Machine Learning and NeuroEngineering

机器学习与神经工程

Lecture 4 – Basic Parameter Estimation Techniques 1

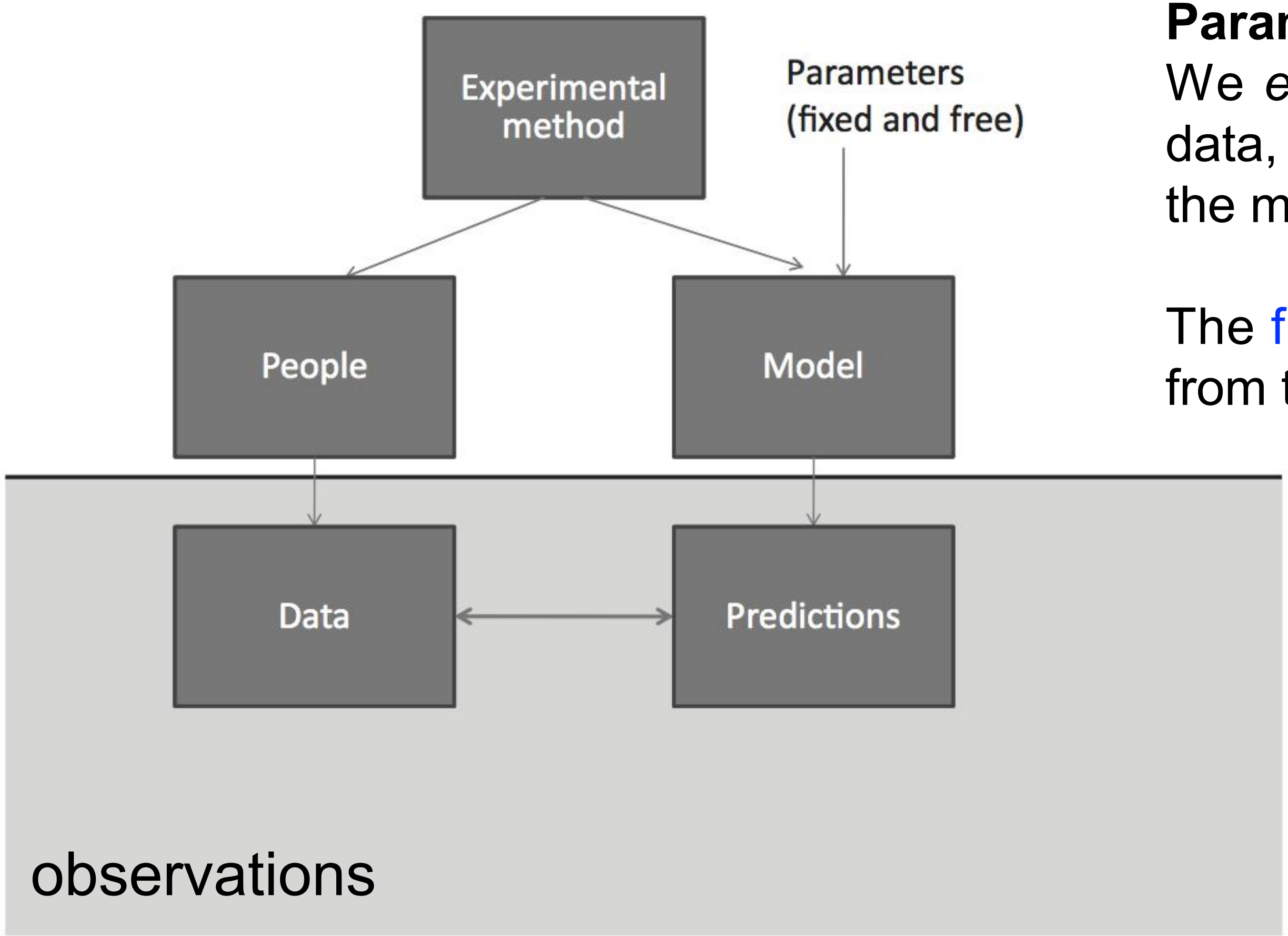
Quanying Liu （刘泉影）

SUSTech, BME department

Email: liuqy@sustech.edu.cn

BMEB212, Spring term, 2021

Connecting Model and Data



Parameters: free & fixed
We *estimate* the free parameters from the data, by finding those values that maximally fit the model's predictions with the data.

The fixed parameters, that are not estimated from the data, are invariant across datasets.

How to estimate free parameters?

Analytical solution
Grid search
***optim* function in R**

Lecture 4

- Linear regression
- Discrepancy Function
  - Continuous data: Root Mean Squared Deviation (RMSD)
  - Discrete data: Chi-Squared (χ^2)
- Least-Squares Estimation
- Parameter Estimation Techniques
  - Grid search
  - Simplex
  - Simulated Annealing
- Variability in Parameter Estimates
  - Bootstrapping

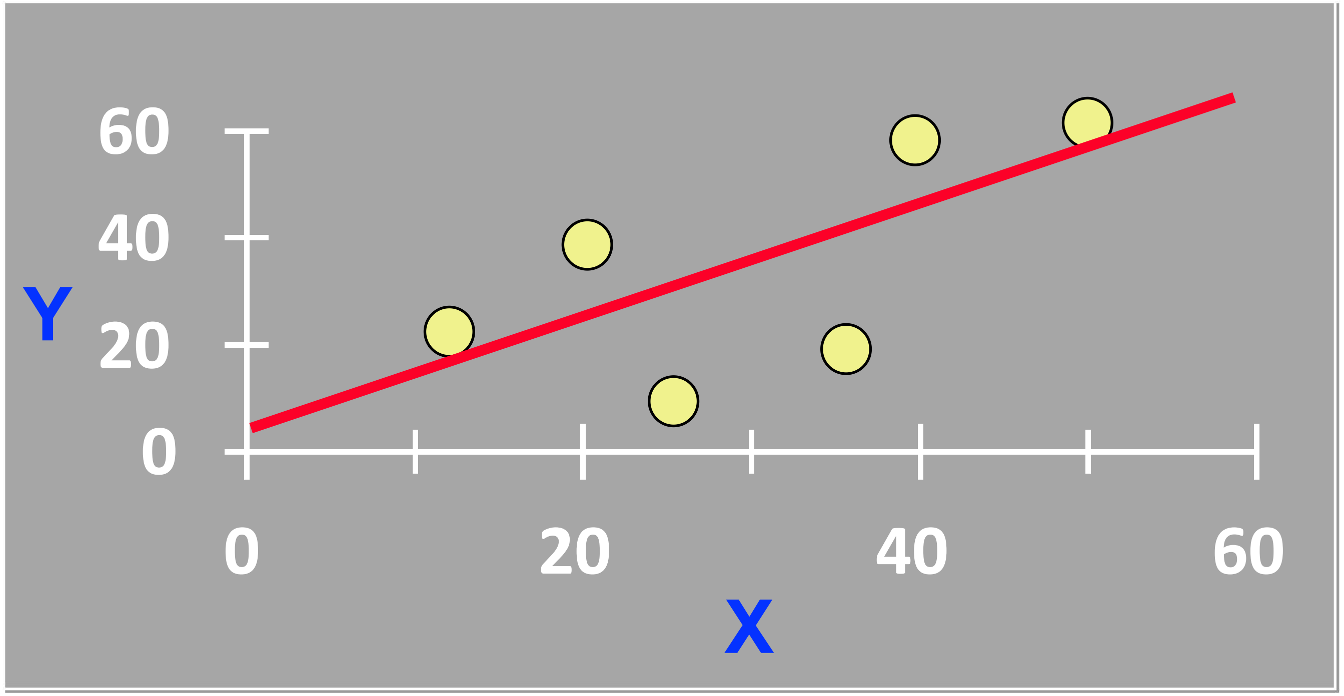
Linear regression

$$y = b_0 + b_1 x + e$$

A model with *two* unknown parameters

b_0 - intercept

b_1 - slop



Find the best parameter values,

to minimize the discrepancy between predictions and data.

