TEWA 1: Advanced Data Analysis

Lecture 11

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

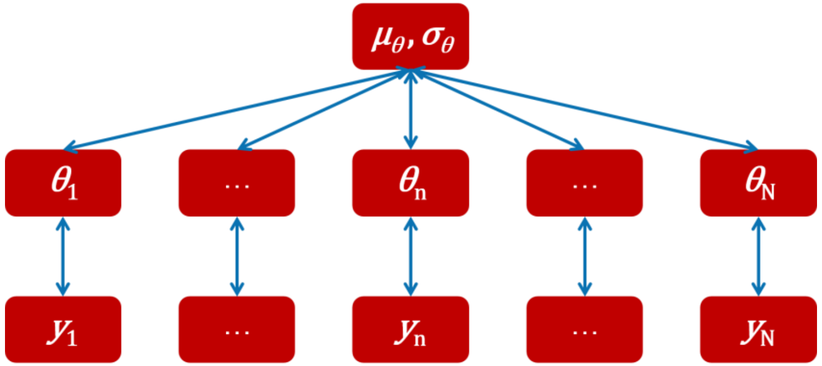
lei.zhang@univie.ac.at
lei-zhang.net
@lei_zhang_lz

https://github.com/lei-zhang/tewa1_univie

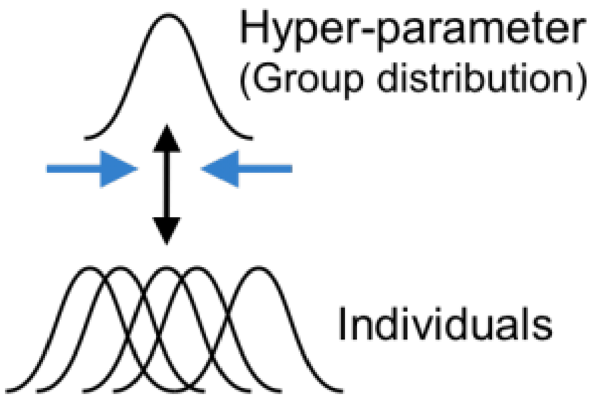
universität wien
Fakultät für Psychologie

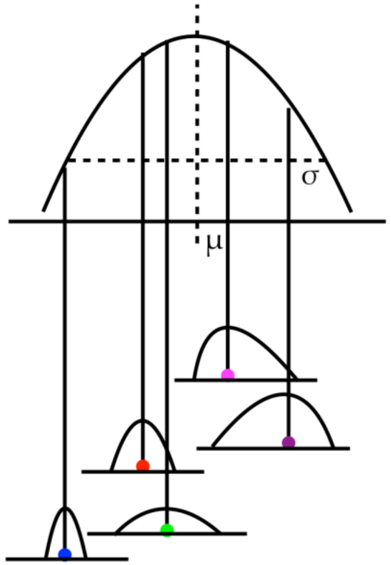
Bayesian warm-up?

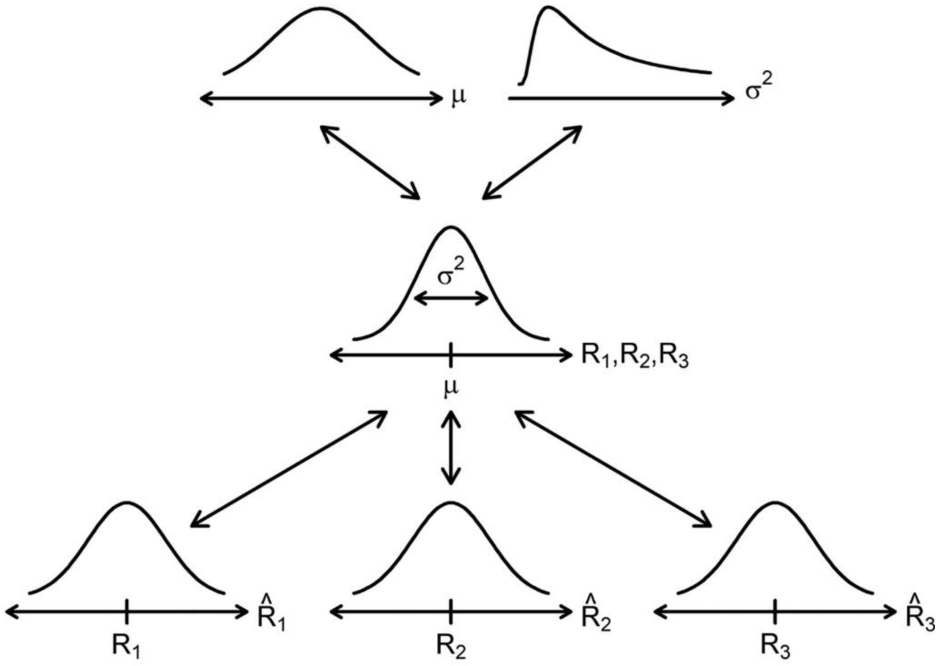
Hierarchical Structure



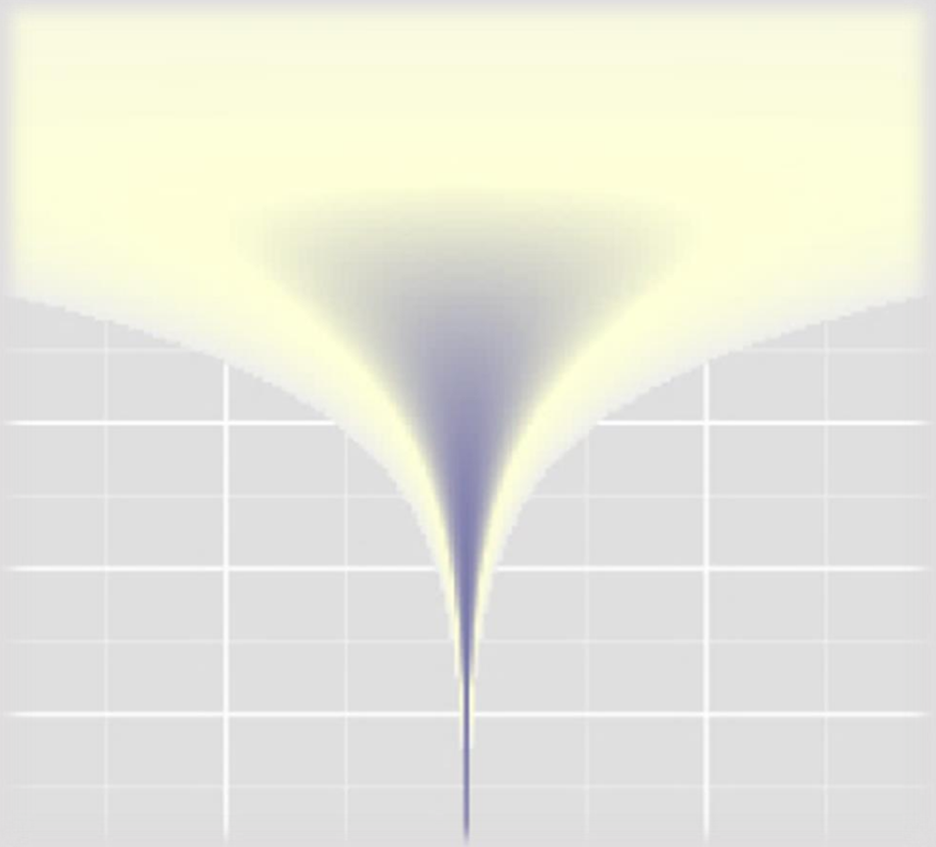
$$P(\Theta,\Phi \mid D) = \frac{P(D \mid \Theta,\Phi)P(\Theta,\Phi)}{P(D)} \propto P(D \mid \Theta)P(\Theta \mid \Phi)P(\Phi)$$

