Reinforcement Learning & Decision-Making with hBayesDM

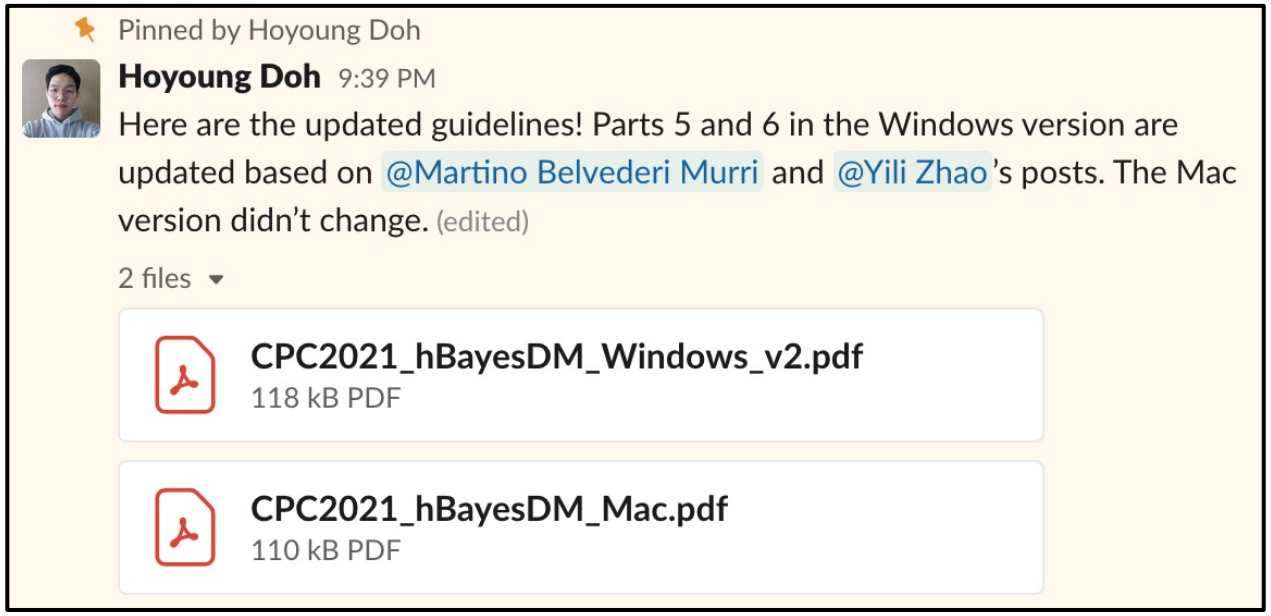
Woo-Young (Young) Ahn
Department of Psychology
Seoul National University
ccs-lab.github.io

Installation for R (recommended for the workshop!)

- *Please check out rl-hbayesdm channel in Slack!*

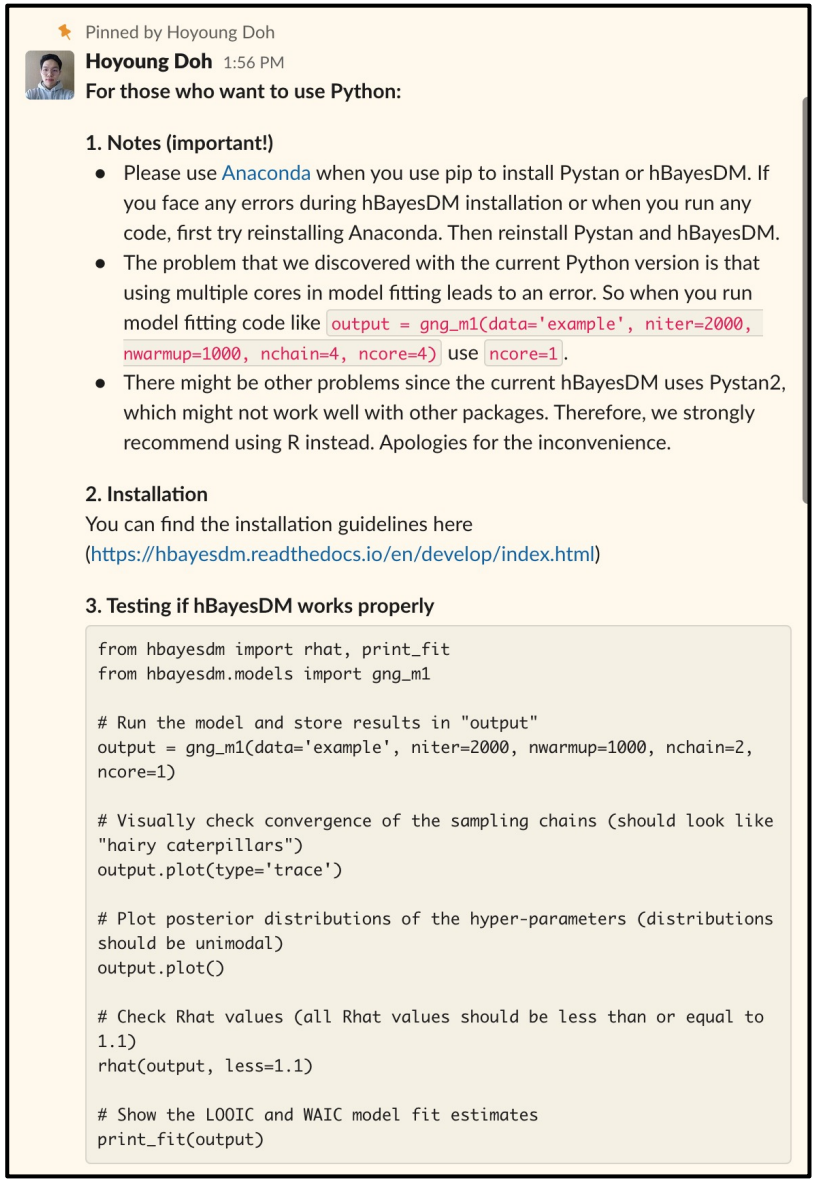
- *For MAC users:*
  - *CPC2021_hBayesDM_Mac.pdf*

- *For Windows users:*
  - *CPC2021_hBayesDM_Windows.pdf*



Installation for Python

- *Please make sure to use Anaconda*
- *Use Pystan 2.19.1.1 (instead of the latest Pystan 3.x)*
- *https://hbayesdm.readthedocs.io/en/develop/index.html*



Pinned by Hoyoung Doh

Hoyoung Doh 1:56 PM

For those who want to use Python:

1. Notes (important!)

- Please use Anaconda when you use pip to install Pystan or hBayesDM. If you face any errors during hBayesDM installation or when you run any code, first try reinstalling Anaconda. Then reinstall Pystan and hBayesDM.
- The problem that we discovered with the current Python version is that using multiple cores in model fitting leads to an error. So when you run model fitting code like `output = gng_m1(data='example', niter=2000, nwarmup=1000, nchain=4, ncore=4)` use `ncore=1`.
- There might be other problems since the current hBayesDM uses Pystan2, which might not work well with other packages. Therefore, we strongly recommend using R instead. Apologies for the inconvenience.

2. Installation

You can find the installation guidelines here (https://hbayesdm.readthedocs.io/en/develop/index.html)

3. Testing if hBayesDM works properly

```
from hbayesdm import rhat, print_fit
from hbayesdm.models import gng_m1

# Run the model and store results in "output"
output = gng_m1(data='example', niter=2000, nwarmup=1000, nchain=2,
ncore=1)

# Visually check convergence of the sampling chains (should look like
"hairy caterpillars")
output.plot(type='trace')

# Plot posterior distributions of the hyper-parameters (distributions
should be unimodal)
output.plot()

# Check Rhat values (all Rhat values should be less than or equal to
1.1)
rhat(output, less=1.1)

# Show the LOOIC and WAIC model fit estimates
print_fit(output)
```

Learning objectives

Participants will…

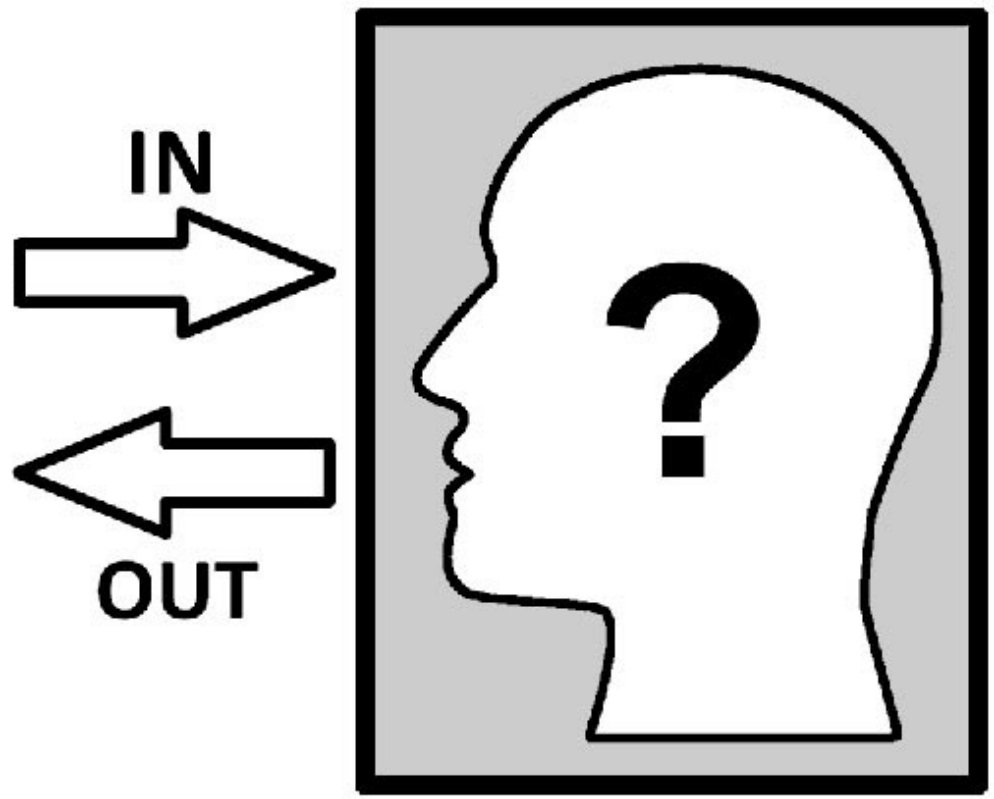
- *Know popular and cutting-edge methods for fitting computational models*
- *Review the concept of Bayesian data analysis*
- *Evaluate outputs from Bayesian data analysis*
- *Use hBayesDM package to model reinforcement learning and decision-making tasks*

Outline for Part I

- *What is computational modeling (a.k.a. cognitive modeling)?*
- *Why/how do we lower the barrier to computational modeling?*
  - *Brief introduction to hBayesDM*
- *How to fit a computational model?*
  - *Maximum likelihood estimation (MLE)*
  - *Bayesian analysis & MCMC sampling*
  - *Hierarchical Bayesian analysis*
  - *Tools for Bayesian data analysis*
- *Things to know when performing MCMC sampling*

Outline

- *Understand the concepts of computational modeling*
- *Know popular and cutting-edge methods for fitting computational models / Understand the concept of Bayesian data analysis*
- *Things to know when performing MCMC sampling*
- *Use hBayesDM package to model several tasks*

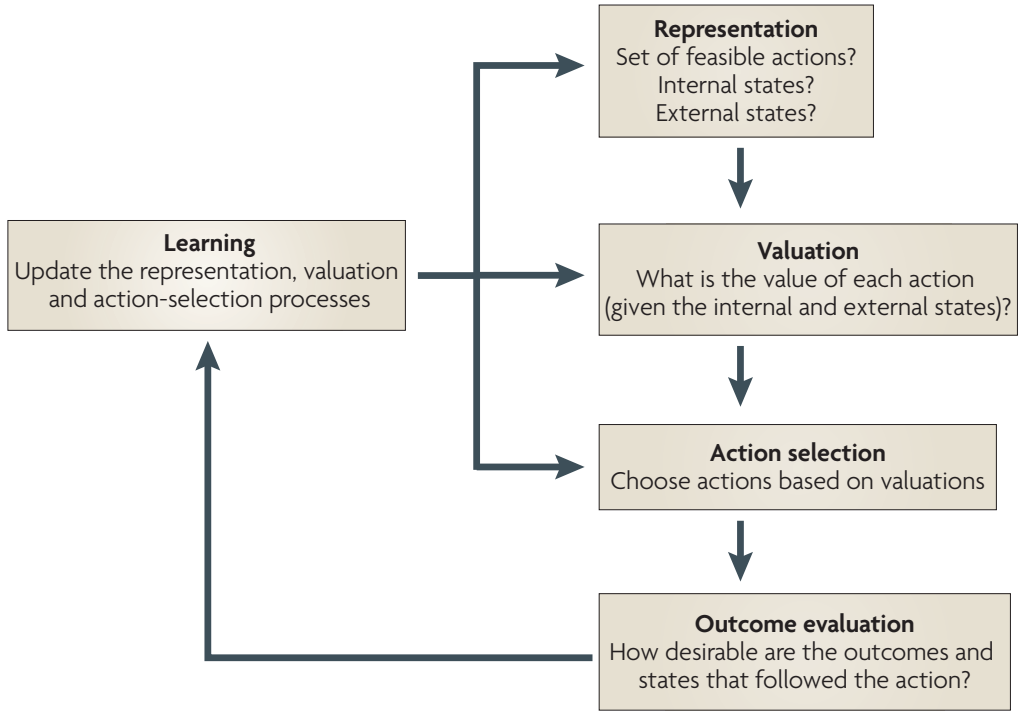

IN
OUT
?