First, we need define a **discrepancy function**.
It is also called cost function, objective function, loss function, error function...

linear regression with R

```
#define parameters to generate data
nDataPts  <- 20
rho        <- 0.8
intercept  <- 0.0

#generate synthetic data
data <- matrix(0,nDataPts,2)
data[ ,2] <- rnorm(nDataPts)     # x, data, independent variable
data[ ,1] <- rnorm(nDataPts)*sqrt(1.0-rho^2) + data[ ,2]*rho + intercept  # y, output

# default regression analysis

lm1 = lm(data[,1] ~ data[,2])     # lm(y ~ x)

summary(lm1)
```

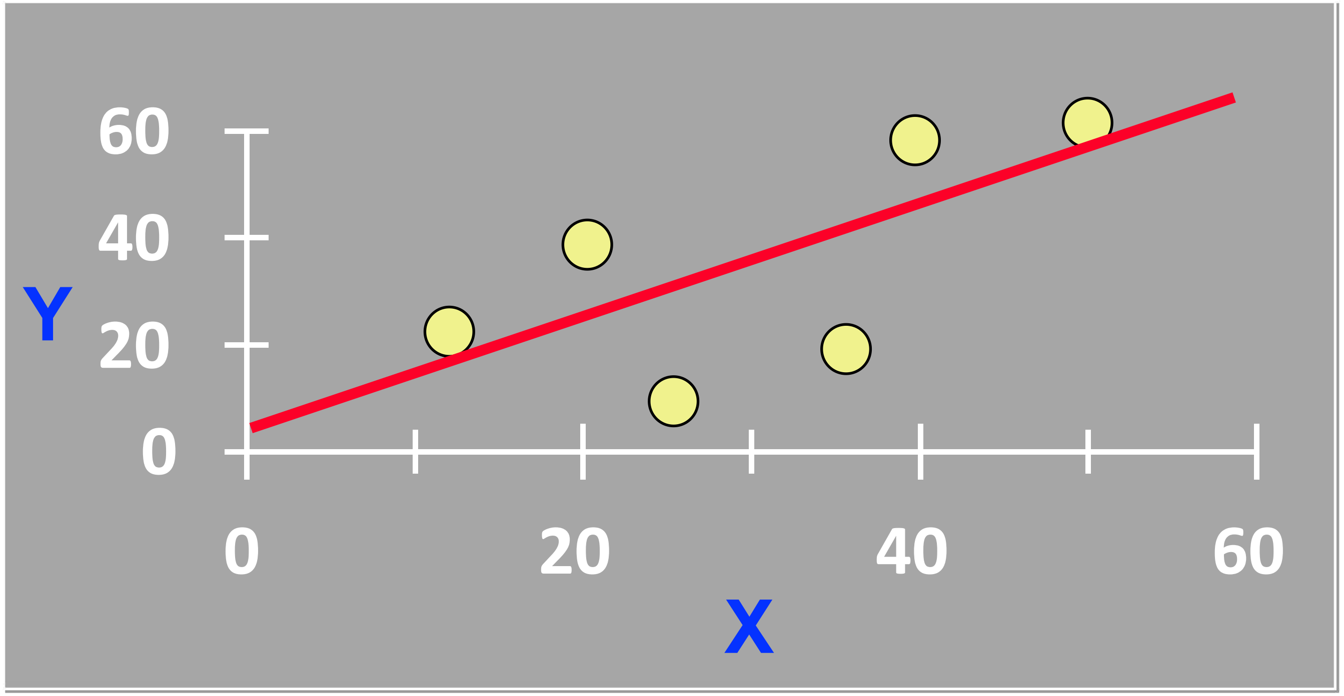
Discrepancy Function - RMSD

Continuous data:

Root Mean Squared Deviation (RMSD)

$y = b_0 + b_1 x + e$

$$RMSD = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}}$$



y_i is the ith real data

$\hat{y}_i$ is the ith prediction from model

n is the number of data points.

Prediction: $\hat{y}_i = b_o + b_1 x$

Error: $\varepsilon_i = y_i - \hat{y}_i$

Squared error: $\varepsilon_i{}^2 = (y_i - \hat{y}_i)^2$

"least-squares" 最小二乘法

Discrepancy Function - χ^2

Discrete data: Chi-Squared (χ^2)

$$\chi^2 = \sum_{j=1}^{J} \frac{(O_j - Np_j)^2}{Np_j}$$

Note: Noise can be amplified by a factor N.

Case1: $p_j = 0.9$, for $\mathrm{N} = 10$, $O_j = 9$

Case2: $p_j = 0.9$, for $\mathrm{N} = 100$, $O_j = 90$

J refers to the number of response categories,

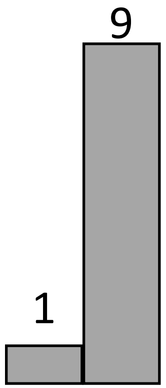
O_j to the number of observed responses within each category j.

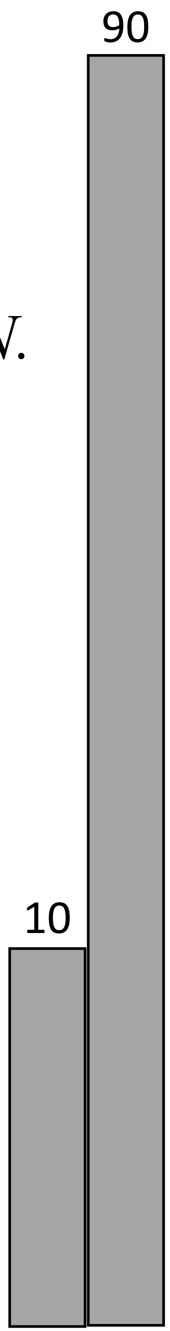
N refers to the total number of observed responses

the sum of all O_j is N

p_j is the probabilities of category j predicted by model.

Please calculate the χ^2 error





Least-square estimation – analytical solution

Squared error: $\varepsilon_i{}^2 = (y_i - \hat{y}_i)^2$

$$y = \beta_0 + \beta_1 x$$

$$\sum_{i=1}^{n} \varepsilon_i^2 = \sum_{i=1}^{n} \left(y_i - \beta_0 - \beta_1 x_i\right)^2$$

Least Squares (L-S): Minimize squared error

Derivation of Parameters = 0

$$0 = \frac{\partial \sum \varepsilon_i^2}{\partial \beta_0} = \frac{\partial \sum \left(y_i - \beta_0 - \beta_1 x_i\right)^2}{\partial \beta_0}$$

$$= -2\left(n\bar{y} - n\beta_0 - n\beta_1 \bar{x}\right)$$

$$0 = \frac{\partial \sum \varepsilon_i^2}{\partial \beta_1} = \frac{\partial \sum \left(y_i - \beta_0 - \beta_1 x_i\right)^2}{\partial \beta_1}$$

$$= -2\sum x_i \left(y_i - \beta_0 - \beta_1 x_i\right)$$

$$= -2\sum x_i \left(y_i - \bar{y} + \beta_1 \bar{x} - \beta_1 x_i\right)$$

$$\beta_1 \sum x_i (x_i - \bar{x}) = \sum x_i (y_i - \bar{y})$$

$$\beta_1 \sum (x_i - \bar{x})(x_i - \bar{x}) = \sum (x_i - \bar{x})(y_i - \bar{y})$$

$$\boxed{\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}}$$

Analytical solution

$$\boxed{\hat{\beta}_1 = \frac{SS_{xy}}{SS_{xx}} = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2}}$$

