Likelihoods and Maximum Likelihood Estimation (MLE)

Casimir Ludwig
c.ludwig@bristol.ac.uk

Parameter estimation

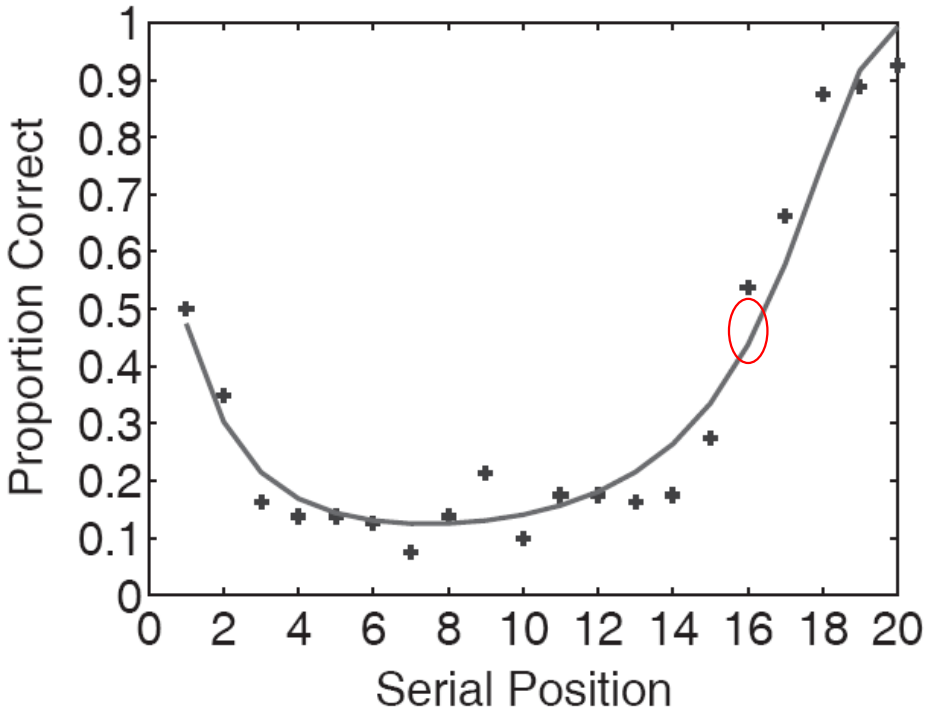
- Varying parameters in the model -> varying predictions
- For a given set of parameters, compute error relative to data
- Find parameters that give a good fit, e.g. grid search or some automated optimisation routine (optim / optimize)
- What “error metric”?
  - RMSE, χ^2, …
- No distributional assumptions, so good for an initial stab
- But no principled basis for model comparison, confidence/credible intervals

Enter likelihoods…

- Statistically desirable properties (sufficiency, consistency, efficiency)
- Basis for principled model comparison (e.g. AIC, BIC, Bayes…)
- Likelihood is a core ingredient for Bayes (posterior distributions of parameters, Bayes Factors)

Linking model predictions to data

- Example: predicted probability of 0.5 for serial position 16
- If you flipped a (fair) coin 10 times, would you expect always 5/10 heads?
- If we do this N times, we would expect slightly different results on every run



Sampling variability

- The model predicts the “true” value
- Real data (either from one individual or from a group of individual) is a noisy sample
- Each sample will be slightly different (as in our N runs of 10 coin flips)
- Allowing for sampling variability help us connect model predictions to real data

Sampling variability

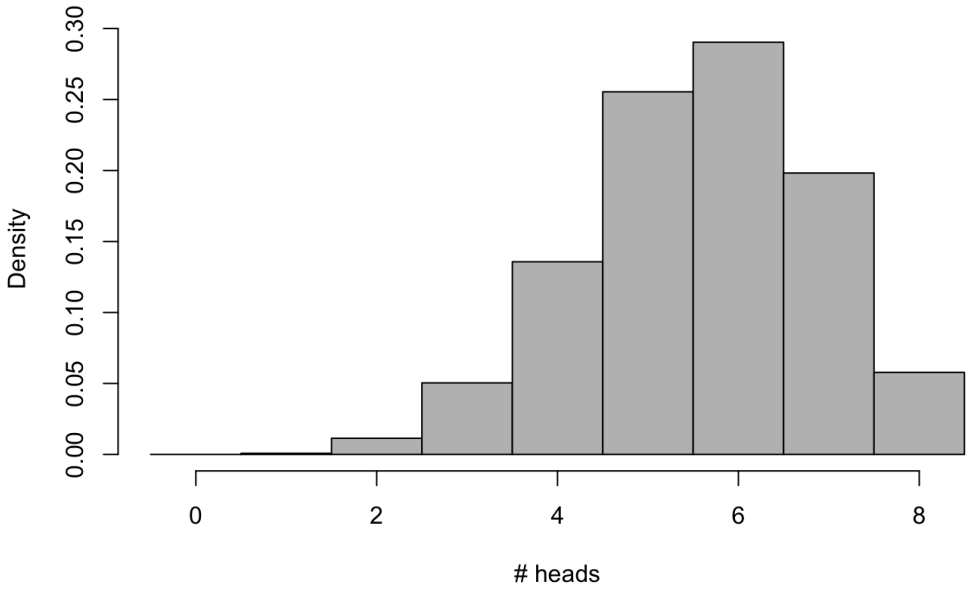
- SIMPLE is a deterministic model: for a given set of parameters, it will always predict the same probability correct at a given serial position
- The SIMPLE prediction is best thought of as the observed probability if you ran an infinite number of trials
- Unless you are a psychophysicist, $N << \infty$
- For a particular set of 10 trials, the subject may get 3, 5, or even 10 items correct

