Temporal Distinctiveness Models of Memory

Plan for this session:

- Describe SIMPLE model of memory
- Implement and explore model's behaviour
- Compare model and data
- Use `optimize` in R to find
- (a) RMSD-minimising parameter values
- (b) maximum likelihood parameter values

Aims of the Model

A model of human memory retrieval: Applied to tasks such as serial recall, free recall, probed recall

- Currently: Many computational models, but task-specific
- Ideal: Same model for many different memory tasks (cf. Ward, Tan, Bhatarah)

Explore assumption that STM and LTM phenomena need different models

- No STM-LTM distinction in model
- Apply to data previously taken as evidence for STM/LTM distinction
- Some long-standing claims for unitary memory system, but need models to explore the possibility

Reference

Brown, G.D.A., Neath, I., & Chater, N. (2007). A temporal ratio model of memory. *Psychological Review, 114,* 539-576.

A more complex version of SIMPLE has three parameters (to allow for omission errors in remembering) — see slides at the end

Here we will use a one-parameter version

Key principles

SIMPLE (Scale-Invariant Memory, Perception, and Learning)

Memory is temporally organised

- Memories organised in terms of their temporal distances
- Memory retrieval is like temporal discrimination (old claim)

Same principles of forgetting and retrieval apply over short and long timescales (seconds to weeks)

No forgetting due to trace decay

- All forgetting due to interference

Starting Point: Exemplar Models of LTM

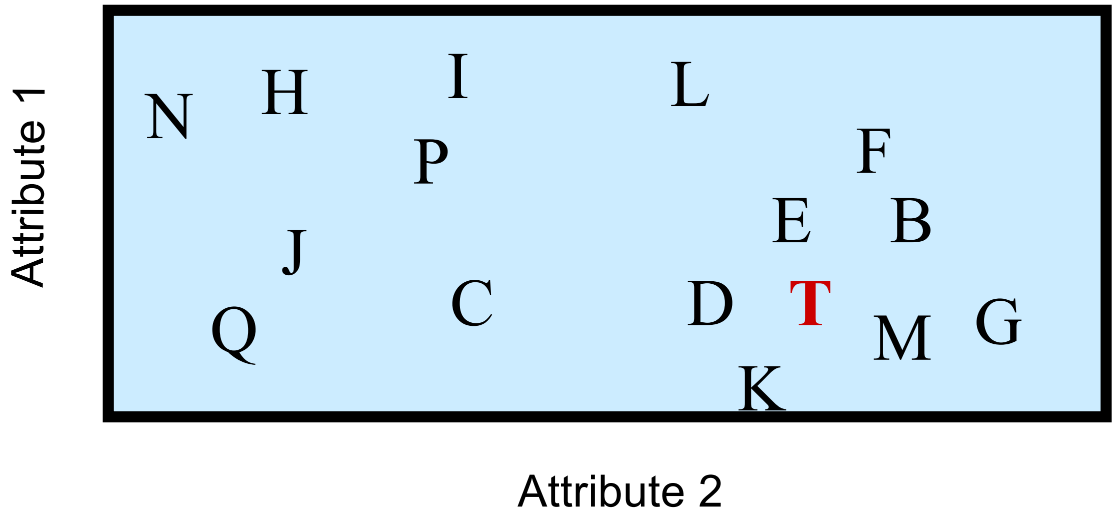
Items are located within a multi-dimensional space

Items close to one another will be more similar

(exponential similarity-distance function)

Similar items: easy to categorise together

Similar items: hard to identify and discriminate

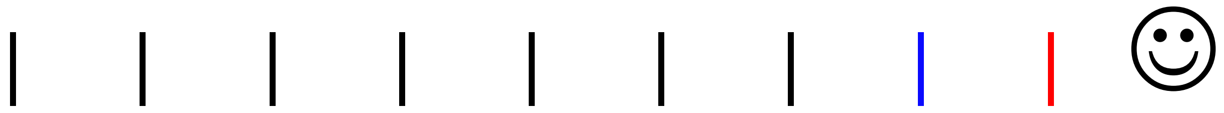


Need to consider closeness/similarity along a *temporal distance* dimension as well?

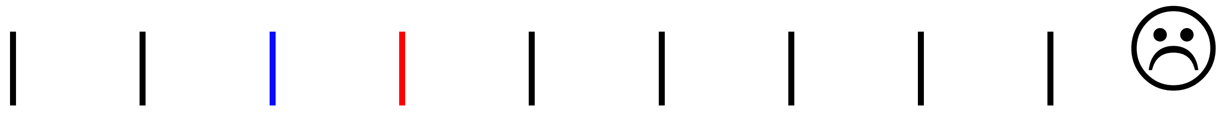
Recency Matters

Distance and Discriminability

Consider two telegraph poles separated by 10 m, with the nearest being 5 m away: Easily discriminable



But two telegraph poles separated by 10 m, with the nearest being 55 m away: Difficult



Led to ratio-rule / temporal discriminability models of recency effects in memory (Baddeley; Baddeley & Hitch; Bjork & Whitten; Crowder; Tan & Ward)

Assumption: The confusability of two items in memory depends on their relative temporal distances at time of recall

The confusability of any two items is a function of the *ratio* of their temporal distances: $S_{i,j} = (T_i/T_j)^c$.

