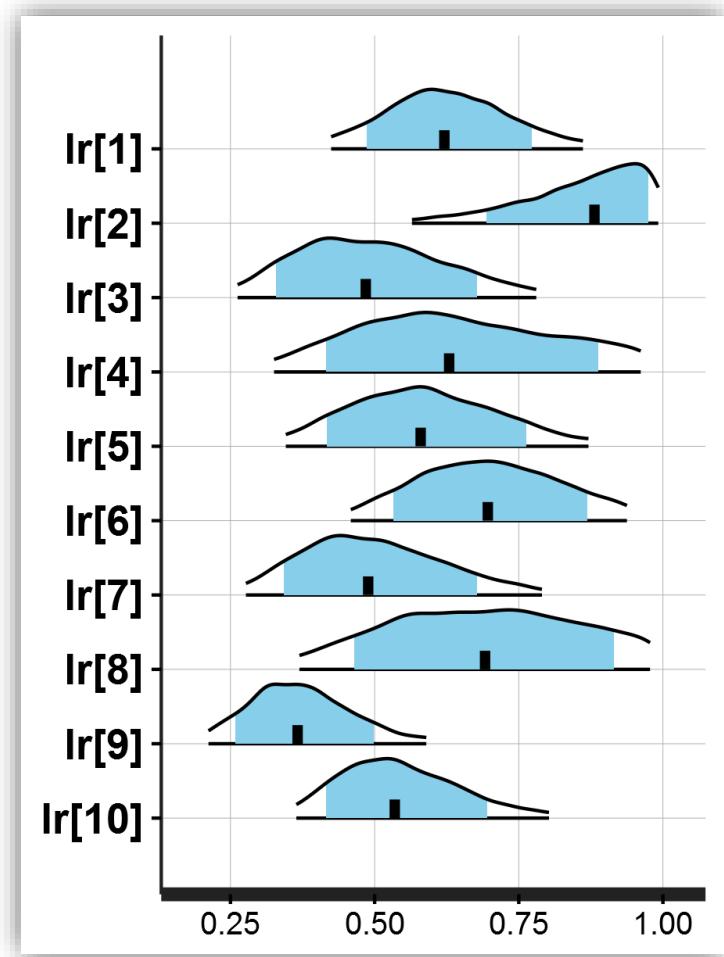
cognitive model
statistics
computing



```
model {  
    for (s in 1:nSubjects) {  
        vector[2] v;  
        real pe;  
        v = initV;  
  
        for (t in 1:nTrials) {  
            choice[s,t] ~ categorical_logit( tau[s] * v );  
            pe = reward[s,t] - v[choice[s,t]];  
            v[choice[s,t]] = v[choice[s,t]] + lr[s] * pe;  
        }  
    }  
}
```

Individual Fitting

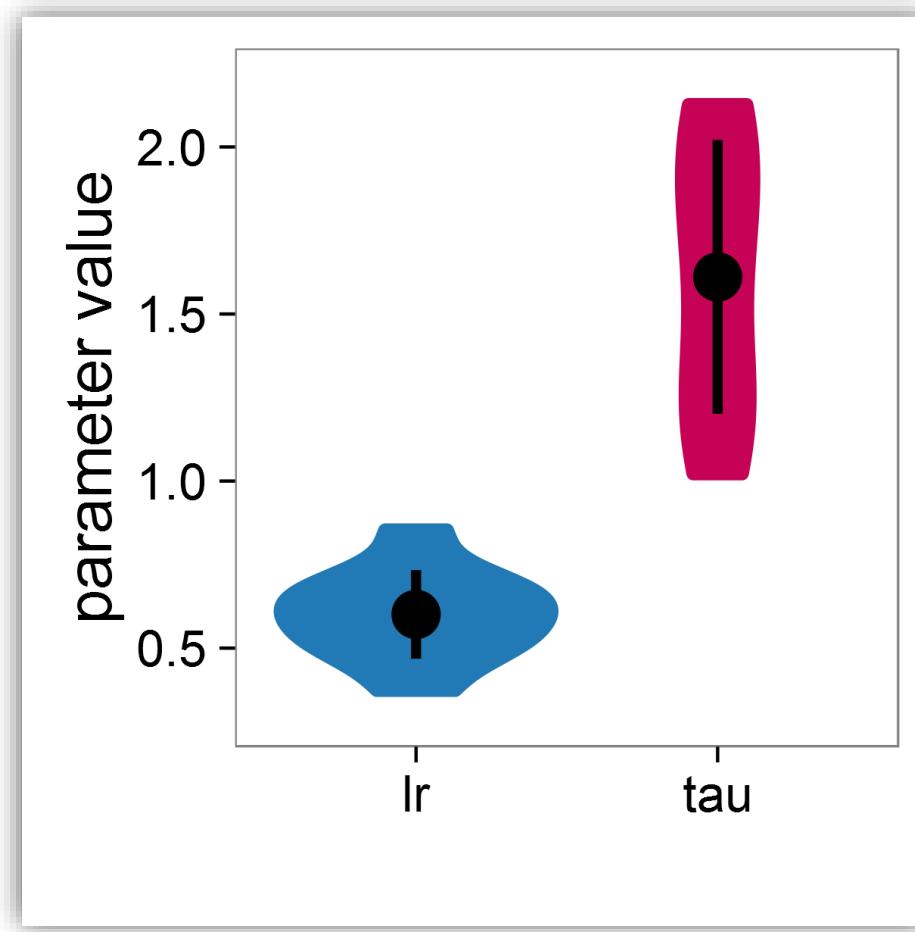
cognitive model
statistics
computing



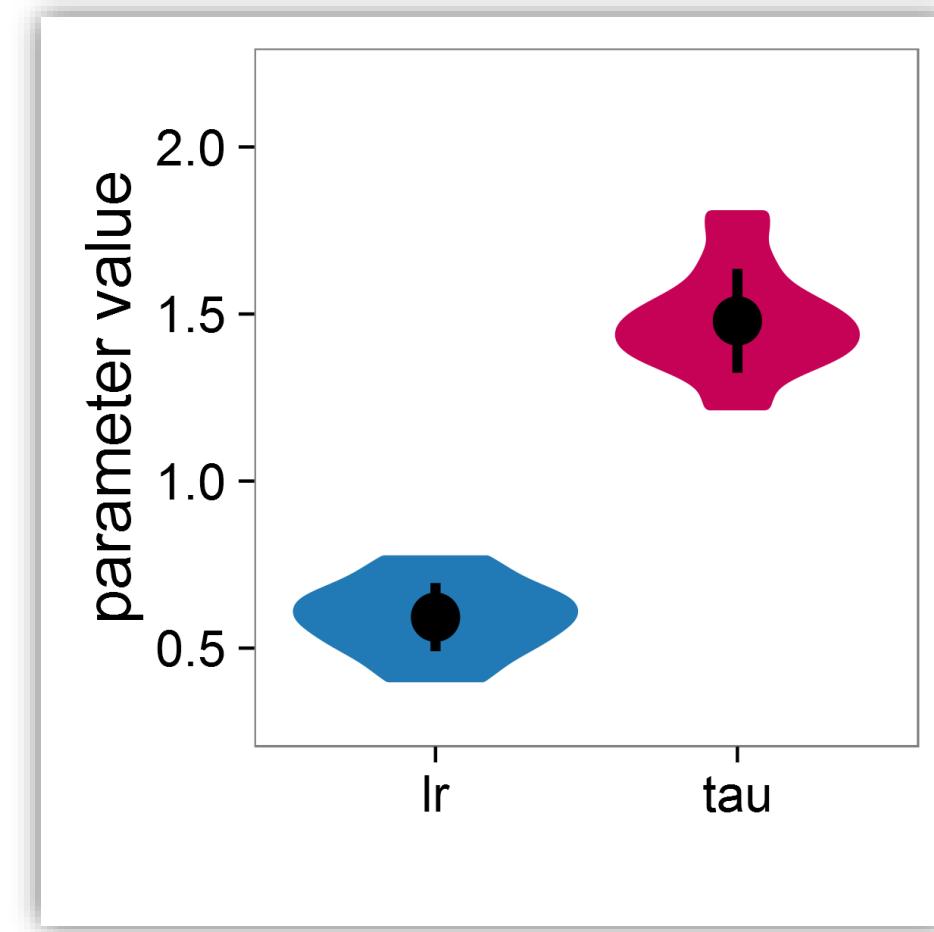
Comparing with True Parameters

cognitive model
statistics
computing

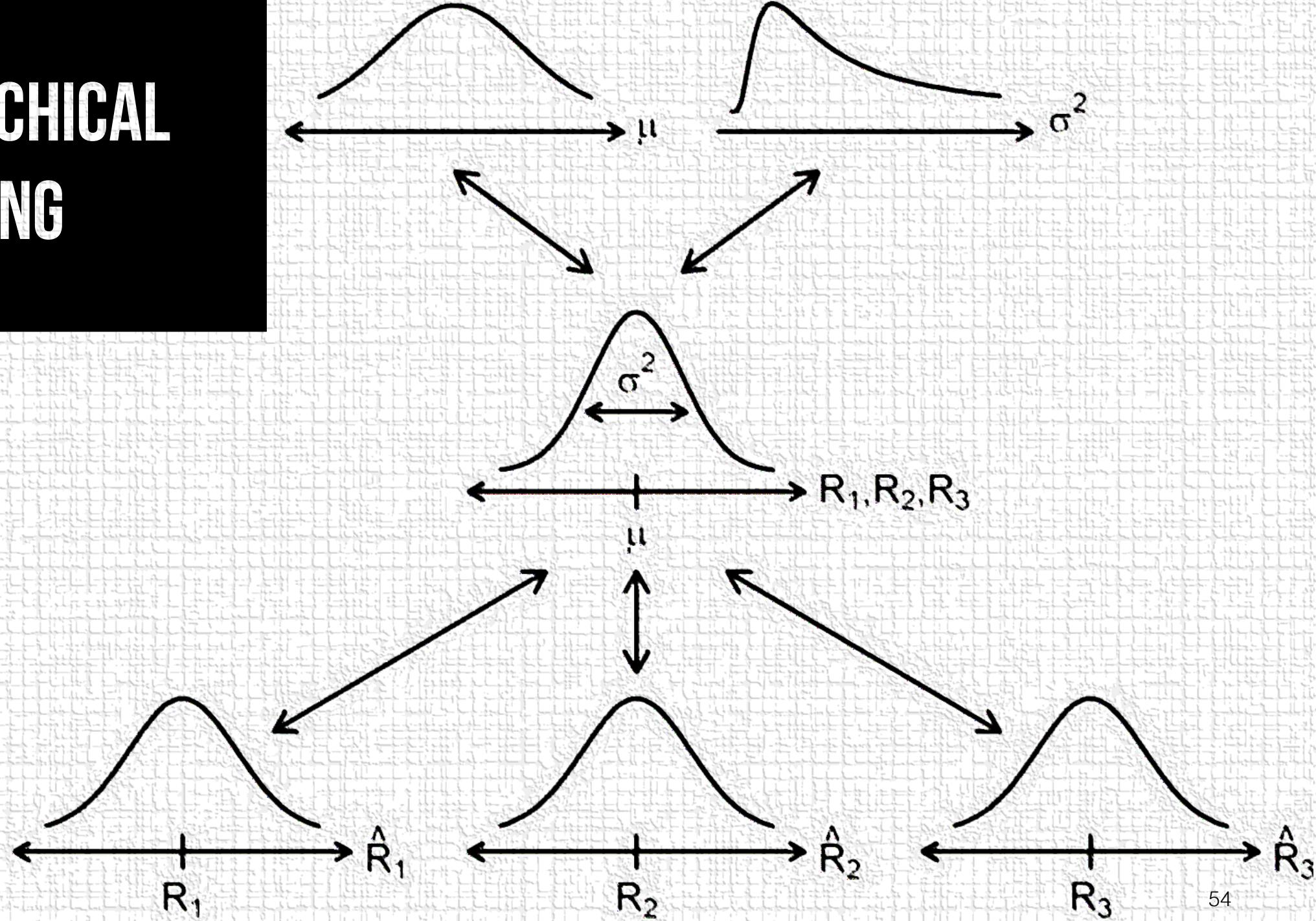
Posterior Means

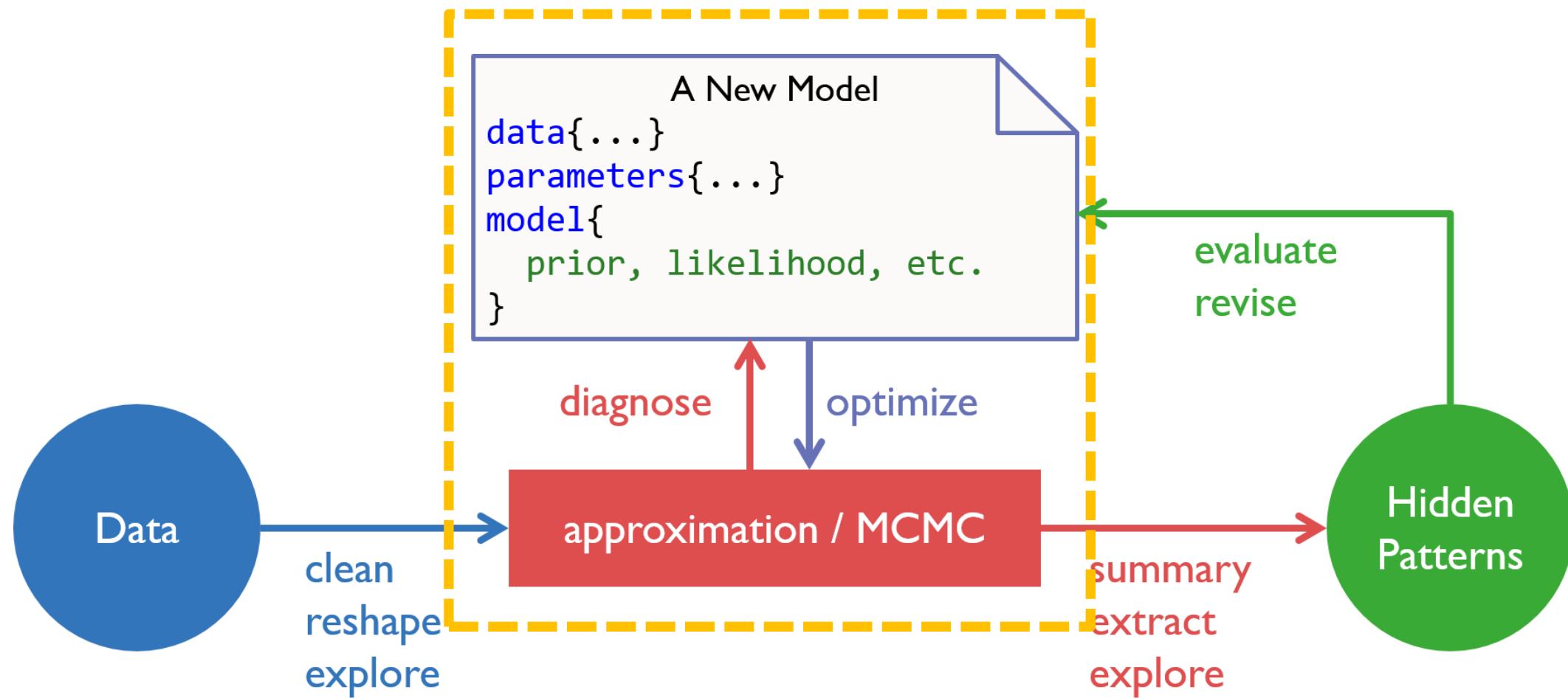


True Parameters

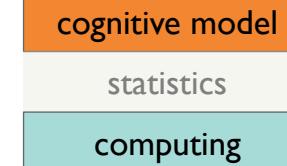


HIERARCHICAL MODELING



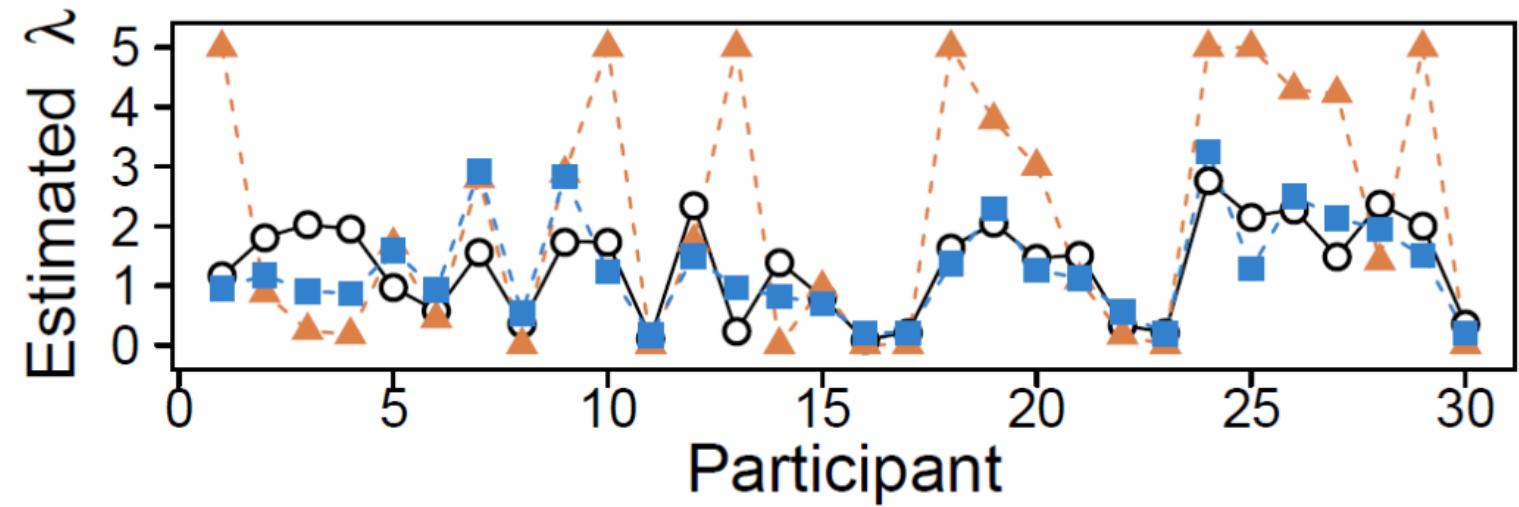


Why Hierarchical Bayesian Cognitive Modeling?



Simulation study

Hierarchical Bayesian ■
Maximum likelihood ▲
Actual values ○



Why Hierarchical Bayesian Cognitive Modeling?

cognitive model
statistics
computing

Fixed effects

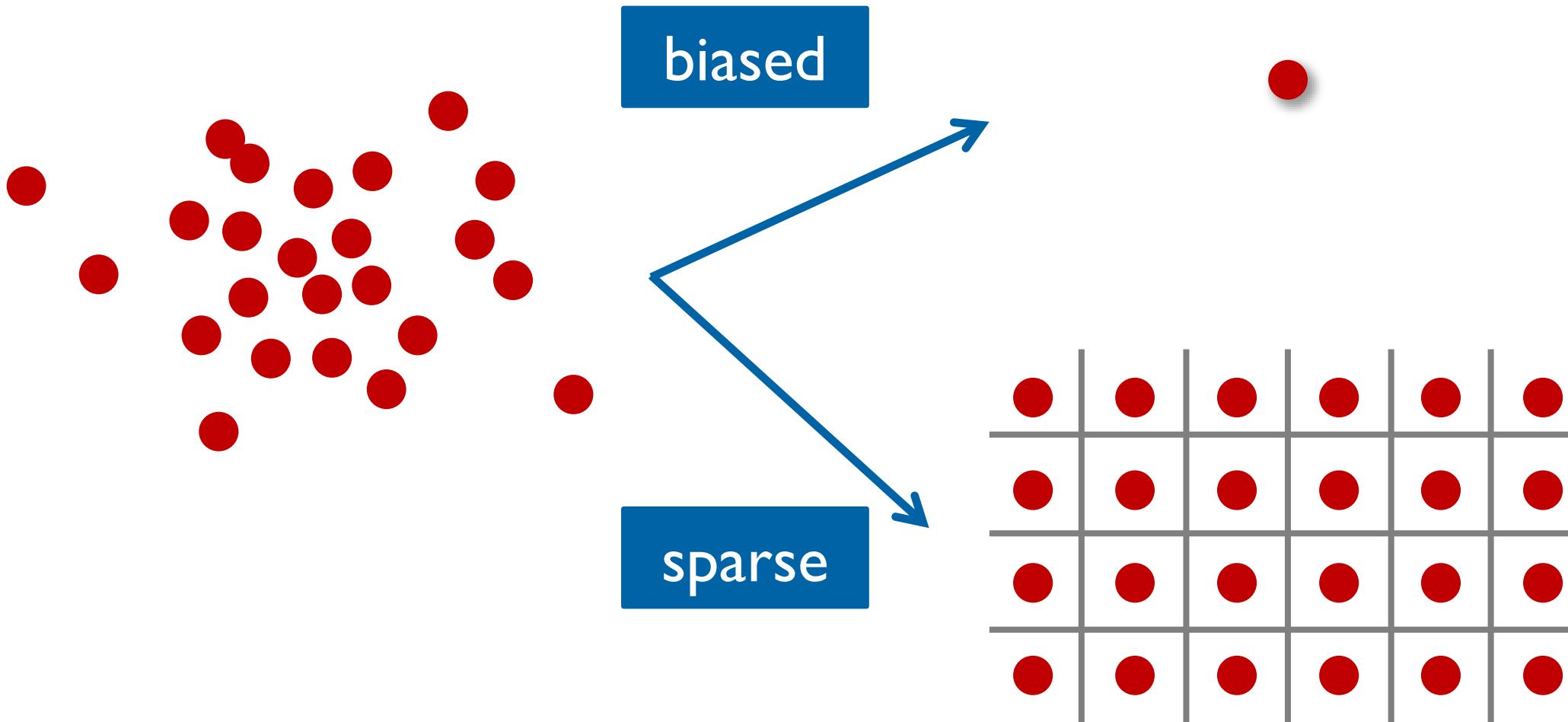
- all subjects are fitted with the same set of parameters
- worse model fit than “random effects”

Random effects

- each subject is fitted independently of the others
- best model fit for each subject
- parameter estimates can be noisy