Play time…

- Monte carlo simulation of flipping of a weighted coin
  - Probability of heads = 0.7
  - 1000 games
  - For each game, simulate 8 coin tosses and record the observed number of heads
    - A vector of length 1000, each element is the number of heads for a single game
  - Plot a histogram of the number of heads
- Hints
  - runif()<p_heads
  - Rbinom

A probability distribution

- Probability of heads on a single coin toss= 0.7
- Each bar: probability of seeing exactly k heads from 8 coin tosses
- p(*k* | *p_heads*, *N*) where k is number of heads
- `rbinom(k, 8, p_heads)`



Binomial distribution

$$p(k|p_{heads}, N) = \binom{N}{k} {p_{heads}}^{k} (1 - p_{heads})^{N-k}$$

- Probability of *k* outcomes actually happening (e.g., getting 5 heads)
  - given *N* total observations (e.g., 8 coin flips)
  - and p_{heads} probability of the event happening on each observation (throwing a head)
- Each *k* has a probability between 0 and 1 (inclusive)

Play time…

- Plot a predicted distribution
- Plot predicted probability of number of heads (0-10) from a coin with p_{heads}=0.5 and with 10 coin tosses in total
- Use the `dbinom` function in R
  - x: the values on the x axis (different possible number of heads)
  - size: total number of tosses
  - prob: the probability of a head
  - Use type="h" when plotting
- Advanced: simulate for p_{heads}=0.7 (N tosses = 8), and plot against the numerical simulation results from earlier

Binomial distribution

- The binomial is a probability distribution (or “probability mass function” given that the outcomes are discrete events)
- Binomial is probability of various **discrete** outcomes given
  - Probability of occurrence on each observation
    - e.g., getting a head, correctly recalling an item from the study list, correctly discriminating motion direction, a child passing the Sally-Anne task, …
  - Number of observations
    - Fixed by the experimenter
- Binomial distribution as a “data model”: connects model prediction (e.g. predicted probability correct) with empirical observations

How does this work for a cognitive model (SIMPLE)?

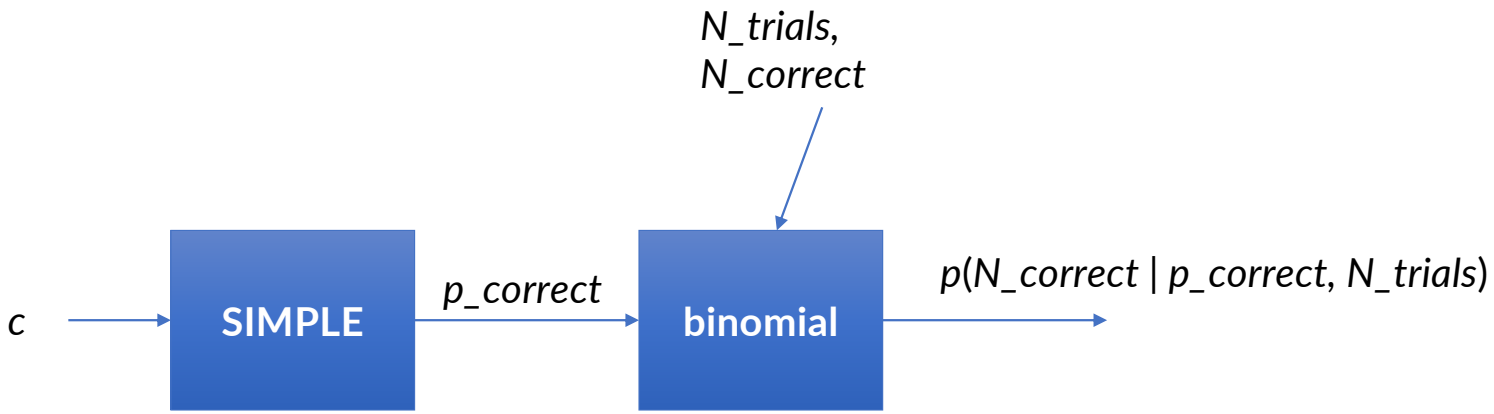
$$p(k|p_{heads}, N) = \binom{N}{k} {p_{heads}}^{k} (1 - p_{heads})^{N-k}$$

