北京大学心理与认知科学学院
School of Psychological and Cognitive Sciences, Peking University

Computational modeling in RStan

--basics and practices

Hanbo Xie

Self introduction

- Name: Hanbo Xie

- Current Affiliation: Research Assistant in Zhou's Lab in Peking University

- Education: BSc of Human Resource Management in SWUFE; Exchange student in UCR

- Research Interests: Interested in decision making, learning (reinforcement learning, social learning and group learning) via computational neuroscience approach.

Goal

- This tutorial is not to:
  - Introduce how to create a good model;
  - Introduce a specific family of models;

- But we expect to:
  - Develop an interest in combining computational models in your own research
  - Know a basic routine to conduct computational model work
  - Use MCMC and Rstan to fit your own models

Content

Basics about computational modeling

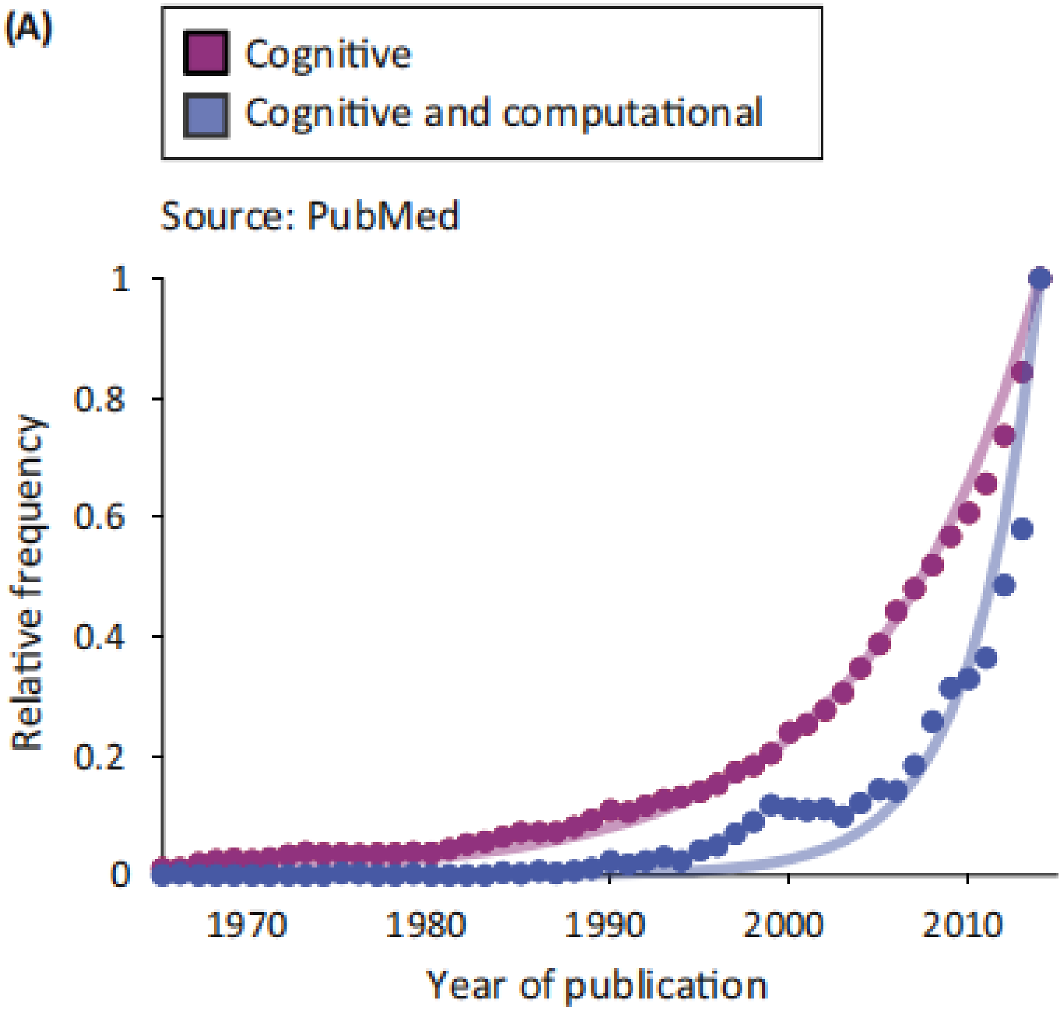
Introduction

- What is computational modeling?
  - computational modeling is a technique that applies **mathematical models** and **computer science** in complex systems (e.g., ecological environment, biology, physics and etc.,) to better understand the underlying mechanisms.

- What can computational modeling relate to psychology and neuroscience?
  - better understand behavioral mechanisms (choice behavior, RT, eye movement and etc.)
  - better understand neural mechanisms (DCM, neuronal spikes and etc.)

- Why is it special?
  - high precision on predicting data
  - inference on laten variables (states)
  - capability of coping large-scale, high-dimensional data
  - ...

Introduction



Computational cognitive research is growing rapidly, due to the development of machine learning techniques and AI.

New interdisciplinary fields emerged:

- neuroeconomics
- computational neuroscience
- computational psychiatry
- ...

Palminteri, et.al. 2017

Introduction

- Basic concepts of a computational model:
  - suppose you have a set of data concerning RT and choice accuracy....

data (observation, things you already know)

output (y) **input (x)** **parameters**

goal (things you wish to know...)

$$\boxed{\begin{array}{l} acc \sim a * log(RT) + b + \varepsilon \\ \\ \varepsilon \sim Normal\ (0,1) \end{array}}$$

model

(hyper parameter)

relationship (hypotheis, things you think you know)

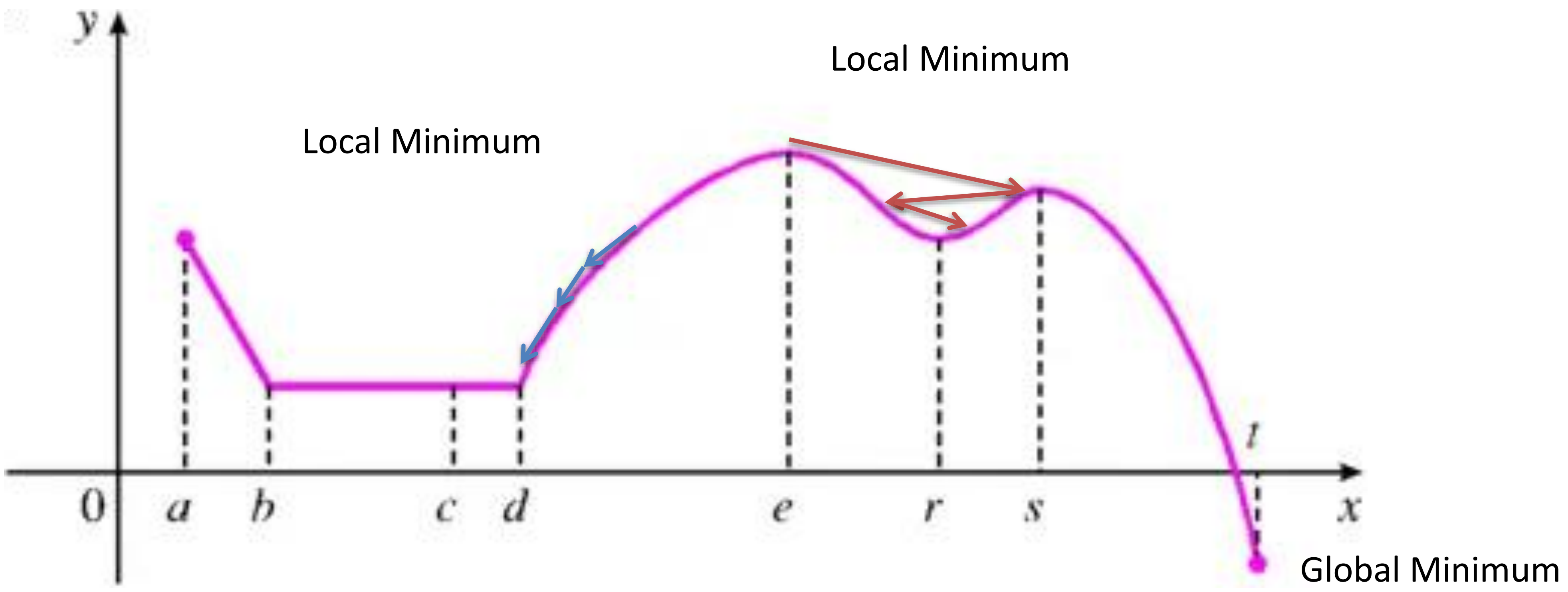
Introduction

- Model fitting:
  - get to the goal
  - optimize the parameters that best suit to your data, given your model
  - how to optimize: minimize the errors between data and model predictions, by adjusting the parameters.

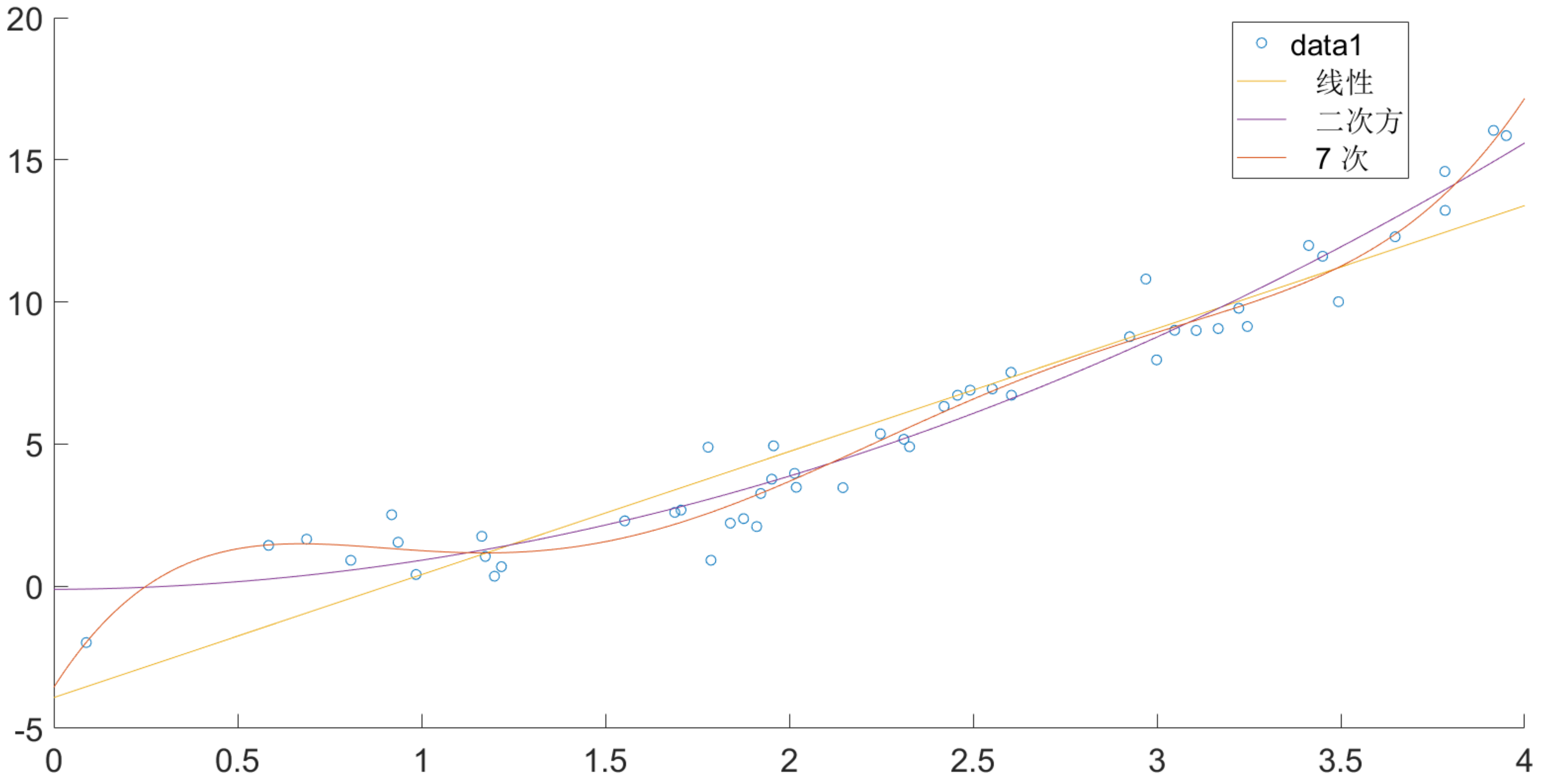
- Cost functions (measurement of 'errors'):
  - mean squared error (MSE)
  - log-likelihood
  - TD error (in deep reinforcement learning)
  - dis-similarities
  - ...

