

Bayesian Statistics and Bayesian Cognitive Modeling

Lei Zhang

Institute of Systems Neuroscience, University Medical Center Hamburg-Eppendorf

10-12. Oct. 2016, Hamburg
lei.zhang@uke.de
lei-zhang.net

Special thanks to Antonius Wiehler!



Universitätsklinikum
Hamburg-Eppendorf

Recap

What we've learned...

- R Basics
- probability distributions
- Bayes' theorem, $p(\theta|D)$
- Binomial model
- MCMC and Stan

BERNOULLI MODEL



Bernoulli Model

cognitive model
statistics
computing

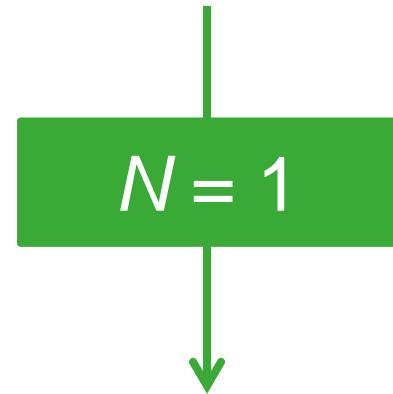
- You are interested in if a coin is biased.
- You will flip the coin.
- You will record whether it comes up a head (h) or a tail (t).
- You might observe 15 heads out of 20 flips.
- What is your degree of belief about the biased parameter ϑ ?



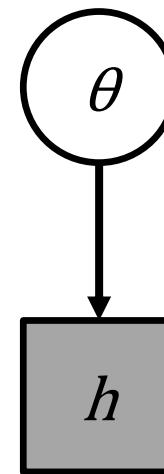
Bernoulli Model

cognitive model
statistics
computing

$$p(w | N, p) = \binom{N}{w} p^w (1-p)^{N-w}$$



$$p(h | \theta) = \theta^h (1 - \theta)^{1-h}$$



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

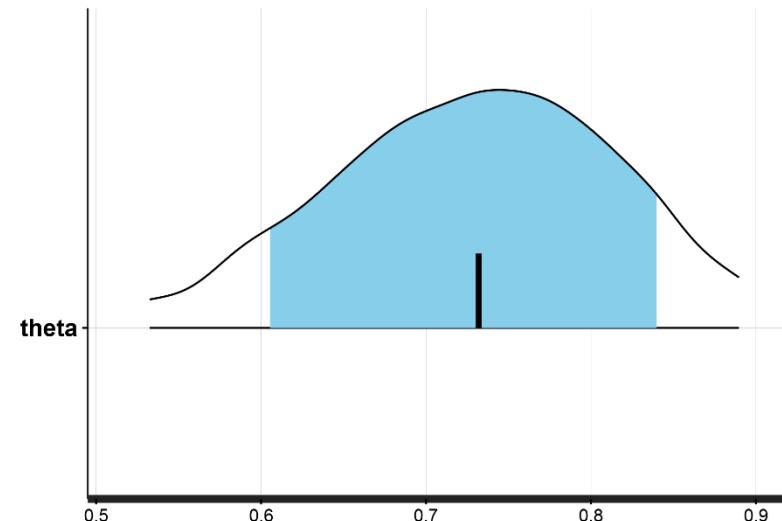
$$h \sim \text{Bernoulli}(\theta)$$

Exercise I

cognitive model
statistics
computing

`.../BayesCog/03.bernoulli_coin/_scripts/bernoulli_coin_main.R`

TASK: complete and fit the Bernoulli model



Possible Optimization?

cognitive model
statistics
computing

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

61.59 secs

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

53.25 secs

Thinking before looping!

DAY2

09:00 – 09:30	Warmup – Bernoulli Model
09:30 – 10:00	Linear Regression Model
10:00 – 10:30	Predictive Check
10:30 – 10:45	Coffee break
10:45 – 11:30	Cognitive Modeling
11:30 – 12:30	Reinforcement Learning Model
12:30 – 13:30	Lunch break
13:30 – 14:00	Fitting Multiple Participants
14:00 – 15:00	Hierarchical Modeling
15:00 – 15:15	Coffee Break
15:15 – 16:15	Optimizing Stan Codes
16:15 – 17:00	Model Comparison

LINEAR REGRESSION

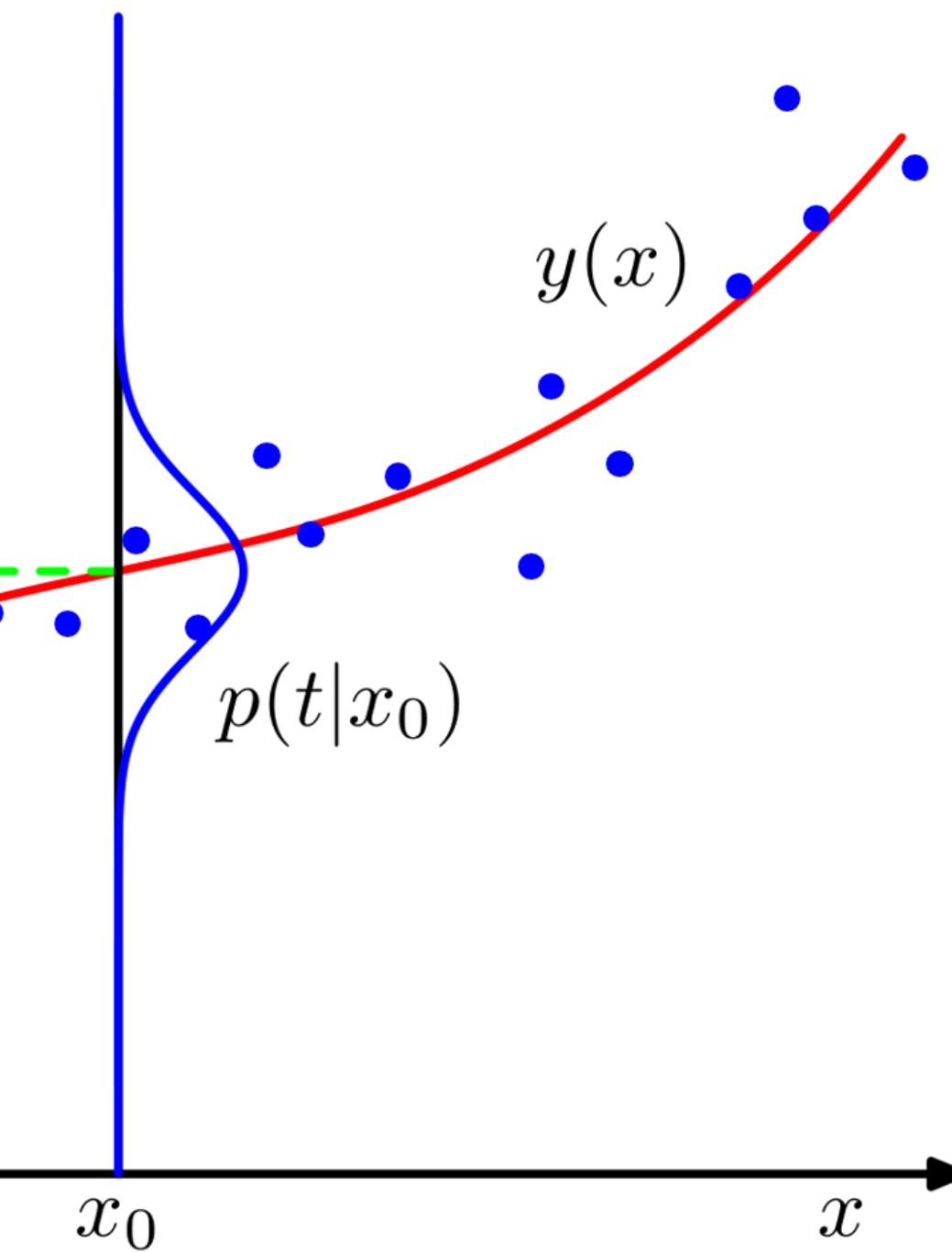
$$y(x_0)$$

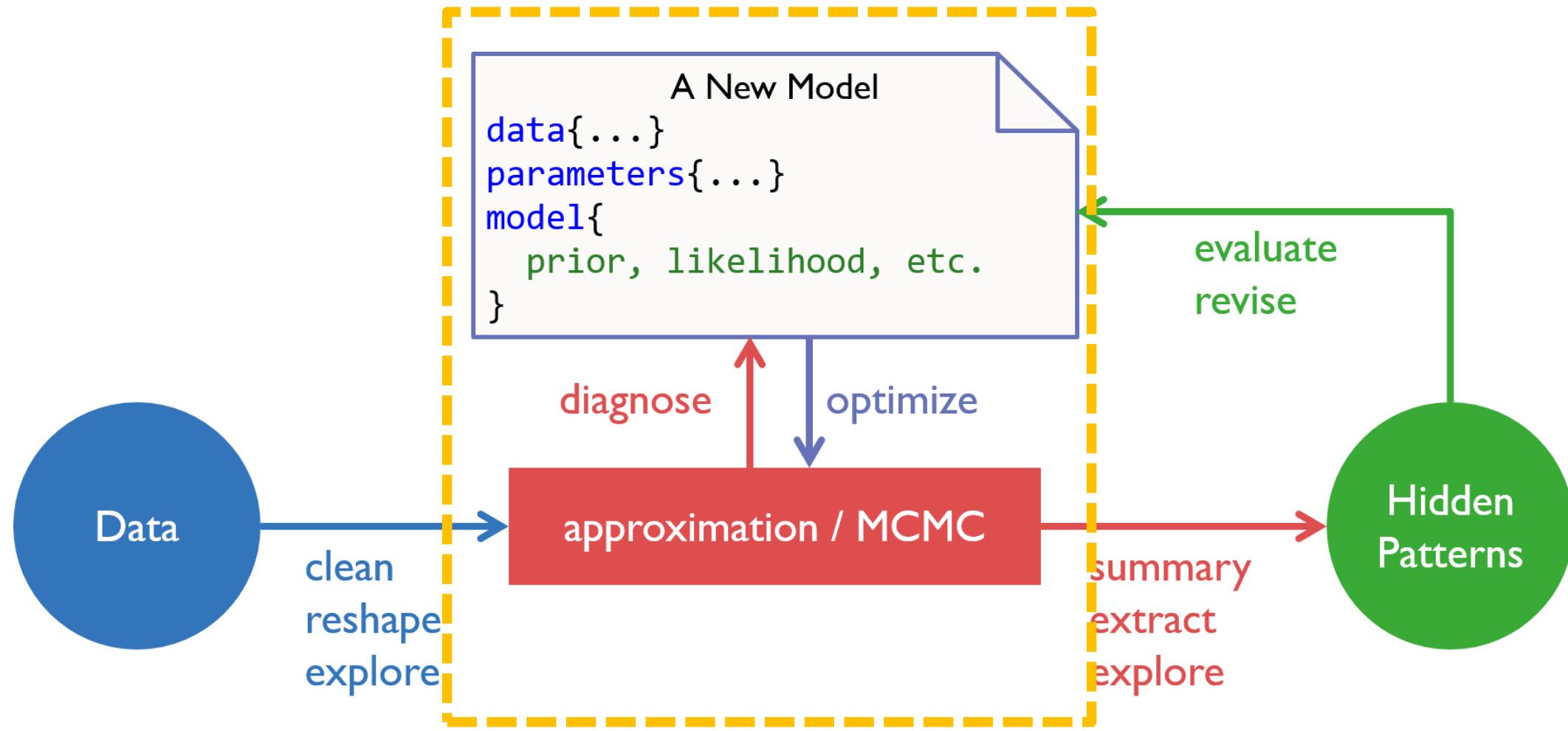
$$x_0$$

$$x$$

$$y(x)$$

$$p(t|x_0)$$



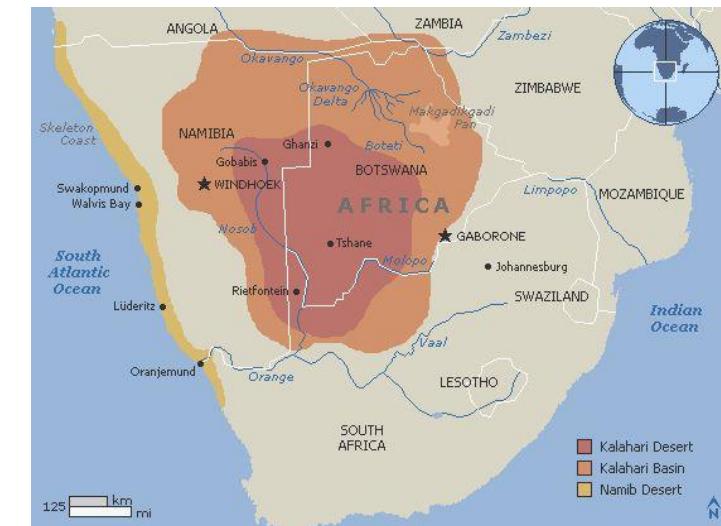


Linear Regression: height ~ weight

cognitive model
statistics
computing

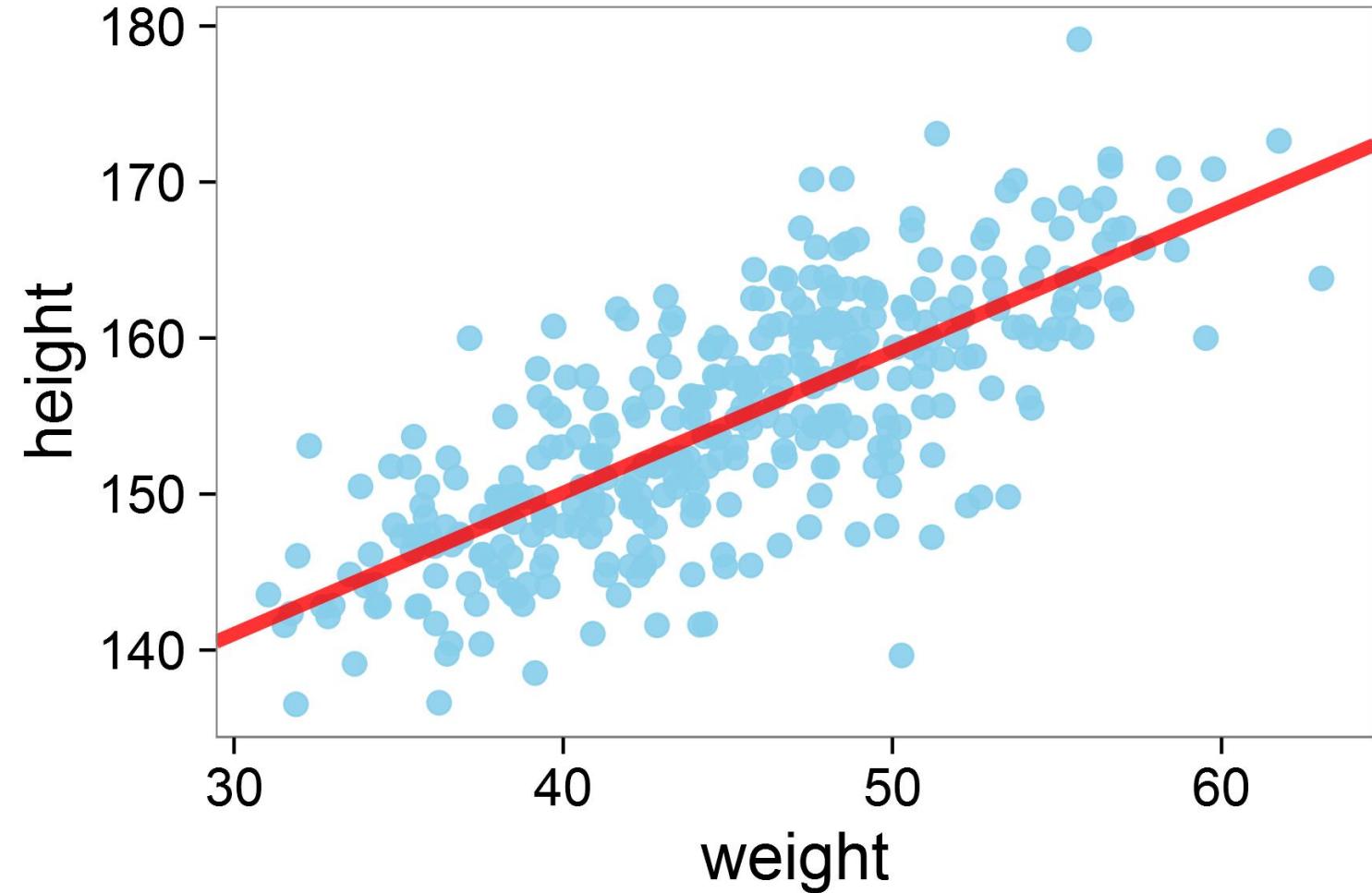
.../BayesCog/04.regression_height/_scripts/regression_height_main.R

make scatter plot and fit the model with lm()



height ~ weight

cognitive model
statistics
computing



Rethinking Regression Model

cognitive model
statistics
computing

$$\mu_i = \alpha + \beta x_i$$

~~$$y_i = \mu_i + \varepsilon$$~~

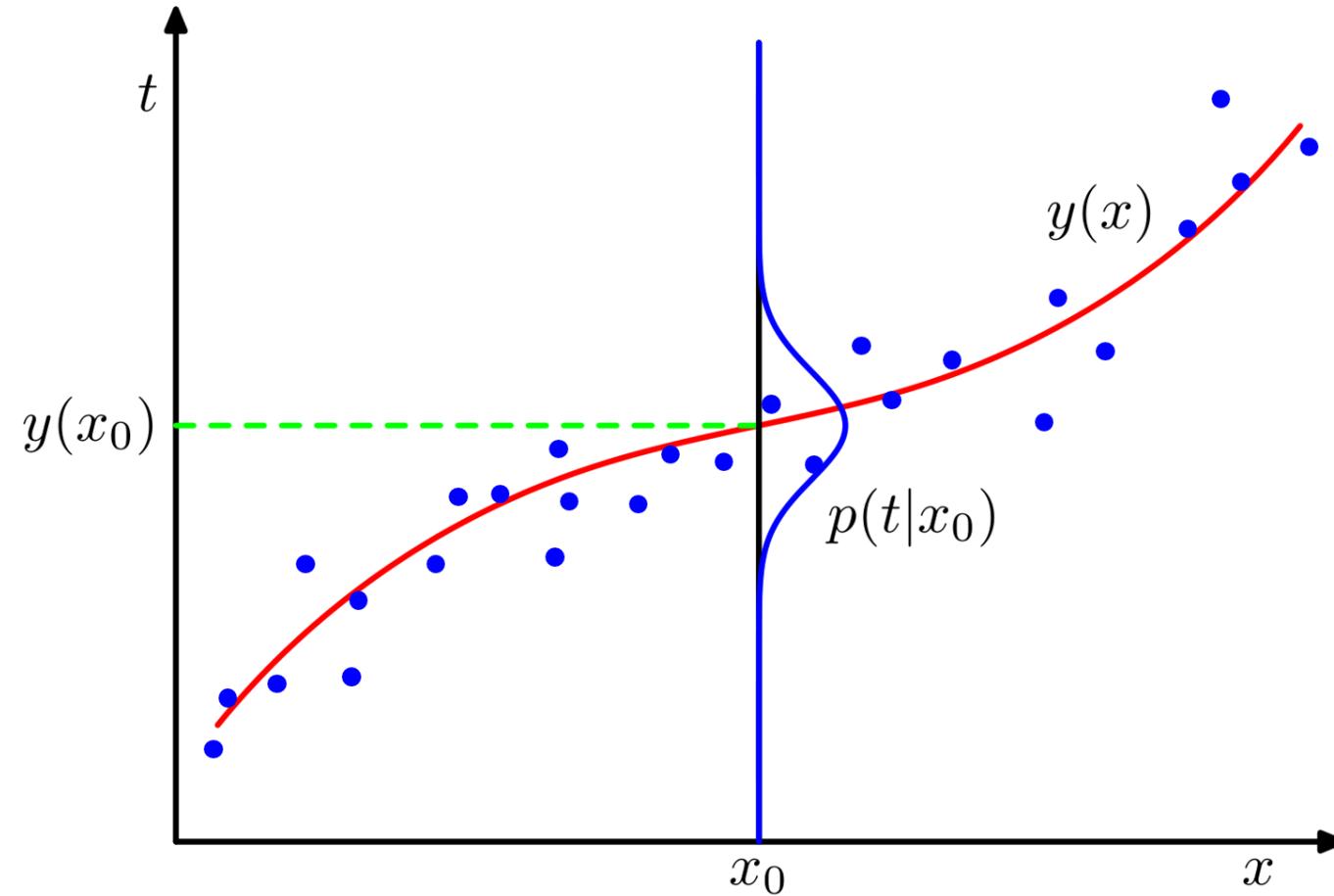
~~$$\varepsilon \sim Normal(0, \sigma)$$~~

$$y_i \sim Normal(\mu_i, \sigma)$$

Rethinking Regression Model

cognitive model
statistics
computing

$$\mu_i = \alpha + \beta x_i$$
$$y_i \sim Normal(\mu_i, \sigma)$$

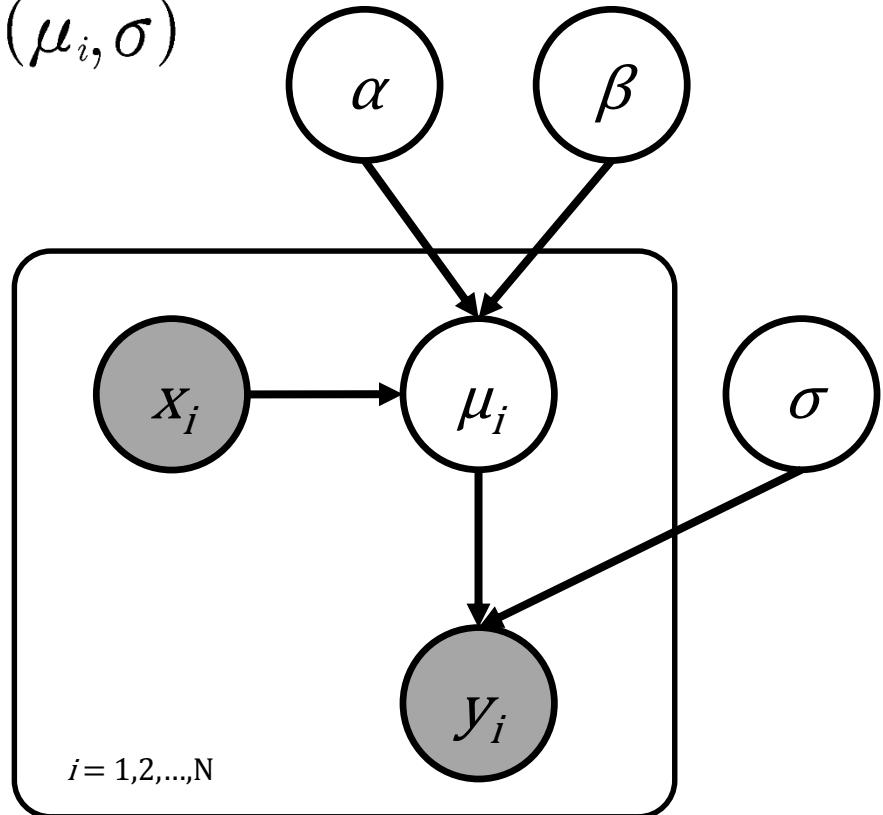


Rethinking Regression Model

cognitive model
statistics
computing

$$\mu_i = \alpha + \beta x_i$$

$$y_i \sim Normal(\mu_i, \sigma)$$



```
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);  
}
```

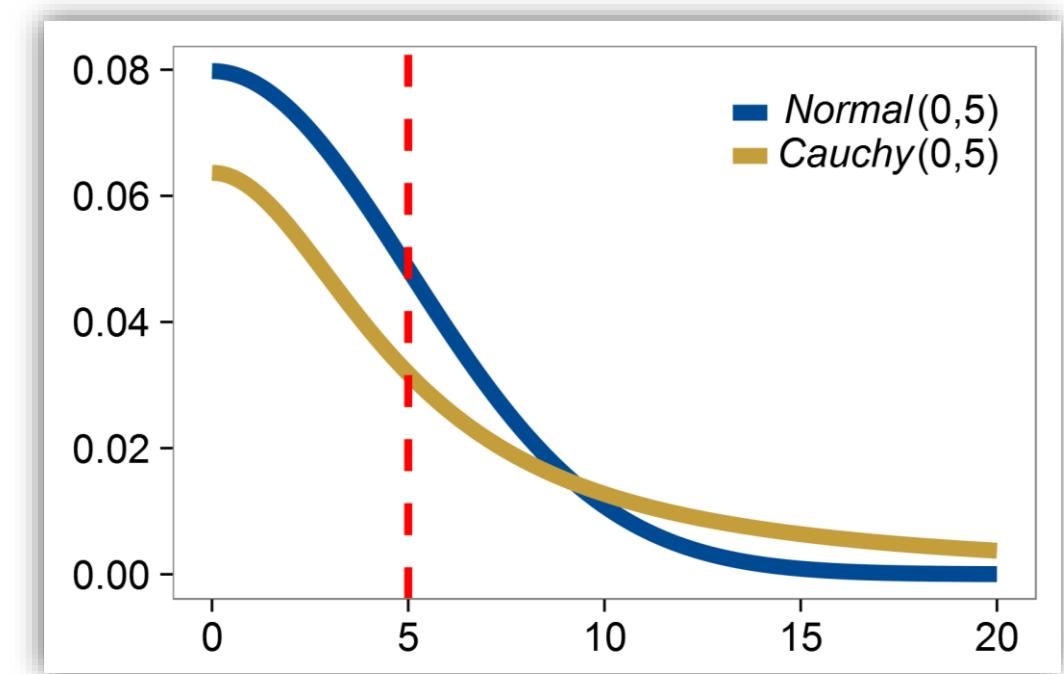
```
model {  
    height ~ normal(alpha + beta * weight, sigma);  
}
```

Thinking about Priors?