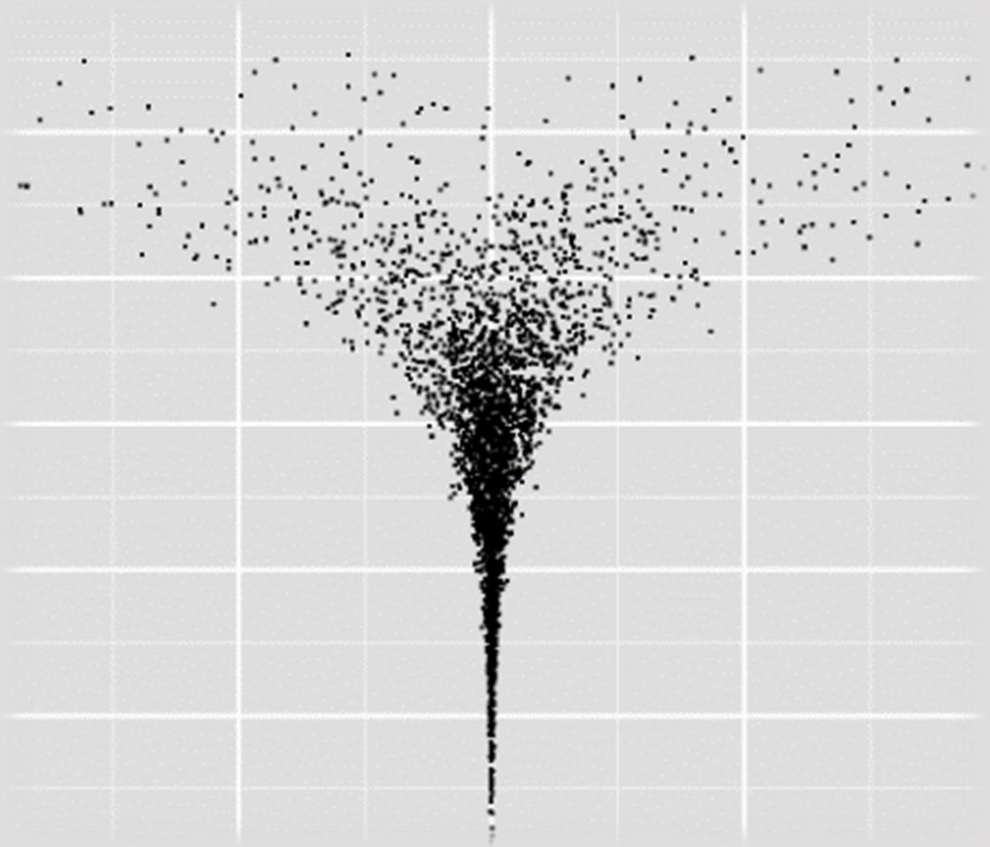




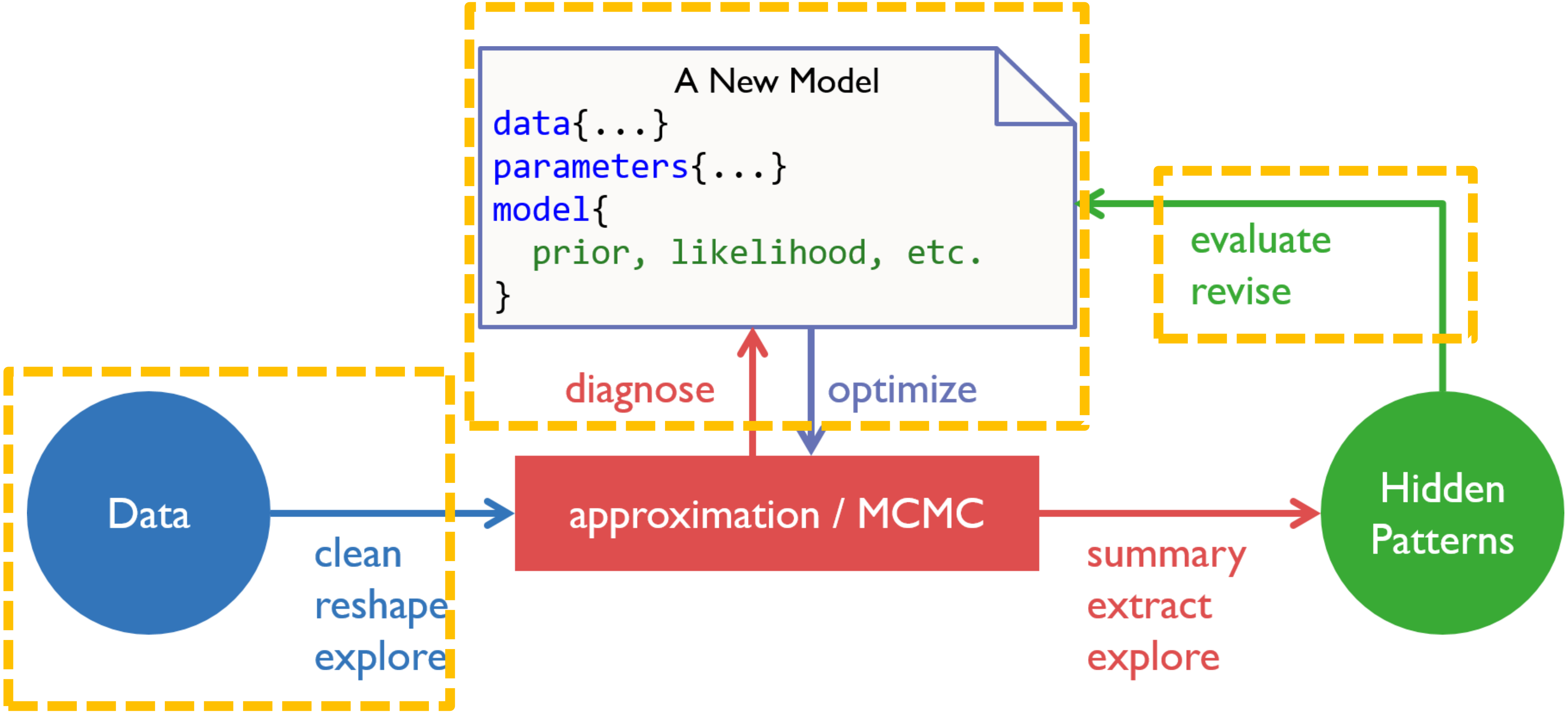


OPTIMIZING STAN CODES





cognitive model
statistics
computing



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

$$\overline{\text{height}} = \alpha + \beta 1 * \text{weight} + \beta 2 * \text{weight}^2$$

```
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

- 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


```
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);
}
```

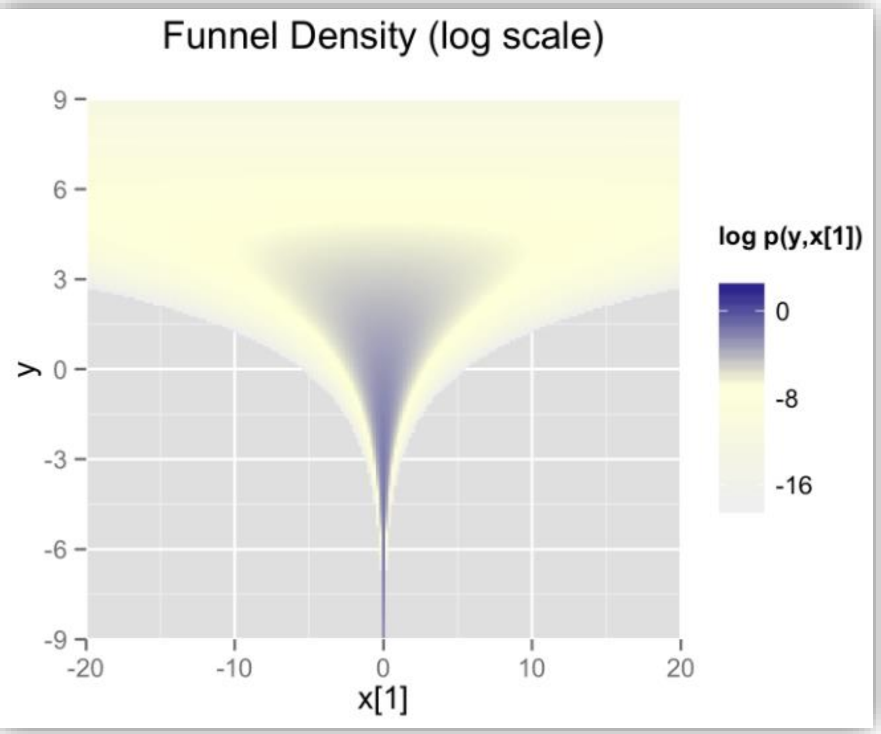
```
model {
  height ~ normal(alpha + beta * weight, sigma);
}
```

Reparameterization

Neal's Funnel

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

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



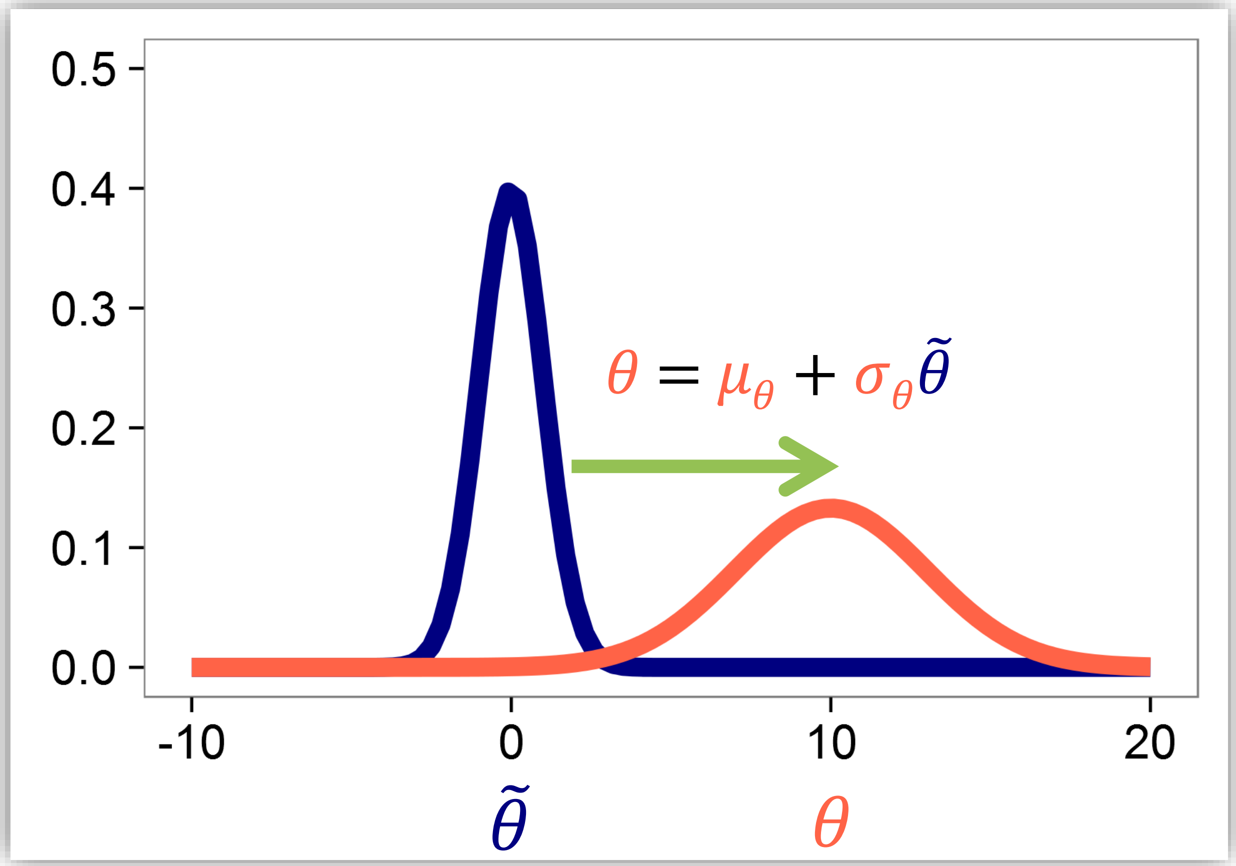
Non-centered Reparameterization*

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!



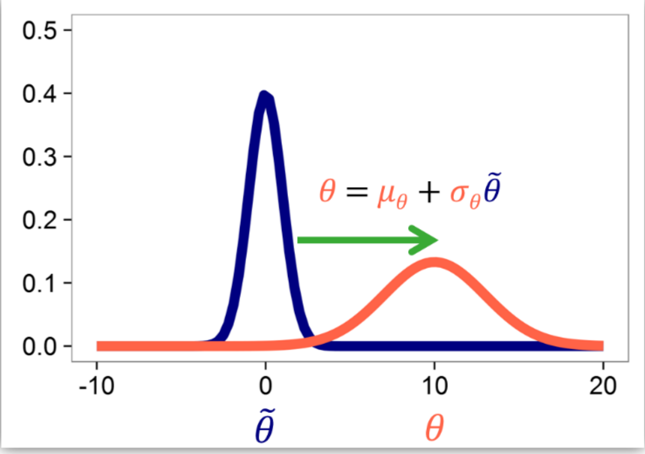
*:Stan Development Team (2016), aka, Matt Trick

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

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

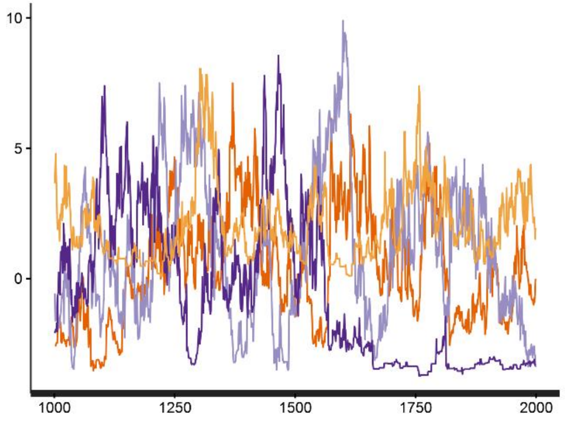
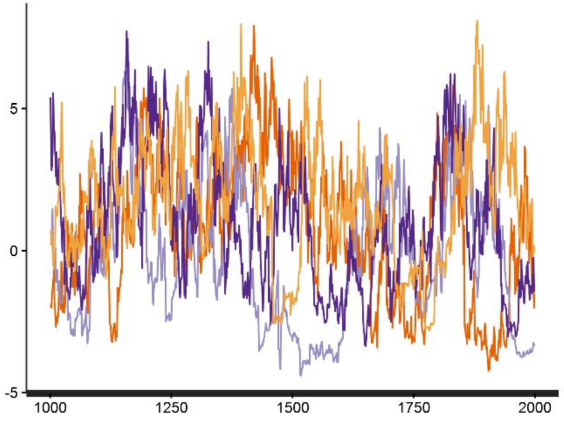
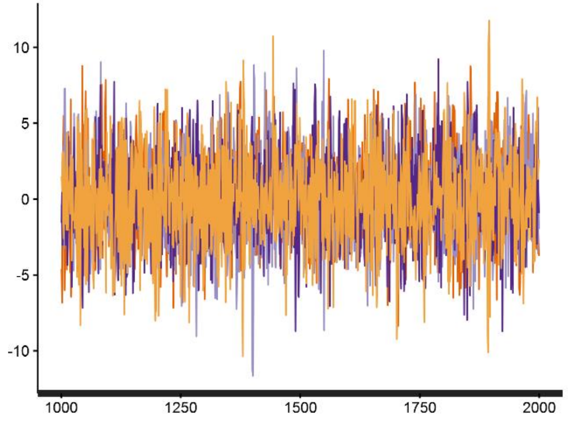
Typical adjustments

- 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))
```