- Two general approaches of estimating the parameters:
  - iterative approaches
    - Gradient descent (MLE), Variational Bayesian Analysis, (mean-field, newton, ...)
    - Quick computing, but may reach a local minimum
  - sampling approaches
    - MCMC, Gibson sampling, Grid search...
    - Computationally expensive, hard to sample for extreme complex models

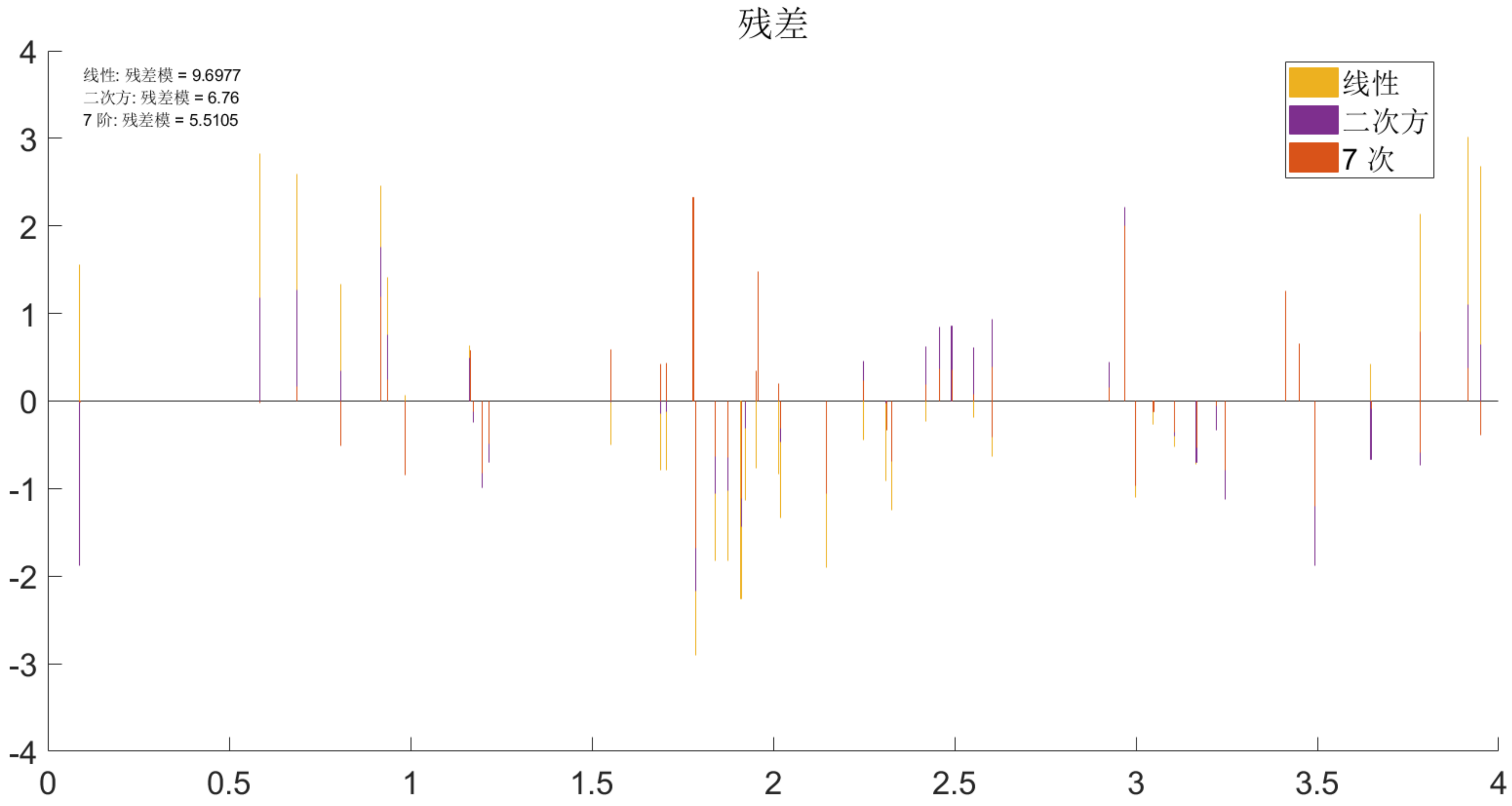
Introduction



Introduction



Introduction



Introduction

First order polynomial (Linear regression)

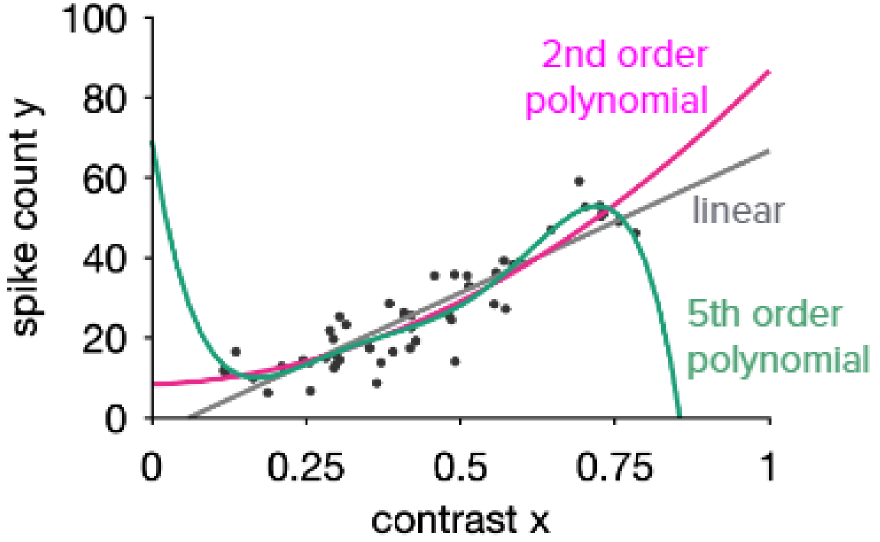
$$y = \theta_1 x + \theta_0$$

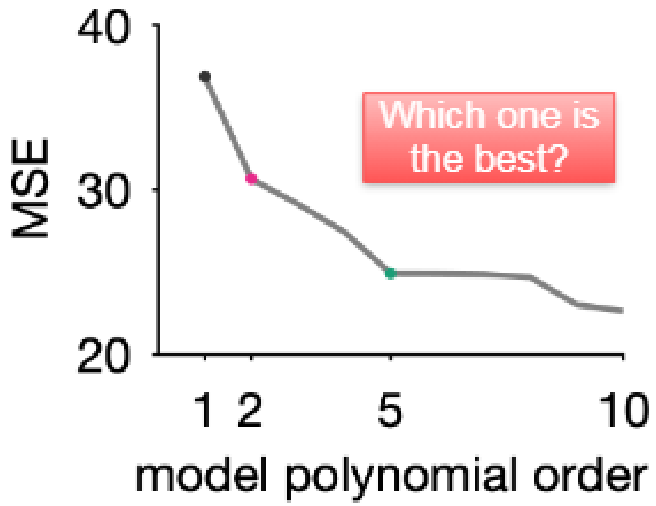
Second order polynomial

$$y = \theta_2 x^2 + \theta_1 x + \theta_0$$

Higher order polynomial

$$y = \sum_{p=0}^{5} \theta_p x^p$$

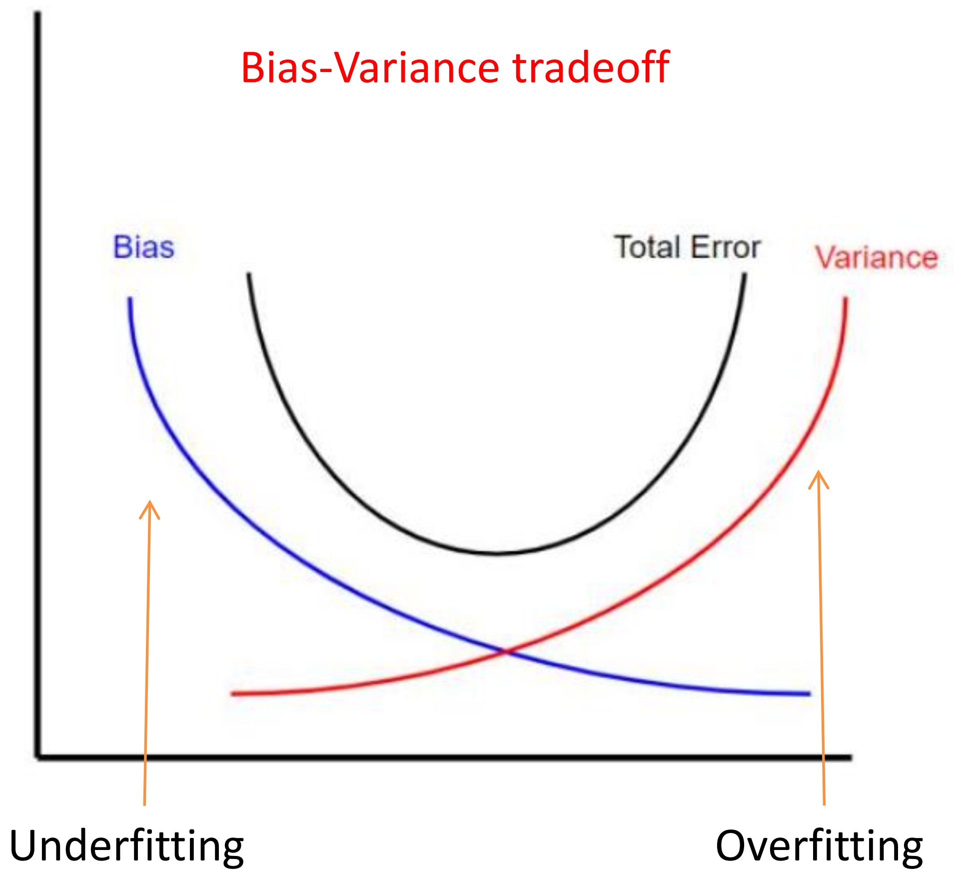


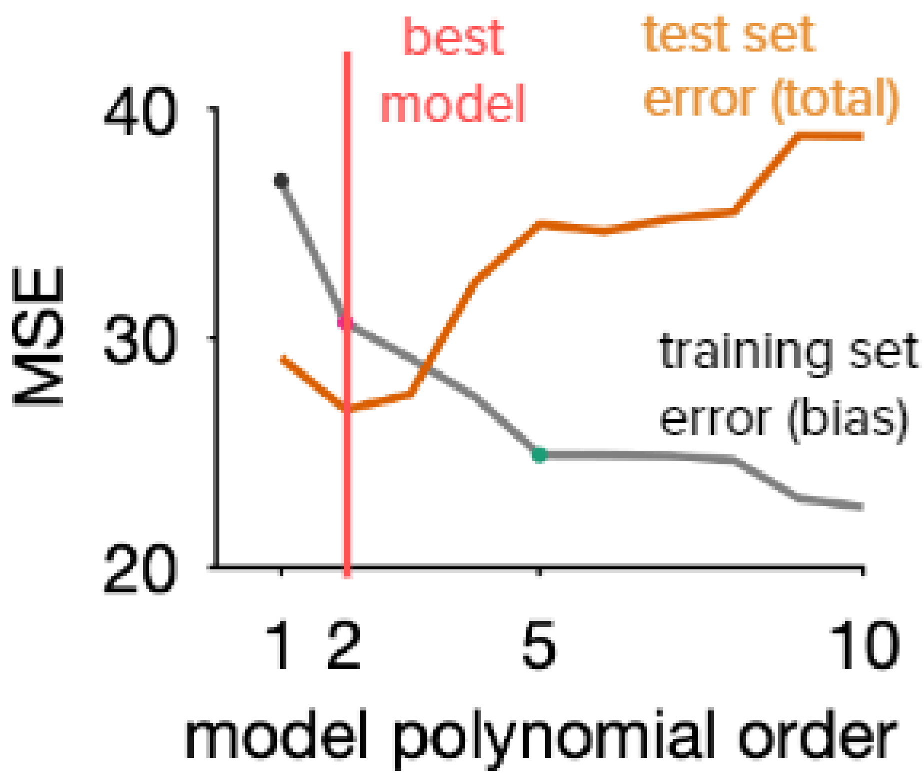


Overfitting!