Number of trials: fixed by the experimenter

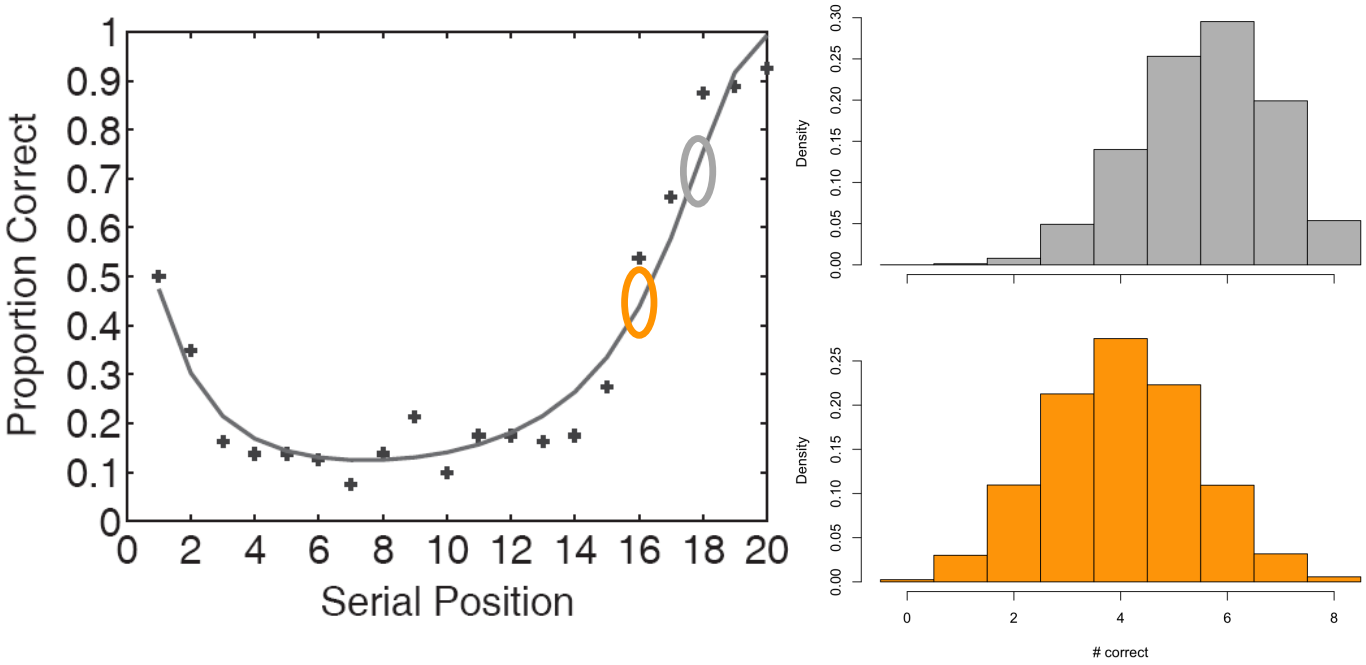
The predicted probability obtained from SIMPLE (e.g., 0.7 if we were looking at serial position 17)

Number of items correct

How does this work for a cognitive model (SIMPLE)?



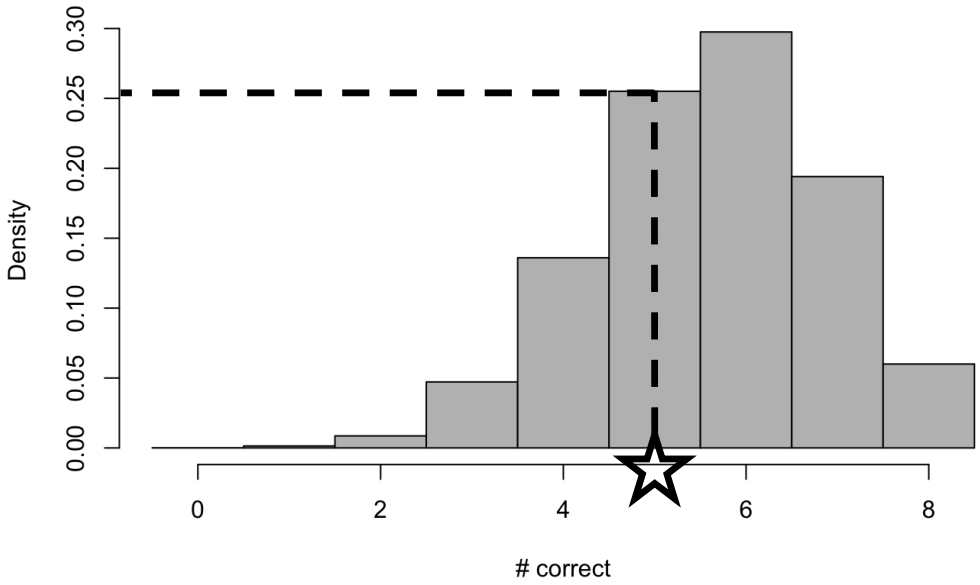
How does this work for a cognitive model (SIMPLE)?



- Predicted probability of getting *k* items correct (for various *k*) from SIMPLE with binomial model
- **For each serial position** we have a binomial probability distribution, where *p_correct* is given by SIMPLE and *N* is fixed

c.ludwig@bristol.ac.uk

How does this work for a cognitive model (SIMPLE)



- Predicted probability from SIMPLE: *p_correct* = 0.7
- Actual data: #correct = 5
- Total number of trials = 8
- Binomial probability of getting 5 trials correct when the "true" probability correct is 0.7: ~0.25

Aside

- Some models inherently generate probabilistic predictions (e.g. models of RT distributions, such as ex-Gaussian or plain diffusion model without bells and whistles)
- Other models do not and so we need to add a "data model" to allow the model to accommodate sampling variability: the observed (sampled) data will never line up exactly with the model, even if the model were truly correct! (Note: all models are wrong! 🤔🤫)
  - Binomial sampling
  - Softmax or Luce's choice rule for binary observations (e.g. in RL models, value-based decision making models, GCM)