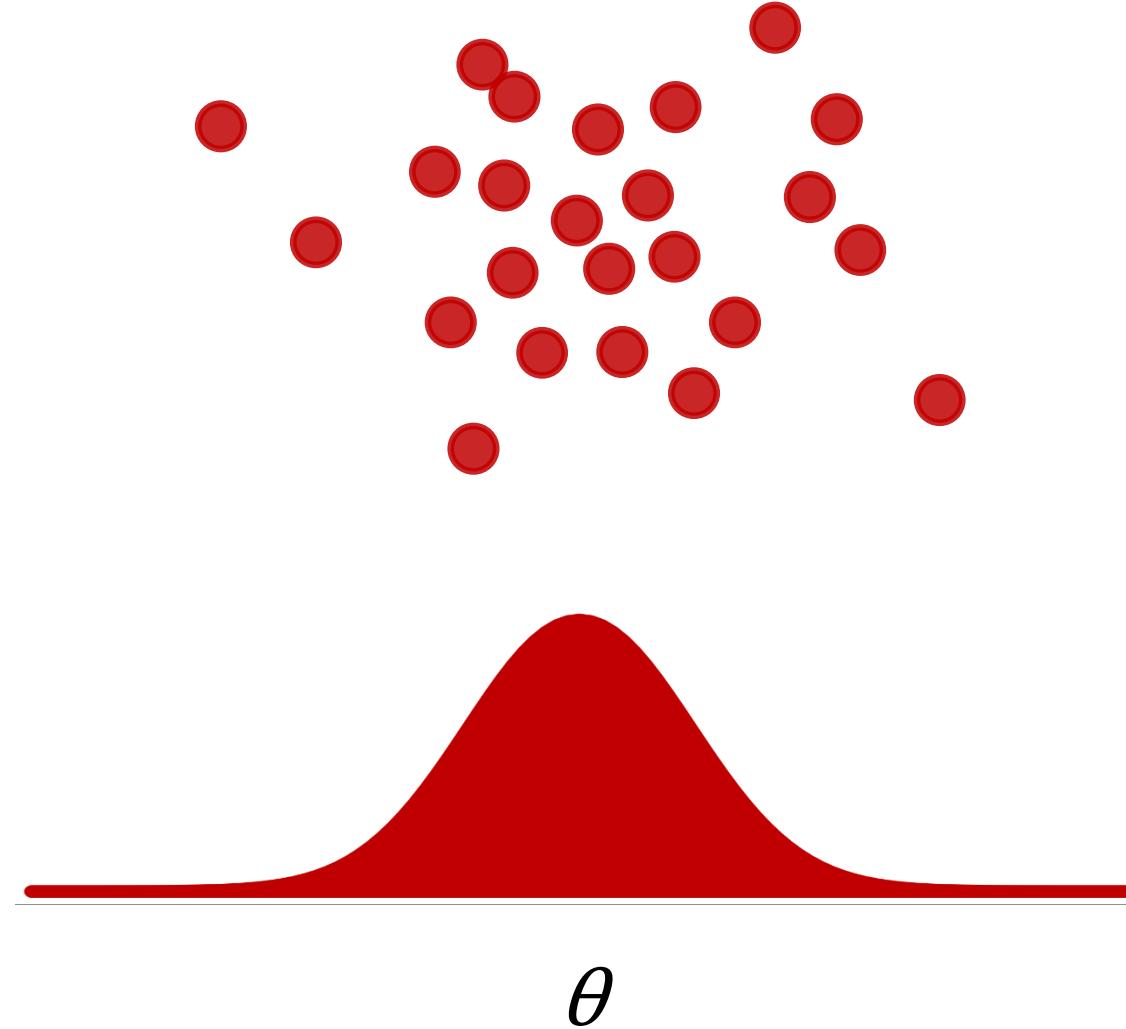
Fitting Multiple Participants

cognitive model
statistics
computing



Fitting Multiple Participants

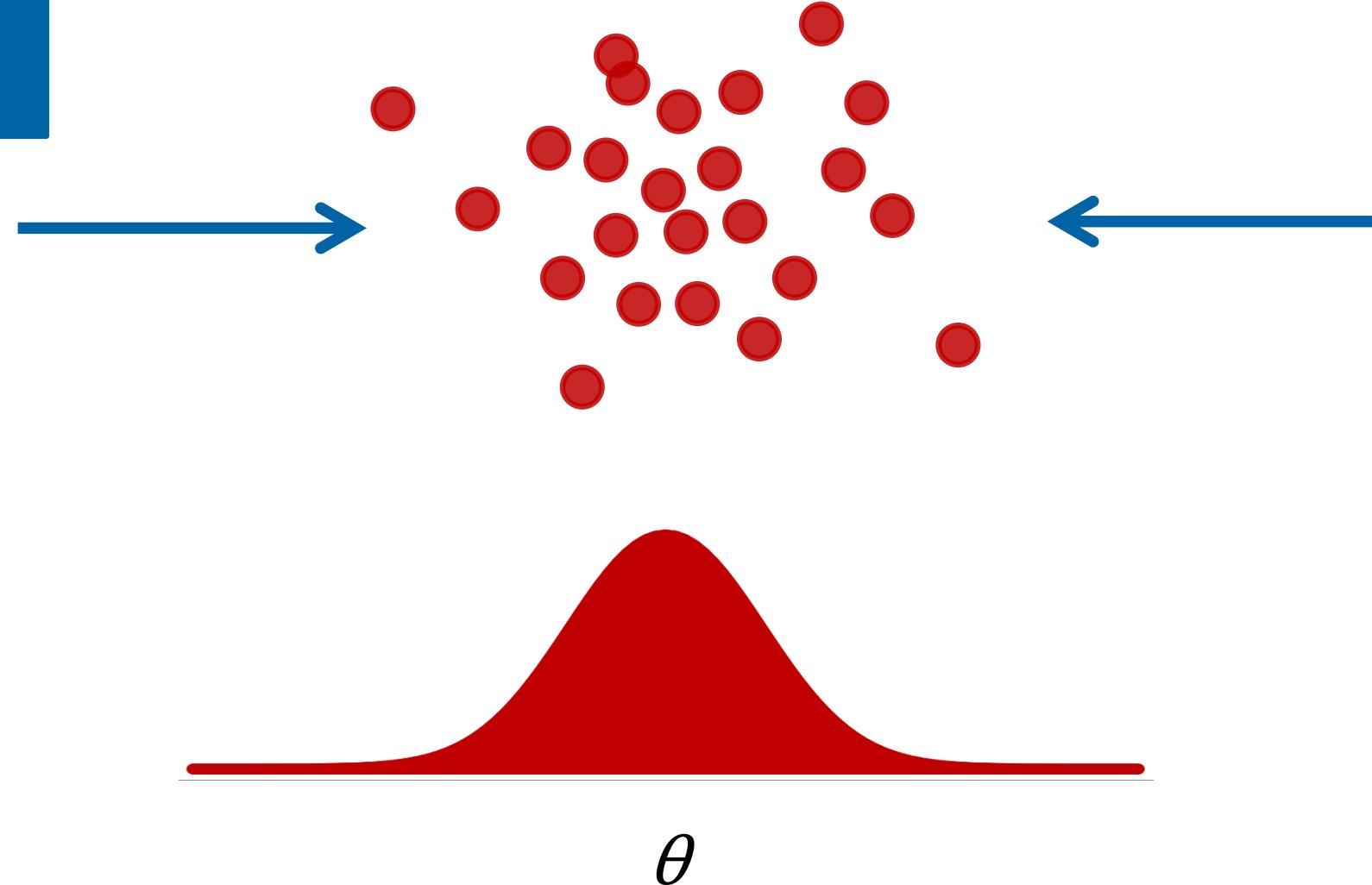
cognitive model
statistics
computing



Fitting Multiple Participants

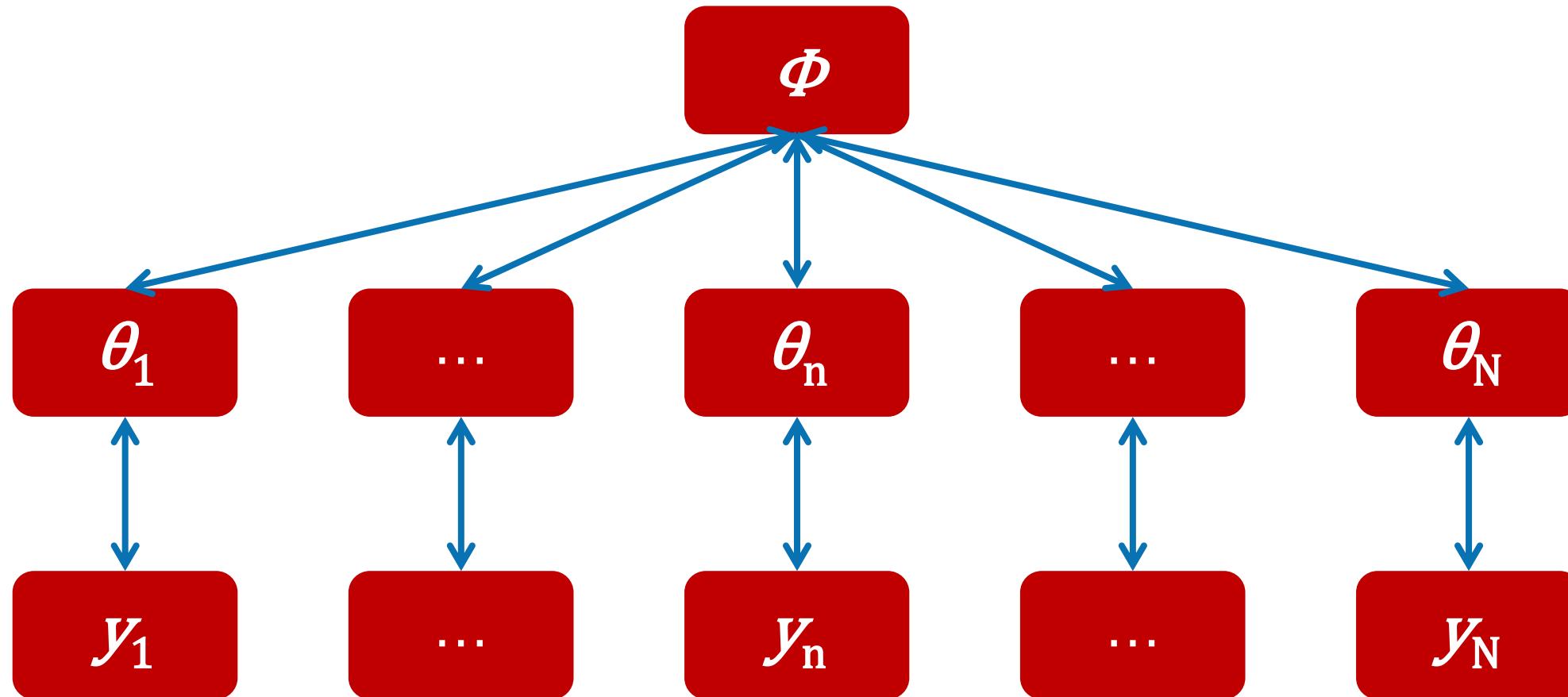
cognitive model
statistics
computing

shrinkage effect



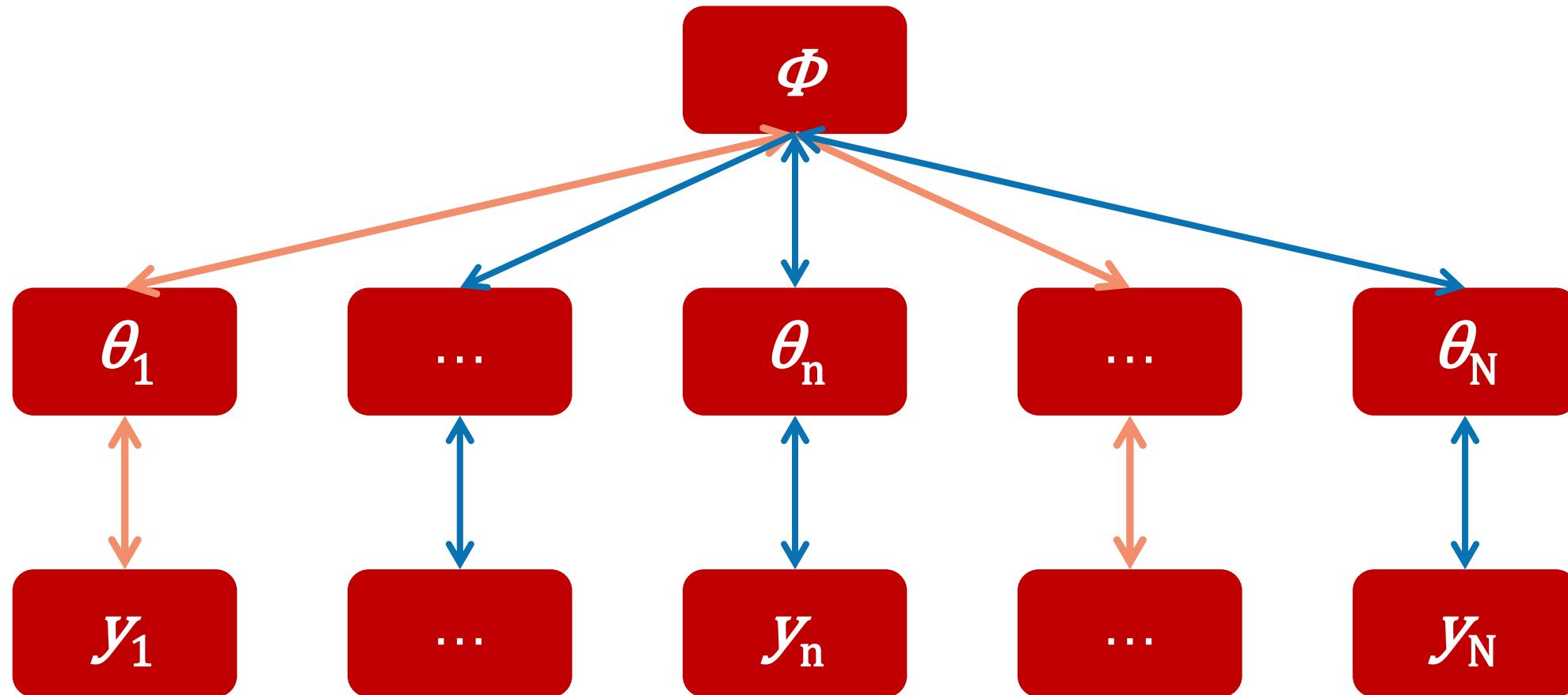
Hierarchical Structure

cognitive model
statistics
computing



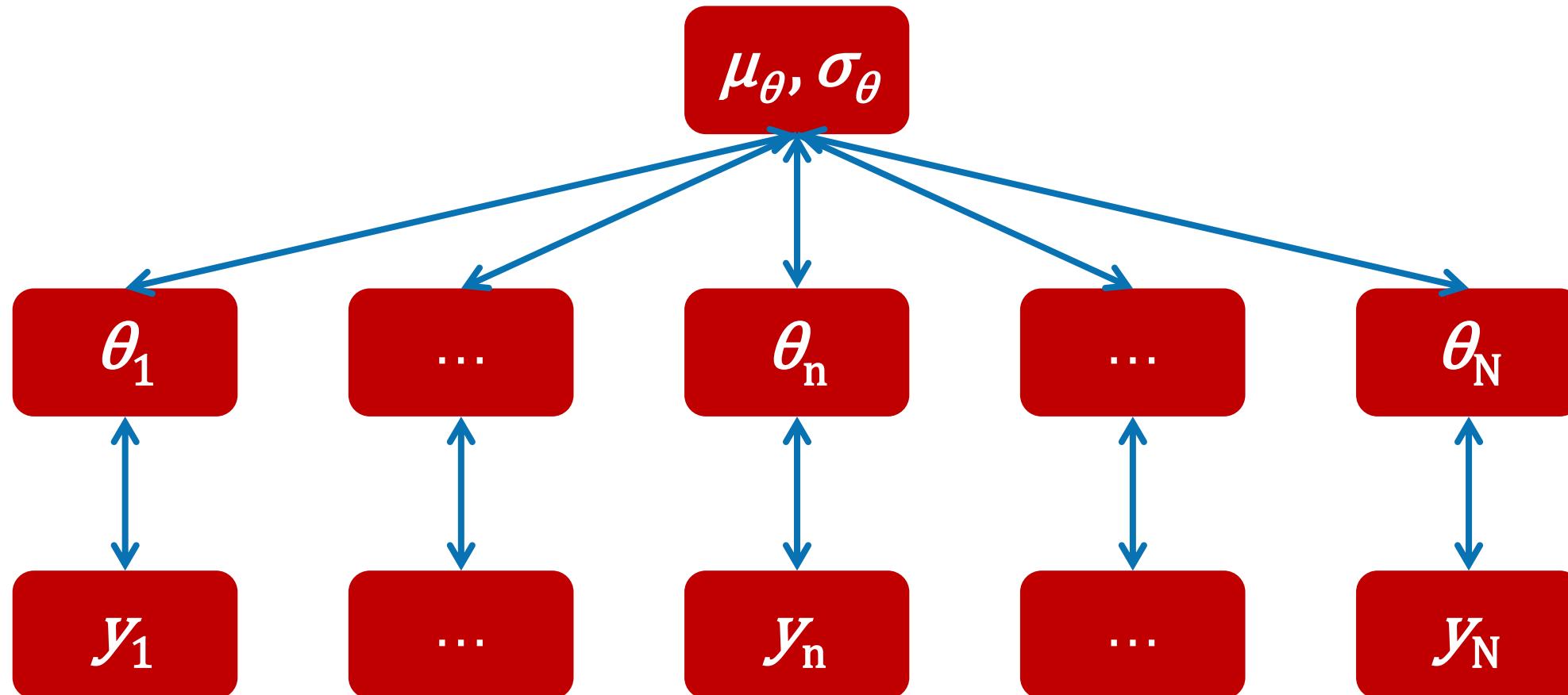
Hierarchical Structure

cognitive model
statistics
computing

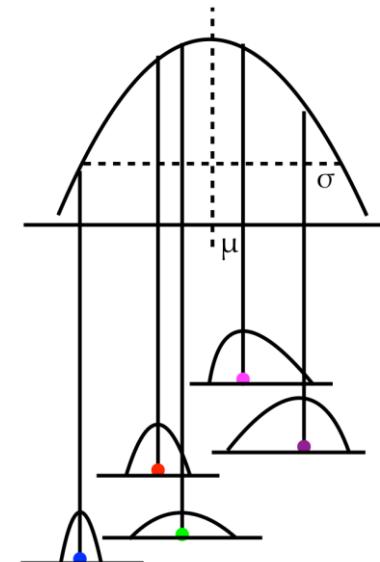
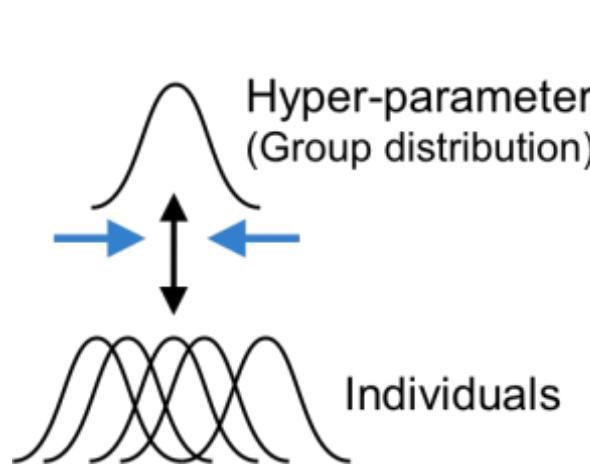
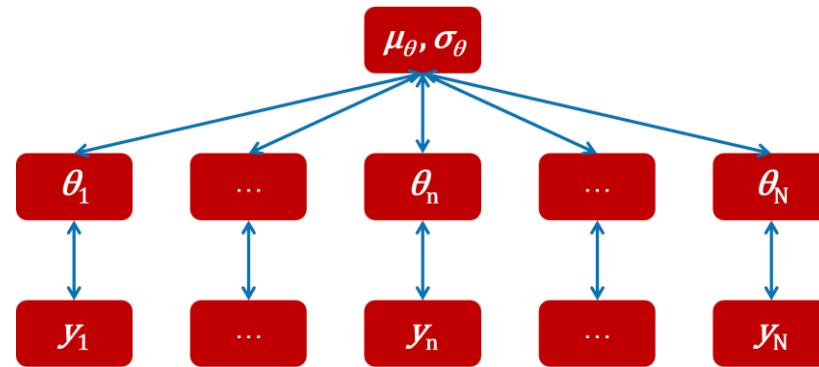


Hierarchical Structure

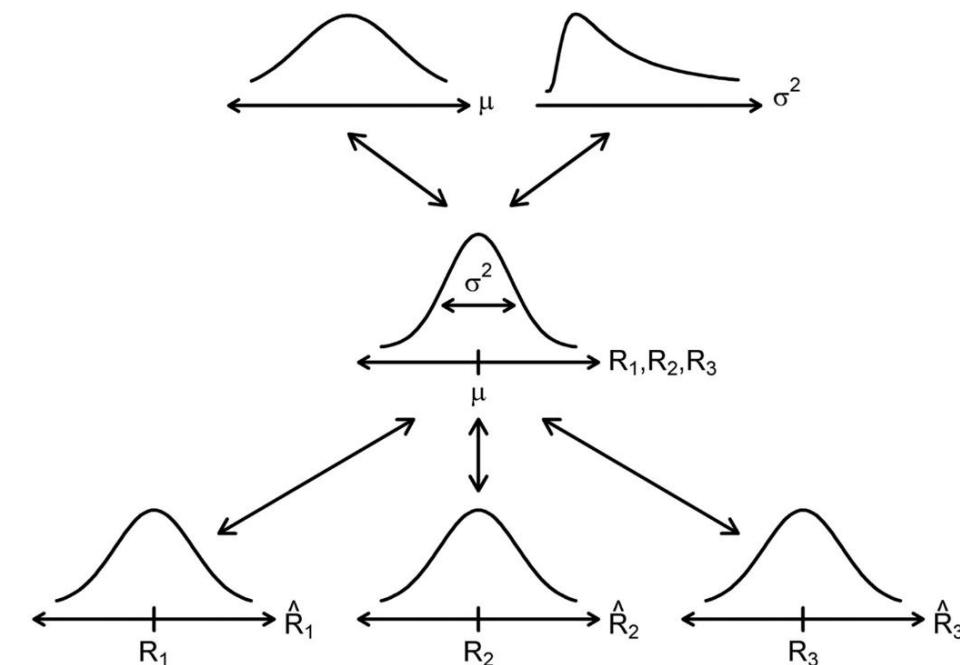
cognitive model
statistics
computing



Hierarchical Structure



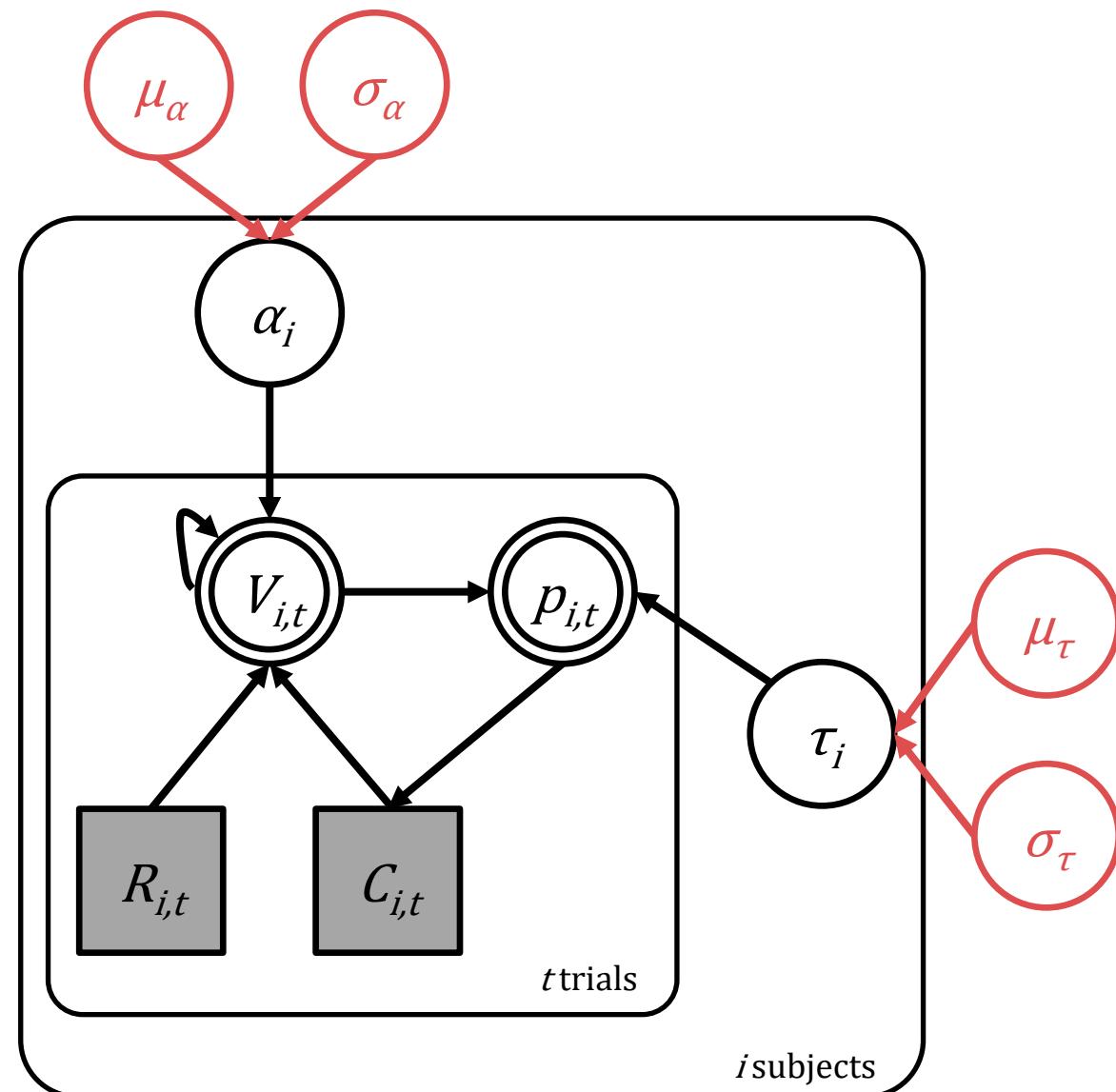
$$P(\Theta, \Phi | D) = \frac{P(D | \Theta, \Phi) P(\Theta, \Phi)}{P(D)} \propto P(D | \Theta) P(\Theta | \Phi) P(\Phi)$$



Hierarchical RL Model

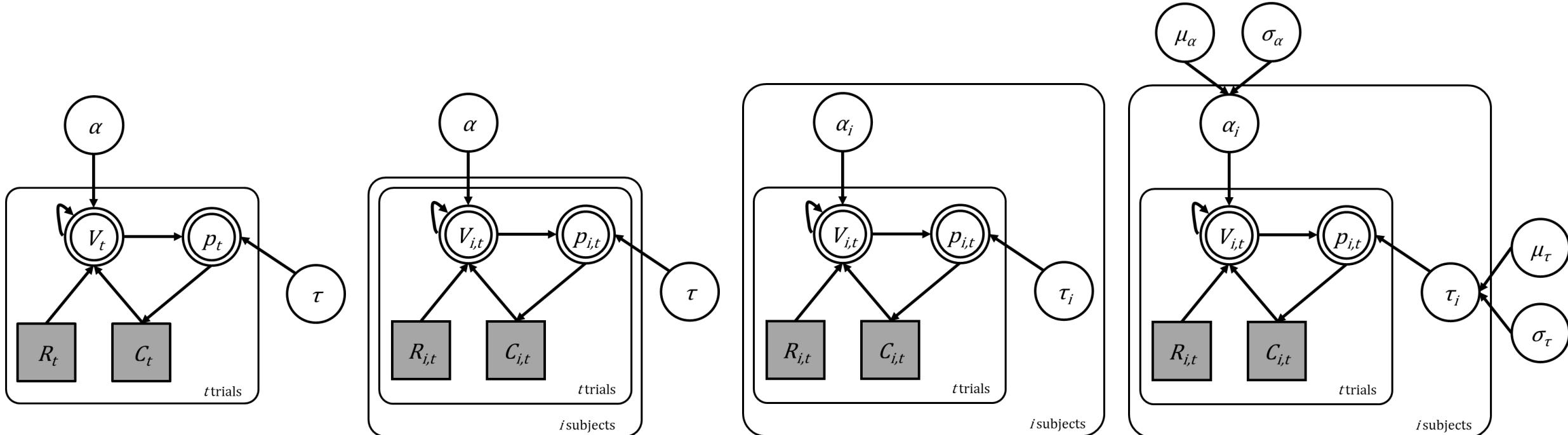
cognitive model
statistics
computing

Voilà!



HOW DID WE GET HERE?

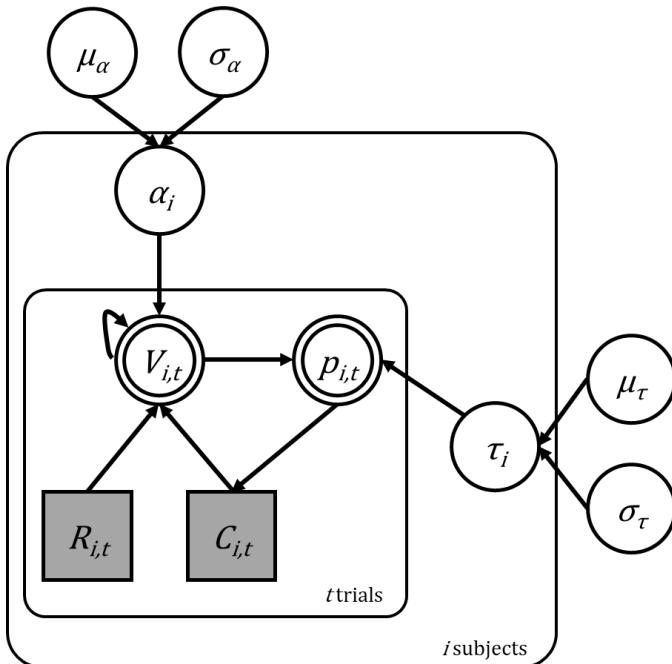
cognitive model
statistics
computing



The cognitive model *per se* is the same!

Implementing Hierarchical RL Model

cognitive model
statistics
computing



$$\begin{aligned}\mu_\alpha &\sim Uniform(0,1) \\ \sigma_\alpha &\sim halfCauchy(0,1) \\ \mu_\tau &\sim Uniform(0,3) \\ \sigma_\tau &\sim halfCauchy(0,3) \\ \alpha_i &\sim Normal(\mu_\alpha, \sigma_\alpha)_{\mathcal{T}(0,1)} \\ \tau_i &\sim Normal(\mu_\tau, \sigma_\tau)_{\mathcal{T}(0,3)}\end{aligned}$$

$$p_{i,t}(C = A) = \frac{1}{1 + e^{\tau_i(V_{i,t}(B) - V_{i,t}(A))}}$$

$$V_{i,t+1}^c = V_{i,t}^C + \alpha_i(R_{i,t} - V_{i,t}^C)$$

```
parameters {
    real<lower=0,upper=1> lr_mu;
    real<lower=0,upper=3> tau_mu;
    real<lower=0> lr_sd;
    real<lower=0> tau_sd;
    real<lower=0,upper=1> lr[nSubjects];
    real<lower=0,upper=3> tau[nSubjects];
}

model {
    lr_sd ~ cauchy(0,1);
    tau_sd ~ cauchy(0,3);
    lr ~ normal(lr_mu, lr_sd) ;
    tau ~ normal(tau_mu, tau_sd) ;

    for (s in 1:nSubjects) {
        vector[2] v;
        real pe;
        v = initV;

        for (t in 1:nTrials) {
            choice[s,t] ~ categorical_logit( tau[s] * v );
            pe = reward[s,t] - v[choice[s,t]];
            v[choice[s,t]] = v[choice[s,t]] + lr[s] * pe;
        }
    }
}
```

Exercise XI

cognitive model
statistics
computing

.../06.reinforcement_learning/_scripts/reinforcement_learning_multi_parm_main.R

TASK: (1) complete the model (TIP: individual ~ group)
(2) fit the hierarchical RL model

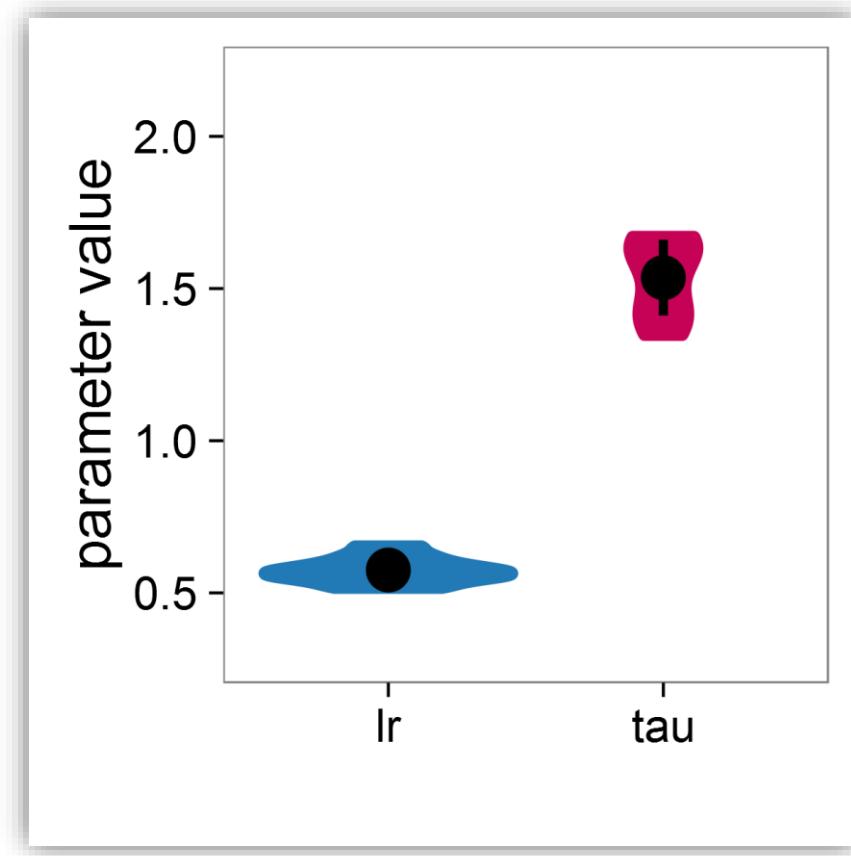
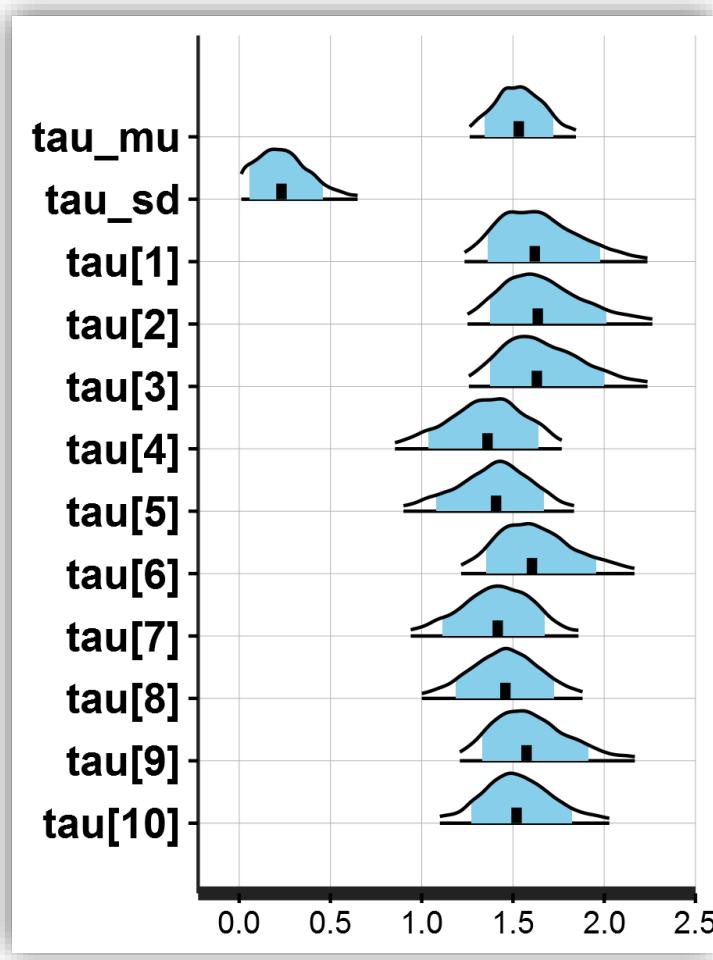
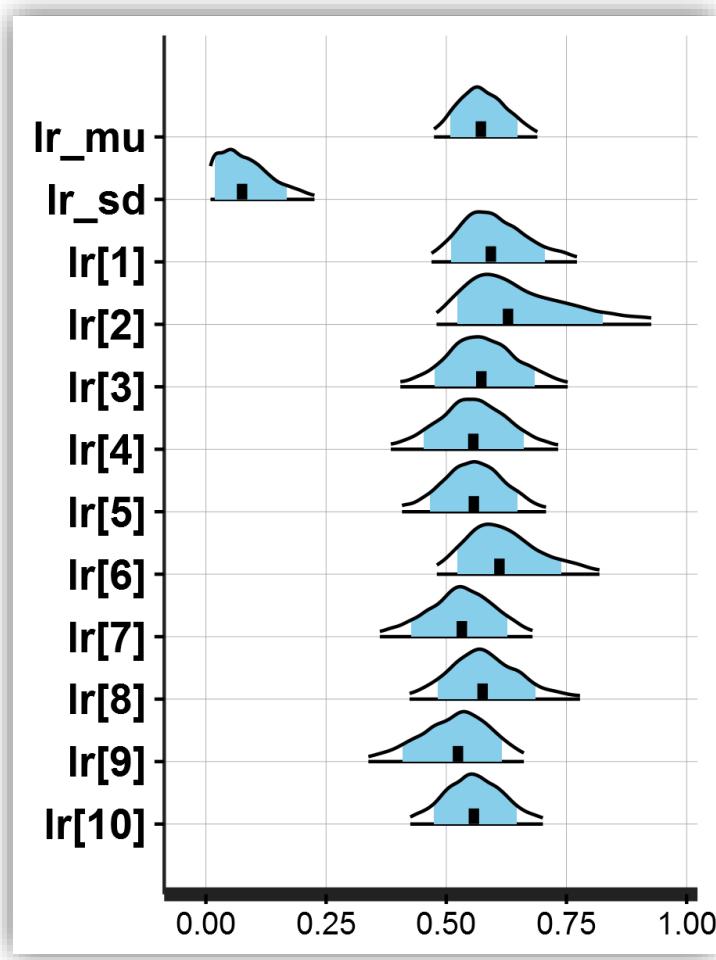
```
> source('_scripts/reinforcement_learning_multi_parm_main.R')  
  
> fit_rl3 <- run_rl_mp( modelType = 'hrch' )
```

In addition: Warning messages:

1: There were 97 divergent transitions after warmup. Increasing adapt_delta above 0.8 may help. See <http://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup>
2: Examine the pairs() plot to diagnose sampling problems

Hierarchical Fitting*

cognitive model
statistics
computing

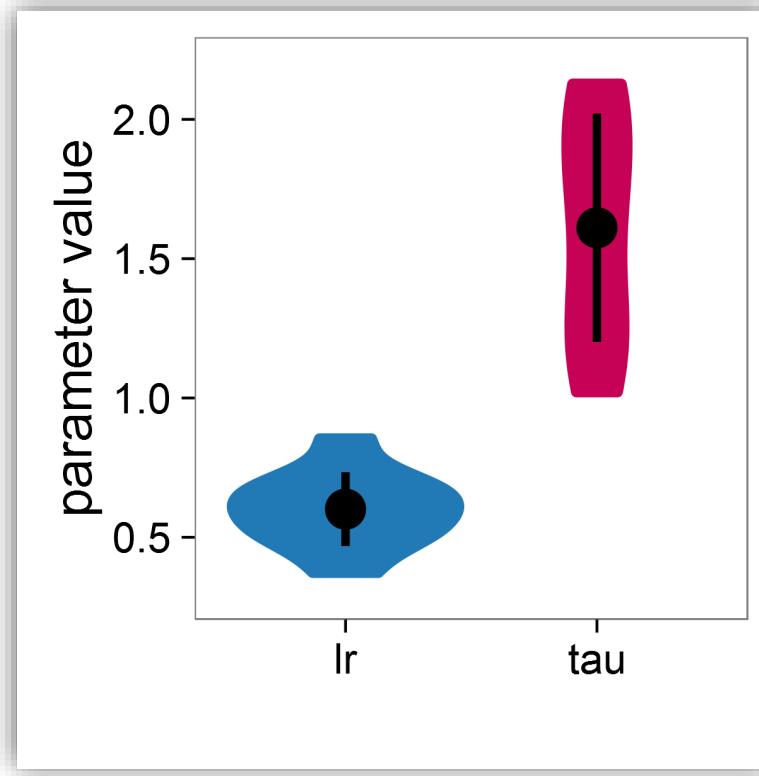


*: adapt_delta=0.999, max_treedepth=100

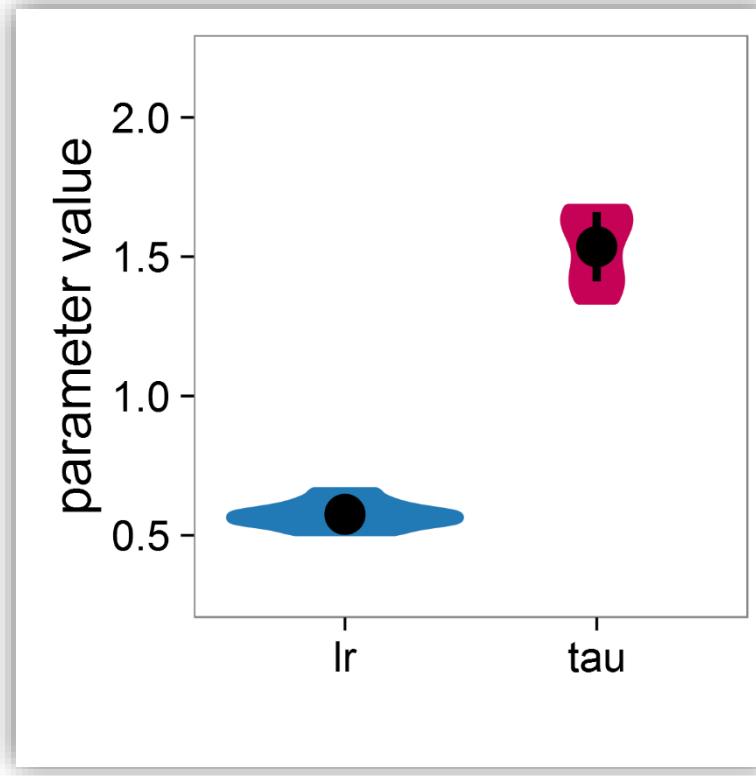
Comparing with True Parameters

cognitive model
statistics
computing

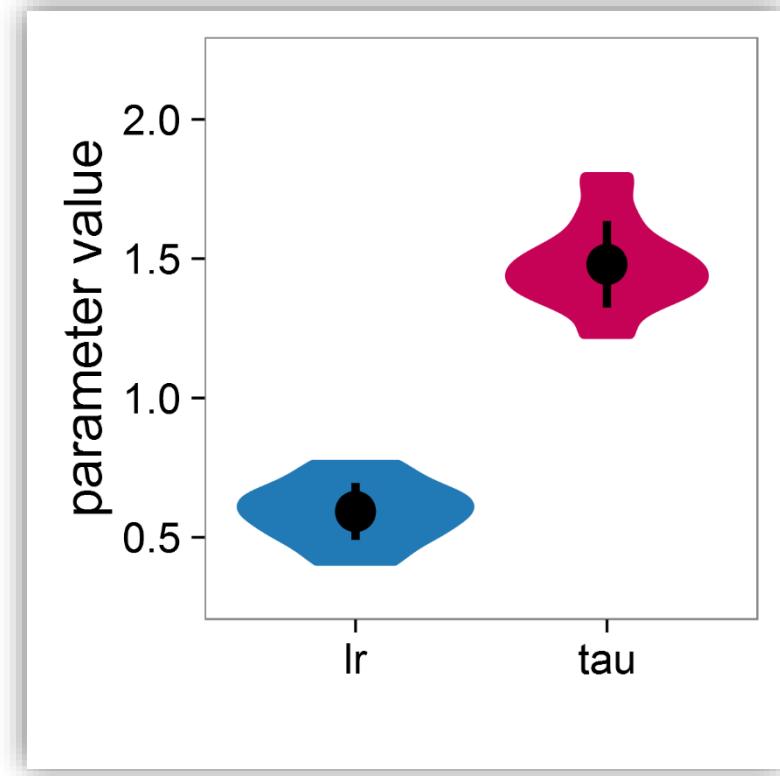
Posterior Means (indv)