Neuromatch Academy Computational Neuroscience Summer school

Introduction





- How to avoid overfitting?
  - Train with more data!
  - Machine learning:
    - Regularization
    - Cross-validation
    - Dropout
    - …
  - Psychology:
    - Model comparison (a simpler model)
    - Information criteria (AIC, BIC, DIC, LOOIC…)

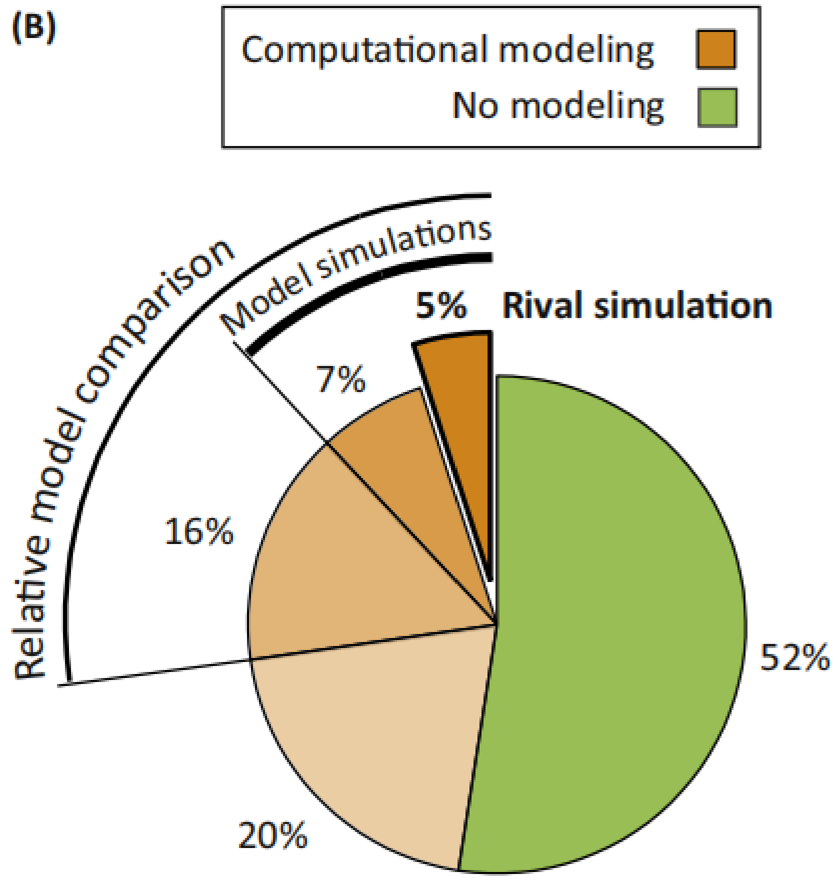
Summary

- What is computational modeling? Why do we need computational modeling?
- Basic concepts of a model
- What is model fitting? Two general approaches for fitting.
- What is overfitting? Why do we need to avoid overfitting?

Standard approaches of modeling

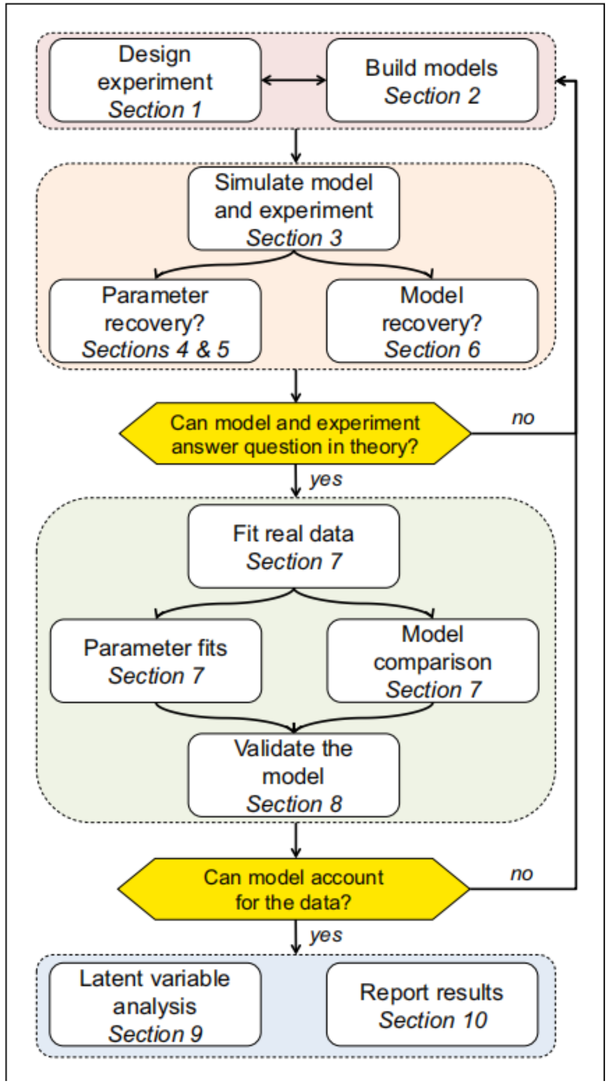
Modeling requires standards

- Like all other tools, computational modeling requires golden standards to be used.



Palminteri, et.al. 2017

Ten Simple Rules

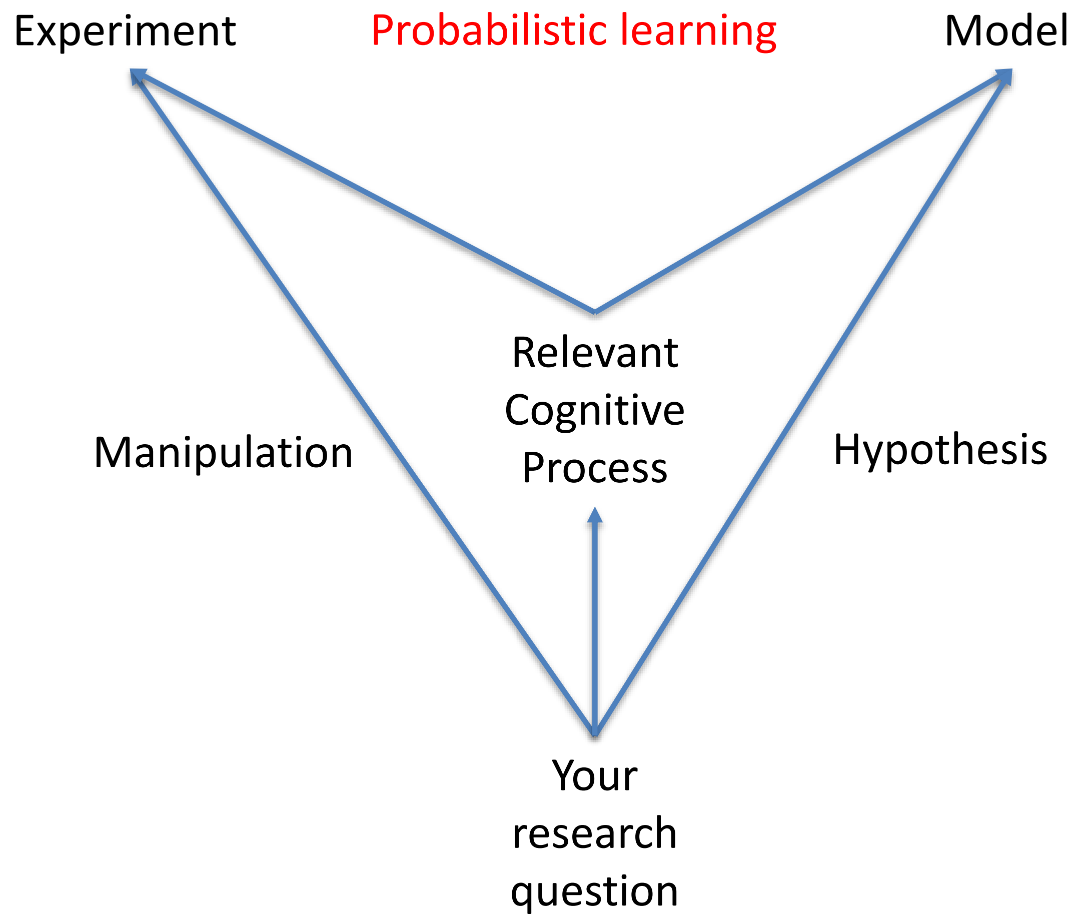


Wilson, Colins,2019

Step 1: experiment vs. model

Probabilistic learning

Health vs. depression



Health: quicker
Depression: slower

e.g. What is difference between healthy people and those with depression in probabilistic learning?

Step2: Simulation

Model ⟶ Fake Data

Pre-set parameter

Health (N=30)

Depression (N=30)

Model: Reinforcement learning model

Environment(Task): Probability reversal learning task

$$PE = R - v_{i,t}$$

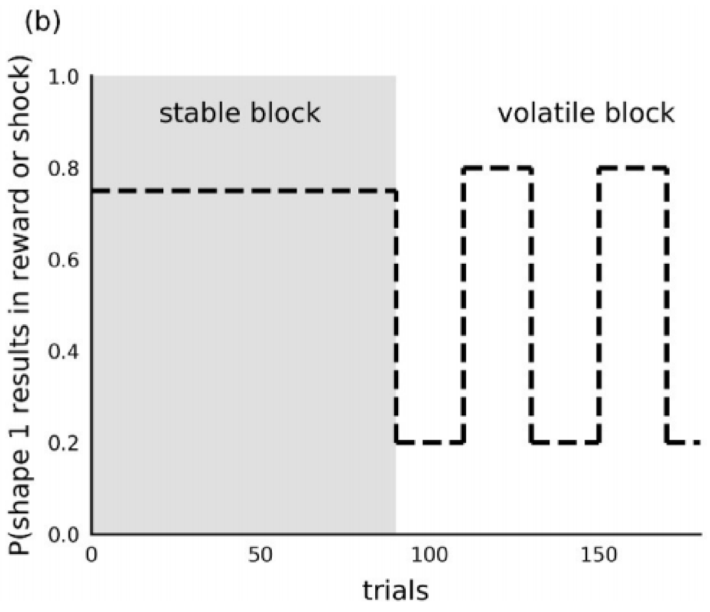
Prediction Error

$$v_{i,t+1} = v_{i,t} + \alpha * PE$$

α:learning rate

$$P(i, t+1) = \frac{e^{v_{i,t+1}}}{\sum e^{v_{i,t+1}}}$$

Rescorla, R. A., & Wagner, A. R. (1972)



Step-by-Step Simulation

1.Define the environment

```
#adopt WSLS strategy
p_reward1=c(0.8,0.2)
p_reward2=c(0.2,0.8)
```

2. Define (an) agent(s)

- Action(choice)
- Reward(feedback)
- Policy (strategy)
- Hidden States…

```
if (t==1){
  choice[t]=sample(c(1:2), size = 1, replace = T, prob = p_choose)
```

```
v[choice[t]]=v[choice[t]]+lr*(tmp_reward-v[choice[t]])
```

3.Define the environment response

```
tmp_reward=sample(c(1,-1), size = 1, replace = T, prob = p_reward[choice[t],])
accumulated_reward2[t]=accumulated_reward2[t-1]+tmp_reward
```

4.Set and run the simulation process!

for-loop: trials and agents

5.Analyze and visualize your simulated data!

Model Validation

Other Possible Models

Model Recovery

Simulation

Parameter Recovery

Model → Fake Data → Recovered parameter

Pre-set parameter

Fit your model

How close are they?

Collect your own data

- If your model is fine with your experiment…
- Run the experiment!
- Why not run your model while collecting your data?

After you collect your data

Model Fitting

Model Comparison

Simulation

Parameter Recovery

Bayesian Model Selection

Other Possible Models

Compare again **Model Recovery**

True Data

Fit your models

Select a proper model

The proper model

True parameter

Fake Data

Refit

The proper model

Recovered parameter

How close are they?

How close are they?

Simulation Accuracy

Recovered Correlation

Analyze results from your model

- Your parameters:
  - Compare different parameters across your conditions;
  - Correlate your parameters with something you measure;
  - Even more flexible analysis approach (depends on your questions)

- Your latent variables:
  - Value
  - Prediction error/ TD error
  - Compare, correlate, cluster……
  - Also more flexible analysis approach

Report your model result

- What your model (usually winning model) reflects about the cognitive, psychological or neuronal process?
- What your parameters mean? How are they bounded?
- Your fitting methods (specific details)
- Your parameter results (usually winning model); frequentist vs. Bayesian.
- Model Comparison (the criterion you choose, BMS result)
- How well is your model fitted, simulated(predicted) and recovered?
- How’s your model relevant analysis conducted?
- How are the results above able to help answer your questions?