- Use `dbinom` function to calculate p(data | *p_correct*) for the SIMPLE example
  - Data: 5 items correct
  - *pcorrect*: 0.7
  - N trials = 8
  - p(getting 5 items correct given the predicted probability of getting an item correct is 0.7, and that there are 8 trials in total)

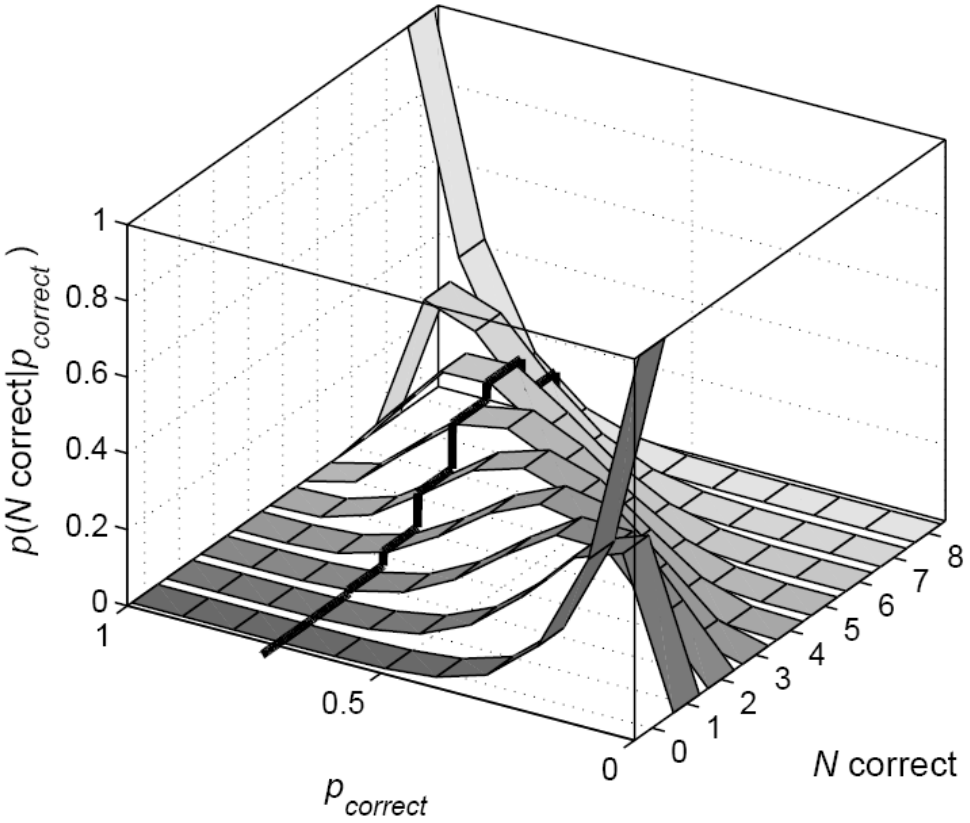
Likelihoods

- What we have just done is compute the probability of a single observation (*N_correct* = 5) under a model with a specific parameter (*c* -> *p_correct* = 0.7)
- We'll extend this to multiple data points…
- Note that the data are fixed
- The probability of this one observation changes as a function of the model parameter(s), *c*
- **Maximum Likelihood Estimation**: given some observed data, find the model parameters that maximise the likelihood of these data

This may sound easy…

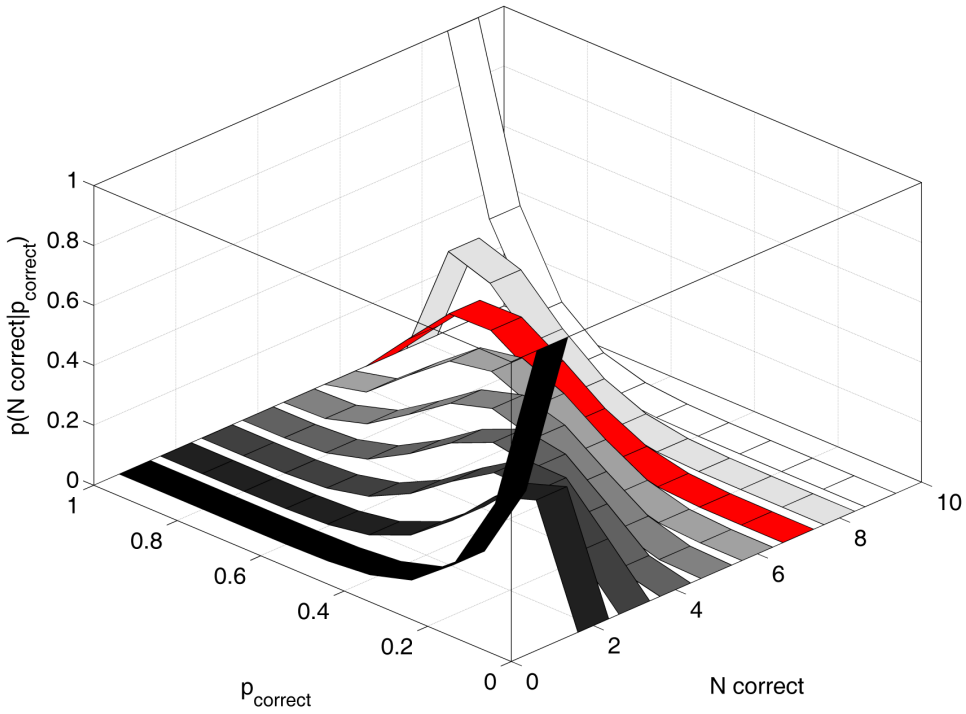
- p(data | parameters)
  - Probability density / mass function; for a given parameter *c* assign probabilities to different possible outcomes (i.e. different values of N_correct)
- Likelihood: p(data | parameters), but recognising that data are fixed and parameters change
  - *L*(parameters | data)
  - Give it a different name to reflect the fact that likelihood is a function of the parameters
- Note that this is not p(parameters| data)
  - Covered in Bayesian statistics