Stan Development Team (2016)

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?

$$\tilde{\theta} \sim Normal(0,1)$$
$$\theta = \mu_\theta + \sigma_\theta \tilde{\theta}$$
$$\theta \in (-\infty, +\infty)$$

$$\tilde{\theta} \sim Normal(0,1)$$
$$\theta = Probit^{-1}(\mu_\theta + \sigma_\theta \tilde{\theta})$$
$$\theta \in [0,1]$$

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

* $Probit^{-1}$: 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


```
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


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

TASK: (1) Complete the Matt Trick

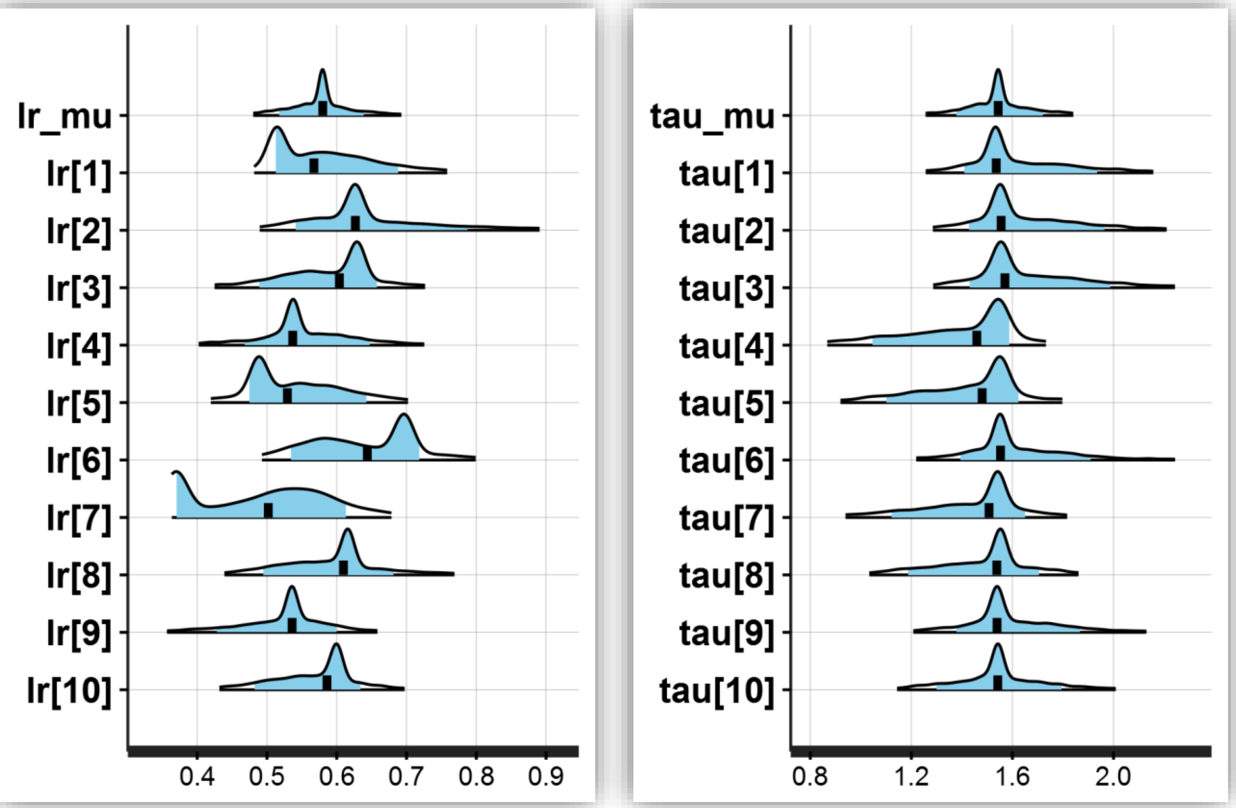
(2) fit the optimized hierarchical RL model

```
> source('_scripts/reinforcement_learning_hrch_main.R')