Reinforcement learning and Decision-making (RLDM)

A framework for studying the neurobiology of value-based decision making

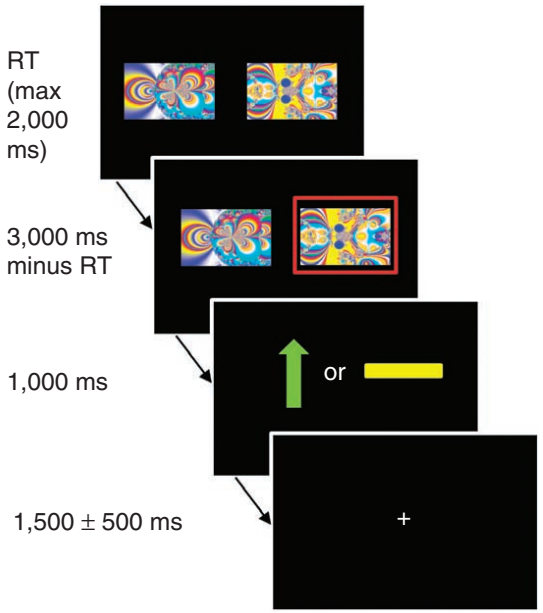
Antonio Rangel, Colin Camerer* and P. Read Montague‡*

Rangel et al (2008) Nature Rev. Neuro

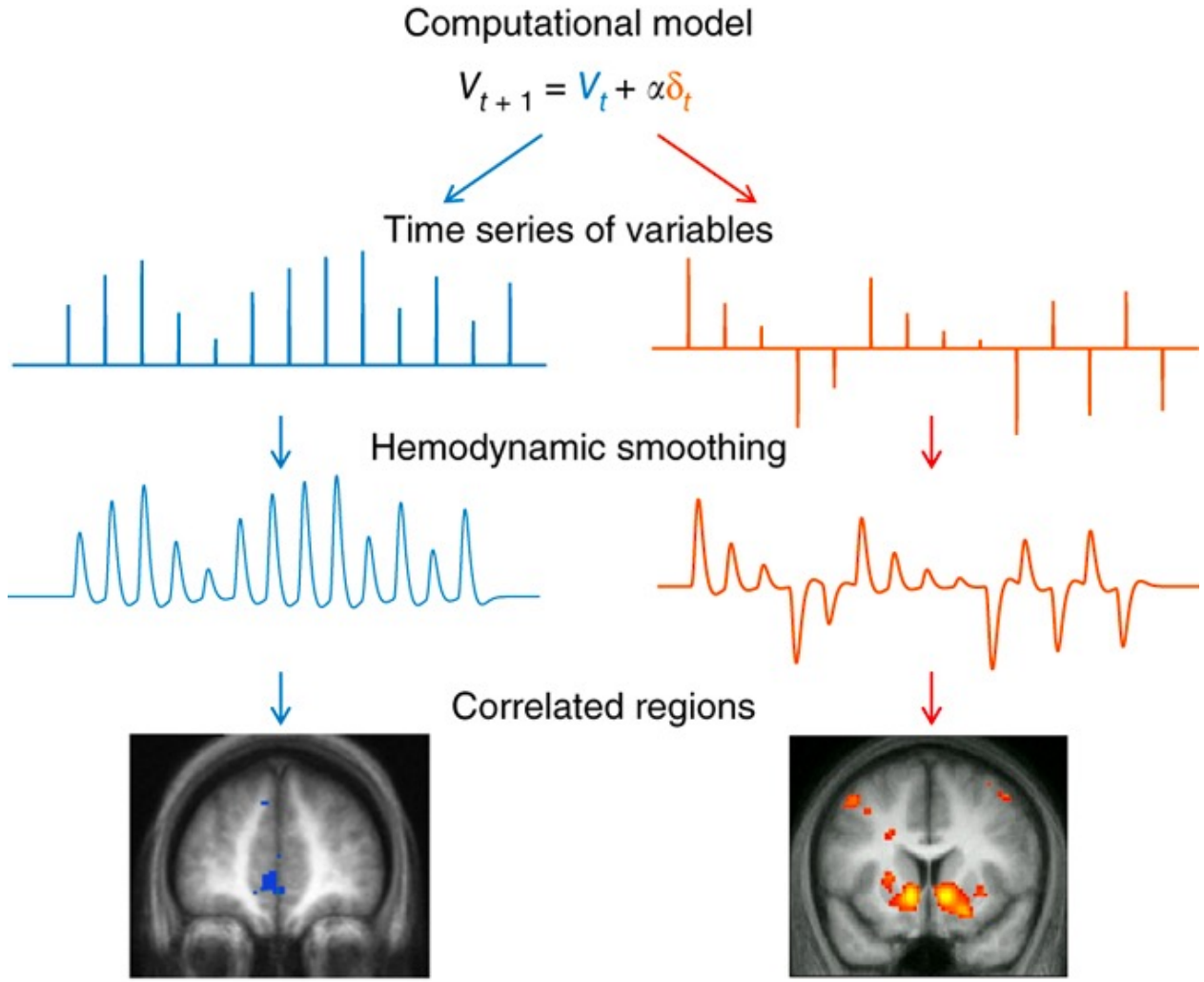


Model-based fMRI/EEG

e.g., Forstmann & Wagenmakers (2015); O'Doherty et al (2007); Cohen et al (2017)



Chowdhury et al (2013) Nature neuro.



Cohen et al (2017) Nature neuro.

Review

Review

Cell PRESS

Special Issue: Cognition in Neuropsychiatric Disorders

Computational psychiatry

P. Read Montague[1,2], Raymond J. Dolan[2], Karl J. Friston[2] and Peter Dayan[3]

CrossMark

Computational psychiatry: the brain as a phantastic organ

Karl J Friston, Klaas Enno Stephan, Read Montague, Raymond J Dolan

Available online at www.sciencedirect.com

ScienceDirect

Current Opinion in Neurobiology

ELSEVIER

Computational approaches to psychiatry

Klaas Enno Stephan[1,2,3] and Christoph Mathys[3]

Computational accounts of abnormal cognition & its biological underpinnings

Limitations

- *Overly simplified "toy" problems*
  - *Violated assumptions (e.g., discrete space/action & Markov..) , (Gershman & Daw, 2016, Annu Rev Psych)*
- *One-shot learning with sparse data*
  - *Episodic memory (hippocampus) (Gabrieli, 1998; Eichenbaum et al., 1999)*
- *Modeling of even toy problems is hard for many people*

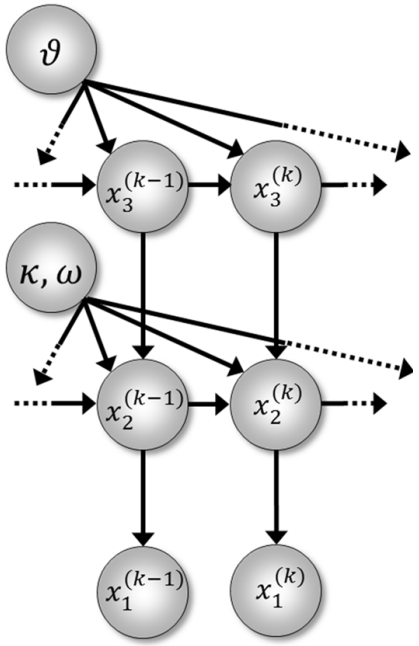
Steps of computational modeling

Busemeyer and Diederich (2010)

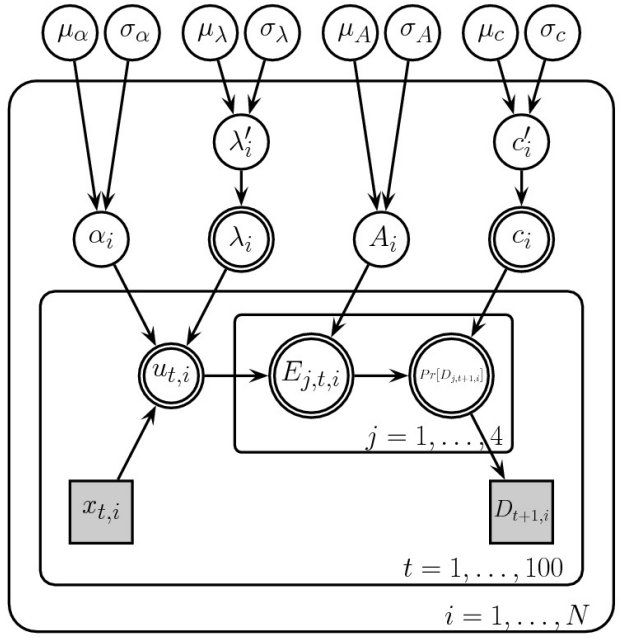
- *Formulate assumptions into a mathematical language*
- *Estimate "free" parameters*
- *Model comparisons*
  - *Compare predictions of competing models in terms of their ability to explain empirical data*
- *Reformulate and construct new models in the light of the feedback…*
- *Use computational modeling for the analysis of neuroimaging data (e.g., model-based fMRI)*
- *"All models are wrong but some are useful"* *Box (1976)*

I like the idea of modeling

But…



Mathys et al (2011) Frontiers

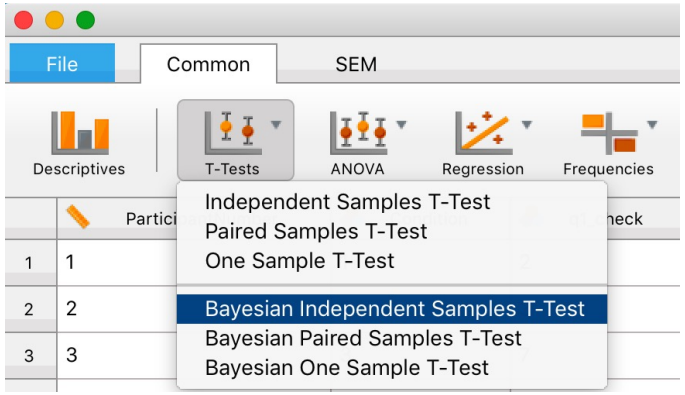


Ahn et al (2011) JNPE

Can we make it easy to do computational modeling?

Q) As easy as doing a T-test?

JASP, SPSS



R

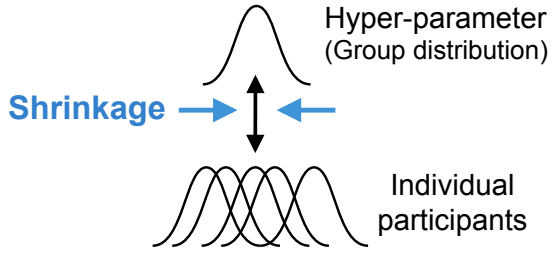
Several packages exist, but …

Matze et al (2013) Frontiers; Wabersich & Vandekerckhove (2014); Wiecki et al (2013) Frontiers; Daunizeau et al (2014) PLoS Comp Biol

hBayesDM *(hierarchical Bayesian modeling of Decision-Making tasks) Package*

- *Models for "many" tasks/paradigms (next slide)*
- *Single-line of coding in R*
  - *Model fitting, visualization, model comparisons*
- *Based on the advanced Bayesian software, Stan (https://mc-stan.org).*
- *Hierarchical Bayesian modeling*
- *All codes are publicly available*

https://github.com/CCS-Lab/hBayesDM



What tasks and models are available?

in CPC2017

Ahn & Busemeyer (2016) Curr Opin Behav Sci

- *Choice reaction time → sequential sampling models*
- *Delay Discounting* *(e.g., Mazur, 1987)*
- *Iowa Gambling* *(Bechara et al, 1994)*
- *(Orthogonalized) Go/Nogo* *(Guitart-Masip et al, 2012)*
- *Two-armed Bandit (Experience-based) including Reversal Learning* *(e.g., Erev et al, 2010)*
- *Four-armed Bandit (Experience-based)* *(e.g., Seymour et al, 2012)*
- *Two-choice Description-based* *(e.g., Sokol-Hessner et al, 2009; Tom et al, 2007)*
- *Ultimatum Game* *(e.g., Xiang et al, 2013)*

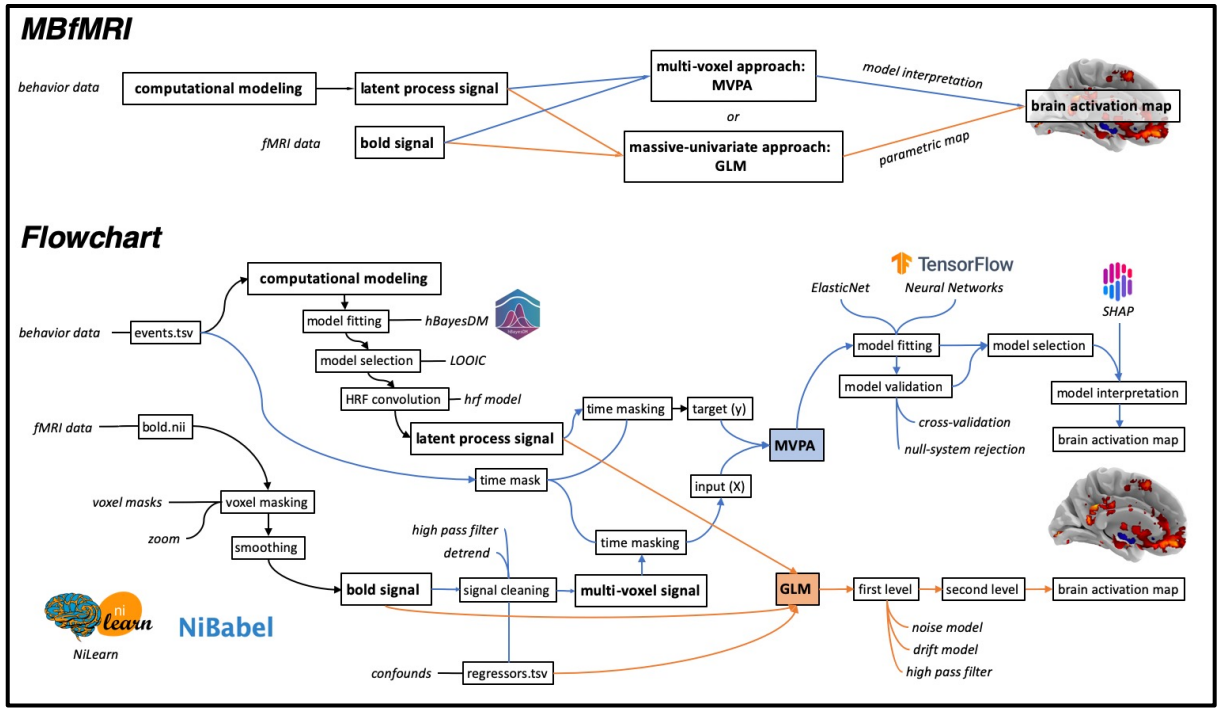
What tasks and models are available in CPC2021?

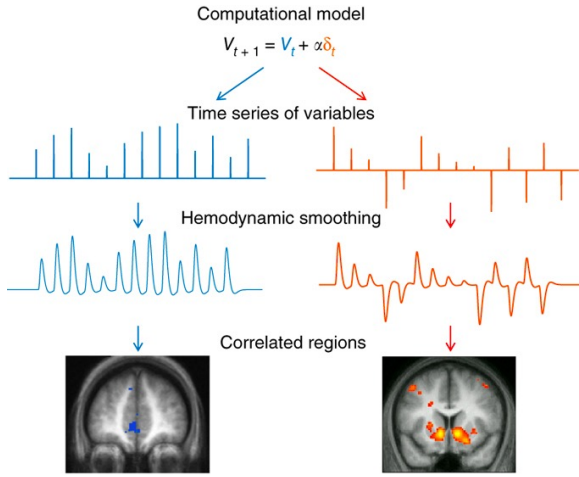
Ahn & Busemeyer (2016) Curr Opin Behav Sci

- *Aversive Learning Task (develop branch)* (Browning et al, 2015)
- *Balloon Analogue Risk Task (BART)* (Wallsten et al, 2005)
- *Choice under Risk and Ambiguity* (Levy et al, 2010)
- *Cambridge Gambling Task* (Rogers et al, 1999)
- *Choice reaction time → sequential sampling models*
- *Delay Discounting* (e.g., Mazur, 1987)
- *Description-based tasks with a probability weighting function* (e.g., Erev et al., 2010; Hertwig et al., 2004; Jessup et al., 2008)
- *Iowa Gambling* (Bechara et al, 1994)
- *"Happiness task"* (Rutledge et al, 2014)
- *(Orthogonalized) Go/Nogo* (Guitart-Masip et al, 2012)
- *Peer influence task* (Chung et al, 2015)
- *Probabilistic Selection task* (e.g., Frank et al, 2007)
- *Two-armed Bandit (Experience-based) including Reversal Learning* (e.g., Erev et al, 2010)
- *Four-armed Bandit (Experience-based)* (e.g., Seymour et al, 2012)
- *N-armed Bandit (Experience-based)*
- *Two-choice Description-based* (e.g., Sokol-Hessner et al, 2009; Tom et al, 2007)
- *Ultimatum Game* (e.g., Xiang et al, 2013)
- *Two-AFC w/ Signal Detection Theory (SDT)*
- *Two Step task* (Daw et al, 2011)
- *Wisconsin Card Sorting task* (Bishara et al, 2010)

hBayesDM in Python!