Model Philosophy

Three levels of description *(David Marr, 1982)*

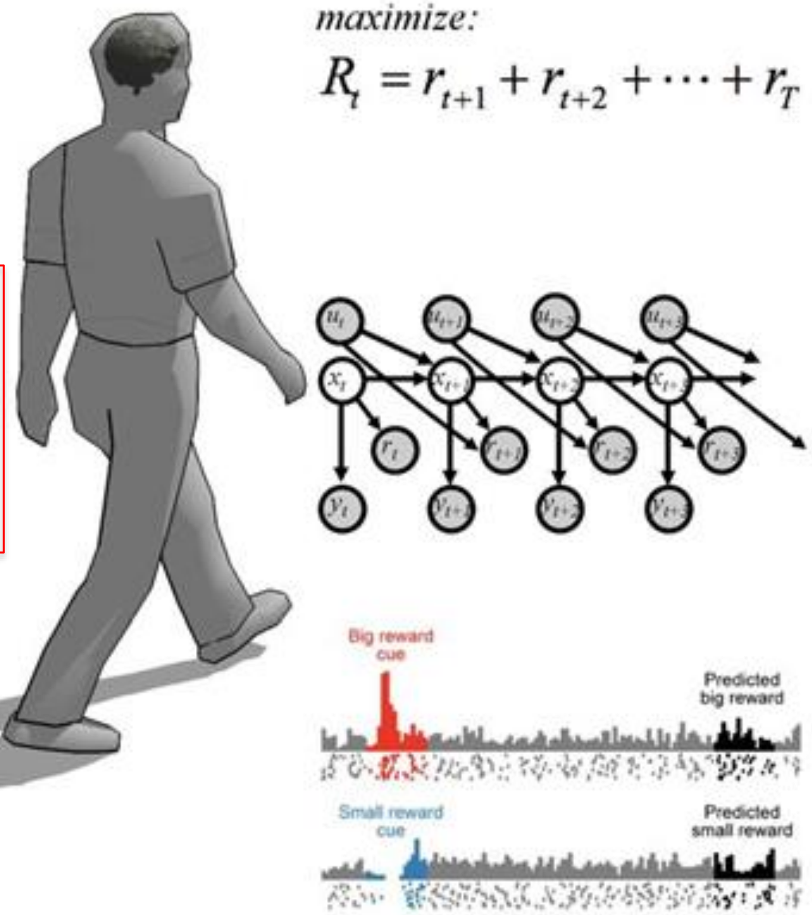
Computational
Why do things work the way they do?
What is the goal of the computation?
What are the unifying principles?

Algorthmic
What representations can implement such computations?
How does the choice of representations determine the algorithm?

Implementational
How can such a system be built in hardware?
How can neurons carry out the computations?

maximize:

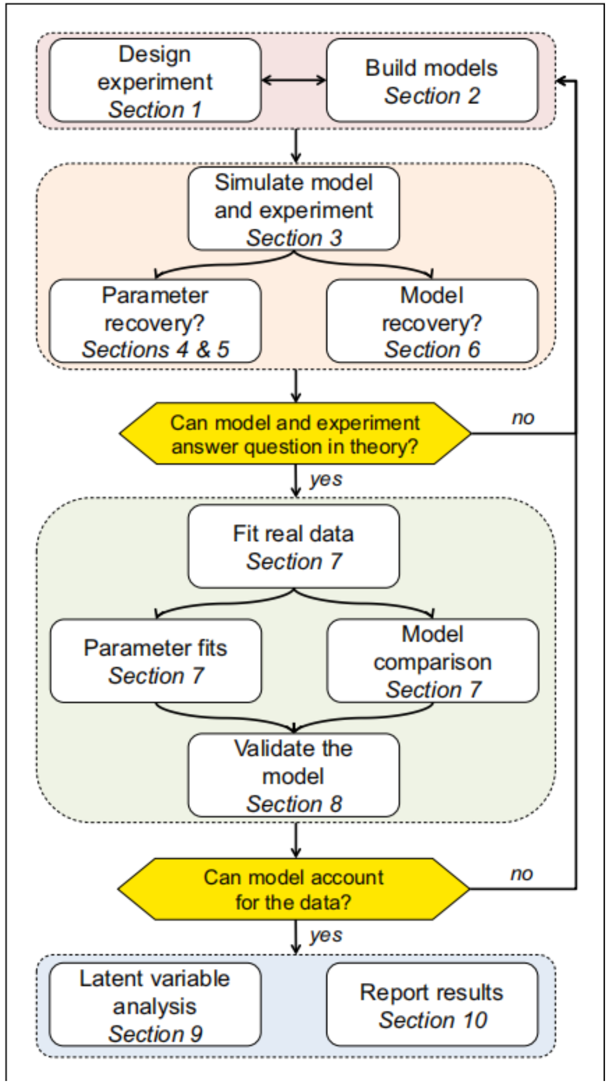
$$R_t = r_{t+1} + r_{t+2} + \cdots + r_T$$



All models are wrong, **but** some are useful……
---– George E. P. Box

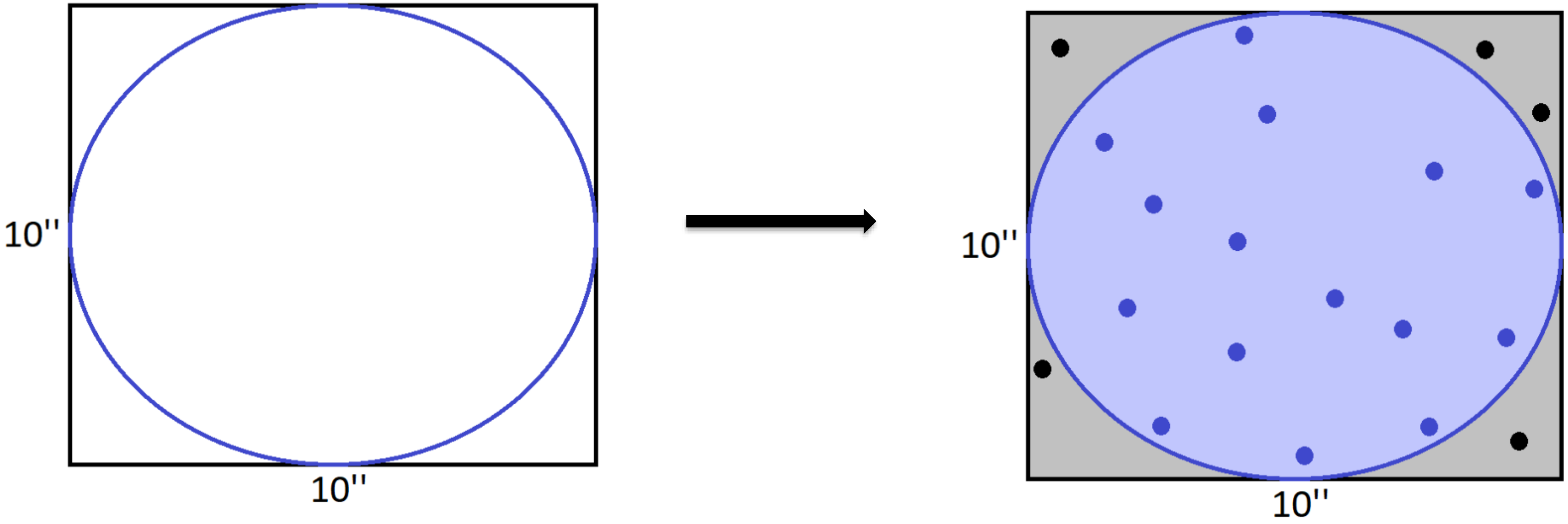
All models are imperfect, **but** some are useful……

Summary


Design
experiment
Section 1
Build models
Section 2
Simulate model
and experiment
Section 3
Parameter
recovery?
Sections 4 & 5
Model
recovery?
Section 6
Can model and experiment
answer question in theory?
no
yes
Fit real data
Section 7
Parameter fits
Section 7
Model
comparison
Section 7
Validate the
model
Section 8
Can model account
for the data?
no
yes
Latent variable
analysis
Section 9
Report results
Section 10

MCMC and Stan

Monte Carlo Sampling



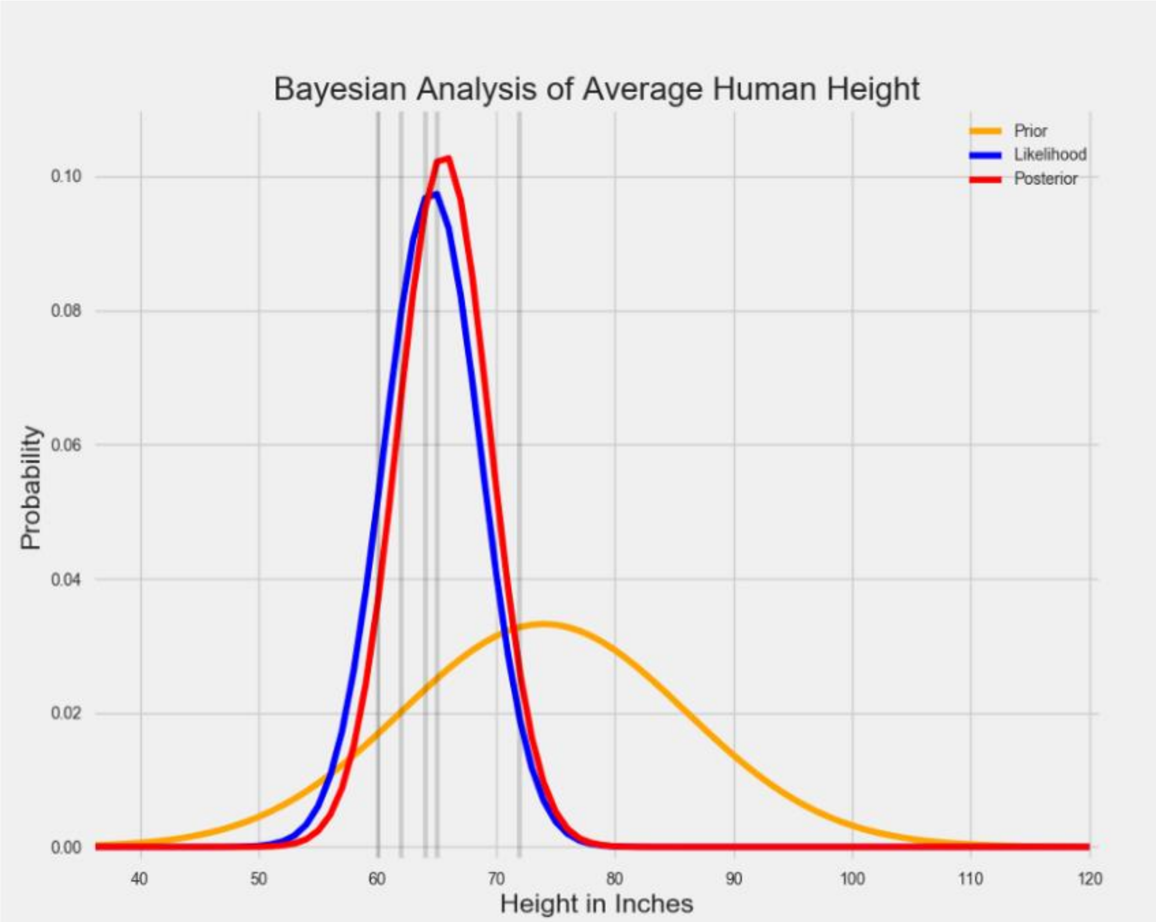
[A Zero-Math Introduction to Markov Chain Monte Carlo Methods]