For example: two items that occurred 4 and 5 s ago will be more similar to one another $(4/5)^c$ than will two items that occurred 1 and 2 s ago $(1/2)^c$.

Note: The parameter *c* marks a difference from earlier ratio-rule models

(A parenthetical note)

Note distinction between:

A core principle of the model:

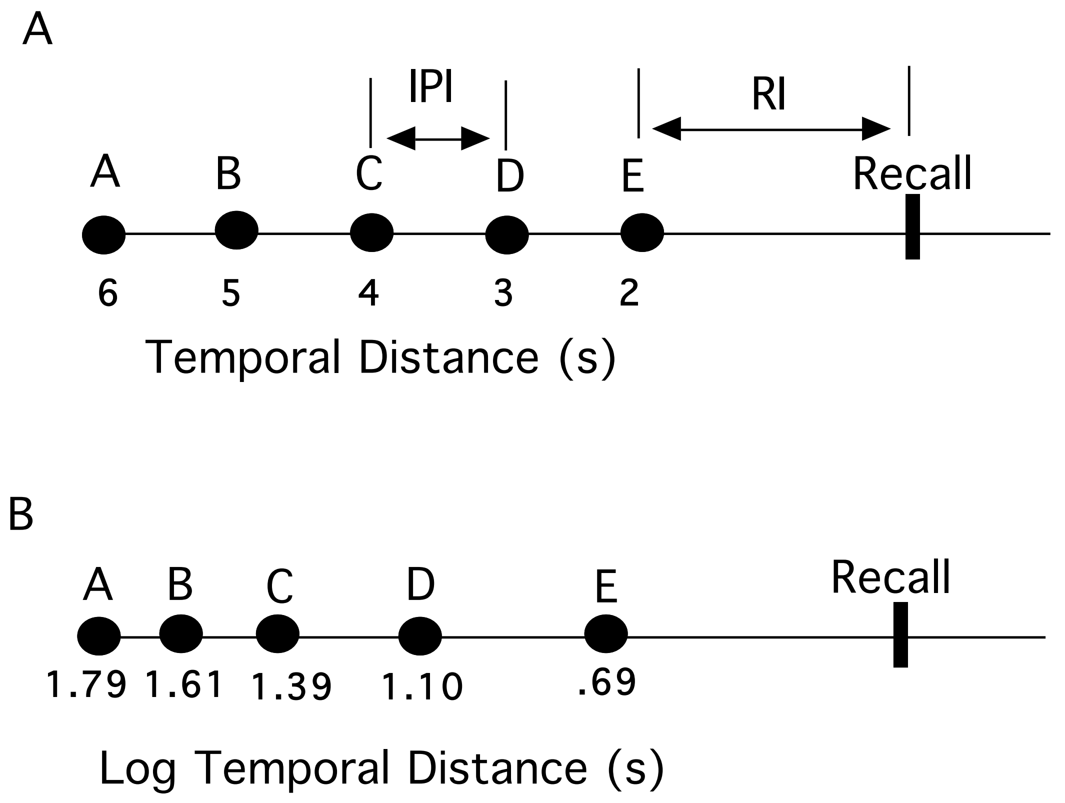
- The confusability of two points in memory depends on temporal distance ratios

Numerous subsidiary and supplementary assumptions needed to get model up and running as applied to a particular paradigm, e.g.:

- Assumptions about output processes in tasks such as free recall
- Simplifying assumptions about the temporal extension of items
- Effect of distractors

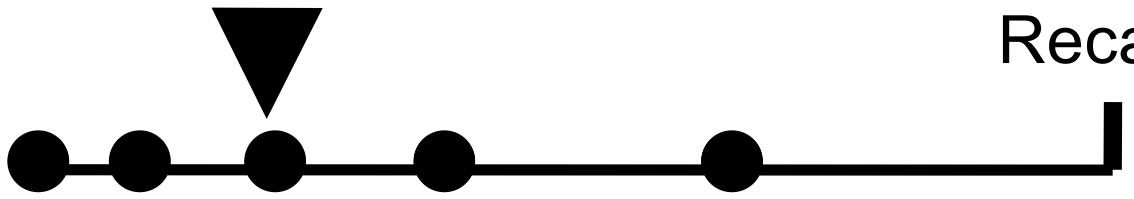
Confusability as Temporal Distance Ratios: The SIMPLE model

Implementing similarity/confusability as a temporal distance ratio effectively instantiates the telegraph pole analogy:



Assumption: The discriminability of a memory item is inversely related to that item's similarity to all other items

In other words, items that are confusable with many others (in terms of their temporal distances) will be less well recalled.