 $\alpha \sim Normal(170, 100)$
 $\beta \sim Normal(0, 20)$

$$\overline{\text{height}} = \alpha + \beta * \text{weight}$$

 $\sigma \sim halfCauchy(0, 20)$

$$\text{height} \sim Normal(\overline{\text{height}}, \sigma)$$


Exercise II

cognitive model
statistics
computing

.../BayesCog/04.regression_height/_scripts/regression_height_main.R

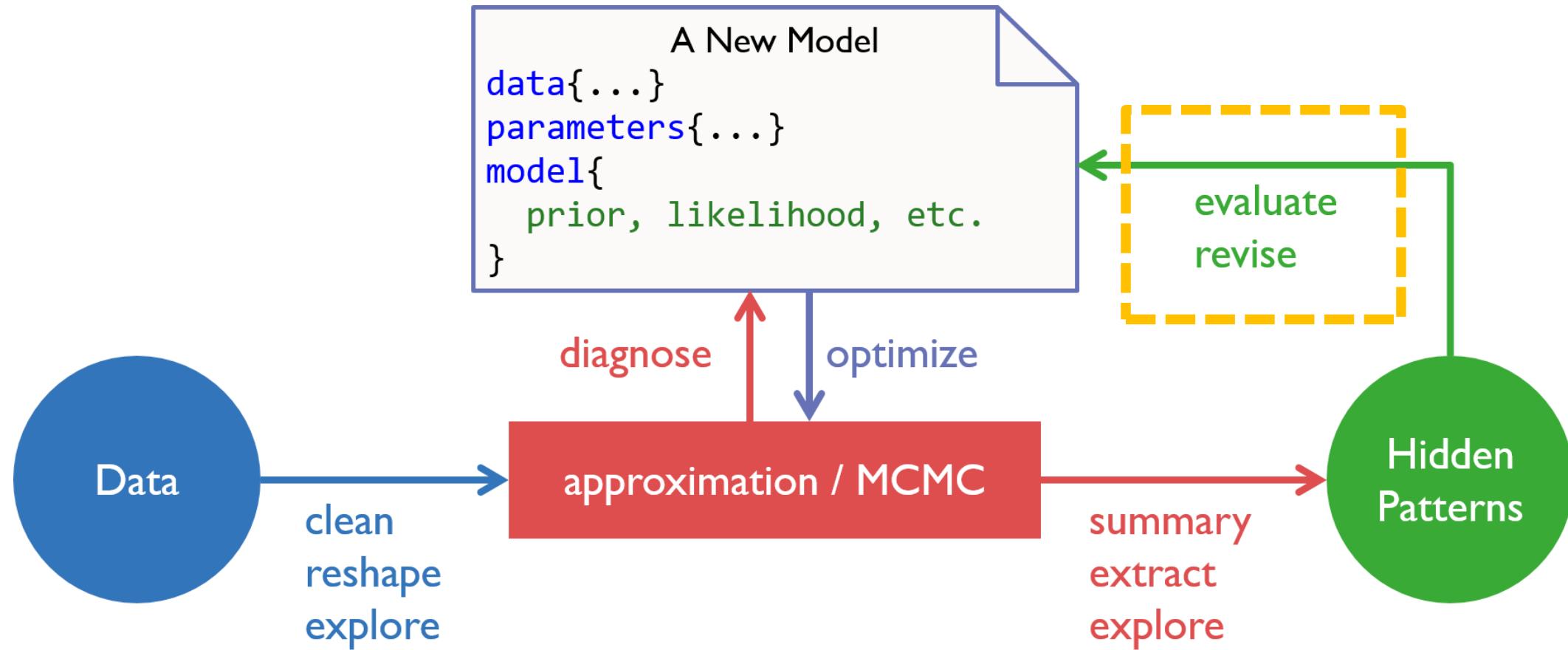
TASK: estimate the model and replicate the results below

Inference for Stan model: regression_height_model.

4 chains, each with iter=2000; warmup=1000; thin=1;

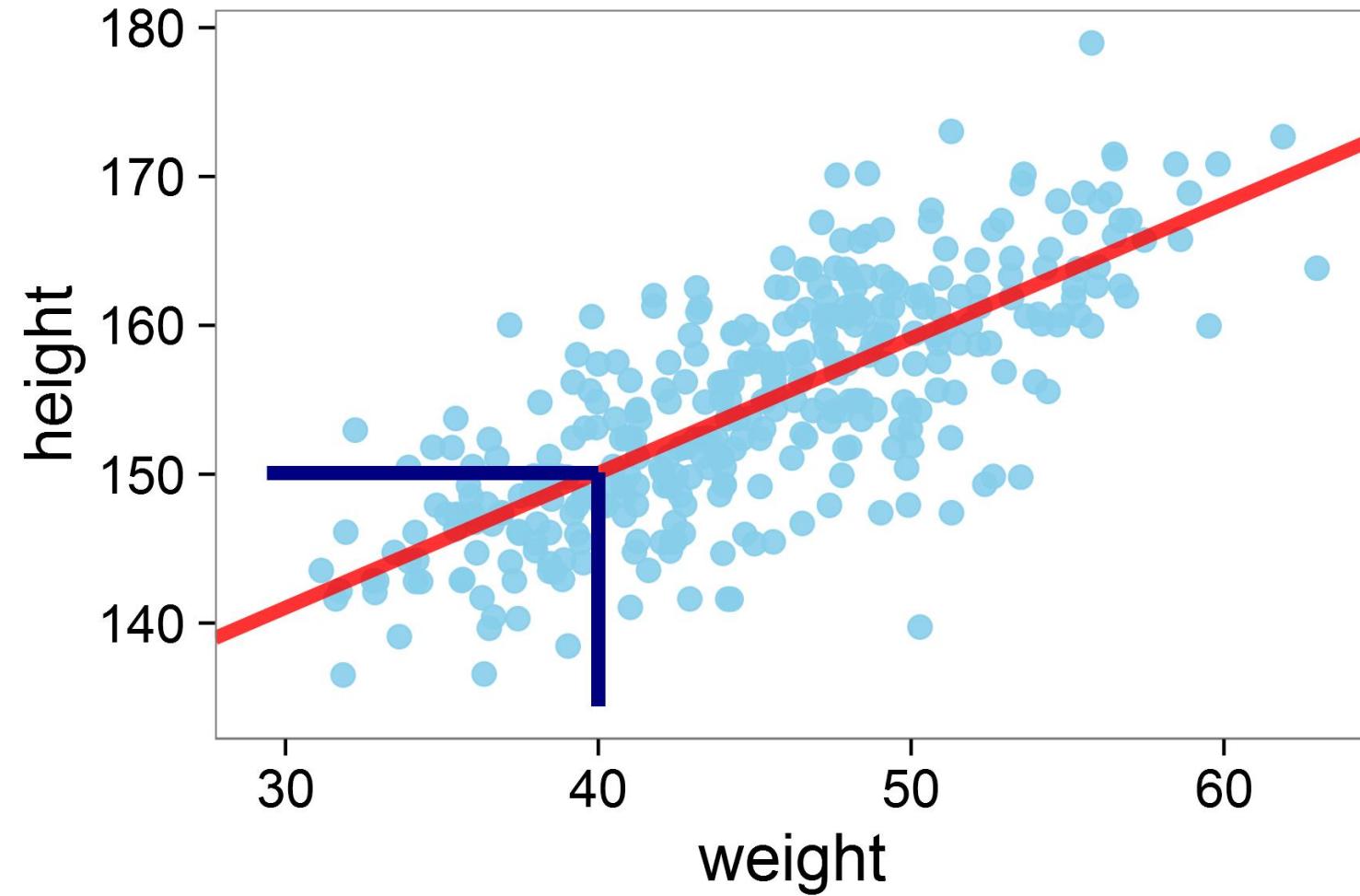
post-warmup draws per chain=1000, total post-warmup draws=4000.

	mean	se_mean	sd	2.5%	25%	50%	75%	97.5%	n_eff	Rhat
alpha	113.97	0.06	1.86	110.27	112.76	113.93	115.20	117.66	934	1
beta	0.90	0.00	0.04	0.82	0.88	0.90	0.93	0.99	922	1
sigma	5.11	0.01	0.19	4.74	4.97	5.10	5.24	5.50	1437	1
lp__	-747.61	0.04	1.23	-750.80	-748.15	-747.28	-746.72	-746.24	993	1



What does the Model Predict?

cognitive model
statistics
computing

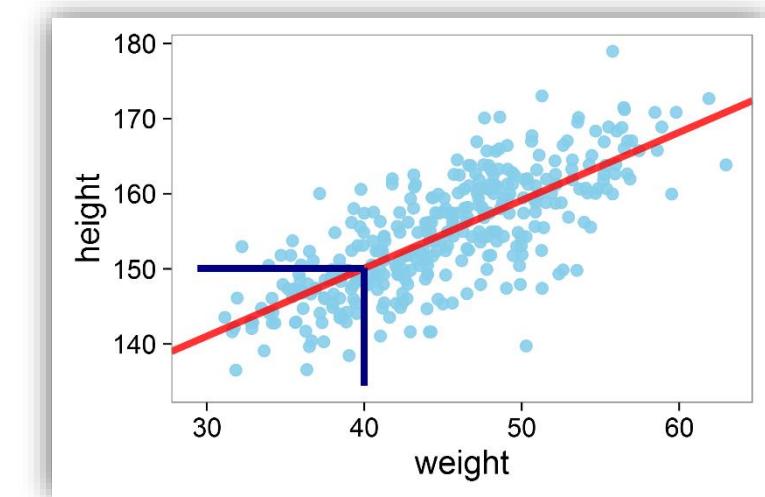


Posterior Predictive Check (PPC)

cognitive model
statistics
computing

```
generated quantities {  
    vector[N] height_bar;  
    for (n in 1:N) {  
        height_bar[n] = normal_rng(alpha + beta * weight[n], sigma);  
    }  
}
```

the generated quantities
block runs only AFTER the
sampling, and the time it costs
can be essentially ignored!



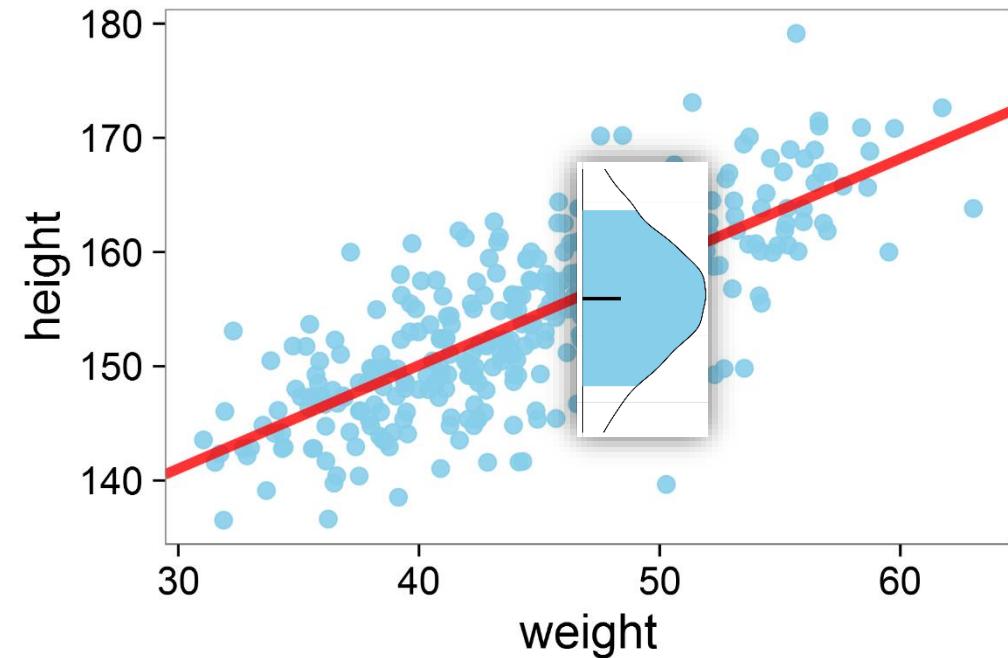
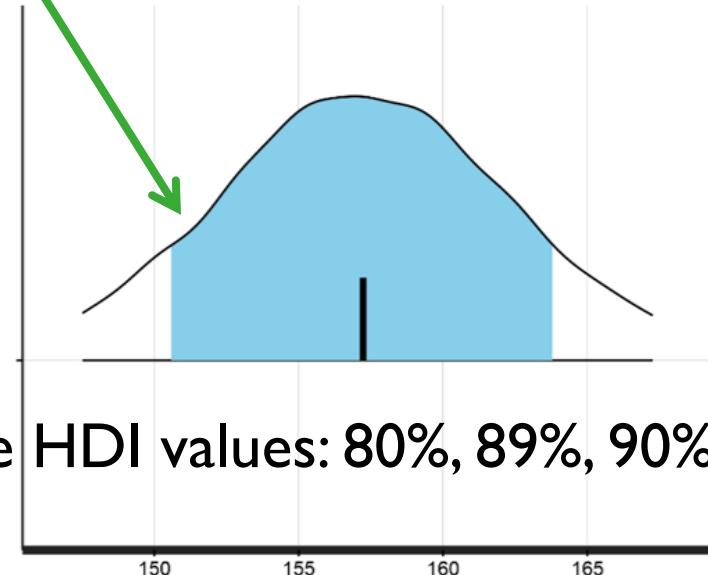
Posterior Predictive Check (PPC)

cognitive model
statistics
computing

Highest density interval (HDI)

`dens(height_bar | x=47.8)`

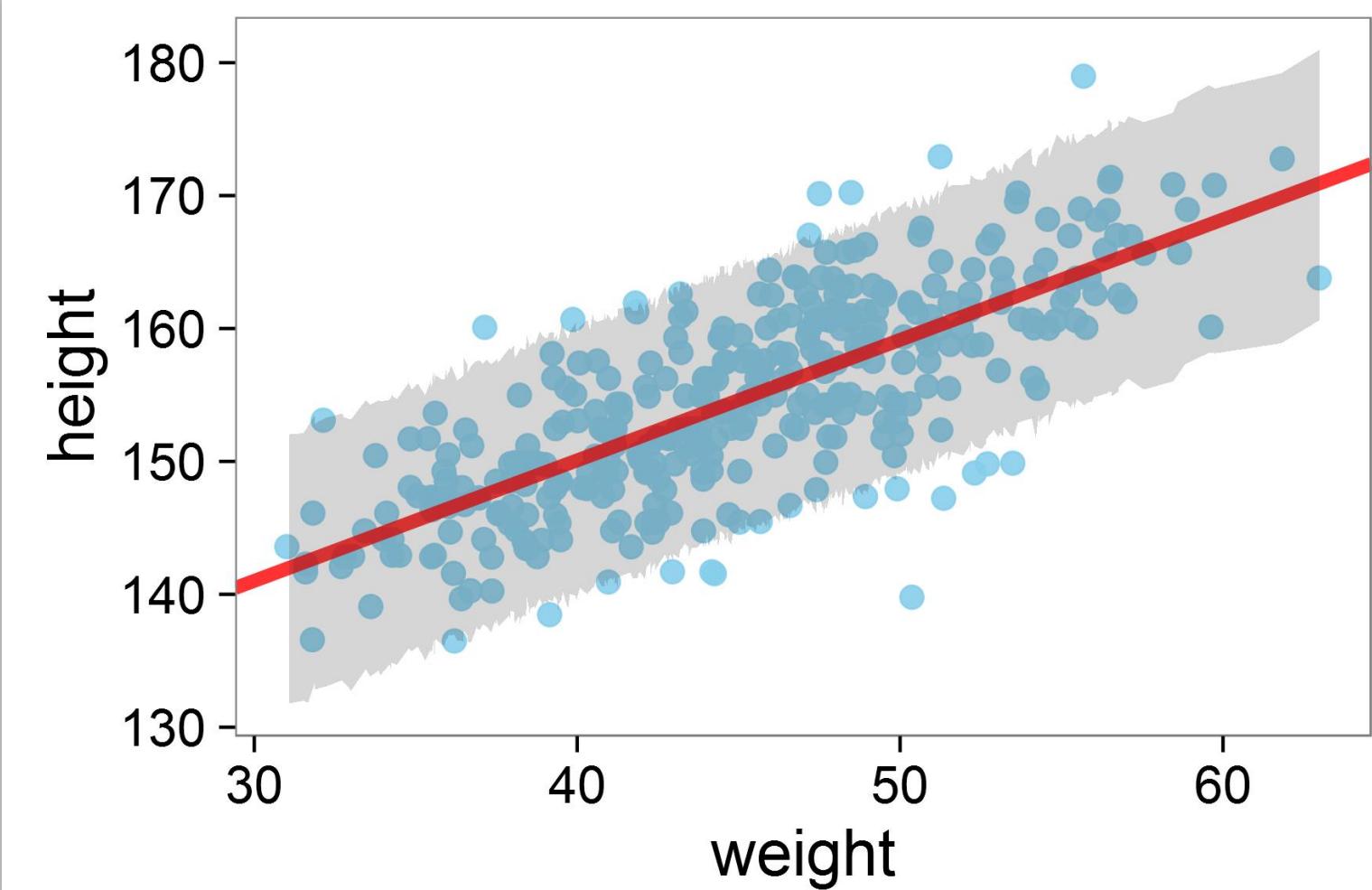
possible HDI values: 80%, 89%, 90%, 95%



```
height_bar <- extract(fit_reg_ppc, pars = 'height_bar',
                      permuted = FALSE)$height_bar
height_HDI <- apply(height_bar, 2, HDIofMCMC)
```

Posterior Predictive Check (PPC)

cognitive model
statistics
computing

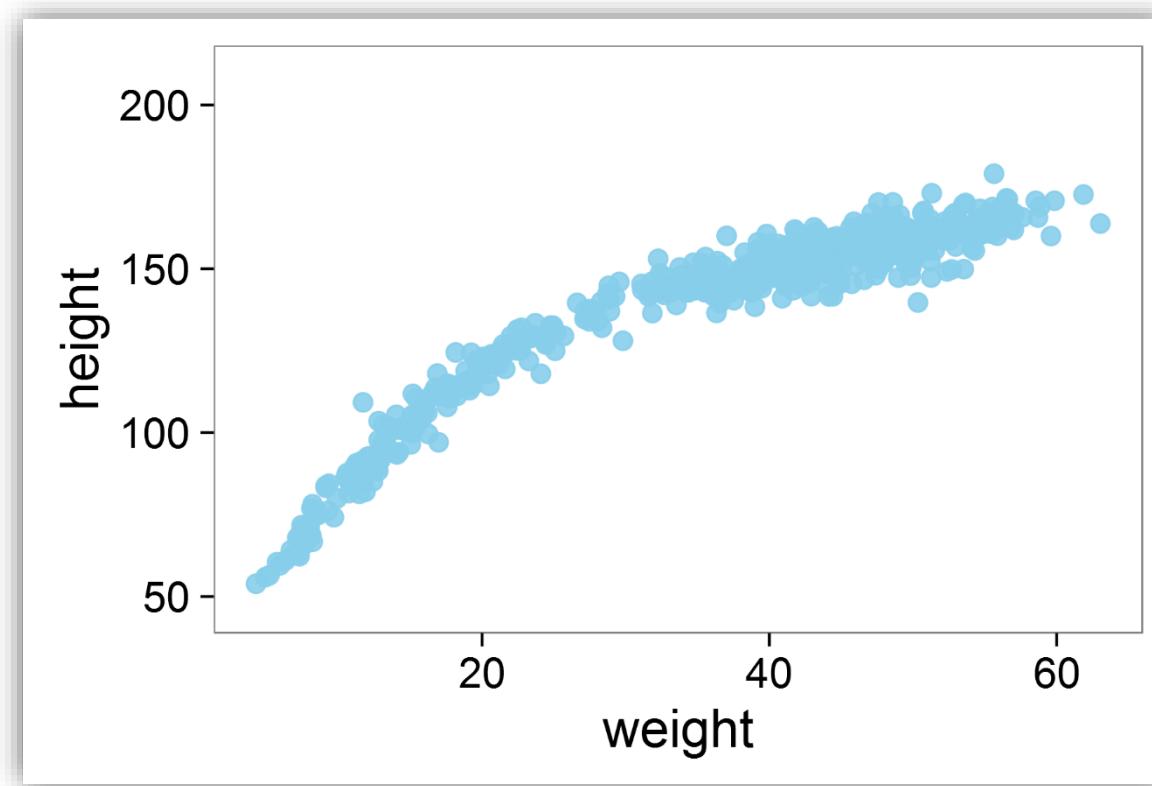


Exercise III

cognitive model
statistics
computing

```
.../BayesCog/05.regression_height_poly/_scripts  
/regression_height_poly_main.R
```

TASK: produce PPC plot for both 1st order and 2nd order polynomial fit



Exercise III – Tips

cognitive model
statistics
computing

```
> source('_scripts/regression_height_poly_main.R')  
  
> out1 <- reg_poly(poly_order = 1)
```

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

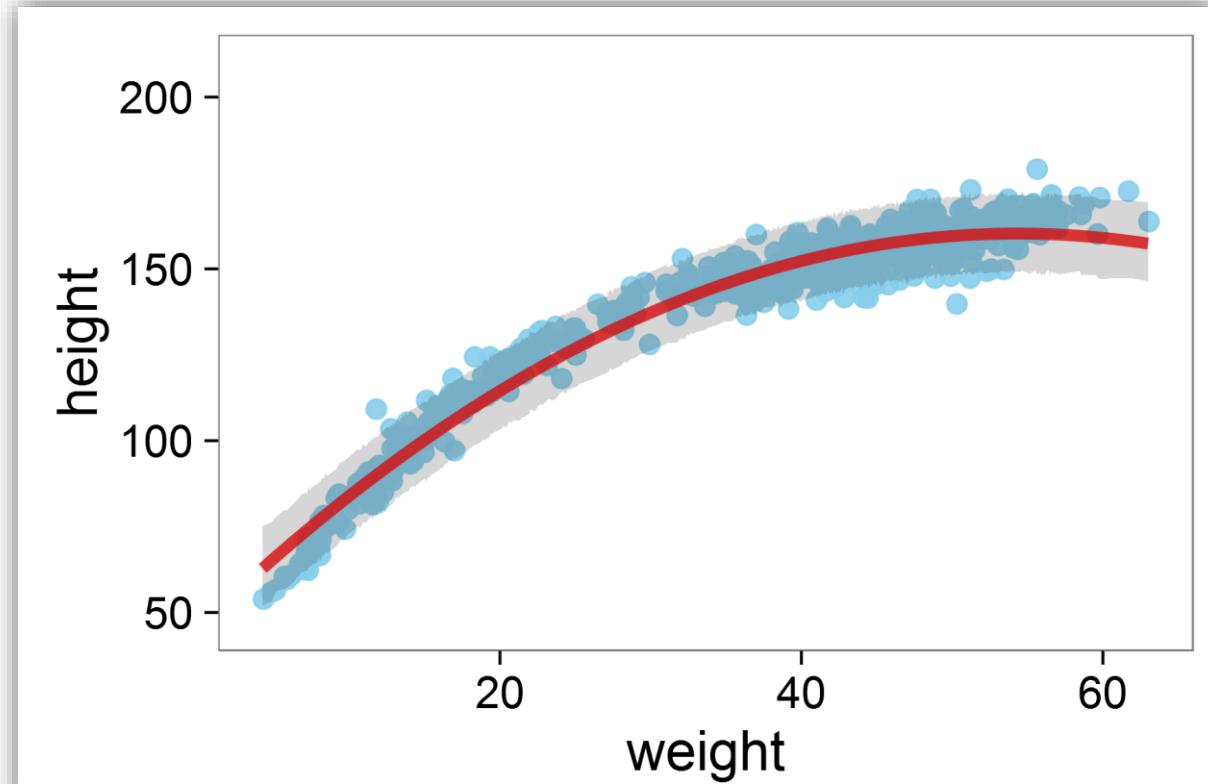
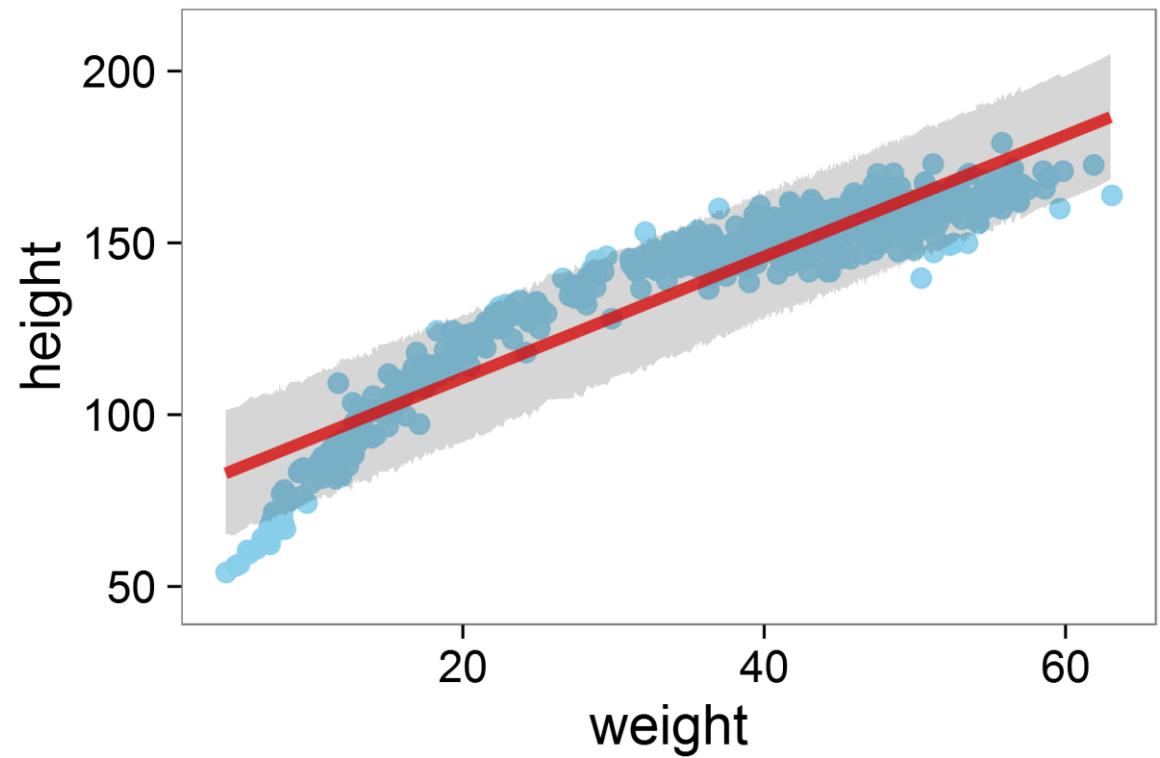
$$\text{height} \sim \text{Normal}(\overline{\text{height}}, \sigma)$$

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

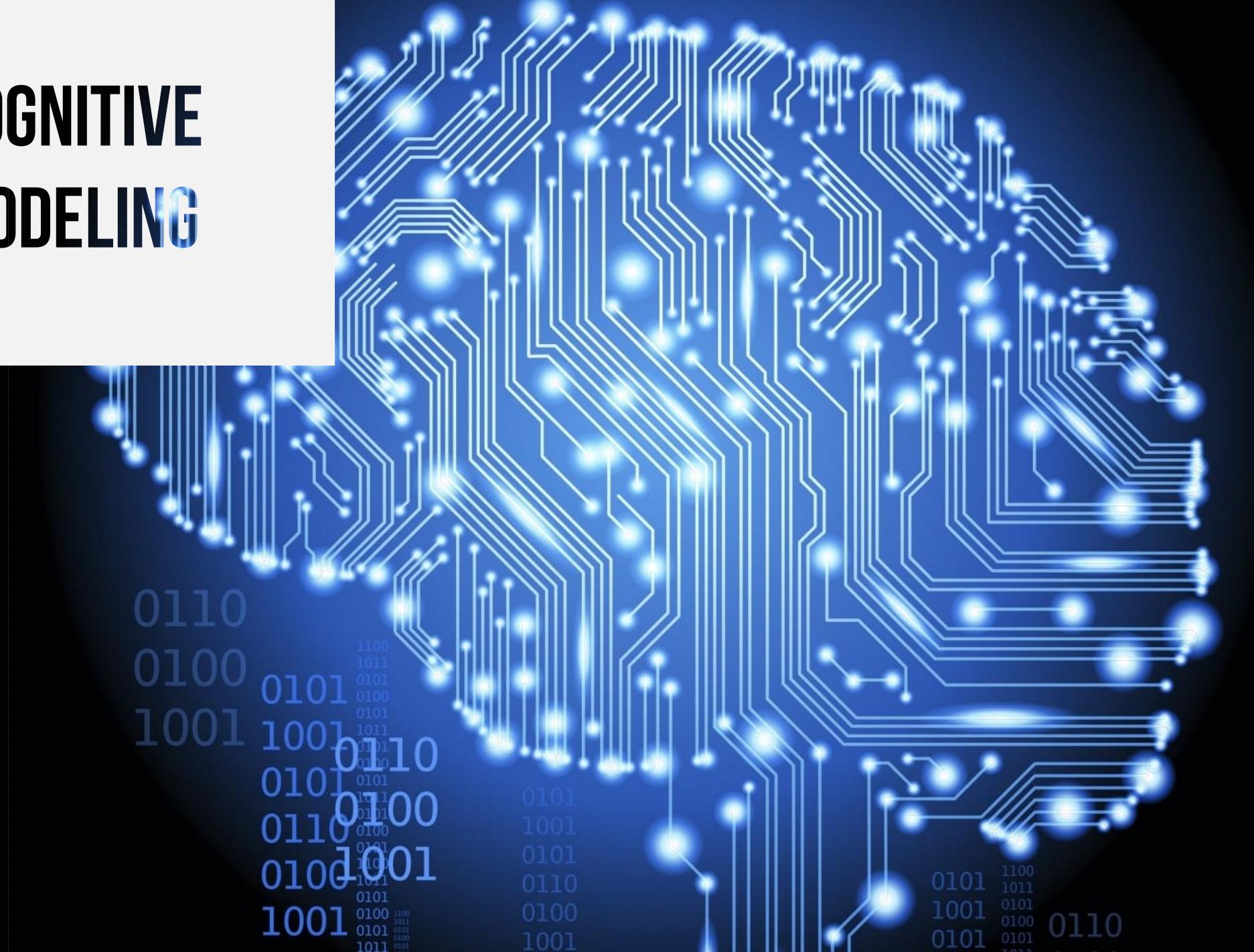
```
height ~ normal(alpha + beta1 * weight + beta2 * weight_sq, sigma);
```