> fit_rl4 <- run_rl_mp2(optimized = TRUE)
```

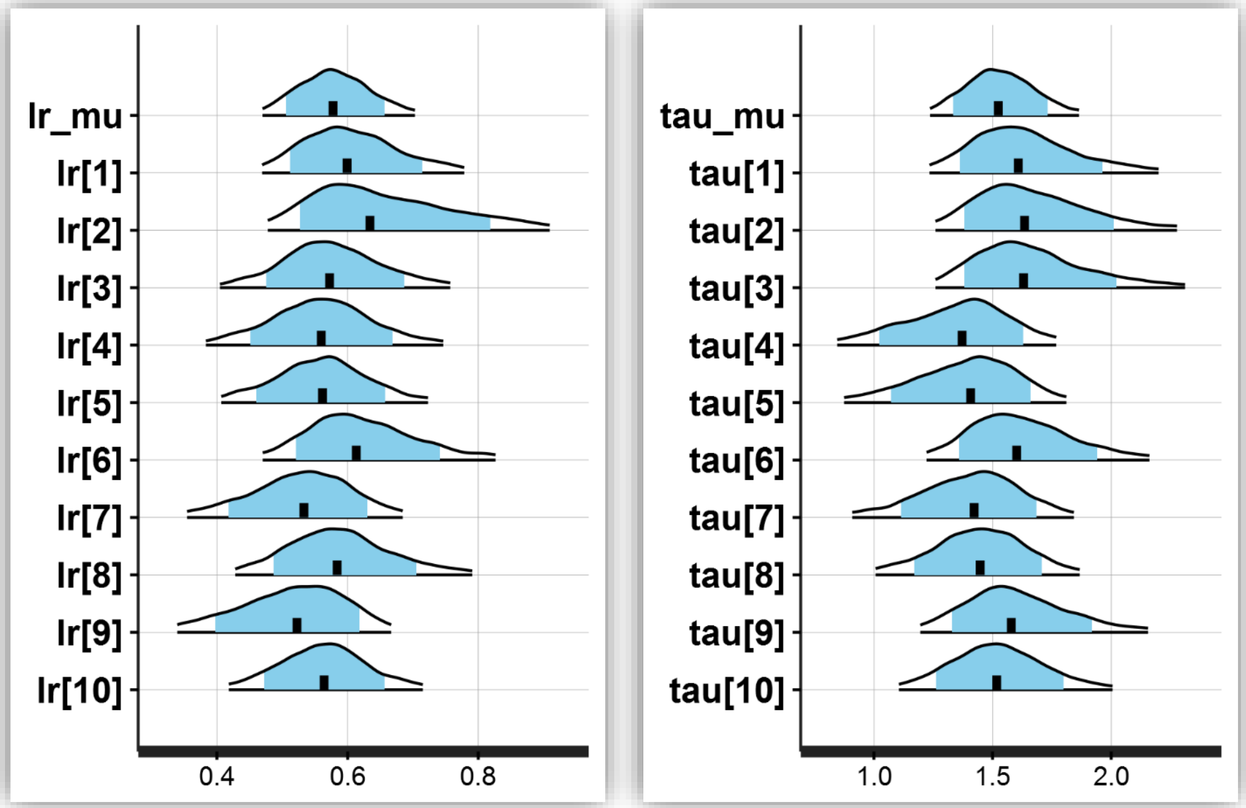
Hierarchical Fitting – Optimized

cognitive model
statistics
computing

Posterior Means (hrch)



Posterior Means (hrch + optm)



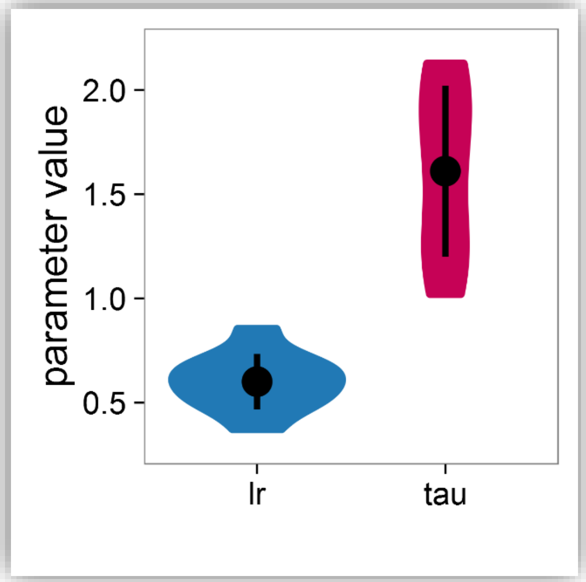
Comparing with True Parameters

cognitive model

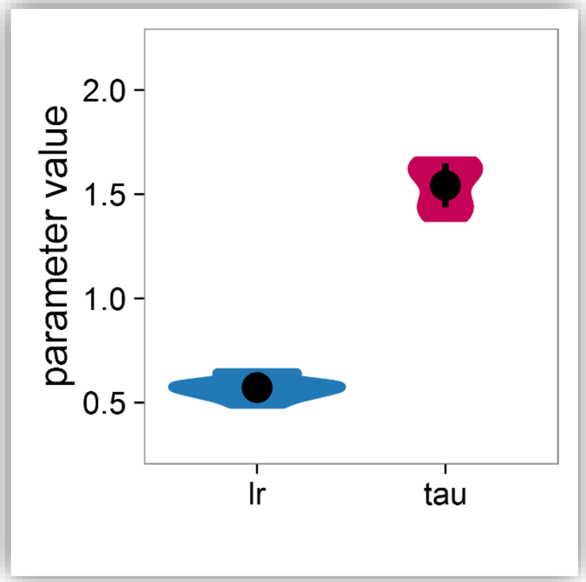
statistics

computing

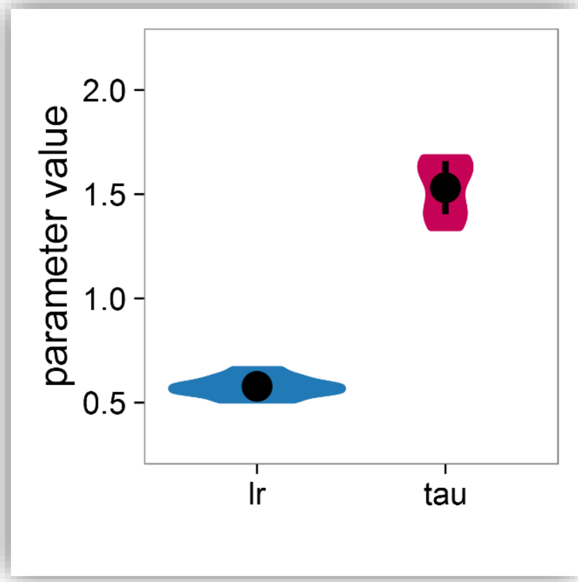
Posterior Means (indv)



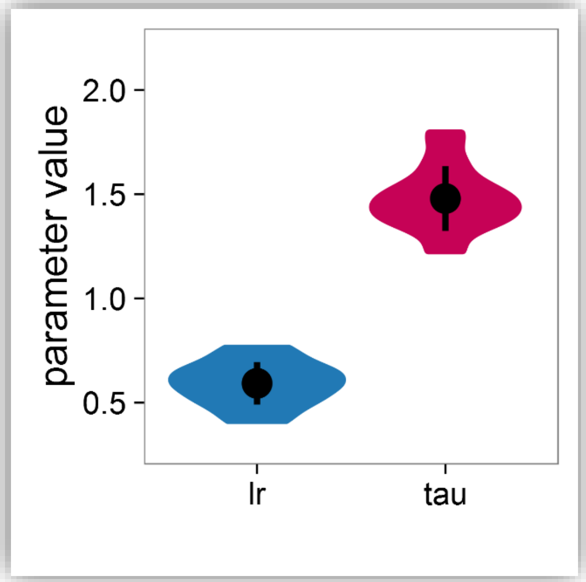
Posterior Means (hrch)

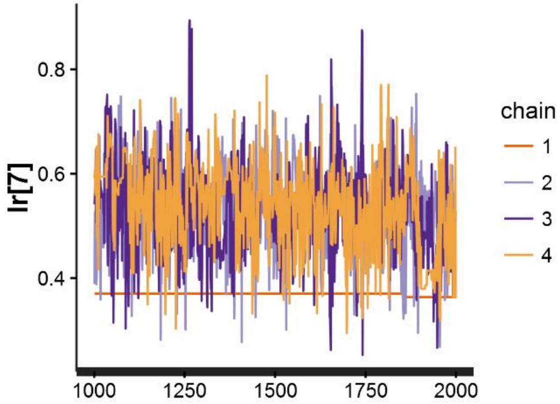


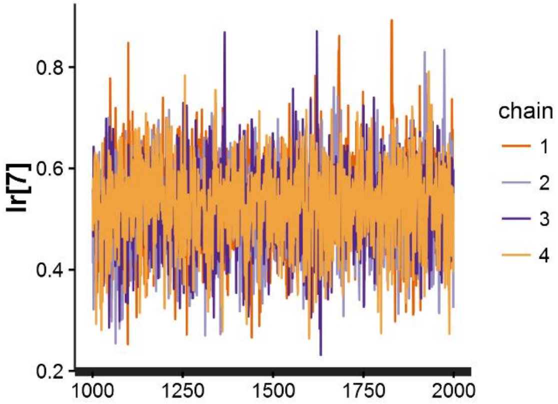
Posterior Means (hrch+optm)



True Parameters

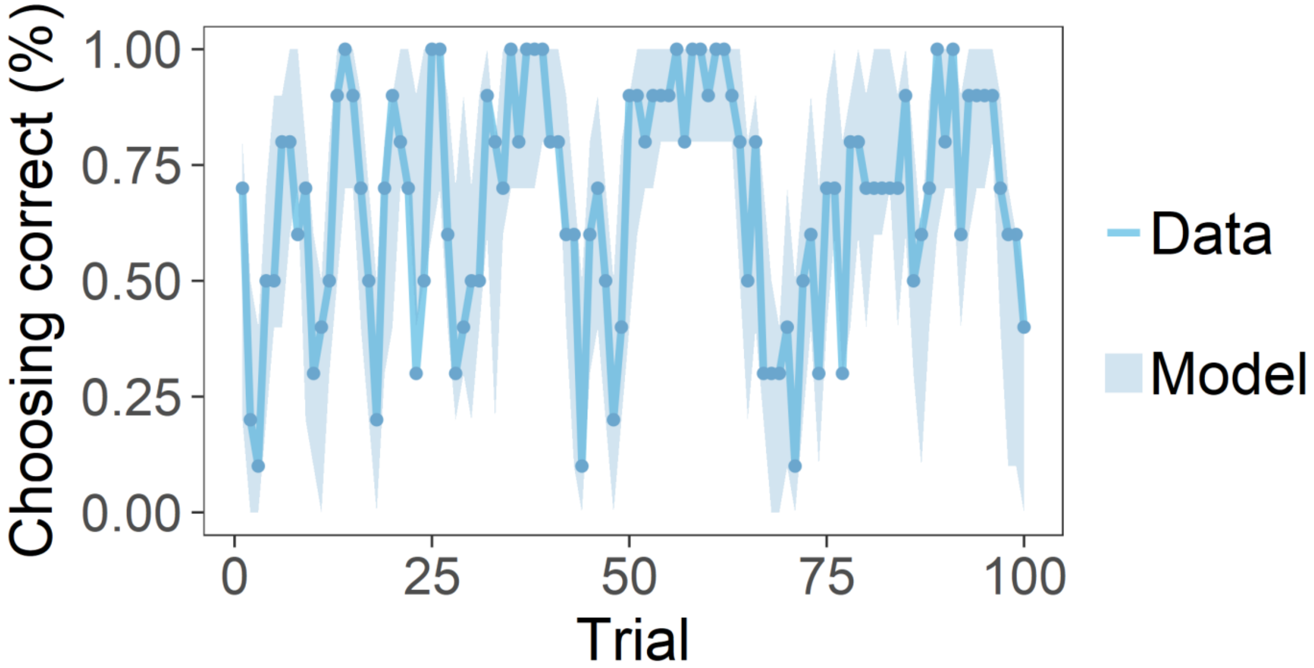


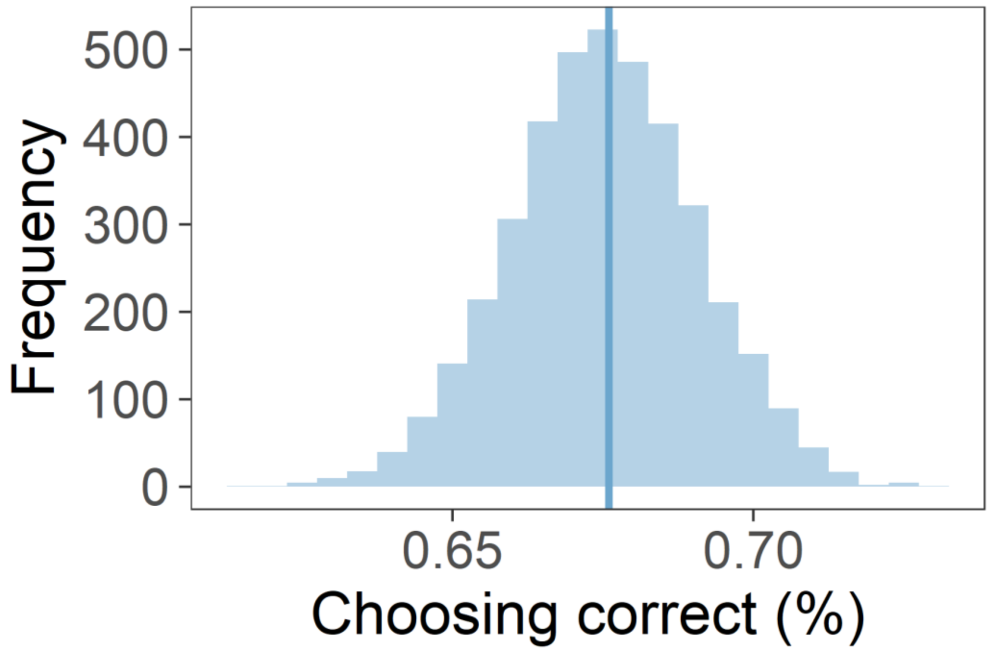


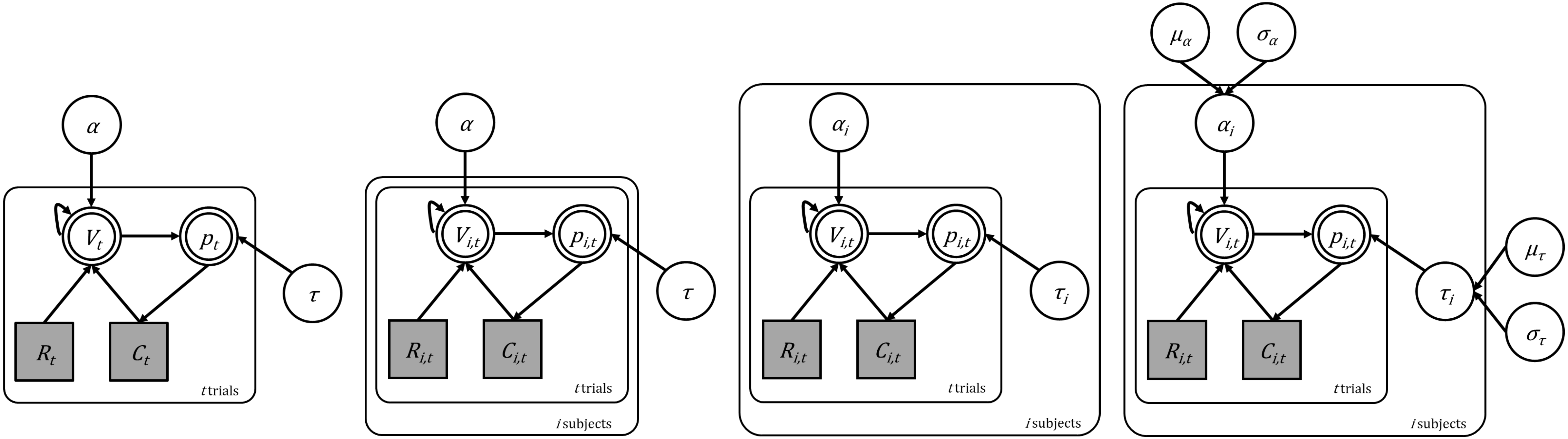


Posterior Predictive Check

cognitive model