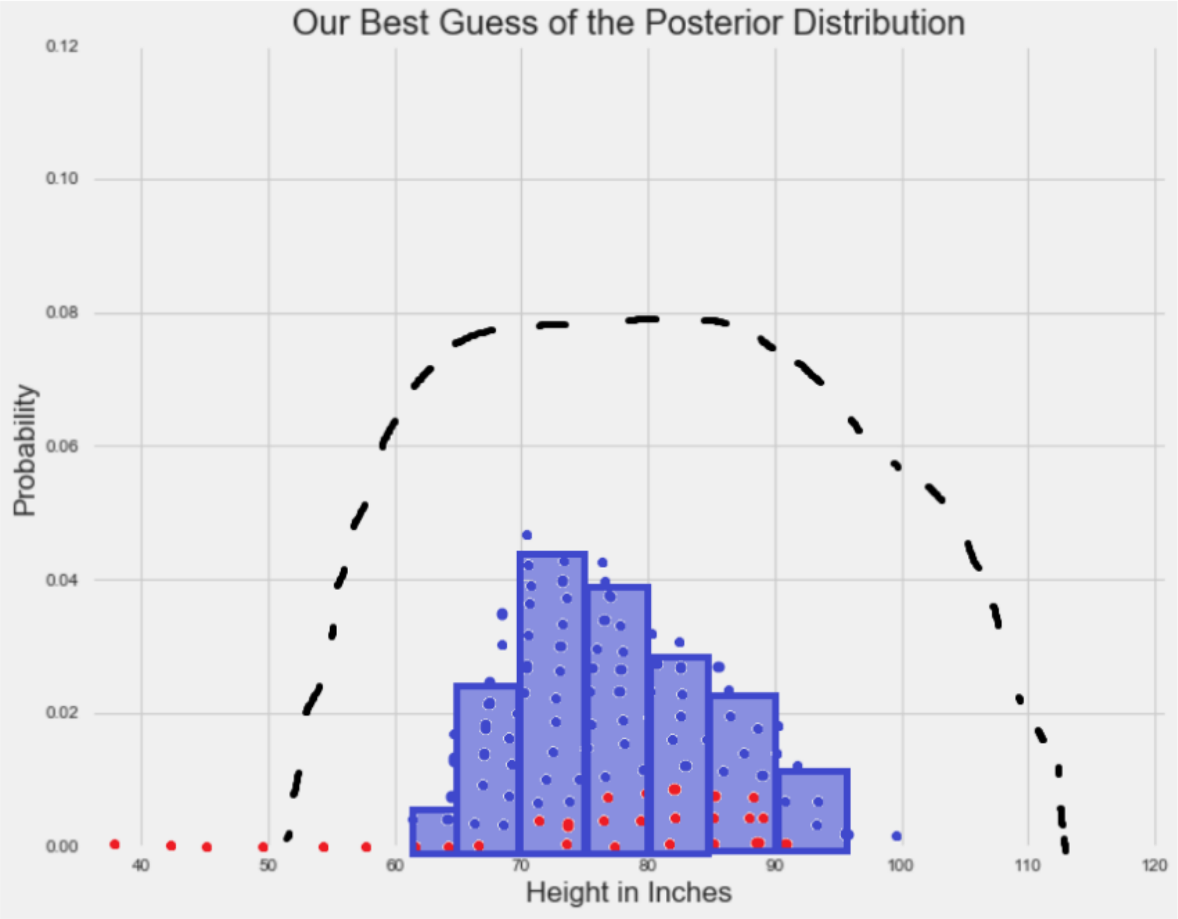
Bayes Theorem

parameters data

$$\underbrace{P(\theta|D)}_{\text{Posterior}} \propto \underbrace{P(D|\theta)}_{\text{Likelihood}}\underbrace{P(\theta)}_{\text{Prior}}$$

MC and Bayesian Theorem



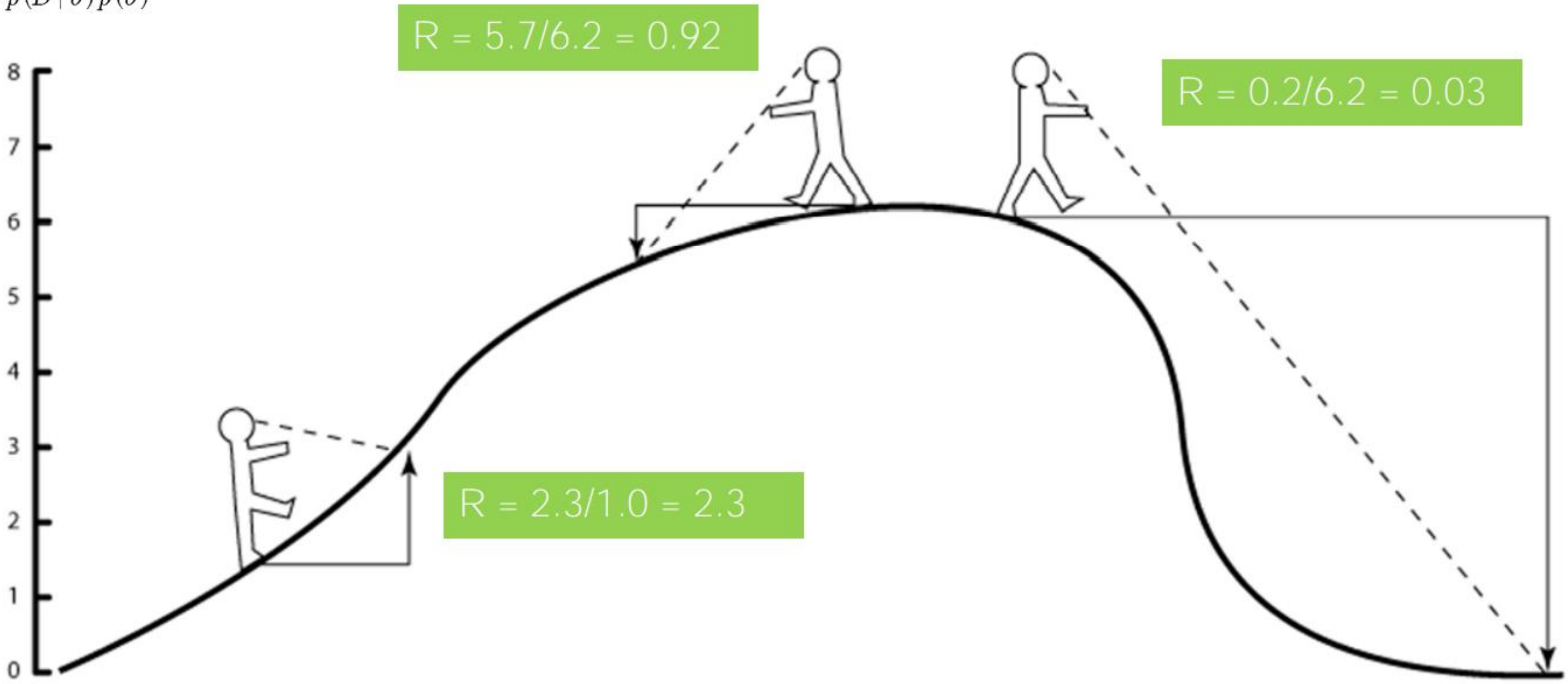


If Only Monte Carlo

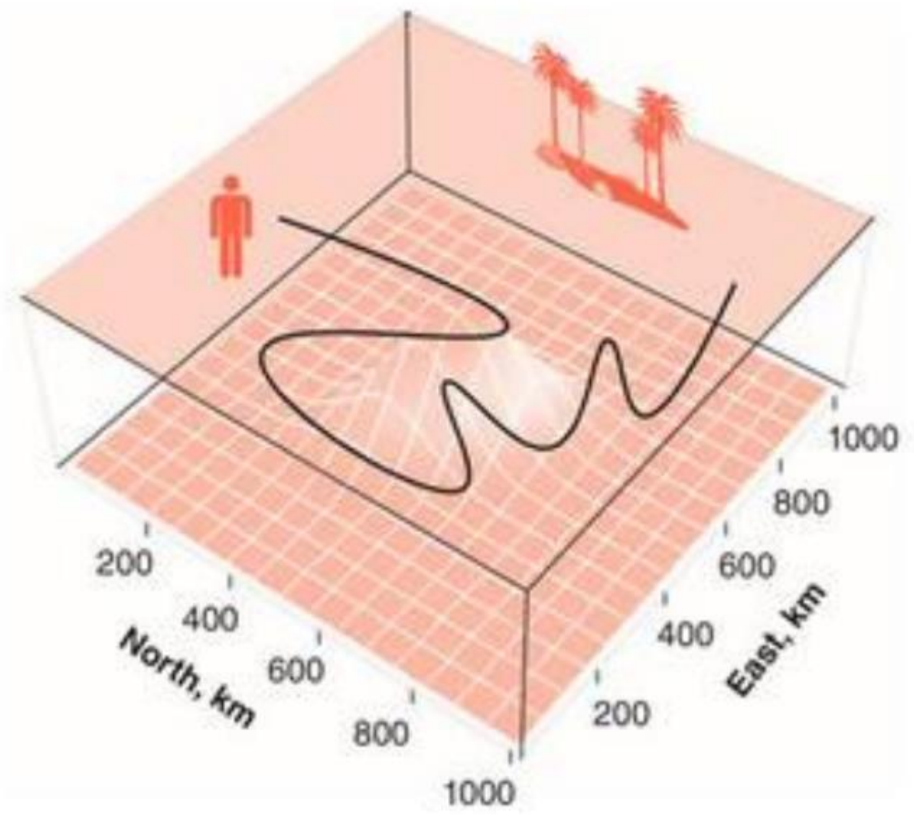
What is different from grid search?

Markov Chain

$$p(\theta \mid D) \propto p(D \mid \theta) p(\theta)$$



An MCMC Robert in 3D



Lambert (2018)

Algorithms

- Rejection Sampling
- Importance Sampling
- Metropolis Sampling
- Gibbs Sampling
- Hamilton MC (Stan)

Stan

- 1. an independent programming language based on C++
- 2. Can be interacted with different platforms (e.g., R, Matlab and Python)

Steps to use Stan

- 1. Prepare your data
- 2. Write your model in Stan
- 3. Check with stan codes
- 4. Set up Stan options
- 5. Run/Call Stan files
- 6. Post-analysis

Stan file

```
data {
  //... read in external data...
}

transformed data {
  //... pre-processing of data ...
}

parameters {
  //... parameters to be sampled by HMC ...
}

transformed parameters {
  //... pre-processing of parameters ...
}

model {
  //... statistical/cognitive model ...
}

generated quantities {
  //... post-processing of the model ...
}
```

Stan Syntax Cheatsheet.pdf

A minimal Stan program implementing a binomial model.

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

Call Stan file

Running a Stan program is usually done from another language such as Python or R.

(Here assuming `model_string` contains the model from the last slide.)

```
library(rstan)
data_list <- list(n = 30, x = 10)
s <- stan(model_code = model_string,
          data = data_list)
```

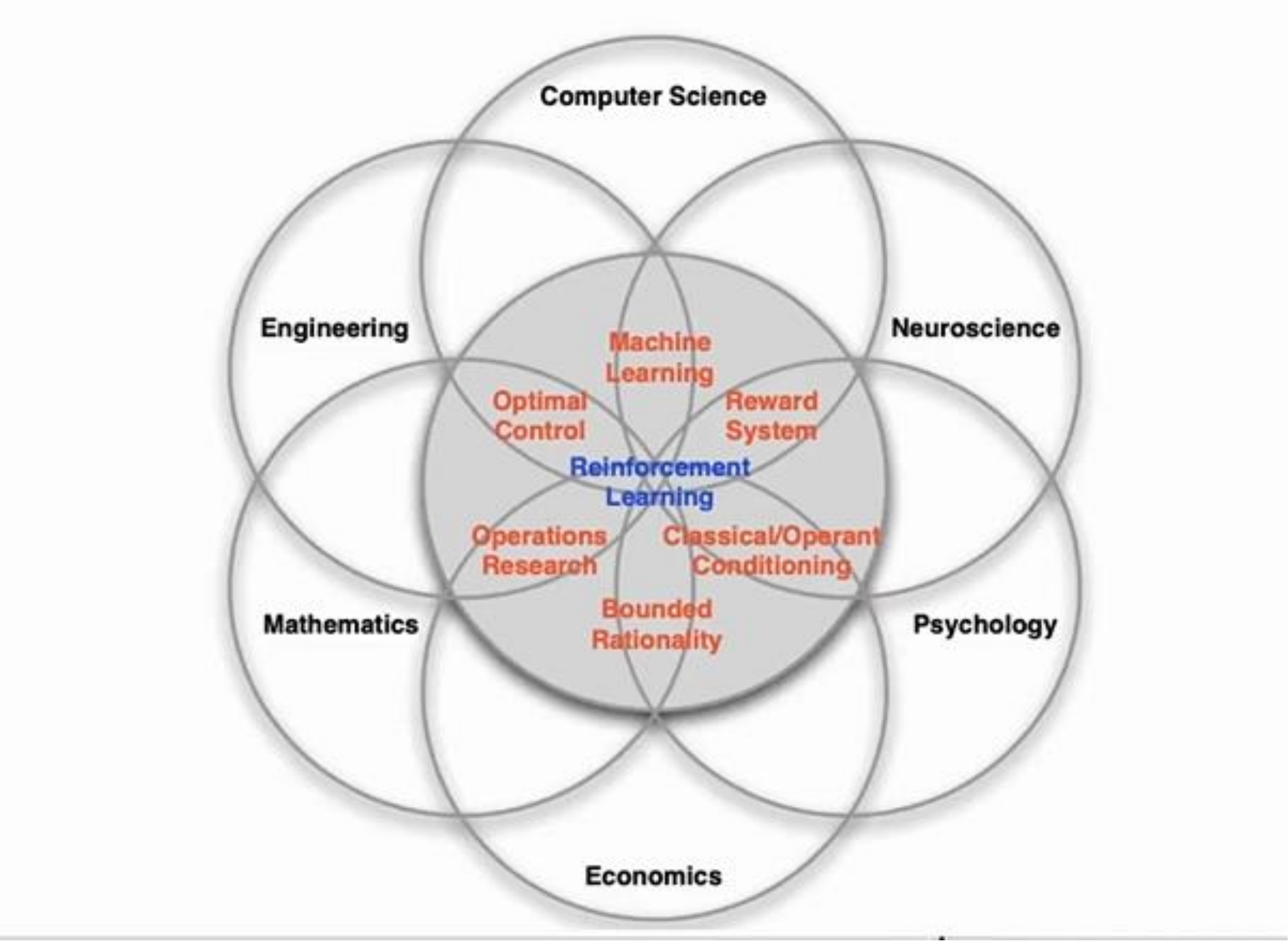
```
import pystan
data_list = dict(n = 30, x = 10)
s = pystan.stan(model_code = model_string,
                data = data_list)
```