Note: The fact that confusability with several nearby items matters, not just the nearest neighbour, marks another difference from earlier ratio-rule models

Assumption: Discriminability

D_i is the discriminability in memory of item *i* of *n*

(Like a simplified choice rule)

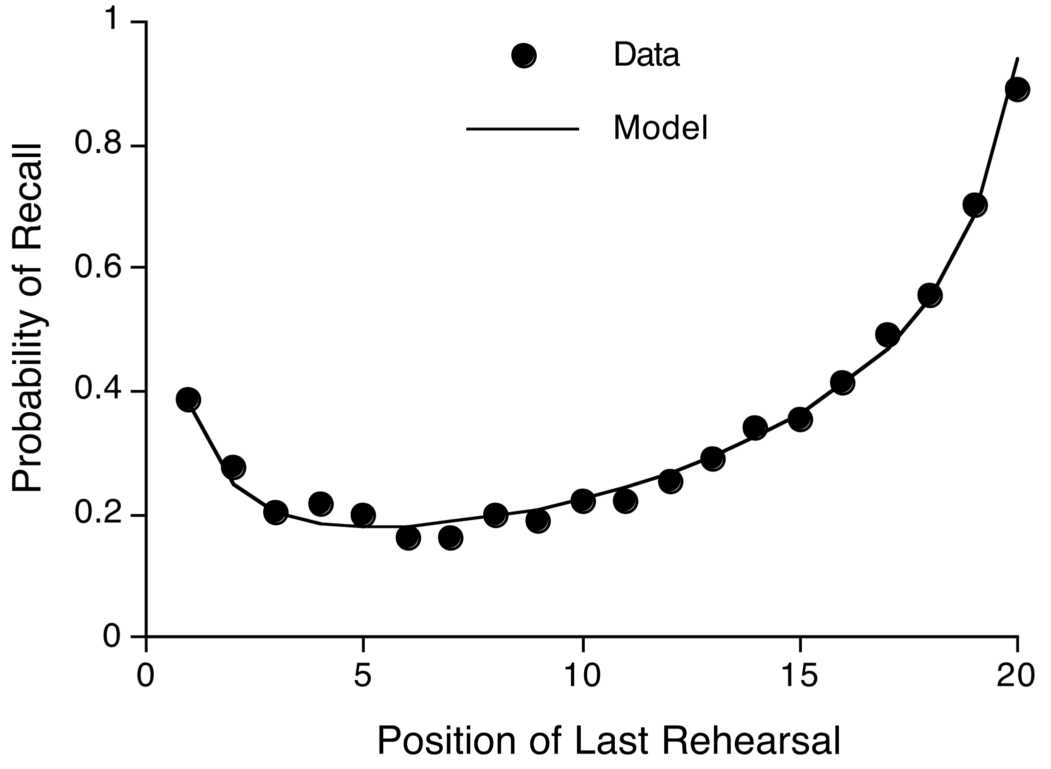
$$D_i = \frac{1}{\sum_{j=1}^{n} \eta_{i,j}}$$

where $$\eta_{i,j} = e^{-c.d_{i,j}}$$

and $d_{i,j}$ is distance between items in **log** space

Serial Position Effects

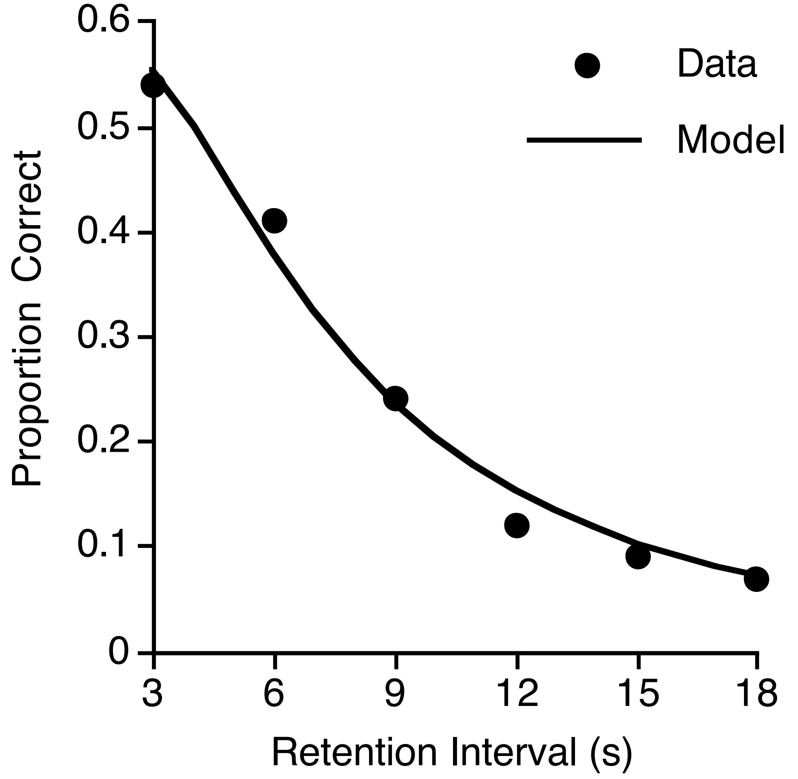
Performance is best plotted as a function of *position of last rehearsal* (e.g. Tan & Ward, 2000):