statistics
computing

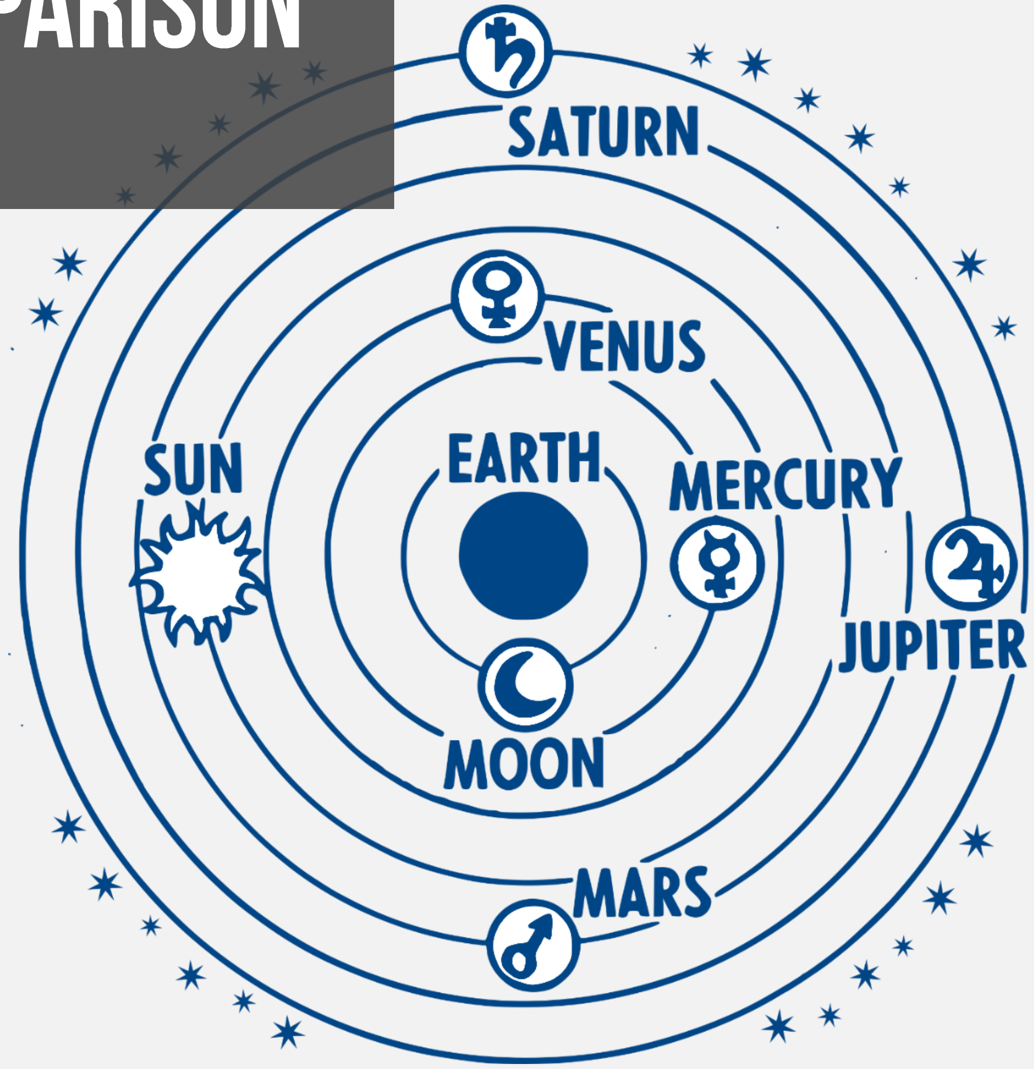


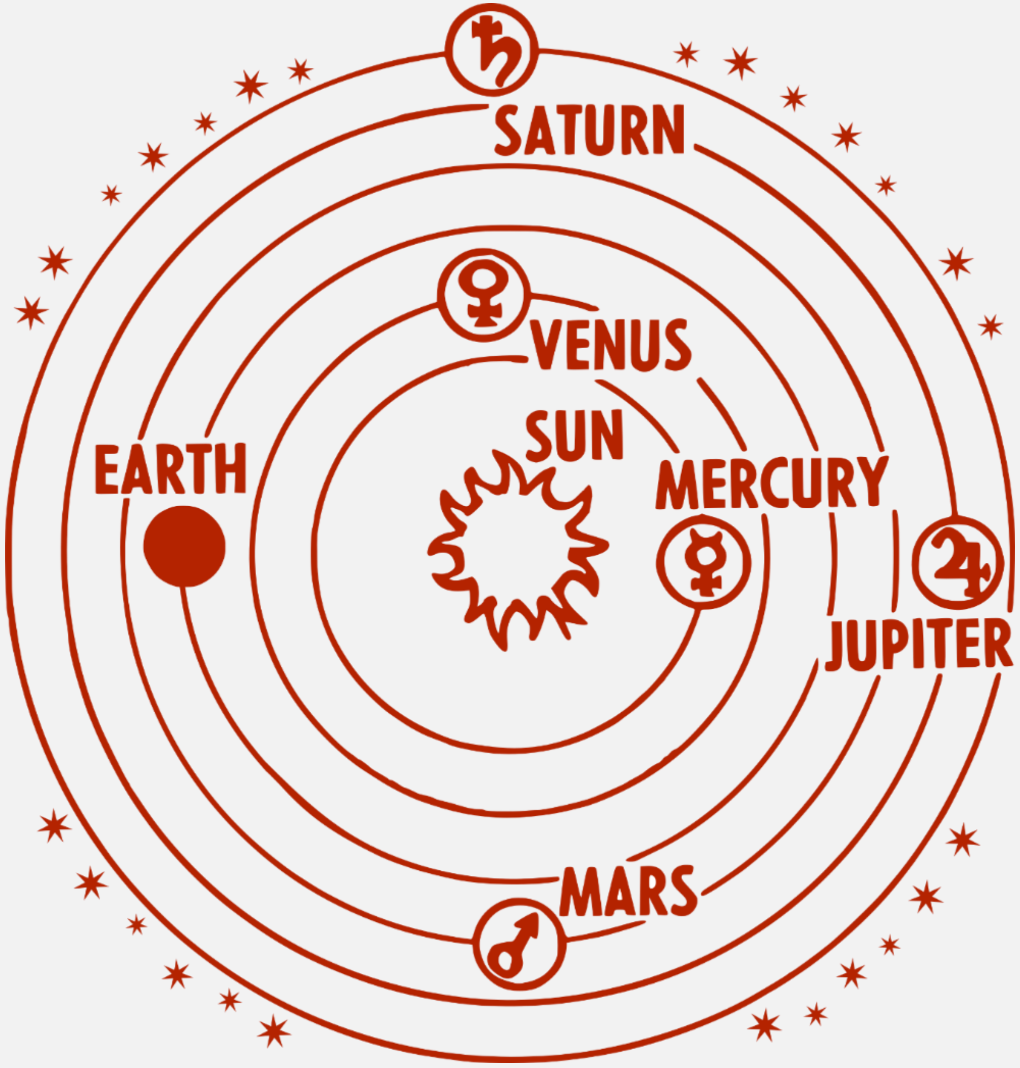




constraint	**reparameterization**
$\theta \in (-\infty, +\infty)$	$\theta = \mu_\theta + \sigma_\theta \tilde{\theta}$
$\theta \in [0, N]$	$\theta = Probit^{-1}(\mu_\theta + \sigma_\theta \tilde{\theta}) \times N$
$\theta \in [M, N]$	$\theta = Probit^{-1}(\mu_\theta + \sigma_\theta \tilde{\theta}) \times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

MODEL COMPARISON

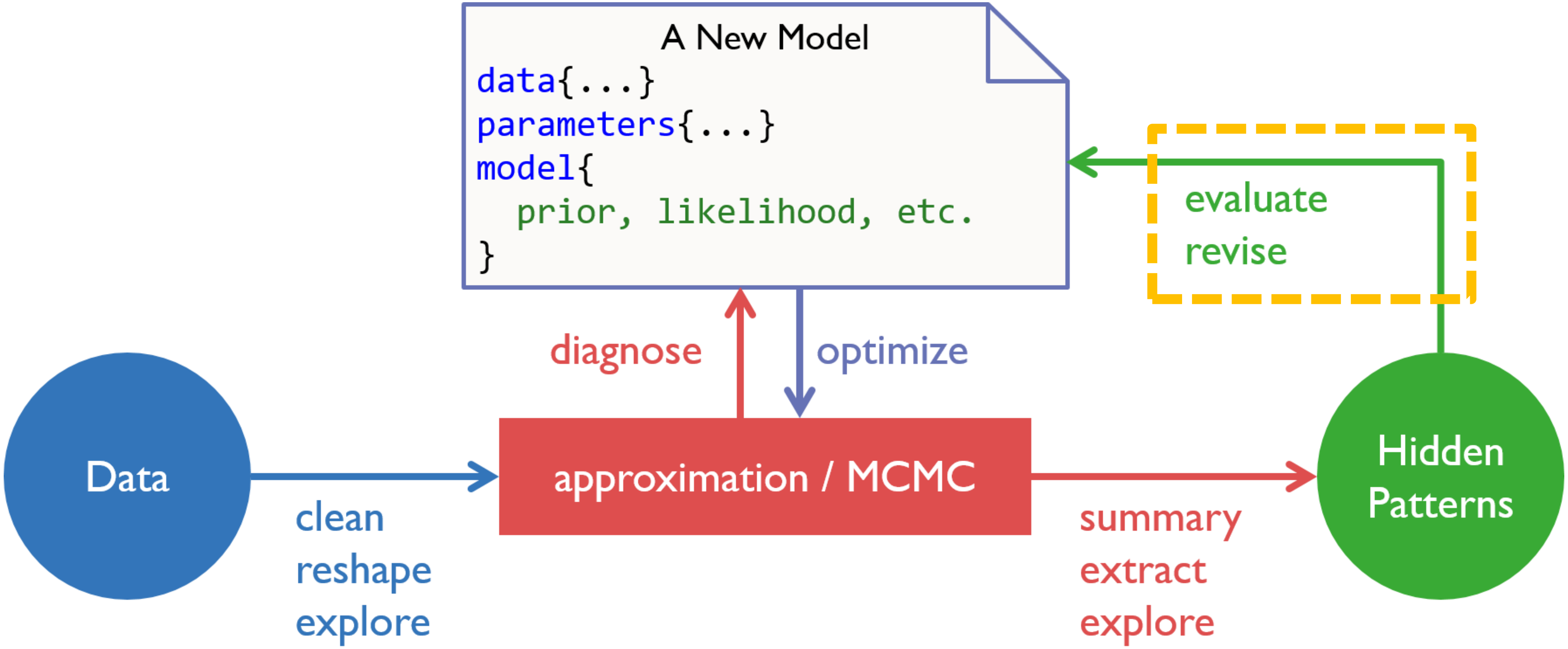




cognitive model

statistics

computing



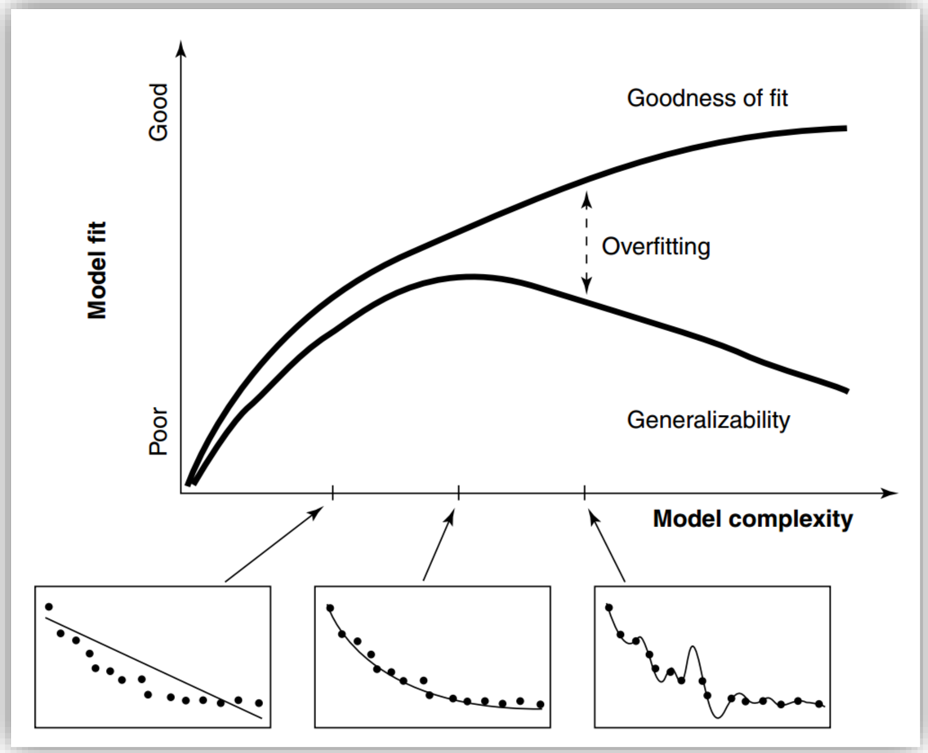
Model Comparison

cognitive model
statistics
computing

Which model provides the best fit?

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

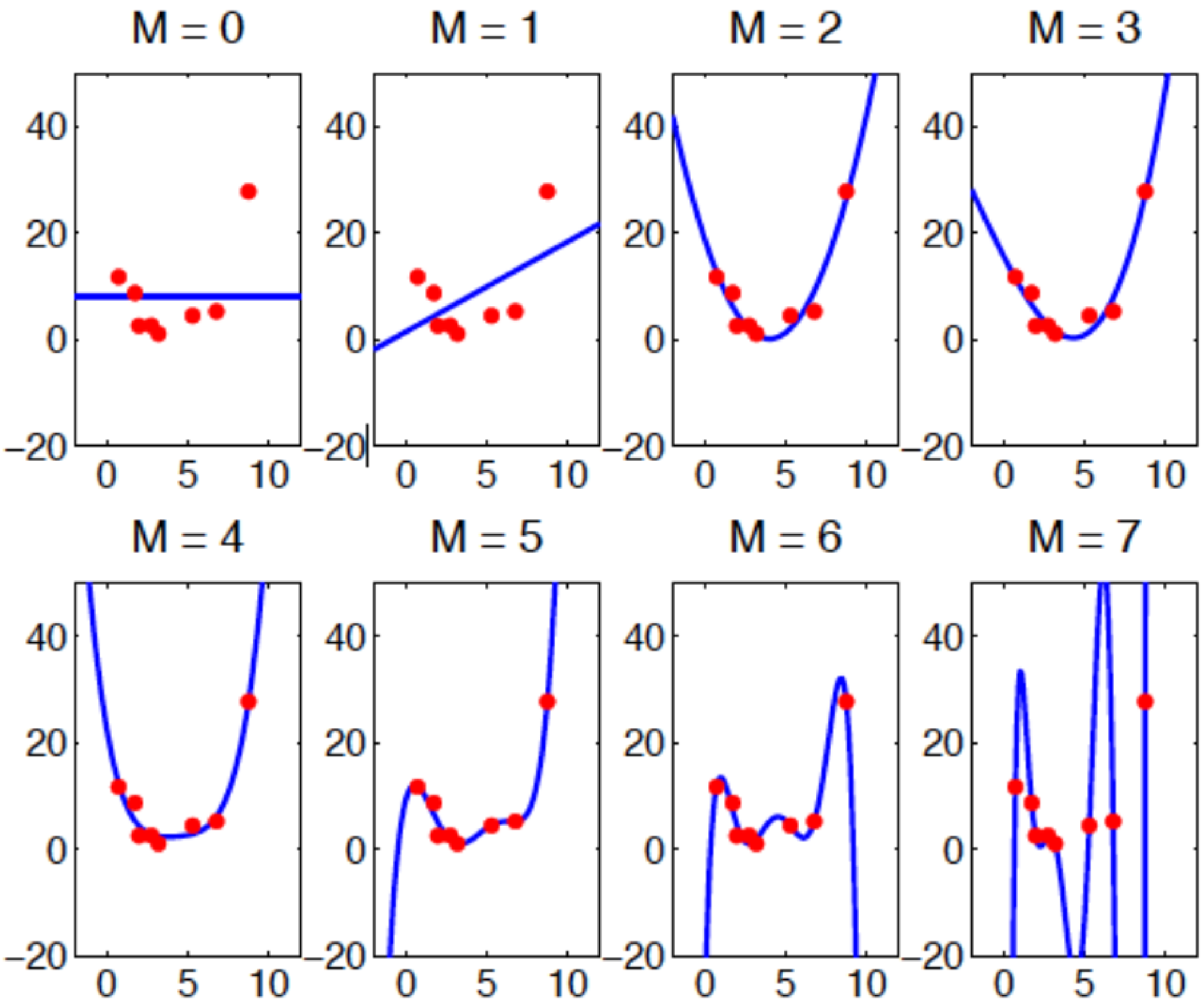