- *Easy to use with Python-based neuroimaging packages (e.g., Nipype)*
- *Model-based fMRI + MVPA (in preparation)*

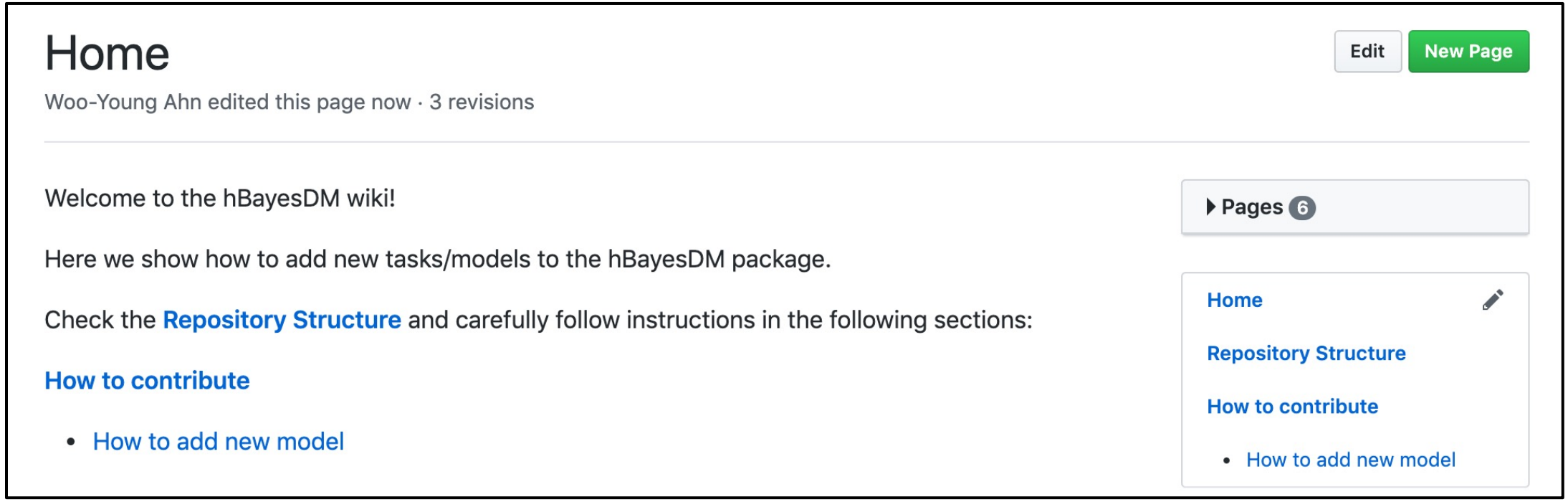




Cohen et al (2017) Nature neuro.

Much easier to contribute to hBayesDM now!

https://github.com/CCS-Lab/hBayesDM/wiki



How can I use it?

Tutorials available at

https://ccs-lab.github.io/hBayesDM/articles/getting_started.html (R)

```
install.packages("hBayesDM", dependencies=TRUE)
```

https://hbayesdm.readthedocs.io/ (Python)

```
pip install hbayesdm  # Install using pip
```

Brief step-by-step tutorials

** Part II for detailed tutorials*

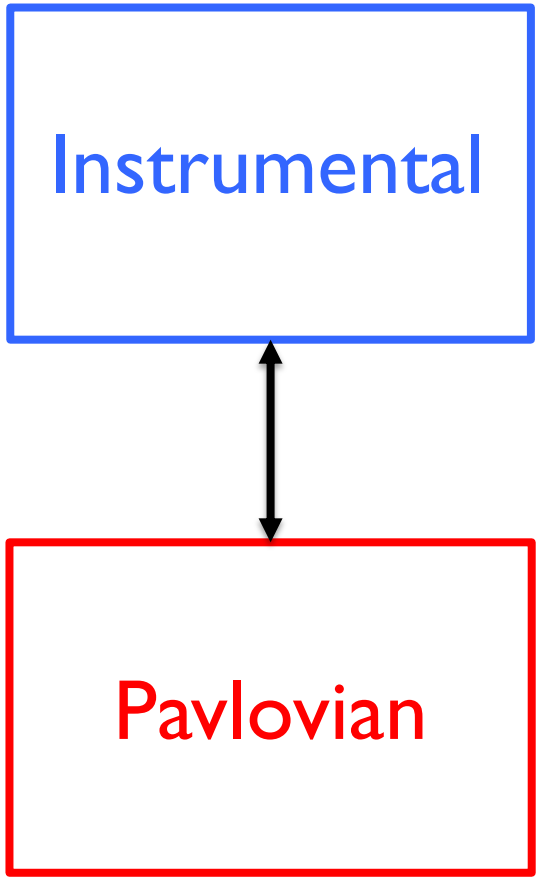
1. *Prepare raw (trial-by-trial) data*
2. *Fit candidate models*
3. *Plot (visualize) and inspect model parameters*
4. *Compare models (if there exist competing models)*

The Orthogonalized Go/Nogo task

- `gng_m1`
- `gng_m2`
- `gng_m3`
- `gng_m4`

Guitart-Masip et al, 2012 Neuroimage
Cavanagh et al, 2013 J Neuro

Orthogonalized Go/Nogo task
Pavlovian-Instrumental competition



"Learn what's best"

Approach for (+)
Withdraw from (-)

"Pavlovian bias"

Balleine & O'Doherty (2010); Dayan et al (2006); Dayan (2013); Dayan & Niv (2008); Dolan & Dayan (2013); Dayan & Berridge (2014); Rangel et al (2008)

Hungry Chicken

Food!

Hershberger (1986)

Hungry Chicken

Food!

Hershberger (1986)

Pavlovian-Instrumental competition

Hershberger (1986)

Pavlovian-Instrumental competition

Nogo

Instrumental

response

Young on diet

Go

Pavlovian

response

Instrumental goal: To lose weight

Orthogonalized Go/Nogo task

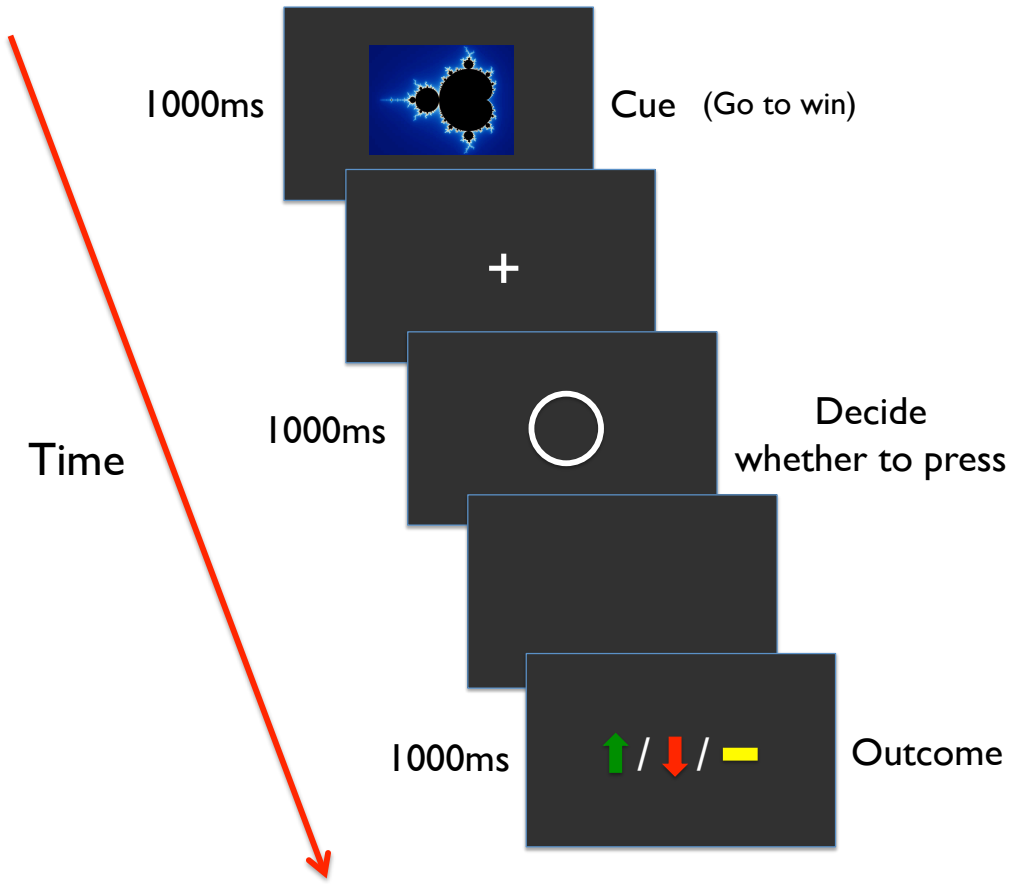
	Loss	Gain
Go	Go to avoid	Go to win
Nogo	Nogo to avoid	Nogo to win

Pavlovian congruent

Pavlovian incongruent

- *4 cues (conditions)*

  2 actions (Go / Nogo) x
  2 valence (Gain / Loss)



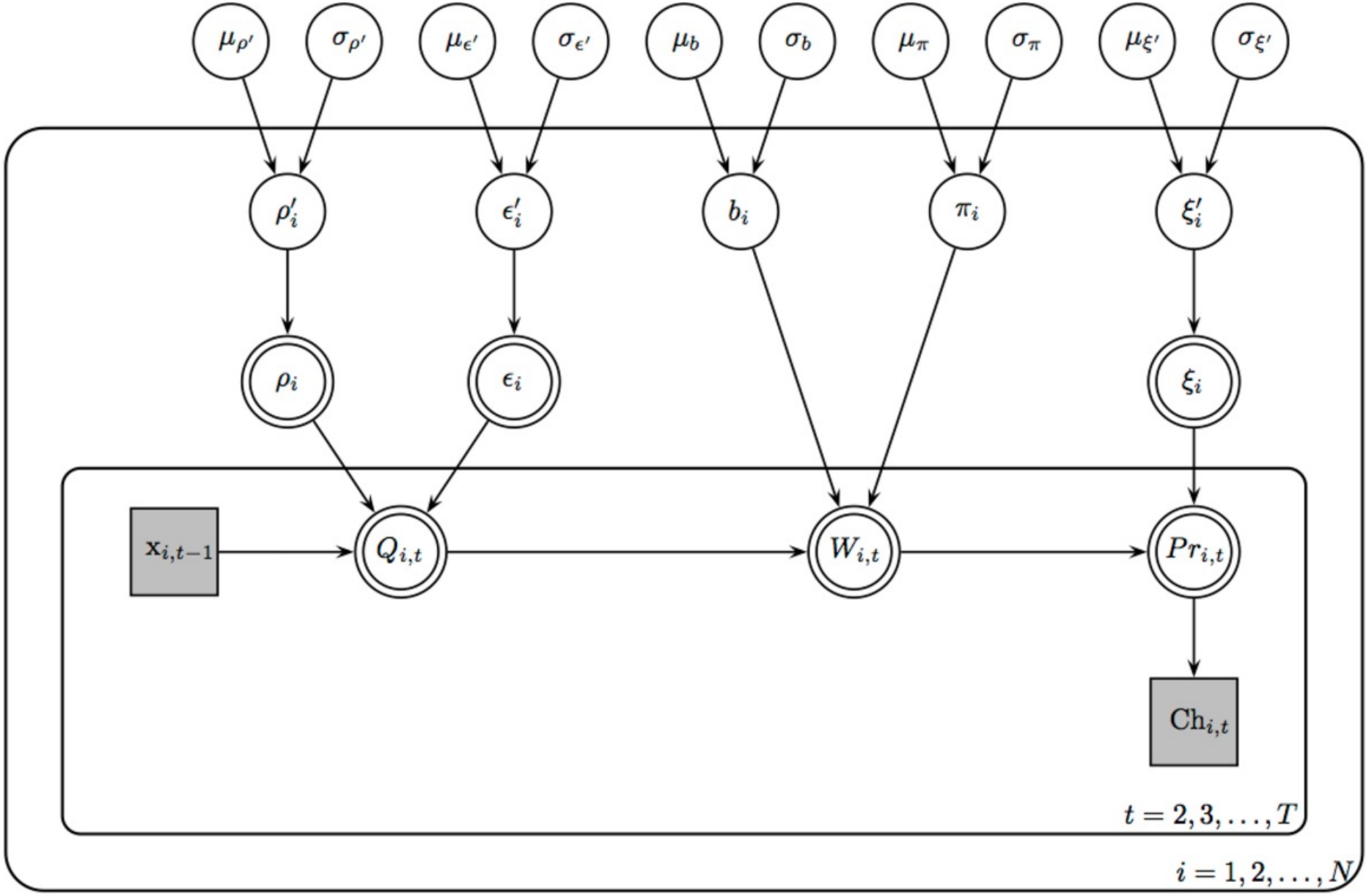
Guitart-Masip et al (2012) Neuroimage
Cavanagh et al (2013) J Neuro

1. *Prepare raw (trial-by-trial) data as a single text file*

	A	B	C	D	E	F	G
1	trialNum	cue	keyPressed	success	congruentOutcome	outcome	subjID
2	1	1	1	1	2	0	1
3	2	2	0	1	1	1	1
4	3	4	0	1	1	0	1
5	4	4	1	0	1	-1	1
6	5	4	0	1	1	0	1
7	6	1	1	1	1	1	1
8	7	3	0	0	1	-1	1
9	8	1	1	1	1	1	1
10	9	3	1	1	1	0	1
11	10	3	0	0	1	-1	1
12	11	4	0	1	1	0	1
13	12	4	0	1	1	0	1
14	13	4	0	1	1	0	1