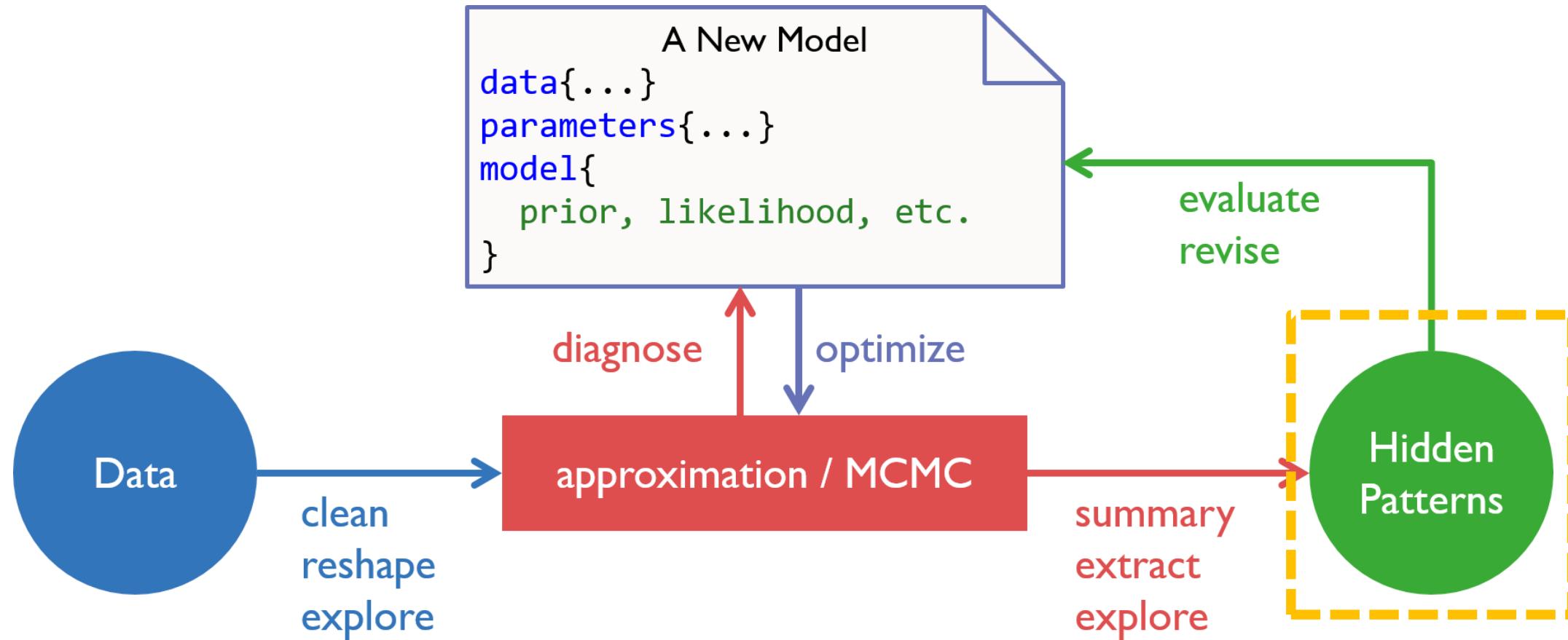
Exercise III – output2

cognitive model
statistics
computing



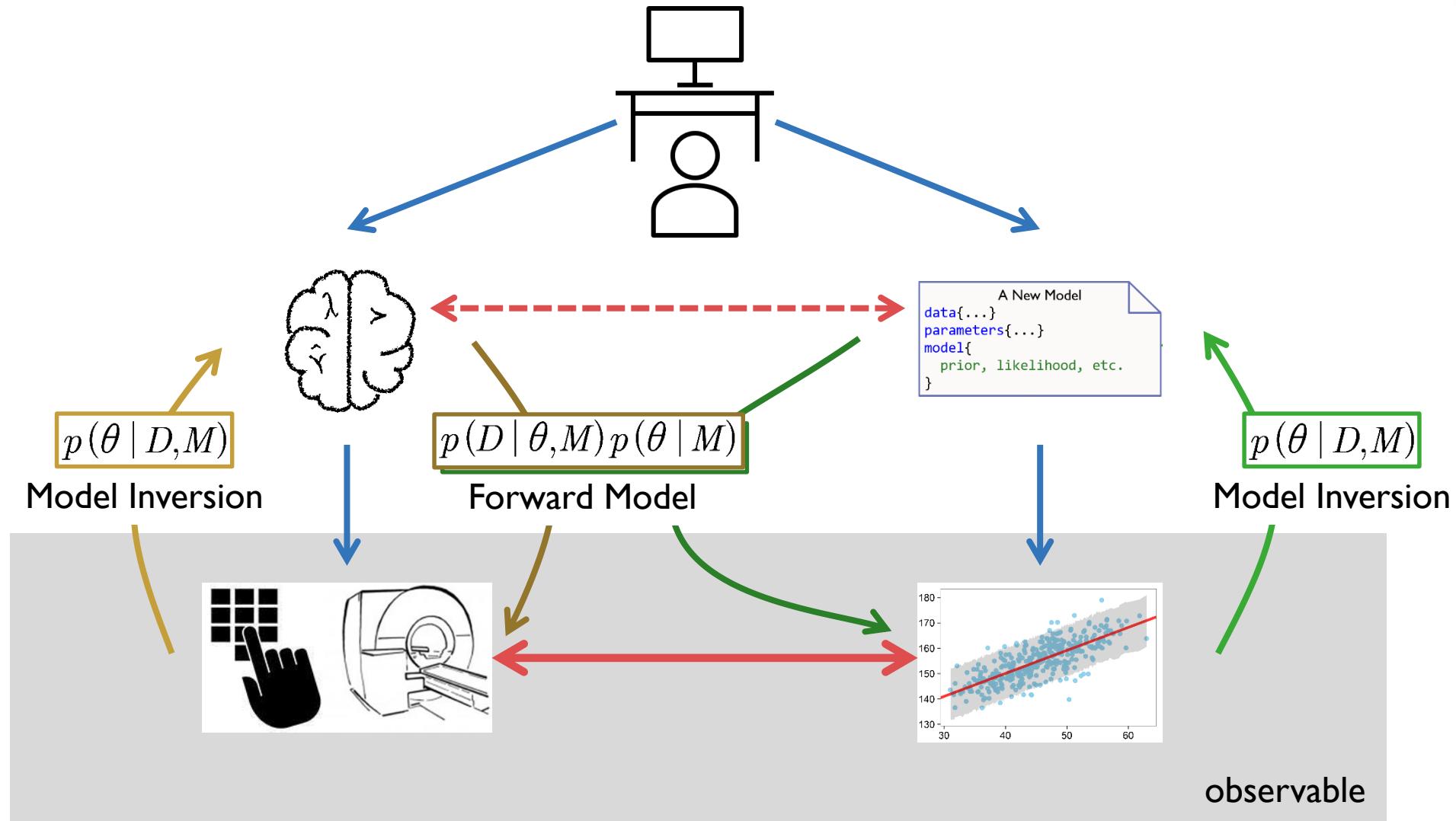
COGNITIVE MODELING





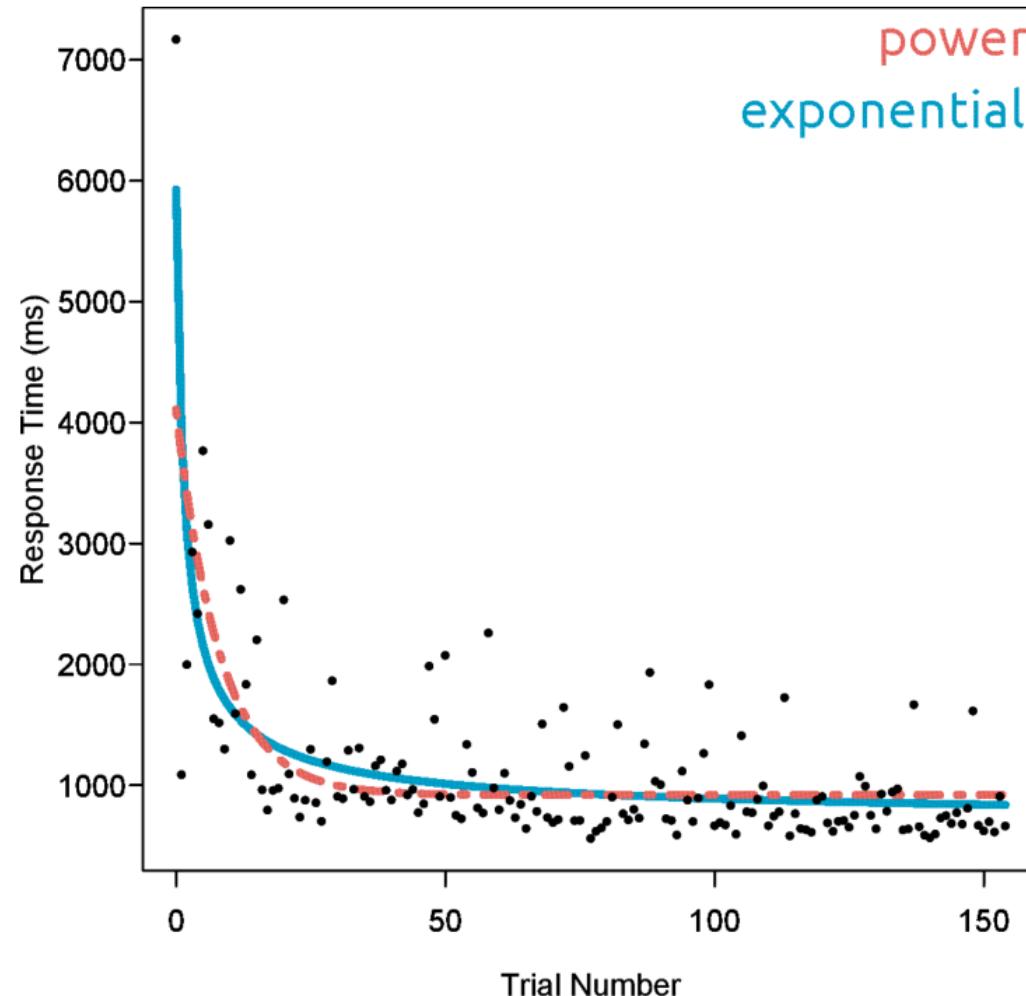
What is Cognitive Modeling?

cognitive model
statistics
computing



Psychological Implications

cognitive model
statistics
computing



decreasing learning rate

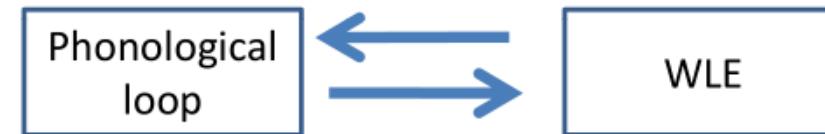
$$RT = N^{-\beta}$$

$$RT = e^{-\alpha N}$$

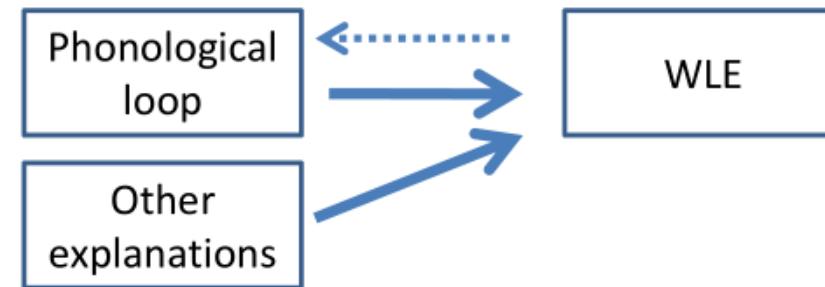
constant learning rate

Mapping Model onto Cognitive Process?

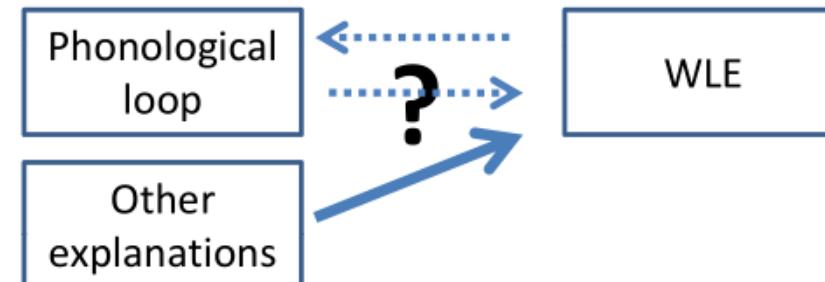
a



b



c



WLE =
word length effect

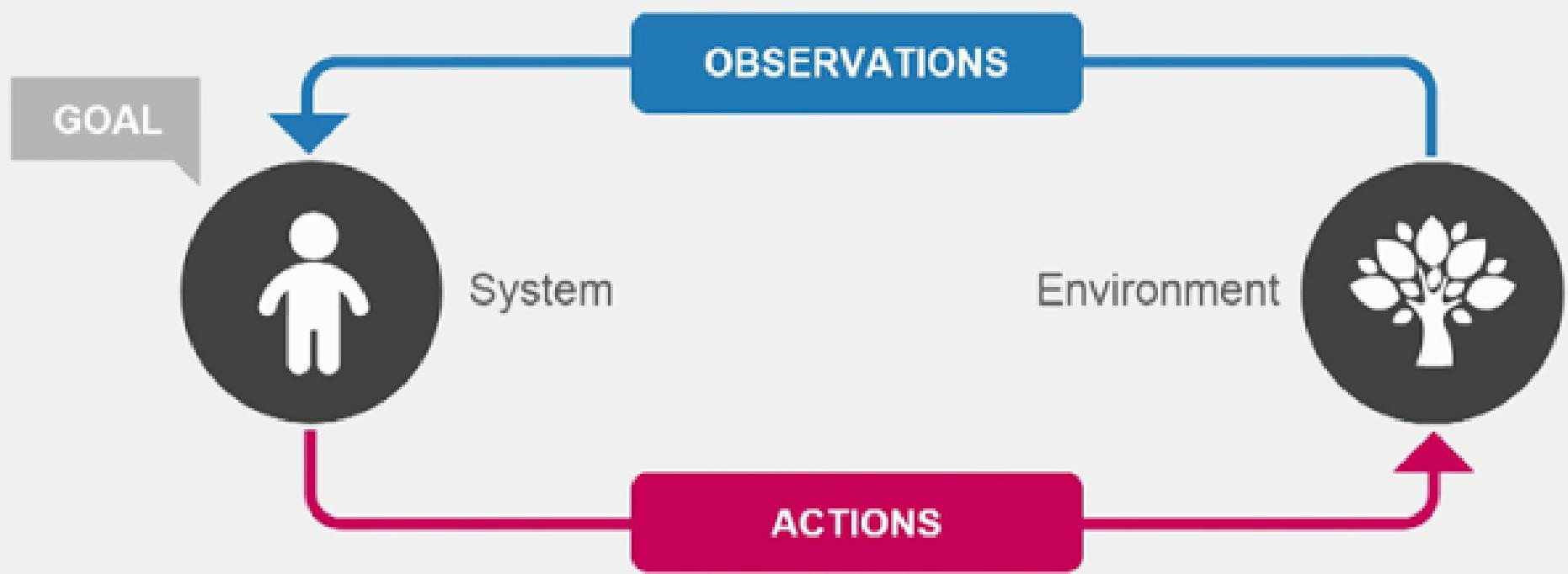
Essentially, all the models are wrong, but some are useful.

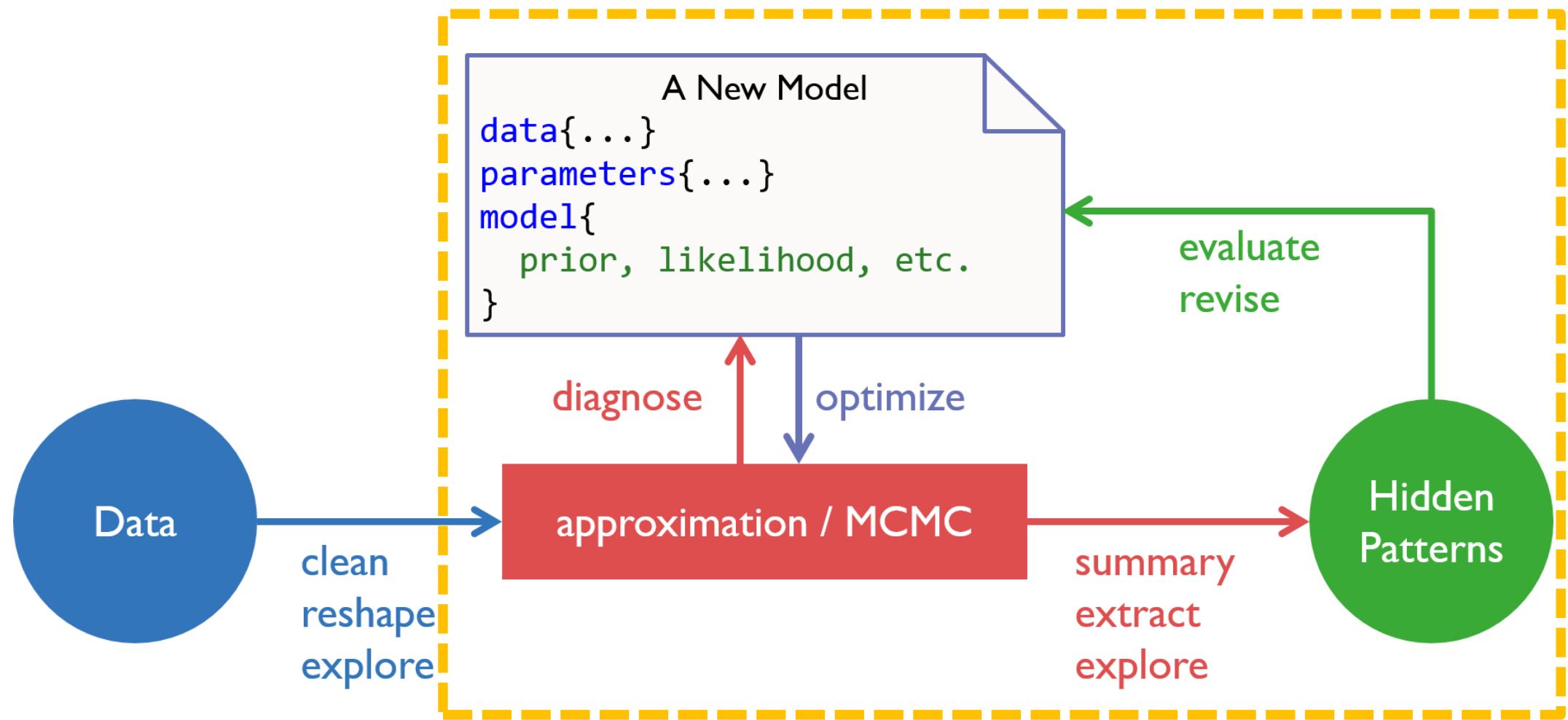


– George E. P. Box

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

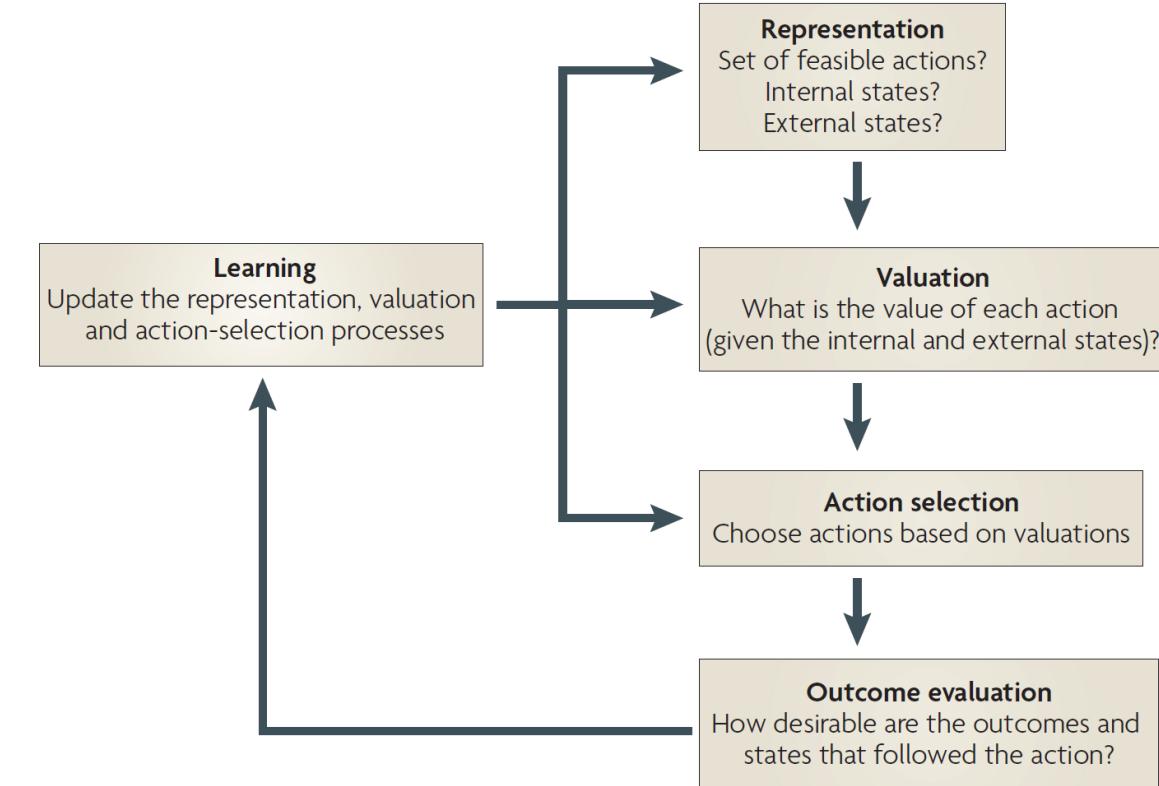
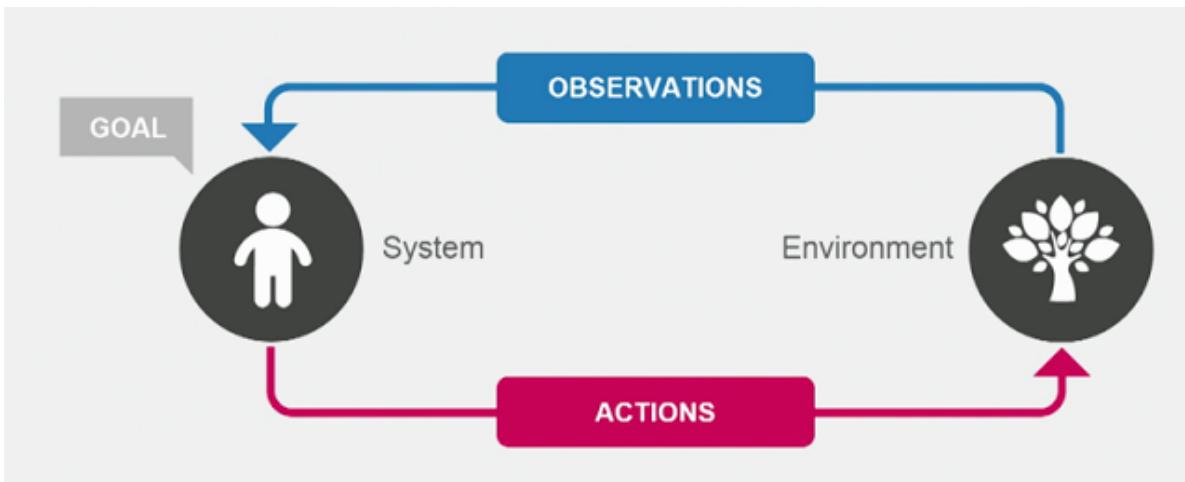
REINFORCEMENT LEARNING FRAMEWORK



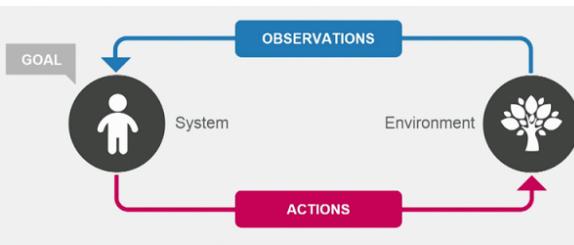


Reinforcement Learning

cognitive model
statistics
computing



Rescorla-Wagner Value Update



Value update:

$$V_{t+1} = V_t + lr * PE$$

Prediction error:

$$PE = R_t - V_t$$

lr - learning rate

PE - reward prediction error

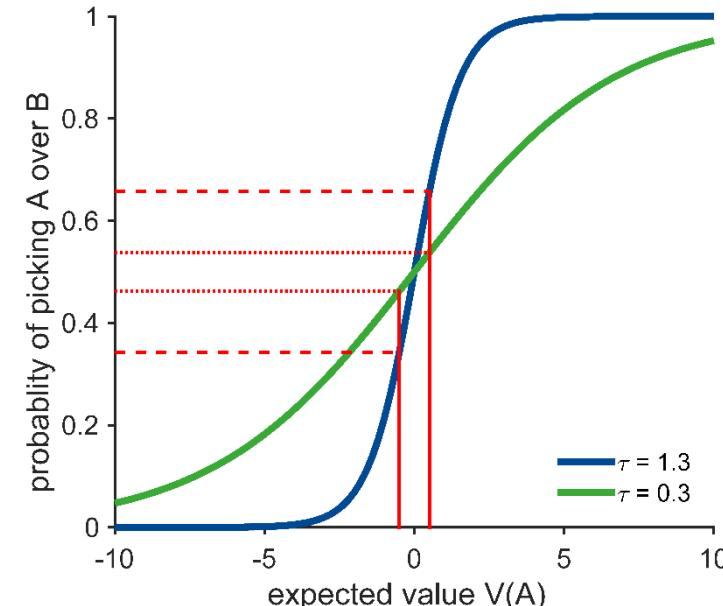
V - value

R - reward

temp - softmax temperature

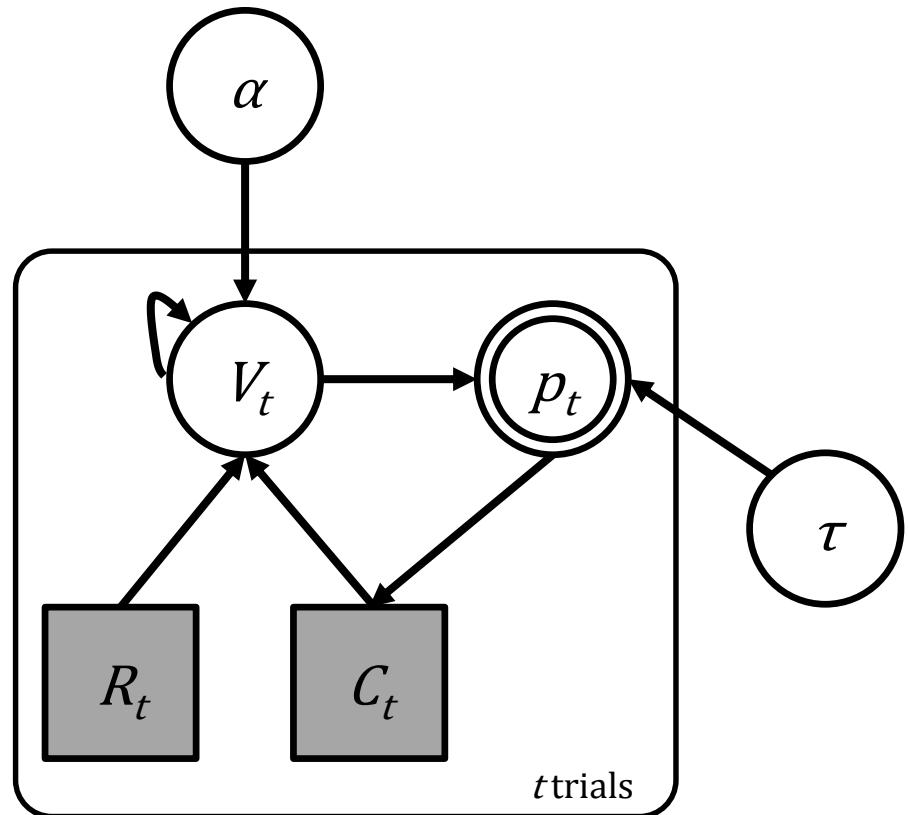
choice rule (sigmoid /softmax):