Ockham's razor:
Models with fewer assumptions are to be preferred



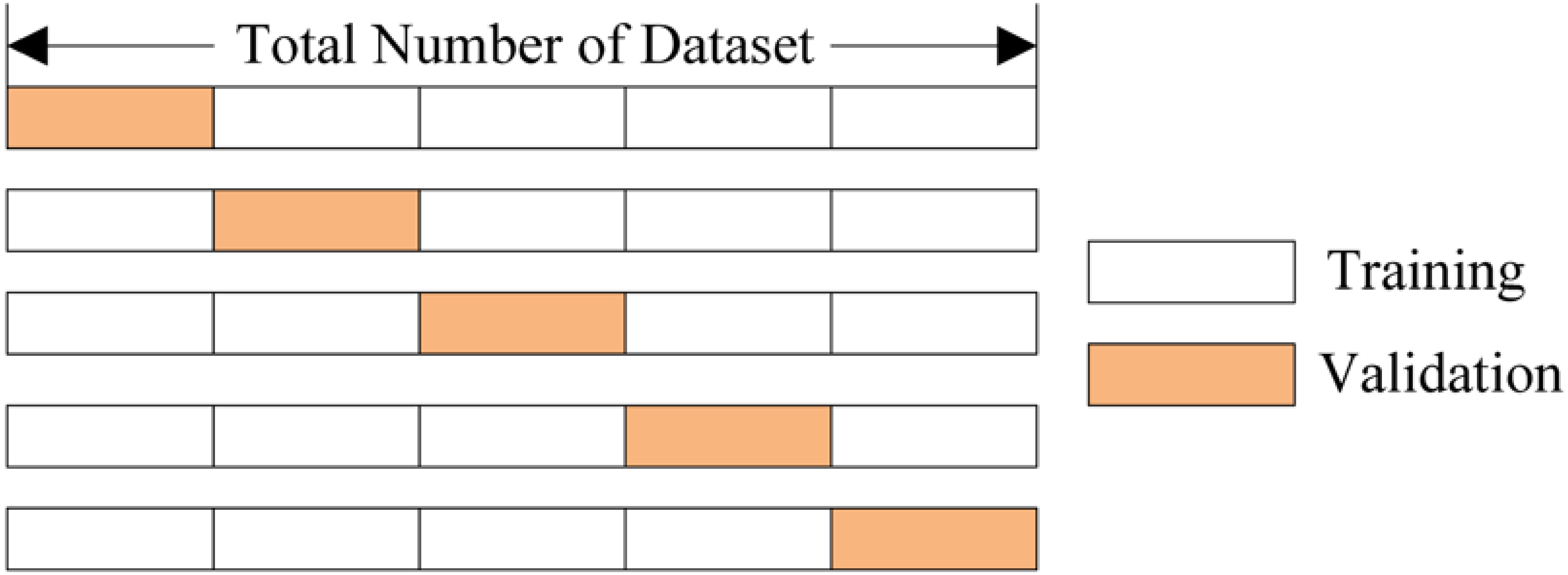
- overfitting: learn too much from the data
- underfitting: learn too little from the data

Pitt & Miyung (2002)

Which model has the highest predictive power?



Focusing on Predictive Accuracy



- 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

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

approximation to LOO

BIC – Bayesian Information Criterion

finding the "true" model

Compute WAIC from Likelihood

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

expected log pointwise predictive density

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

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

estimated effective number of parameters