2. Fit candidate models

```
output1 =                                                    4)
```

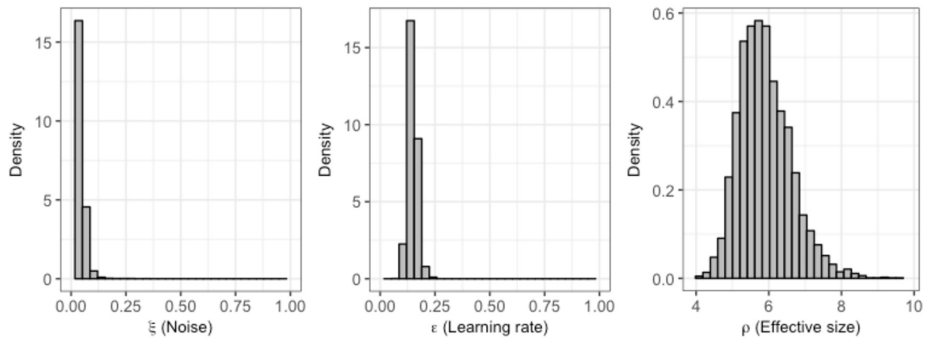


```
Details:
 # of chains                     =  4
 # of cores used                 =  4
 # of MCMC samples (per chain)   =  2000
 # of burn-in samples            =  1000
 # of subjects                   =  10
 # of (max) trials per subject   =  240

************************************
**  Loading a precompiled model  **
************************************
starting worker pid=75130 on localhost:11950 at 08:25:48.905
starting worker pid=75138 on localhost:11950 at 08:25:49.101

SAMPLING FOR MODEL 'gng_m1' NOW (CHAIN 1).

Chain 1, Iteration:    1 / 2000 [  0%]  (Warmup)
SAMPLING FOR MODEL 'gng_m1' NOW (CHAIN 2).
...
```

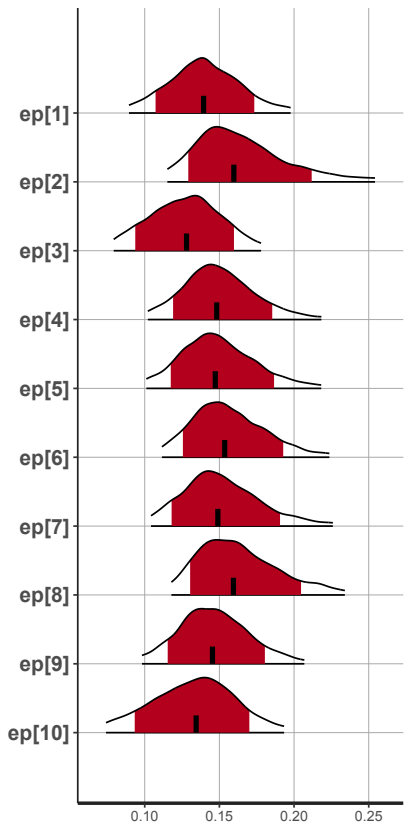
```
************************************
**** Model fitting is complete! ****
************************************
```

3. Plot (visualize) and inspect model parameters

```
plot(output1)
```




```
plotInd(output1, "ep")
```




```
> output1$allIndPars
           xi        ep      rho subjID
1  0.03688558 0.1397615 5.902901      1
2  0.02934812 0.1653435 6.066120      2
3  0.04467025 0.1268796 5.898099      3
4  0.02103926 0.1499842 6.185020      4
5  0.02620808 0.1498962 6.081908      5
...
```

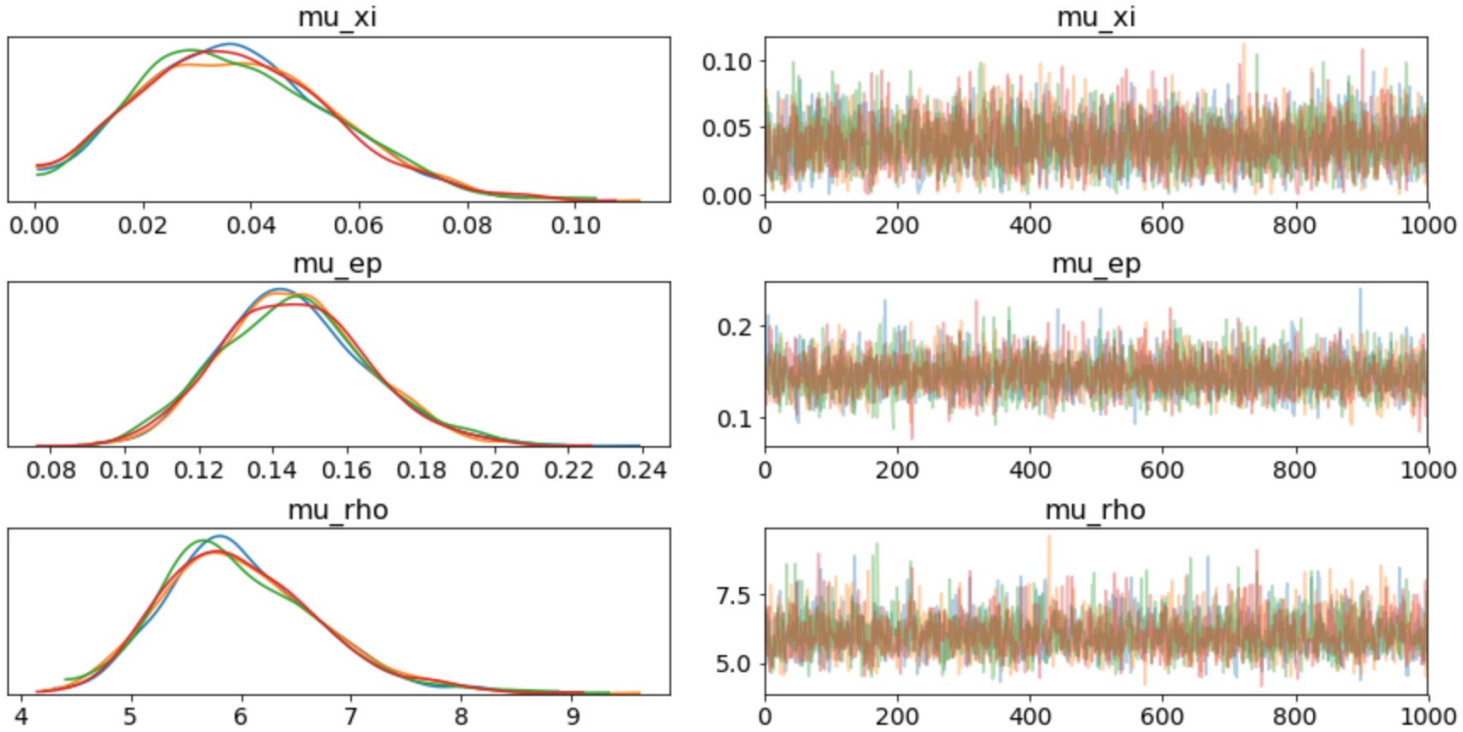
Python version

```
# Load hBayesDM

from hbayesdm import rhat, print_fit
from hbayesdm.models import gng_m1
```

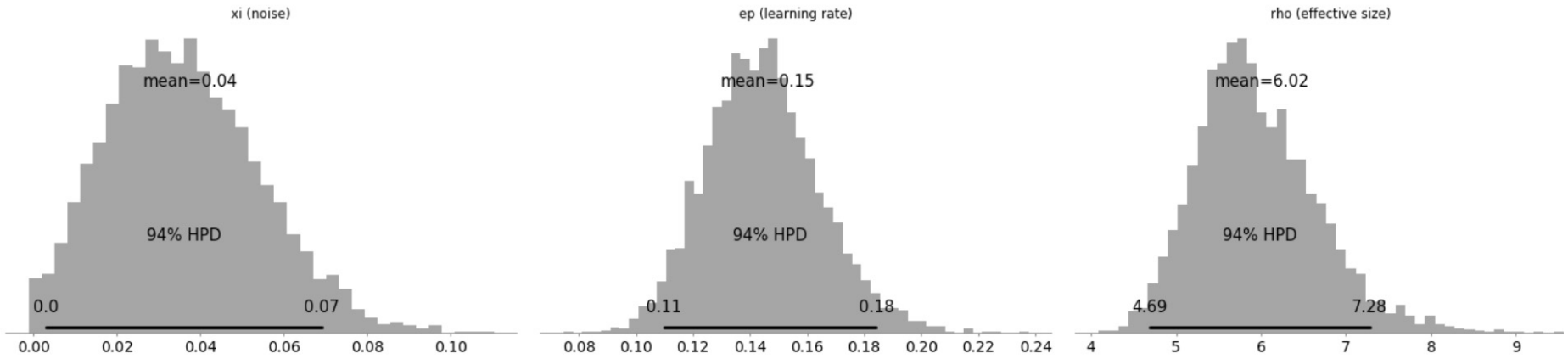
```
# Run the model and store results in "output"
output = gng_m1(data='example', niter=2000, nwarmup=1000, nchain=4, ncore=4)
```

```
INFO:pystan:COMPILING THE C++ CODE FOR MODEL gng_m1_5ae9a279bb0dc4d00c887e56e86cead2 NOW.
Model  = gng_m1
Data   = example

Details:
 # of chains                     = 4
 # of cores used                 = 4
 # of MCMC samples (per chain)   = 2000
 # of burn-in samples            = 1000
 # of subjects                   = 10
 # of (max) trials per subject   = 240
```

```
# Visually check convergence of the sampling chains (should look like "hairy caterpillars")
output.plot(type='trace')
```




```
# Plot posterior distributions of the hyper-parameters (distributions should be unimodal)
output.plot()
```



4. Bayesian model comparisons

Vehtari et al. (2016)

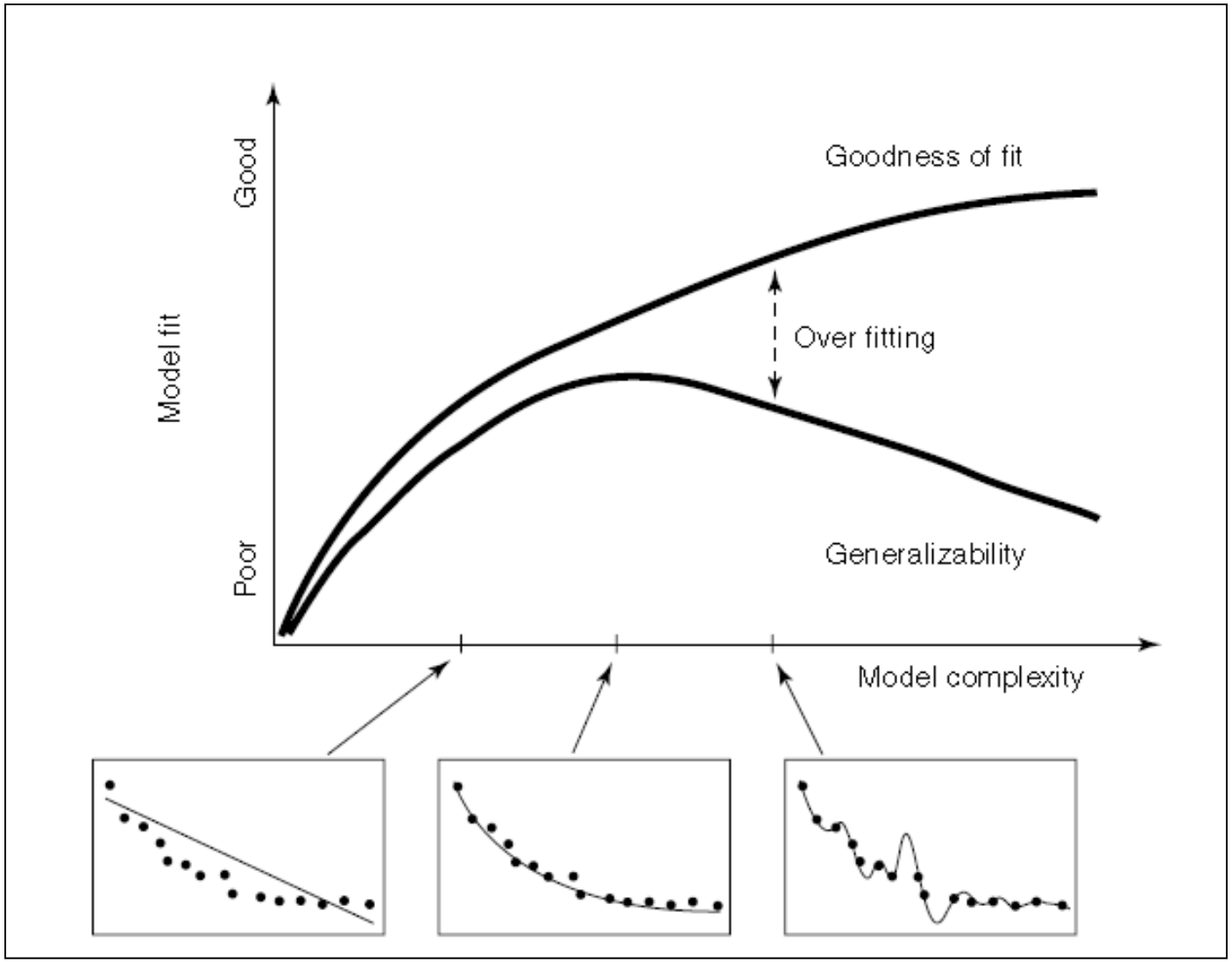
Leave-One-Out Information Criterion (LOOIC) - default

Widely Applicable Information Criterion (WAIC)

```
> printFit(output1, output2, output3, output4)
```

Model #4 is the best model
(in terms of LOOIC)