Unitary ratio model of serial position effects may be possible? (note: primacy effects in normal population are at least partly due to rehearsal).

Forgetting over Time

Data from Peterson & Peterson (1959)



Forgetting occurs due to increasing proactive interference from earlier items

No forgetting due to trace decay

Implementation Steps

1) Create a vector of the temporal distances

2) Log transform the temporal distances

3) Calculate the summed similarity of each item's temporal distance to every other item's temporal distance using:

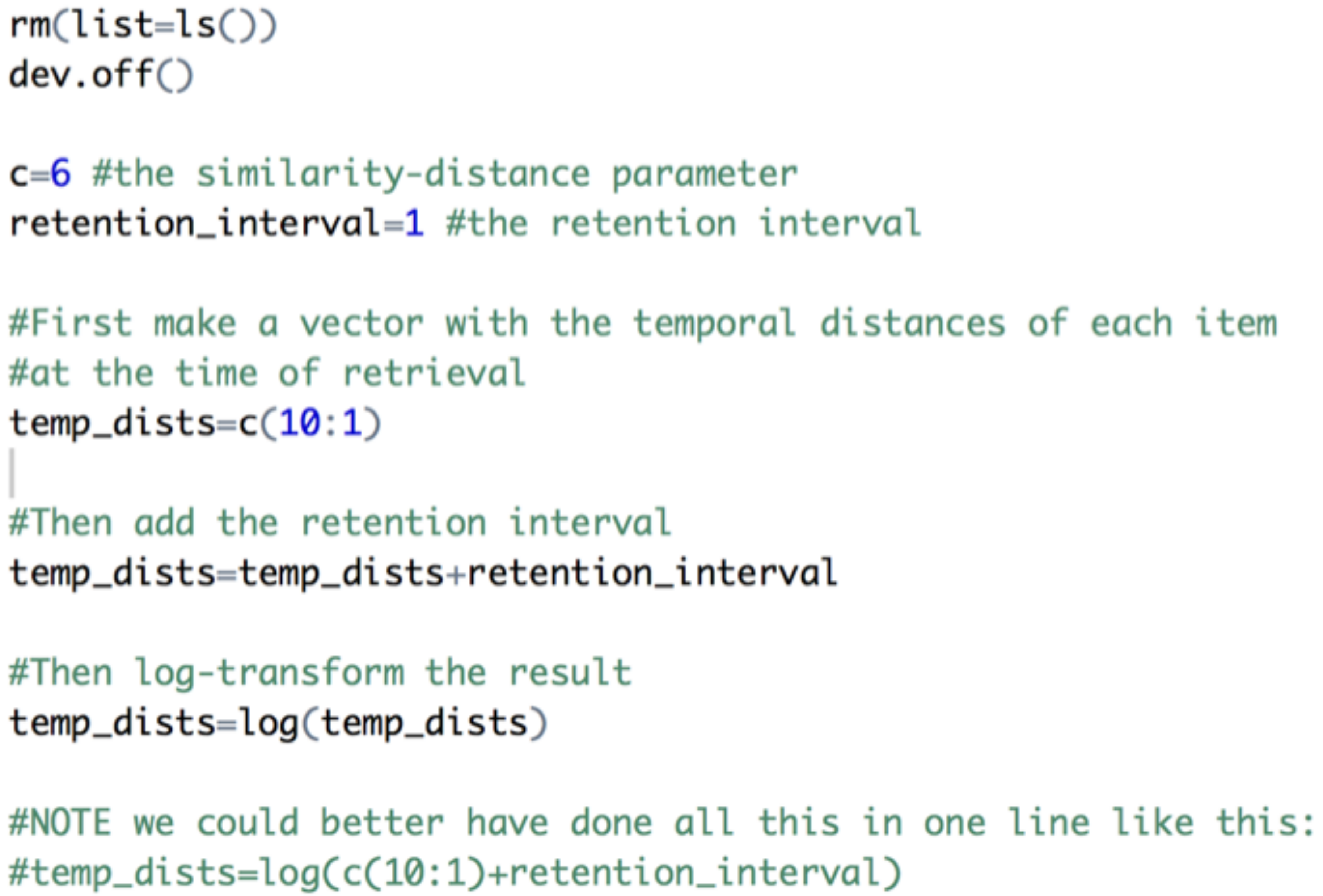
$$\eta_{i,j} = e^{-c.d_{i,j}}$$