Parameter Estimation – grid search

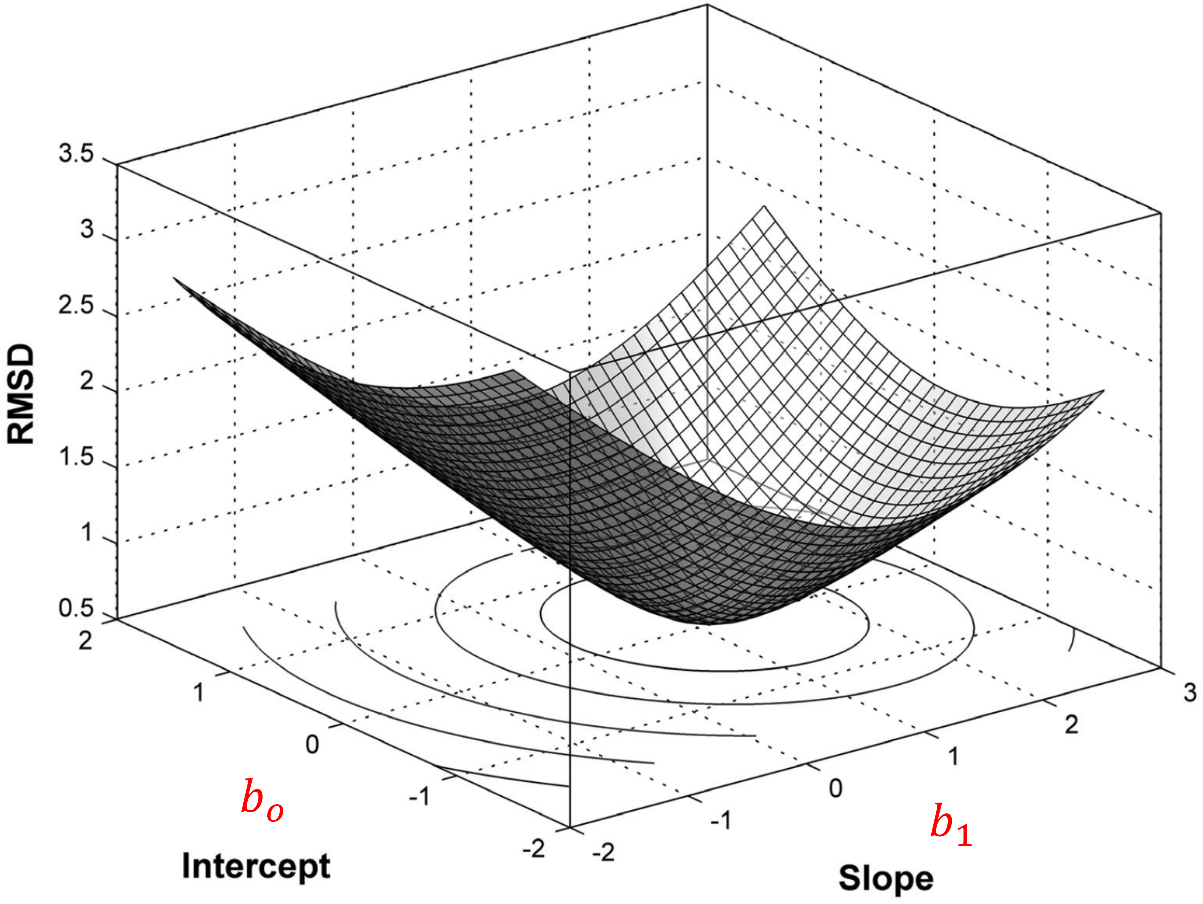
Linear regression

Model: $y = b_0 + b_1 x$

training data: (x_i, y_i) with i =1,2,…,n

1. Generate a grid for (b_0, b_1)
2. Calculate $\hat{y}_i = b_o + b_1 x$
3. Calculate $RMSD = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}}$

An “error surface”



pros: Simple, straightforward, easy to use.

cons: Exponential increase with the number of parameters

Parameter Estimation - optim function in R

optim is a general-purpose optimization function.

Input:

1. Starting values of the model parameters,
2. A *function* to be minimized   # such as minimize a discrepancy function
3. The method to be used (optional):

   method = c("Nelder-Mead", "BFGS", "CG", "L-BFGS-B", "SANN", "Brent")

Output:

the best-fitting parameter estimates (a list structure)

discrepancy

Parameter Estimation - optim function in R

```
#plot data and current predictions
getregpred <- function(parms, data) {
  getregpred <- parms["b0"] + parms["b1"]*data[ ,2]        # prediction
}

#obtain current predictions and compute discrepancy
rmsd <- function(parms, data1) {
  preds <- getregpred(parms, data1)  # parms["b0"] + parms["b1"]*data[ ,2]
  rmsd  <- sqrt(sum((preds-data1[ ,1])^2)/length(preds))  # calculate RMSD
}

#assign starting values
startParms <- c(-1., .2)
names(startParms) <- c("b1", "b0")

#obtain parameter estimates
xout <- optim(startParms, rmsd, data1=data)
```

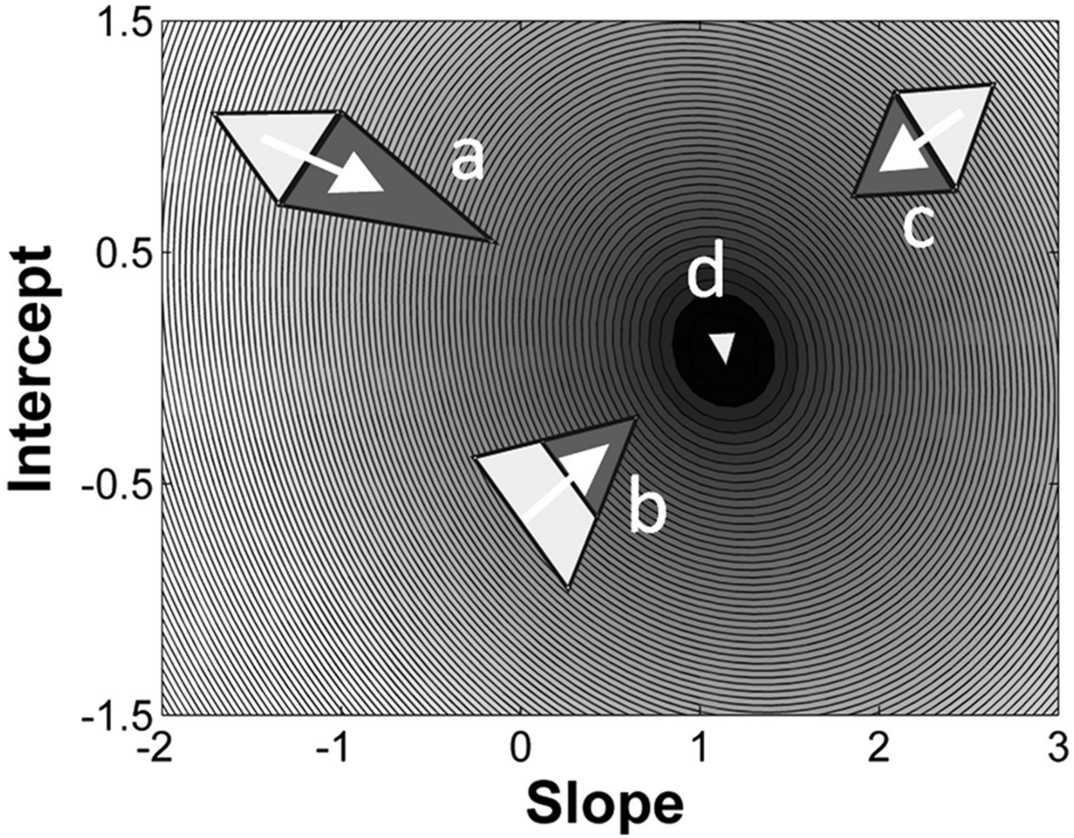
$$RMSD = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}}$$

Parameter Estimation - Simplex

A **simplex** is a geometrical figure with M+1 interconnected points in M dimensions.

e.g. Simplex for Linear regression: 3 points in 2 D.