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



RL – Implementation

cognitive model
statistics
computing



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

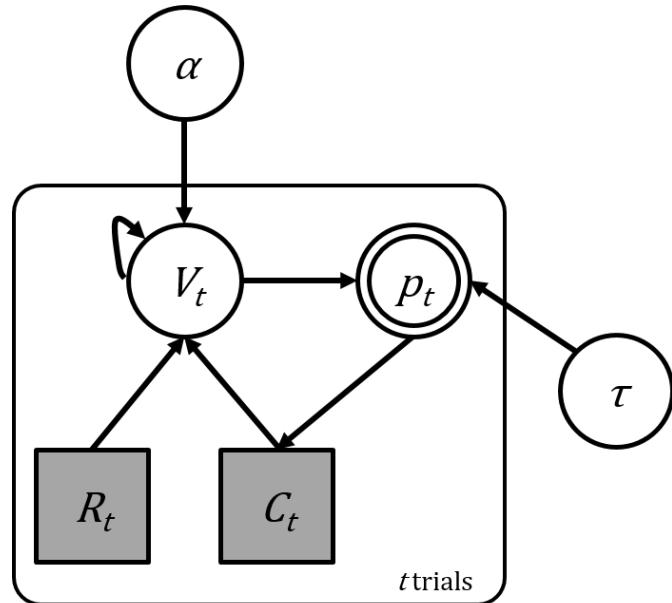
$$\tau \sim Uniform(0, 3)$$

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

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

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;  
    }  
}
```

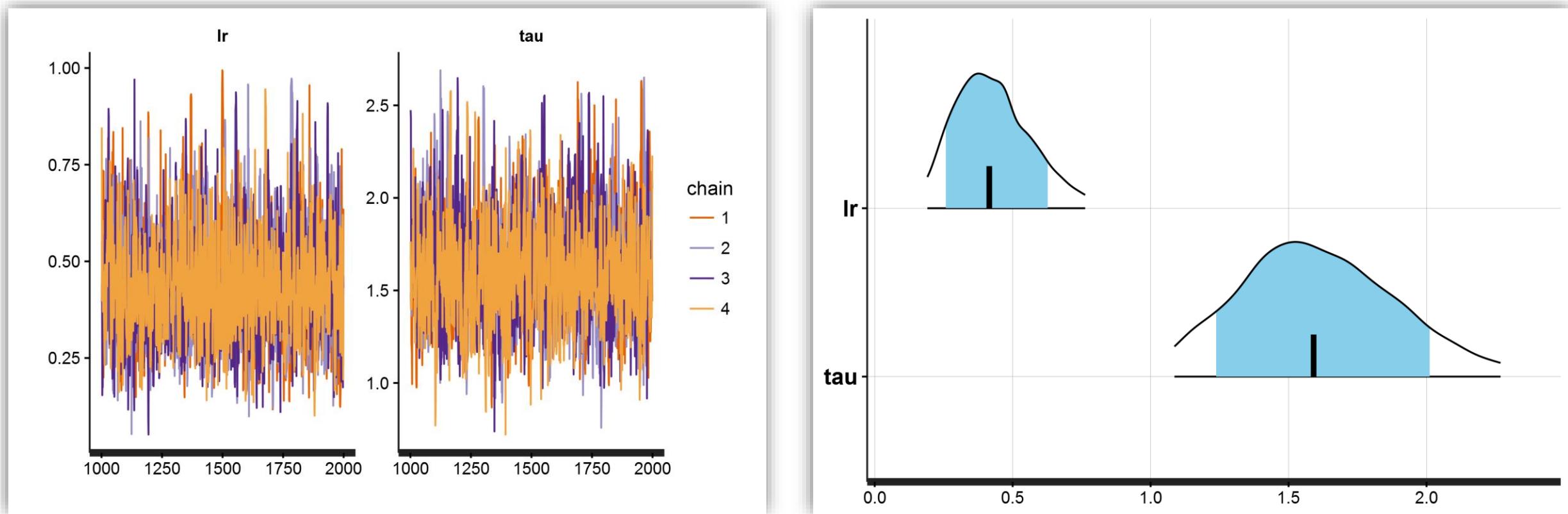
RL – Fitting with Stan

```
.../BayesCog/06.reinforcement_learning/_scripts/reinforcement_learning_single_parm_main.R
```

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

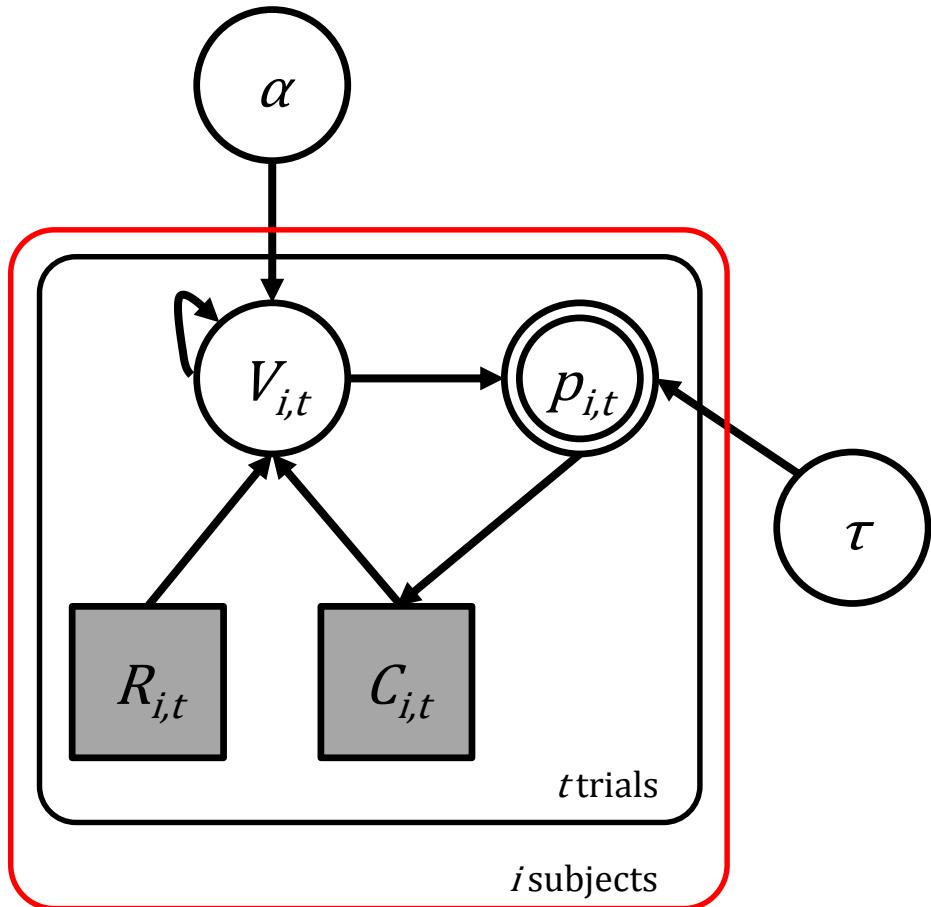
RL - MCMC Output

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 IV

cognitive model
statistics
computing

```
.../BayesCog/06.reinforcement_learning/_scripts/reinforcement_learning_single_
parm_main.R
```

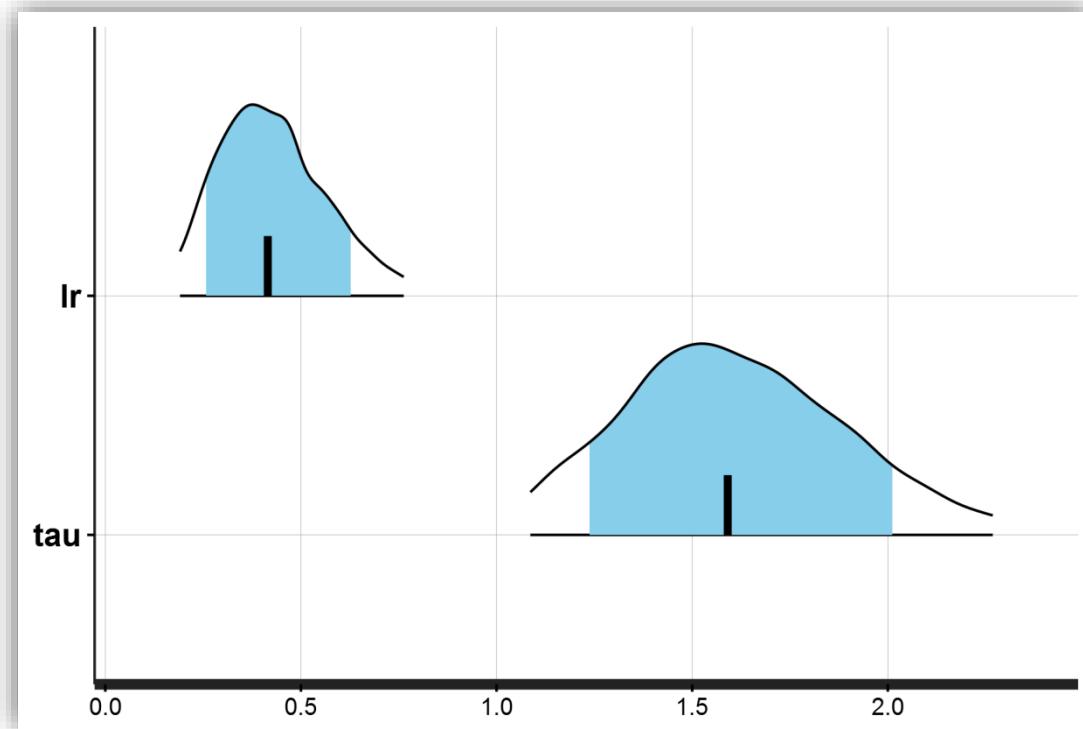
TASK: complete and fit the model for multiple participants (assuming same parameters)

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

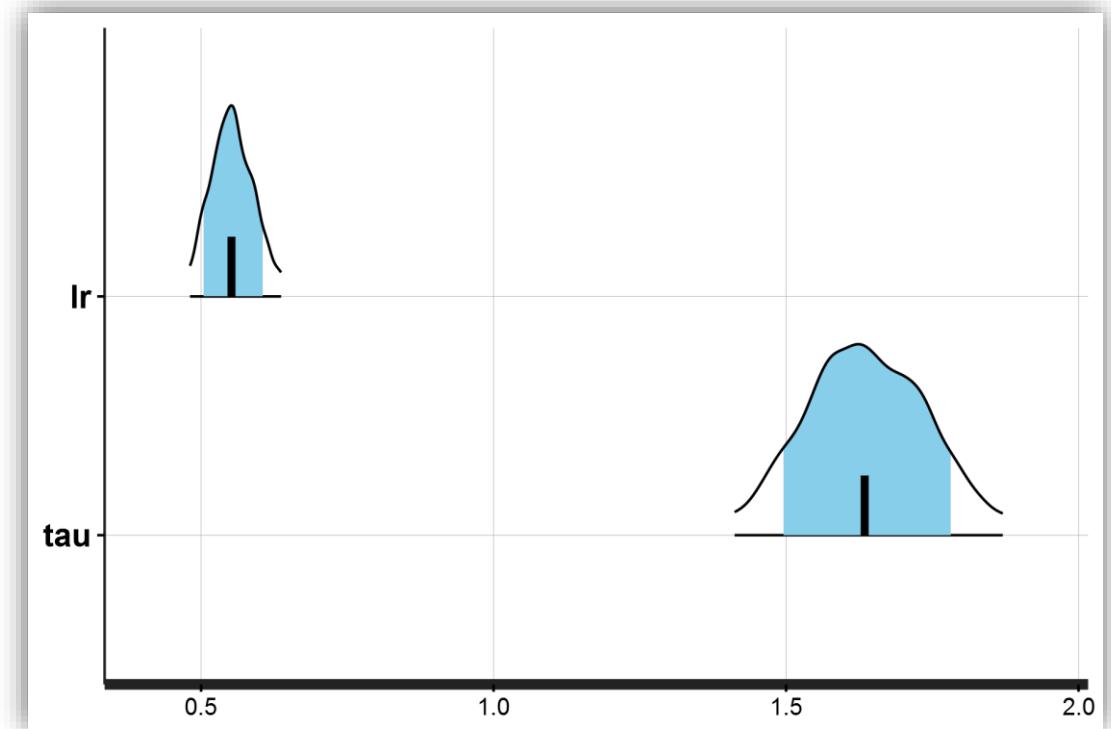
Exercise IV

cognitive model
statistics
computing

$N = 1$

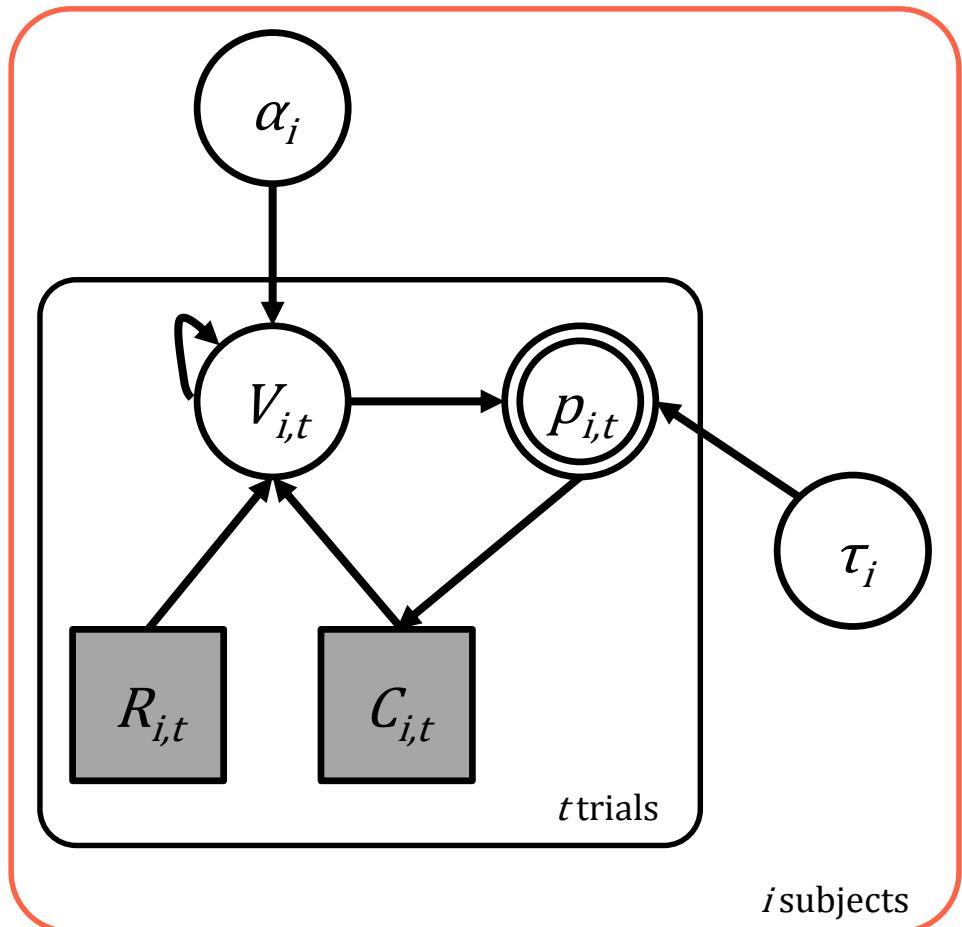


$N = 10$



Fitting Multiple Participants Independently

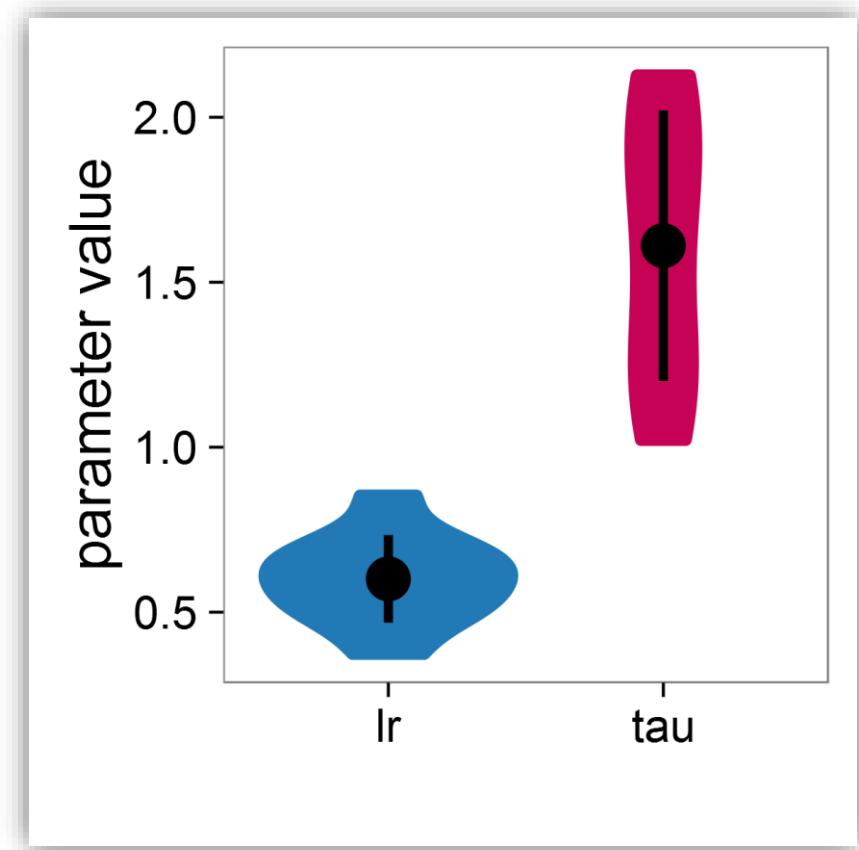
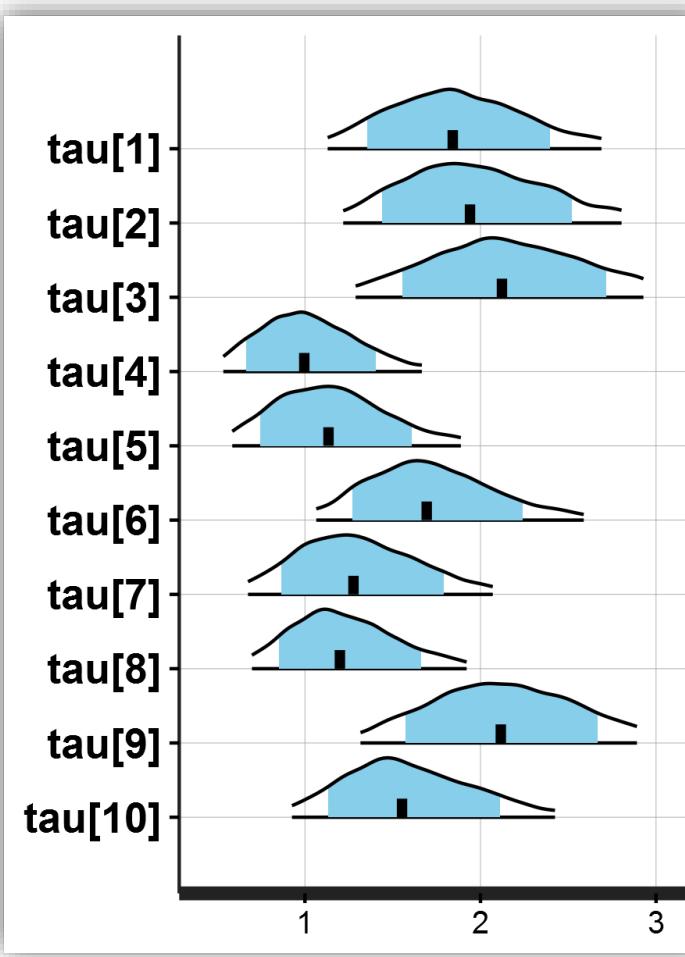
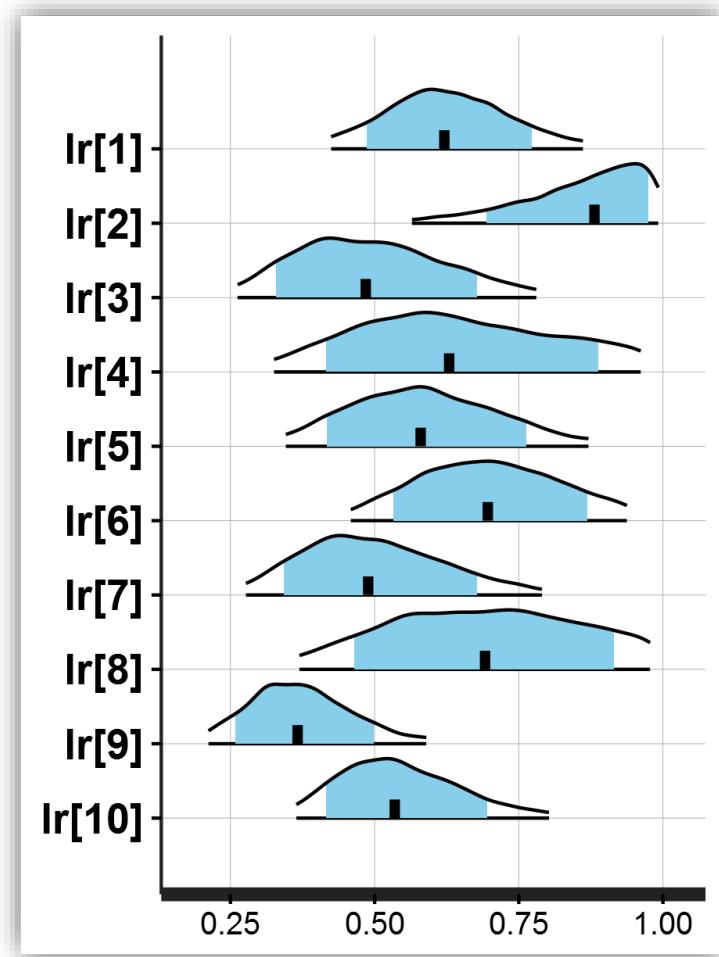
cognitive model
statistics
computing