Posterior Means (hrch)*



True Parameters



*: adapt_delta=0.999, max_treedepth=100

Group-level Parameters

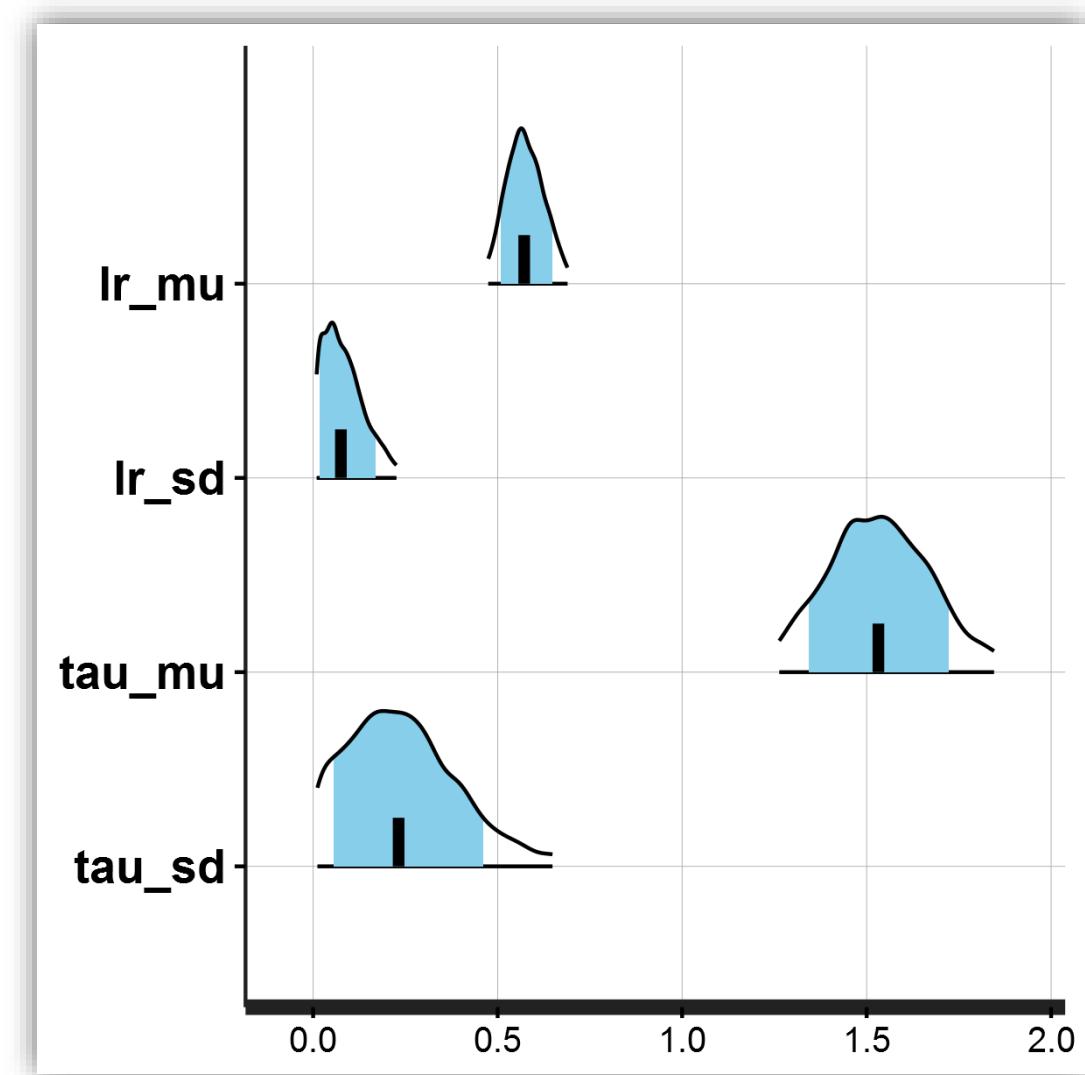
cognitive model
statistics
computing

True group parameters

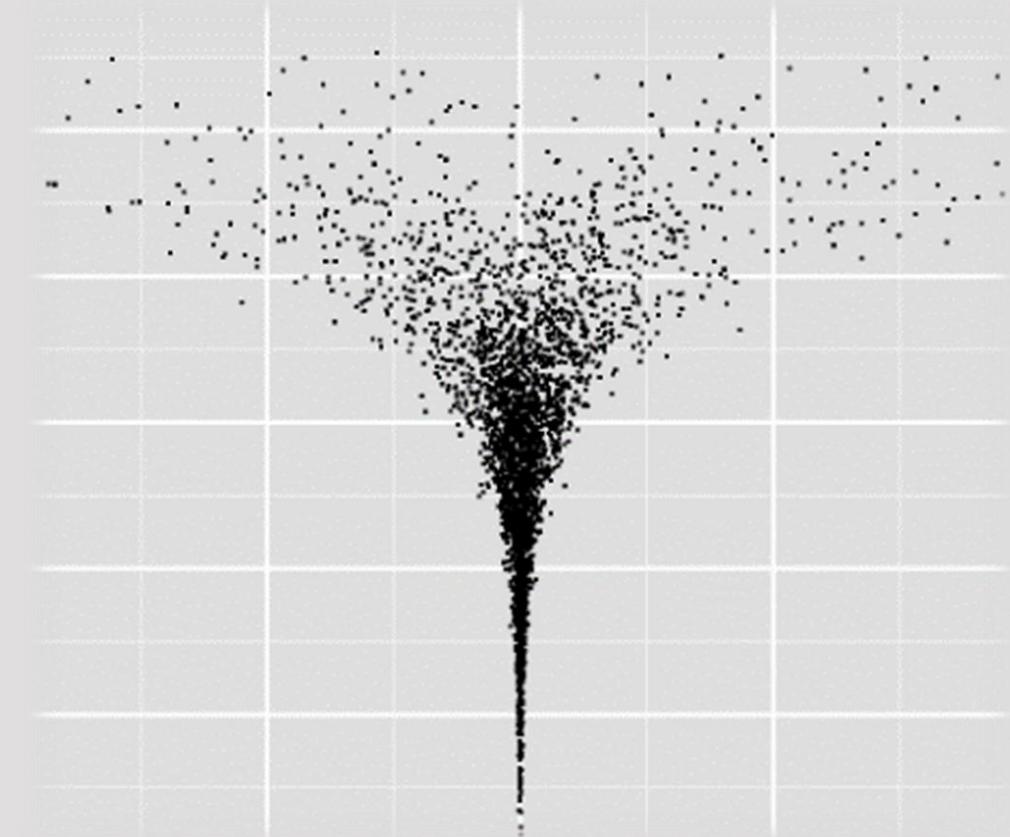
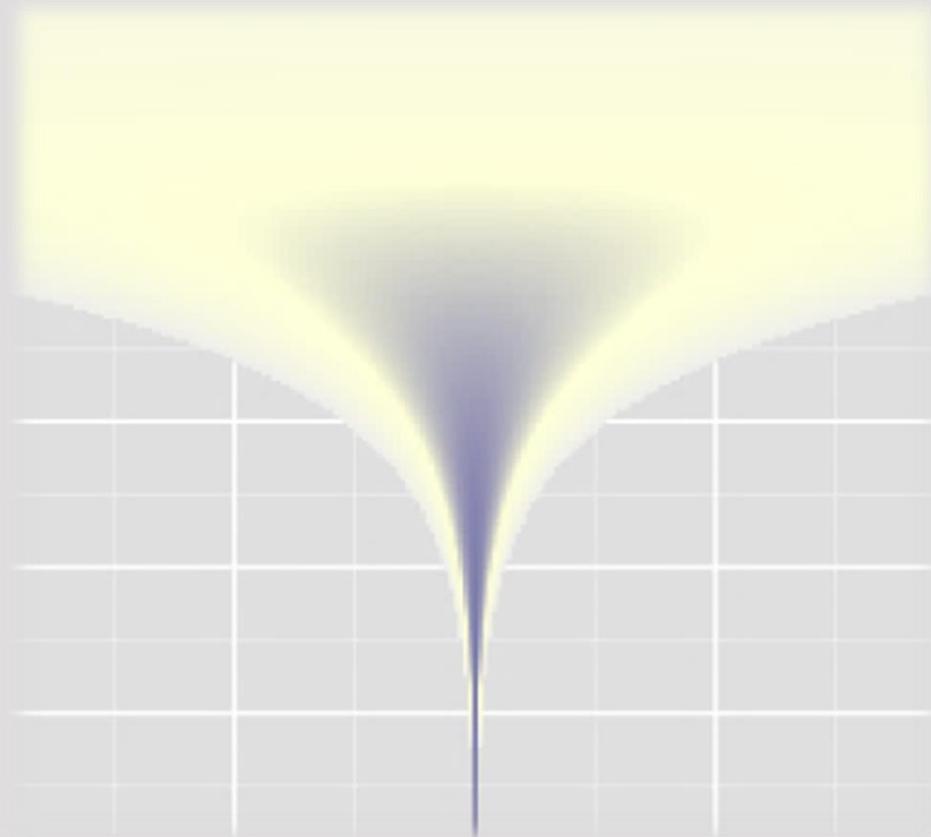
```
lr = rnorm(10, mean=0.6, sd=0.12)  
tau = rnorm(10, mean=1.5, sd=0.2)
```

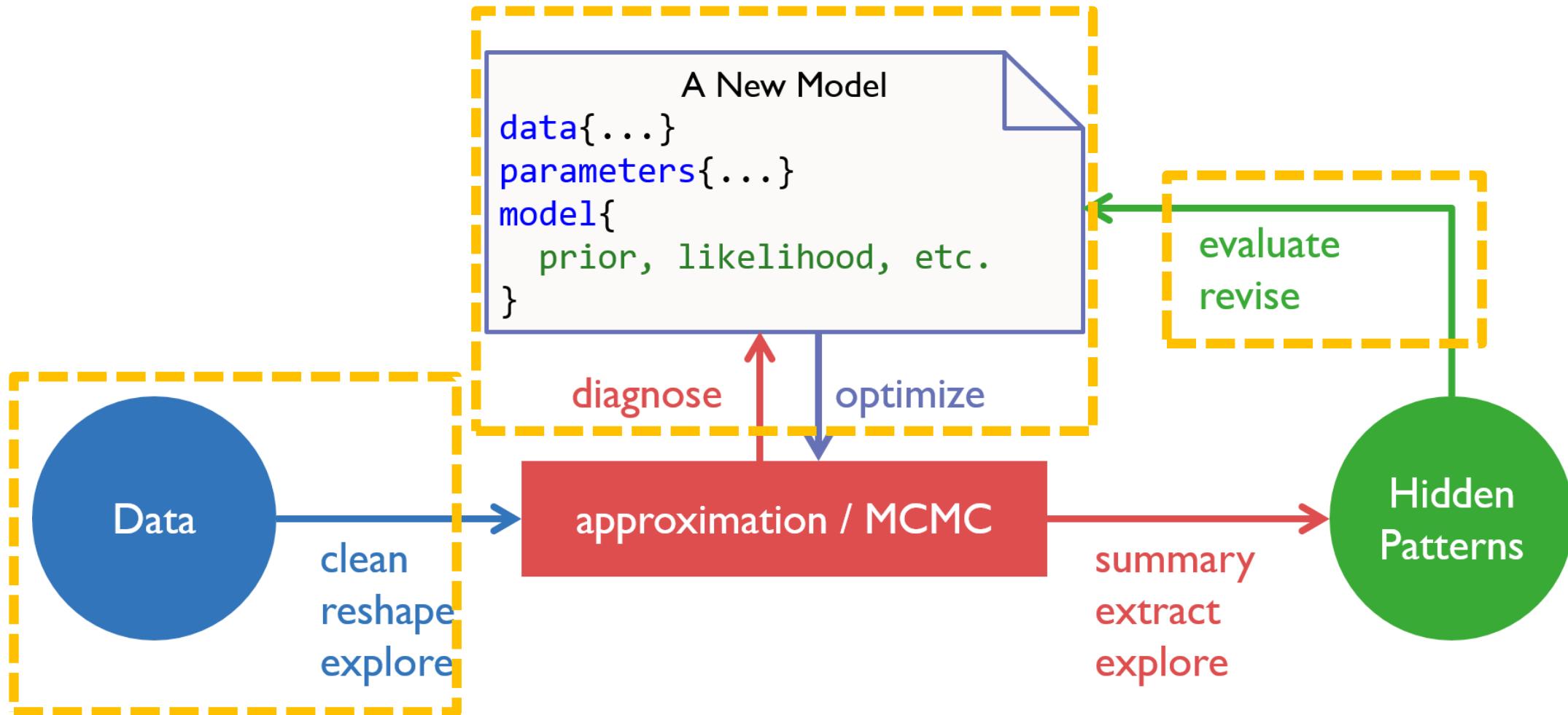
Estimated group parameters

	mean	2.5%	25%	50%	75%	97.5%
lr_mu	0.58	0.47	0.54	0.57	0.61	0.69
lr_sd	0.09	0.01	0.04	0.08	0.12	0.23
tau_mu	1.54	1.26	1.43	1.53	1.63	1.85
tau_sd	0.25	0.01	0.13	0.23	0.34	0.65



OPTIMIZING STAN CODES







Optimizing Stan Code

cognitive model
statistics
computing

Preprocess data

run as many calculations as you can outside Stan

Specify a proper model

follow literature, supervision, experience, etc.

Vectorizing

vectorize Stan code whenever you can

Reparameterizing

reparameterize target parameter to simple distributions

Preprocess Data

cognitive model
statistics
computing

$$\overline{\text{height}} = \alpha + \beta_1 * \text{weight} + \beta_2 * \text{weight}^2$$

```
d$weight_sq <- d$weight^2
```

```
data {
  int<lower=0> N;
  vector<lower=0>[N] height;
  vector<lower=0>[N] weight;
  vector<lower=0>[N] weight_sq;
}
```

Specify a Proper Model

cognitive model
statistics
computing

- Visualize your data
- Follow Literatures
- Start from simple and then build complexities
- Simulate data and run model recovery

A New Model

```
data{...}  
parameters{...}  
model{  
    prior, likelihood, etc.  
}
```

Vectorization

cognitive model
statistics
computing

```
model {  
    for (n in 1:N) {  
        flip[n] ~ bernoulli(theta);  
    }  
}
```

```
model {  
    flip ~ bernoulli(theta);  
}
```

```
parameters {  
    ...  
    real<lower=0,upper=1> lr[nSubjects];  
    real<lower=0,upper=3> tau[nSubjects];  
}  
  
model {  
    ...  
    lr      ~ normal(lr_mu, lr_sd) ;  
    tau    ~ normal(tau_mu, tau_sd) ;  
    ...  
}
```

```
model {  
    vector[N] mu;  
    for (i in 1:N) {  
        mu[i] = alpha + beta * weight[i];  
        height[i] ~ normal(mu[i], sigma)  
    }  
}
```

```
model {  
    vector[N] mu;  
    mu = alpha + beta * weight;  
    height ~ normal(mu, sigma);  
}
```

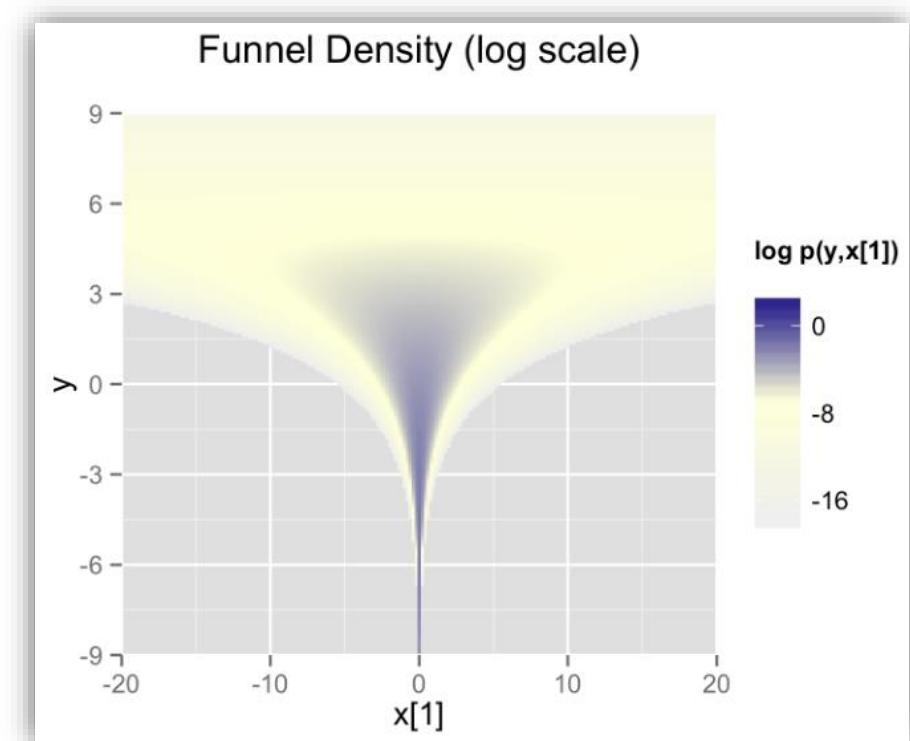
```
model {  
    height ~ normal(alpha + beta * weight, sigma);  
}
```

Reparameterization

Neal's Funnel

$$p(y, \mathbf{x}) = \text{Normal}(y|0, 3) \times \prod_{n=1}^9 \text{Normal}(x_n|0, \exp(y/2))$$

```
parameters {
  real y;
  vector[9] x;
}
model {
  y ~ normal(0,3);
  x ~ normal(0,exp(y/2));
}
```



Non-centered Reparameterization*

cognitive model
statistics
computing

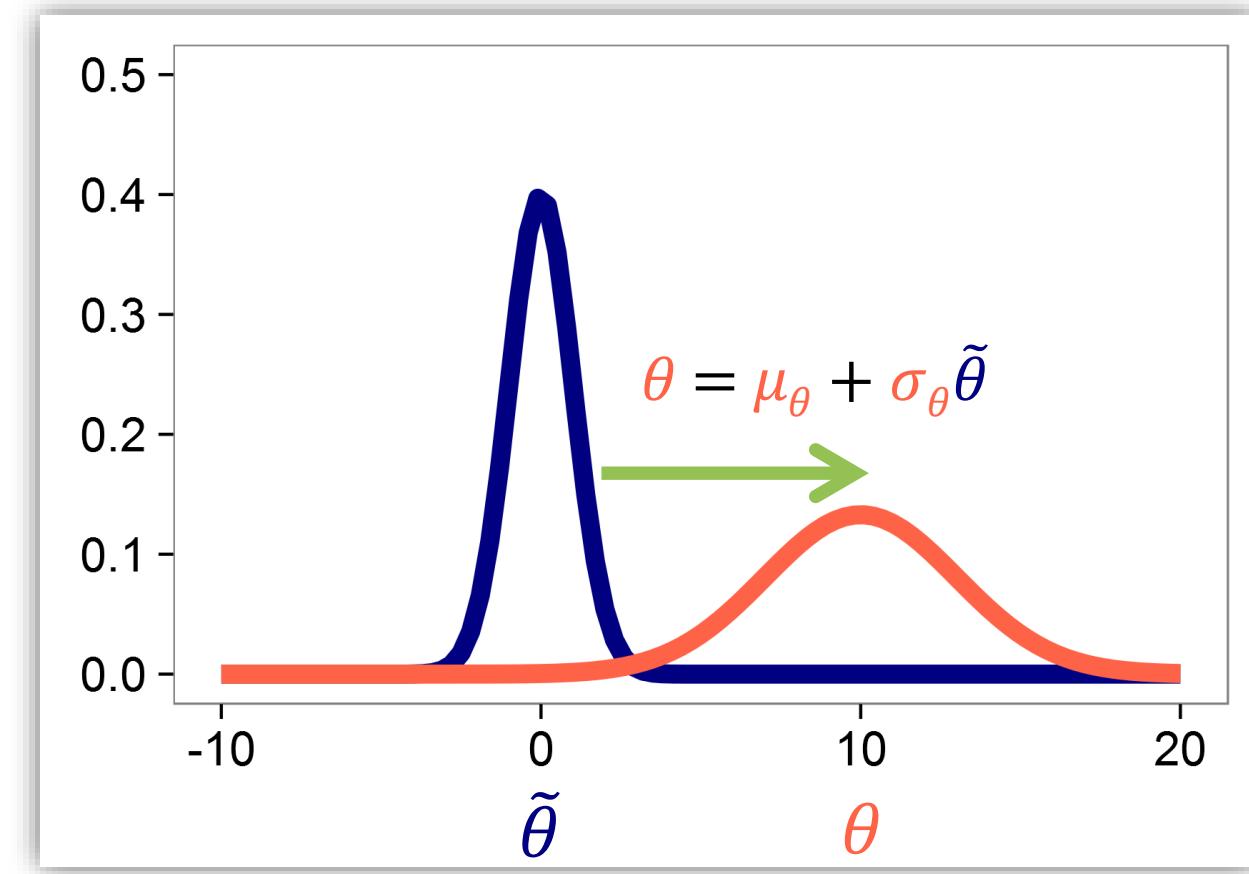
$$\theta \sim Normal(\mu_\theta, \sigma_\theta)$$



$$\tilde{\theta} \sim Normal(0, 1)$$

$$\theta = \mu_\theta + \sigma_\theta \tilde{\theta}$$

Stan likes **simple** distributions!

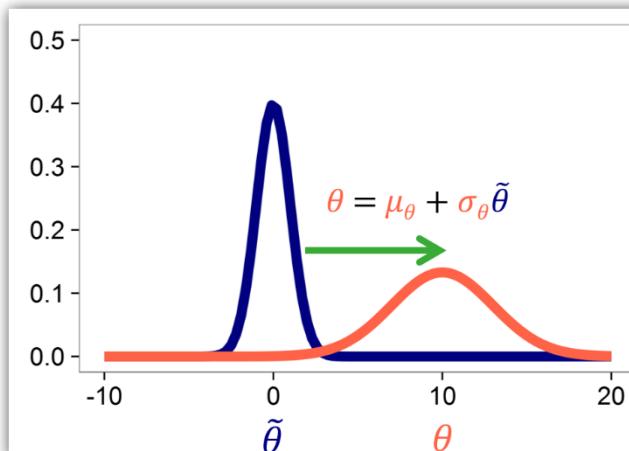


Reparameterization

Neal's Funnel

$$p(y, x) = \text{Normal}(y|0, 3) \times \prod_{n=1}^9 \text{Normal}(x_n|0, \exp(y/2))$$

```
parameters {
  real y;
  vector[9] x;
}
model {
  y ~ normal(0,3);
  x ~ normal(0,exp(y/2));
}
```



```
parameters {
  real y_raw;
  vector[9] x_raw;
}
transformed parameters {
  real y;
  vector[9] x;
}

y = 3.0 * y_raw;
x = exp(y/2) * x_raw;

model {
  y_raw ~ normal(0,1);
  x_raw ~ normal(0,1);
}
```

Stan Sampling Parameters

cognitive model
statistics
computing

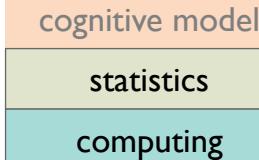
parameter	description	constraint	default
iterations	number of MCMC samples (per chain)	int, > 0	2000
delta: δ	target Metropolis acceptance rate	$\delta \in [0, 1]$	0.80
stepsize: ε	initial HMC step size	real, $\varepsilon > 0$	2.0
max_treedepth: L	maximum HMC steps per iteration	int, $L > 0$	10

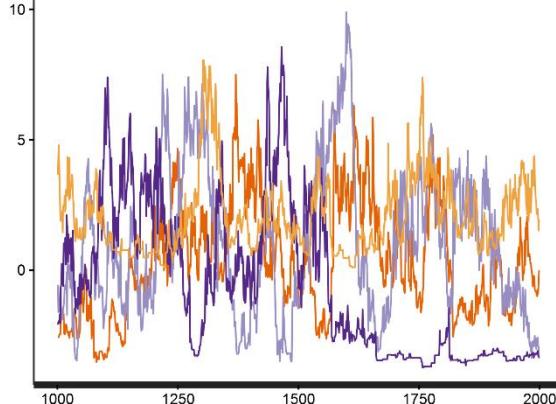
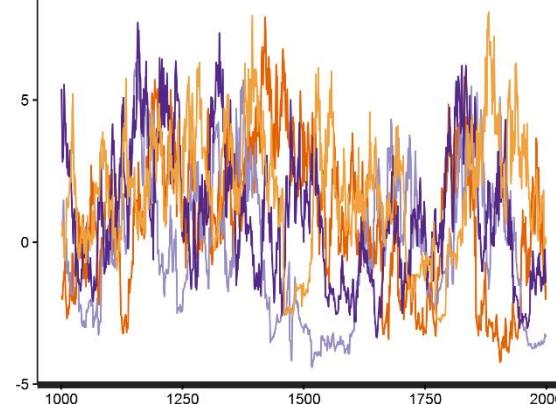
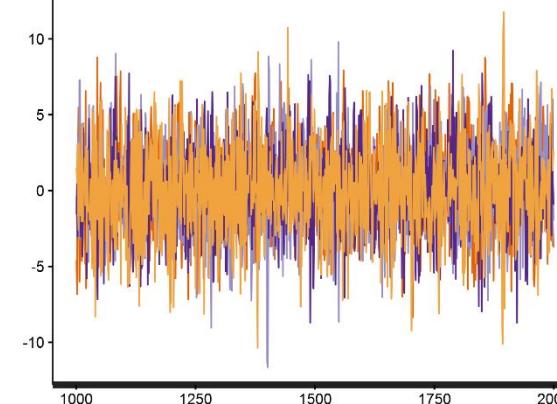
Typical adjustments

- Increase iterations
- Increase delta
- Decrease stepsize
- Might have to increase max_treedepth

```
funnel_fit2 <- stan("_scripts/funnel.stan",
  iter = 4000,
  control = list(adapt_delta = 0.999,
                 stepsize = 1.0,
                 max_treedepth = 20))
```

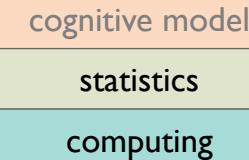
Neal's Funnel: Comparing Performance



	direct model	adjusted direct model	reparameterized model
Rhat (y)	1.22	1.1	1.0
n_eff (y)	18	42	3886
runtime*	48.50 sec	50.76 sec	50.12 sec
n_eff (y) / runtime	0.37 / sec	0.82 / sec	77.53 / sec
n_divergent	53	0	0
traceplot (y)			

*: 2 cores in parallel, including compiling time

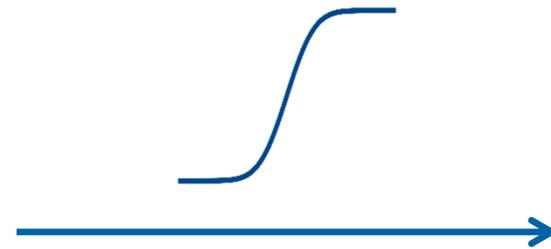
How about Bounded Parameters?



$$\tilde{\theta} \sim \text{Normal}(0, 1)$$

$$\theta = \mu_\theta + \sigma_\theta \tilde{\theta}$$

$$\theta \in (-\infty, +\infty)$$



$$\tilde{\theta} \sim \text{Normal}(0, 1)$$

$$\theta = \text{Probit}^{-1}(\mu_\theta + \sigma_\theta \tilde{\theta})$$

$$\theta \in [0, 1]$$

constraint	reparameterization
$\theta \in (-\infty, +\infty)$	$\theta = \mu_\theta + \sigma_\theta \tilde{\theta}$
$\theta \in [0, N]$	$\theta = \text{Probit}^{-1}(\mu_\theta + \sigma_\theta \tilde{\theta}) \times N$
$\theta \in [M, N]$	$\theta = \text{Probit}^{-1}(\mu_\theta + \sigma_\theta \tilde{\theta}) \times (N-M) + M$
$\theta \in (0, +\infty)$	$\theta = \exp(\mu_\theta + \sigma_\theta \tilde{\theta}); \theta = \log(1+\exp(\mu_\theta + \sigma_\theta \tilde{\theta}))$

* Probit⁻¹: Normal cumulative distribution function (normcdf)

Apply to Our Hierarchical RL Model

cognitive model
statistics
computing

```
parameters {  
    real<lower=0,upper=1> lr_mu;  
    real<lower=0,upper=3> tau_mu;  
  
    real<lower=0> lr_sd;  
    real<lower=0> tau_sd;  
  
    real<lower=0,upper=1> lr[nSubjects];  
    real<lower=0,upper=3> tau[nSubjects];  
}
```



```
parameters {  
    # group-Level parameters  
    real lr_mu_raw;  
    real tau_mu_raw;  
    real<lower=0> lr_sd_raw;  
    real<lower=0> tau_sd_raw;  
  
    # subject-Level raw parameters  
    vector[nSubjects] lr_raw;  
    vector[nSubjects] tau_raw;  
}  
  
transformed parameters {  
    vector<lower=0,upper=1>[nSubjects] lr;  
    vector<lower=0,upper=3>[nSubjects] tau;  
  
    for (s in 1:nSubjects) {  
        lr[s] = Phi_approx( lr_mu_raw + lr_sd_raw * lr_raw[s] );  
        tau[s] = Phi_approx( tau_mu_raw + tau_sd_raw * tau_raw[s] ) * 3;  
    }  
}
```

Apply to Our Hierarchical RL Model

cognitive model
statistics
computing

```
model {  
    lr_sd ~ cauchy(0,1);  
    tau_sd ~ cauchy(0,3);  
    lr ~ normal(lr_mu, lr_sd) ;  
    tau ~ normal(tau_mu, tau_sd) ;  
  
    for (s in 1:nSubjects) {  
        vector[2] v;  
        real pe;  
        v = initV;  
  
        for (t in 1:nTrials) {  
            choice[s,t] ~ categorical_logit( tau[s] * v );  
            pe = reward[s,t] - v[choice[s,t]];  
            v[choice[s,t]] = v[choice[s,t]] + lr[s] * pe;  
        }  
    }  
}
```



```
model {  
    lr_mu_raw ~ normal(0,1);  
    tau_mu_raw ~ normal(0,1);  
    lr_sd_raw ~ cauchy(0,3);  
    tau_sd_raw ~ cauchy(0,3);  
  
    lr_raw ~ normal(0,1);  
    tau_raw ~ normal(0,1);  
  
    for (s in 1:nSubjects) {  
        ...  
  
generated quantities {  
    real<lower=0,upper=1> lr_mu;  
    real<lower=0,upper=3> tau_mu;  
  
    lr_mu = Phi_approx(lr_mu_raw);  
    tau_mu = Phi_approx(tau_mu_raw) * 3;  
}
```

Exercise XII

cognitive model
statistics
computing

```
.../07.optm_rl/_scripts/reinforcement_learning_hrch_main.R
```