2-D projection of the error surface



Algorithm:

Create a simplex at a location given by the starting values, and calculate the *discrepancy function* for each point of the simplex.

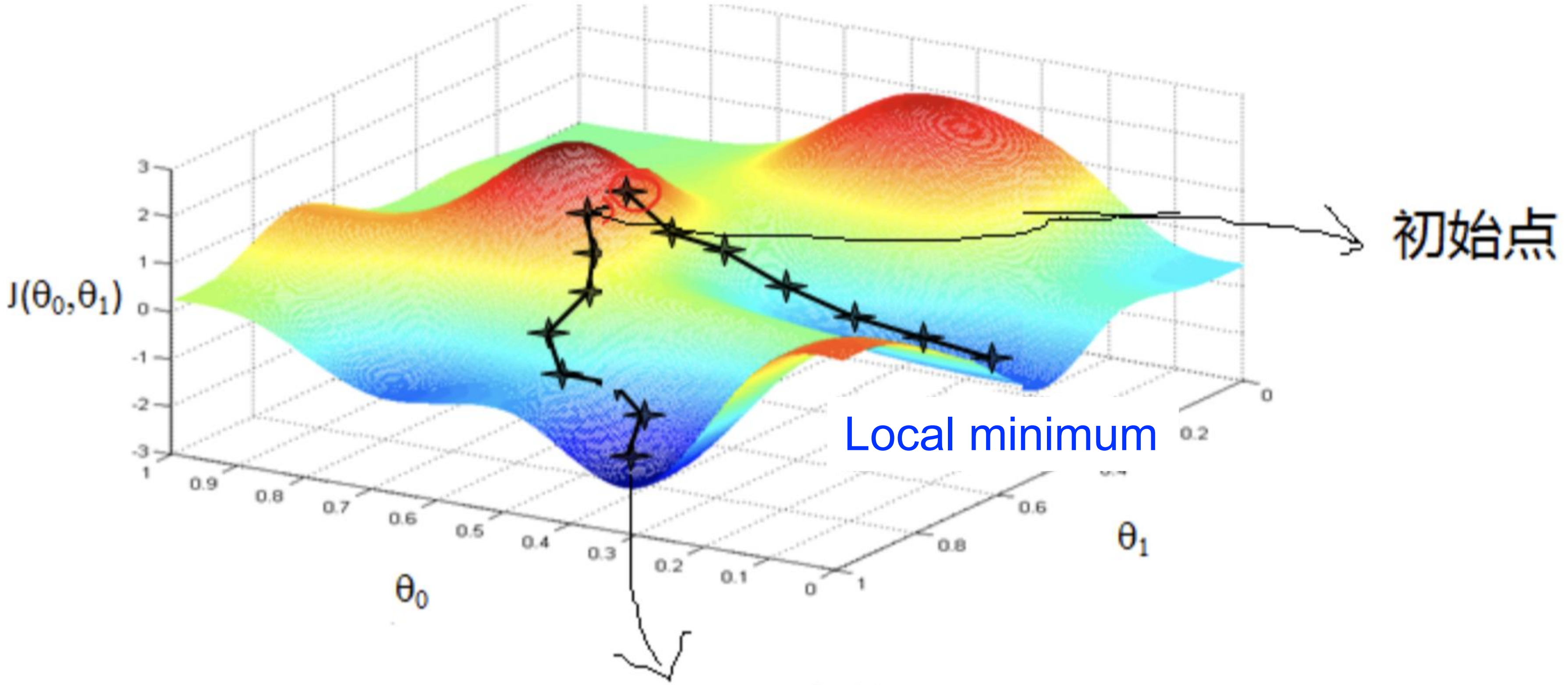
Then, the simplex move through the parameter space:

1) <u>be reflected/expanded</u>: the point with the greatest discrepancy (worst fit) is flipped to the opposite side;

2) <u>be contracted</u>: moving the point (or points) with the worst fit closer toward the center.

Until it **converges** at the best-fitting parameter values.

However, sometimes we are facing a **non-convex** error surface, especially for the complex models.



Global minimum 最小值

It is possible that Simplex descends into a *local* minimum rather than the ***global*** minimum.

We want a method which can “jump” out of local minima --> Simulated Annealing

Parameter Estimation - Simulated Annealing

Candidate update $\boldsymbol{\theta}_c^{(t+1)} = D(\boldsymbol{\theta}^{(t)})$

D is a "candidate function"

Difference of discrepancy value

$$\Delta f = f\left(\boldsymbol{\theta}_{\mathrm{c}}^{(t+1)}\right) - f\left(\boldsymbol{\theta}^{(t)}\right)$$

Stochastic decision

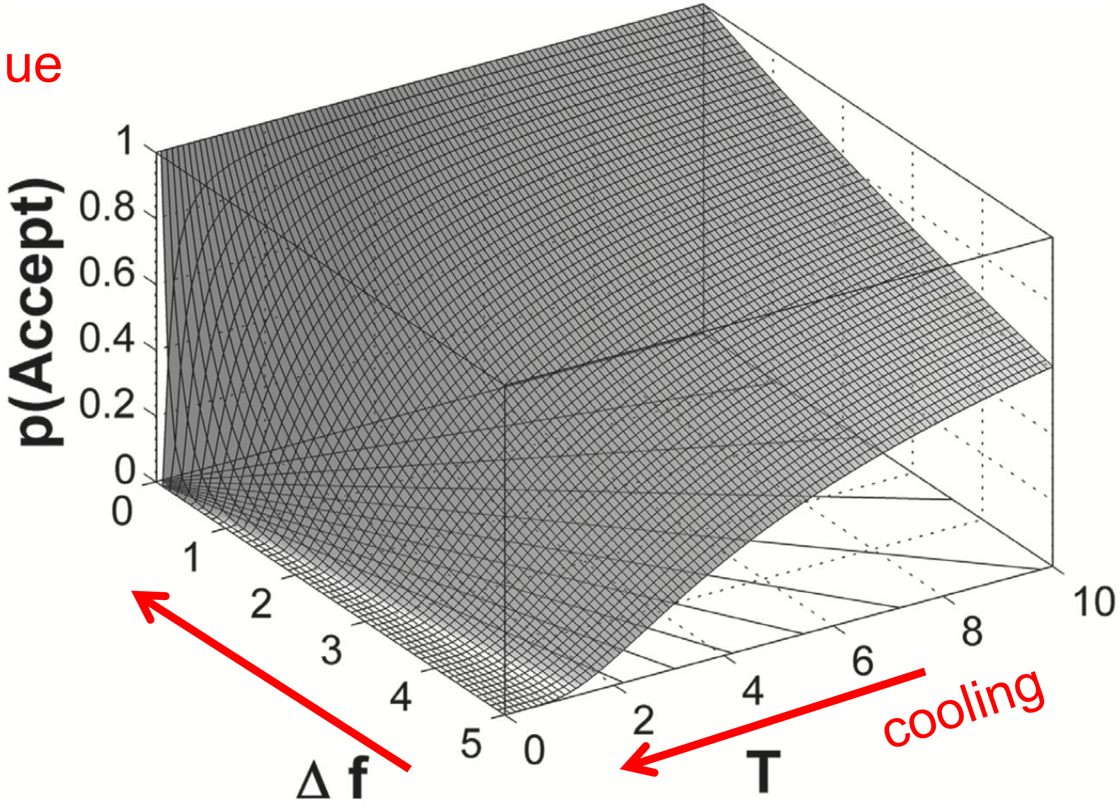
$$\boldsymbol{\theta}^{(t+1)} = \begin{cases} A(\boldsymbol{\theta}_c^{(t+1)}, \boldsymbol{\theta}^{(t)}, T^{(t)}) & \text{if } \Delta f > 0 \\ \boldsymbol{\theta}_c^{(t+1)} & \text{if } \Delta f \le 0 \end{cases}$$