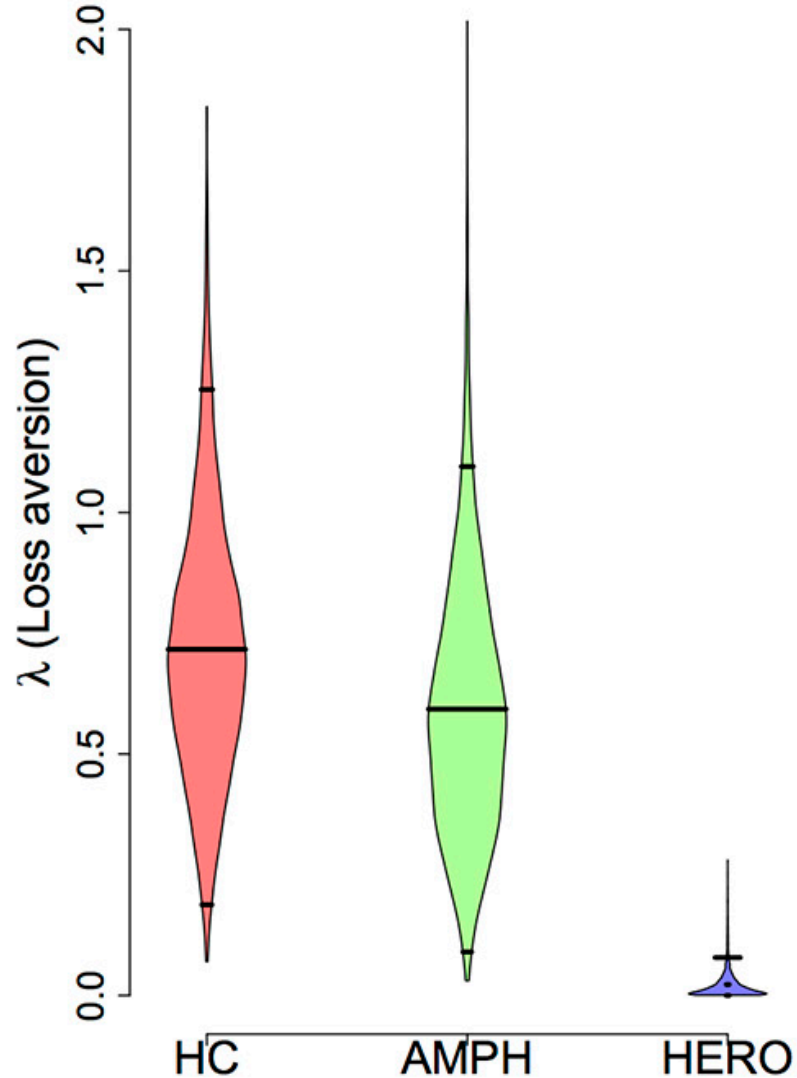
Goodness of Fit vs Model Complexity

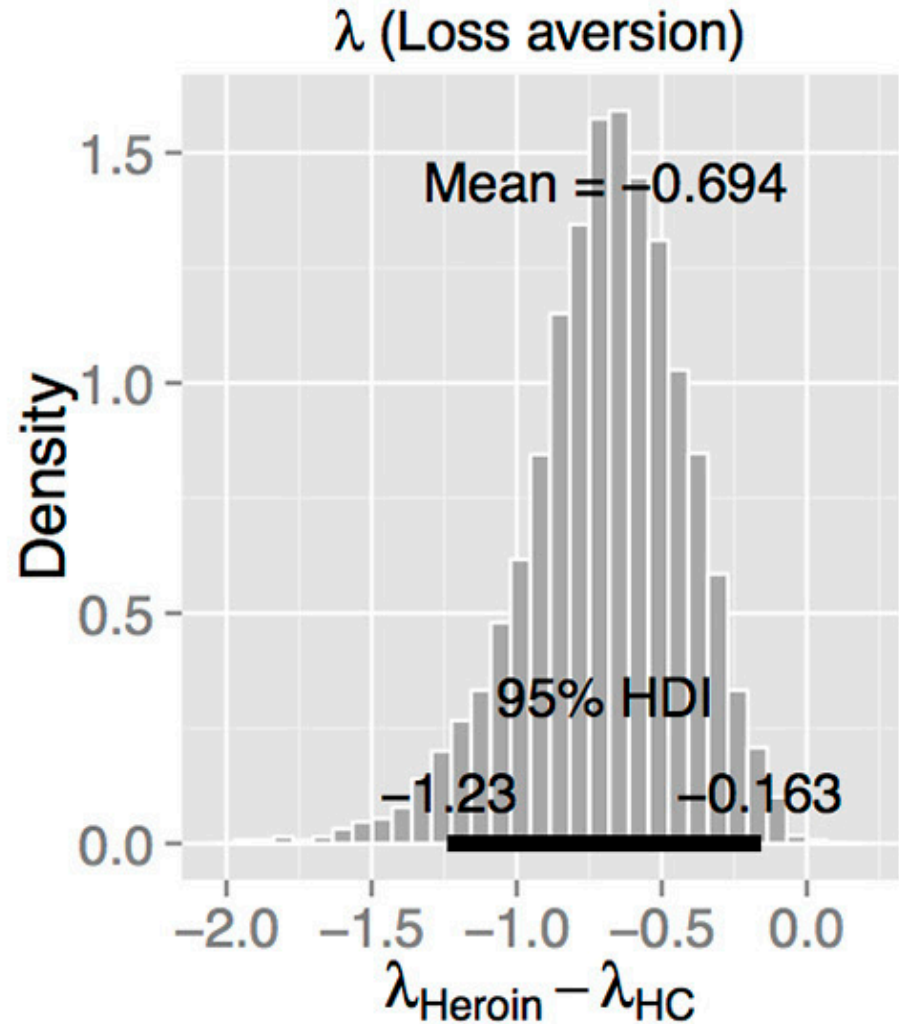


Pitt & Myung (2002) TICS

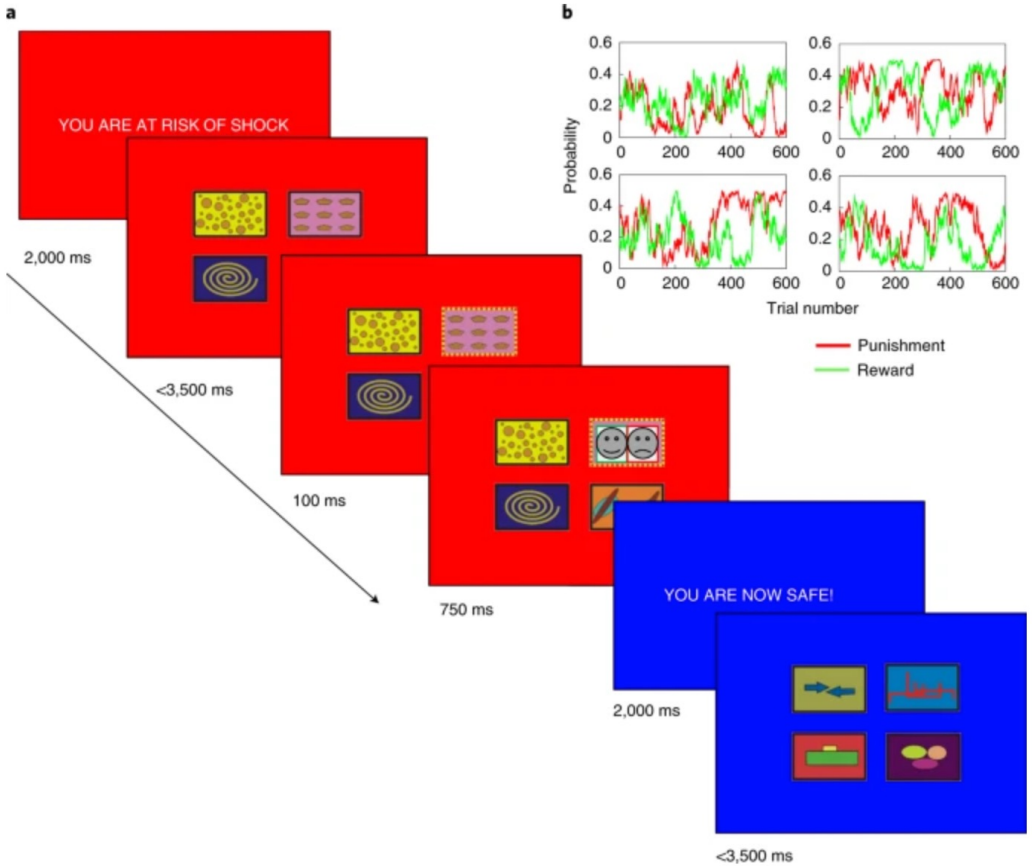
5. Group comparisons (between-subject comparisons)

```
data_group1 = "~/Project_folder/gng_data_group1.txt"  # data file for group1
data_group2 = "~/Project_folder/gng_data_group2.txt"  # data file for group2

output_group1 = gng_m4(data_group1)  # fit group1 data with the gng_m4 model
output_group2 = gng_m4(data_group2)  # fit group2 data with the gng_m4 model

# After model fitting is complete for both groups,
# evaluate the group difference (e.g., on the 'pi' parameter) by examining the posterior distribution of
 group mean differences.

diffDist = output_group1$parVals$mu_pi - output_group2$parVals$mu_pi  # group1 - group2
HDIofMCMC( diffDist )  # Compute the 95% Highest Density Interval (HDI).
plotHDI( diffDist )    # plot the group mean differences
```

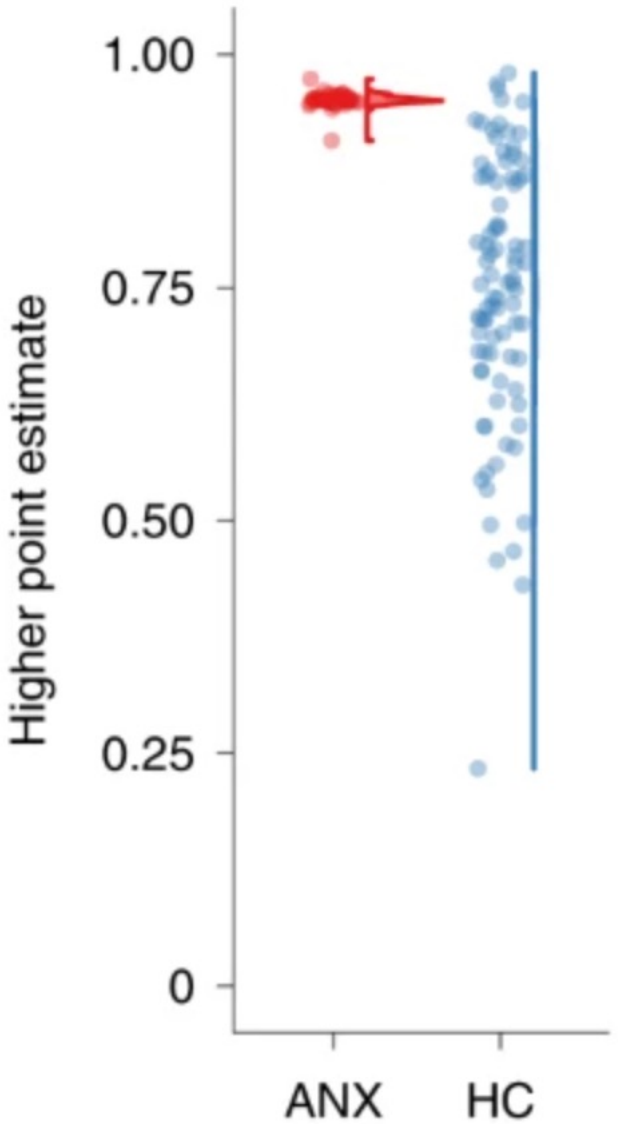




Ahn et al (2014) Frontiers in Psychology

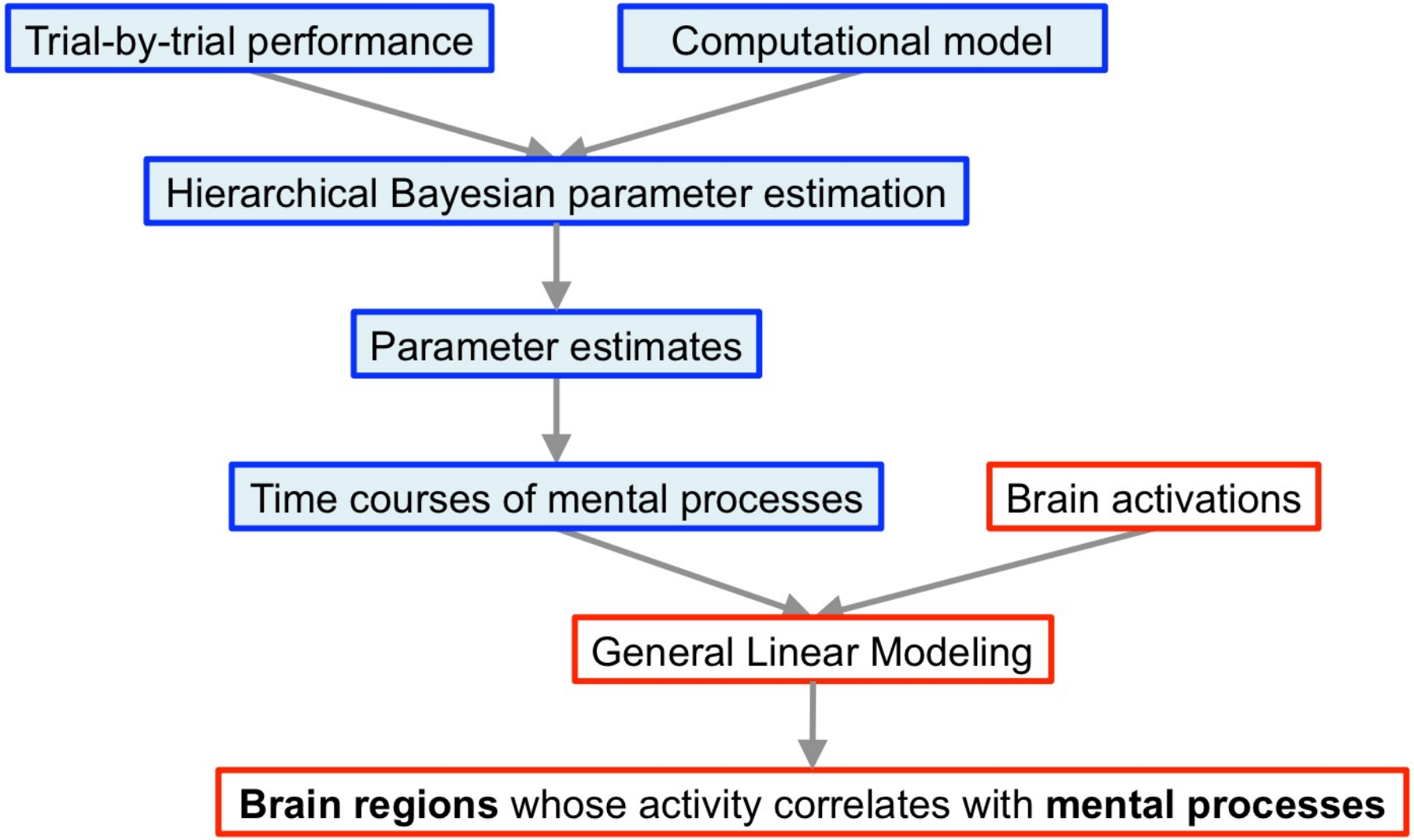


Punishment learning rate



Alyward, Valton, Ahn, et al (2019) Nature Human Behaviour

6. Model-based fMRI/EEG



Model-based fMRI/EEG

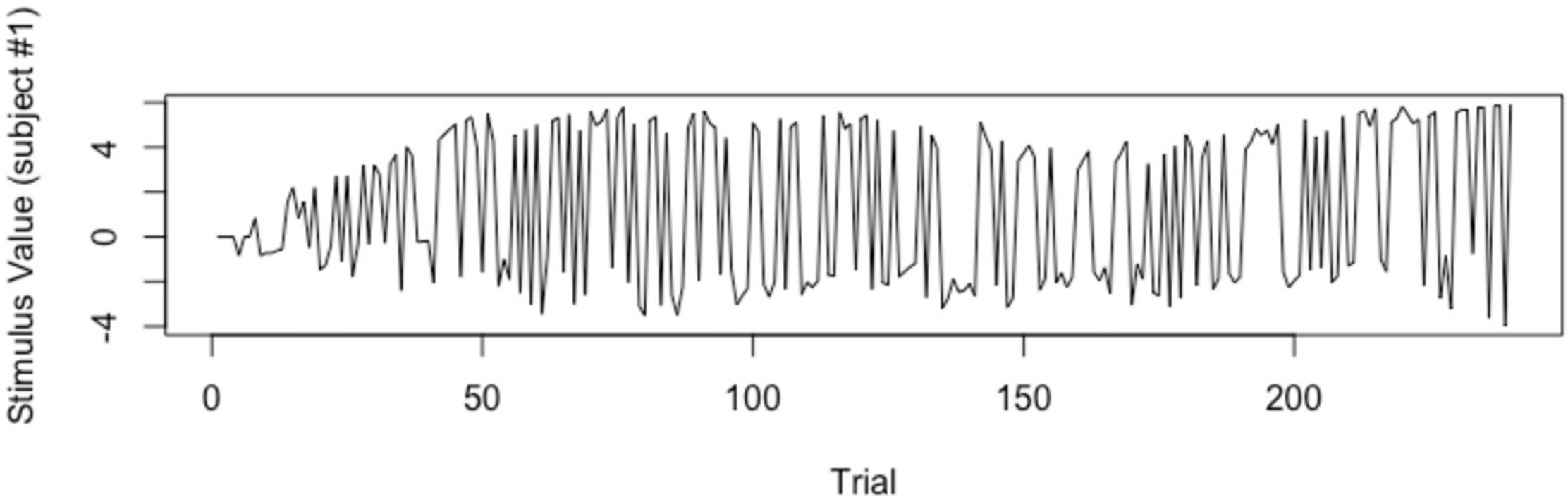
- *Trial-by-trial regressors are readily available*

```
# fit example data with the gng_m3 model
output = gng_m3(data="example", niter=3000, nwarmup=1000, nchain=3, ncore=3, modelRegressor=TRUE)
```

```r
# store all subjects' stimulus value (SV) in 'sv_all'
sv_all = output$modelRegressor$SV

dim(output$modelRegressor$SV)  # number of rows=# of subjects (=10), number of columns=# of trials (=240)
```