TASK: (1) Complete the Matt Trick

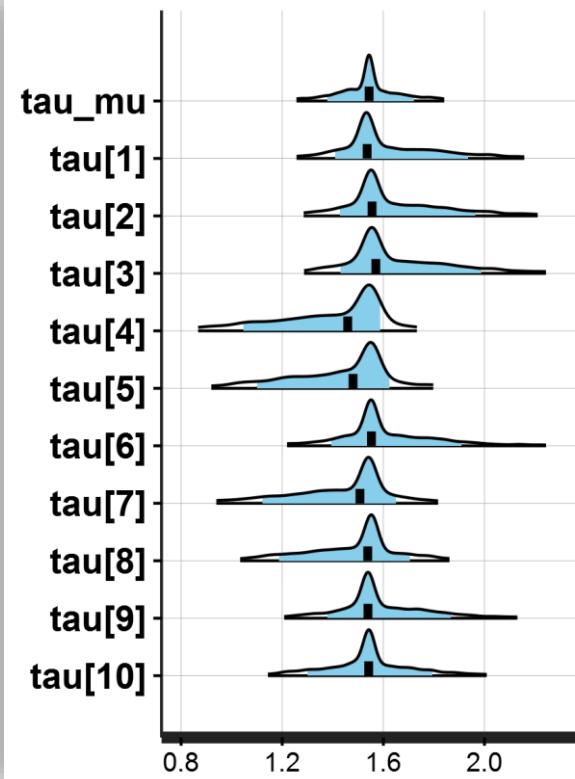
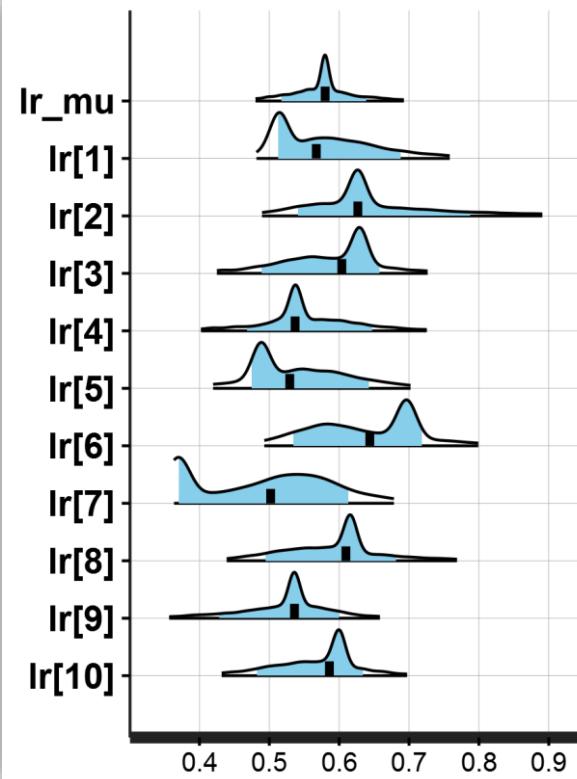
(2) fit the optimized hierarchical RL model

```
> source('_scripts/reinforcement_learning_hrch_main.R')  
  
> fit_rl4 <- run_rl_mp2(optimized = TRUE)
```

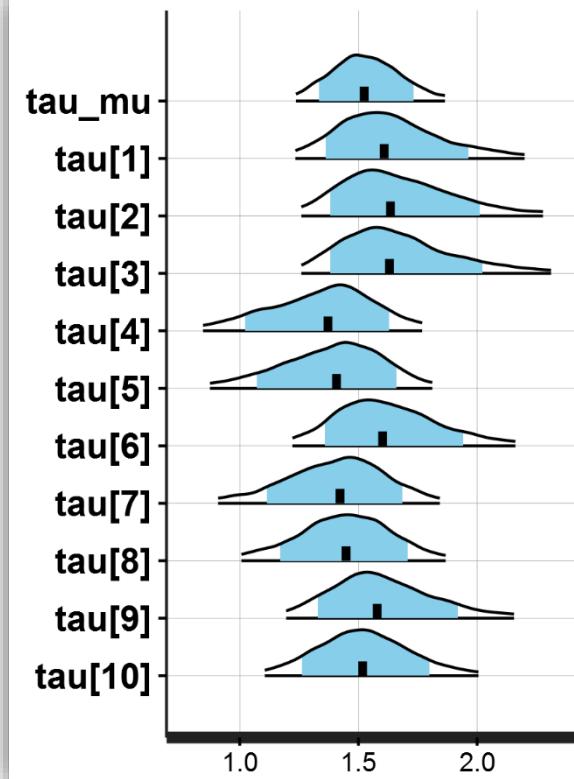
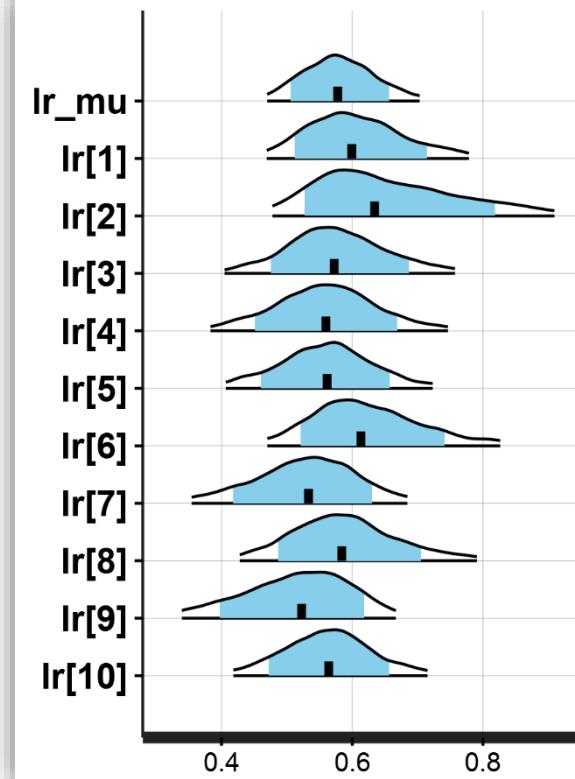
Hierarchical Fitting – Optimized

cognitive model
statistics
computing

Posterior Means (hrch)



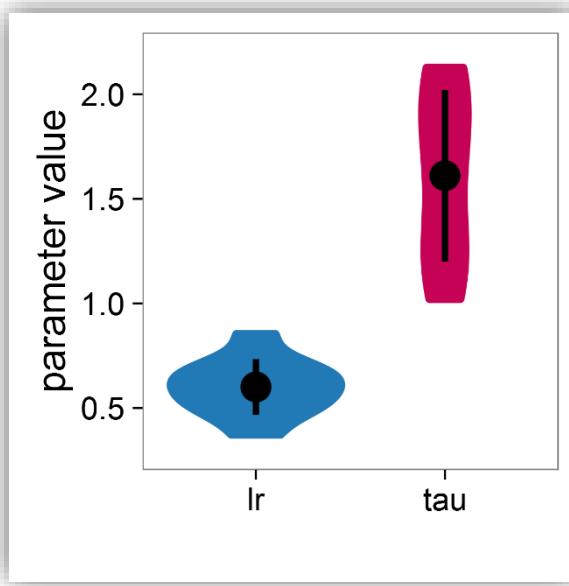
Posterior Means (hrch + optim)



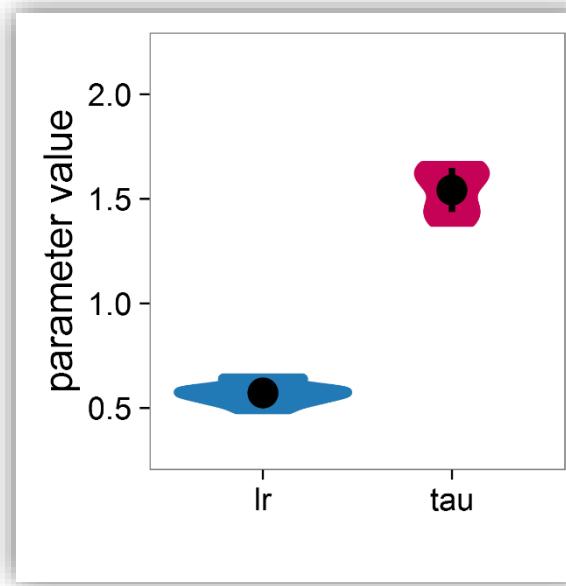
Comparing with True Parameters

cognitive model
statistics
computing

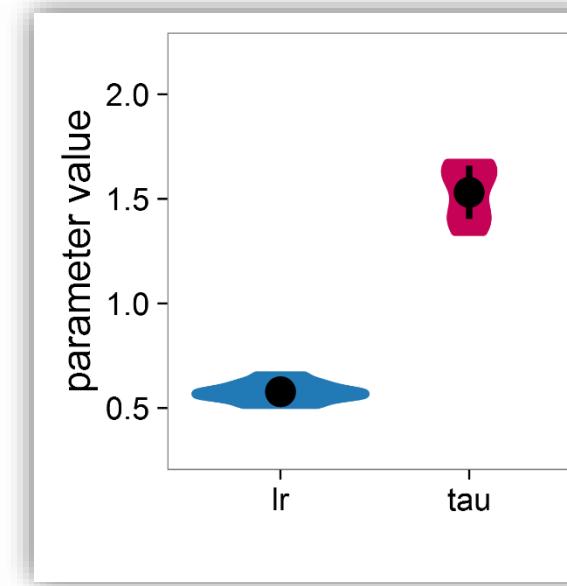
Posterior Means (indv)



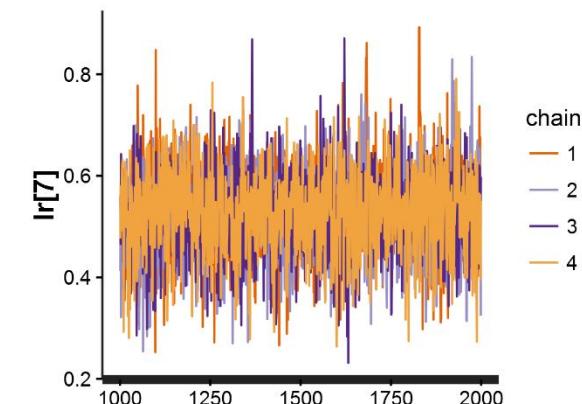
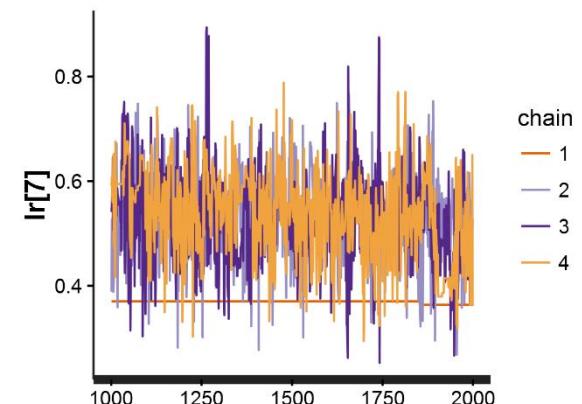
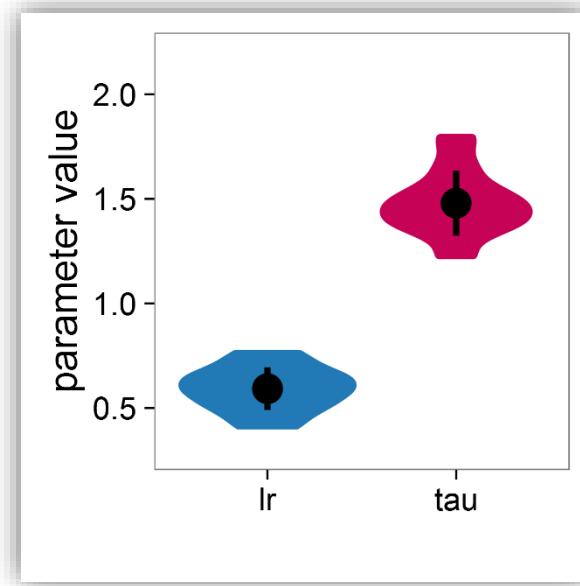
Posterior Means (hrch)



Posterior Means (hrch+optm)

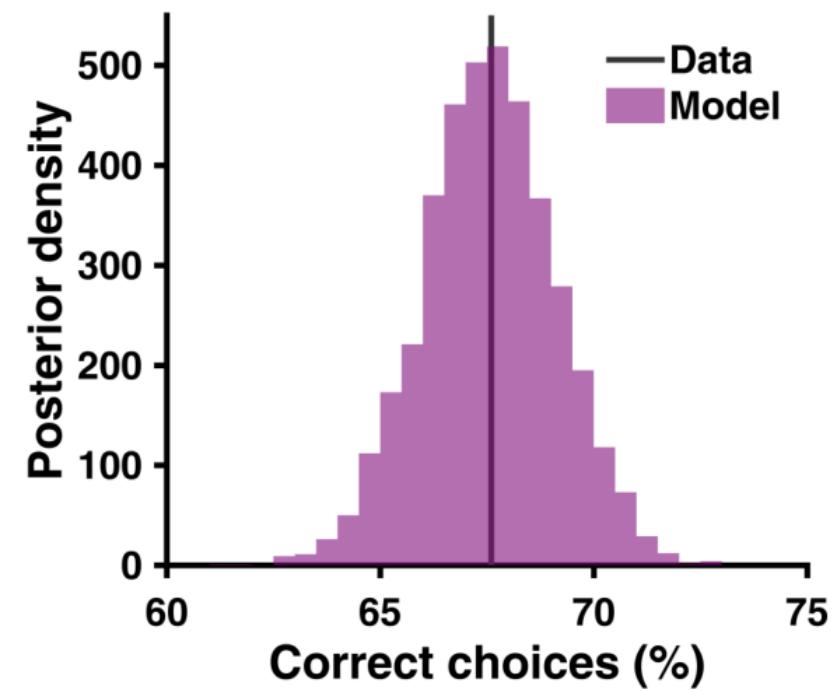
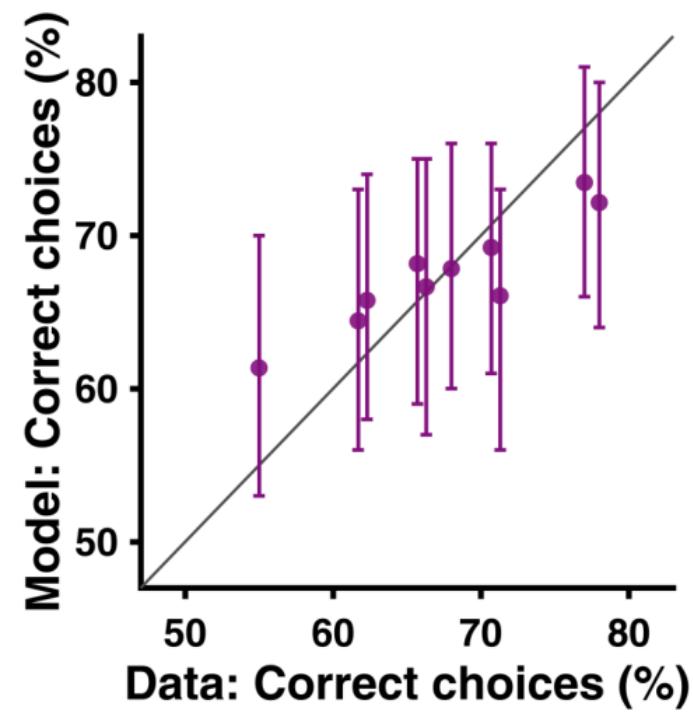
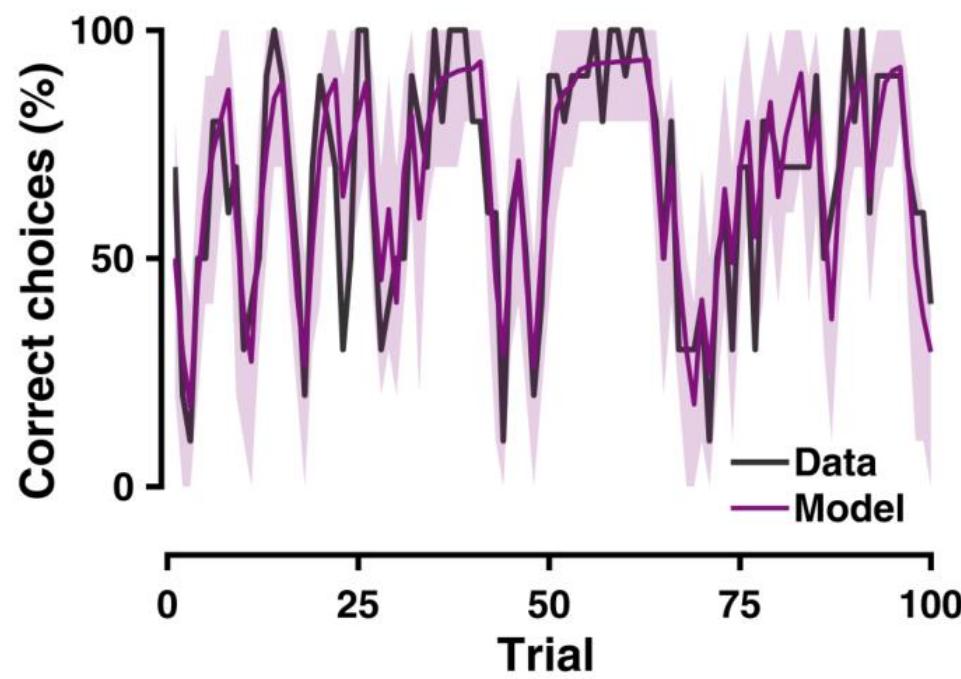


True Parameters

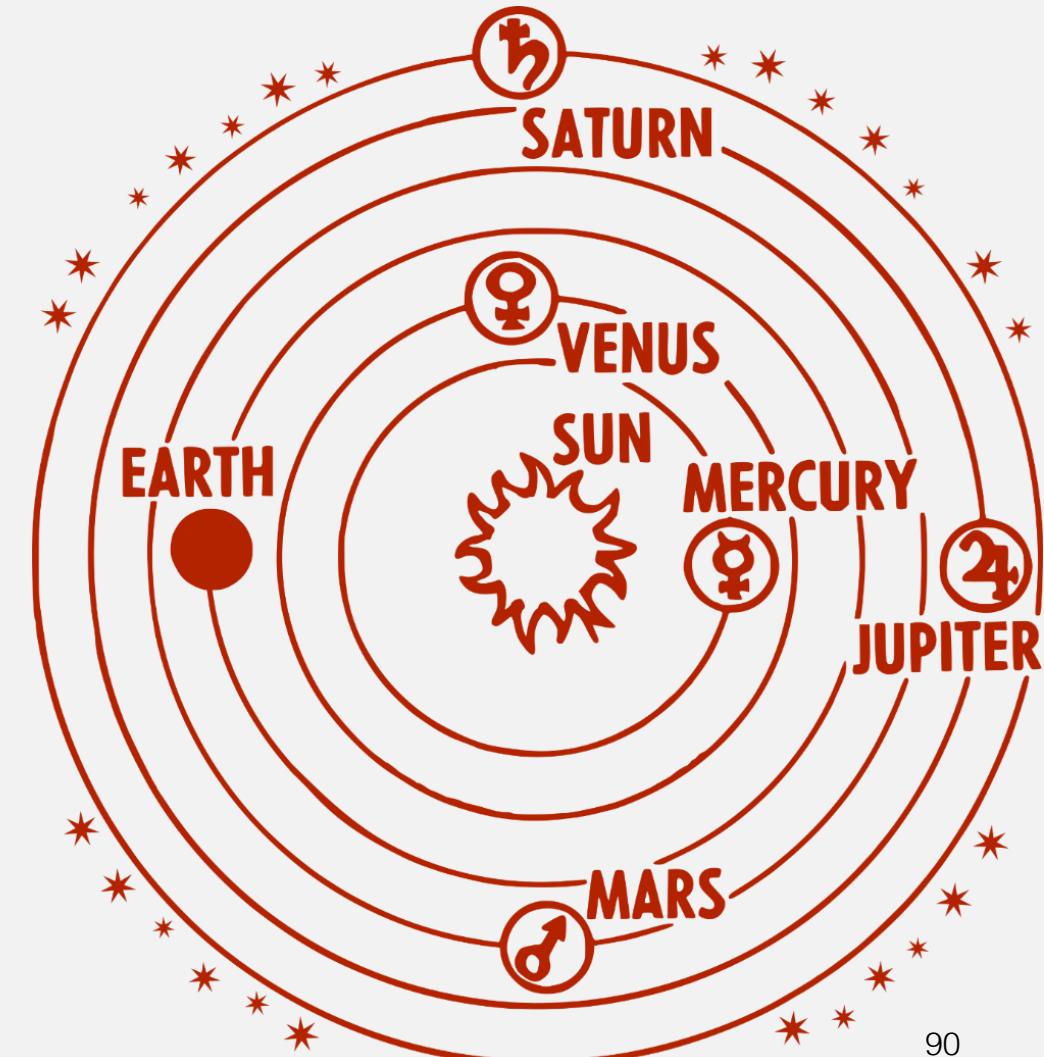
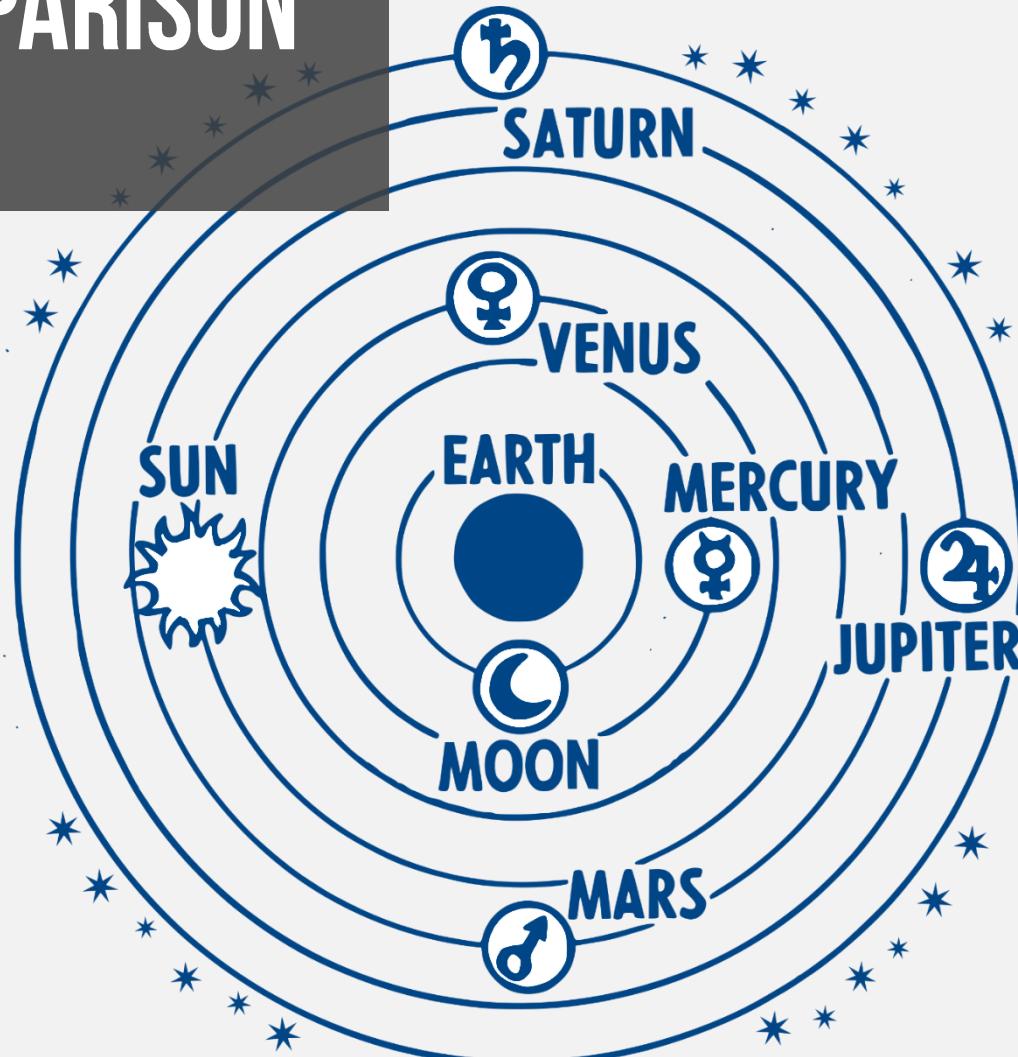


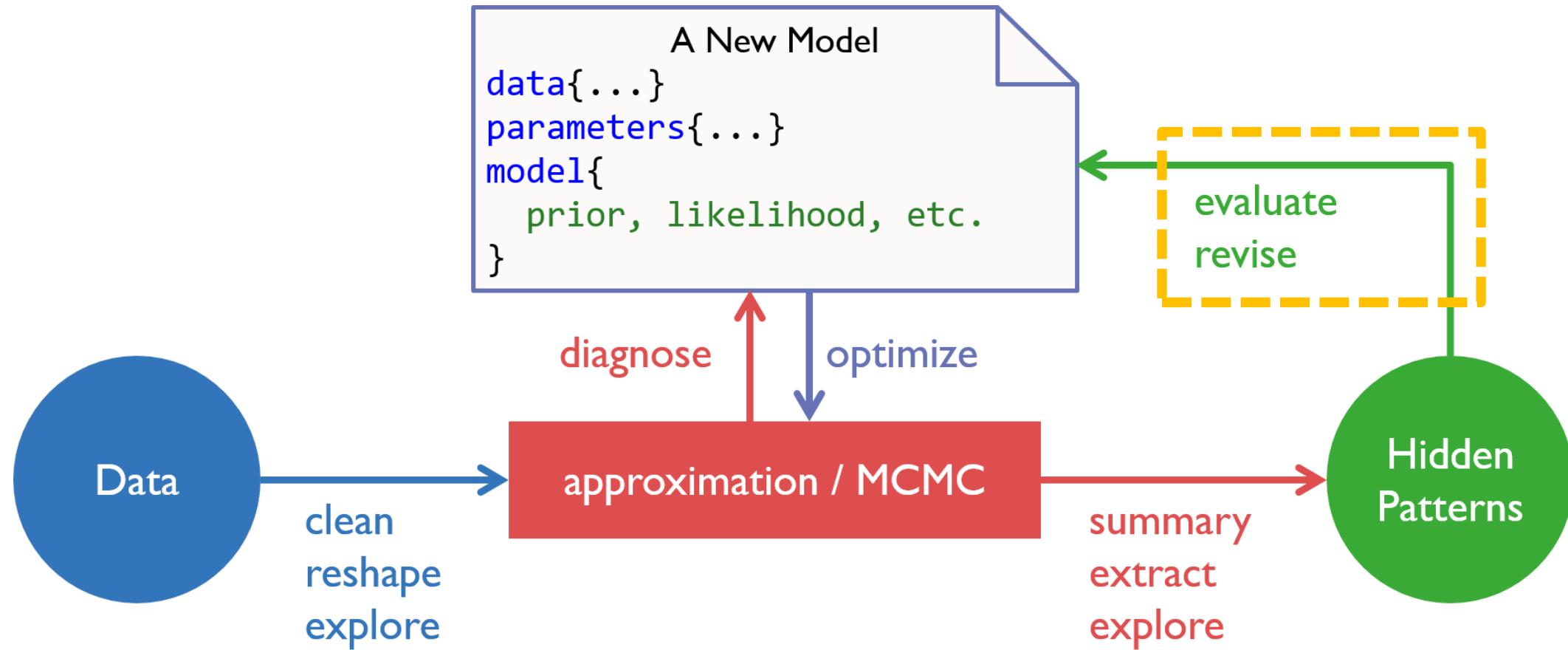
Posterior Predictive Check

cognitive model
statistics
computing



MODEL COMPARISON





Model Comparison

cognitive model
statistics
computing

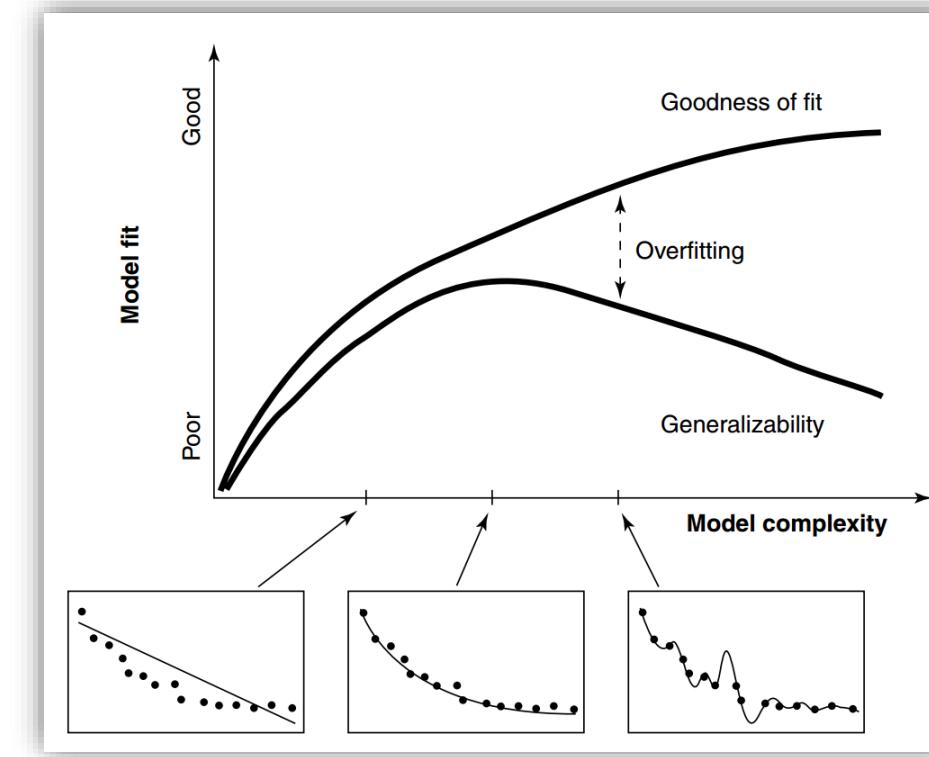
Which model provides the best **fit**?



Which model represents the best **balance** between model fit and model complexity?