Acceptance function

$$A(\boldsymbol{\theta}_c^{(t+1)}, \boldsymbol{\theta}^{(t)}, T^{(t)}) = \begin{cases} \boldsymbol{\theta}_c^{(t+1)} & \text{if } p < e^{-\Delta f / T^{(t)}} \\ \boldsymbol{\theta}^{(t)} & \text{otherwise,} \end{cases}$$

Cooling schedule $T^{(t)} = T_0\,\alpha^t$ $\quad T^{(t)} = T_0 - \eta\, t$

Interactions between Δf and T



Parameter Estimation - optim function in R

```
#plot data and current predictions
getregpred <- function(parms, data) {
  getregpred <- parms[“b0”] + parms[“b1”]*data[ ,2]          # prediction
}

#obtain current predictions and compute discrepancy
rmsd <- function(parms, data1) {
  preds <- getregpred(parms, data1)  # parms["b0"] + parms["b1"]*data[ ,2]
  rmsd  <- sqrt(sum((preds-data1[ ,1])^2)/length(preds))  # calculate RMSD
}

#assign starting values
startParms <- c(-1., .2)
names(startParms) <- c("b1", "b0")

#obtain parameter estimates
xout <- optim(startParms, rmsd, data1=data, method = "SANN")
```

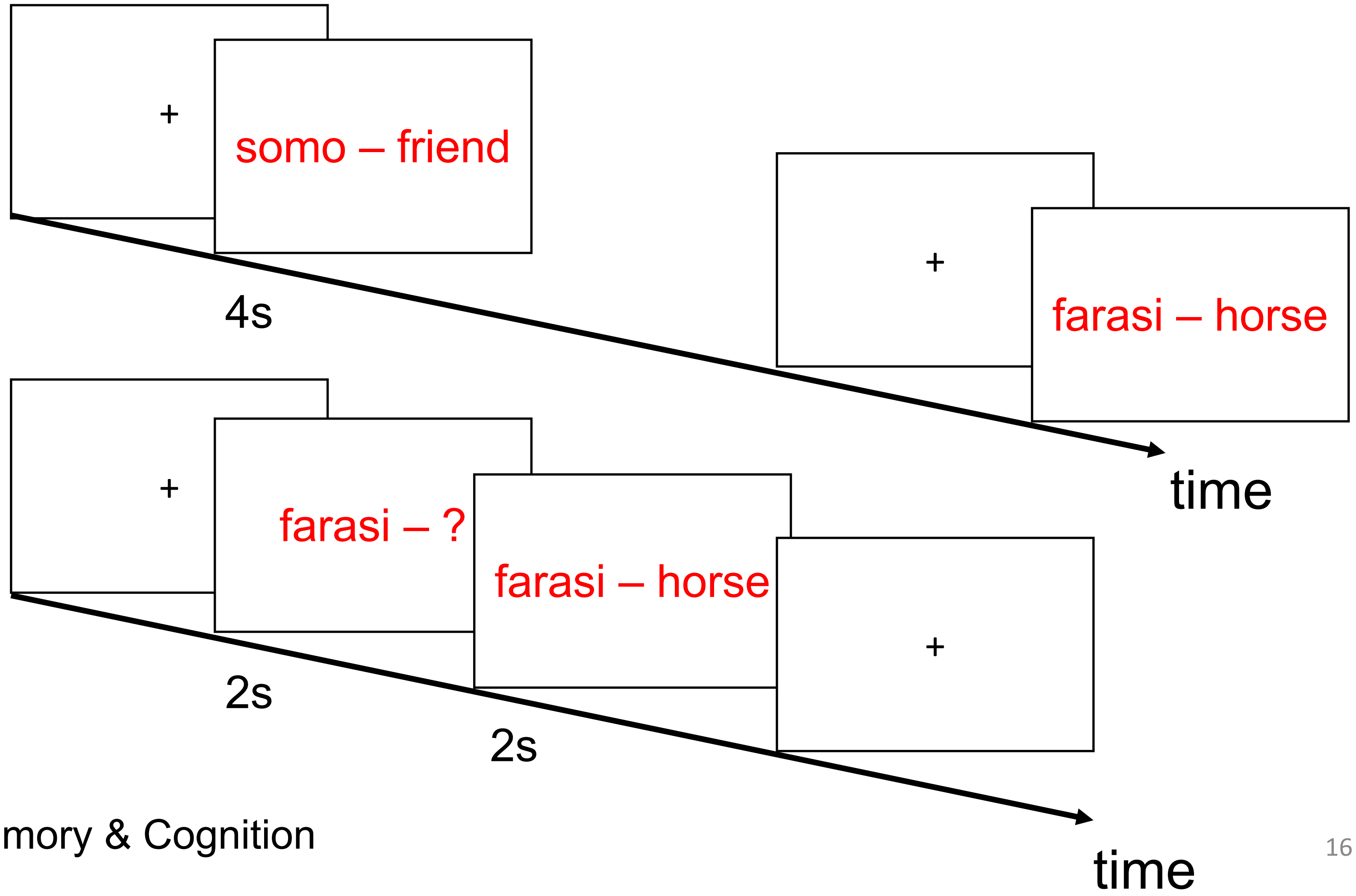
A memory task

You have to remember *60* Swahili–English word pairs.