where $d_{i,j}$ is distance between items in log space and c is a free parameter

4) Calculate the resulting discriminability of each item

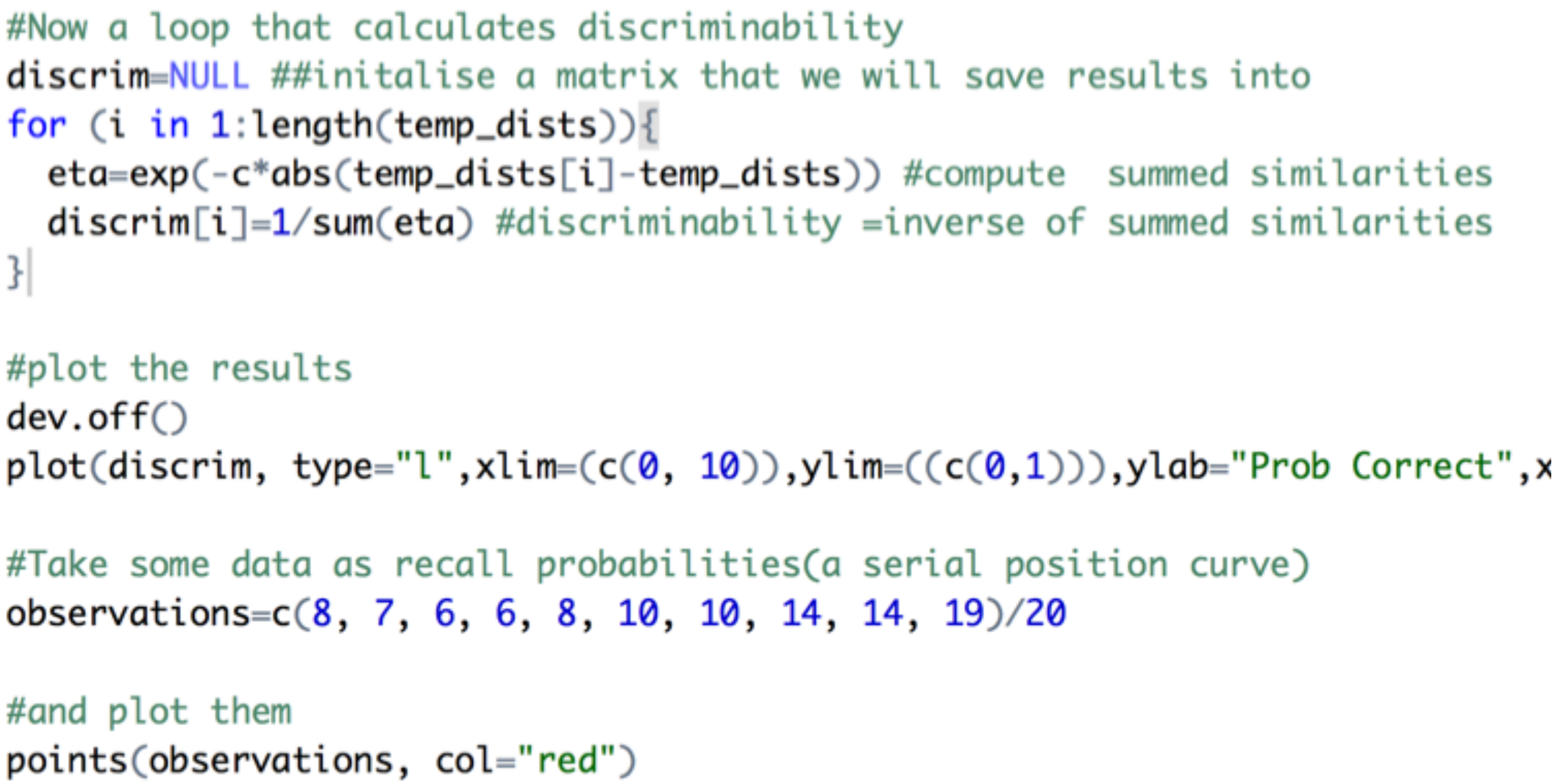
5) Plot the serial position curve

Code Part 1



```r
rm(list=ls())
dev.off()

c=6 #the similarity-distance parameter
retention_interval=1 #the retention interval

#First make a vector with the temporal distances of each item
#at the time of retrieval
temp_dists=c(10:1)

#Then add the retention interval
temp_dists=temp_dists+retention_interval

#Then log-transform the result
temp_dists=log(temp_dists)

#NOTE we could better have done all this in one line like this:
#temp_dists=log(c(10:1)+retention_interval)
```