Ockham's razor:

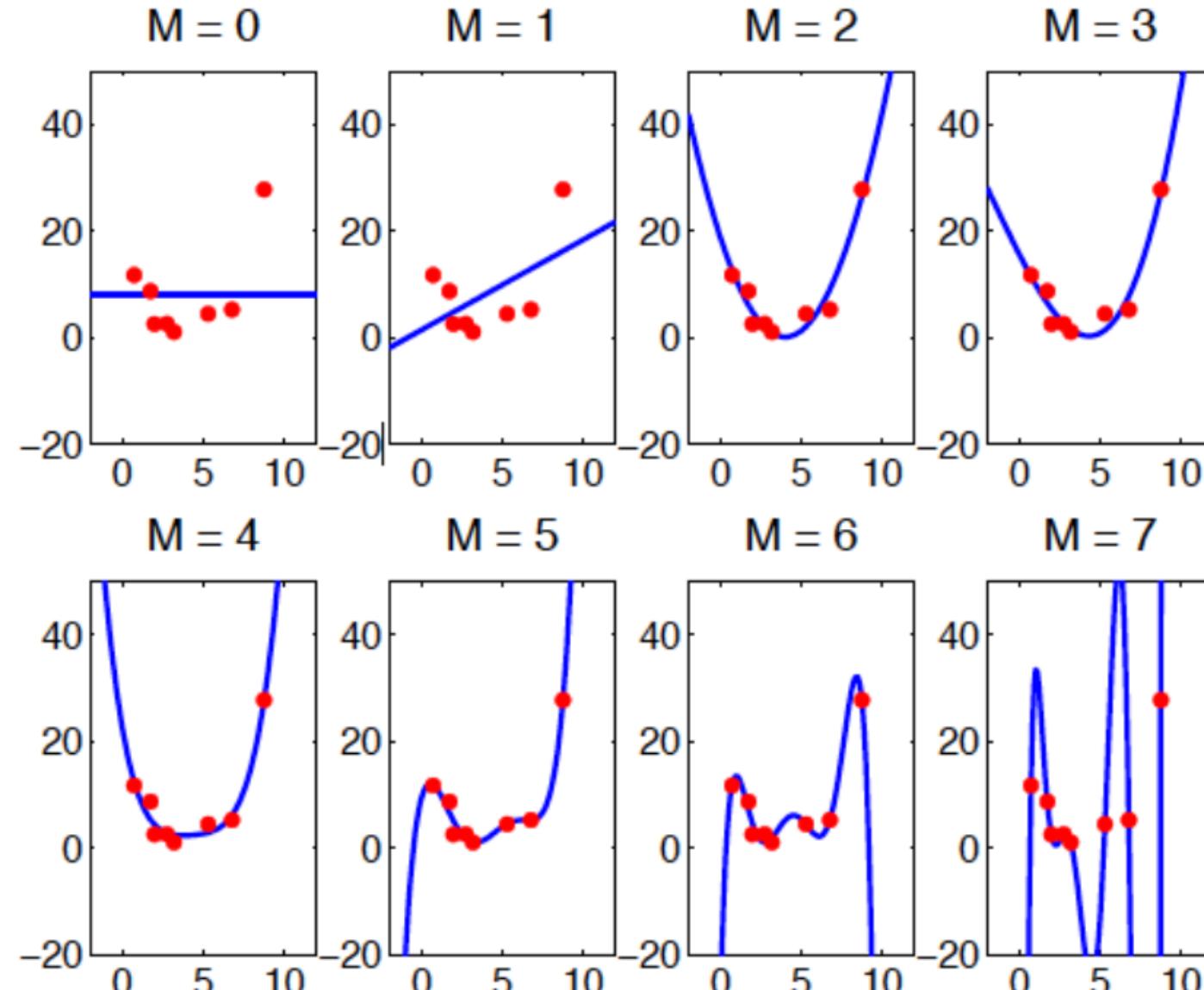
Models with fewer assumptions are to be preferred



- overfitting: learn **too much** from the data
- underfitting: learn **too little** from the data

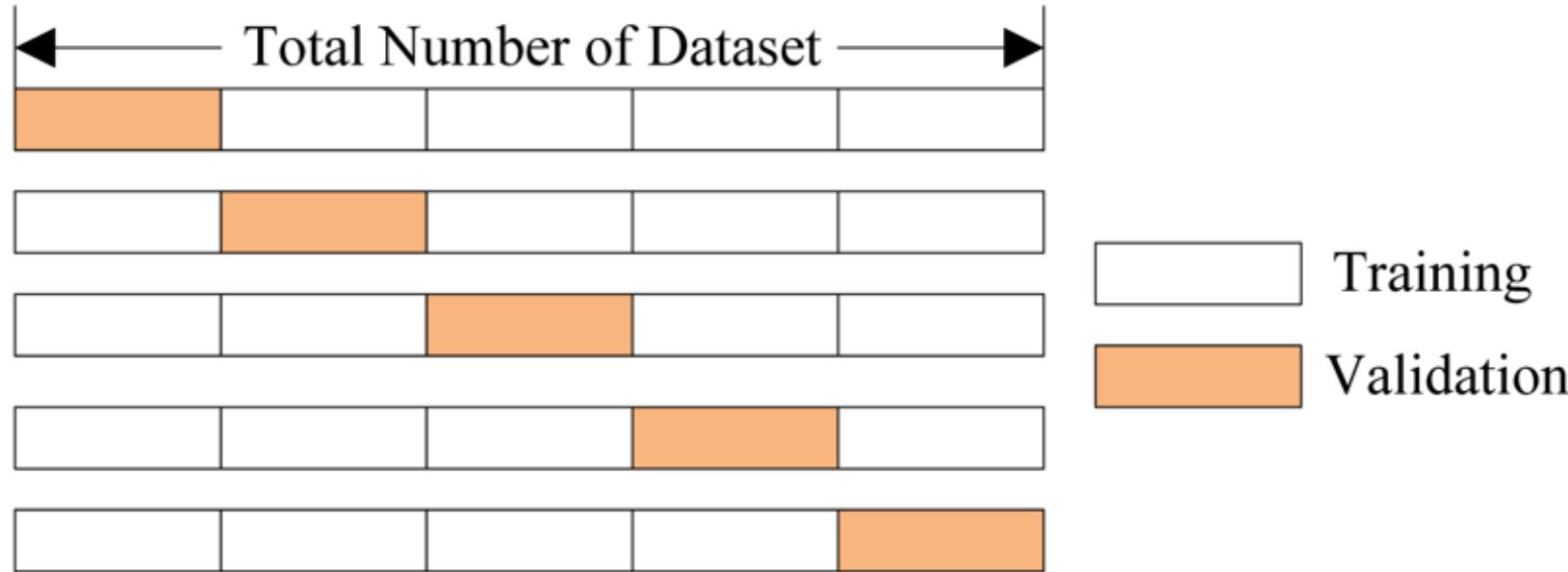
Which model has the highest predictive power?

cognitive model
statistics
computing



Focusing on Predictive Accuracy

cognitive model
statistics
computing



- Nothing prevents you from doing that in a Bayesian context but holding out data makes your posterior distribution more diffuse
- Bayesians usually condition on *all* the data and evaluate how well a model is expected to **predict out of sample** using "information criteria": model with the **highest expected log predictive density (ELPD)** for new data

Information Criteria

cognitive model
statistics
computing

AIC – Akaike information criterion

DIC – Deviance Information Criterion

WAIC – Widely Applicable Information Criterion

finding the model that has the highest out-of-sample predictive accuracy

BIC – Bayesian Information Criterion

approximation to LOO

finding the “true” model

Compute WAIC from Likelihood

cognitive model
statistics
computing

$$\text{WAIC} = -2 \widehat{\text{elpd}}_{\text{waic}}$$

expected log pointwise predictive density

$$\widehat{\text{elpd}}_{\text{waic}} = \widehat{\text{lpd}} - \widehat{p}_{\text{waic}}$$

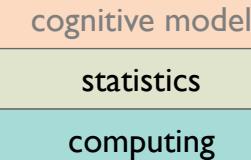
$$\begin{aligned}\widehat{\text{lpd}} &= \text{computed log pointwise predictive density} \\ &= \sum_{i=1}^n \log \left(\frac{1}{S} \sum_{s=1}^S p(y_i | \theta^s) \right).\end{aligned}$$

$$\begin{aligned}\widehat{p}_{\text{waic}} &= \text{estimated effective number of parameters} \\ &= \sum_{i=1}^n V_{s=1}^S (\log p(y_i | \theta^s))\end{aligned}$$

```
lpd <- log(colMeans(exp(log_lik)))
```

```
p_waic <- colVars(log_lik)
```

*IC comparisons



		No pooling $(\tau = \infty)$	Complete pooling $(\tau = 0)$	Hierarchical model $(\tau \text{ estimated})$
AIC	$-2 \text{lpd} = -2 \log p(y \hat{\theta}_{\text{mle}})$	54.6	59.4	
	k	8.0	1.0	
	$\text{AIC} = -2 \widehat{\text{elpd}}_{\text{AIC}}$	70.6	61.4	
DIC	$-2 \text{lpd} = -2 \log p(y \hat{\theta}_{\text{Bayes}})$	54.6	59.4	57.4
	p_{DIC}	8.0	1.0	2.8
	$\text{DIC} = -2 \widehat{\text{elpd}}_{\text{DIC}}$	70.6	61.4	63.0
WAIC	$-2 \text{lppd} = -2 \sum_i \log p_{\text{post}}(y_i)$	60.2	59.8	59.2
	$p_{\text{WAIC } 1}$	2.5	0.6	1.0
	$p_{\text{WAIC } 2}$	4.0	0.7	1.3
	$\text{WAIC} = -2 \widehat{\text{elppd}}_{\text{WAIC } 2}$	68.2	61.2	61.8
LOO-CV	-2lppd		59.8	59.2
	$p_{\text{loo-cv}}$		0.5	1.8
	$-2 \text{lppd}_{\text{loo-cv}}$		60.8	62.8

Recording the Log-Likelihood in Stan

cognitive model
statistics
computing

```
generated quantities {
  ...
  real log_lik[nSubjects];
  ...

  { # Local section, this saves time and space
    for (s in 1:nSubjects) {
      vector[2] v;
      real pe;

      log_lik[s] = 0;
      v = initV;

      for (t in 1:nTrials) {
        log_lik[s] = log_lik[s] + categorical_logit_lpmf(choice[s,t] | tau[s] * v);

        pe = reward[s,t] - v[choice[s,t]];
        v[choice[s,t]] = v[choice[s,t]] + lr[s] * pe;
      }
    }
  }
}
```

The {loo} Package

cognitive model
statistics
computing

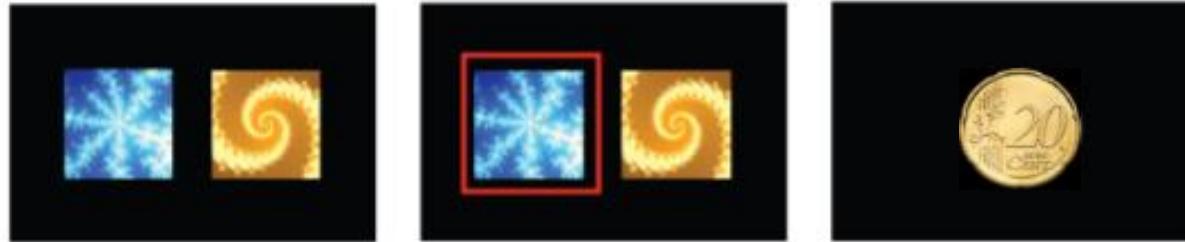
```
> library(loo)
> LL1    <- extract_log_lik(stanfit)
> loo1   <- loo(LL1)    # PSIS leave-one-out
> waic1 <- waic(LL1)   # WAIC
```

Computed from 4000 by 20 log-likelihood matrix

	Estimate	SE
elpd_loo	-29.5	3.3
p_loo	2.7	1.0
looic	58.9	6.7

Pareto Smoothed Importance Sampling

Reversal Learning Task



Fictitious RL
(Counterfactual RL)

Value update:

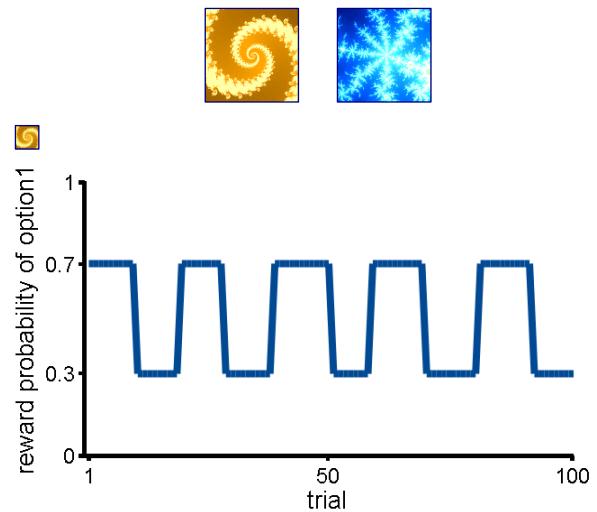
$$V_{t+1}^c = V_t^c + \alpha * PE$$

$$V_{t+1}^{nc} = V_t^{nc} + \alpha * PEnc$$

Prediction error:

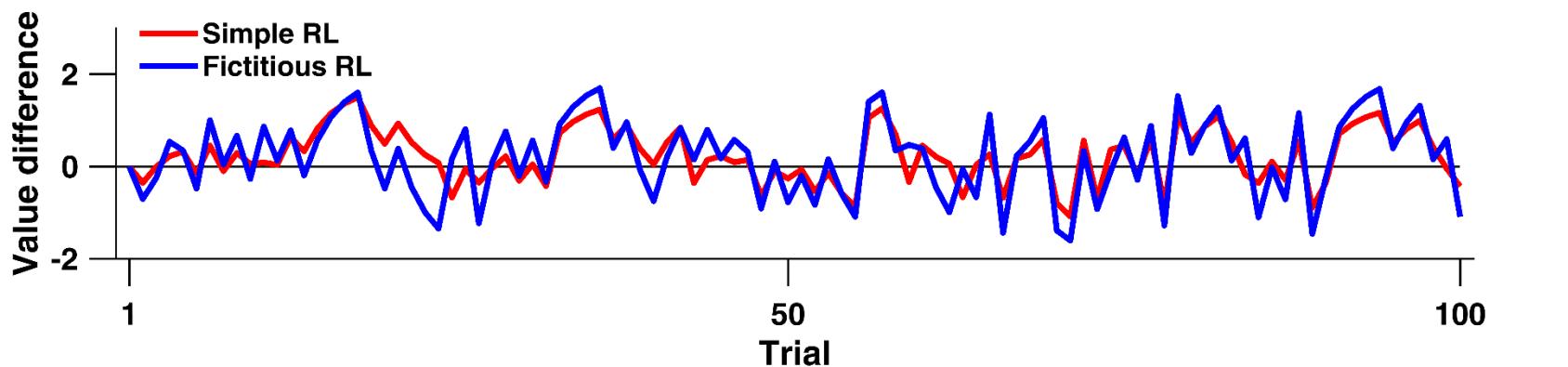
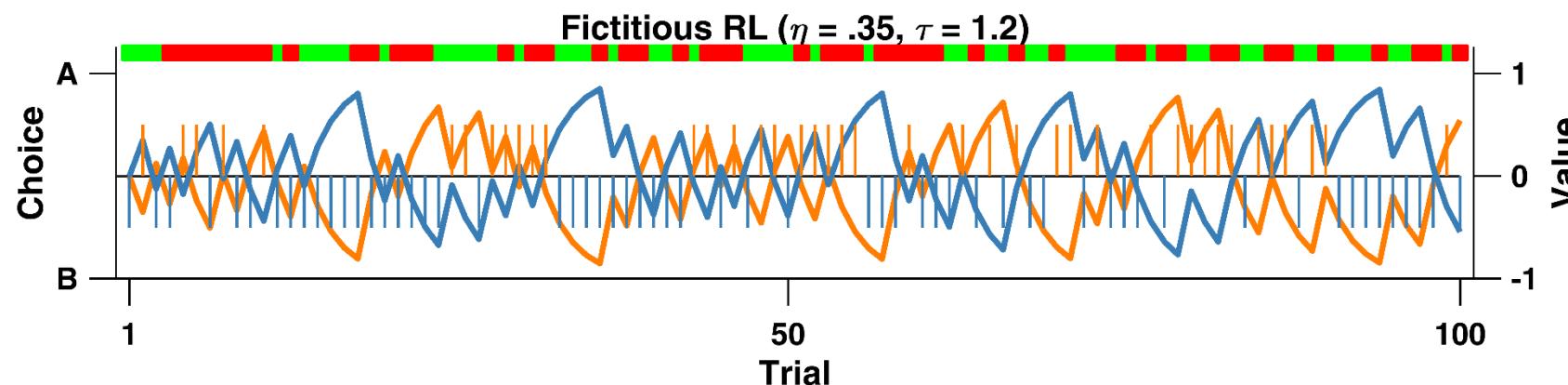
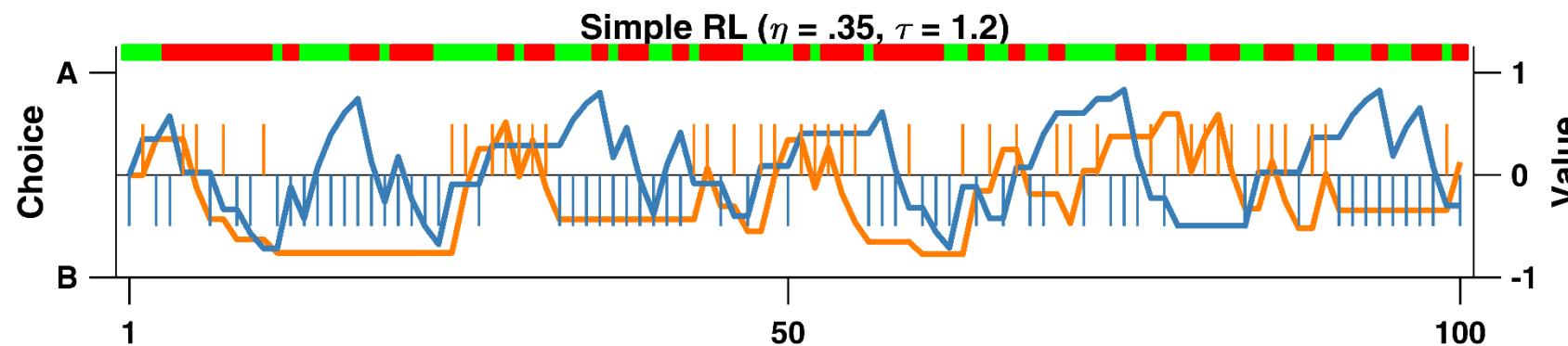
$$PE = R_t - V_t^c$$

$$PEnc = -R_t - V_t^{nc}$$



More on Fictitious RL

cognitive model
statistics
computing



Exercise XIII

cognitive model
statistics
computing

.../08.compare_models/_scripts/compare_models_main.R

TASK: (1) complete the fictitious RL model (model2, loglik)
(2) fit and compare the 2 models

Exercise XIII – output

```
> LL1 <- extract_log_lik(fit_r1)
> ( loo1 <- loo(LL1) )
```

Computed from 4000 by 10 log-likelihood matrix

	Estimate	SE
elpd_loo	-389.8	15.4
p_loo	3.8	0.8
looic	779.5	30.9

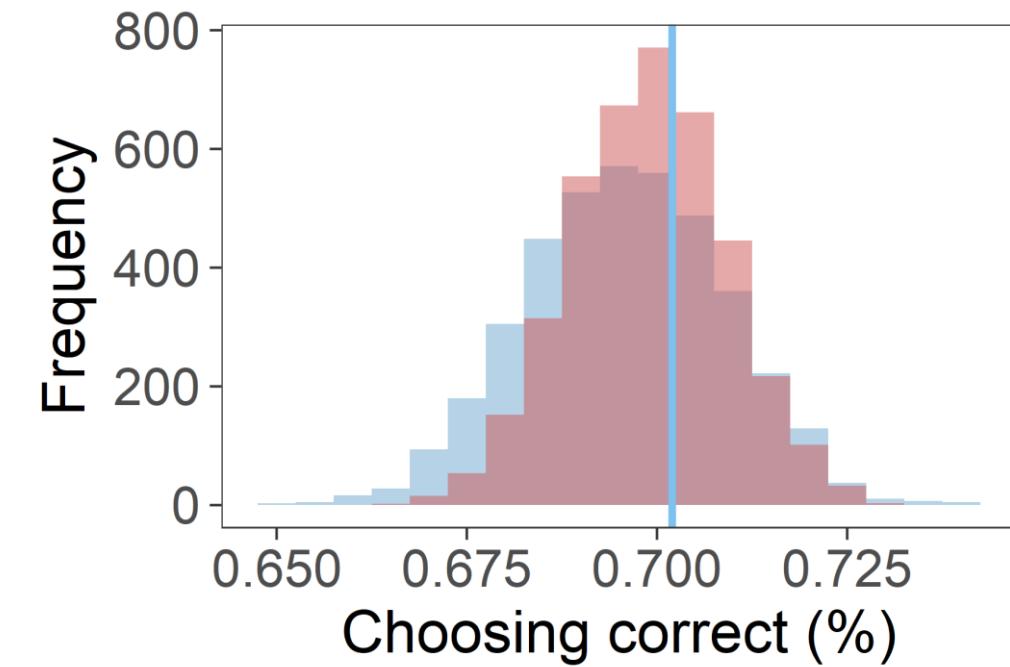
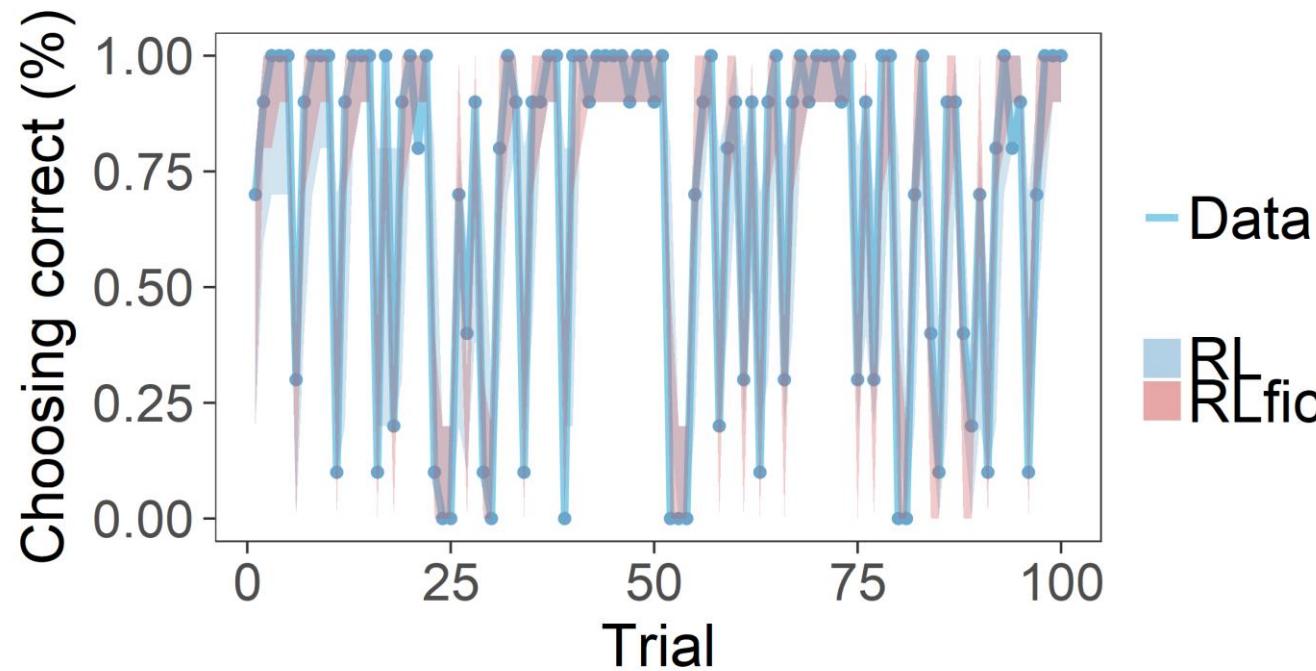
```
> ( loo2 <- loo(LL2) )
```

Computed from 4000 by 10 log-likelihood matrix

	Estimate	SE
elpd_loo	-281.3	17.5
p_loo	3.4	0.5
looic	562.6	35.0

Posterior Predictive Check

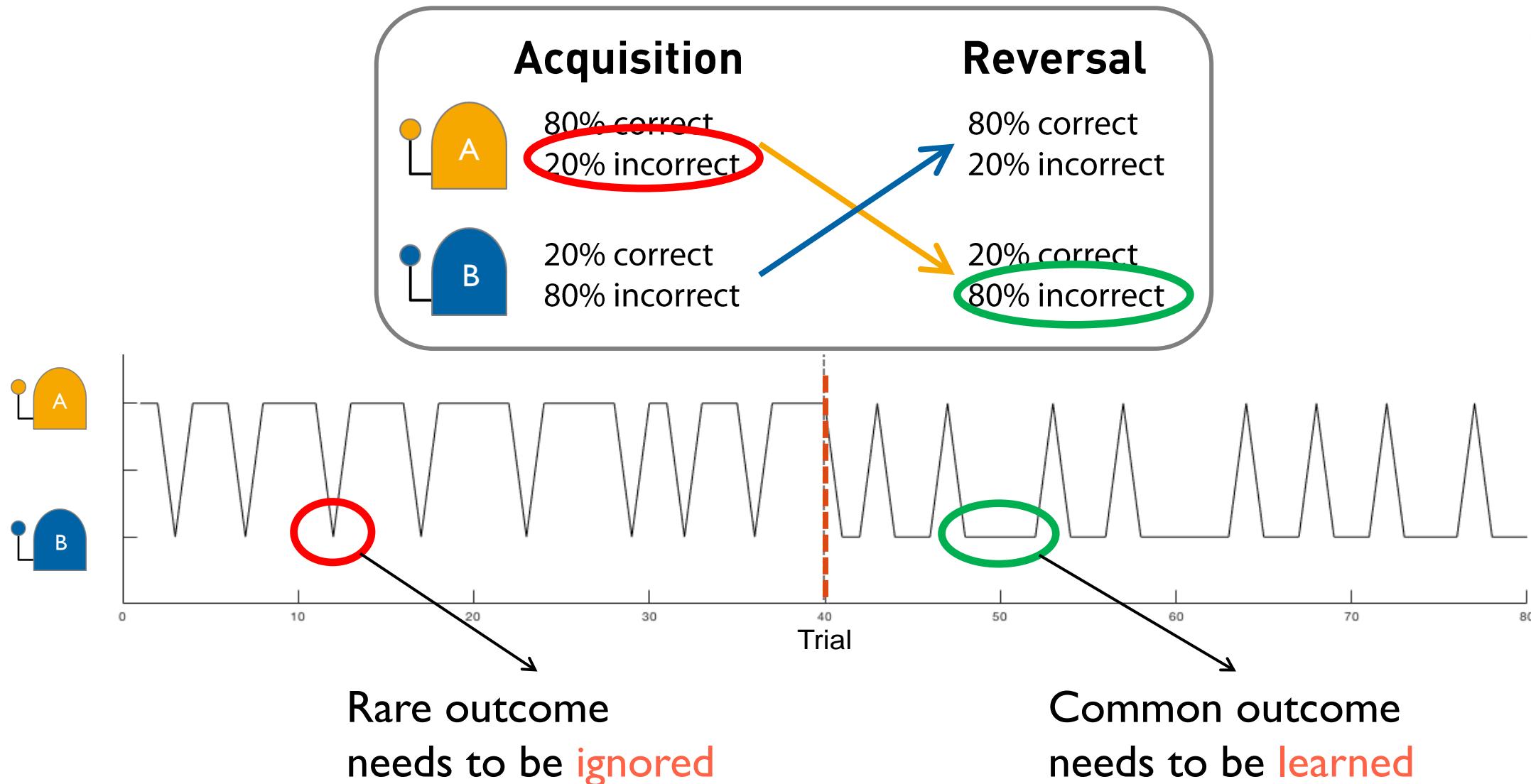
cognitive model
statistics
computing



A PRINCIPLED MODELING WORKFLOW

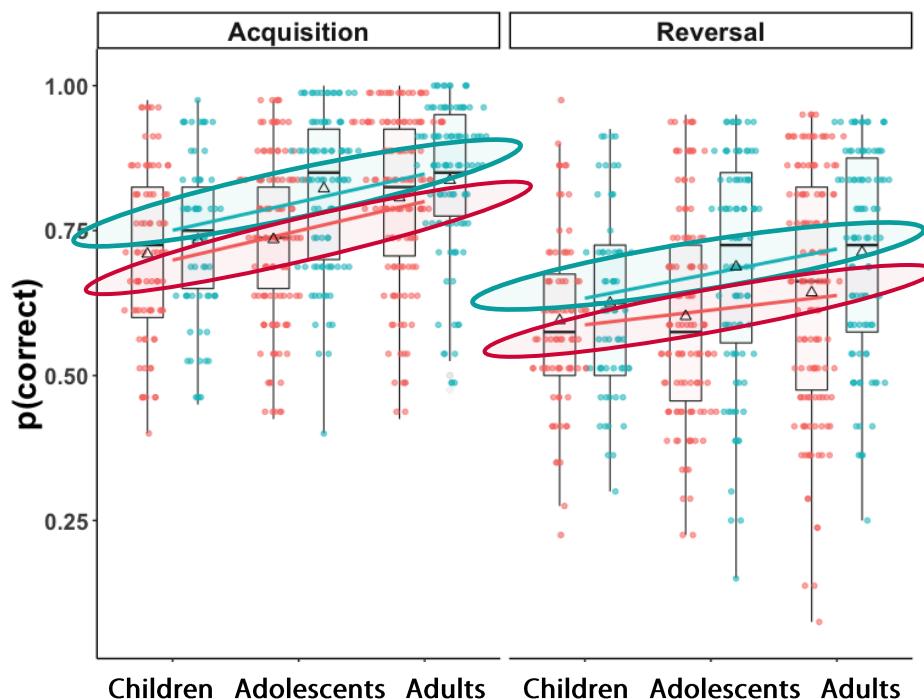


Cognitive flexibility in reversal learning

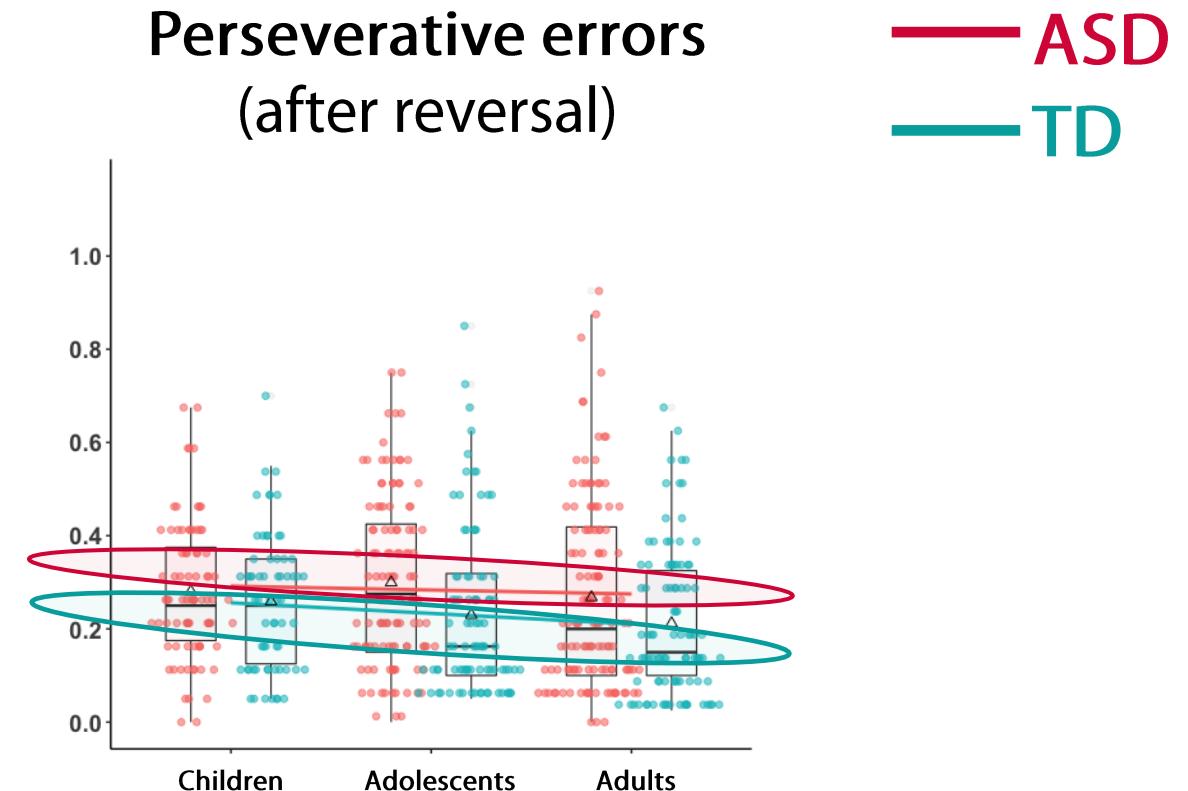


Worse performance in ASD

Choice accuracy

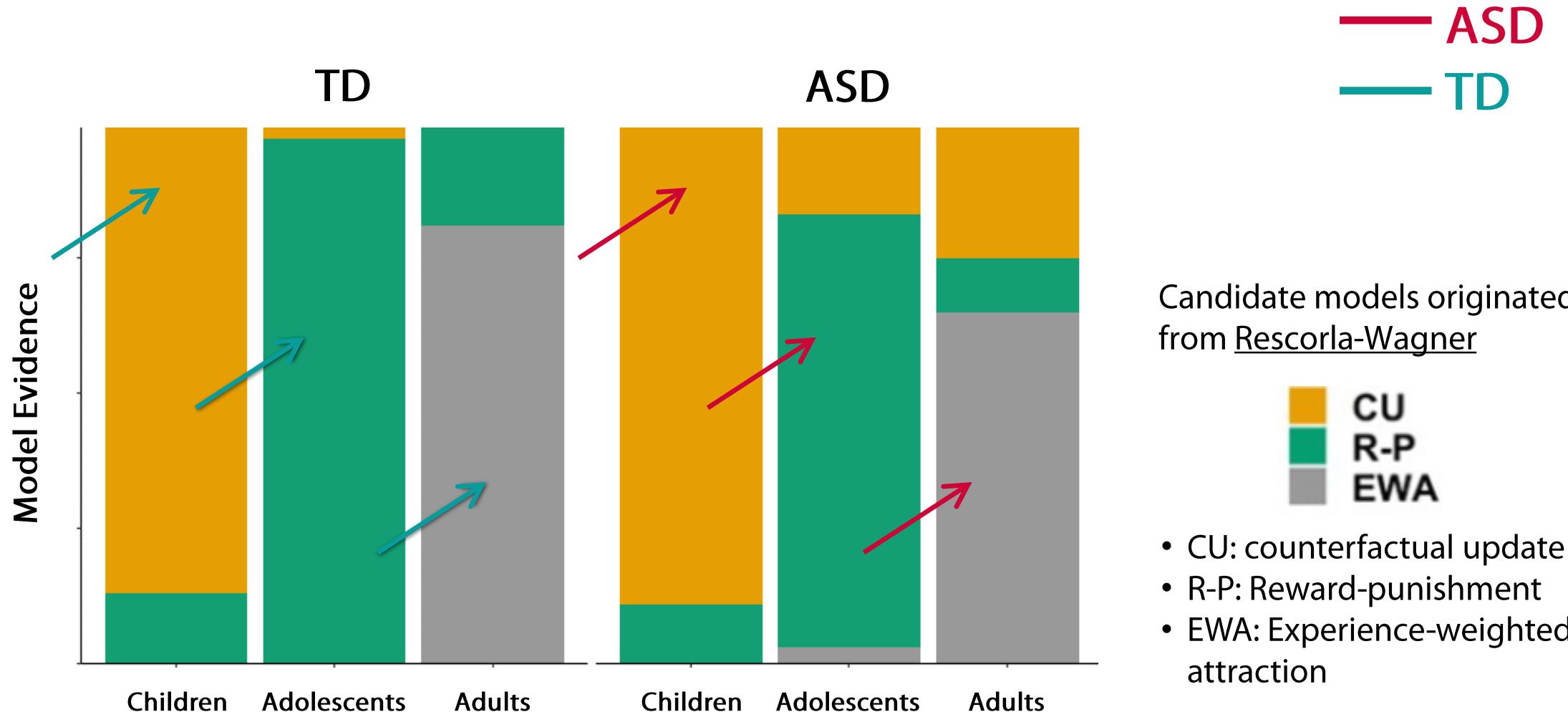


Perseverative errors (after reversal)