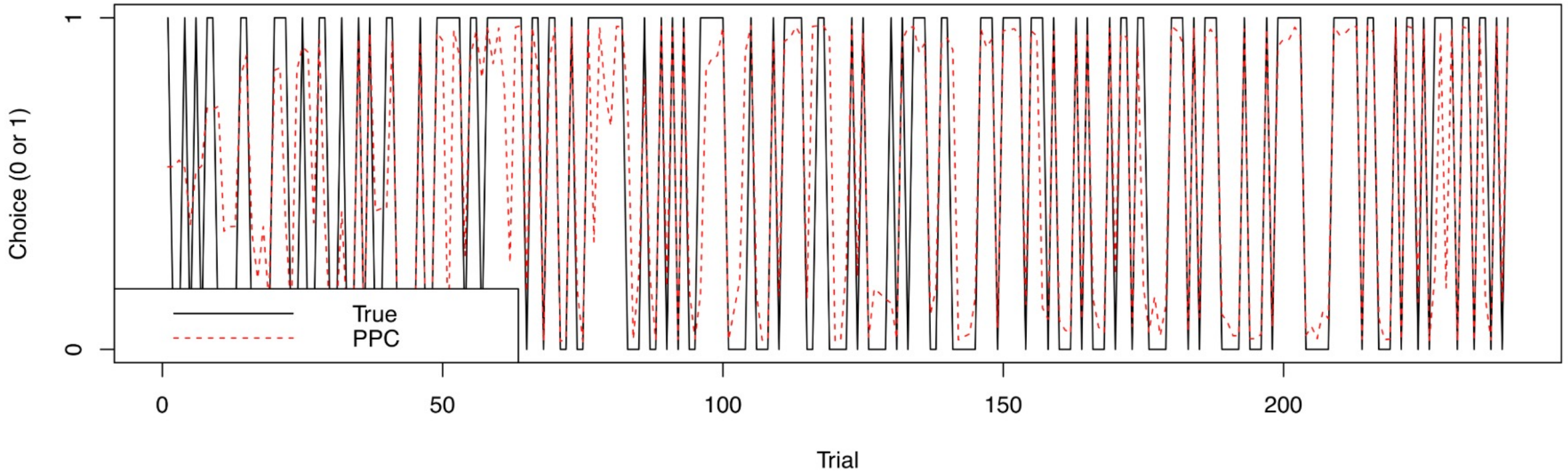
```
## [1]  10 240
```

```r
# visualize SV (Subject #1)
plot(sv_all[1, ], type="l", xlab="Trial", ylab="Stimulus Value (subject #1)")
```



Posterior predictive checks

```
# fit example data with the gng_m3 model and run posterior predictive checks
x = gng_m3(data="example", niter=2000, nwarmup=1000, nchain=4, ncore=4, inc_postpred = TRUE)
```



Outline

- *Understand the concepts of computational modeling*
- *Know popular and cutting-edge methods for fitting computational models / Understand the concept of Bayesian data analysis*
- *Things to know when performing MCMC sampling*
- *Use hBayesDM package to model several tasks*

Popular methods for model fitting

- *Maximum likelihood estimation (MLE)*
- *Bayesian parameter estimation (or Bayesian data analysis)*
- *Hierarchical Bayesian analysis*

Maximum likelihood estimation

- *Ronald A. Fisher (1920s): finds the value of a parameter that makes the observed data "most likely".*
- *Likelihood function*
  - L(data | θ): *likelihood of getting data given* θ
- *Analytical solutions often don't exist*
- *Nonlinear optimization*
  - *gradient decent*
  - `fminsearch` *(Matlab),* `optim` *(R), etc.*
- *MLE estimates = point estimates*

Quantify uncertainty in parameter estimates

- *In MLE*
  - *Bootstrapping: samples with replacement from the original data*
  - *Distribution of point estimates → classical measure of uncertainty*
  - *Hessian matrix*
- *In Bayesian*
  - *Plotting posterior distributions*

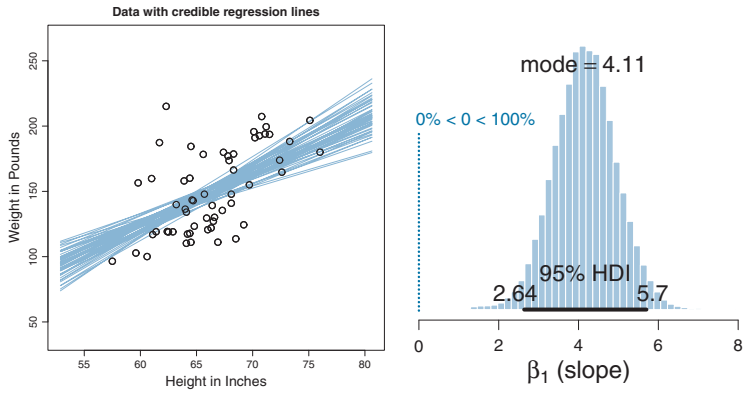
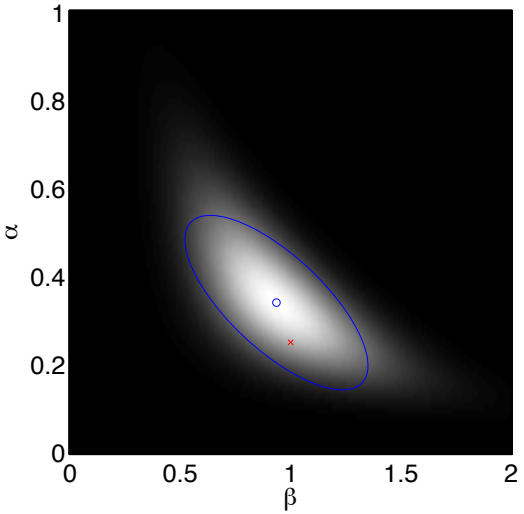


Figure 2.5 Data are plotted as circles in the scatter plot of the left panel. The left panel also shows a smattering of credible regression lines from the posterior distribution superimposed on the data. The right panel shows the posterior distribution of the slope parameter (i.e., β_1 in Equation 2.1).

Kruschke (2015)



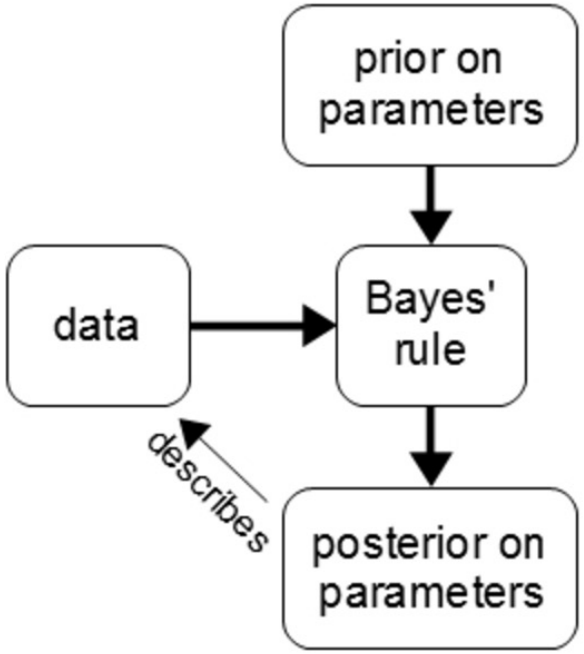
"Likelihood surface" from Daw (2011)

Brief Introduction to Bayesian data analysis

Bayesian data analysis

Bayesian data analysis vs. *Bayesian model of mind*

http://doingbayesiandataanalysis.blogspot.hk/2011/10/bayesian-models-of-mind-psychometric.html

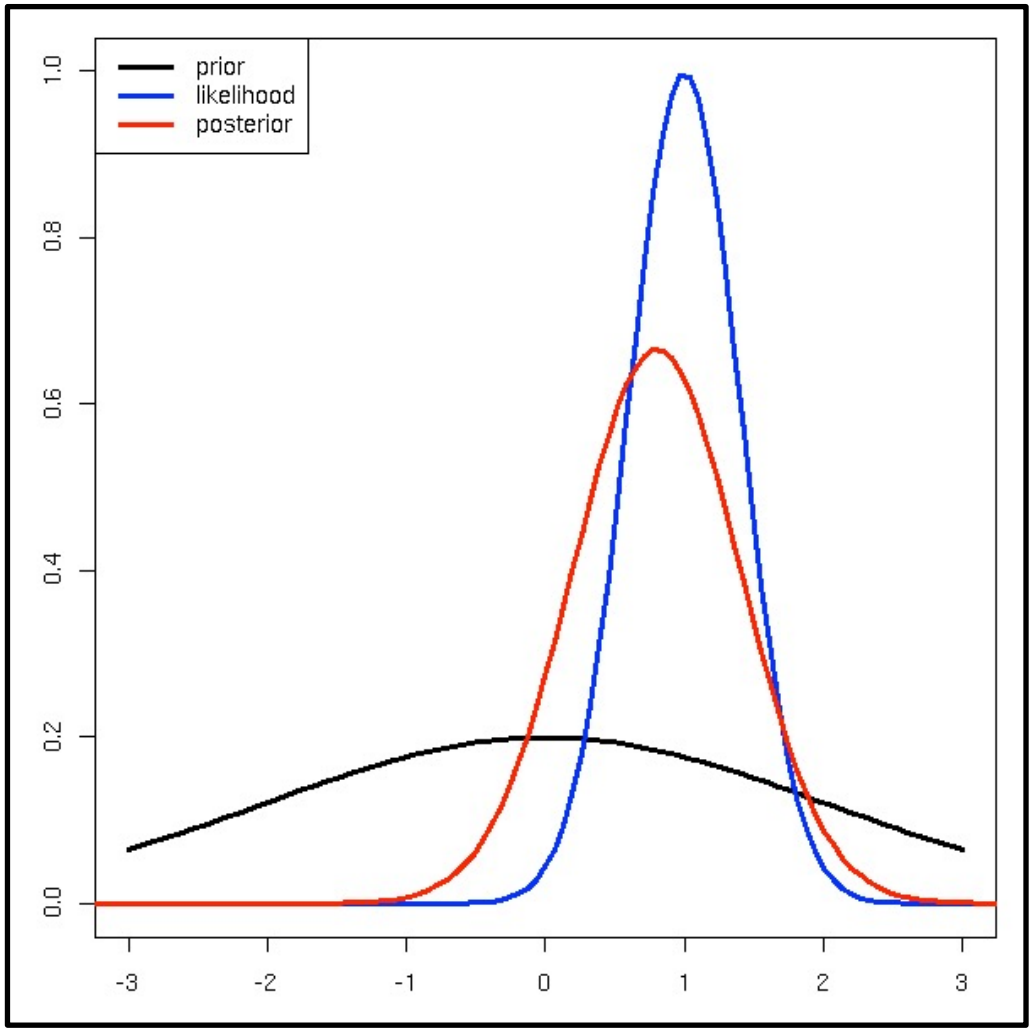


Θ: Model parameters (e.g., discounting rate, learning rate, etc.)