Learning session

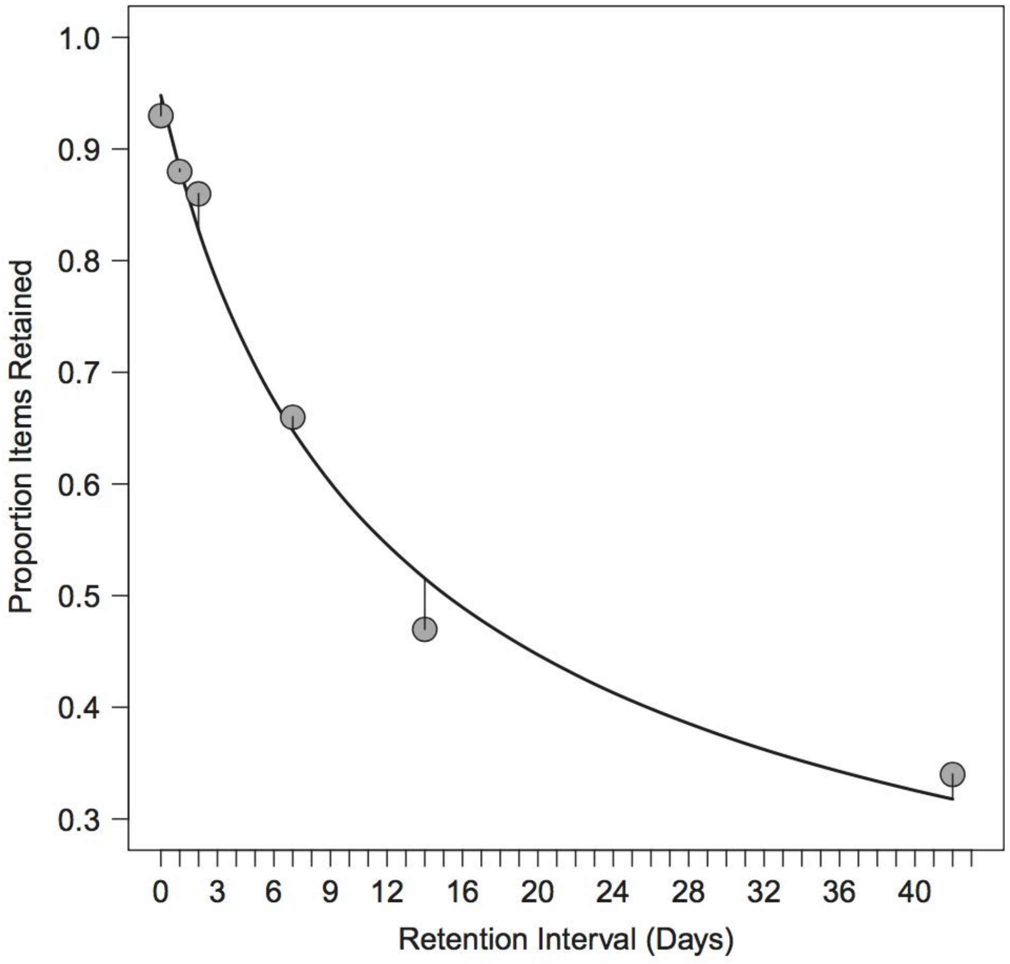
Test session

5 min, 1 day, 2 days,
7 days, 14 days, or 42 days



Carpenter et al. (2008), Memory & Cognition

A power model to characterize forgetting



We have behavioral data with 5 min, 1 day, 2 days, 7 days, 14 days, or 42 days of retention intervals.

```
rec <- c(.93,.88,.86,.66,.47,.34)
ri  <- c(.0035, 1, 2, 7, 14, 42)
```

A power model: $p = a(bt + 1)^{-c}$

Please use **optim** function to find the best-fitting a, b, c.

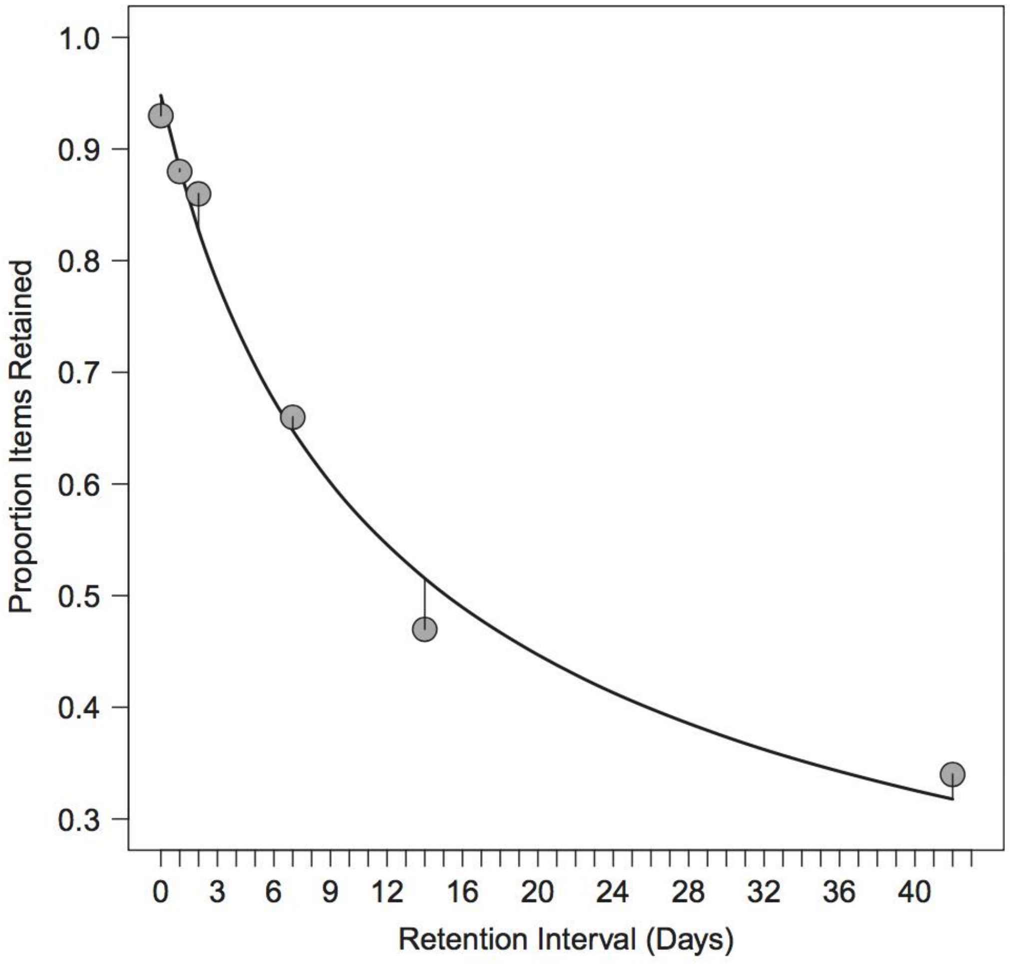
Carpenter et al. (2008), Memory & Cognition

```
#discrepancy function for power forgetting function
powdiscrep <- function (parms,rec,ri) {
  if (any(parms<0)||any(parms>1)) return(1e6)
  pow_pred <- parms["a"] *(parms["b"]*ri + 1)^(-parms["c"])    # Prediction
  return( sqrt( sum((pow_pred-rec)^2)/length(ri) ) )}          # RMSE

#Carpenter et al. (2008) Experiment 1
rec <- c(.93,.88,.86,.66,.47,.34)  # y: recall proportion
ri  <- c(.0035, 1, 2, 7, 14, 42)   # x: retention interval

#initialize starting values
sparms <- c(1,.05,.7)
names(sparms) <- c("a","b","c")
#obtain best-fitting estimates
pout <- optim(sparms, powdiscrep, rec=rec, ri=ri)
pow_pred <- pout$par["a"] *(pout$par["b"]*c(0:max(ri)) + 1)^(-pout$par["c"])
```

Models to characterize forgetting



Carpenter et al. (2008), Memory & Cognition

We have behavioral data with 5 min, 1 day, 2 days, 7 days, 14 days, or 42 days of retention intervals.

```
rec <- c(.93,.88,.86,.66,.47,.34)
ri  <- c(.0035, 1, 2, 7, 14, 42)
```

A power model: $p = a(bt + 1)^{-c}$

Please use **optim** function to find the best-fitting a, b, c.

$a = 0.95$
$b = 0.13$
$c = 0.58$

How about an exponential model?

$$p = a + be^{-ct}$$

```r
#discrepancy function for power forgetting function
ediscrep <- function (parms,rec,ri) {
  if (any(parms<0)||any(parms>1)) return(1e6)
  e_pred <- parms["a"] + parms["b"]*exp( -parms["c"]*ri )  # Prediction
  return(sqrt( sum((e_pred-rec)^2)/length(ri) ))            # RMSE
}

#Carpenter et al. (2008) Experiment 1
rec <- c(.93,.88,.86,.66,.47,.34)  # y: recall proportion
ri  <- c(.0035, 1, 2, 7, 14, 42)   # x: retention interval

#initialize starting values
sparms <- c(1,.05,.7)
names(sparms) <- c("a","b","c")
#obtain best-fitting estimates
pout <- optim(sparms, ediscrep, rec=rec, ri=ri)
e_pred <- pout$par["a"] + pout$par["b"]*exp( -pout$par["c"]*c(0:(max(ri))) )
```