Summary

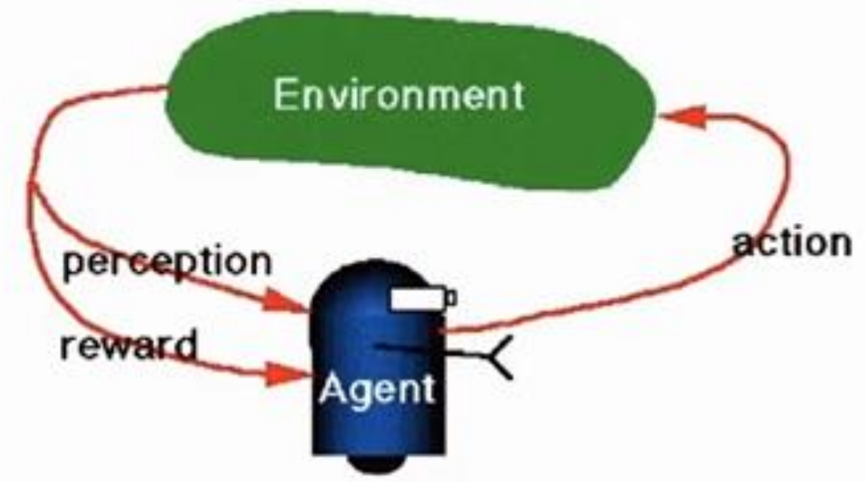
- 1. What is MCMC?
- 2. What is Stan?
- 3. How to use Stan?

Practices: implementing a reinforcement learning model in Rstan

Brief introduction about RL



Brief introduction about RL



- Agent perceives the environment state s
- Agent takes action a
- Agent receive reward r
- And agent adjust his action a'

RL tutorial

Rescorla-Wagner Model

$$PE = R - v_{i,t}$$

Prediction Error

$$v_{i,t+1} = v_{i,t} + \alpha * PE$$

α:learning rate

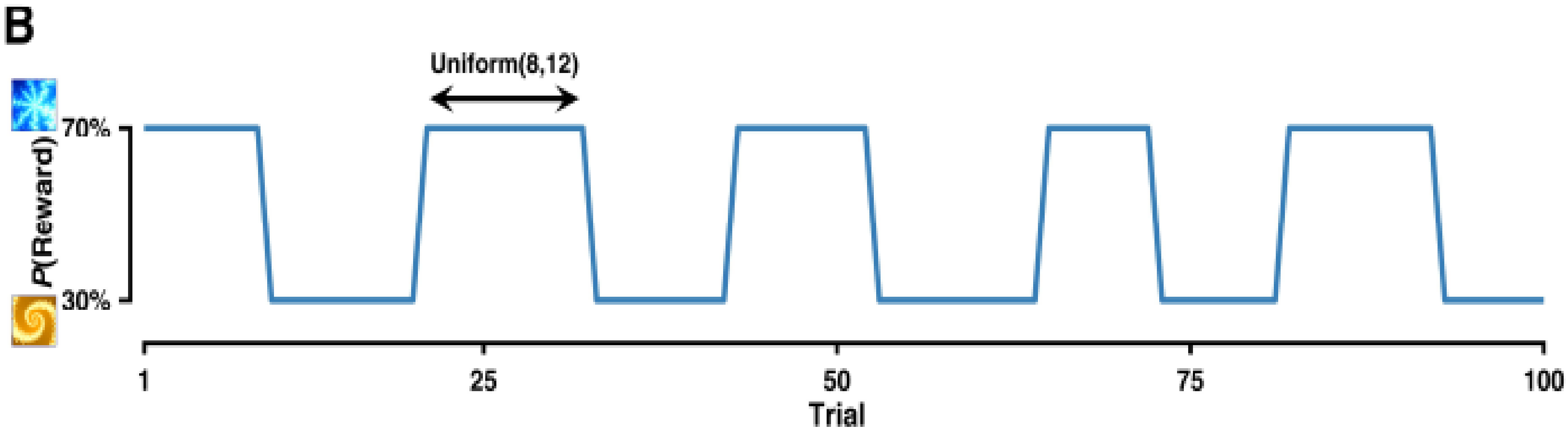
$$P(i, t+1) = \frac{e^{v_{i,t+1} * \beta}}{\sum e^{v_{i,t+1} * \beta}}$$

Softmax function

$$\beta = 3^{\tau} - 1$$

Rescorla, R. A., & Wagner, A. R. (1972)

Probability Reversal Learning



(Zhang, Glascher, 2020)

Run the simulation

Task setting

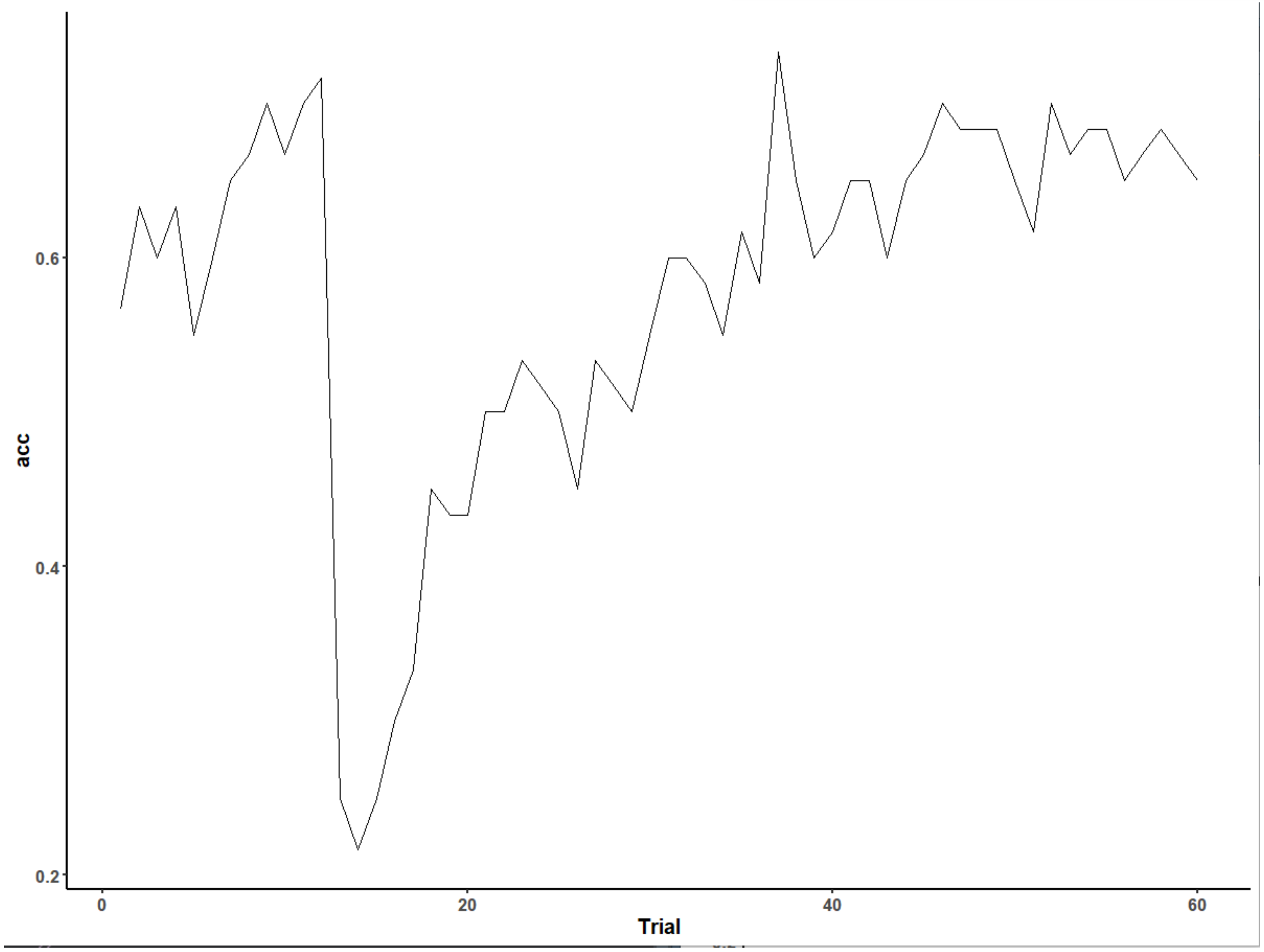
```
##task setting (probability reversal learing task)
set.seed(2021)
reverse_trials=sample(c(8,9,10,11,12,13), replace = F)
ntrials=60
reward_prob=matrix(c(0.7,0.3,0.3,0.7),ncol=2,nrow=2)
```

Agent setting

```
##agent setting
nsubjects=60
lr=rnorm(n=nsubjects,mean=0.4,sd=0.2)
tau=rnorm(n=nsubjects,mean=3,sd=1)
```

```
#run the simulation
agent_data=list()
for (i in 1:nsubjects){
  tmp_lr=lr[i]
  tmp_tau=tau[i]
  v=matrix(rep(0,2*ntrials),nrow=2,ncol=ntrials)
  p=c(0,0)
  consistency=3**tmp_tau-1
  choice=rep(0,ntrials)
  reward=rep(0,ntrials)
  correct=rep(0,ntrials)
  n_reverse=1
  count_trial=0
  for (t in 1:ntrials){
    if(t>1){
      v[choice[t-1],t]<-v[choice[t-1],t-1]+tmp_lr*(reward[t-1]-v[choice[t-1],t-1])
    }
    p<-softmax(v[,t]*consistency)
    choice[t]=sample(c(1,2),prob = p,replace=T)
    reward[t]=sample(c(1,0),prob = reward_prob[choice[t],],replace=T)
    count_trial=count_trial+1
    if(count_trial==reverse_trials[n_reverse]){
      reward_prob=1-reward_prob
      count_trials=0
      n_reverse=n_reverse+1
    }
    if (reward_prob[choice[t],1]==0.7){
      correct[t]=1
    }else{
      correct[t]=0
    }
  }
  agent_data[[i]]=list(c(1:ntrials),choice,reward,correct,v,tmp_lr,tmp_tau)
}
```

Visualization



R and RStan

$$\boxed{\text{main.R}} \xrightarrow{\text{call}} \boxed{\text{model.stan}} \xrightarrow{\text{compelling}} \boxed{\text{C++}}$$