— ASD
— TD

Same computational model within age group



Candidate models originated from Rescorla-Wagner

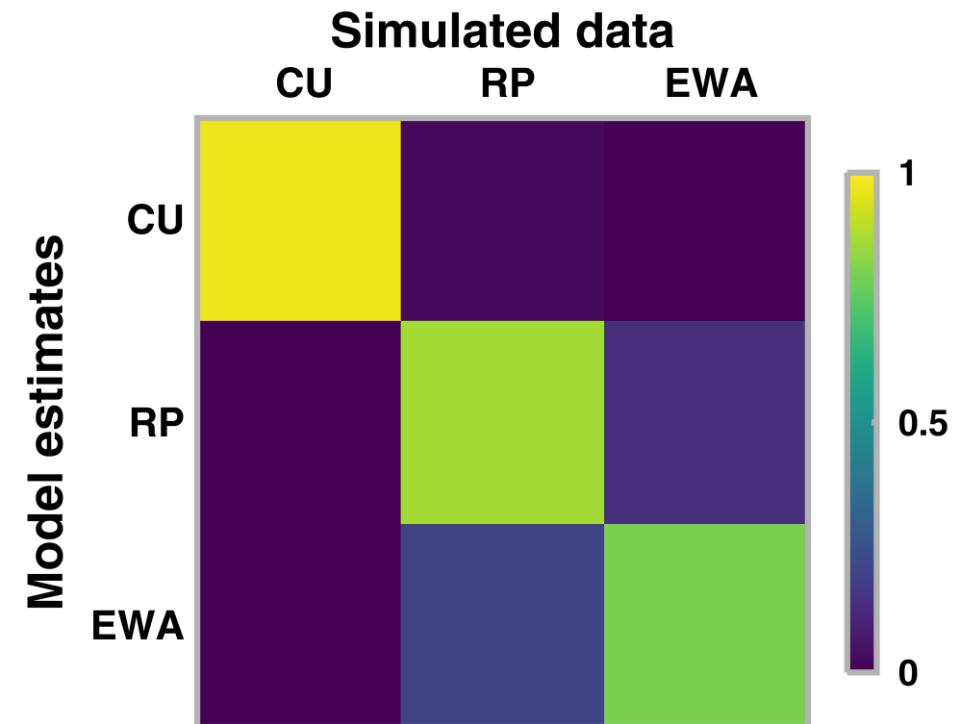
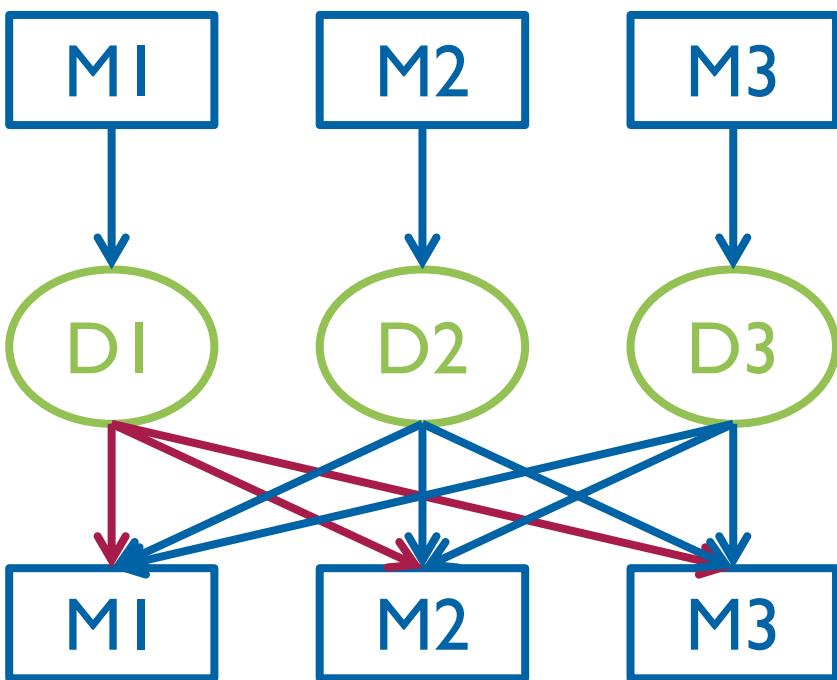


- CU: counterfactual update
- R-P: Reward-punishment
- EWA: Experience-weighted attraction

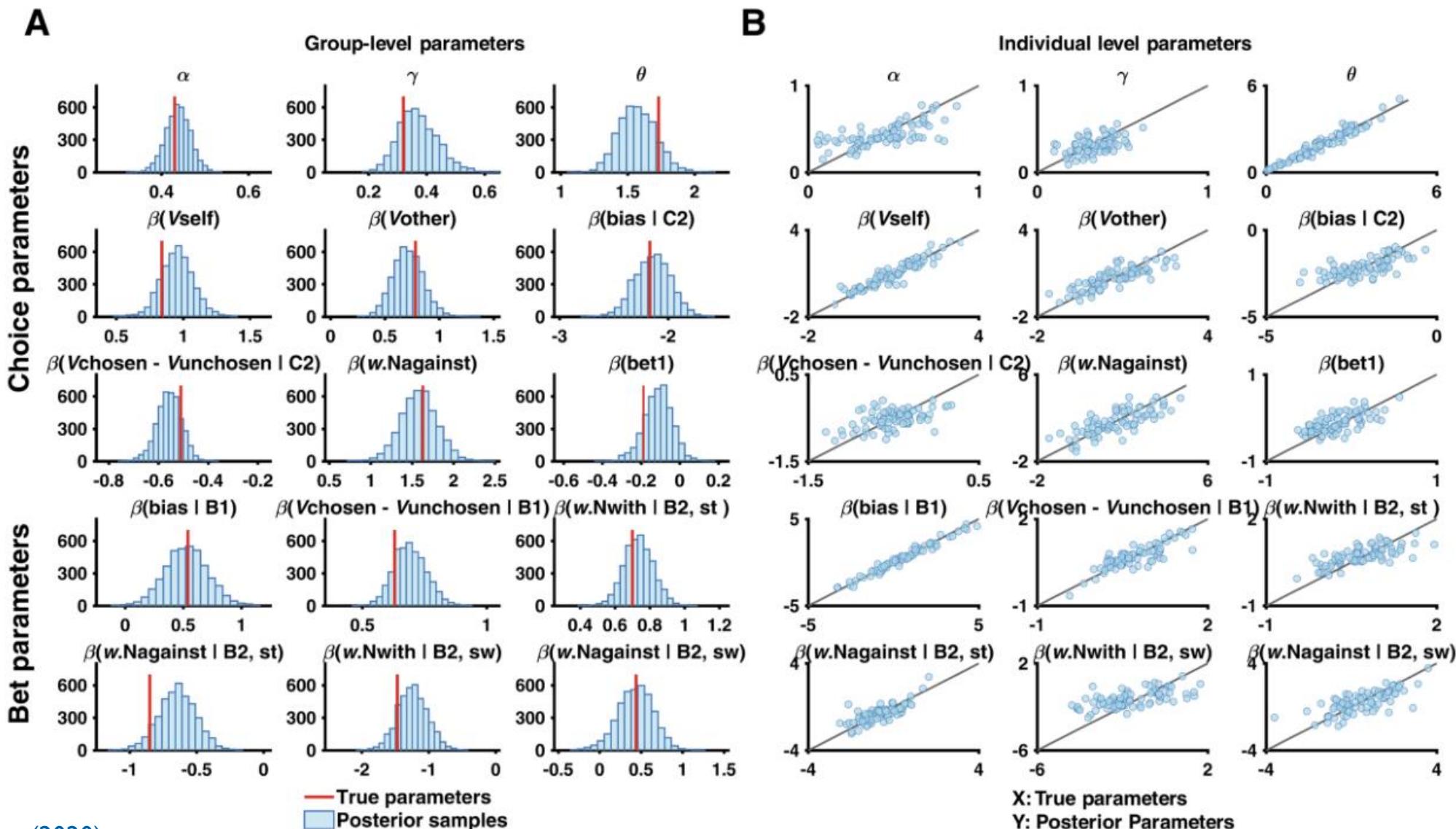
Model recovery: are models identifiable?

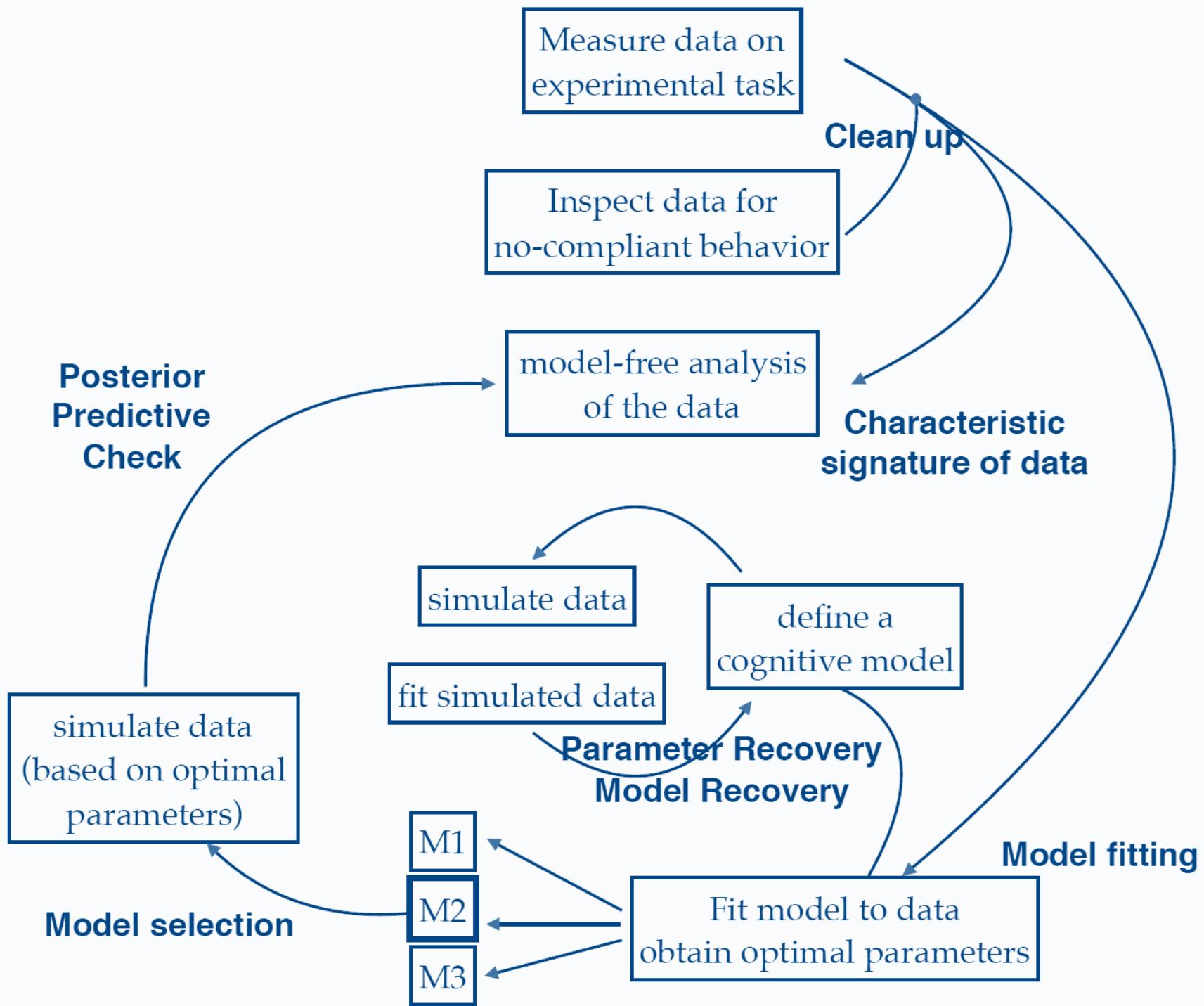
generative
process

fitting
process

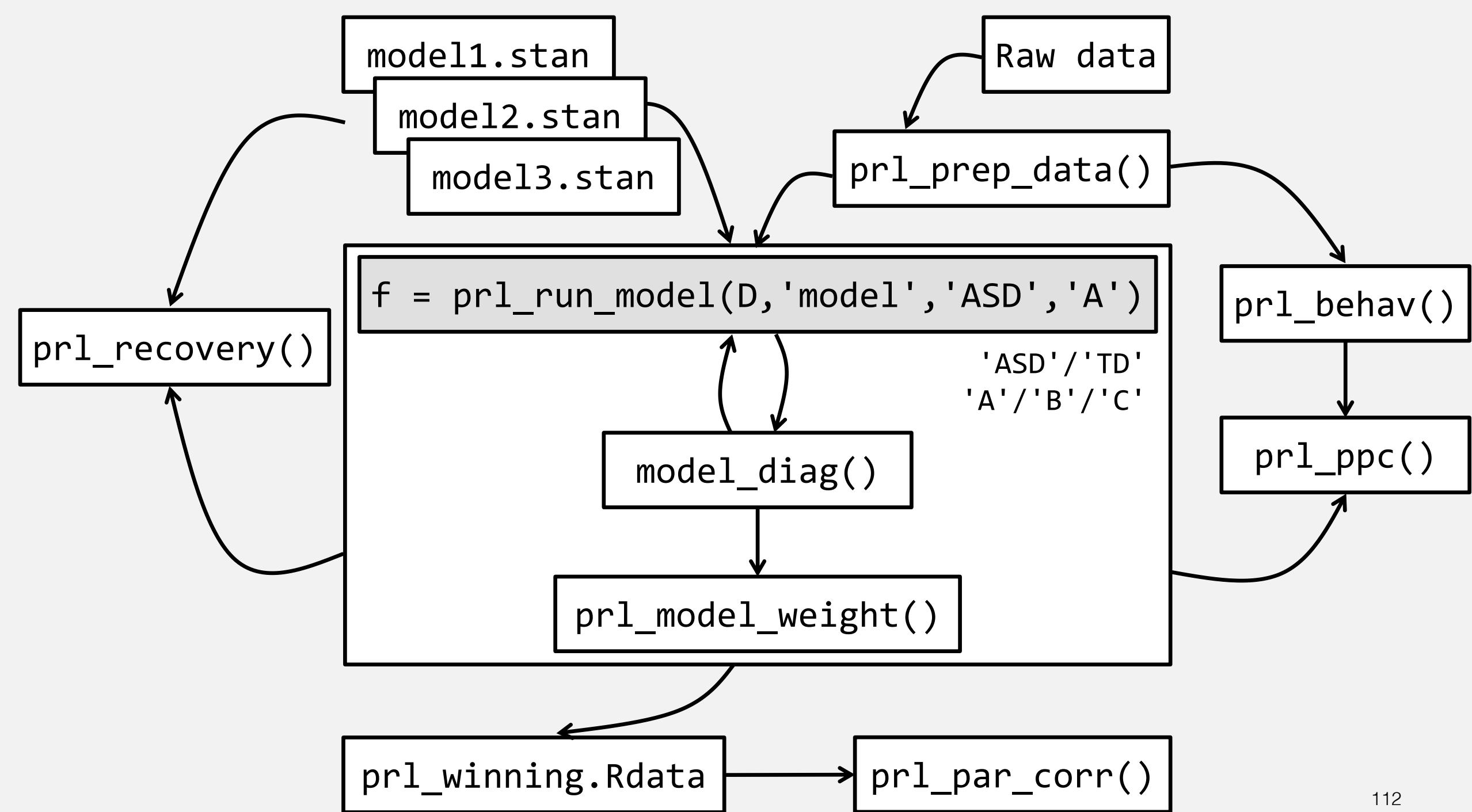


Parameter recovery: are parameters identifiable?

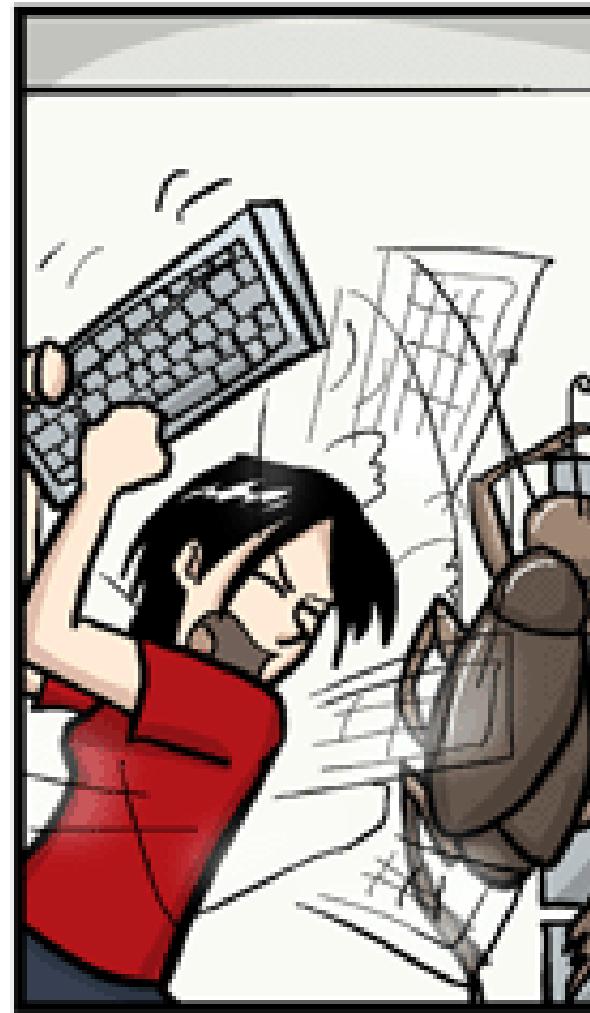
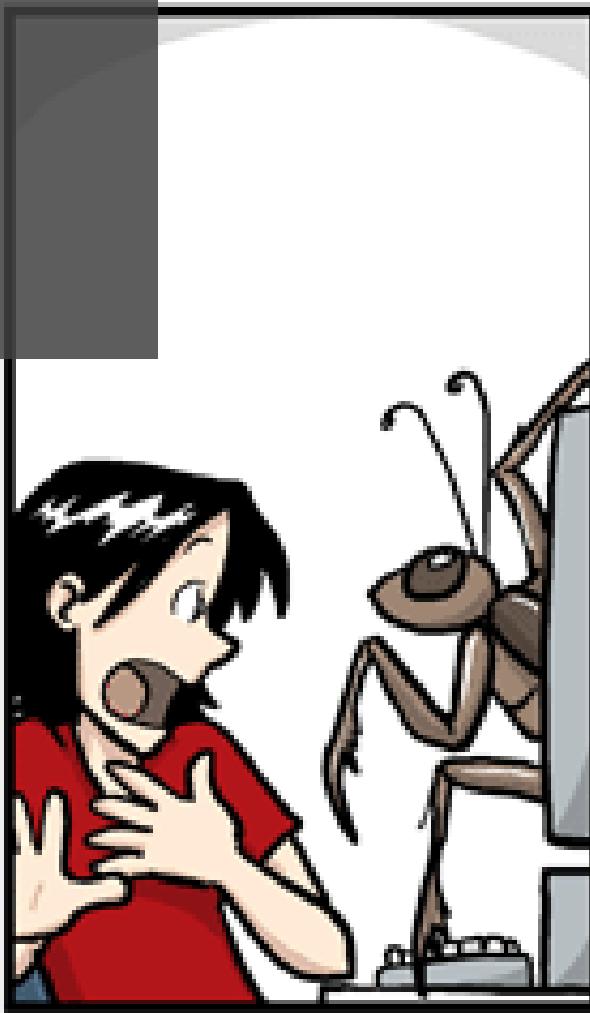




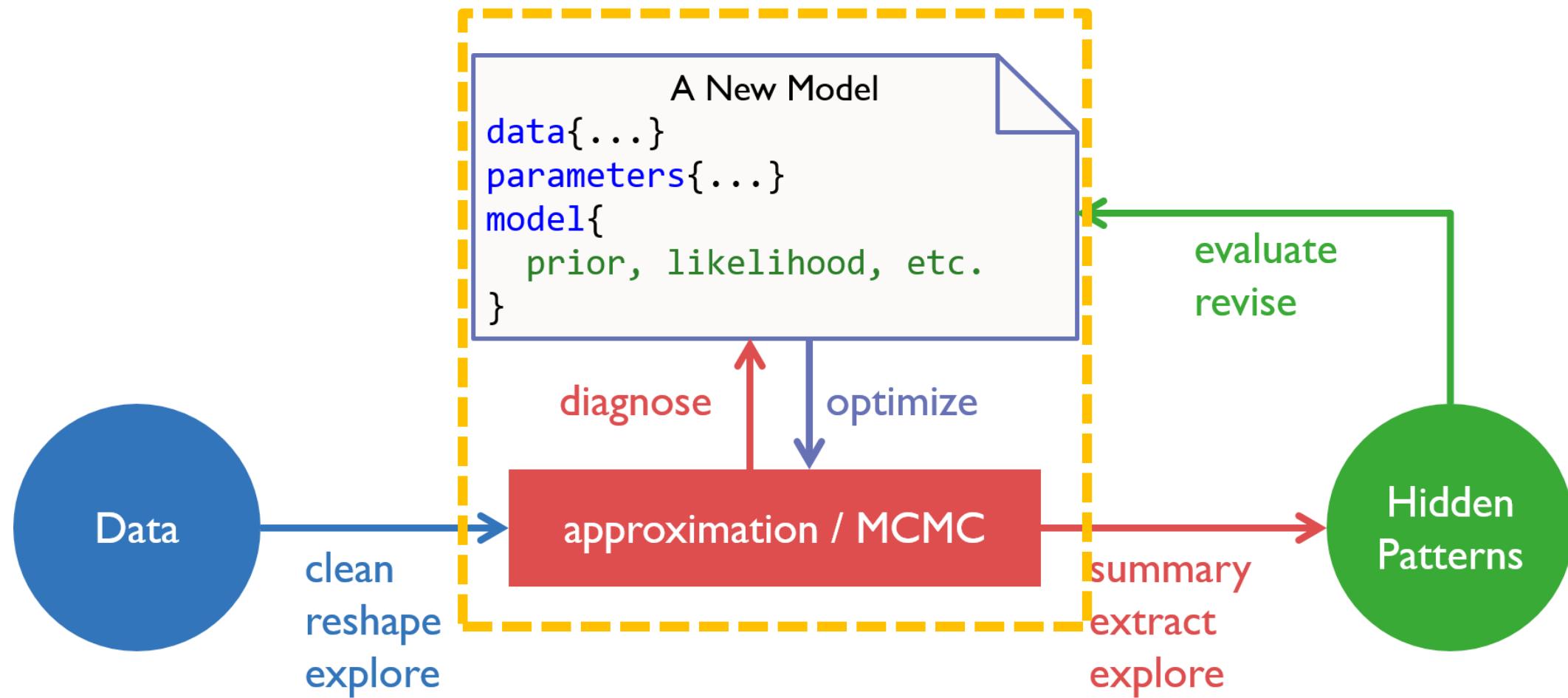
Adapted from Jan Gläscher's workshop



STAN DEBUGGING



JORGE CHAM © 2005



Stan Style Tips

cognitive model
statistics
computing

Make it Reproducible

- Scripts are good documentations!
- Save your seed (not cross platform*)

Make it Readable

- Choose a consistent style
- Give meaningful variable names

Start with Simulated Data

Design Top-Down, Code Bottom-Up

Write Comments

- Code never lies!



* Stan seed depends on hardware etc.

The Editor of your Choice

cognitive model
statistics
computing



```
data {  
  int<lower=0> w;  
  int<lower=0> N;  
}  
  
parameters {  
  real<lower=0,upper=1> p;  
}  
  
model {  
  p ~ uniform(0,1);  
  w ~ binomial(N, p);  
}
```



```
data {  
  int<lower=0> w;  
  int<lower=0> N;  
}  
  
parameters {  
  real<lower=0,upper=1> p;  
}  
  
model {  
  p ~ uniform(0,1);  
  w ~ binomial(N, p);  
}
```



```
data {  
  int<lower=0> w;  
  int<lower=0> N;  
}  
  
parameters {  
  real<lower=0,upper=1> p;  
}  
  
model {  
  p ~ uniform(0,1);  
  w ~ binomial(N, p);  
}
```



```
data {  
  int<lower=0> w;  
  int<lower=0> N;  
}  
  
parameters {  
  real<lower=0,upper=1> p;  
}  
  
model {  
  p ~ uniform(0,1);  
  w ~ binomial(N, p);  
}
```

* Click on each logo to visit their homepage.

** [Comparison](#)

Common Error / Warning Types

cognitive model
statistics
computing

ERRORS

forget “ ; ”
mis-indexing: mismatch or constant
support mismatch
improper constrain
improper dimension declaration
vectorizing when not supported
wrong data type
wrong distribution names
forget priors
miss spelling

WARNINGS

forget last blank line
use earlier version of Stan
numerical problems
divergent transitions
hit max_treedepth
BFI too low
improper prior

Debugging in Stan

- always use a *.stan file
- press  in RStudio
- use `lookup()`
- start with simulated data
- be careful with copy/paste
- run 1 chain, 1 sample
- debugging by printing

```
for (s in 1:1) {
    vector[2] v;
    real pe;
    v <- initV;

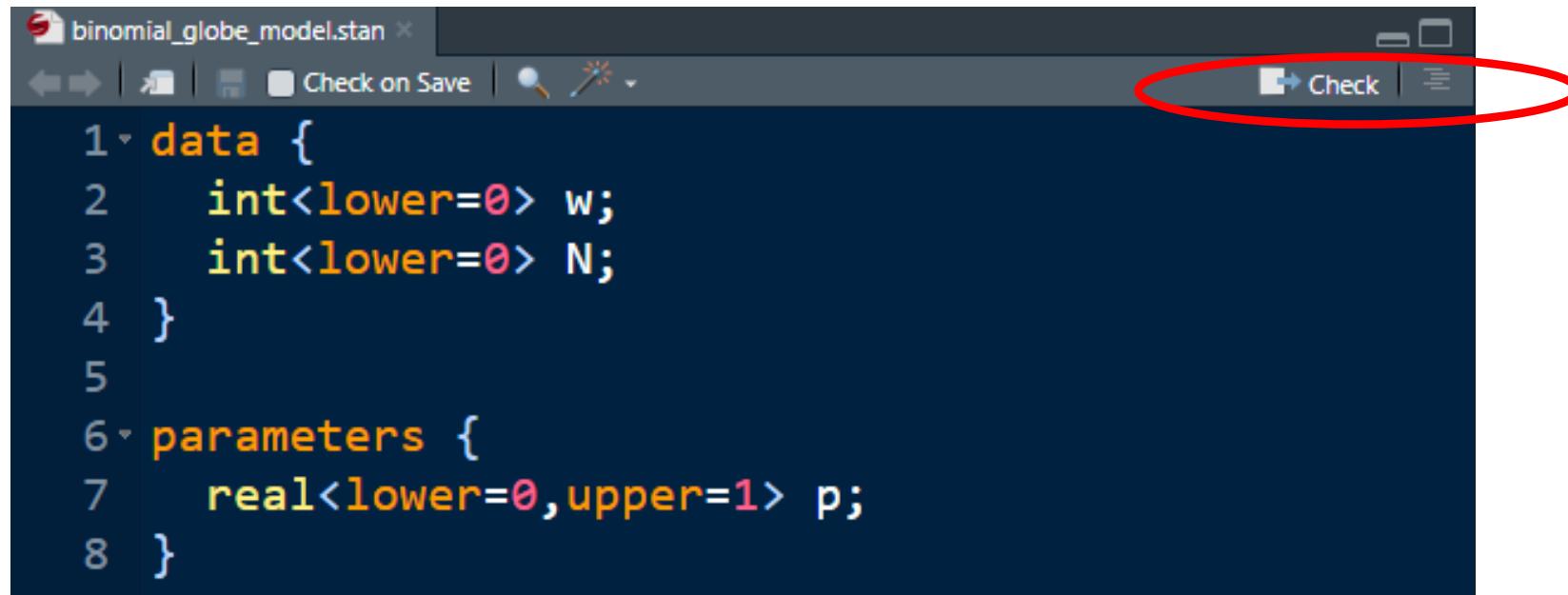
    for (t in 1:nTrials) {
        choice[s,t] ~ categorical_logit( tau[s] * v );

        print("s = ", s, ", t = ", t, ", v = ", v);

        pe <- reward[s,t] - v[choice[s,t]];
        v[choice[s,t]] <- v[choice[s,t]] + lr[s] * pe;
    }
}
```

```
> lookup(dnorm)
      StanFunction          Arguments ReturnType Page SamplingStatement
344     normal           (reals mu, reals sigma)    real   369        TRUE
348     normal_log (reals y, reals mu, reals sigma)    real   369       FALSE
```

Debugging Stan in RStudio



The screenshot shows the RStudio interface with a Stan script named "binomial_globe_model.stan" open. The code is as follows:

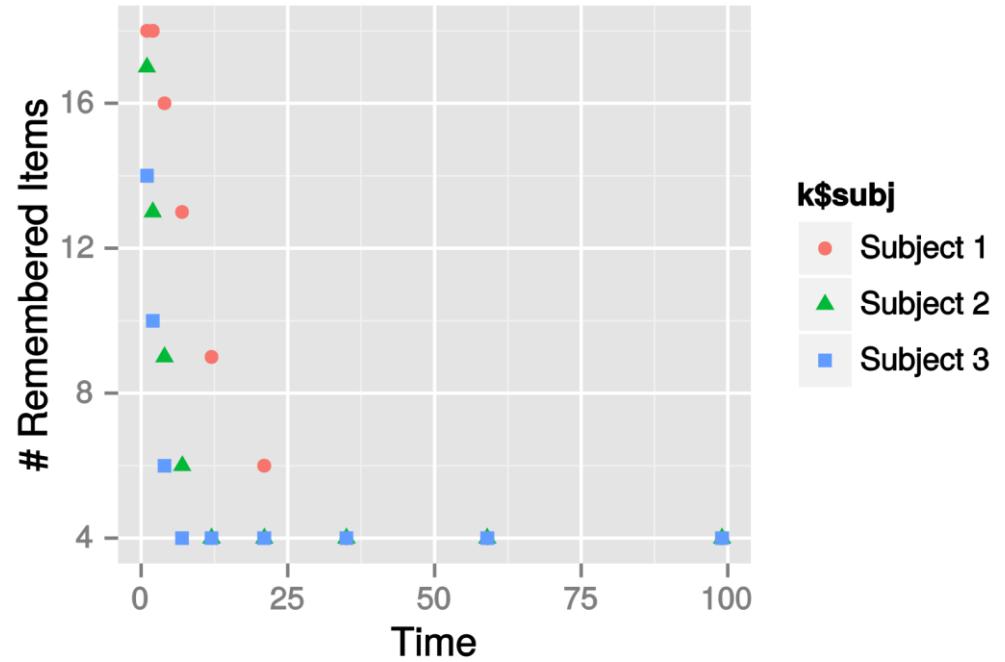
```
1 data {  
2     int<lower=0> w;  
3     int<lower=0> N;  
4 }  
5  
6 parameters {  
7     real<lower=0,upper=1> p;  
8 }
```

The toolbar at the top right includes a "Check" button, which is circled in red to indicate it's the focus of the discussion.

```
rstan::rstudio_stanc("_scripts/binomial_globe_model.stan")
```