- Strips and the dark line are all binomial distributions
- Dark line is probability mass function: p(data|parameters) – `dbinom(x, N, p)`
- Strips are likelihood functions: L(parameters|data) – `dbinom(x_obs, N, p)`
- Remember, *p_correct* is a predicted probability from the cognitive model (not core model parameter itself)



c.ludwig@bristol.ac.uk

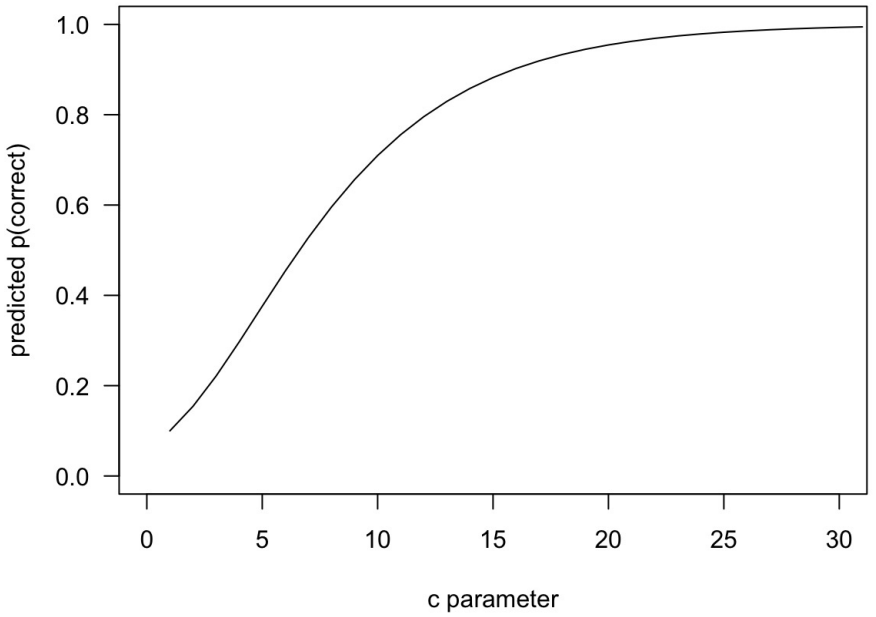
- Remember we have observed a single data point, e.g. *N_correct* = 7
- The likelihood of observing this datapoint, is given by the red strip
- As a function of the model predicted probability correct (i.e. as a function of *c*)



c.ludwig@bristol.ac.uk

A likelihood surface

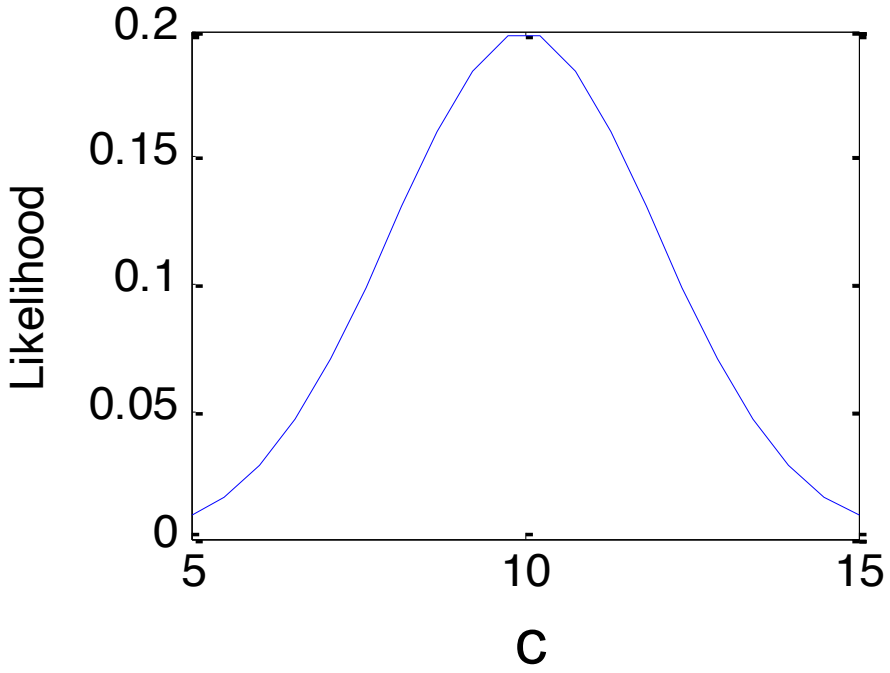
- But we want to talk about the "real" model probability in SIMPLE (the *c* parameter)
- Likelihood function across *c* in SIMPLE: *c* varies, the data are fixed
- E.g. for a single observation *N_correct* = 7 out of *N_total* = 10, the likelihood will be maximal for whatever value of *c* generates a predicted *p_correct* = 0.7



c.ludwig@bristol.ac.uk

A likelihood surface

- But we want to talk about the "real" model probability in SIMPLE (the *c* parameter)
- Likelihood function across *c* in SIMPLE: *c* varies, the data are fixed
- E.g. for a single observation *N_correct* = 7 out of *N_total* = 10, the likelihood will be maximal for whatever value of *c* generates a predicted *p_correct* = 0.7