```
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

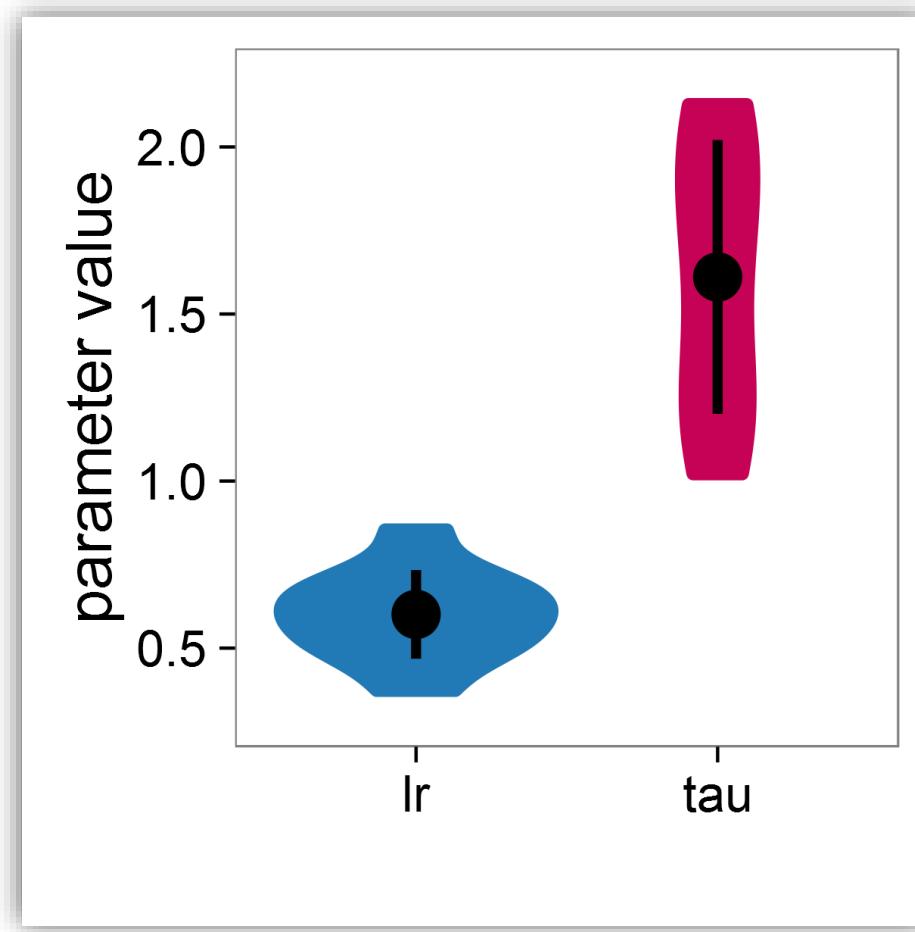
cognitive model
statistics
computing



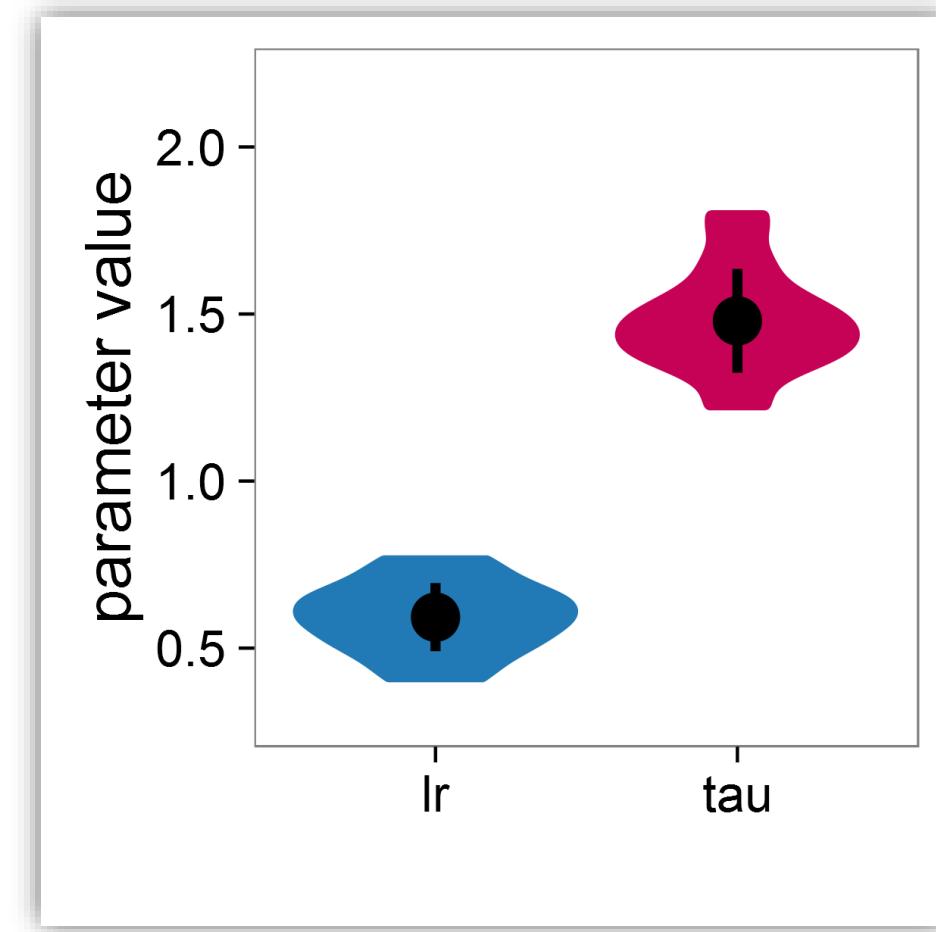
Comparing with True Parameters

cognitive model
statistics
computing

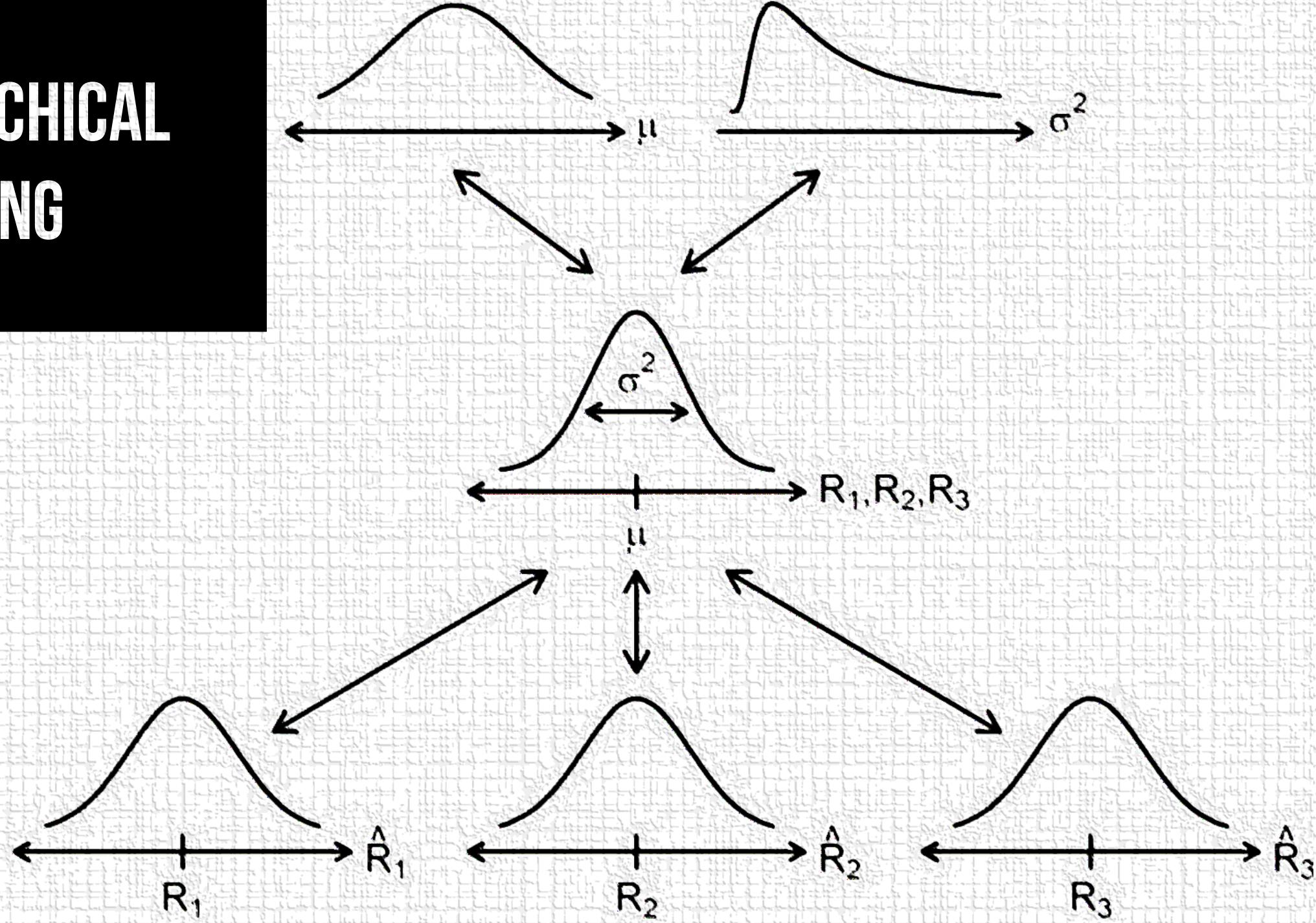
Posterior Means

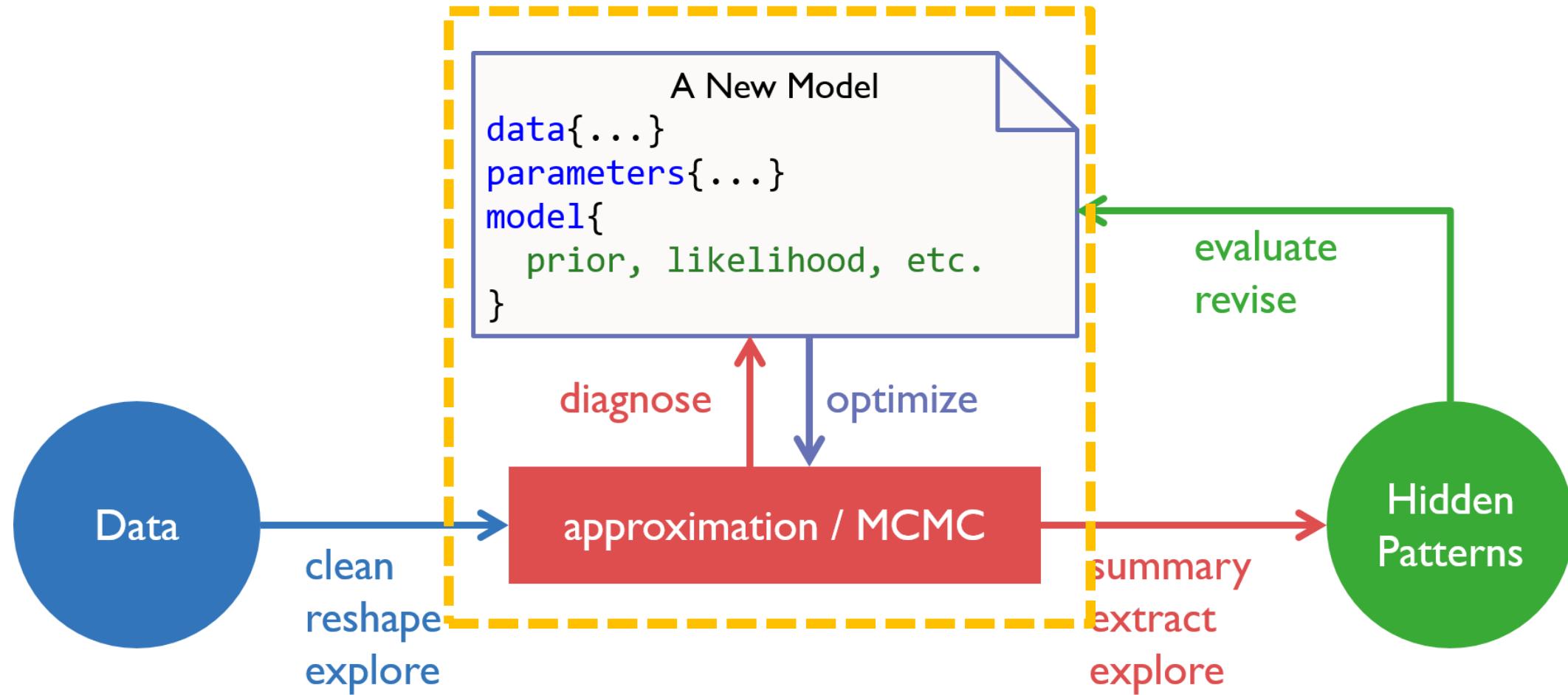


True Parameters

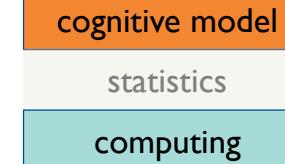


HIERARCHICAL MODELING



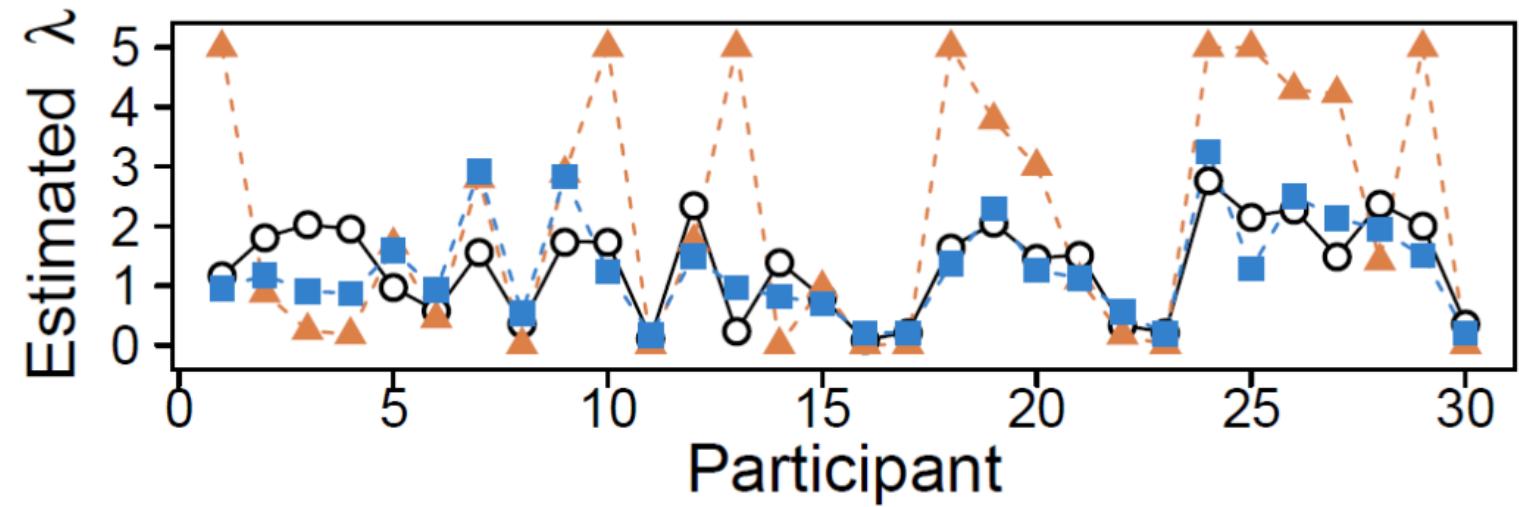


Why Hierarchical Bayesian Cognitive Modeling?