Posterior *Likelihood* *Prior*

$$P(\Theta_i \mid D_i) = \frac{P(D_i \mid \Theta_i)P(\Theta_i)}{P(D_i)} = \frac{P(D_i \mid \Theta_i)P(\Theta_i)}{\int P(D_i \mid \Theta_{i'})P(\Theta_{i'})d\Theta_i'}$$

Evidence



http://zoonek.free.fr/blosxom/R/2006-06-22_useR2006.html

Bayesian data analysis

Update prior distributions for model parameters into posterior distributions given the data (e.g., trial-by-trial choices and outcomes) using Bayes' rule

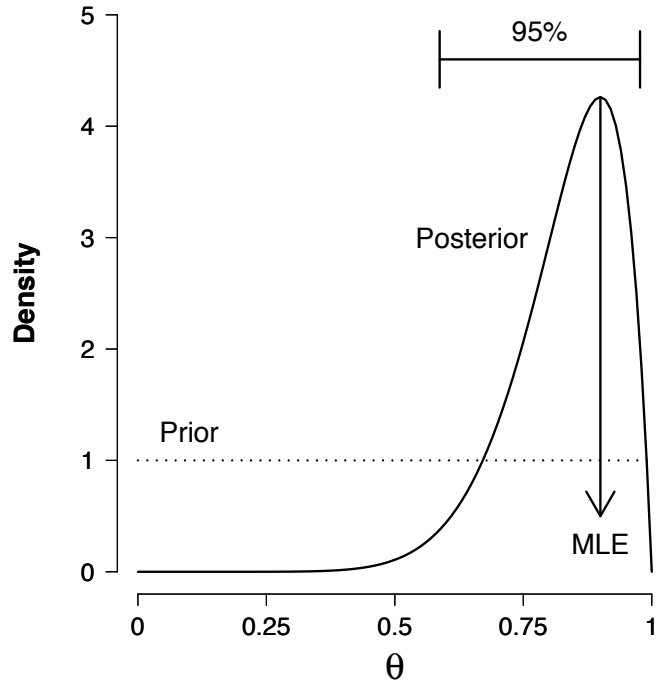
Relationship between MLE and Bayesian data analysis

An example from Lee & Wagenmakers (2012)

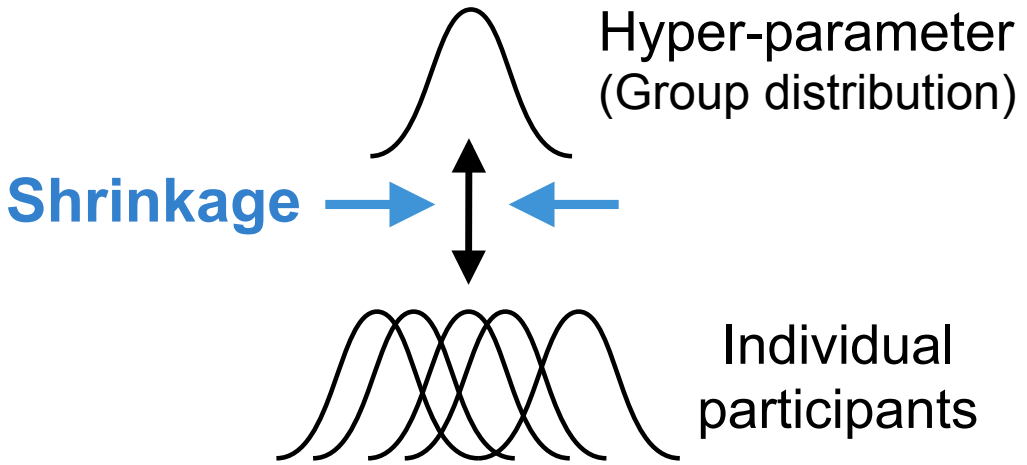
- *A test with 10 questions*
- *Parameter of interest: your ability* ($0 < \theta < 1$)
- *Data: 9 correct answers out of 10 (k=9, n=10)*
- *Likelihood function: Bernoulli*

Bayesian $\boxed{p(\theta \mid D)} = \dfrac{p(D \mid \theta)\, p(\theta)}{p(D)}$

MLE $\boxed{\hat{\theta}} = k/n = 0.9$



Hierarchical Bayesian analysis



Θ: Individual model parameters
Φ: Group model parameters

$$\Theta_i = \{\alpha_i, \beta_i, \gamma_i, \cdots\}$$

$$\Phi = \{\mu_\alpha, \mu_\beta, \mu_\gamma, \sigma_\alpha, \sigma_\beta, \sigma_\gamma, \cdots\}$$

- *Similarities & differences across participants*
- *Small amount of data from each subject*

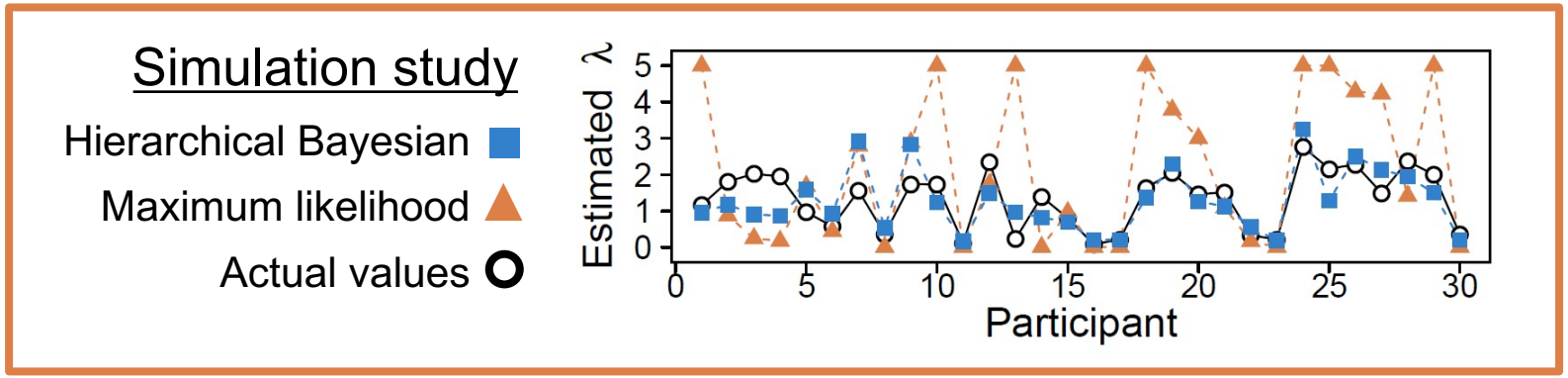
Likelihood Prior

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

Posterior Evidence

Hierarchical Bayesian data analysis

leads to more accurate parameter estimates



Ahn et al. (2011, JNPE)

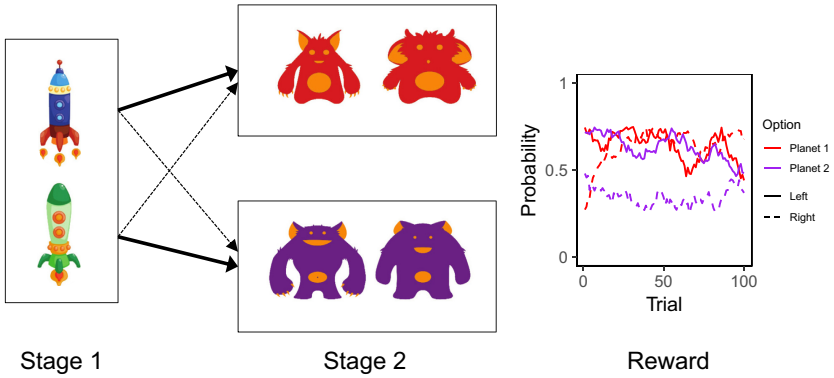
Limitations

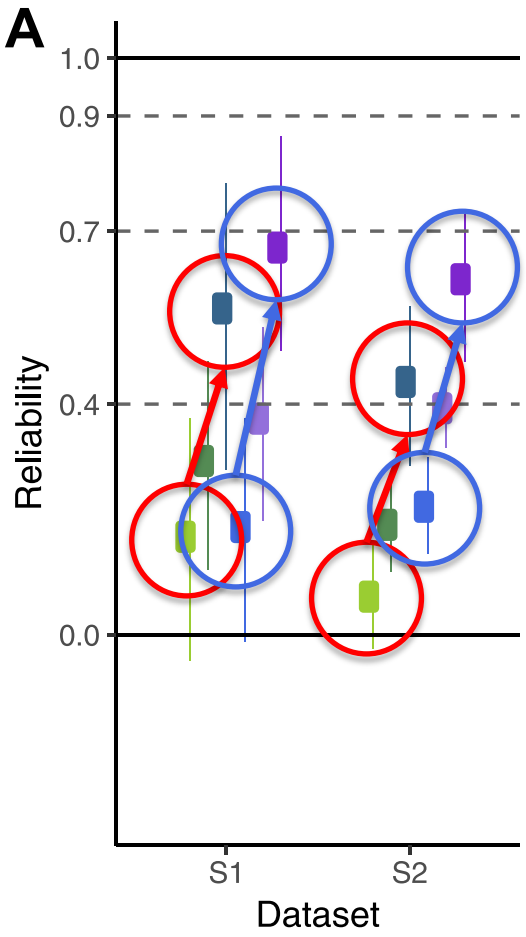
- *Assumption of a single hyper-group*
- *Shrinkage → regression toward the group mean*

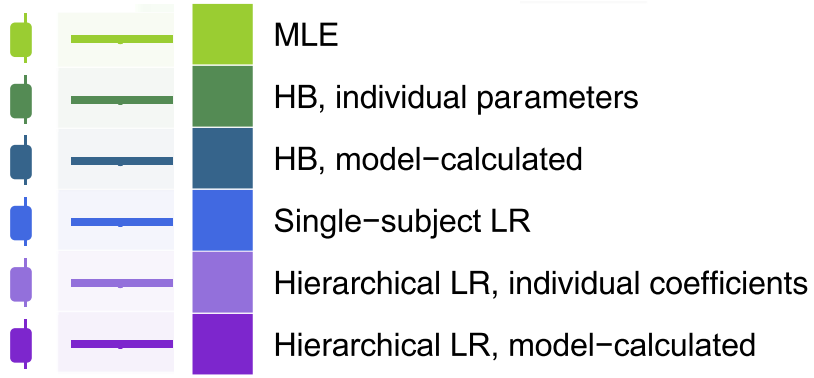
Ahn, W.-Y., Krawitz, A., Kim, W., Busemeyer, J. R., & Brown, J. W. (2011). A Model-Based fMRI Analysis with Hierarchical Bayesian Parameter Estimation. *Journal of Neuroscience, Psychology, and Economics*, *4*(2), 95-110.

model-based vs. model-free

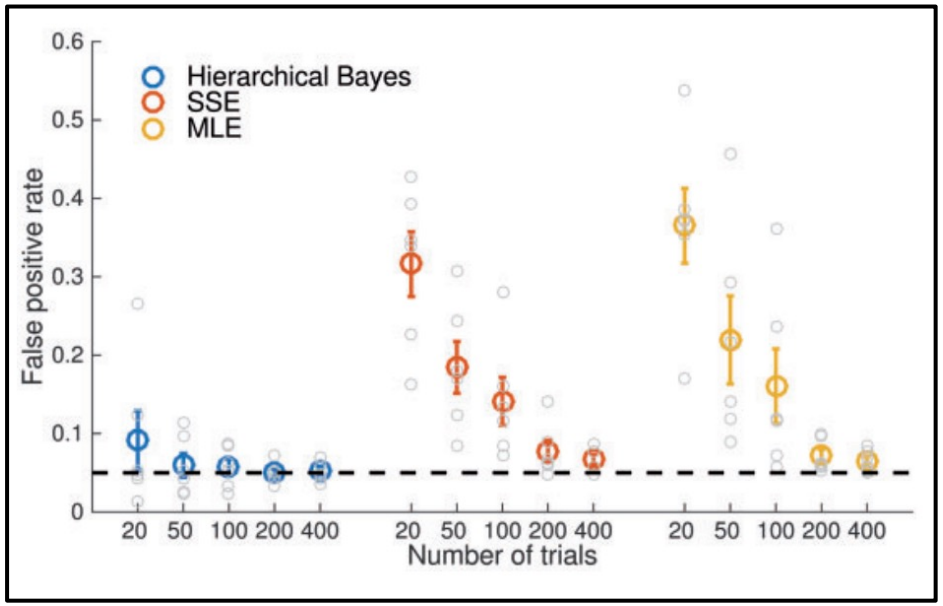
In the Two-Step Task







Brown et al (2018) Biological Psychiatry: CNN



Fleming (2017) Neuroscience of Consciousness

Okay sounds good… but how can we actually do Bayesian updating?

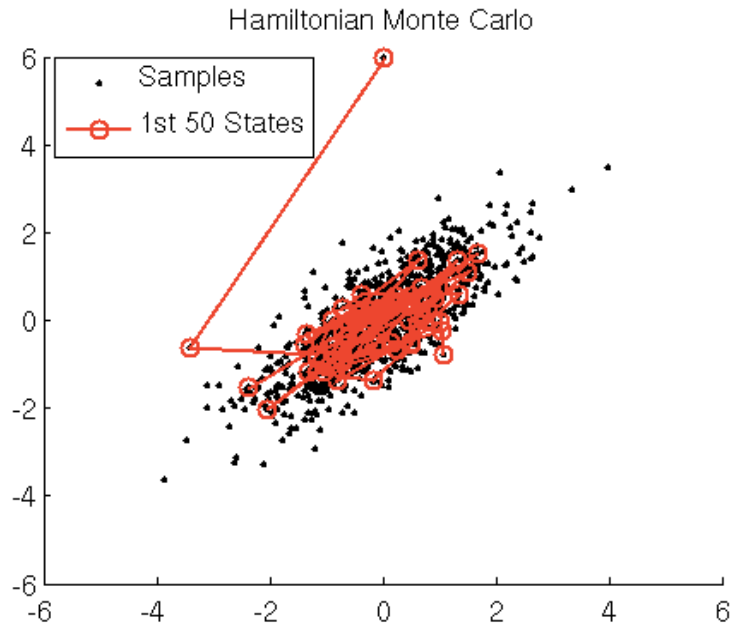
$$P(\Theta_i \mid D_i) = \frac{P(D_i \mid \Theta_i)P(\Theta_i)}{P(D_i)} = \frac{P(D_i \mid \Theta_i)P(\Theta_i)}{\int P(D_i \mid \Theta_{i'})P(\Theta_{i'})d\Theta_i'}$$

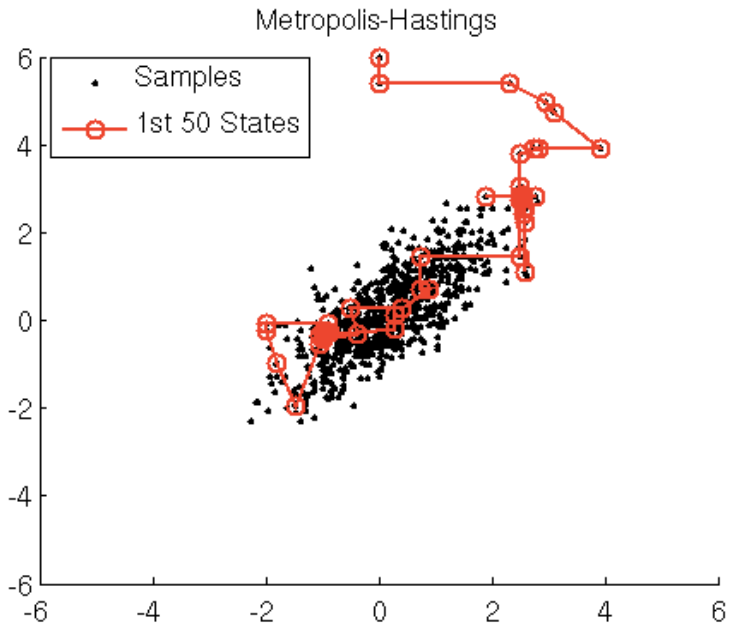
1. *Analytical solutions (often don't exist)*
2. *Grid approximation*
3. *Variational Bayes*
4. *Markov Chain Monte Carlo (MCMC)*

Markov Chain Monte Carlo (MCMC)