$$\widehat{p}_{\text{waic}} = \sum_{i=1}^{n} V_{s=1}^{S}\left(\log p(y_i|\theta^s)\right)$$

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

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

Vehtari & Gelman (2014)

*IC comparisons

		No pooling ($\tau = \infty$)	Complete pooling ($\tau = 0$)	Hierarchical model (τ 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	$-2\,\text{lppd}$		59.8	59.2
	$p_{\text{loo-cv}}$		0.5	1.8
	$-2\,\text{lppd}_{\text{loo-cv}}$		60.8	62.8

Gelman et al. (2013)

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

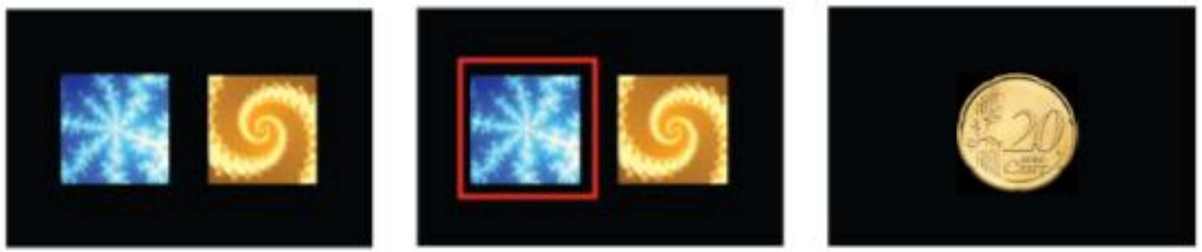
```
> library(loo)
> LL1   <- extract_log_lik(stanfit)
> loo1  <- loo(LL1)    # PSIS leave-one-out
> waic1 <- waic(LL1)   # WAIC
```

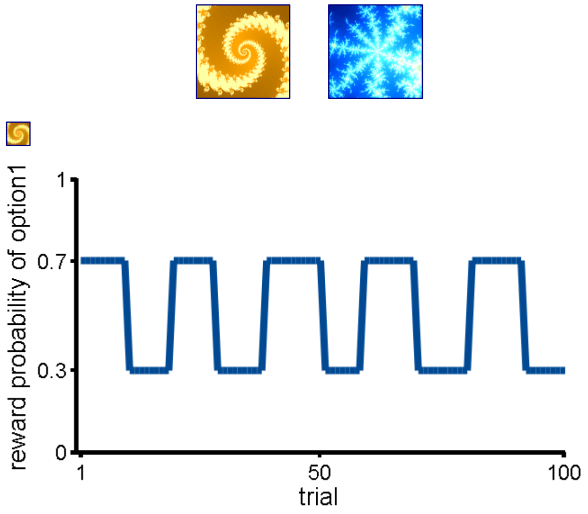
Pareto Smoothed Importance Sampling