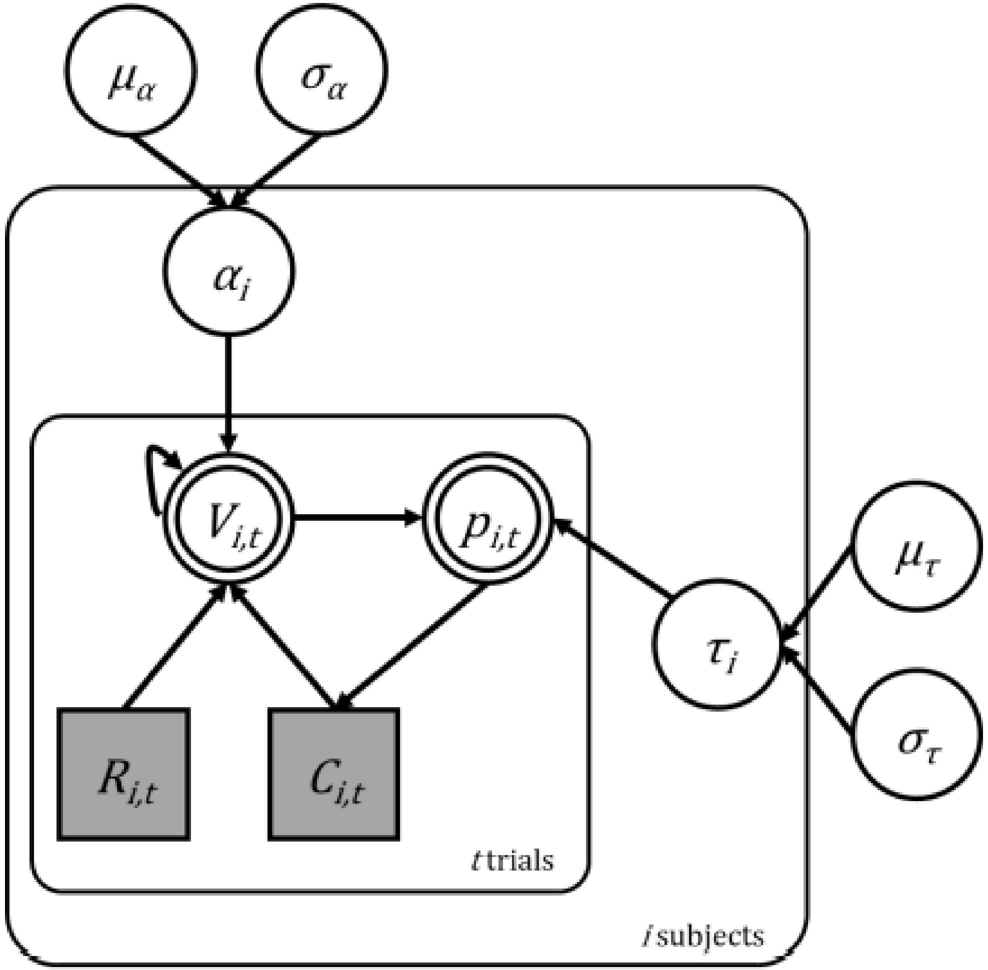
Stan model

Define your data

```
data {
  int<lower=1> nSubjects;
  int<lower=1> nTrials;
  int<lower=1,upper=2> choice[nSubjects, nTrials];
  real<lower=0, upper=1> reward[nSubjects, nTrials];
}
```

```
transformed data {
  vector[2] initV;  // initial values for V
  initV = rep_vector(0.0, 2);
}
```

Stan model



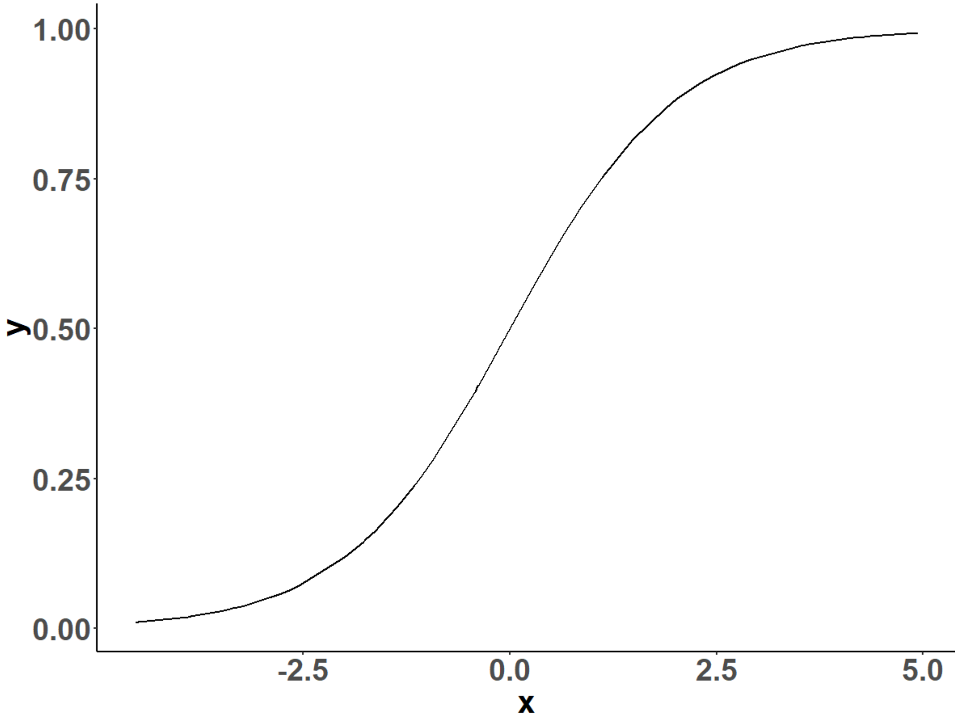
Hierarchical Bayesian Model

Stan model

```
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=5>[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] ) * 5;
  }
}
```

Phi_approx: sigmoid

$$y = \frac{1}{1 + e^{-x}} \qquad y \in (0,1)$$



Stan model

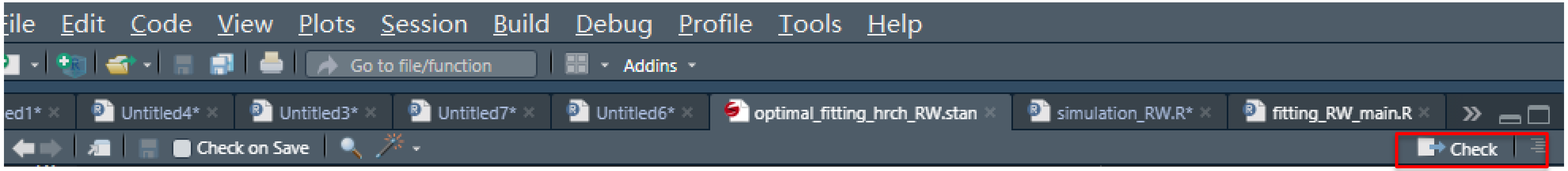
```
model {
  lr_mu_raw  ~ normal(0,1);
  tau_mu_raw ~ normal(0,1);
  lr_sd_raw  ~ cauchy(0,3);
  tau_sd_raw ~ cauchy(0,3);

  lr_raw  ~ normal(0,1);
  tau_raw ~ normal(0,1);

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

    for (t in 1:nTrials) {
      choice[s,t] ~ categorical_logit((pow(3,tau[s])-1)* v );

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

Stan model

```
generated quantities {
  real<lower=0,upper=1> lr_mu;
  real<lower=0,upper=3> tau_mu;

  real log_lik[nSubjects];

  int y_pred[nSubjects, nTrials];

  lr_mu  = Phi_approx(lr_mu_raw);
  tau_mu = Phi_approx(tau_mu_raw) * 5;
  y_pred = rep_array(-999,nSubjects ,nTrials);

  { // local block
    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] | ((pow(3,tau[s])-1) * v));
          y_pred[s,t] = categorical_logit_rng( (pow(3,tau[s])-1) * v );
          pe = reward[s,t] - v[choice[s,t]];
          v[choice[s,t]] = v[choice[s,t]] + lr[s] * pe;
        }
    }
  }
}
```

Stan file

- Check your codes




```
> rstan:::rstudio_stanc("D:/Hanbo/psychology/科研/实验室申请/北大周晓林实验室/modeling 教学/codes/optimal_fitting_hrch_RW.stan")
D:/Hanbo/psychology/科研/实验室申请/北大周晓林实验室/modeling 教学/codes/optimal_fitting_hrch_RW.stan is syntactically correct.
> |
```

- Bugs?

  [The Stan Forums (mc-stan.org)](https://discourse.mc-stan.org)

R file

- Prepare data
- Specify stan file
- Set options for fitting
- Run the fitting process

R file

Original data: list ⟶ Target data: array

```r
#suppose we have completed simulation process and get the agent_data

nSubjects <- length(agent_data)
nTrials   <- length(agent_data[[1]][[1]])

##reshape the data from a list to an array to be fitted
data_array<-array(rep(0,nSubjects*nTrials*2),dim=c(nSubjects, nTrials, 2))
for (i in 1:nSubjects){
    for(t in 1:nTrials){
        data_array[i,t,1]=agent_data[[i]][[2]][t]
        data_array[i,t,2]=agent_data[[i]][[3]][t]
    }
}


dataList <- list(nSubjects=nSubjects,
                 nTrials=nTrials,
                 choice=data_array[,,1],
                 reward=data_array[,,2])
```

R file

Your own path

```
rstan_options(auto_write = TRUE)
options(mc.cores = 4)

modelFile <- 'D:\\Hanbo\\psychology\\科研\\实验室申请\\北大周晓林实验室\\modeling 教学\\codes\\optimal_fitt

nIter     <- 16000
nChains   <- 4
nWarmup   <- floor(nIter/2)
nThin     <- 1

cat("Estimating", modelFile, "model... \n")
startTime = Sys.time(); print(startTime)
cat("Calling", nChains, "simulations in Stan... \n")

fit_rl <- stan(modelFile,
               data    = dataList,
               chains  = nChains,
               iter    = nIter,
               warmup  = nWarmup,
               thin    = nThin,
               init    = "random",
               control  = list(adapt_delta=0.999, max_treedepth=100),
               seed    = 2021
               )

cat("Finishing", modelFile, "model simulation ... \n")
endTime = Sys.time(); print(endTime)
cat("It took",as.character.Date(endTime - startTime), "\n")
```

R file

```
run_RW_fitting <- function() {
    # ===========================================================================
    #### Construct Data ####
    # ===========================================================================
```

Run the whole function to define the function

```
+ }
> fit_rl<-run_RW_fitting()
```

Call the function through command

R file

Environment History Connections Tutorial

Files Plots Packages Help Viewer

```
Click the Refresh button to see progress of the chains
starting worker pid=15180 on localhost:11696 at 15:45:07.415
starting worker pid=2264 on localhost:11696 at 15:45:07.572
starting worker pid=6536 on localhost:11696 at 15:45:07.742
starting worker pid=11176 on localhost:11696 at 15:45:07.899

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

SAMPLING FOR MODEL 'optimal_fitting_hrch_RW' NOW (CHAIN 2).

SAMPLING FOR MODEL 'optimal_fitting_hrch_RW' NOW (CHAIN 3).
Chain 1:
Chain 1: Gradient evaluation took 0.004 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 40 seconds.
Chain 1: Adjust your expectations accordingly!
Chain 1:
Chain 1:
Chain 1: Iteration:     1 / 16000 [  0%]  (Warmup)

SAMPLING FOR MODEL 'optimal_fitting_hrch_RW' NOW (CHAIN 4).
Chain 2:
Chain 2: Gradient evaluation took 0.003 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 30 seconds.
Chain 2: Adjust your expectations accordingly!
Chain 2:
Chain 2:
Chain 2: Iteration:     1 / 16000 [  0%]  (Warmup)
Chain 3:
Chain 3: Gradient evaluation took 0.003 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 30 seconds.
Chain 3: Adjust your expectations accordingly!
Chain 3:
Chain 3:
Chain 3: Iteration:     1 / 16000 [  0%]  (Warmup)
Chain 4:
Chain 4: Gradient evaluation took 0.004 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 40 seconds.
Chain 4: Adjust your expectations accordingly!
Chain 4:
Chain 4:
Chain 4: Iteration:     1 / 16000 [  0%]  (Warmup)
```

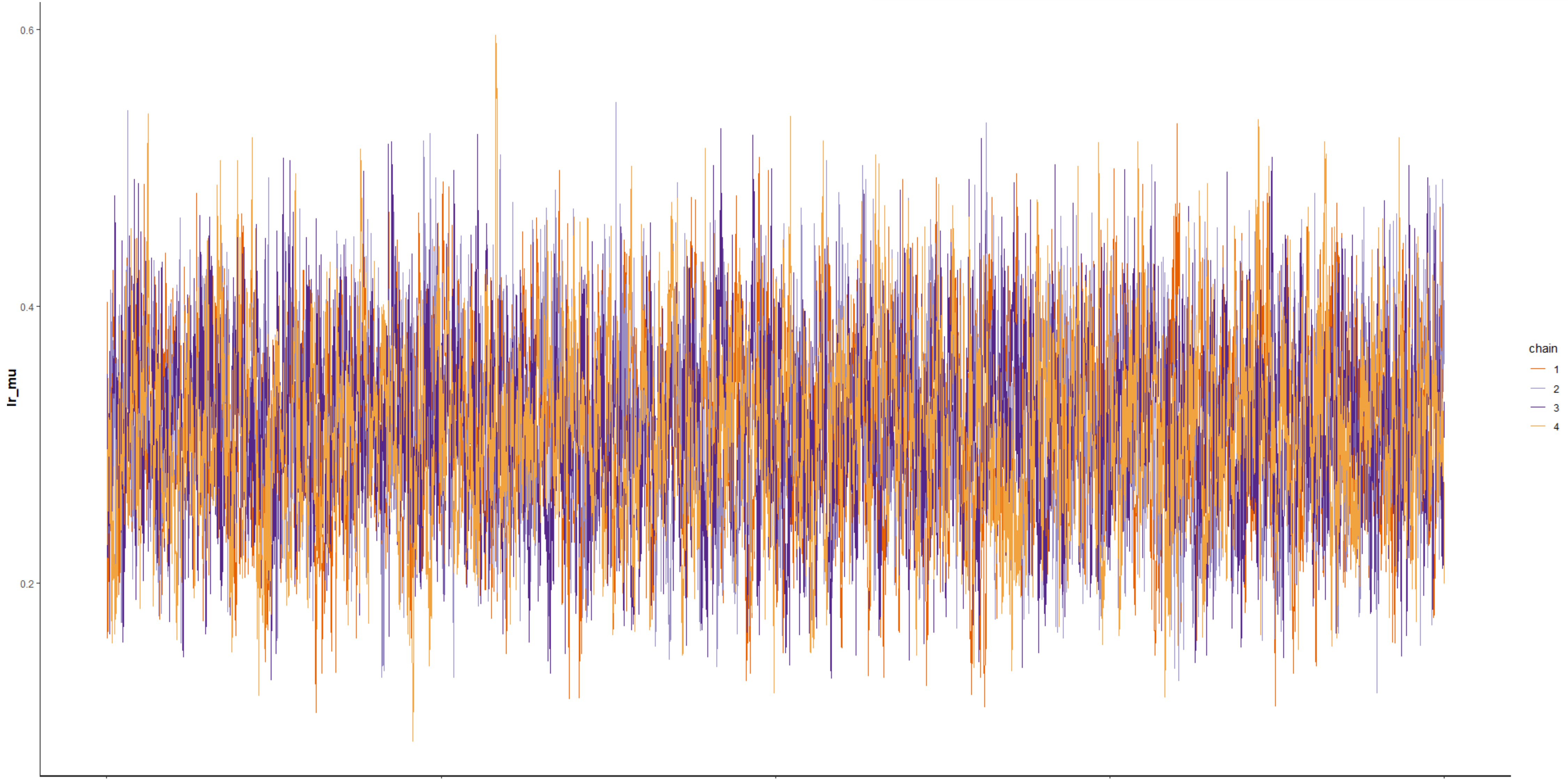
Result Diagnose