Code Part 2



```
#Now a loop that calculates discriminability
discrim=NULL ##initalise a matrix that we will save results into
for (i in 1:length(temp_dists)){
  eta=exp(-c*abs(temp_dists[i]-temp_dists)) #compute  summed similarities
  discrim[i]=1/sum(eta) #discriminability =inverse of summed similarities
}

#plot the results
dev.off()
plot(discrim, type="l",xlim=(c(0, 10)),ylim=((c(0,1))),ylab="Prob Correct",x

#Take some data as recall probabilities(a serial position curve)
observations=c(8, 7, 6, 6, 8, 10, 10, 14, 14, 19)/20

#and plot them
points(observations, col="red")
```

Explore basic SIMPLE model

Calculate a serial position curve for a 10-item list (code provided)

Assume items take 1 s to present, and recall occurs 1 s after the last item – i.e., if `dist` contains the temporal distances of each item at the time of retrieval:

```
dist=[10:-1:1]
```

What happens to the curve if you add a retention interval?

```
retint = 20; dist=[10:-1:1]+retint
```

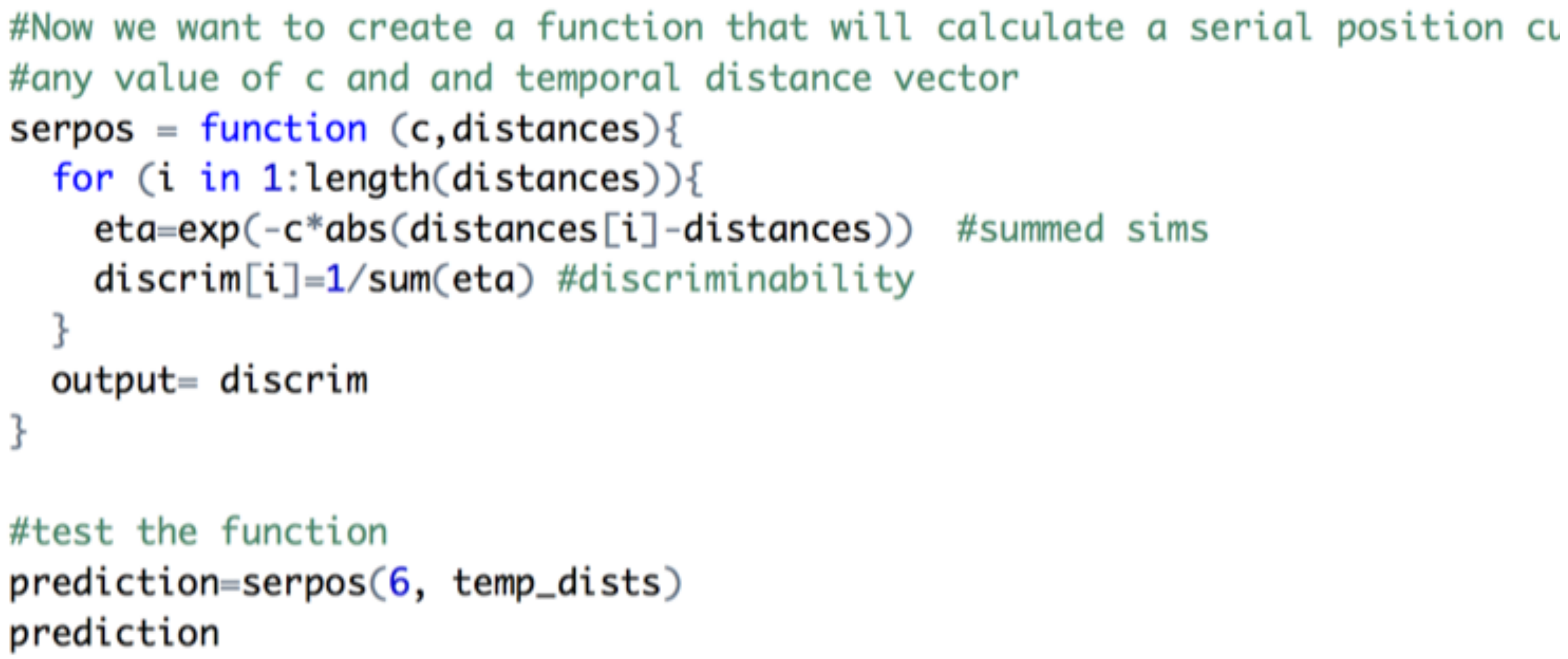
What happens to the curve if you add a temporal gap around one item?

```
dist=[12:-1:8 6 4:-1:1]
```

What happens if you change the list length?