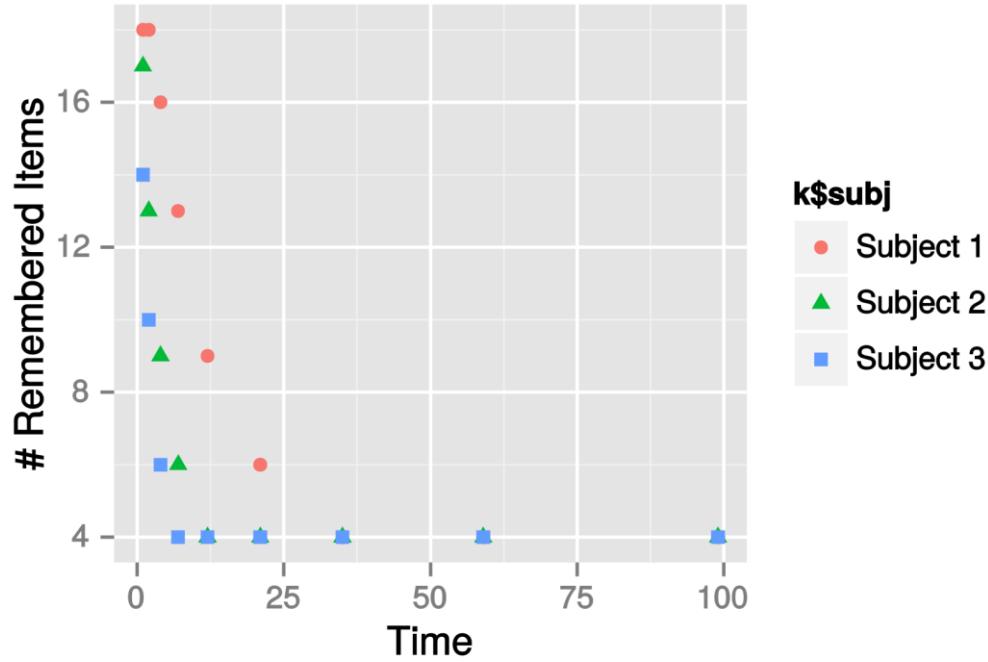


Example: Memory Retention



Subject	Time Interval								
	1	2	4	7	12	21	35	59	99
1	18	18	16	13	9	6	4	4	4
2	17	13	9	6	4	4	4	4	4
3	14	10	6	4	4	4	4	4	4

Simple Exponential Decay Model



$$\theta_t = \min(1.0, \exp(-\alpha t) + \beta)$$

↓
 $p(\text{remember})$ decay rate baseline

Exercise XIV

cognitive model
statistics
computing

.../09.debugging/_scripts/exp_decay_main.R

TASK: Debugging the Memory retention model

>= 8 errors!

```
> dataList
$`k`
  [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
[1,]  18   18   16   13    9    6    4    4    4
[2,]  17   13    9    6    4    4    4    4    4
[3,]  14   10    6    4    4    4    4    4    4

$nItem
[1] 18

$intervals
[1] 1 2 4 7 12 21 35 59 99

$ns
[1] 3

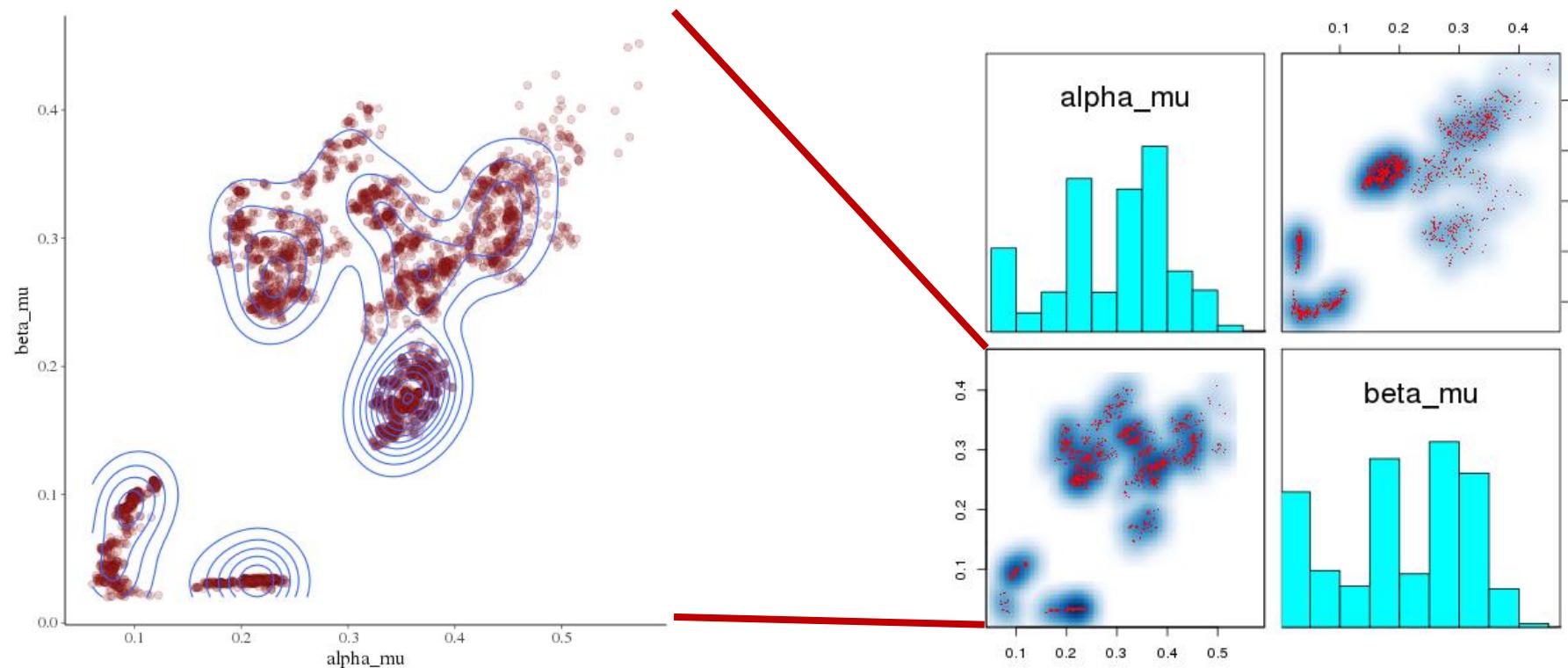
$nt
[1] 9
```



Satisfied with the results?

cognitive model
statistics
computing

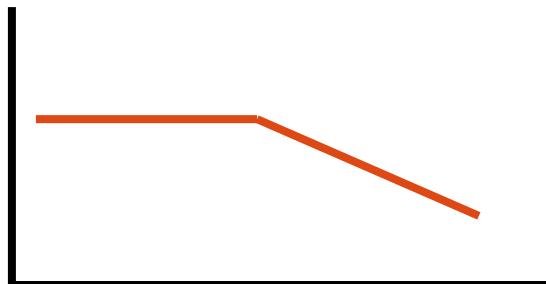
```
Warning messages:  
1: There were 3998 divergent transitions after warmup. Increasing adapt_delta above 0.8 may help. See  
http://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup  
2: Examine the pairs() plot to diagnose sampling problems
```



Why Stan Fails?

```
for (s in 1:ns) {  
  for (t in 1:nt) {  
    theta[s,t] = fmin(1.0, exp(-alpha[s] * intervals[t]) + beta[s]);  
    k[s,t] ~ binomial(nItem, theta[s,t]);  
  }  
}
```

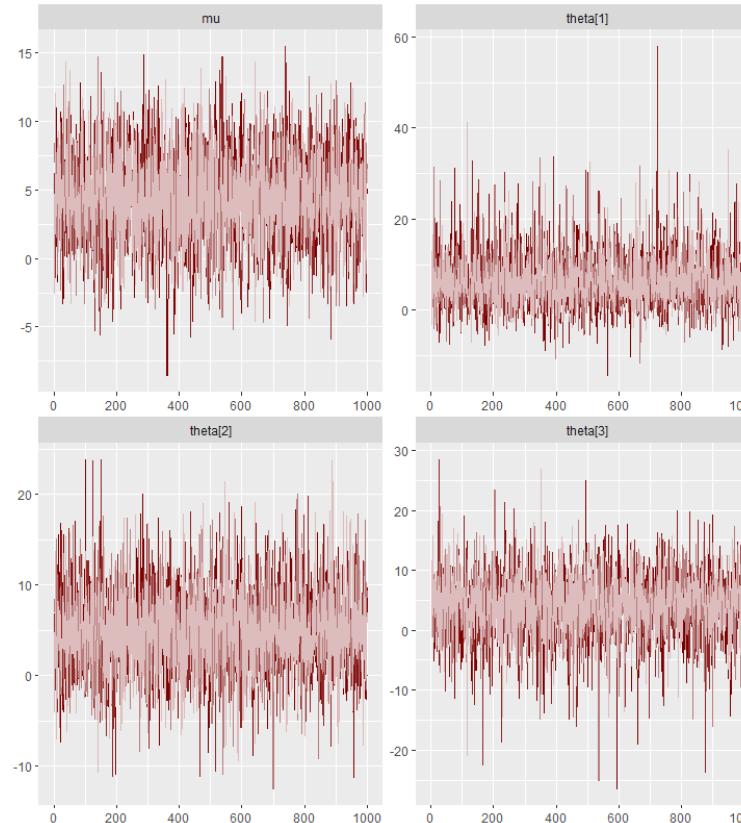
Non-differentiable link (likelihood) functions are bad news,
particularly in Stan, which relies on derivatives.



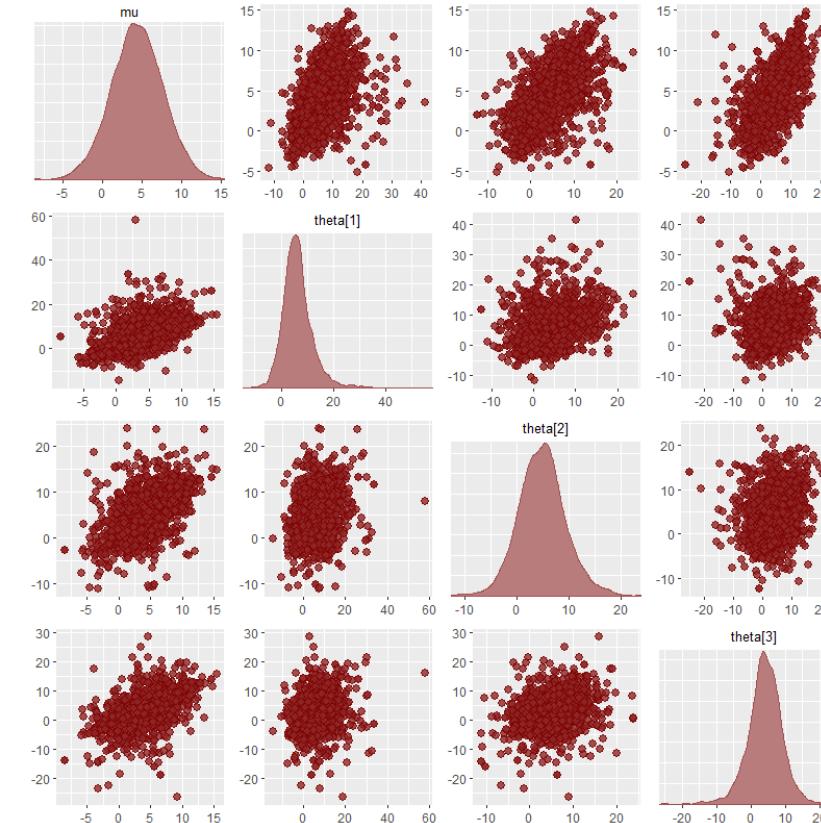
What to look for?

cognitive model
statistics
computing

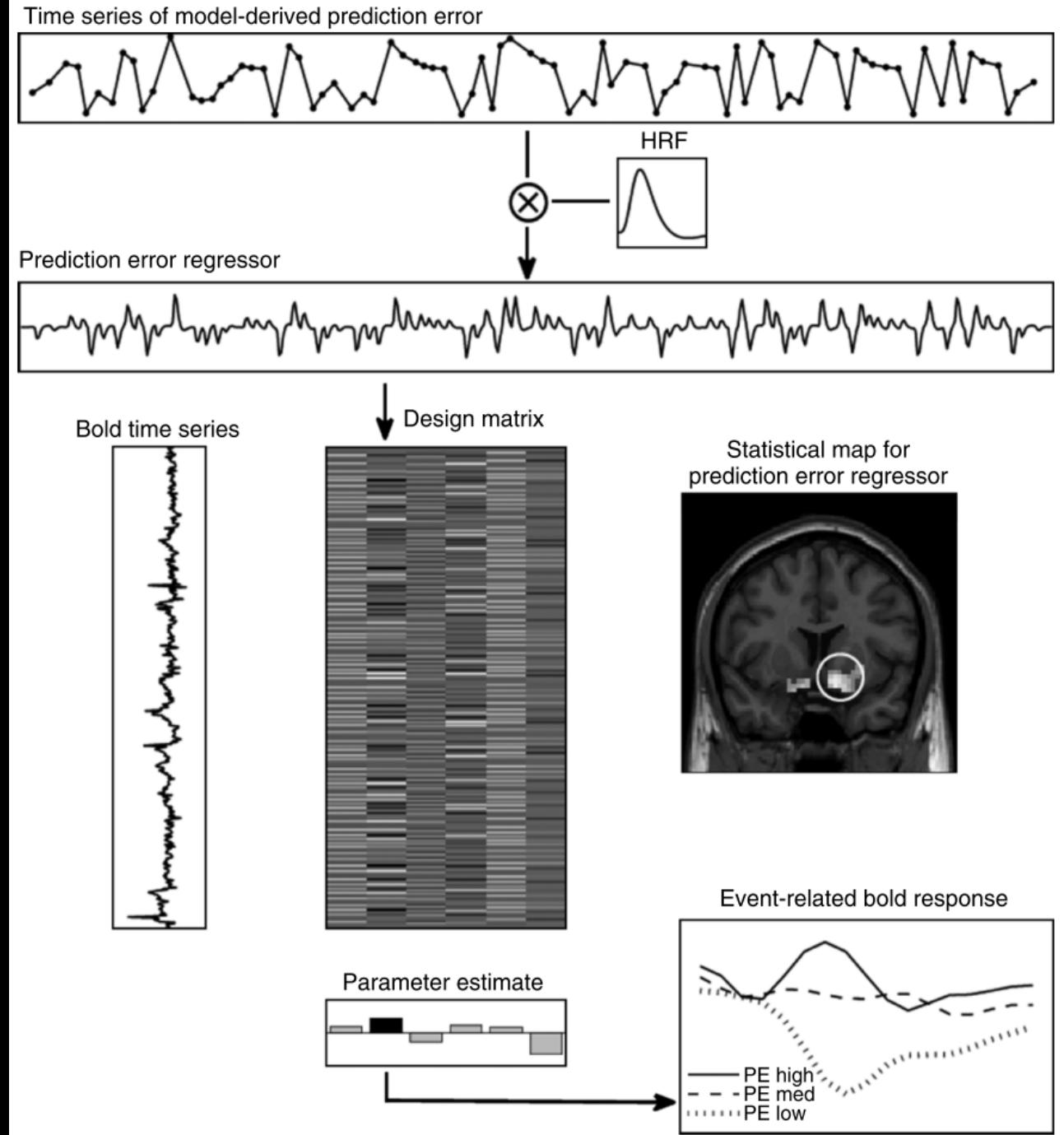
```
> source('stan_utility.R')
> check_all_diagnostics(fit)
[1] "n_eff / iter looks reasonable for all parameters"
[1] "Rhat looks reasonable for all parameters"
[1] "0 of 4000 iterations ended with a divergence (0%)"
[1] "0 of 4000 iterations saturated the maximum tree depth of 10 (0%)"
[1] "E-BFMI indicated no pathological behavior"
```



Chain
— 1
— 2
— 3
— 4

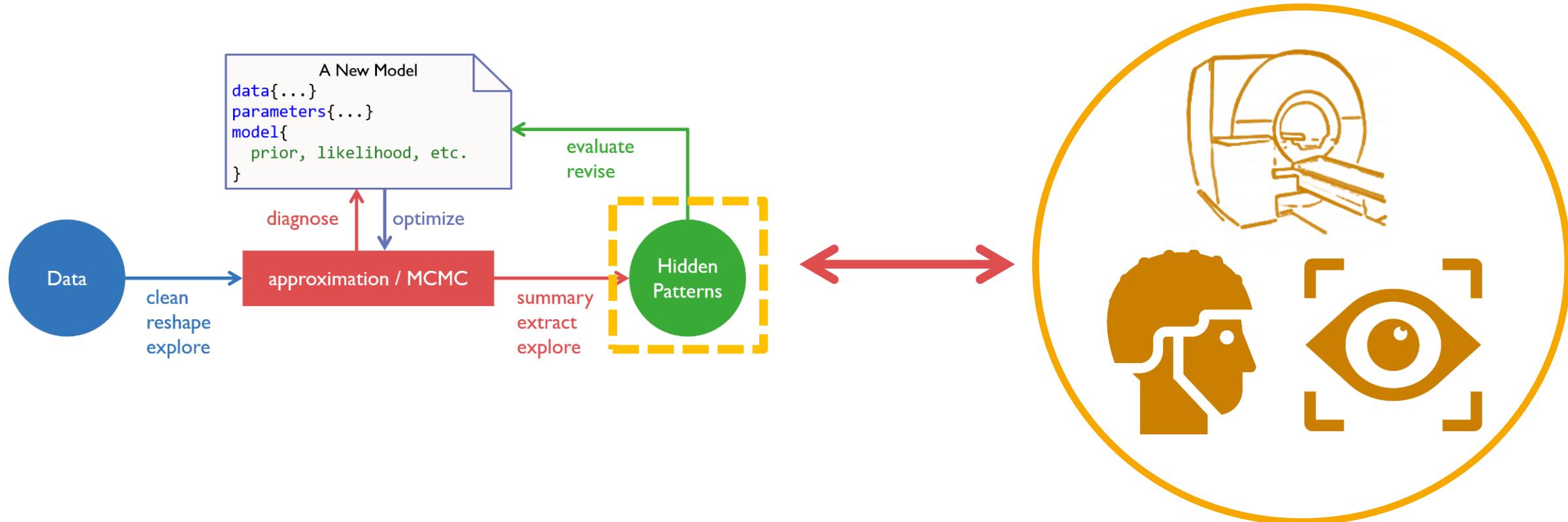


INTRODUCTION TO MODEL-BASED FMRI



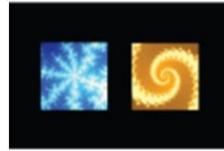
Model-based Analysis

cognitive model
statistics
computing

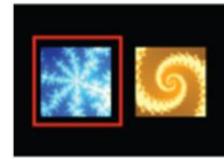


Perform Model-based fMRI

cognitive model
statistics
computing



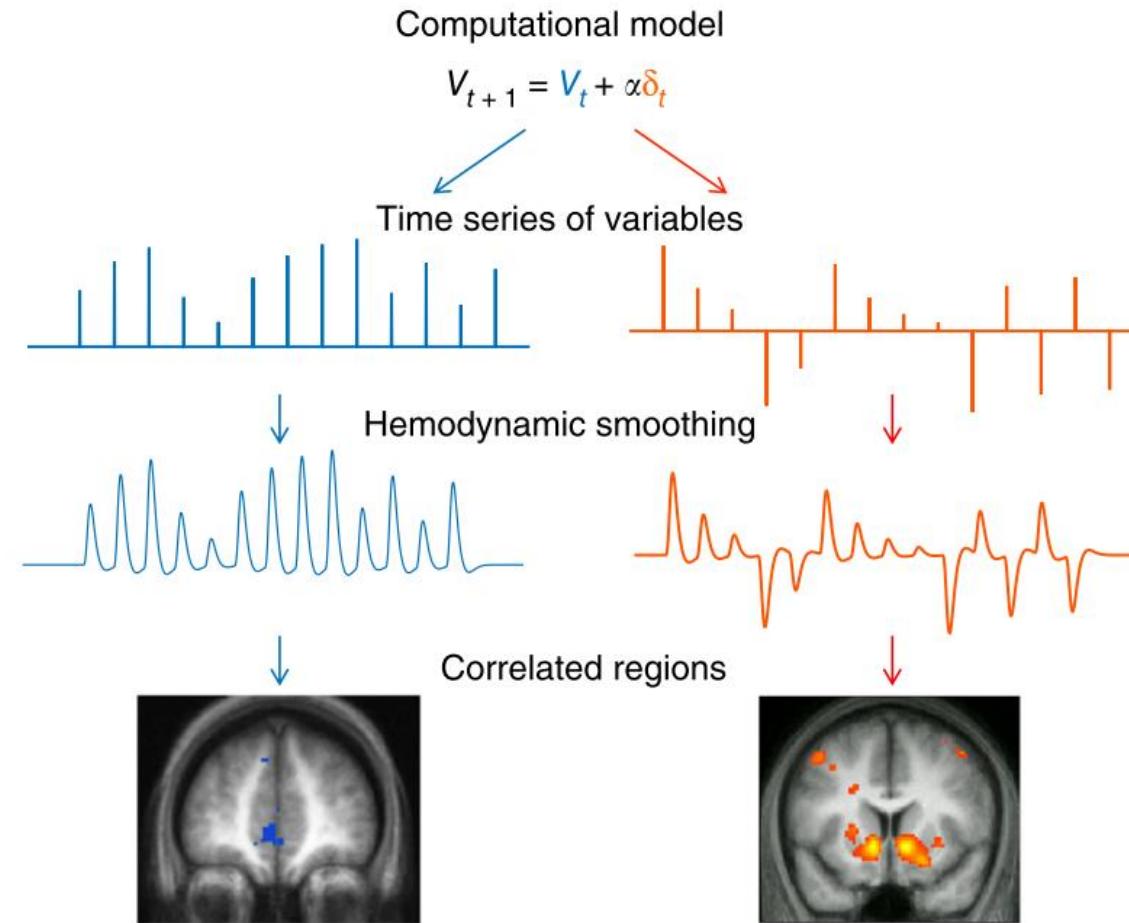
choice presentation



action selection

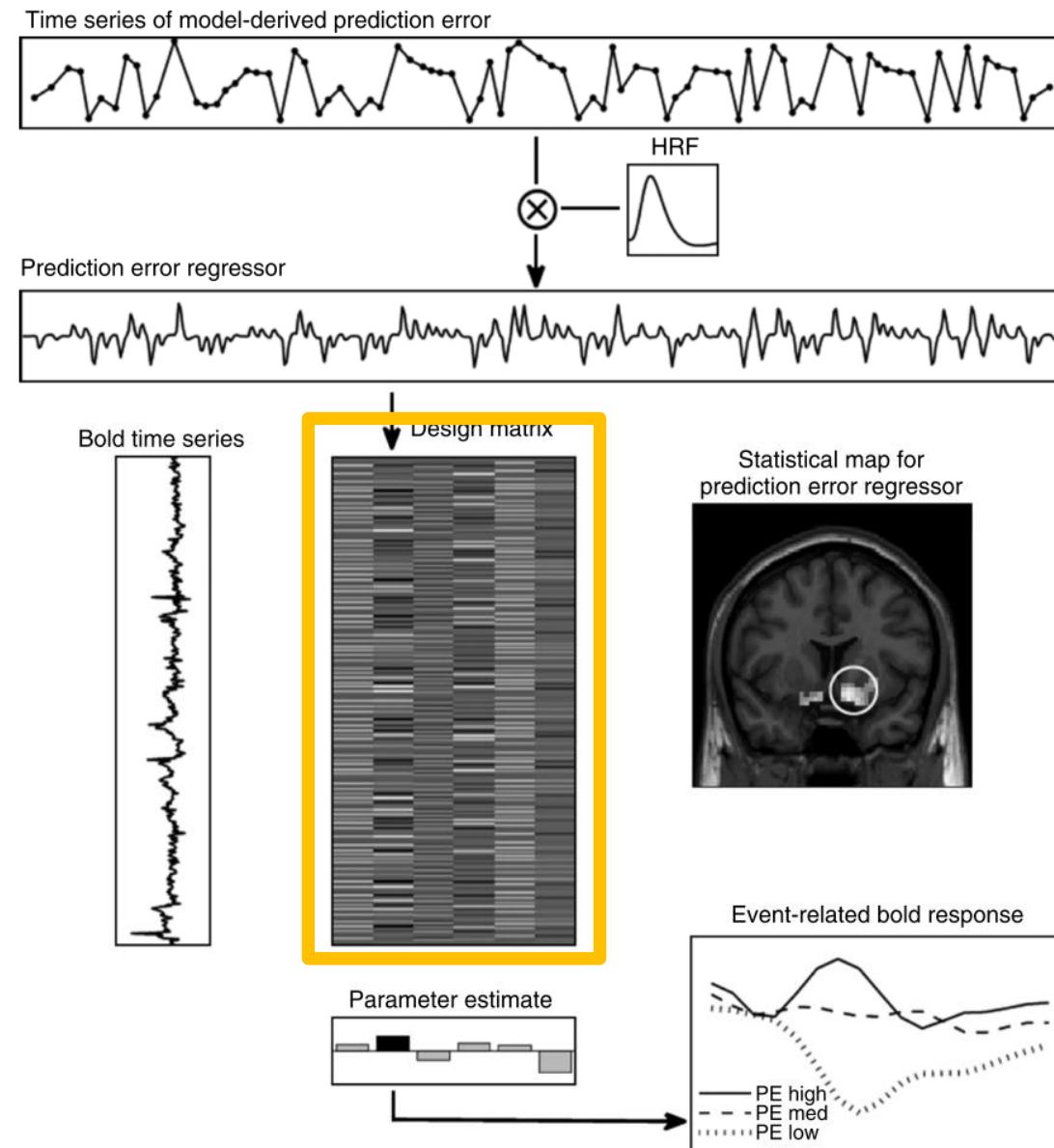


outcome



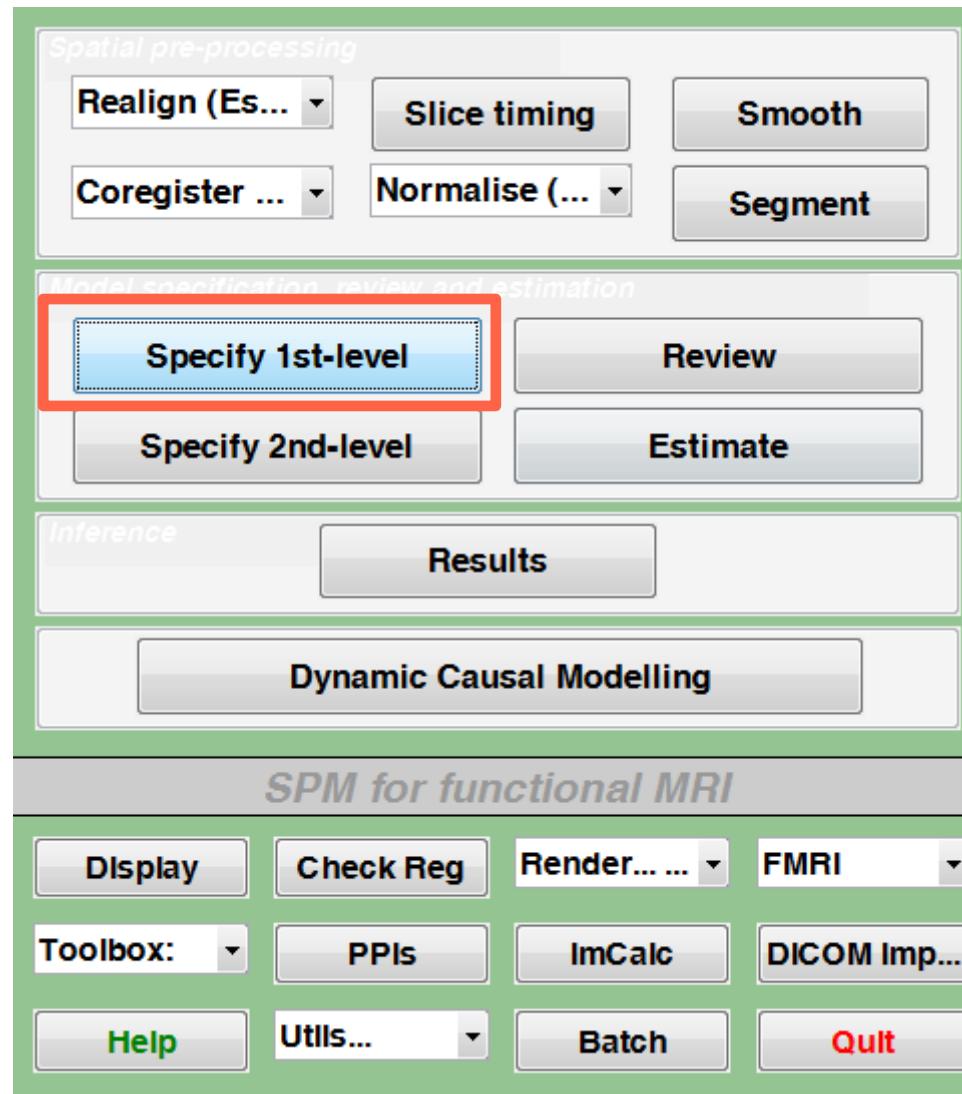
Perform Model-based fMRI (cont.)

cognitive model
statistics
computing



Implementing in SPM12

cognitive model
statistics
computing



The fMRI model specification dialog box shows the following settings:

- fMRI model specific:** Microtime resolution (16), Microtime onset (8).
- Data & Design:** Subject/Session, Scans, Conditions, Condition Name, Onsets, Durations, Time Modulation (No Time Modulation).
- Parametric Modulations:** (highlighted with a red box)
 - Parameter, Name, Values, Polynomial Expansion, Orthogonalise modulations (Yes).
 - multiple conditions, Regressors, Multiple regressors, High-pass filter (128).
- Factorial design:** Basis Functions, Canonical HRF, Model derivatives (No derivatives).
- Current Item:** Parametric Modulations.
- New:** Parameter, Replicate: Parameter (1), Delete: Parameter (1).

SPM12 – batch scripting

cognitive model
statistics
computing

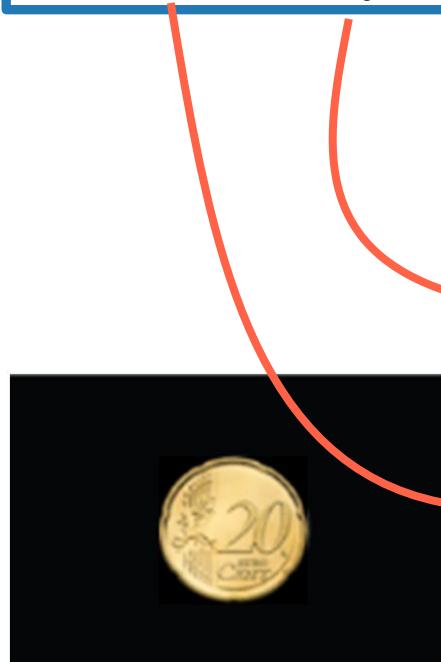
```
matlabbatch{1}.spm.stats.fmri_spec.sess.cond(cnt).name = 'onsetPE';
matlabbatch{1}.spm.stats.fmri_spec.sess.cond(cnt).onset = onscat.sub(i_sub).cueoutcome;
matlabbatch{1}.spm.stats.fmri_spec.sess.cond(cnt).duration = 0;
matlabbatch{1}.spm.stats.fmri_spec.sess.cond(cnt).tmod = 0;
matlabbatch{1}.spm.stats.fmri_spec.sess.cond(cnt).pmod.name = 'PE';
matlabbatch{1}.spm.stats.fmri_spec.sess.cond(cnt).pmod.param = pe(i_sub);
matlabbatch{1}.spm.stats.fmri_spec.sess.cond(cnt).pmod.poly = 1;
matlabbatch{1}.spm.stats.fmri_spec.sess.cond(cnt).orth = 0;
```

make sure: length(onset) == length(PE)

A closer look at PE

Prediction error:

$$PE = R_t - V_t$$



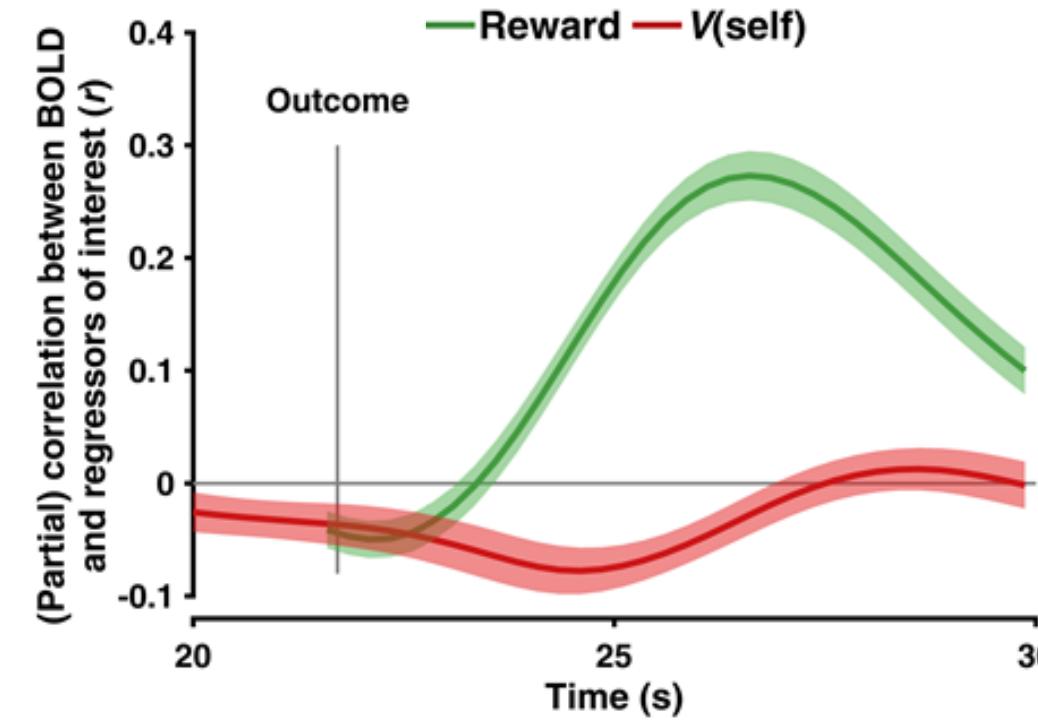
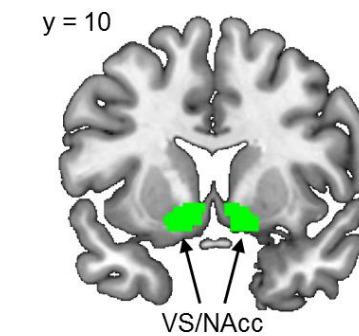
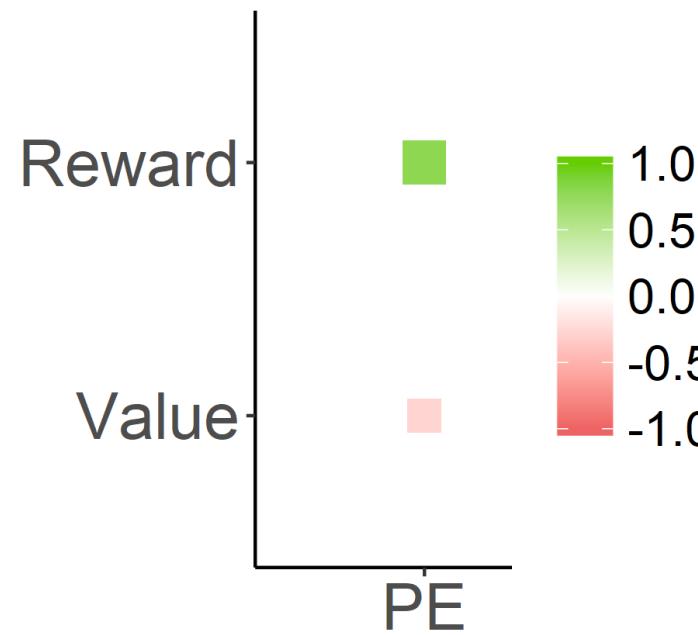
outcome

Q: how to justify the striatal activity is indeed associated with PE, rather than reward?