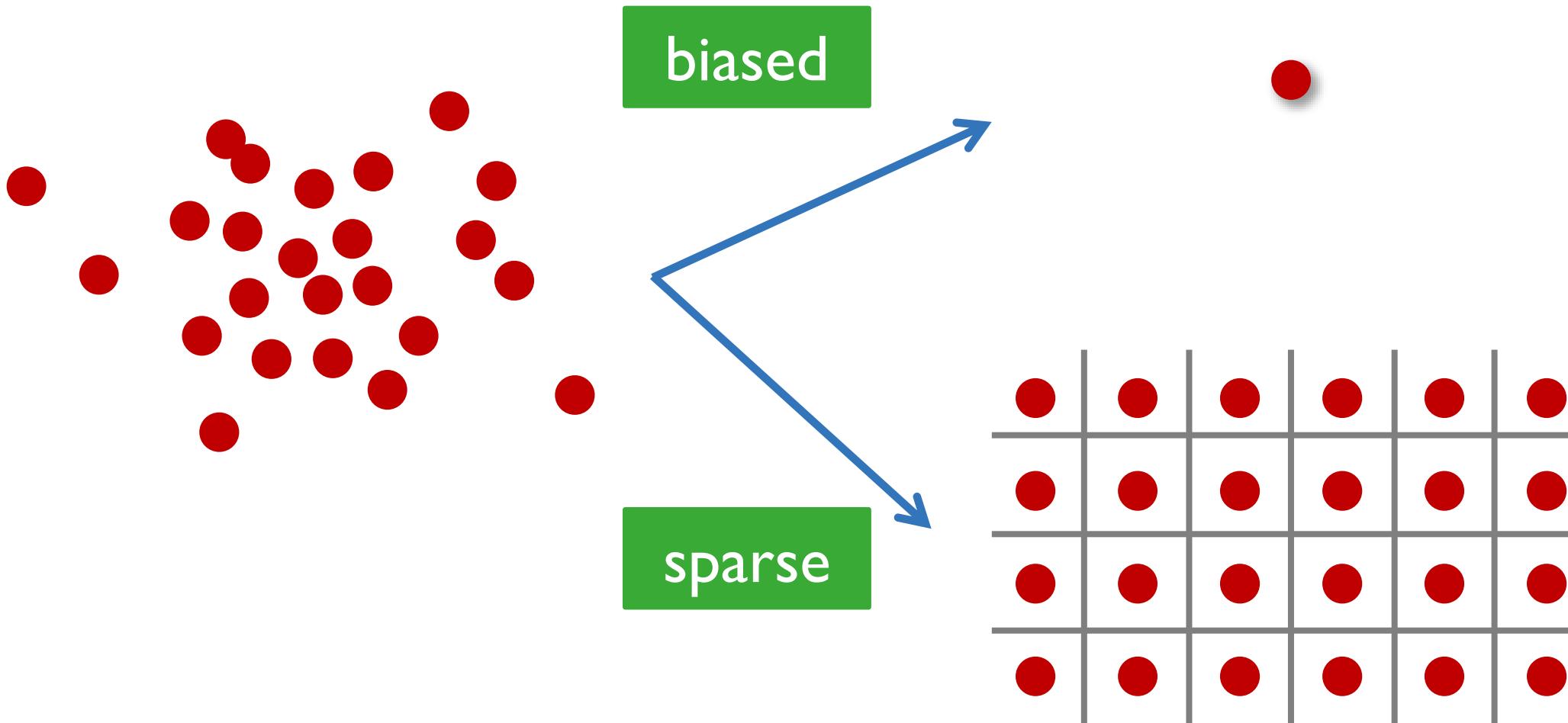
Simulation study

Hierarchical Bayesian ■
Maximum likelihood ▲
Actual values ○



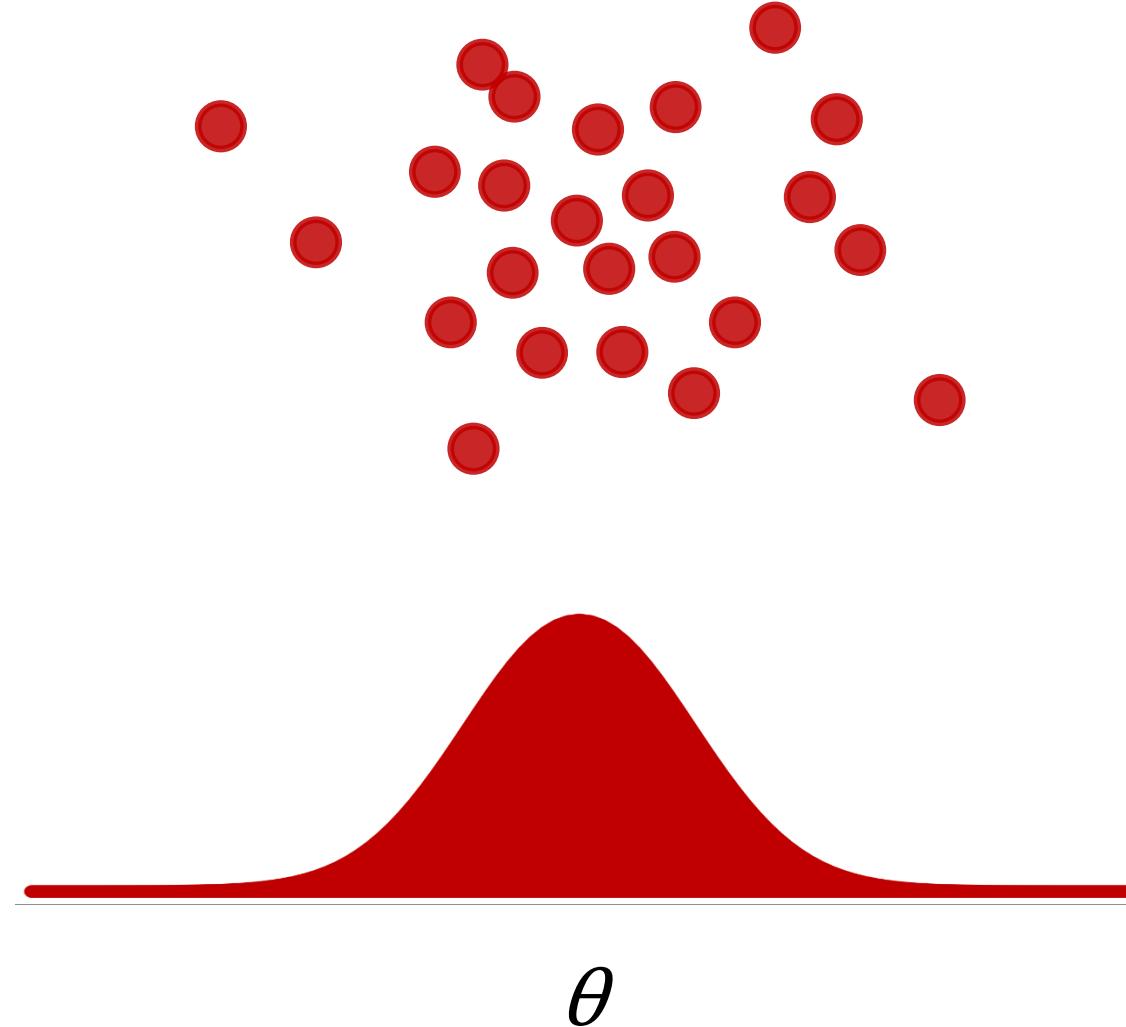
Fitting Multiple Participants

cognitive model
statistics
computing



Fitting Multiple Participants

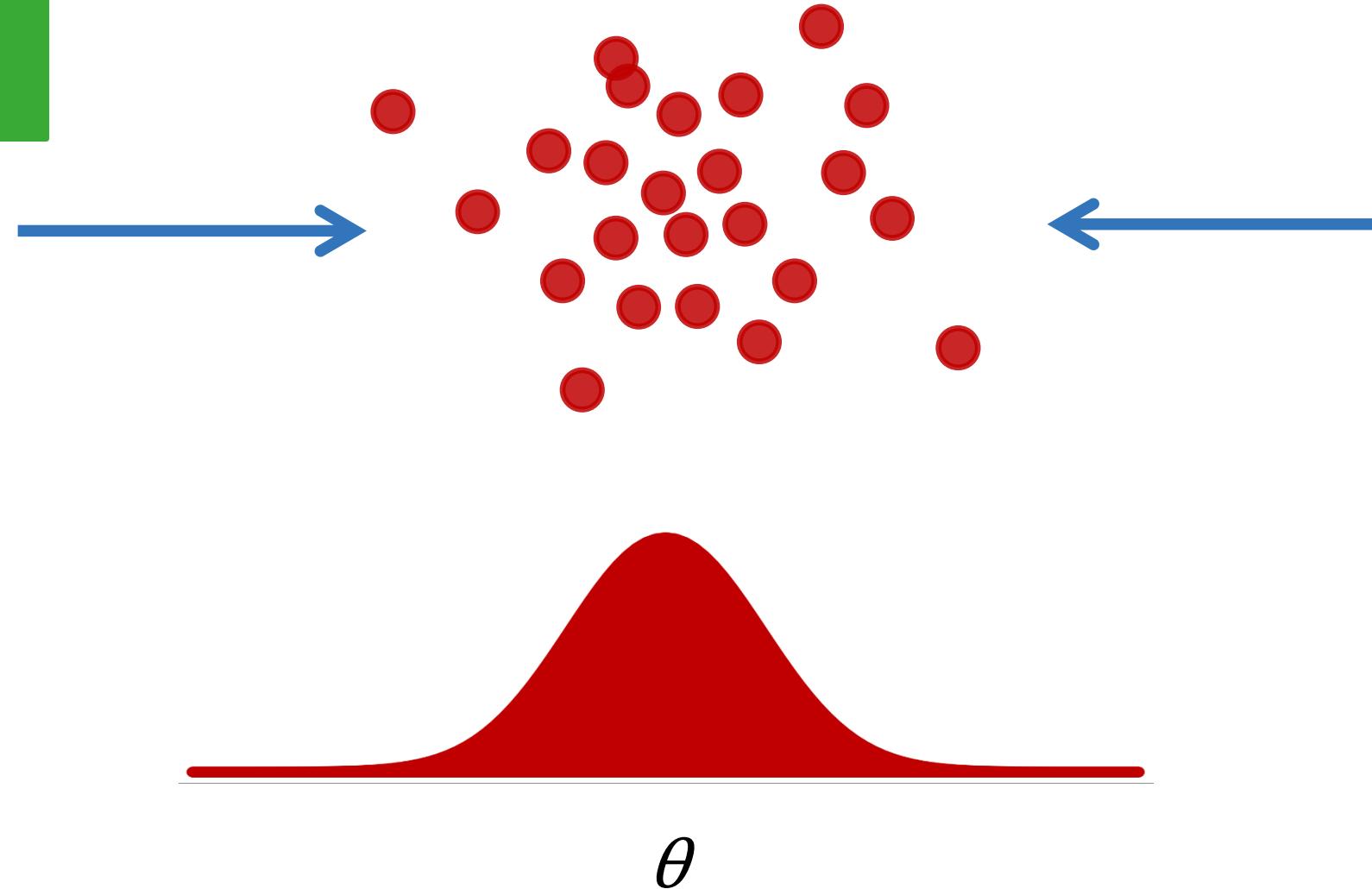
cognitive model
statistics
computing



Fitting Multiple Participants

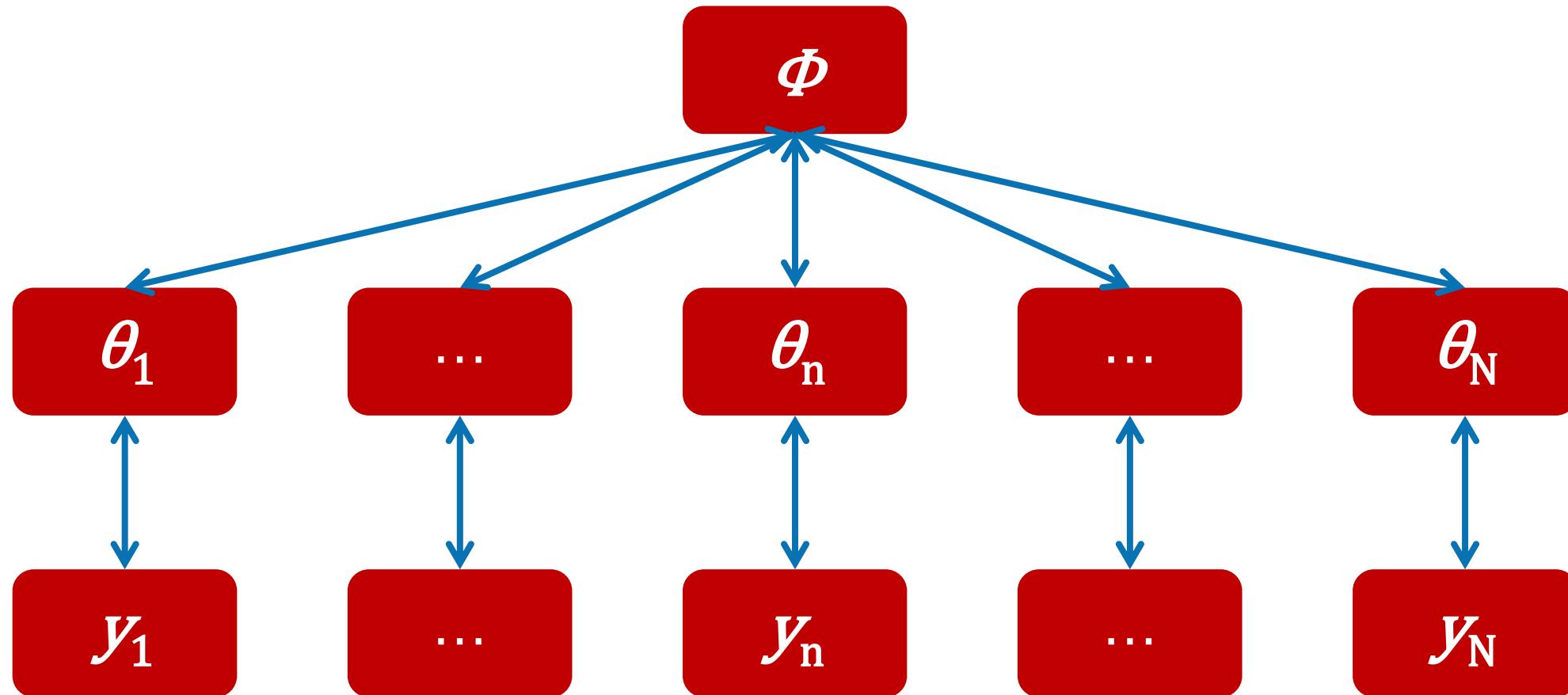
cognitive model
statistics
computing

shrinkage effect



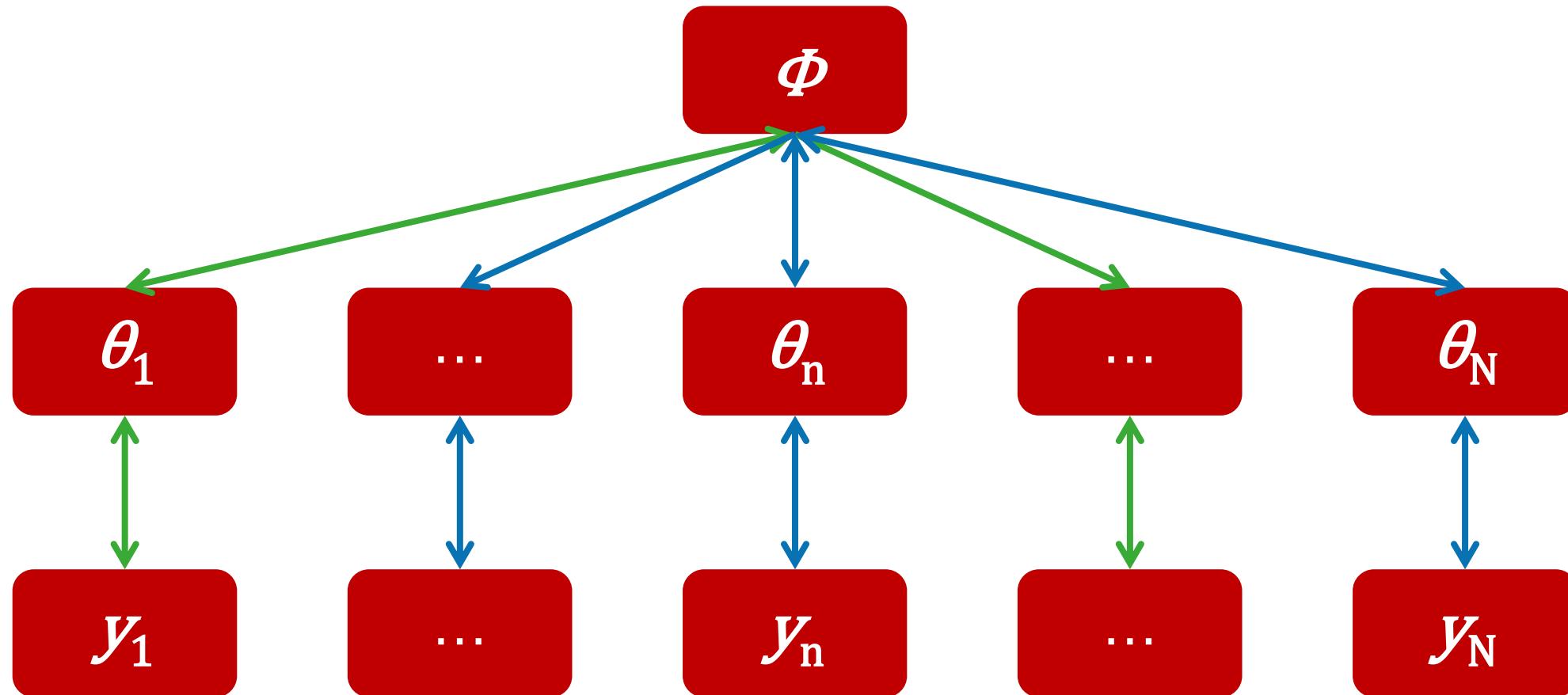
Hierarchical Structure

cognitive model
statistics
computing



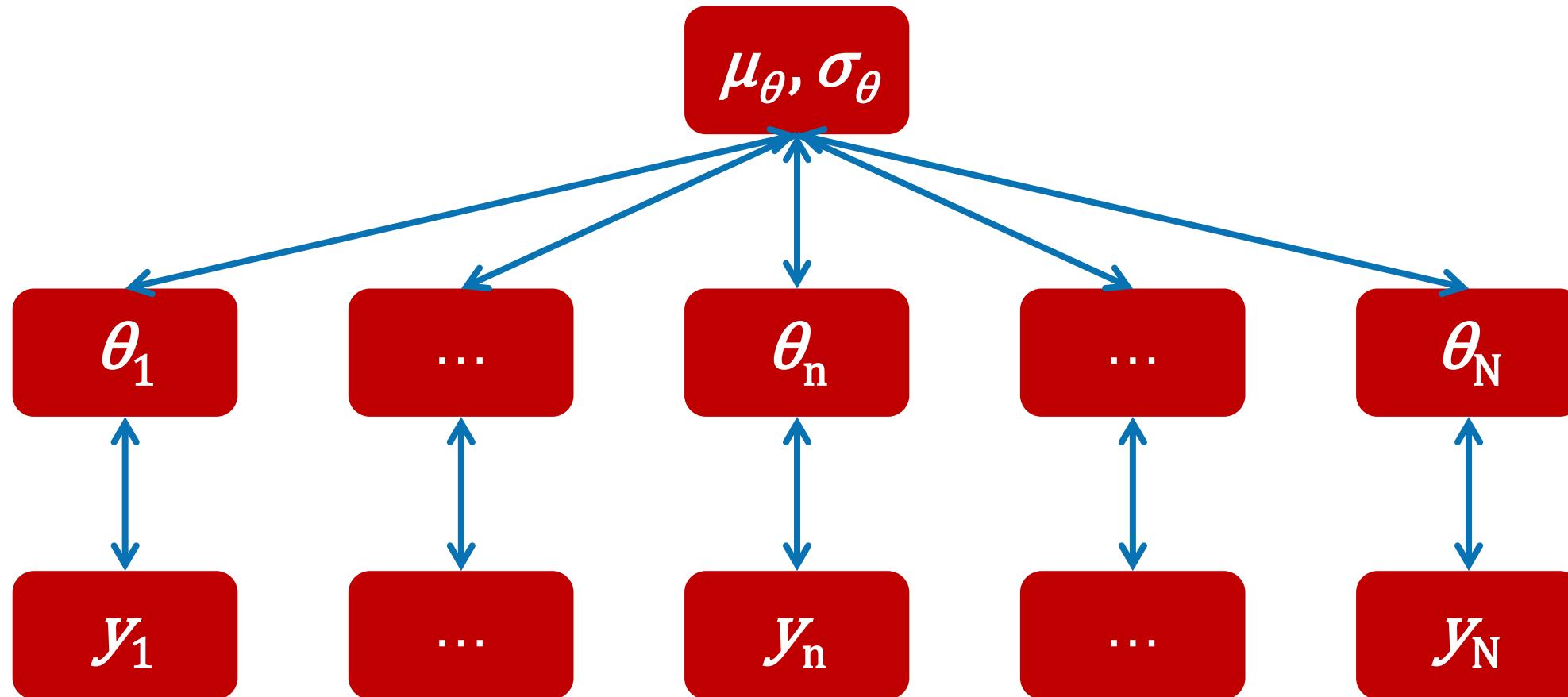
Hierarchical Structure

cognitive model
statistics
computing

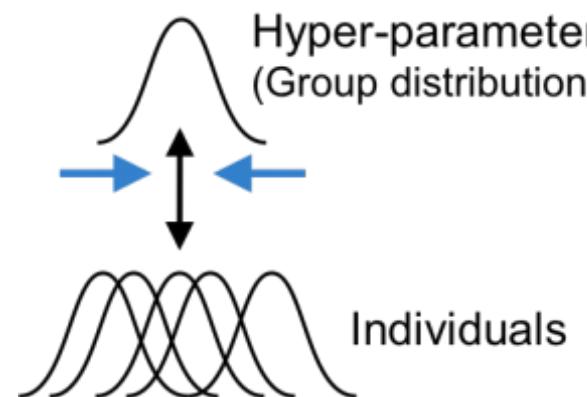
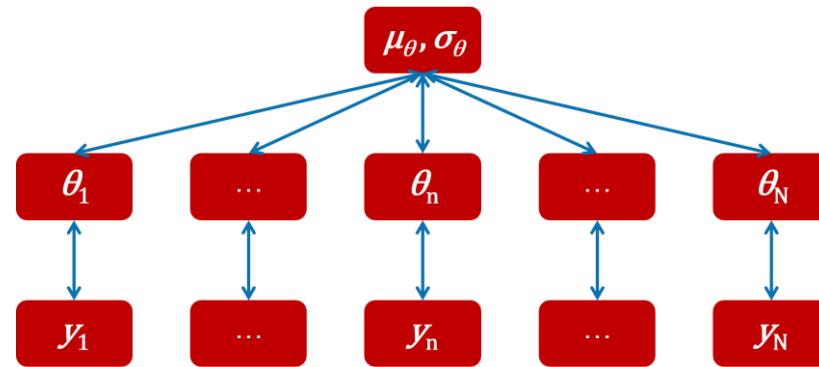


Hierarchical Structure

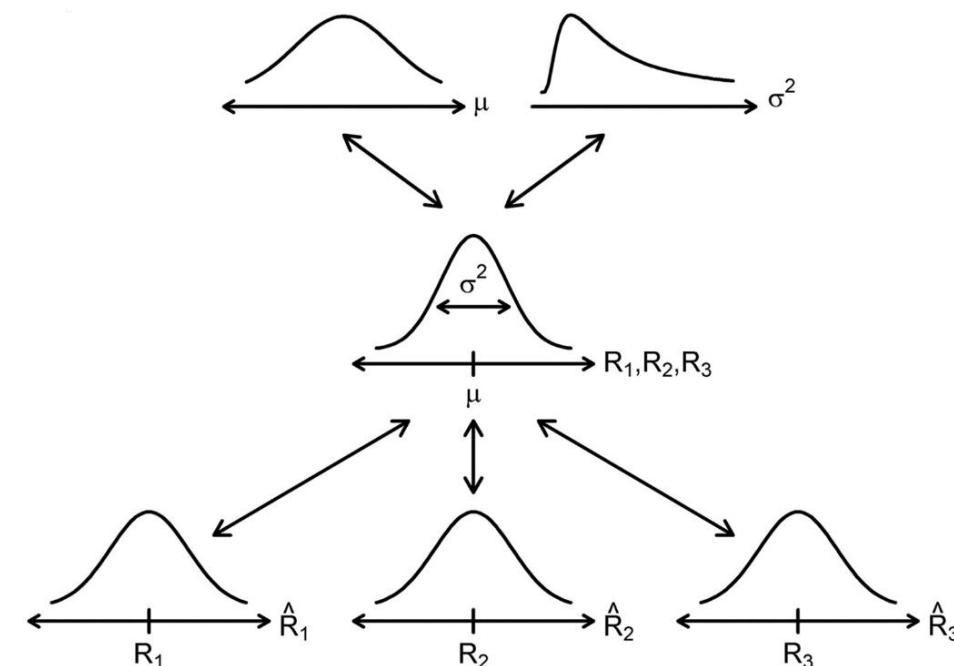
cognitive model
statistics
computing



Hierarchical Structure

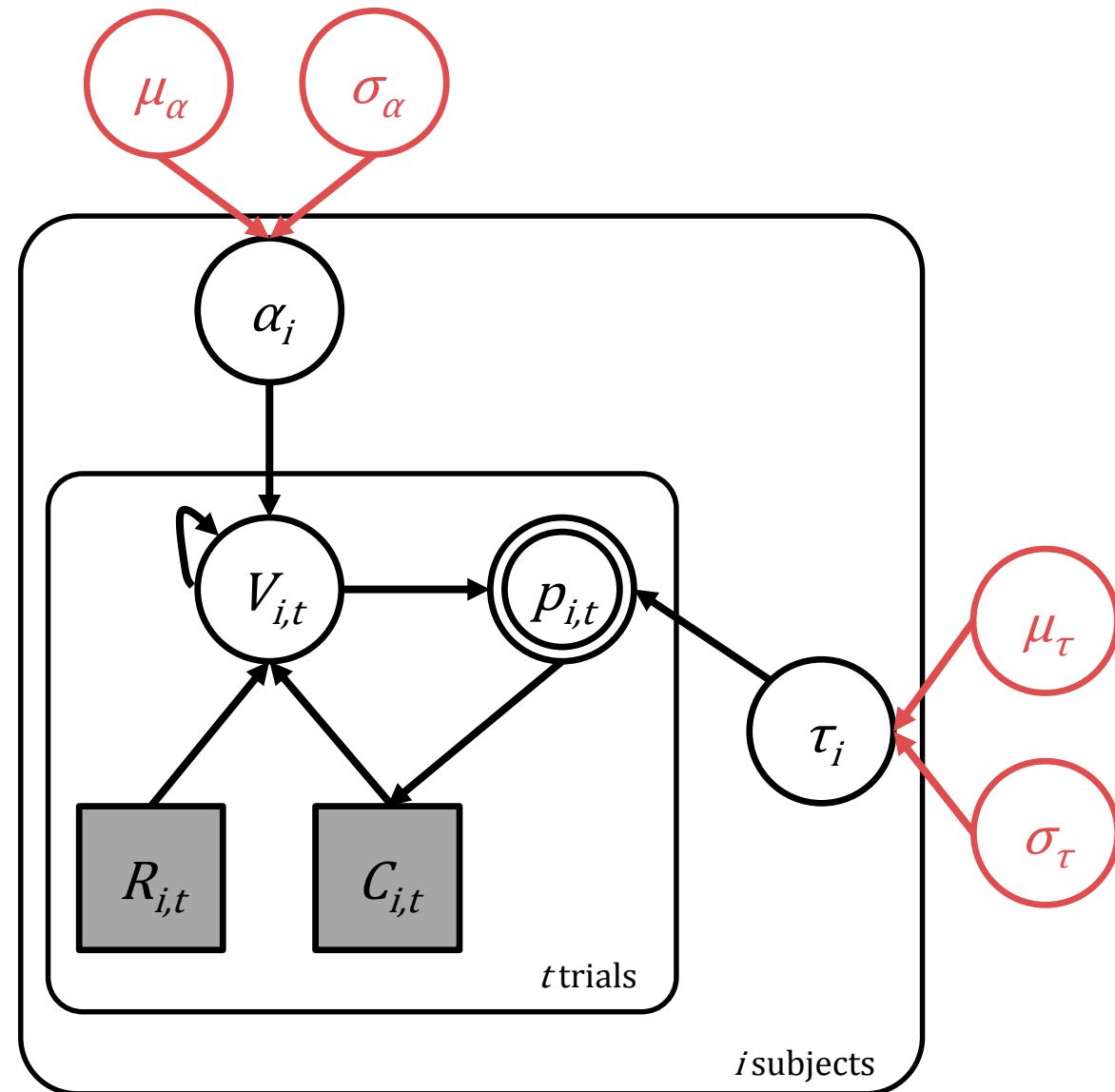


$$P(\Theta, \Phi | D) = \frac{P(D | \Theta, \Phi)P(\Theta, \Phi)}{P(D)} \propto P(D | \Theta)P(\Theta | \Phi)P(\Phi)$$



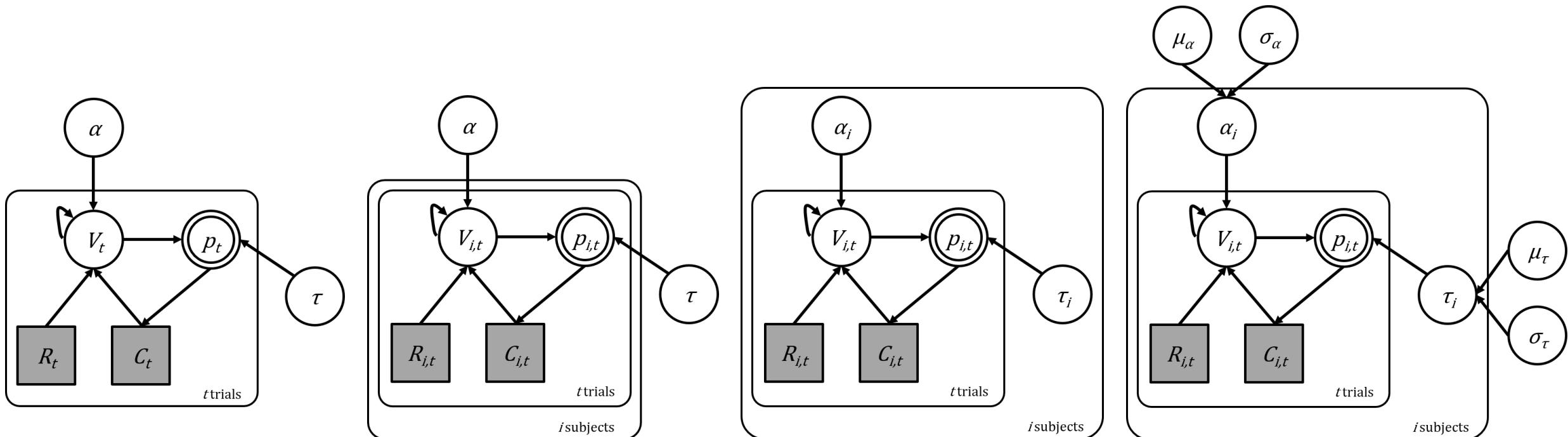
Hierarchical RL Model

cognitive model
statistics
computing



HOW DID WE GET HERE?

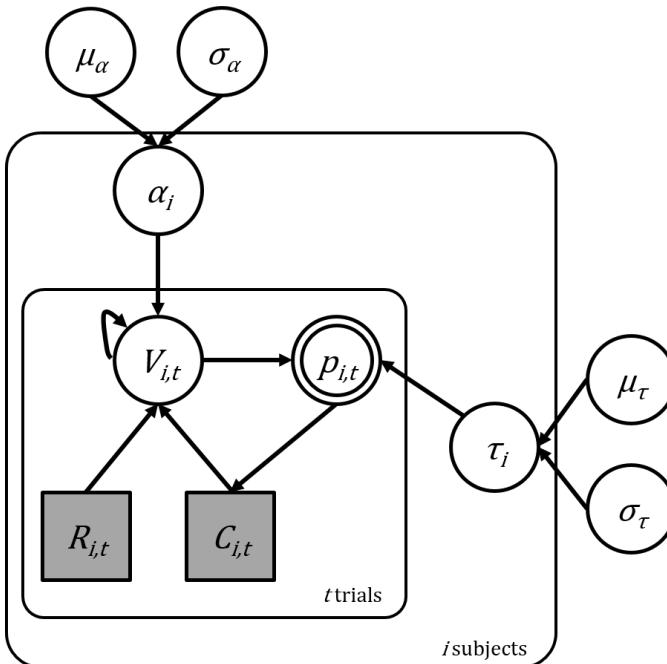
cognitive model
statistics
computing



The cognitive model *per se* is the same!

Implementing Hierarchical RL Model

cognitive model
statistics
computing



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

$$\sigma_\alpha \sim halfCauchy(0,1)$$

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

$$\sigma_\tau \sim halfCauchy(0,3)$$

$$\alpha_i \sim Normal(\mu_\alpha, \sigma_\alpha)_{\mathcal{T}(0,1)}$$

$$\tau_i \sim Normal(\mu_\tau, \sigma_\tau)_{\mathcal{T}(0,3)}$$

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

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

```

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 V

cognitive model
statistics
computing

```
.../BayesCog/06.reinforcement_learning/_scripts/reinforcement_learning_multi_parm_main.R
```

TASK: fit the hierarchical RL model

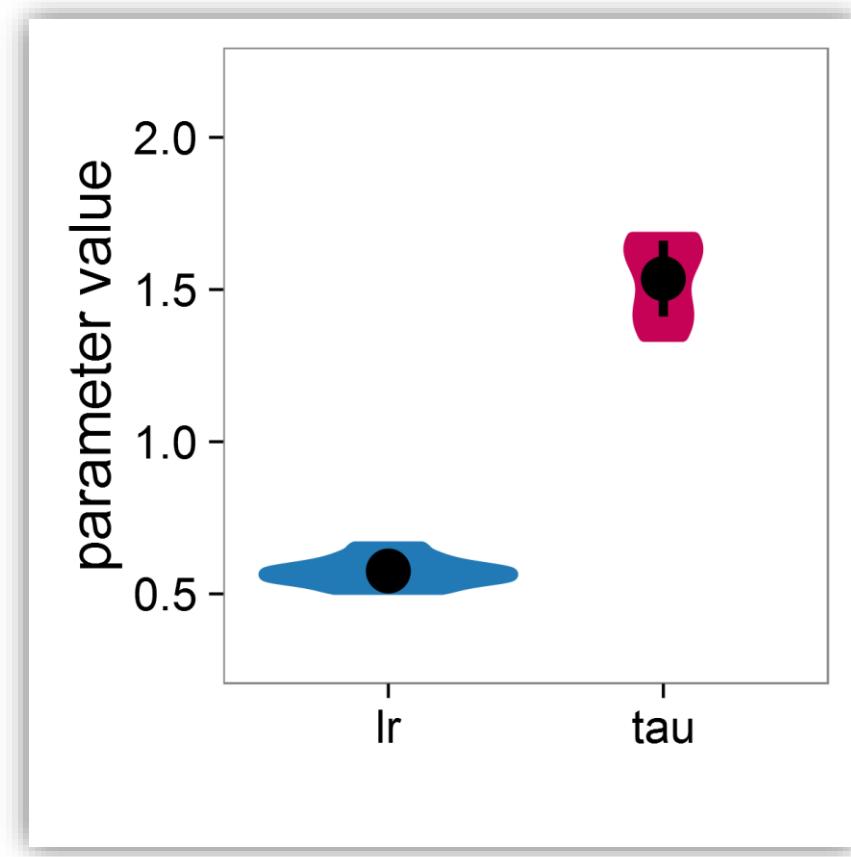
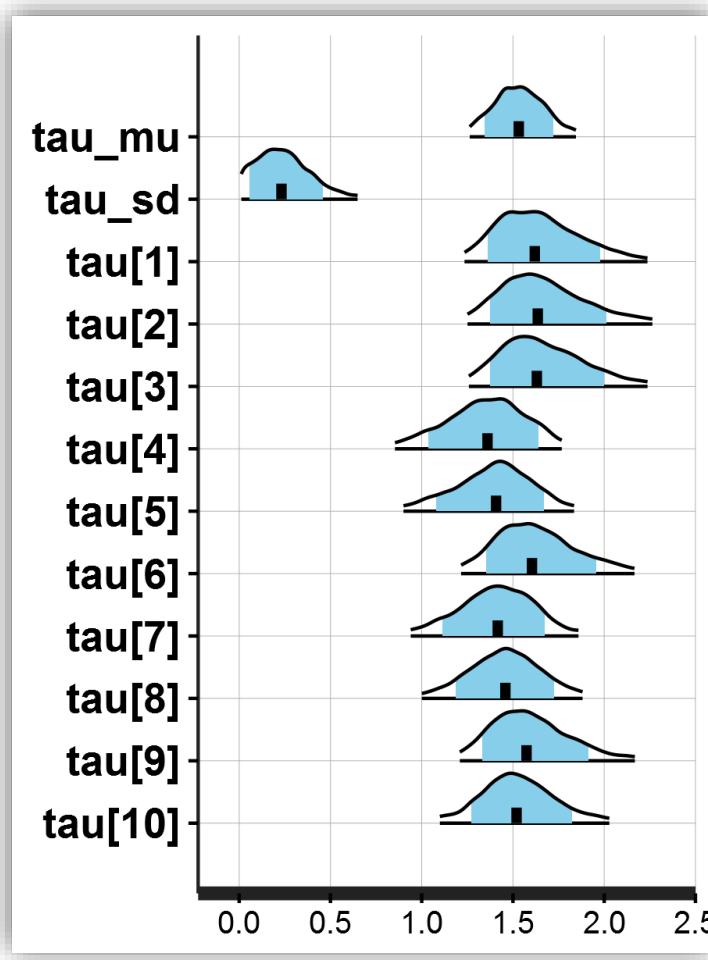
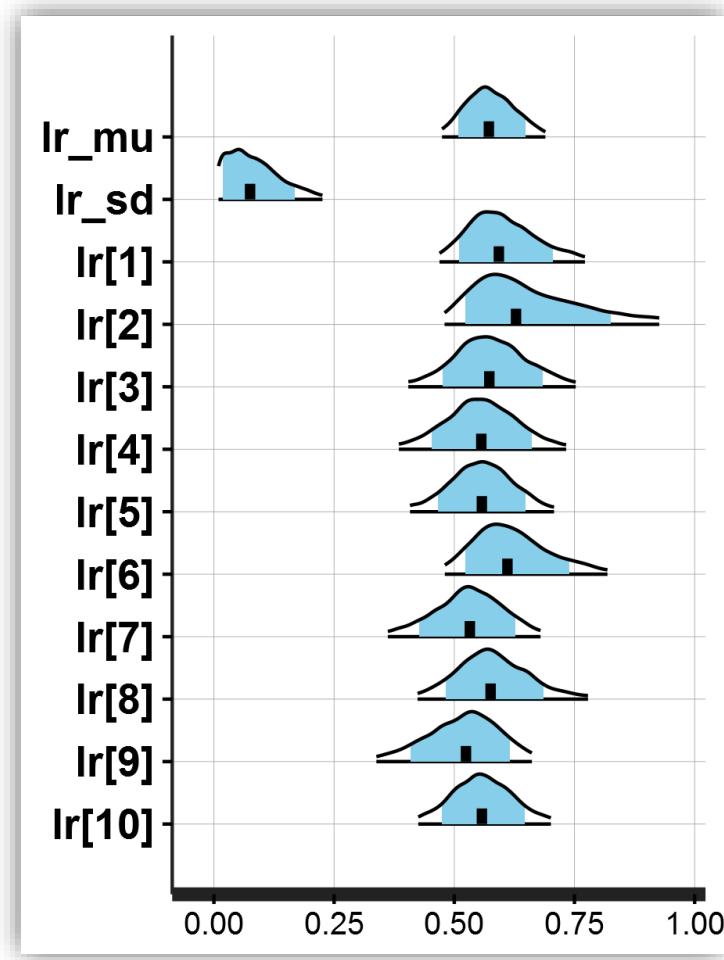
```
> source('_scripts/reinforcement_learning_multi_parm_main.R')  
  