Now: Parameter Estimation 1

First we need to turn the code into a function:

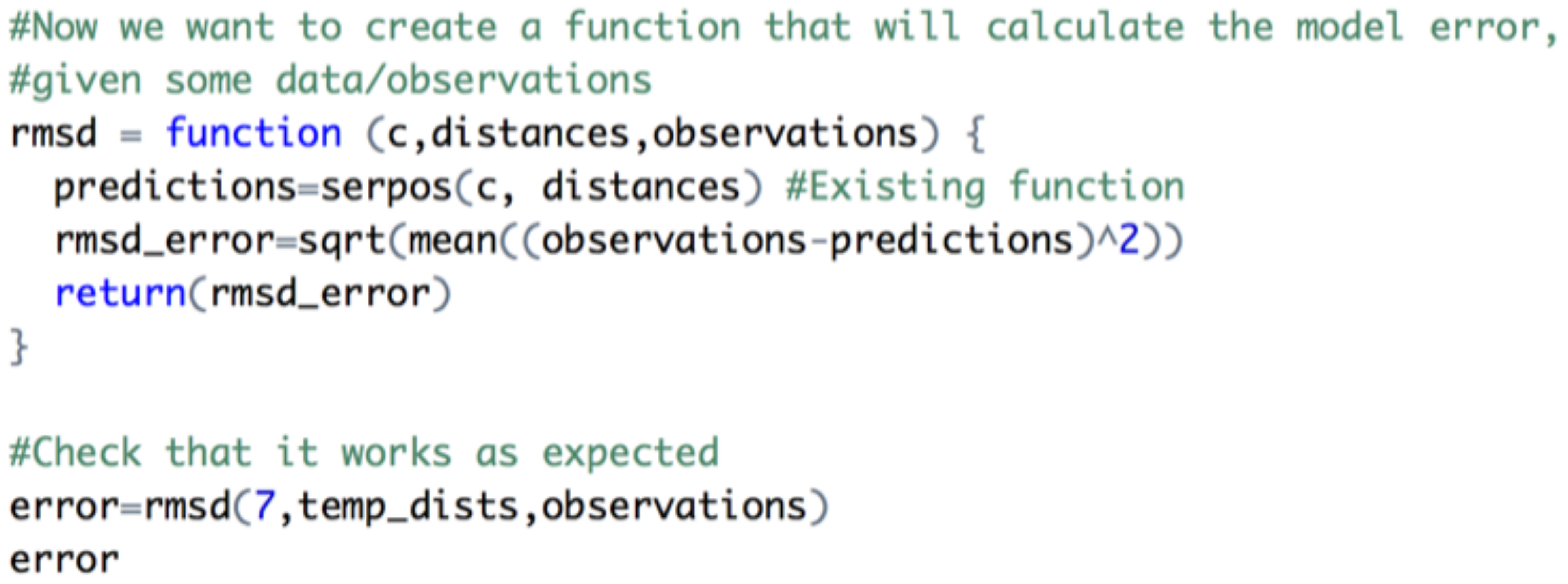


```
#Now we want to create a function that will calculate a serial position cu
#any value of c and and temporal distance vector
serpos = function (c,distances){
  for (i in 1:length(distances)){
    eta=exp(-c*abs(distances[i]-distances))  #summed sims
    discrim[i]=1/sum(eta) #discriminability
  }
  output= discrim
}

#test the function
prediction=serpos(6, temp_dists)
prediction
```