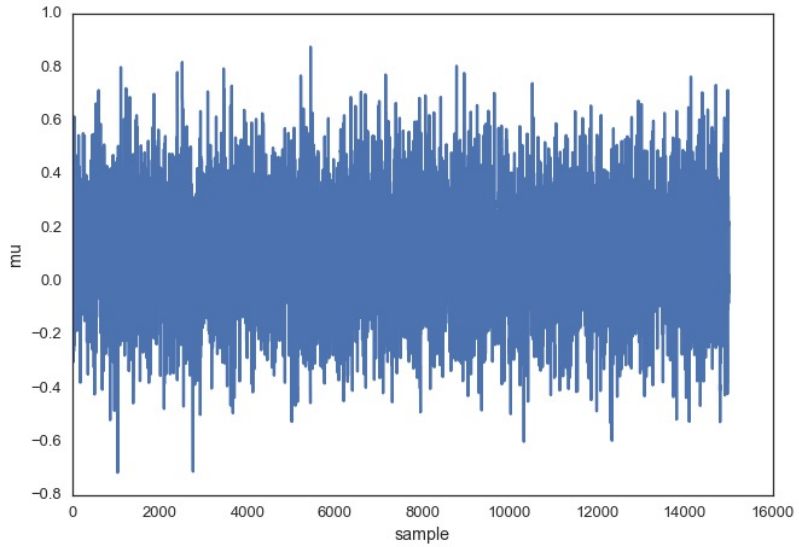
Approximate posterior distributions by drawing "random" samples

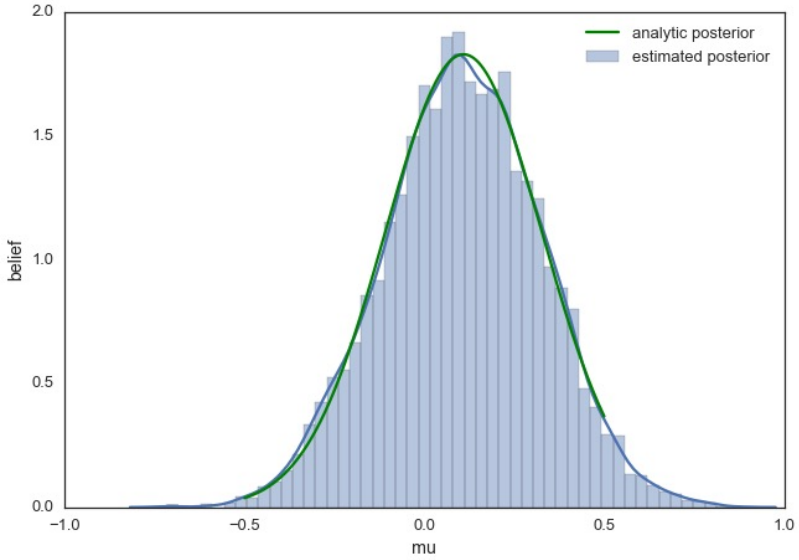
Some samplers are more efficient than others!





https://theclevermachine.wordpress.com/2012/11/18/





http://twiecki.github.io/blog/2015/11/10/mcmc-sampling/

Tools for MCMC sampling

1. *Program your own!?*
2. *User-friendly packages*
   - *WinBUGS/OpenBUGS*
   - *JAGS*
   - *PyMC3 (in Python)*
   - *Stan*

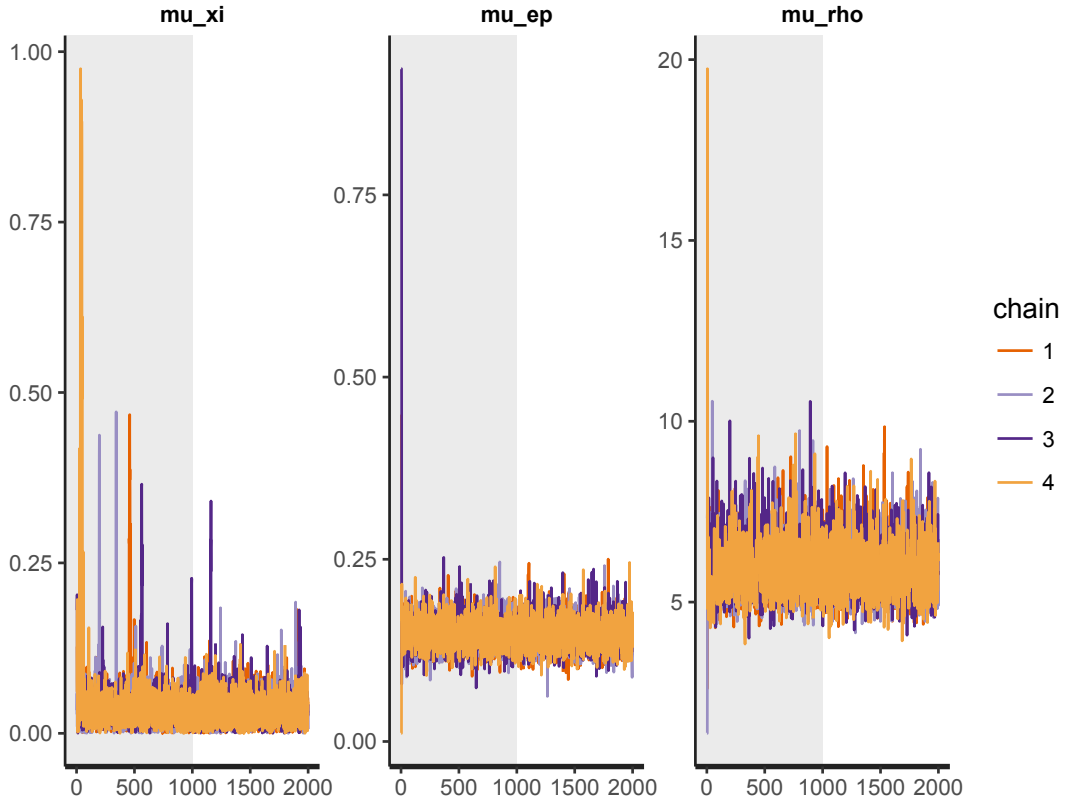
3. *hBayesDM utilizes Stan (http://mc-stan.org/), which is well-maintained by a team led by Andrew Gelman.*
4. *Stan & hBayesDM support VB as well!*

Outline

- *Understand the concepts of computational modeling*
- *Know popular and cutting-edge methods for fitting computational models / Understand the concept of Bayesian data analysis*
- *Things to know when performing MCMC sampling*
- *Use hBayesDM package to model several tasks*

Things to know when performing MCMC sampling

"Beware: hBayesDM can be dangerous!"

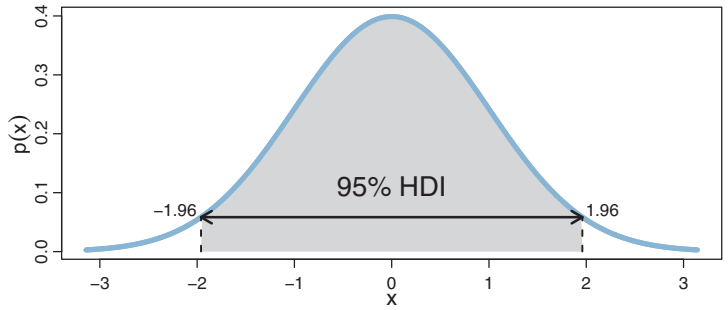


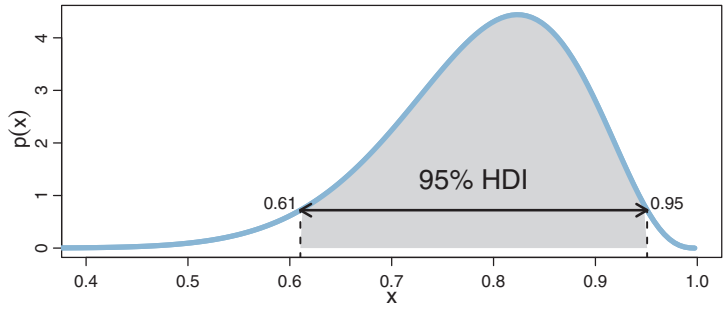

```
plot(output1, type="trace", inc_warmup=T)
```

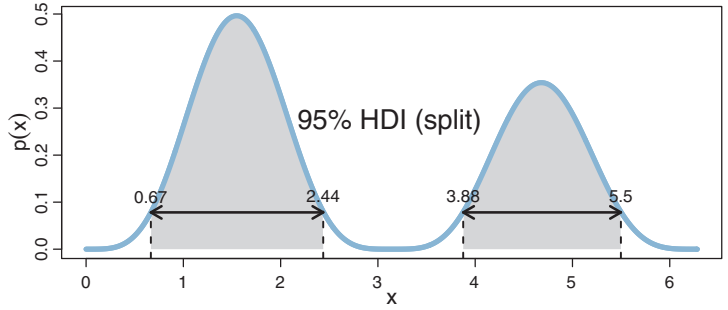
Check the convergence!

- *Number of chains → at least 4*
- *Burn-in (warm-up) samples*
- *Convergence index (e.g., $\hat{R}$)*
  → Close to 1.00
  At least less than ~1.10

Highest Density Interval (HDI)

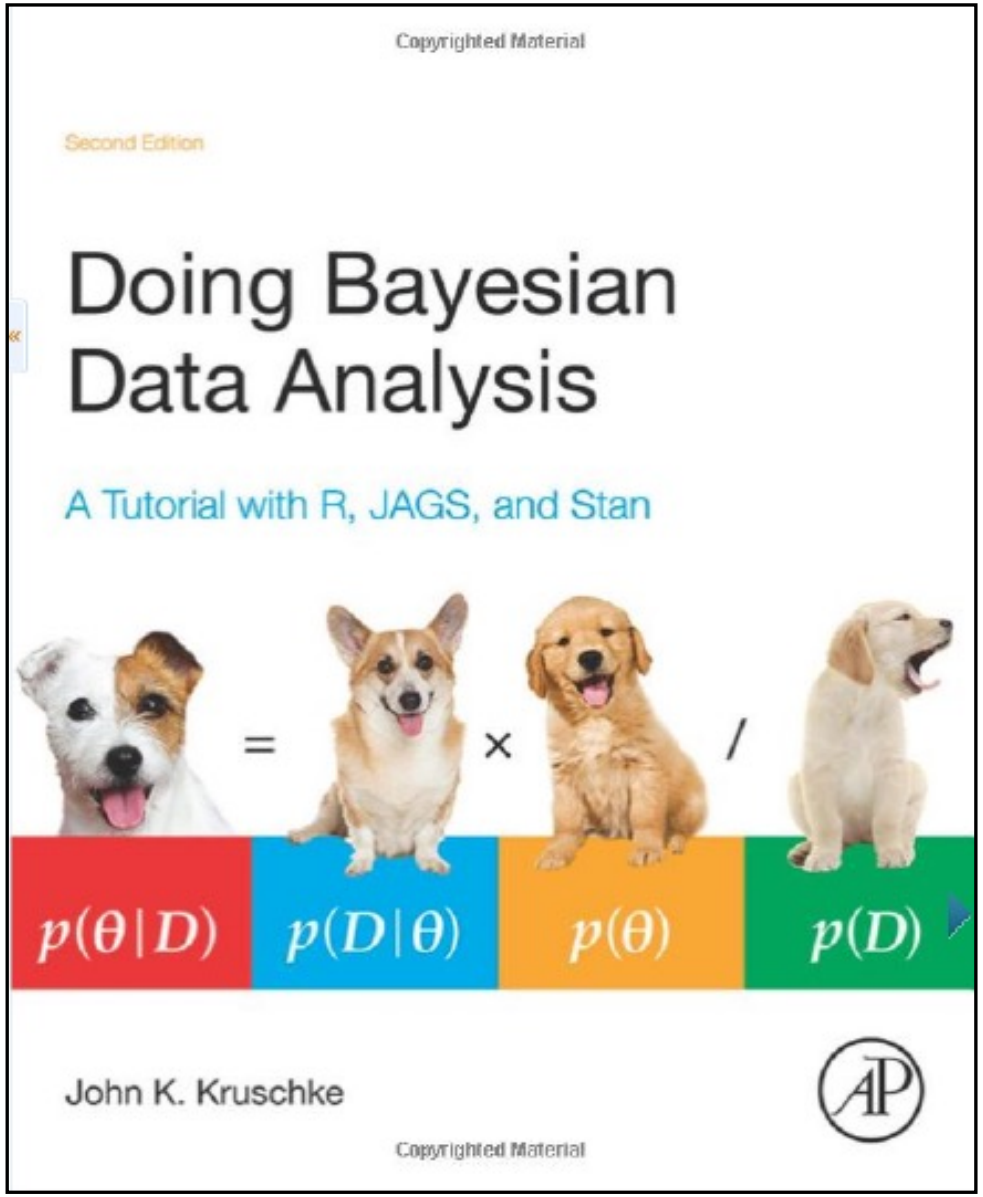


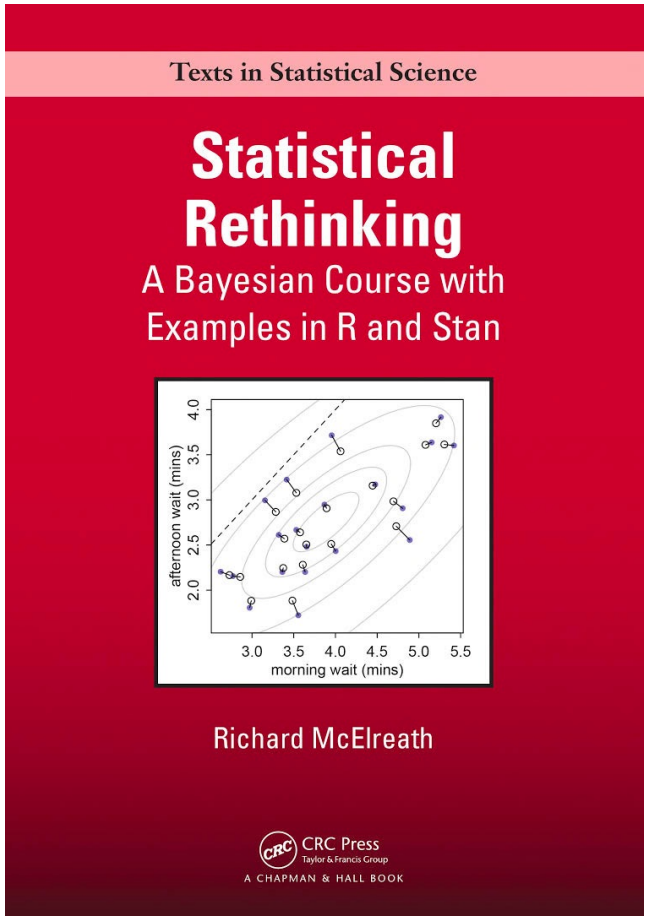




- *"Which points of a distribution are most credible"*
- *Skewed vs non-skewed*

Doing Bayesian Data Analysis, 2nd ed. (2014)





Outline

- *Understand the concepts of computational modeling*
- *Know popular and cutting-edge methods for fitting computational models*
- *Understand the concept of Bayesian data analysis*
- *Evaluate outputs from Bayesian data analysis*
- *Use hBayesDM package to model several tasks (Part II)*
  - *If you learn how to model one task, you know how to model all!*