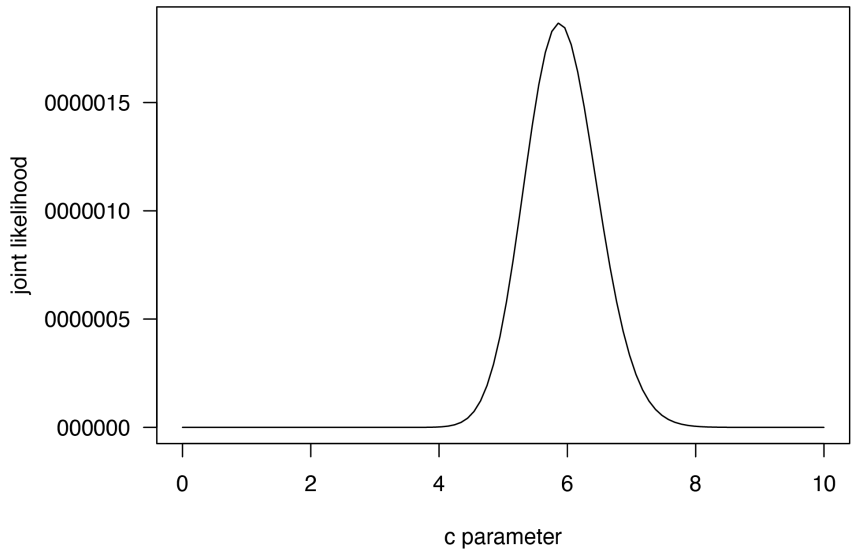
A likelihood surface – multiple data points

- However, we have multiple data points (i.e. observed *N_correct*)
- In SIMPLE, a single parameter *c* generates predicted probabilities for **all** serial positions probed in the experiment
- So *c* -> *p_correct* (serial position) -> likelihood of observing *N_correct* (serial position)
- Assumption: responses in different trials are independent 🤔
- Overall **joint likelihood** of the model parameters, given multiple observations is the product of the likelihoods of M individual datapoints: $\mathcal{L}(c|N_{correct}) = \prod_{i=1}^{M} \mathcal{L}(c|N_{correct,i})$

Likelihood

```
N_correct <- c(8, 7, 6, 6, 8, 10, 10, 14, 14, 19)
N_total <- 20
cParm <- seq(0,10, length.out=100)
likSIMPLE <- computeJointLikelihood(cParm, N_correct, Ntotal)
```

Joint likelihood as a function of *c* for multiple observations



Maximum Likelihood Estimation (MLE)

c.ludwig@bristol.ac.uk

How does this help us estimate parameters?

- We want to maximize the likelihood
  - Find the peak of the likelihood surface
  - 1D for the SIMPLE model so far (just a single parameter *c*)
  - For more realistic models, multiple dimensions
- Or minimize the negative likelihood
  - Remember, optim() does function minimization
- Can do this using methods from earlier today (optim / optimize)
- But first…

Joint likelihood

- Joint likelihood: product of probability mass/density of individual observations
- Imagine you have many observations: product of many small numbers may get you to the limit of the numerical precision of your machine quite quickly
- Convention is to work with natural logarithm of the likelihoods:

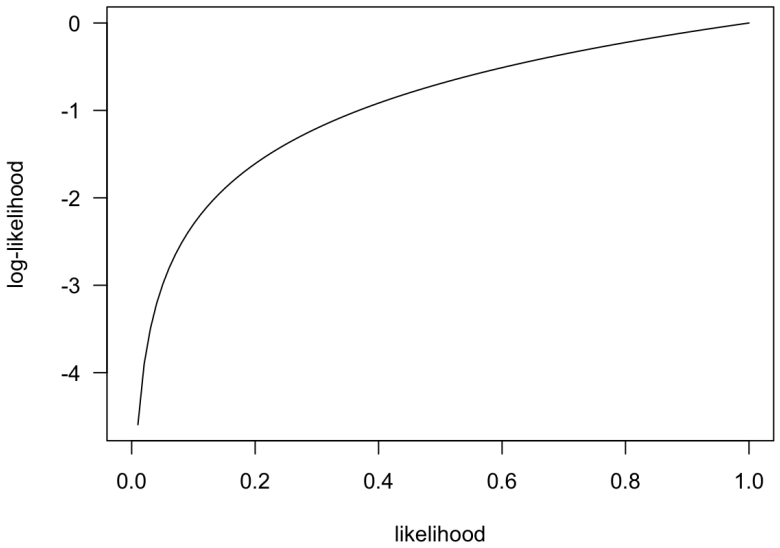
$$\mathcal{L}(c|N_{correct}) = \prod_{i=1}^{M} \mathcal{L}(c|N_{correct,i})$$

$$\text{->}$$

$$\ln \mathcal{L}(c|N_{correct}) = \sum_{i=1}^{M} \ln \mathcal{L}(c|N_{correct,i})$$