> fit_rl3 <- run_rl_mp( modelType ='hrch' )
```

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

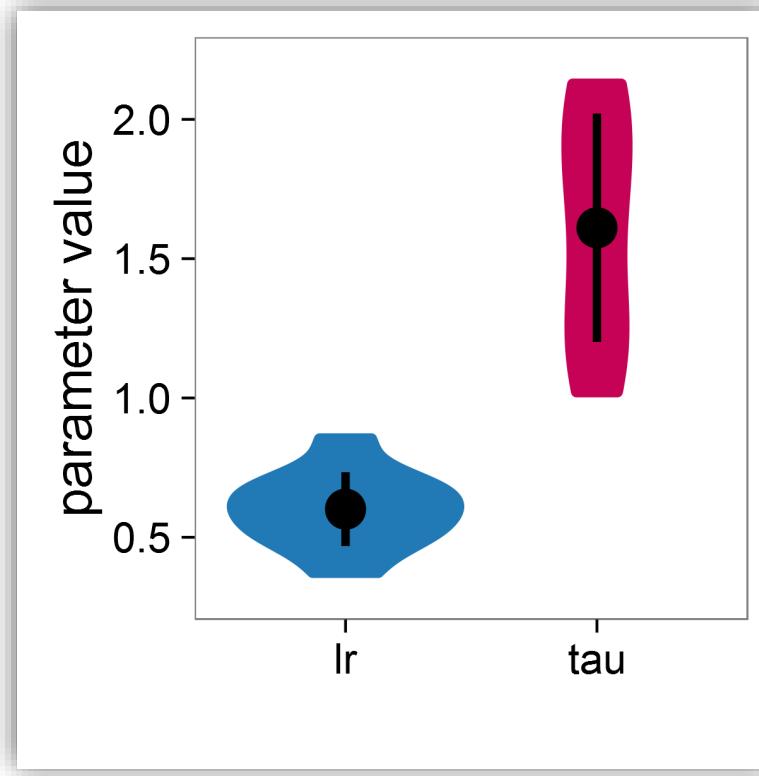


*: adapt_delta=0.999, max_treedepth=100

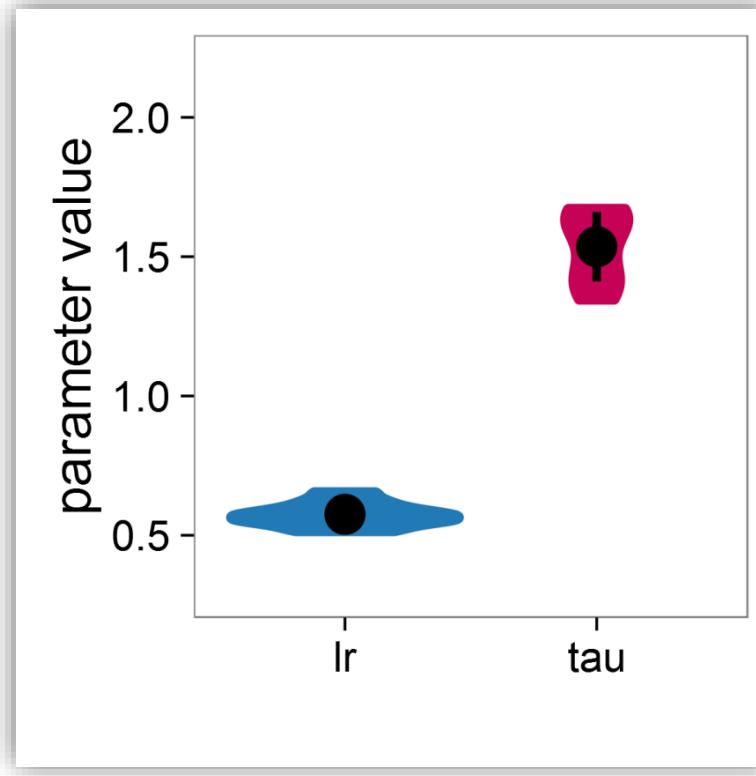
Comparing with True Parameters

cognitive model
statistics
computing

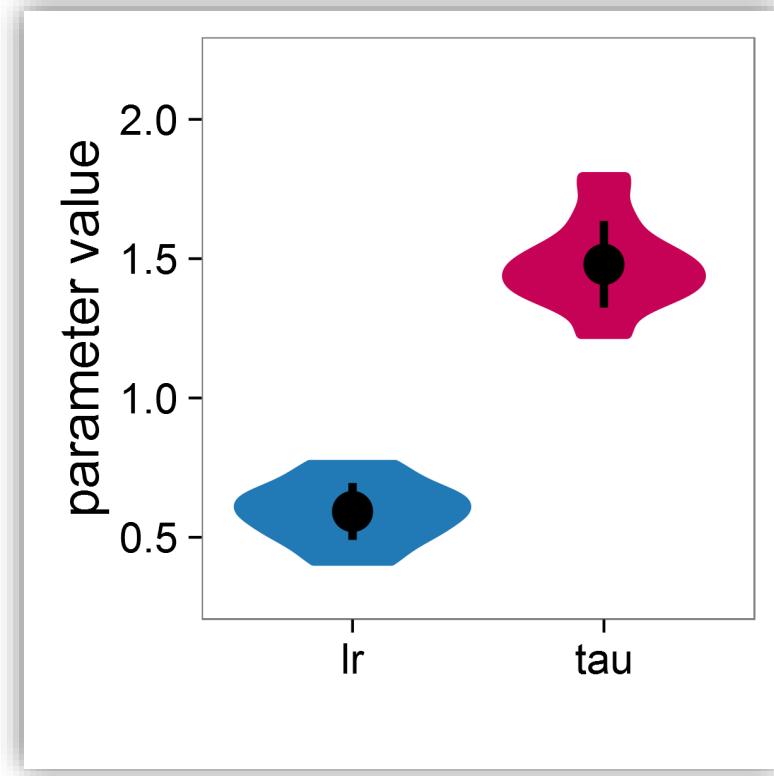
Posterior Means (indv)



Posterior Means (hrch)*



True Parameters



*: adapt_delta=0.999, max_treedepth=100

Group-level Parameters

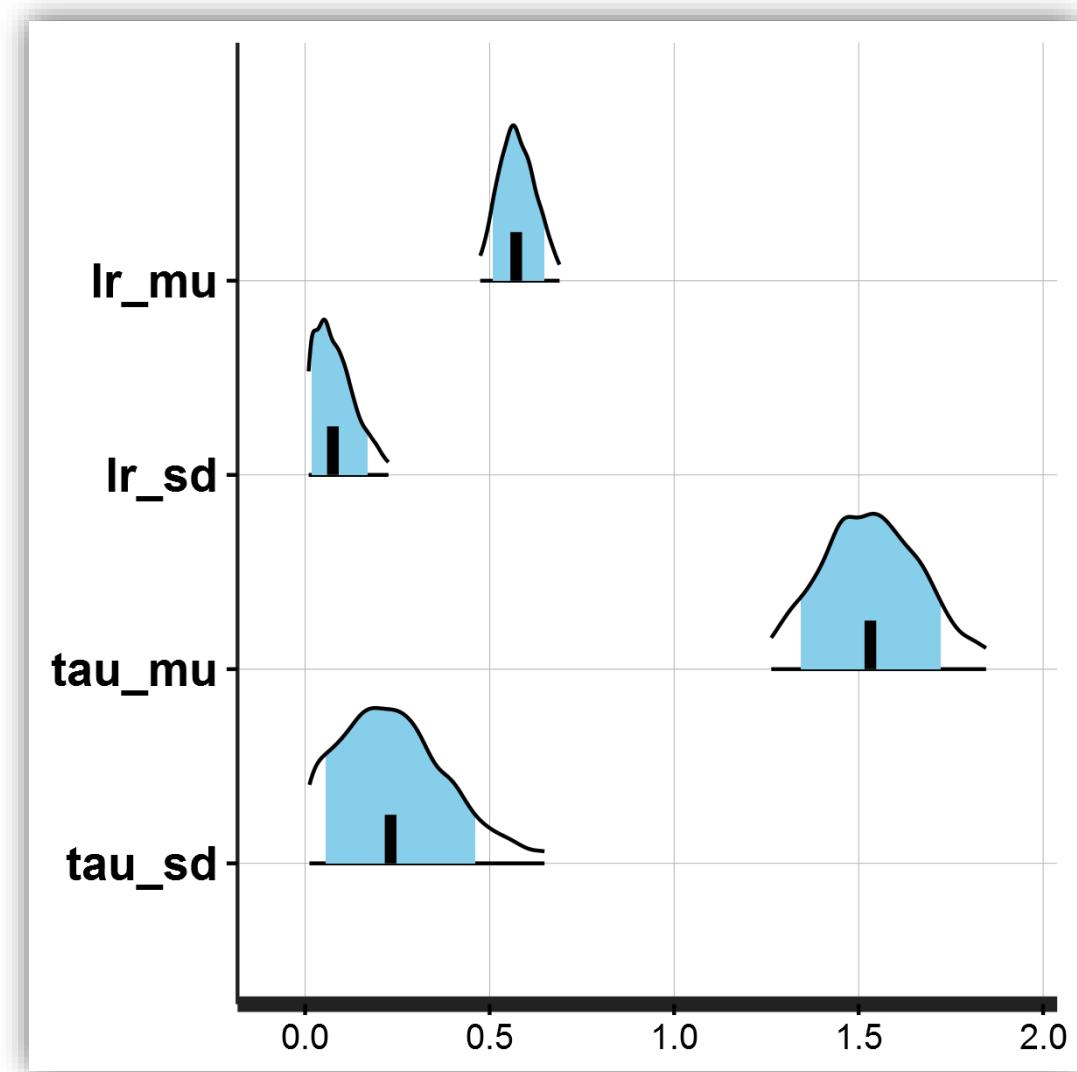
cognitive model
statistics
computing

True group parameters

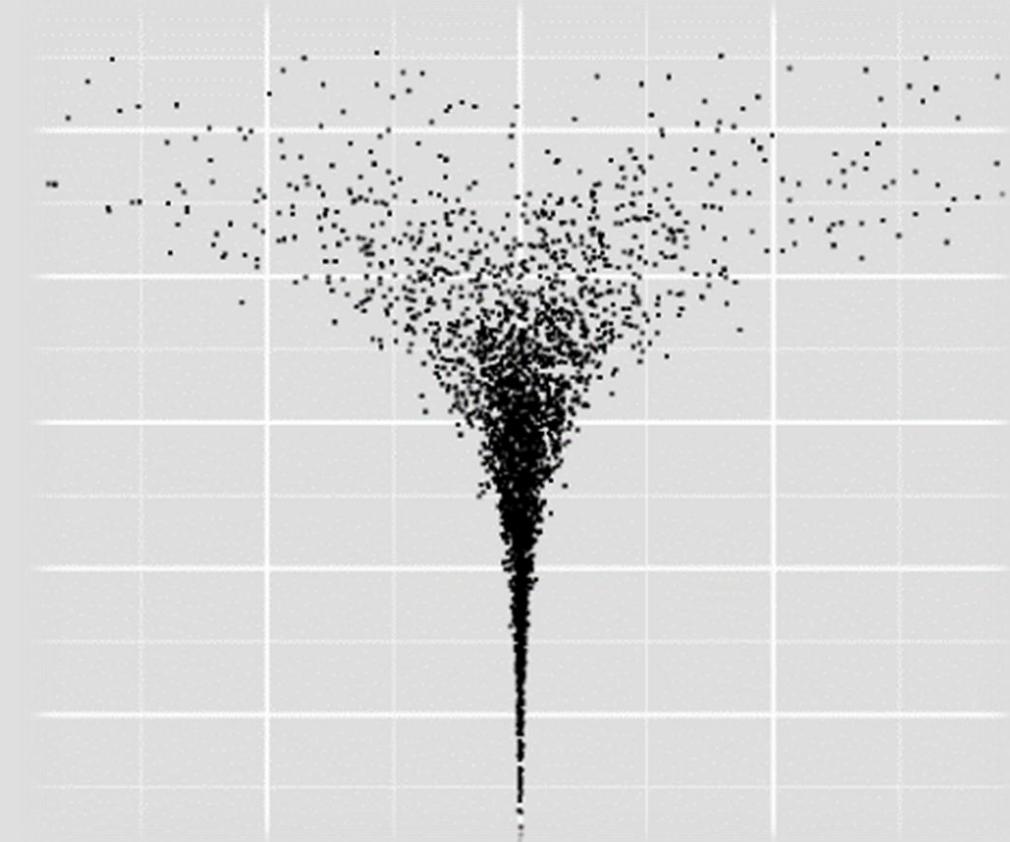
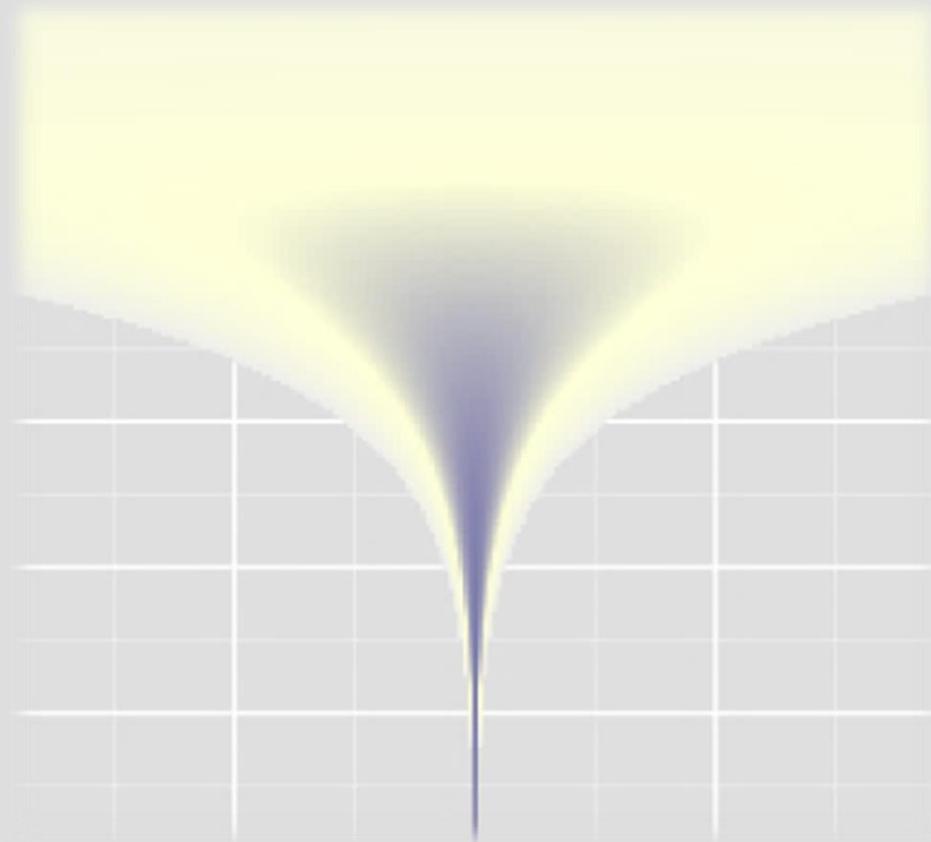
```
lr = rnorm(10, mean=0.6, sd=0.12)  
tau = rnorm(10, mean=1.5, sd=0.2)
```

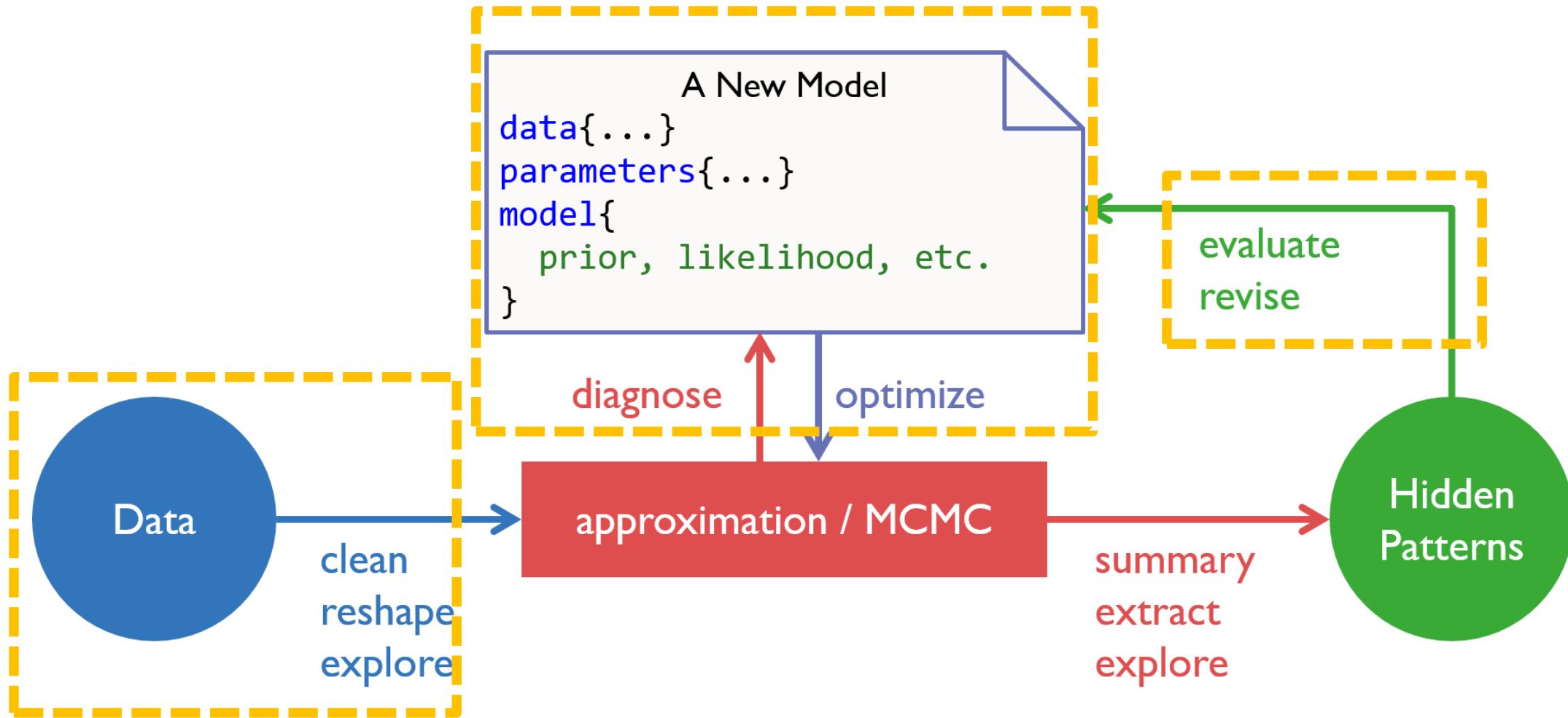
Estimated group parameters

	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

- 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);  
}
```

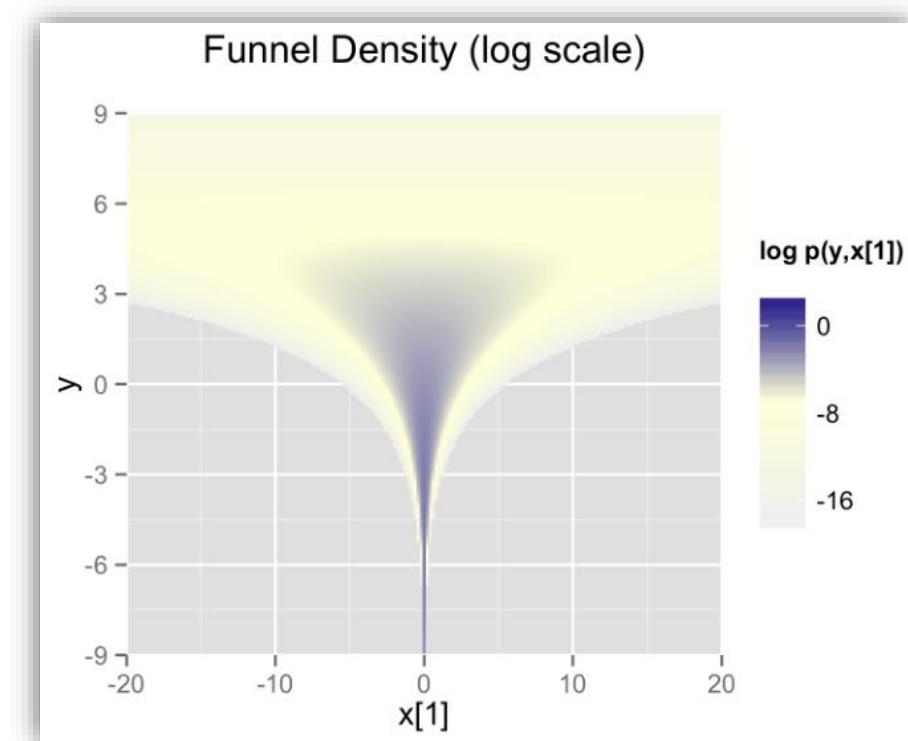
```
model {  
  height ~ normal(alpha + beta * weight, sigma);  
}
```

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

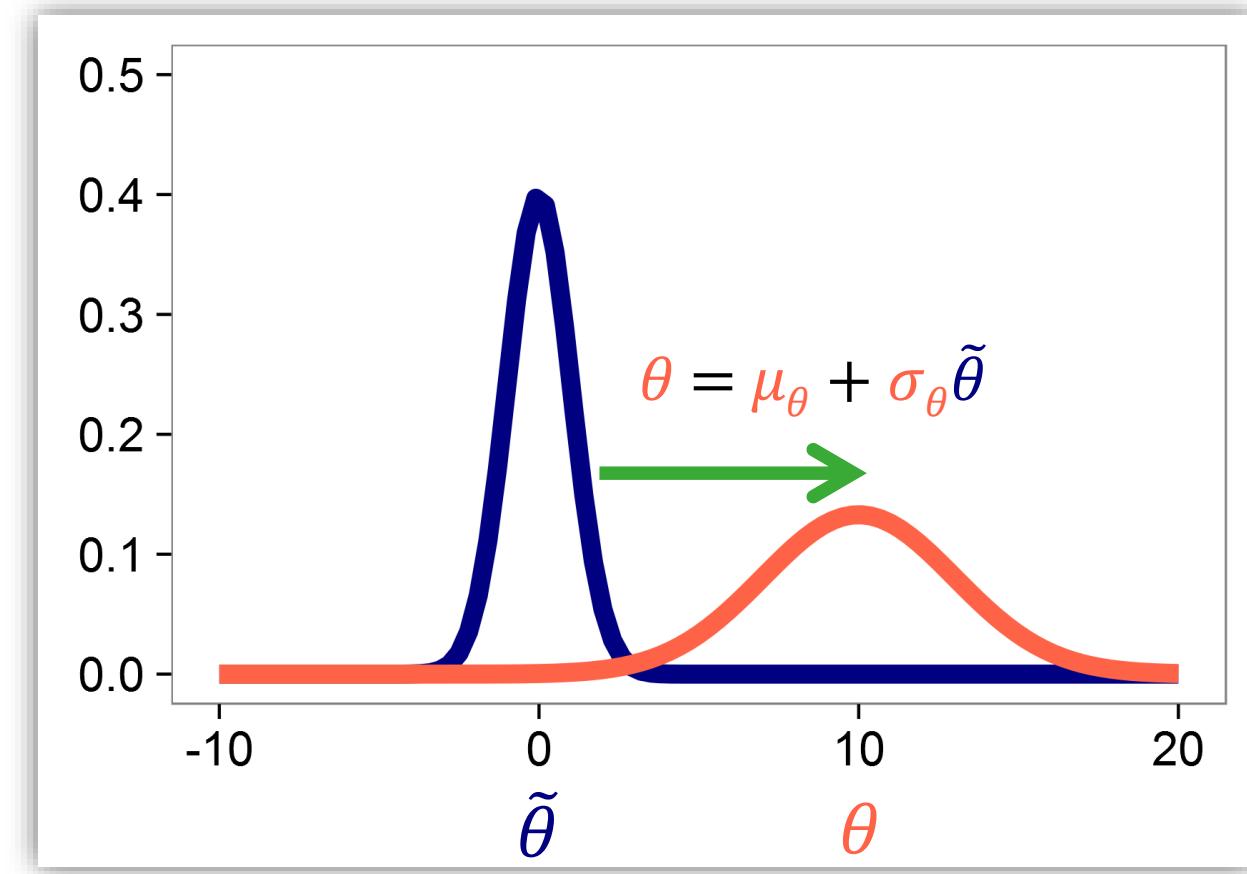
$$\theta \sim Normal(\mu_\theta, \sigma_\theta)$$



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

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

Stan likes simple distributions!

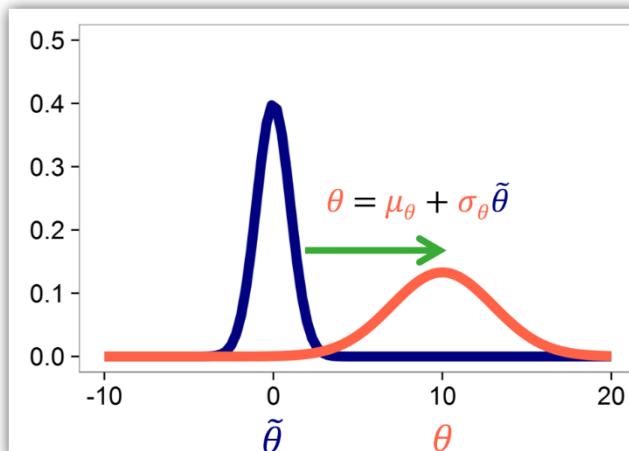


Reparameterization

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
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

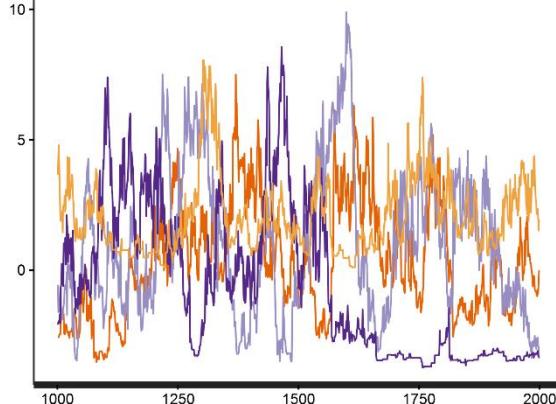
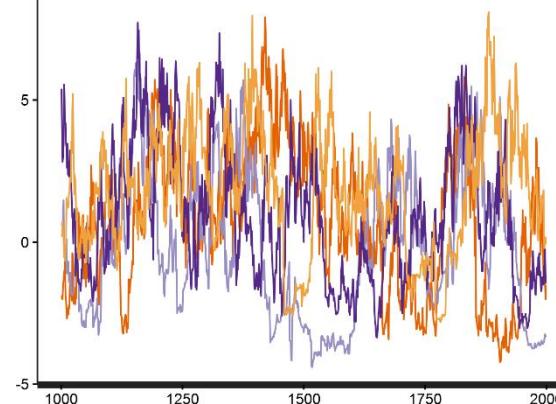
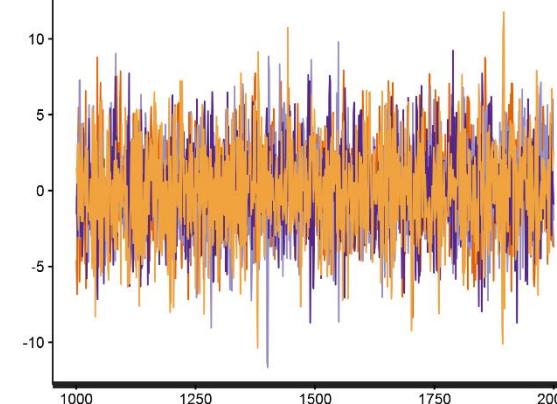
Typical adjustments

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

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

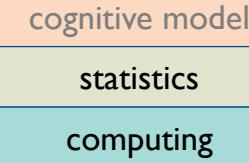
Neal's Funnel: Comparing Performance

cognitive model
statistics
computing

	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

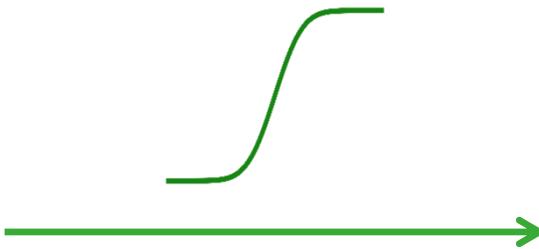
How about Bounded Parameters?