A closer look at PE

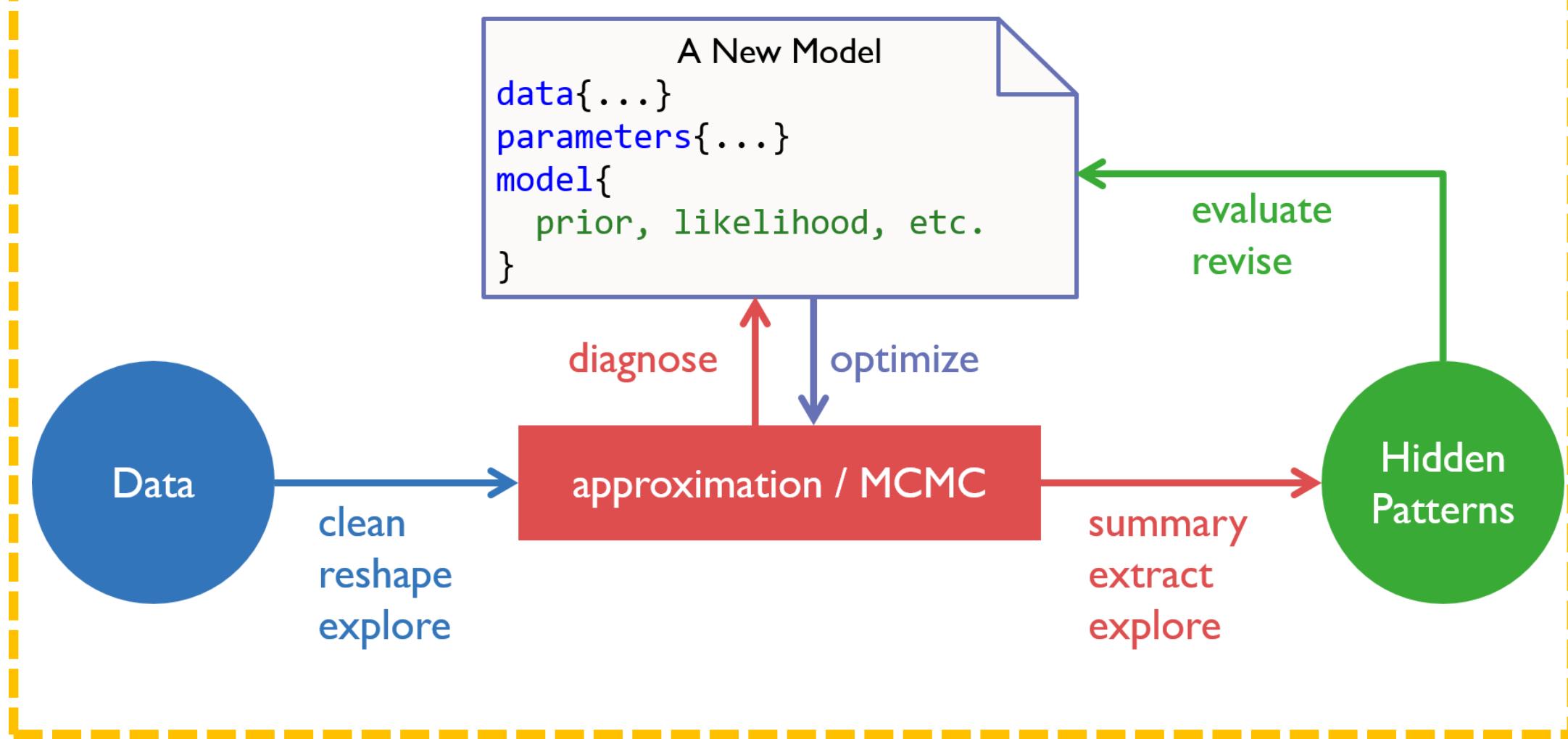
Prediction error:

$$PE = R_t - V_t$$



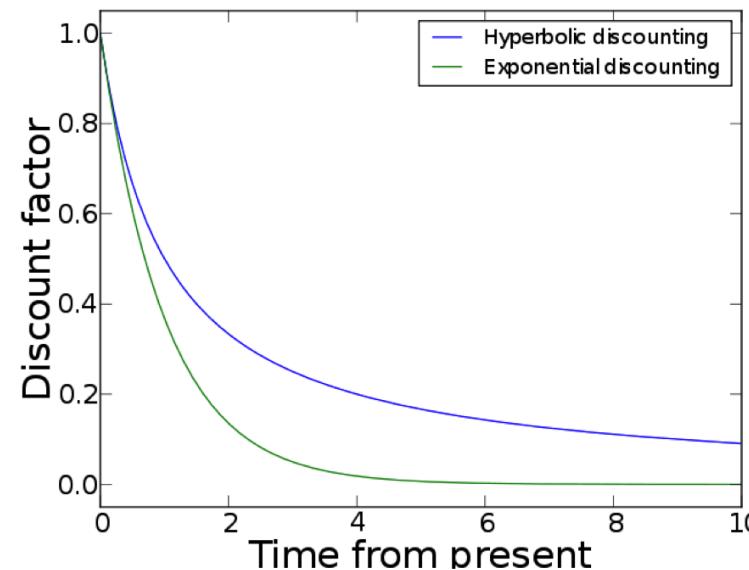
DELAY DISCOUNTING





Delay Discounting Task and Models

cognitive model
statistics
computing



Hyperbolic Discounting Model

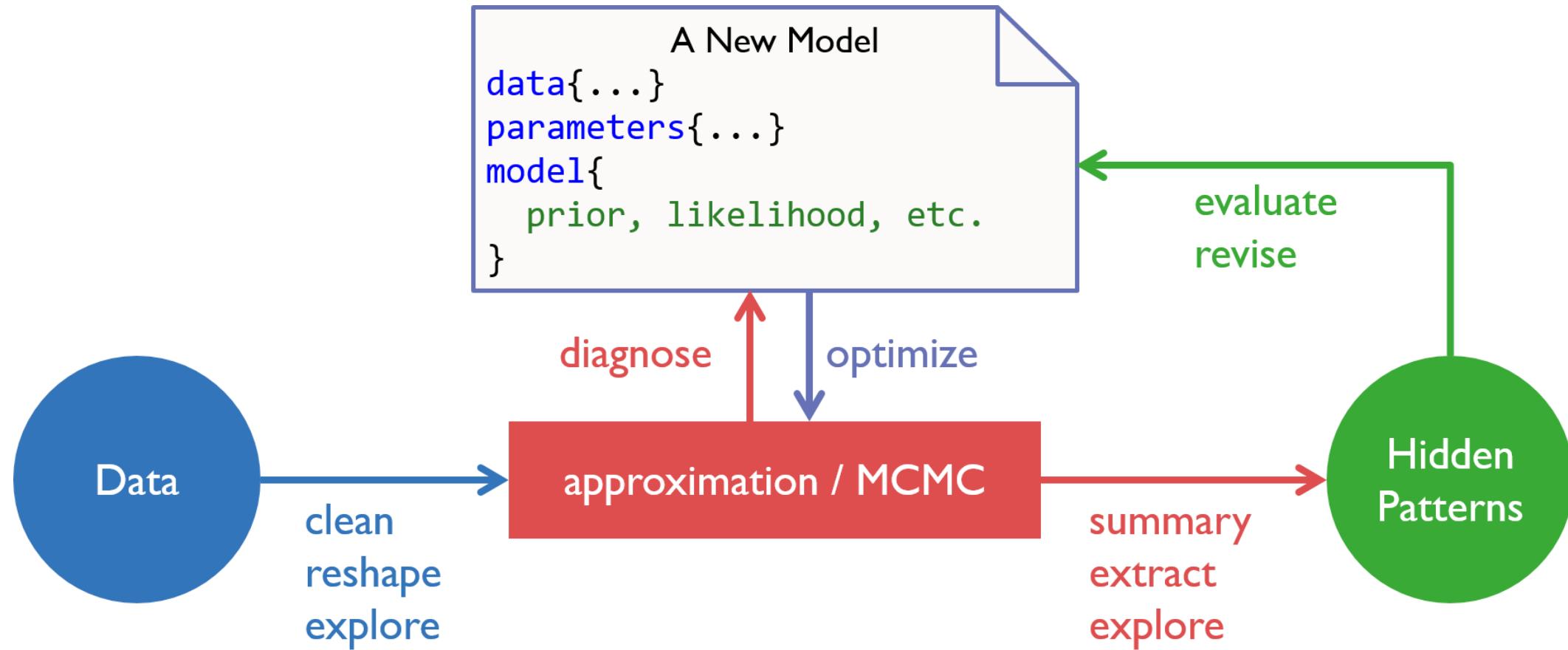
$$SV = \frac{A}{1 + k * delay}$$

Exponential Discounting Model

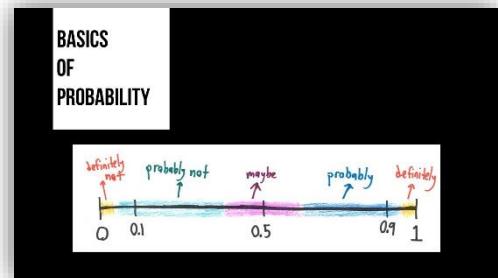
$$SV = A * \exp(-r * delay)$$
$$p(LL) = \frac{1}{1 + \exp^{\text{temp}(v(SS) - v(LL))}}$$

LL - late large option
SS - soon small option

Summary

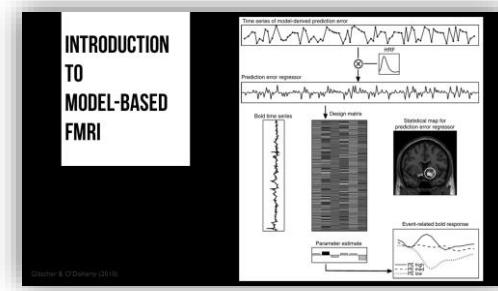
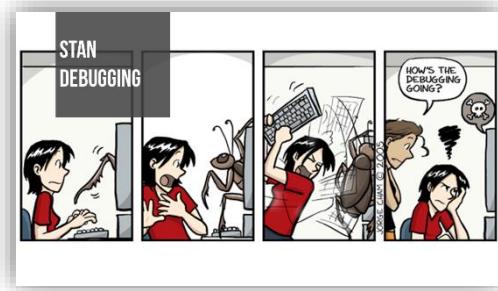
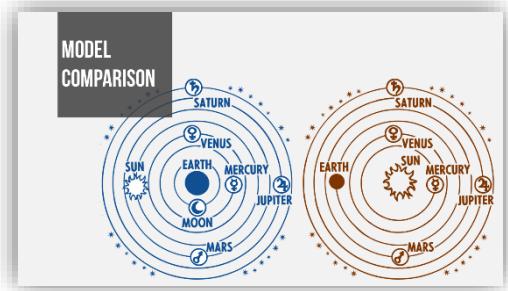
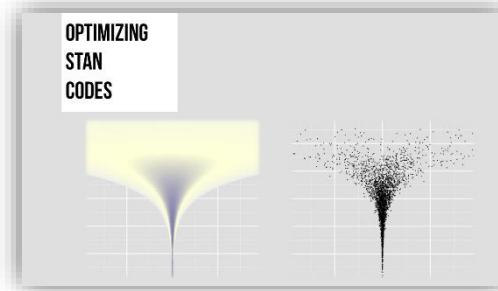
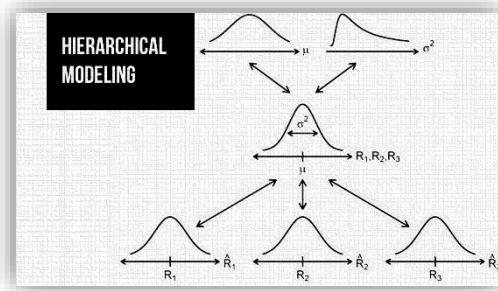
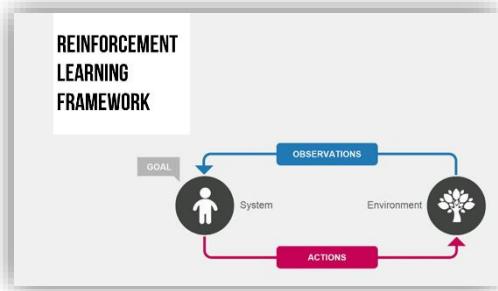
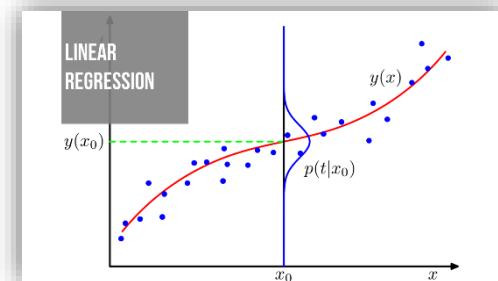
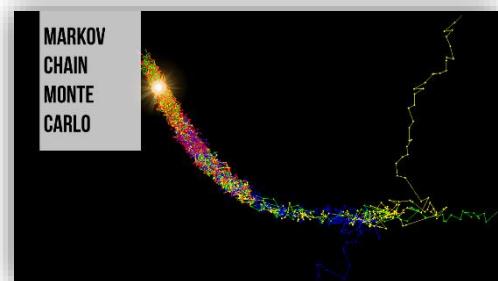
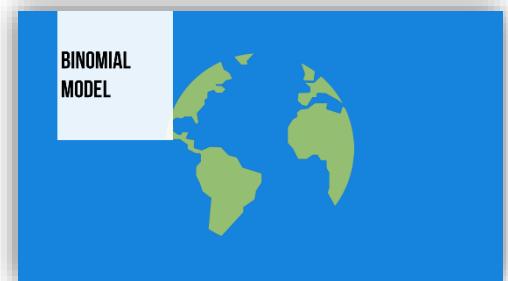
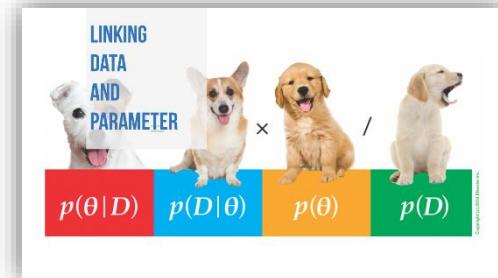


Summary of Topics



BAYES' THEOREM

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$



Summary of Examples/Exercises

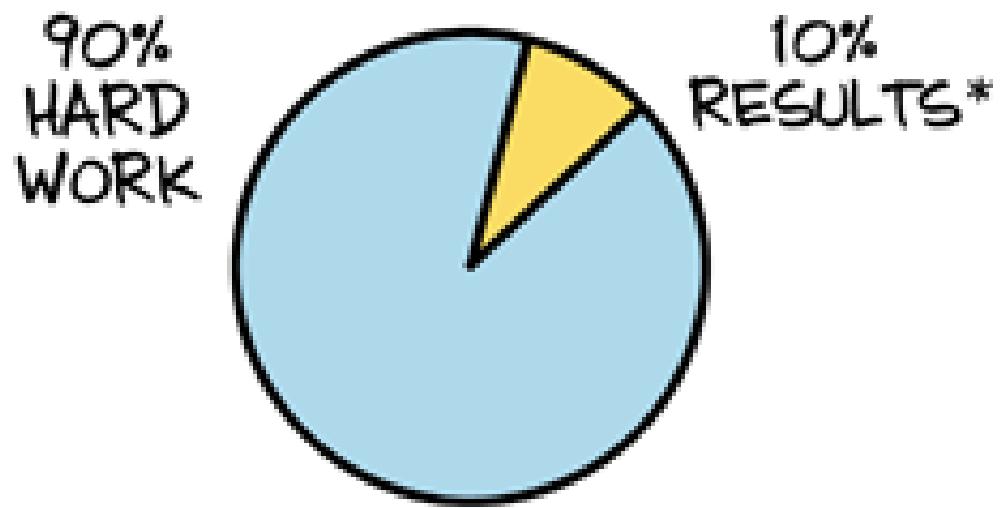
FOLDER	TASK	MODEL
01.R_basics	NA	NA
02.binomial_globe	Globe toss	Binomial Model
03.bernoulli_coin	Coin flip	Bernoulli Model
04.regression_height	Observed weight and height	Linear regression model
05.regression_height_poly		
06.reinforcement_learning	2-armed bandit task	Simple reinforcement learning (RL) model
07.optm_rl		
08.compare_models	Probabilistic reversal learning task	Simple and fictitious RL models
09.debugging	Memory Retention	Exponential decay model
10.model_based	2-armed bandit task	Simple RL model
11.delay_discounting	Delay discounting task	Hyperbolic and exponential discounting model

After the Workshop, you...

- ...are able to implement your own model
- ...feel comfortable with reading mathematical equations
- ...consider the implementation of the “computational modeling” section
- ...gain insightful understanding of Bayesian stats and modeling
- ...take it as a good start and work on it later

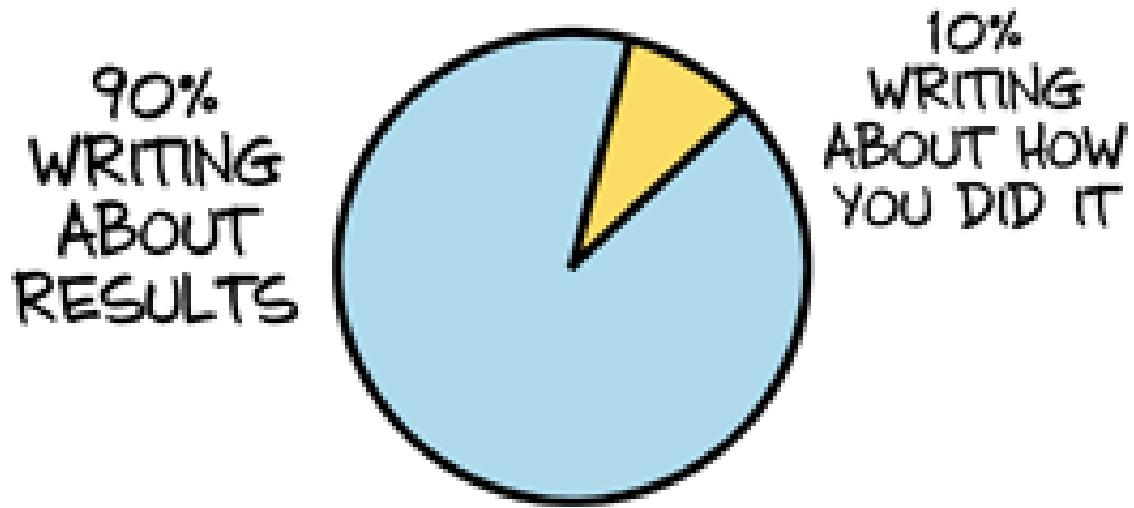
Remember: practice makes perfect!

DOING RESEARCH:



* BEST CASE SCENARIO

WRITING ABOUT RESEARCH:



hBayesDM package

hBayesDM

repo status Active build passing CRAN 1.0.2 – 2019-11-13 downloads 33K

DOI 10.1162/CPSY_a_00002

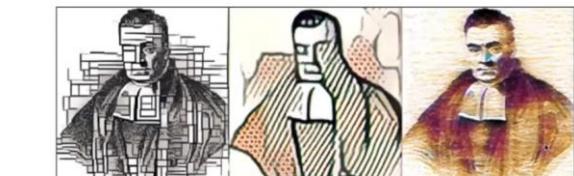
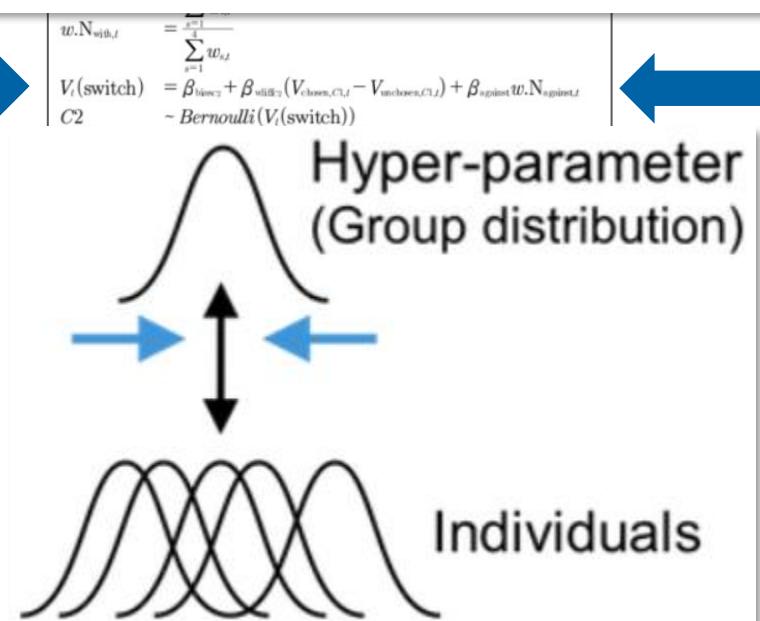
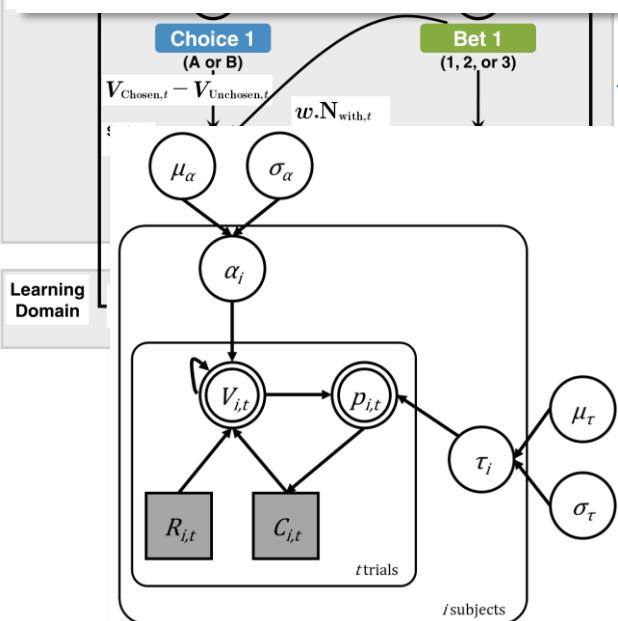
hBayesDM (hierarchical Bayesian modeling of Decision-Making tasks) is a user-friendly package that offers hierarchical Bayesian analysis of various computational models on an array of decision-making tasks. hBayesDM uses Stan for Bayesian inference.



Cited by 115



Stan



Bayesian Statistics and Hierarchical Bayesian Modeling for Psychological Science

Lecture 01

Lei Zhang

Social, Cognitive and Affective Neuroscience Unit (SCAN-Unit)
Department of Cognition, Emotion, and Methods in Psychology

https://github.com/lei-zhang/BayesCog_Wien



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

Woo-Young Ahn¹, Nathaniel Haines¹, and Lei Zhang²

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

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

Keywords: Reinforcement learning; Decision-making; Hierarchical Bayesian modeling; Model-based fMRI

cognitive model
statistics
computing

Task (alphabetical order)	Model name	hBayesDM function	References (see below for full citations)
Balloon Analogue Risk Task	4 parameter model	bart_4par	Wallsten et al. (2005)
Choice reaction time Task	Drift diffusion model Linear Ballistic Accumulator model	choiceRT_ddm choiceRT_lba	Ratcliff (1978) S. Brown & Heathcote (2008) Annis et al. (2017)
Choice under Risk and Ambiguity (CRA) Task	Linear model Exponential model	cra_linear cra_exp	Levy et al. (2009)
Delay Discounting Task	Constant-Sensitivity (CS) model Exponential model Hyperbolic model	dd_cs dd_exp dd_hyp	Ebert & Prelec (2007) Samuelson (1937) Mazur (1987)
Iowa Gambling Task (IGT)	Prospect Valence Learning-DecayRI Prospect Valence Learning-Delta Value-Plus-Perseverance (VPP) Outcome-Represent. Learning (ORL)	igt_pvl_decay igt_pvl_delta igt_vpp igt_orl	Ahn et al. (2011; 2014) Ahn et al. (2008) Worthy et al. (2013) Haines et al. (in press)
Orthogonalized Go/Nogo Task	RW+noise RW+noise+go bias RW+noise+go bias+Pav. bias M5 (see Table 1 of the reference)	gng_m1 gng_m2 gng_m3 gng_m4	Guitart-Masip et al. (2012) Guitart-Masip et al. (2012) Guitart-Masip et al. (2012) Cavanagh et al. (2013)
Peer influence task	Other-conferred utility (OCU)	peer_ocu	Chung et al. (2015)
Probabilistic Reversal Learning (PRL) Task	Experience-Weighted Attraction Fictitious update Reward-Punishment (Rew.-Pun.) Fictitious + Rew.-Pun. Fictitious + Rew.-Pun. w/o alpha Fictitious w/o alpha	prl_ewa prl_fictitious prl_rp prl_fictitious_rp prl_fictitious_rp_woa prl_fictitious_woa	Ouden et al. (2013) Glässcher et al. (2009) Ouden et al. (2013)
Probabilistic Selection Task	Q-learning with two learning rates	pst_gainloss_Q	M. J. Frank et al. (2007)
Risk-Aversion Task	Prospect Theory (PT) PT without loss aversion (LA) PT without risk aversion (RA)	ra_prospect ra_noLA ra_noRA	Sokol-Hessner et al. (2009) Tom et al. (2007)
Risky Decision Task	Happiness model	rdt_happiness	Rutledge et al. (2014)
Two-Armed Bandit (Experience-based) Task	Rescorla-Wagner (delta) model	bandit2arm_delta	Erev et al. (2010) Hertwig et al. (2004)
Two Step (TS) Task	7 parameter model 6 parameter model 4 parameter model	ts_7par ts_6par ts_4par	Daw et al. (2011) Wunderlich et al. (2012)
Four-Armed Bandit (Experience-based) Task	Fictive upd.+rew/pun sens. Fictive upd.+rew/pun sens.+lapse	bandit4arm_4par bandit4arm_lapse	Seymour et al. (2012) Seymour et al. (2012)
Ultimatum Game	Ideal Bayesian observer model Rescorla-Wagner (delta) model	ug_bayes ug_delta	Xiang et al. (2013) Gu et al. (2015)
Wisconsin Card Sorting Task	Sequential learning model	wcs_sql	A. J. Bishara et al. (2010)



Workshops / Summer schools

- Cognitive Modeling Academy Hamburg (inaugural, 2021!)
- JAGS and WinBUGS Workshop @ Amsterdam, NL (annual)
- Model-based Neuroscience Summer School @ Amsterdam, NL (annual)
- European Summer School on Computational and Mathematical Modeling of Cognition @ multiple EU sites (biannual)
- Computational Psychiatry Course @ Zürich, CH (annual)
- London Computational Psychiatry Course @ London, UK (annual?)
- Methods in Neuroscience at Dartmouth Computational Summer School @ Dartmouth, USA (annual)
- Brains, Minds & Machines Summer Course @ MIT, USA (annual)
- Kavli Summer Institute in Cognitive Neuroscience @UCSB, USA (annual)
- Neuromatch Academy on Computational Neuroscience @online! (annual)

References

- Ahn, W.-Y., Haines, N., & Zhang, L. (2017). Revealing Neurocomputational Mechanisms of Reinforcement Learning and Decision-Making With the hBayesDM Package. *Computational Psychiatry*, 1, 24–57.
- Carpenter, B., Gelman, A., Hoffman, M. D., Lee, D., Goodrich, B., Betancourt, M., ... & Riddell, A. (2017). Stan: A probabilistic programming language. *Journal of statistical software*, 76(1).
- Cohen, J. D., Daw, N., Engelhardt, B., Hasson, U., Li, K., Niv, Y., ... & Willke, T. L. (2017). Computational approaches to fMRI analysis. *Nature Neuroscience*, 20(3), 304.
- Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., & Rubin, D. B. (2014). *Bayesian data analysis* (3rd ed.). New York, NY: CRC Press.
- Gelman, A., & Rubin, D. B. (1992). Inference from iterative simulation using multiple sequences. *Statistical Science*, 7, 457–472.
- Gläscher, J., Hampton, A. N., & O'Doherty, J. P. (2009). Determining a role for ventromedial prefrontal cortex in encoding action-based value signals during reward-related decision making. *Cerebral Cortex*, 19(2), 483-495.
- Gläscher, J. & O'Doherty, J. P. (2010). Model-based approaches to neuroimaging: combining reinforcement learning theory with fMRI data. *Wiley Interdisciplinary Reviews: Cognitive Science*, 1(4), 501-510.
- Hampton, A. N., Adolphs, R., Tyszka, J. M., & O'Doherty, J. P. (2007). Contributions of the amygdala to reward expectancy and choice signals in human prefrontal cortex. *Neuron*, 55(4), 545-555.
- Kruschke, J. (2014). *Doing Bayesian data analysis: A tutorial with R, JAGS, and Stan*. Academic Press.
- Lee, M. D., & Wagenmakers, E. J. (2014). *Bayesian cognitive modeling: A practical course*. Cambridge university press.
- Lewandowsky, S., & Farrell, S. (2010). *Computational modeling in cognition: Principles and practice*. Sage Publications.
- McElreath, R. (2018). *Statistical Rethinking: A Bayesian Course with Examples in R and Stan*. CRC Press.
- O'Doherty, J. P., Hampton, A., & Kim, H. (2007). Model-based fMRI and its application to reward learning and decision making. *Annals of the New York Academy of Sciences*, 1104(1), 35-53.
- Rescorla, R. A., & 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, 64-99.
- Sutton, R. S., Barto, A. G., & Bach, F. (1998). *Reinforcement learning: An introduction*. MIT press.
- Vehtari, A., Gelman, A., & Gabry, J. (2017). Practical Bayesian model evaluation using leave-one-out cross-validation and WAIC. *Statistics and Computing*, 27(5), 1413-1432.

Acknowledgement

Antonius Wiegler (MMB, Paris)

Jan Gläscher (UKE, Hamburg)

Woo Young Ahn (SNU, Seoul)

Nate Haines (OSU, OH)

ANY
QUESTIONS?
?

Happy Computing!