Log-likelihood

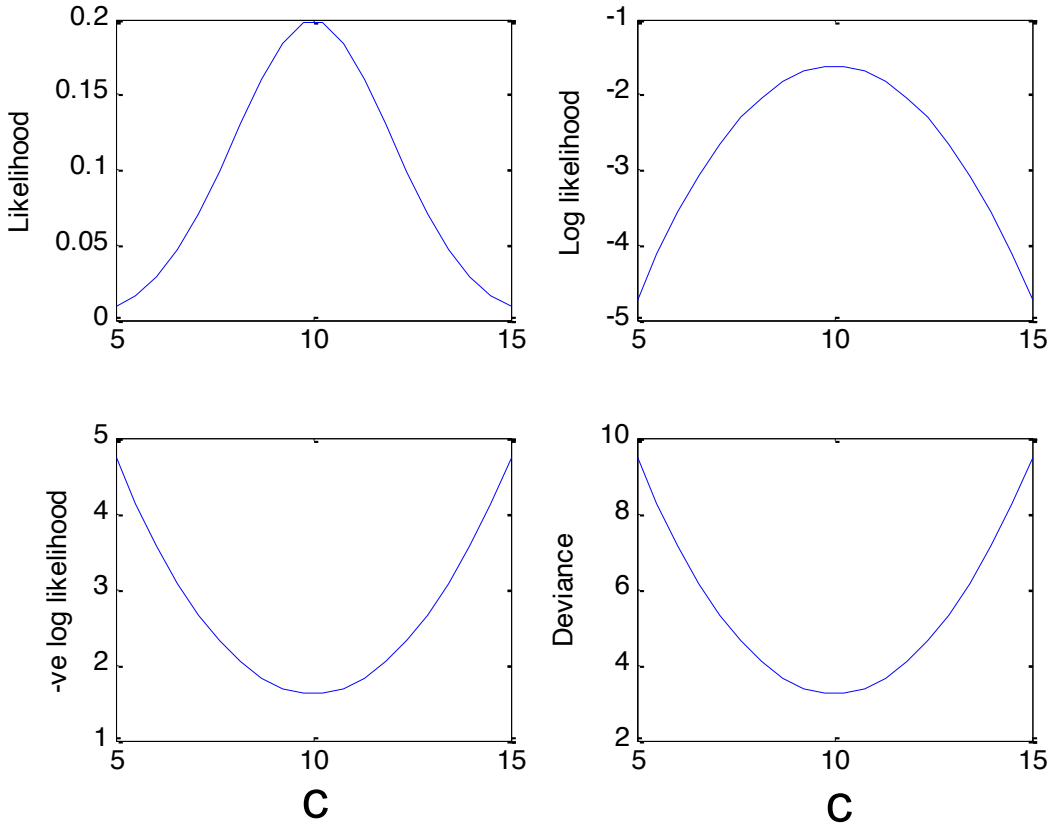
- Log-likelihood is a monotonic, compressive transformation of the likelihood
- Parameters that maximise the likelihood will also maximise the log-likelihood
- This makes numerical life easier
- But optim / optimize finds the **minimum** of a function



Deviance

- Instead of maximising log-likelihood, minimise negative likelihood
- Deviance: $-2 \times \ln \mathcal{L}(\theta|y)$
- Related to chi-square that Steve talked about earlier (briefly)
- Statistical measure of discrepancy between model and data (or "reality")
- As we'll see tomorrow, deviance can be used to compare fit of different models (AIC, BIC)

Likelihood, log-likelihood, deviance



c.ludwig@bristol.ac.uk

Likelihood: Explore the function `dbinom`

```
#Now we have NUMBER CORRECT FROM 20 TRIALS
observations=c(8, 7, 6, 6, 8, 10, 10, 14, 14, 19)
ntrials<-20

#Illustrating dbinom

#probability of observing 8 recalls in 20 trials if
#predicted recall probability is 0.5
dbinom(8,ntrials,.5)

#We can do this with a whole vector of recall frequencies
#and a matching vector of predicted recall probs
dbinom(observations,ntrials,discrim)
```

Now: A function to calculate neg LL

```r
#We can then log the probabilites, then sum them
#then take the negative
-sum(log(dbinom(observations,ntrials,discrim)))

#Now write a function to calculate LL
LL = function (c,N,distances,observations) {
  predictions=serpos(c, distances) #Using the function we made before
  LL_est=-sum(log(dbinom(observations,N,predictions)))
  LL_est=return(LL_est)
}

#now test it
LL(5,ntrials,temp_dists,observations)
```

Now we can use `optimize`

Same as before, but now we are minimising –LL, instead of minimising RMSD

```
#Now we need to use optimize to find maximum likelihood params
results<-optimize(LL,c(0,100),N=ntrials,distances=temp_dists,observations=observatic

#The outcome is again stored in "results"
param_ests<-results$minimum #param_ests has the best-fit parame estimates
error<-results$objective    #smallest error that could be obtained
param_ests
error

#Now add the curve onto the graph we already have:
#Now we need to calculate predictions with optim's estimated parameters
best_predictions <- serpos(param_ests, temp_dists)
```