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

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

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



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

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

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

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

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 VI

cognitive model
statistics
computing

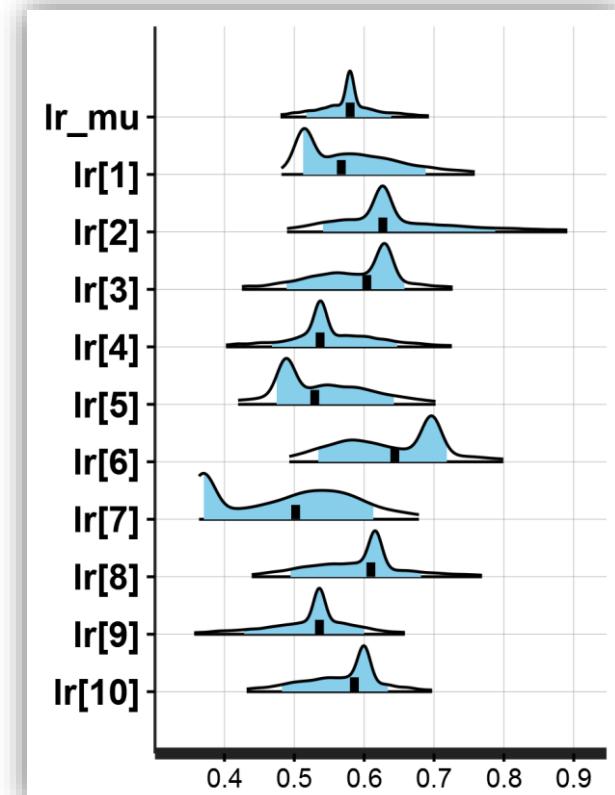
```
.../BayesCog/07.optm_rl/_scripts/reinforcement_learning_hrch_main.R
```

TASK: fit the optimized hierarchical RL model

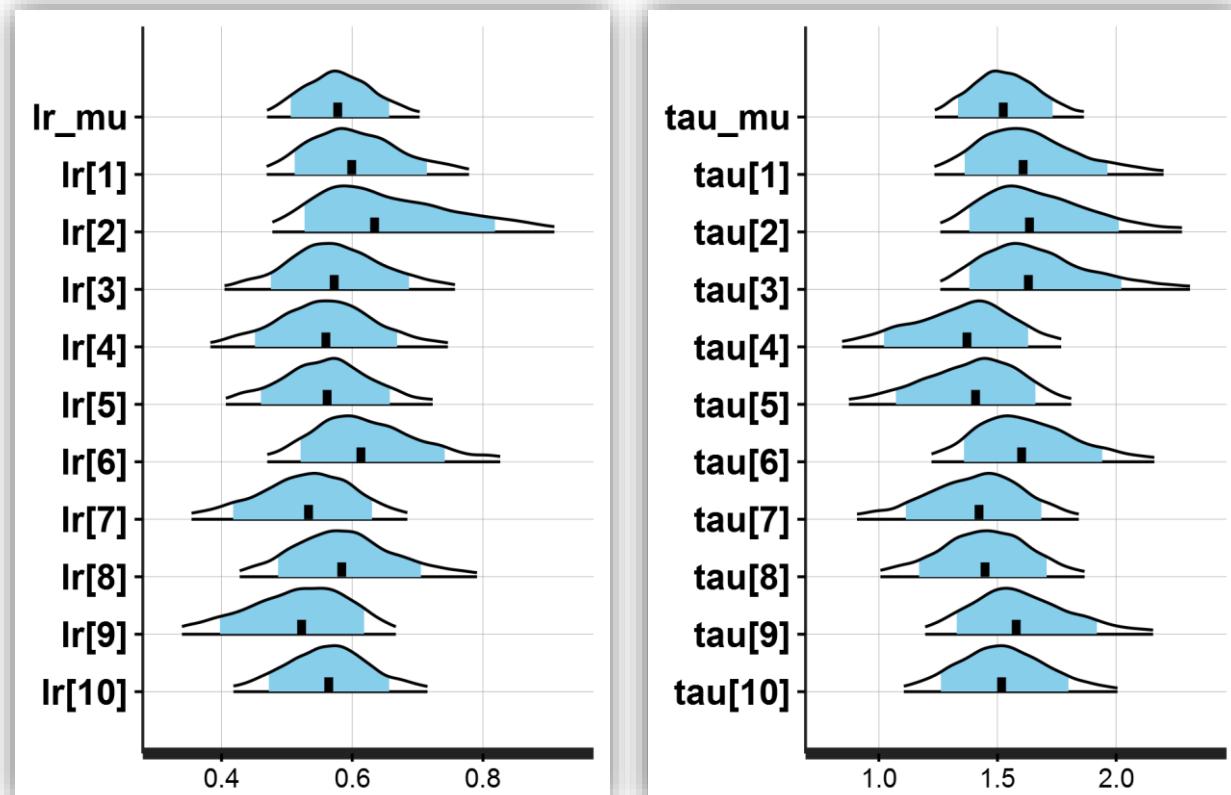
```
> source('_scripts/reinforcement_learning_hrch_main.R')  
  
> fit_rl4 <- run_rl_mp2(optimized = TRUE)
```

Hierarchical Fitting – Optimized

Posterior Means (hrch)



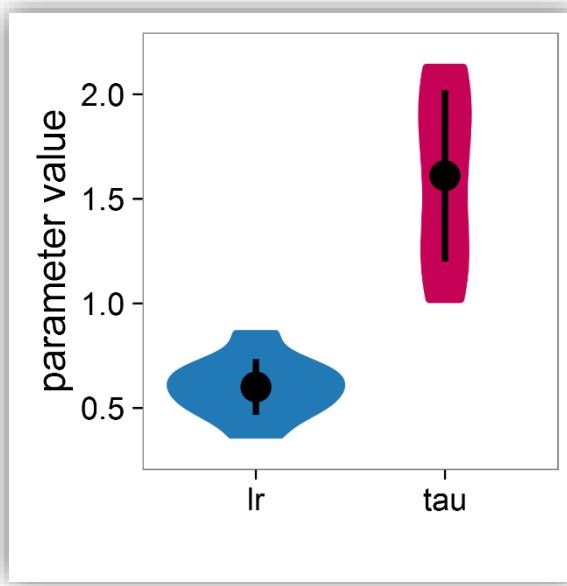
Posterior Means (hrch + optim)



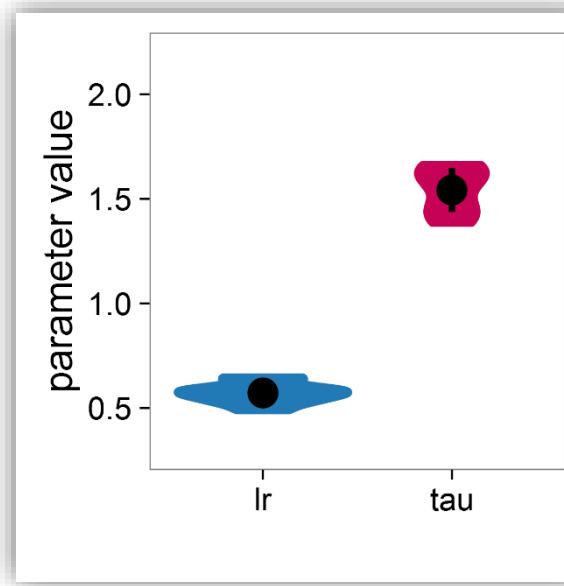
Comparing with True Parameters

cognitive model
statistics
computing

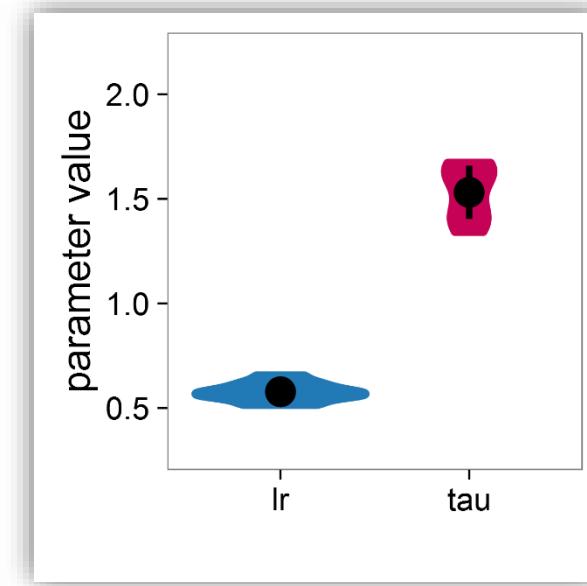
Posterior Means (indv)



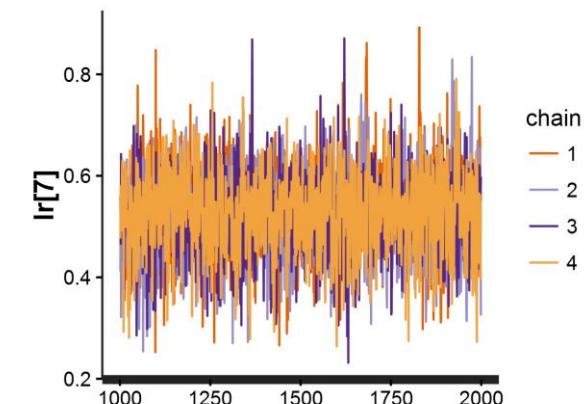
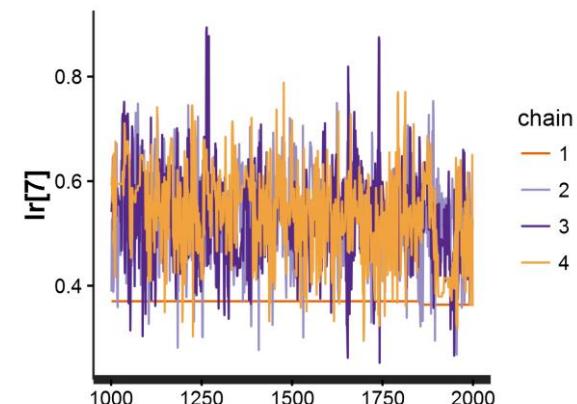
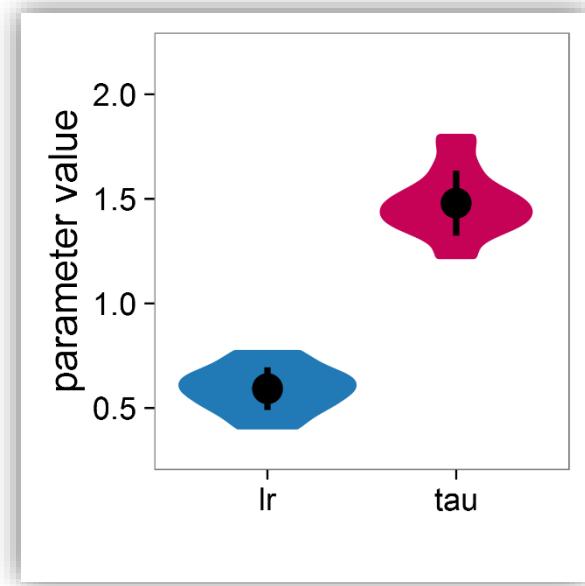
Posterior Means (hrch)



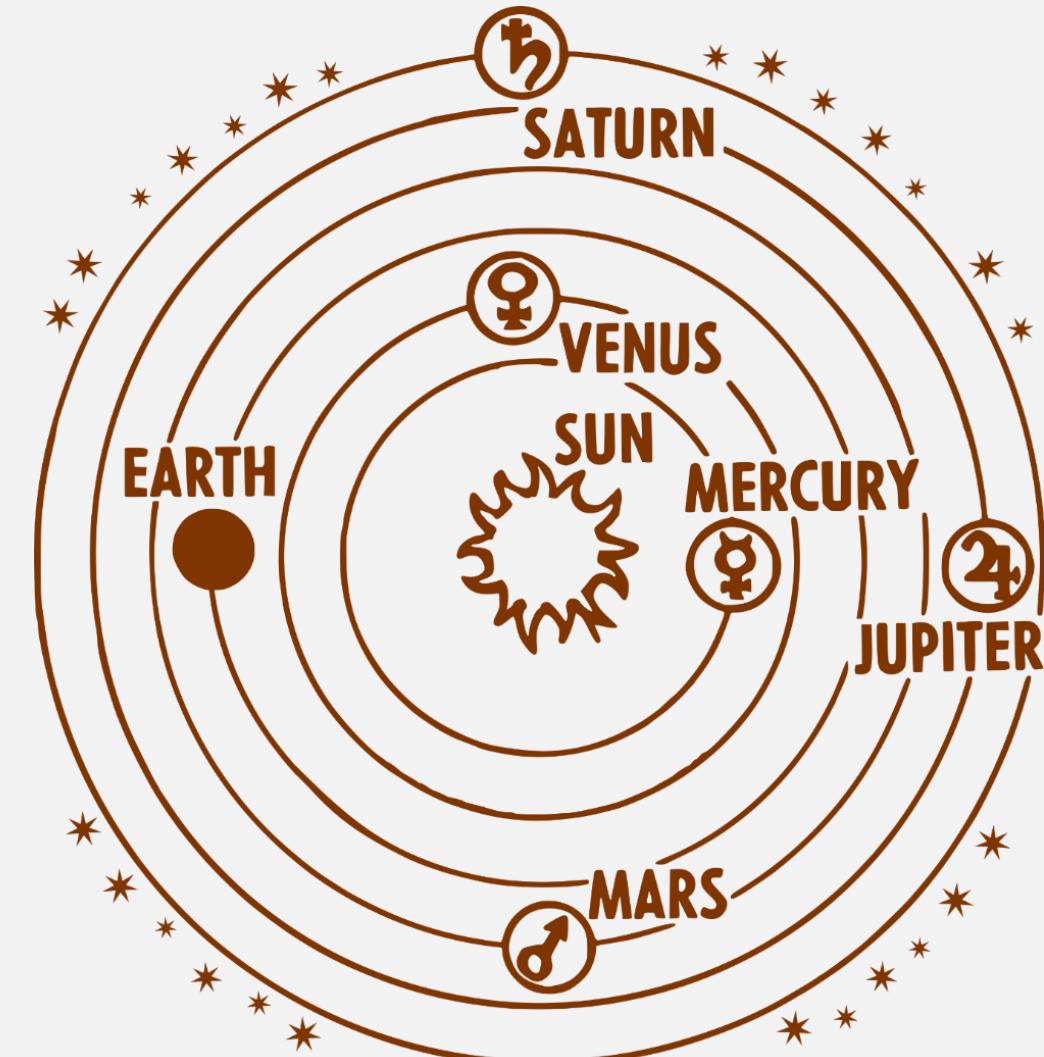
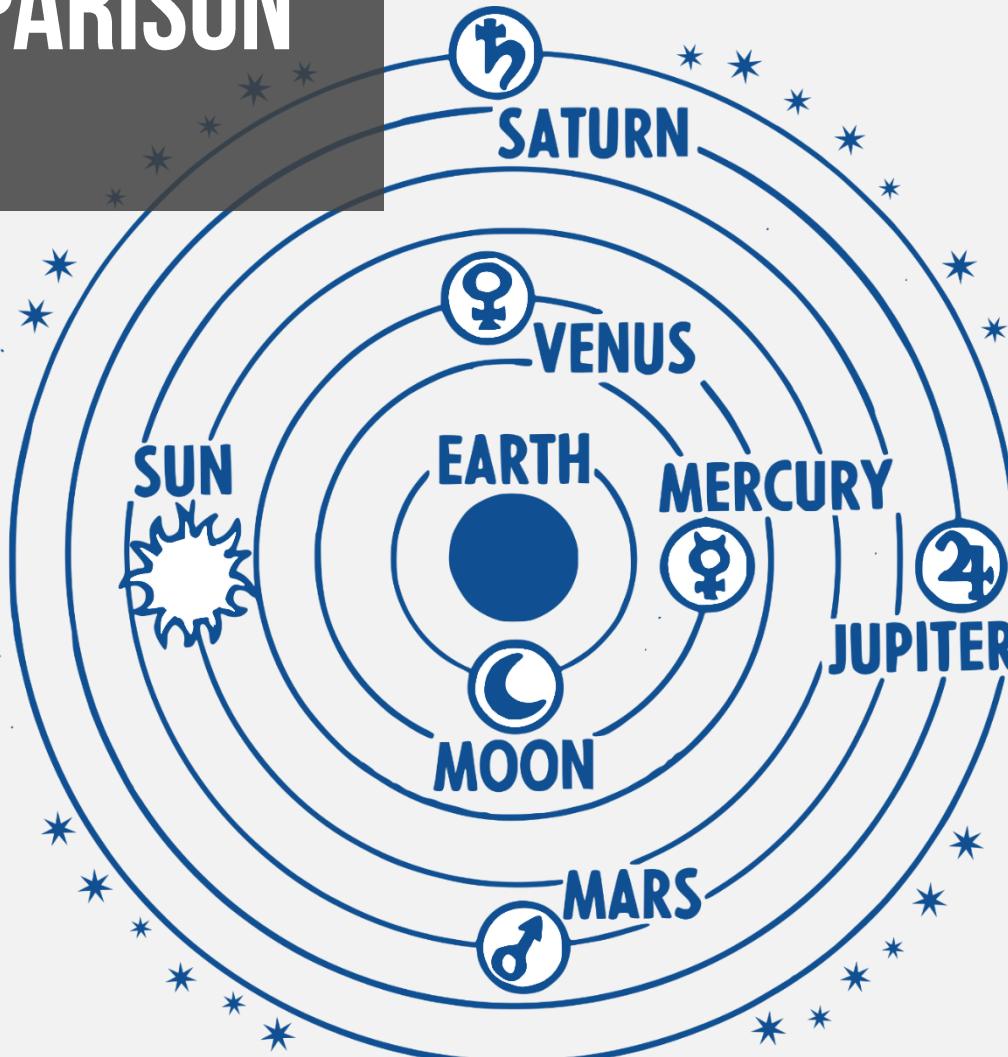
Posterior Means (hrch+optm)

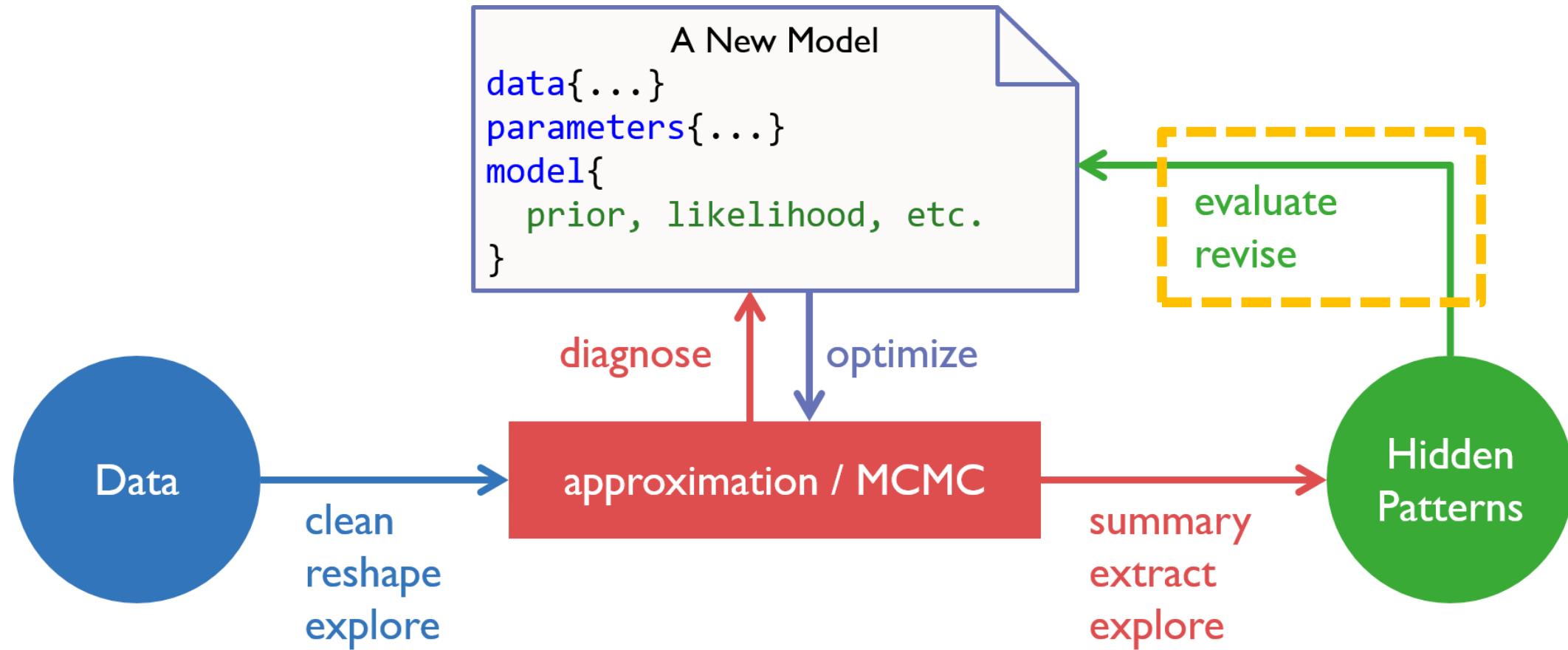


True Parameters



MODEL COMPARISON





Model Comparison

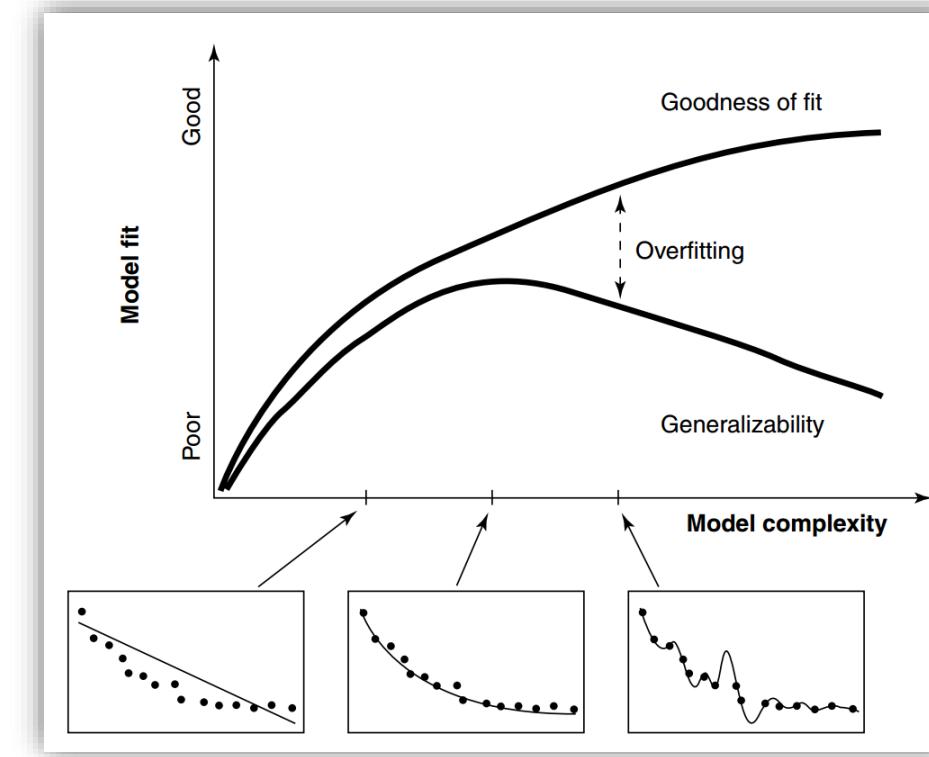
cognitive model
statistics
computing

Which model provides the best **fit**?

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

Ockham's razor:

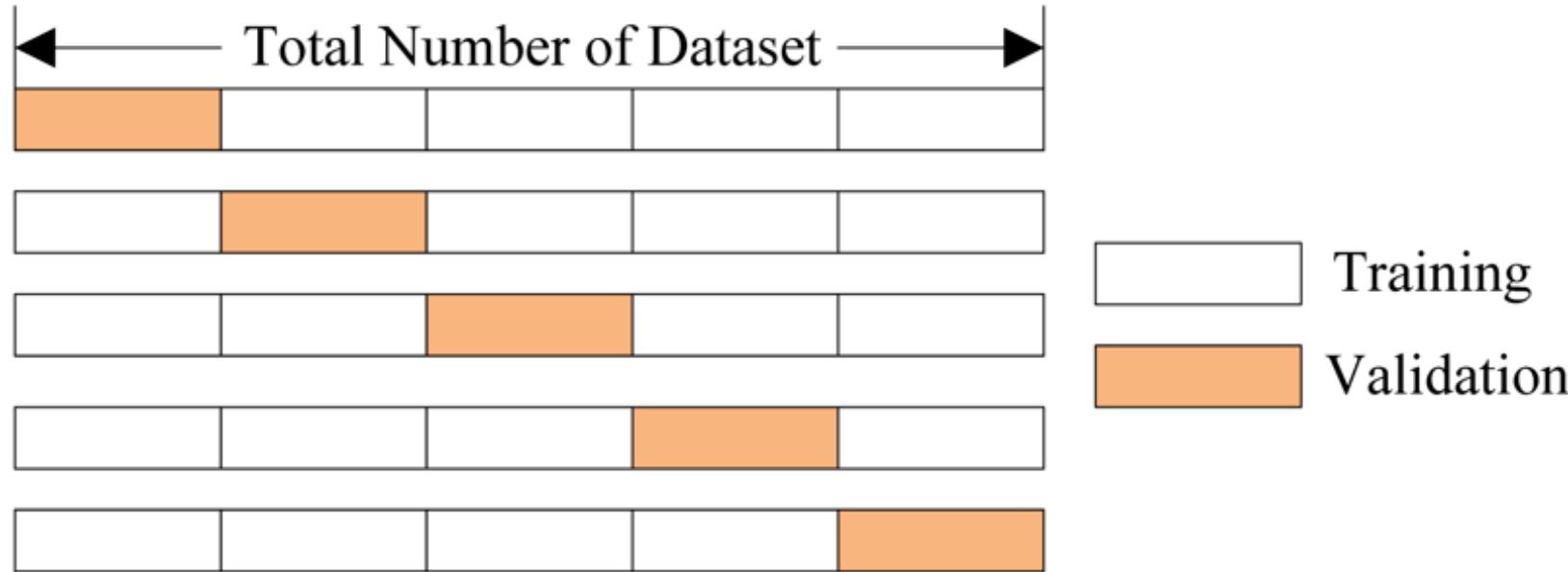
Models with fewer assumptions are to be preferred



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

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

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)
```

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

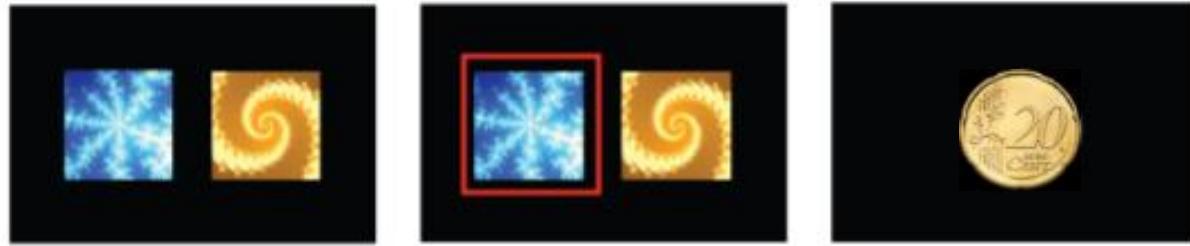
```
> library(loo)
> LL1    <- extract_log_lik(stanfit)
> loo1   <- loo(LL1)    # PSIS leave-one-out
> waic1 <- waic(LL1)   # WAIC
```

Pareto Smoothed Importance Sampling

Reversal Learning Task

cognitive model
statistics
computing

fictitious RL
model



Value update:

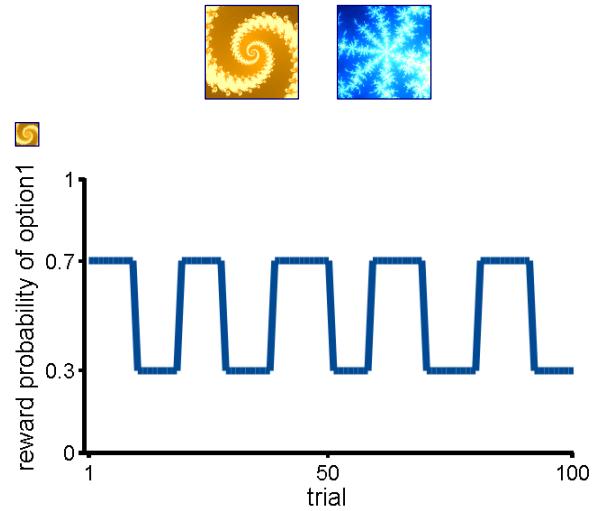
$$V_{t+1}^c = V_t^c + lr * PE$$

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

Prediction error:

$$PE = R_t - V_t^c$$

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



Exercise VII

cognitive model
statistics
computing

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

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

Exercise VII – 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

ANY
QUESTIONS
?

Stay tuned and
bis morgen!