```
> print(fit_rl)
Inference for Stan model: optimal_fitting_hrch_RW.
4 chains, each with iter=16000; warmup=8000; thin=1;
post-warmup draws per chain=8000, total post-warmup draws=32000.

             mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
lr_mu_raw   -0.50    0.00 0.18 -0.87 -0.62 -0.49 -0.38 -0.16  2701    1
tau_mu_raw   1.83    0.00 0.42  1.12  1.53  1.79  2.08  2.77 11334    1
lr_sd_raw    1.06    0.00 0.16  0.78  0.94  1.05  1.16  1.42  3042    1
tau_sd_raw   2.00    0.01 0.61  1.03  1.58  1.93  2.34  3.41  4095    1
lr_raw[1]   -0.91    0.01 1.18 -2.78 -1.82 -1.08 -0.07  1.55  9682    1
lr_raw[2]    0.69    0.00 0.36  0.05  0.44  0.67  0.92  1.46 14462    1
lr_raw[3]   -0.22    0.00 0.46 -1.15 -0.54 -0.21  0.10  0.65 17157    1
lr_raw[4]    0.49    0.00 0.35 -0.13  0.26  0.47  0.71  1.24 13418    1
lr_raw[5]    1.84    0.00 0.49  0.85  1.52  1.85  2.16  2.78 13315    1
lr_raw[6]   -0.27    0.00 0.50 -1.28 -0.60 -0.25  0.09  0.64 19402    1
lr_raw[7]    0.49    0.00 0.34 -0.10  0.26  0.46  0.70  1.25 12884    1
lr_raw[8]   -0.70    0.00 0.28 -1.25 -0.88 -0.69 -0.51 -0.15  3689    1
lr_raw[9]    0.63    0.00 0.45 -0.42  0.39  0.69  0.93  1.39 12864    1
lr_raw[10]   1.57    0.01 0.68  0.24  1.08  1.63  2.06  2.81 12984    1
lr_raw[11]  -1.31    0.01 0.37 -2.05 -1.56 -1.31 -1.07 -0.61  3577    1
lr_raw[12]   0.77    0.01 0.75 -0.43  0.23  0.63  1.29  2.35  7887    1
lr_raw[13]   0.61    0.00 0.43 -0.13  0.31  0.57  0.86  1.57 15384    1
lr_raw[14]   0.36    0.01 0.70 -1.16 -0.10  0.48  0.87  1.46 18117    1
lr_raw[15]   0.53    0.00 0.28  0.01  0.33  0.51  0.70  1.13  9824    1
```

Rhat should generally be smaller than 1.01

[‘shinystan’]

A good example



Stan Sampling Parameters

cognitive model
statistics
computing

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

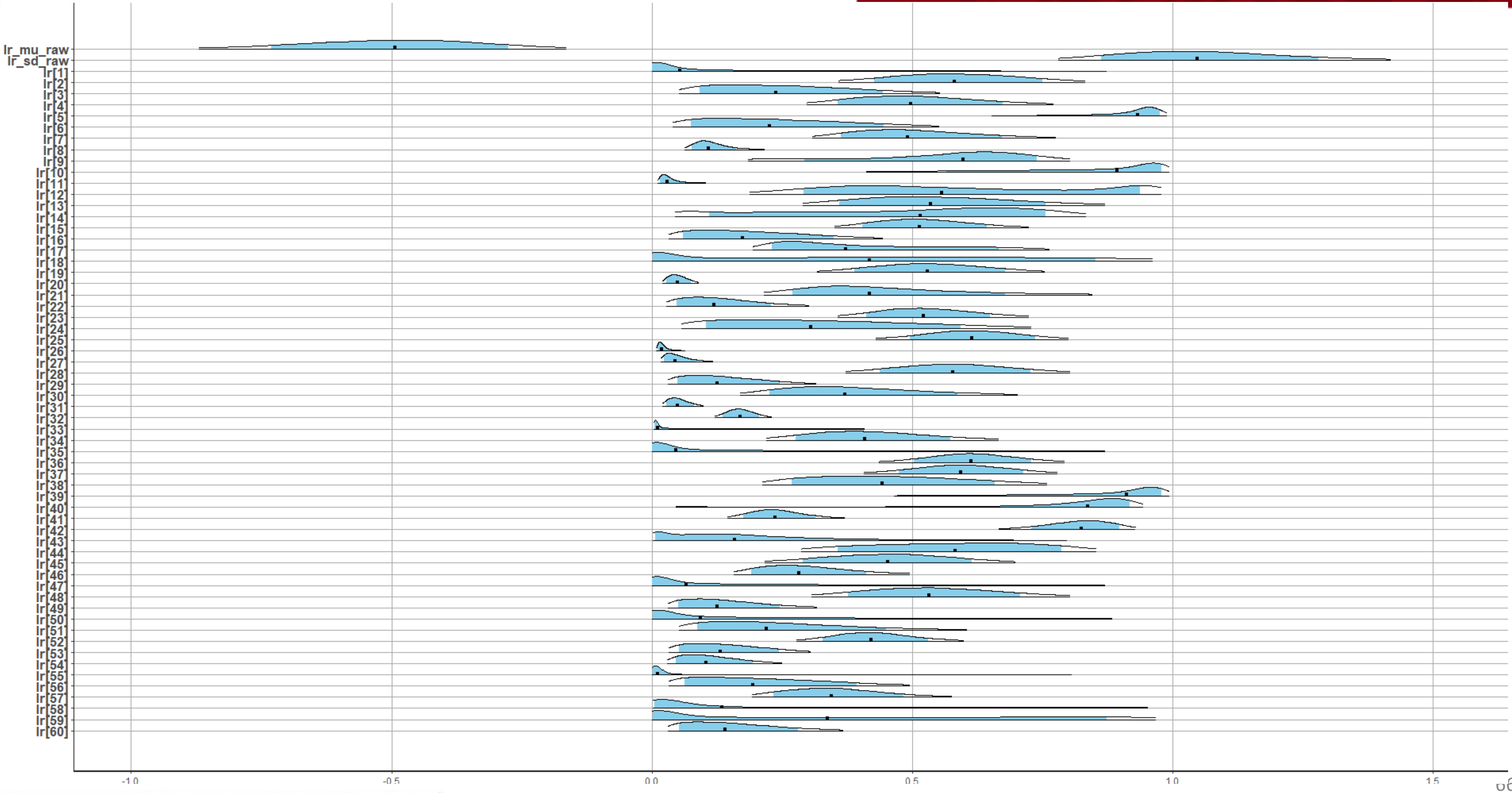
Typical adjustments

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

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

Stan Development Team (2016)

How to analyze?



How to analyze?

- Mean
- HDI=highest density interval (credential interval)

```
#' Compute Highest-Density Interval
#'
#' @description
#' Computes the highest density interval from a sample of representative values, estimated as shortest credible inte
#' Downloaded from John Kruschke's website \url{http://www.indiana.edu/~kruschke/DoingBayesianDataAnalysis/}
#'
#' @param samplevec A vector of representative values from a probability distribution (e.g., MCMC samples).
#' @param credMass A scalar between 0 and 1, indicating the mass within the credible interval that is to be estimate
#'
#' @return A vector containing the limits of the HDI
#'
#' @export

HDIofMCMC = function(sampleVec,
                     credMass = 0.95 ) {
    if ( class(sampleVec) == "mcmc.list" ) {
        sampleVec = as.matrix(sampleVec)
    }
    sortedPts = sort( sampleVec )
    ciIdxInc = floor( credMass * length( sortedPts ) )
    nCIs = length( sortedPts ) - ciIdxInc
    ciWidth = rep( 0 , nCIs )
    for ( i in 1:nCIs ) {
        ciWidth[ i ] = sortedPts[ i + ciIdxInc ] - sortedPts[ i ]
    }
    HDImin = sortedPts[ which.min( ciWidth ) ]
    HDImax = sortedPts[ which.min( ciWidth ) + ciIdxInc ]
    HDIlim = c( HDImin , HDImax )
    return( HDIlim )
}
```

How to analyze?

Statistical Rethinking

A Bayesian Course
with Examples
in R and Stan

Richard McElreath

Model Comparison

Generally, you can compare your models by using various indicators:

- Log-likelihood
- Alike Information Criterion (AIC)
- Bayesian Information Criterion (BIC)
- **Leave-one-out Information Criterion (LOOIC)**

```
##model comparison
    library(loo)
    loo(fit_rl)
```

```
Computed from 32000 by 60 log-likelihood matrix

         Estimate    SE
elpd_loo   -879.4 104.2
p_loo        62.8   4.2
looic      1758.8 208.5
------
Monte Carlo SE of elpd_loo is NA.
```

Summary

- 1. How to write a stan code?
- 2. How to write an R code?
- 3. How to diagnose the fitting result?
- 4. How to analyze the result?

Take them all

- We introduce concepts of computational modeling and why we need to learn about it in psychology and cognitive sciences.
- We introduce standard approaches (ten simple rules) on conducting a computational modeling research.
- We briefly go through how Stan works with MCMC.
- We dived into the codes to realize from the very beginning of simulation to analyze model fitting results.

- How to create/think up a good model?
  - Sorry, I don't know either…

- How to learn about computational modeling?
  - Knowledge perspective:
    - Math (linear algebra, theory of probability, Calculus)
    - Programming languages (R, Matlab, Python, Julia, C++, Java……)
    - Algorithms (basics about machine learning)
  - Practice perspective:
    - Learn by demanding, doing and achieving!

Recommended resources

- Summer schools:
  - Neuromatch Academy Summer School (Annually)
  - Computational Psychiatry Course @ Zürich, CH (annual)
  - London Computational Psychiatry Course @ London, UK (annual)
  - Brains, Minds & Machines Summer Course @ MIT, US (annual)
  - And so on (keep searching!)

Recommended resources

- Online courses:
  - Machine learning (@Coursera- Andrew Ng)
  - Bayesian Modeling (@Bilibili- Lei Zhang)
  - Reinforcement learning model basics (myself)
  - Reinforcement learning (@coursera)

Recommended resources

- Books:
  - R语言实战
  - Statistical Rethinking
  - 认知和行为的计算建模

Acknowledgement

Thank Dr. Zhou for his supervision and my lab members for their support!

Special thanks to:

- Wenqian Lin (RA at Beijing Normal University)
- Shen Xu (PhD student at Peking University)

For their advices on the tutorial.

Thank you for your listening!