```
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
```

Vehtari et al. (2015)

Reversal Learning Task





Fictitious RL
(Counterfactual RL)

Value update:

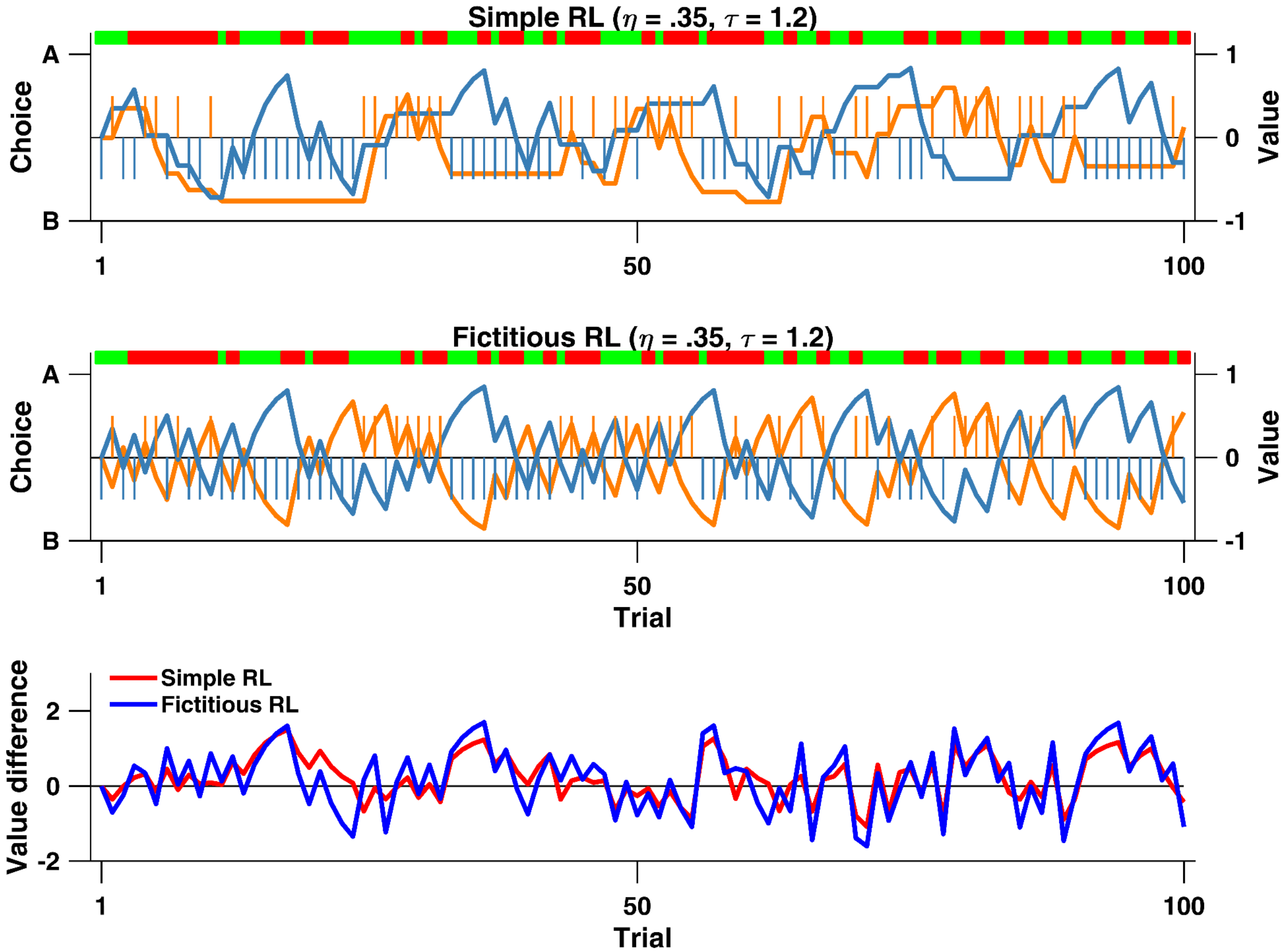
$$V_{t+1}^{c} = V_t^{c} + \alpha * PE$$
$$V_{t+1}^{nc} = V_t^{nc} + \alpha * PEnc$$

Prediction error:

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

Hampton et al. (2007)
Gläscher et al. (2009)

More on Fictitious RL



Exercise XIII

cognitive model

statistics

computing

`…/08.compare_models/_scripts/compare_models_main.R`

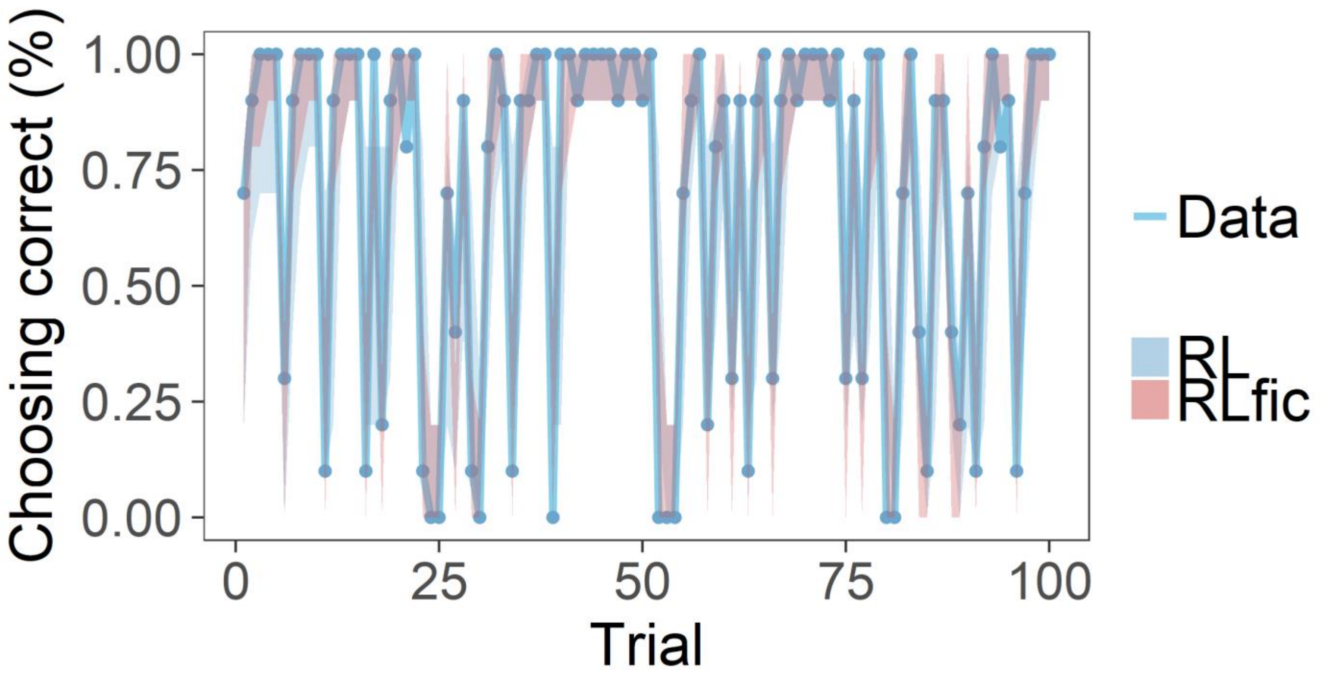
TASK: (1) complete the fictitious RL model (model2, `loglik`)

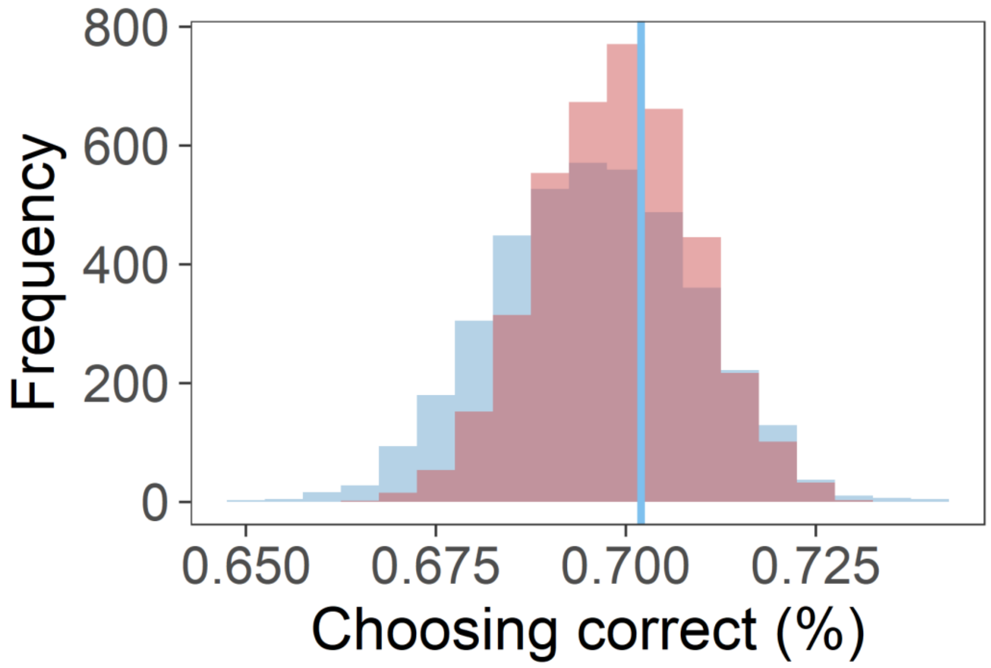
(2) fit and compare the 2 models

Exercise XIII – output


```
> LL1 <- extract_log_lik(fit_rl1)
> ( 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






ANY
QUESTIONS
?

Happy Computing!