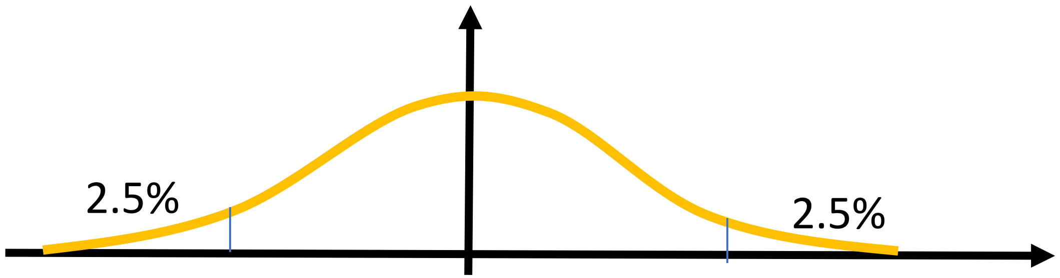
Variability in Parameter Estimates

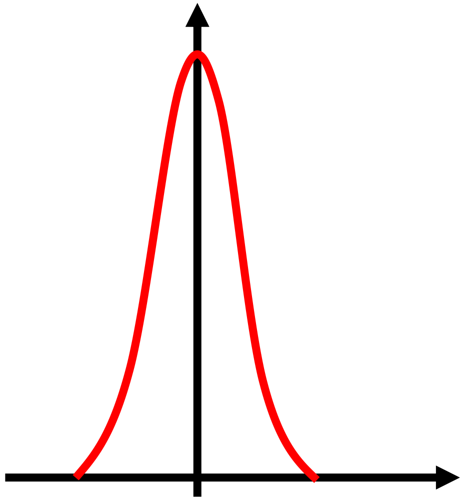
The *parameter estimation techniques* have shown so far provide a *best estimate* of the parameter values. But they are point estimates.

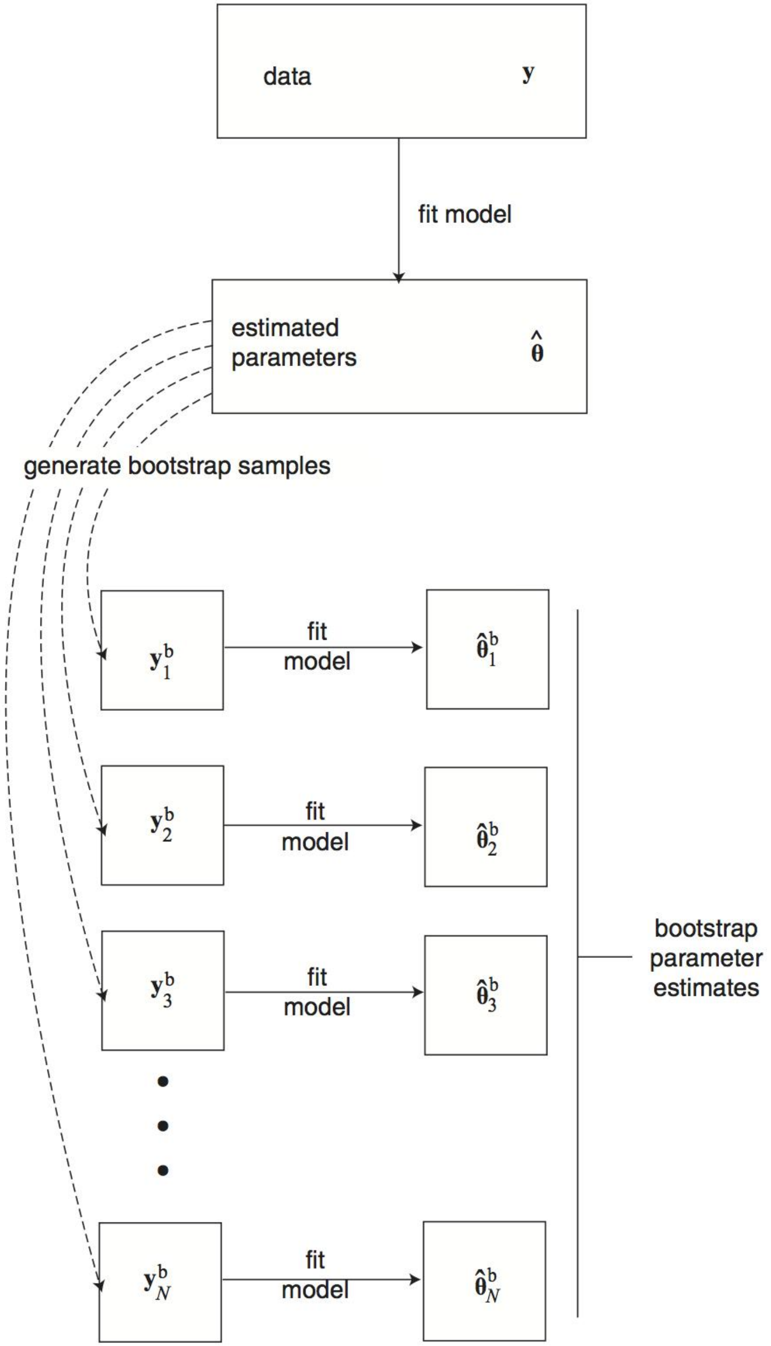
They do **not** carry any information about the **variability** in these estimates.

They **lack** of any known statistical properties.

We do **not** know the Confidence Interval.







Bootstrapping

Procedure:

We estimate parameters by fitting the model.

We generate T samples by running T simulations from the model using the estimated parameters.

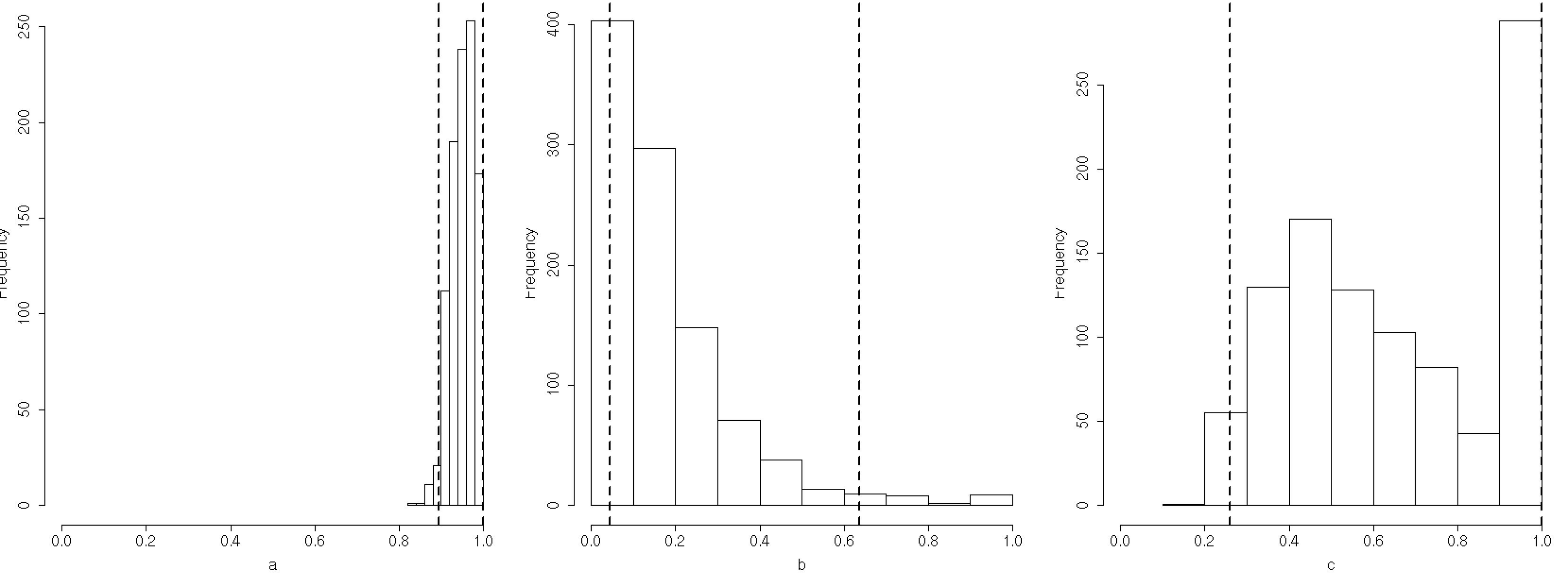
(Each generated sample should contain N data points, where N is the number of data points in the original sample.)