And plot the final results

```
#Now add the curve onto the graph we already have:
#Now we need to calculate predictions with optim's estimated parameters
best_predictions <- serpos(param_ests, temp_dists)

#then we can add the points to the graph
points(best_predictions, type="p", col="red", las=1)
```

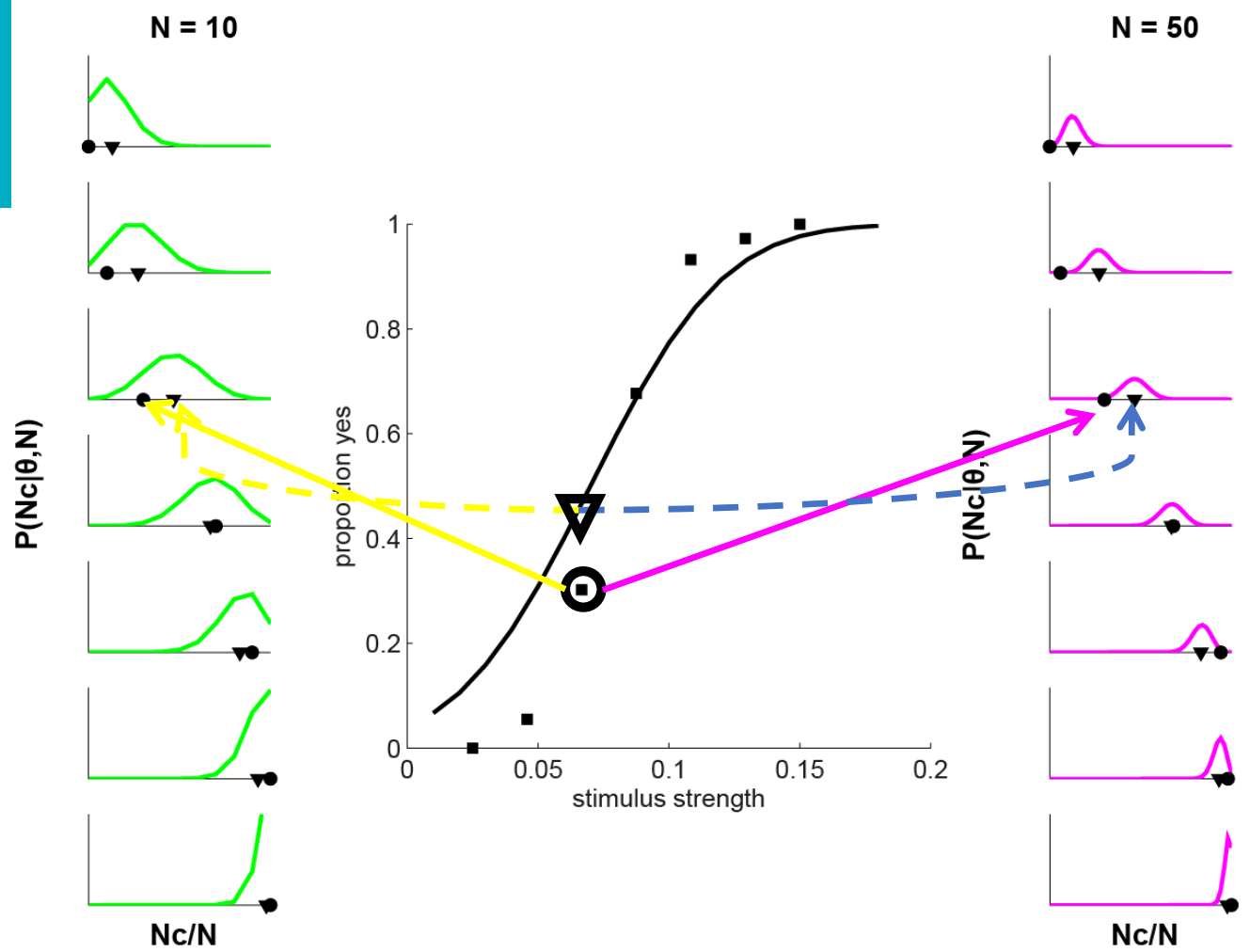
Summary #1

- Multiple ways of finding good (or “best”) parameters
- All involve finding the minimum of some function of the model parameters θ, given fixed data y : $f(\theta|y)$
- This function returns some measure of discrepancy between model prediction and data
  - Absolute error
  - RMSD
  - χ^2
  - …

Summary #2

- Likelihood (or negative log-likelihood or Deviance) is a principled distance metric
  - Statistically desirable characteristics
  - Basis for model comparison (tomorrow!)
  - Core ingredient of Bayesian parameter estimation and model selection
- Some models directly predict the probability (density) of the data -> likelihood of a set of parameters given the data (we'll get to models of choice and RT)
- Deterministic models need a "linking function" that connect the model predictions to noisy data

c.ludwig@bristol.ac.uk


N = 10
N = 50
P(Nc|θ,N)
P(Nc|θ,N)
proportion yes
stimulus strength
0
0.2
0.4
0.6
0.8
1
0
0.05
0.1
0.15
0.2
Nc/N
Nc/N