We then fit the model to each of the T generated samples.

The variability across the T samples in the parameter estimates then gives us some idea about the variability in the parameters.

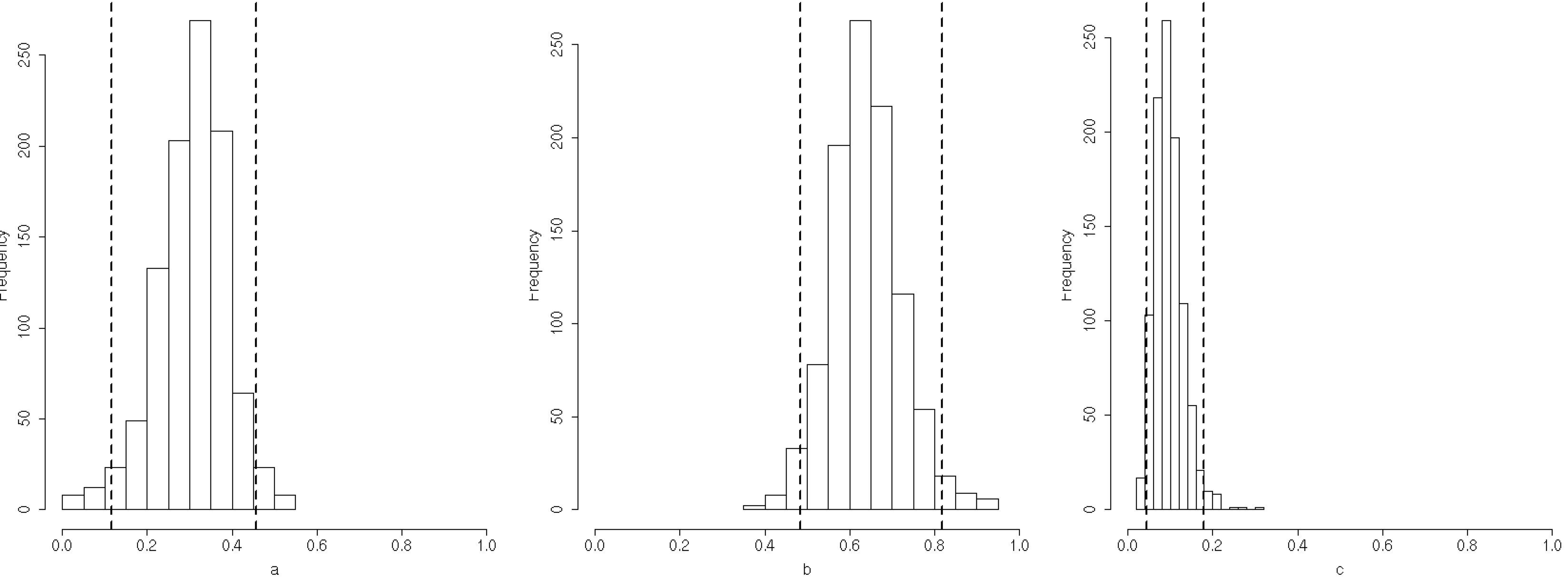
Bootstrapping results – power model

$$p = a(bt+1)^{-c}$$



Bootstrapping results – exponential model

$$p = a + be^{-ct}$$



Summary of Lecture 4

- Linear regression
- Discrepancy Function
  - Continuous data: Root Mean Squared Deviation (RMSD)
  - Discrete data: Chi-Squared (χ^2)
- Least-Squares Estimation （最小二乘法）
- Parameter Estimation Techniques
  - Grid search （网格搜索法）
  - Simplex （单纯形法）
  - Simulated Annealing （模拟退火算法）
- Variability in Parameter Estimates
  - Bootstrapping （自助法）

Recommended materials

Textbook

- Computational Modeling of Cognition and Behavior, Chapter 3

Must read.

Research Paper

- Carpenter et al. (2008), The effects of tests on learning and forgetting, Memory & Cognition

Not obliged. For fun.

A reminder

In order to get 10% credits, you have to form your team for course project by **next week**.

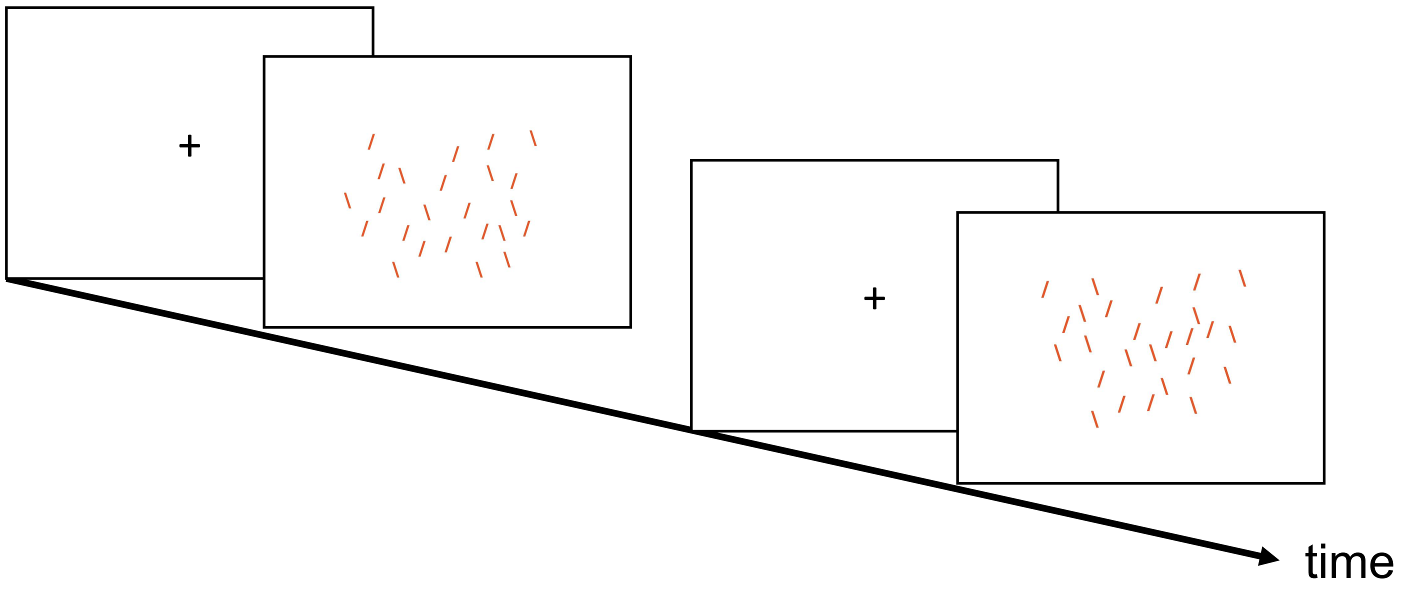
Tip: An example on YouTube
https://www.youtube.com/watch?v=i6xMBig-pP4

Reminder - Homework

Some code is available at github

Implement the task with Python （可以用PyGame）

DDL: March 8, 2021



Requirements

1. **60 trials in total**
2. The **number** of \ and / is sampled from a uniform distribution [20, 40] and cannot be equal.
3. Randomize the **locations** of \ and /
4. **Record the following information**

1) Subject ID	5) The number of \ in a trial
2) Trial ID	6) The number of / in a trial
3) Time: onset of stimulus	7) The response: left or right
4) Time: response is made	8) Response time: time in 4) – time in 3)

输出到一个csv文件
一行就是一个trial的信息
每一行有8列