Again, try it out for various values of *c*

Now: Parameter Estimation 2

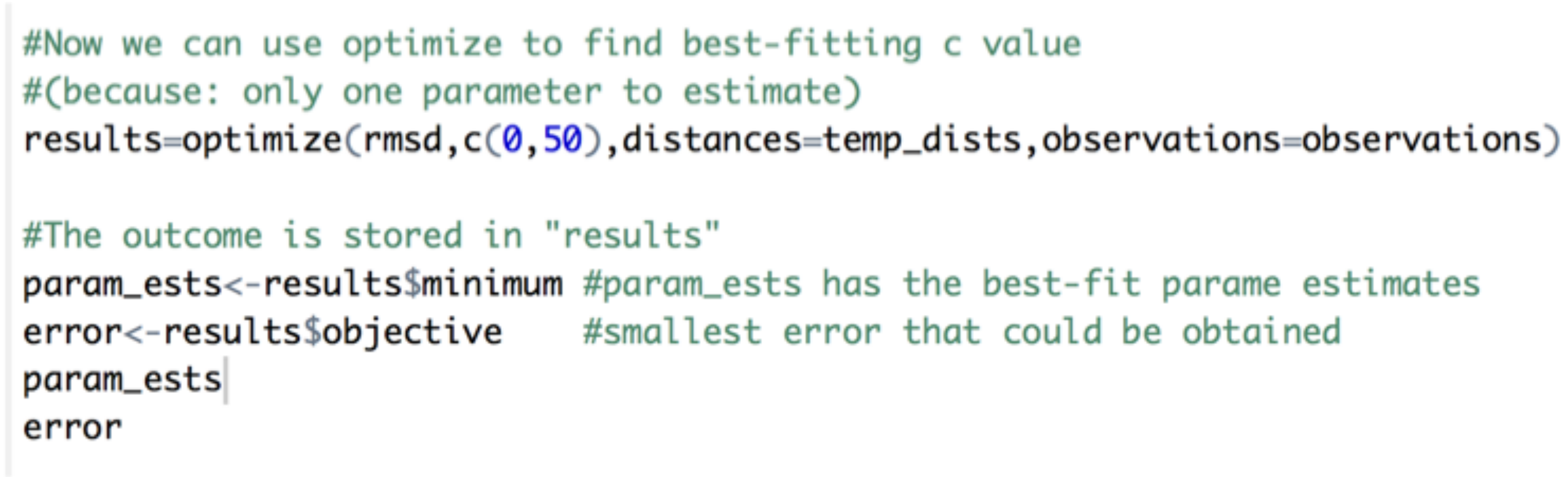
Now we need a function to calculate RMSD error:



```
#Now we want to create a function that will calculate the model error,
#given some data/observations
rmsd = function (c,distances,observations) {
  predictions=serpos(c, distances) #Existing function
  rmsd_error=sqrt(mean((observations-predictions)^2))
  return(rmsd_error)
}

#Check that it works as expected
error=rmsd(7,temp_dists,observations)
error
```

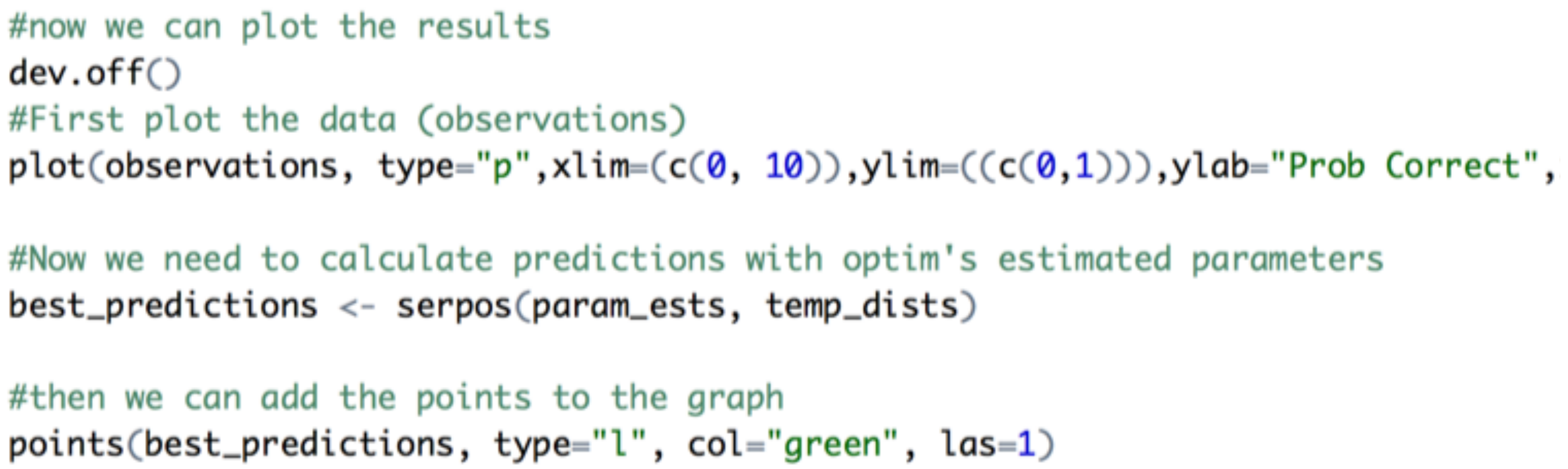
Estimating

Because we only have one parameter *c* to estimate, it is easier to use `optimize`



```
#Now we can use optimize to find best-fitting c value
#(because: only one parameter to estimate)
results=optimize(rmsd,c(0,50),distances=temp_dists,observations=observations)

#The outcome is 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
```

Plotting the Best-fit Results

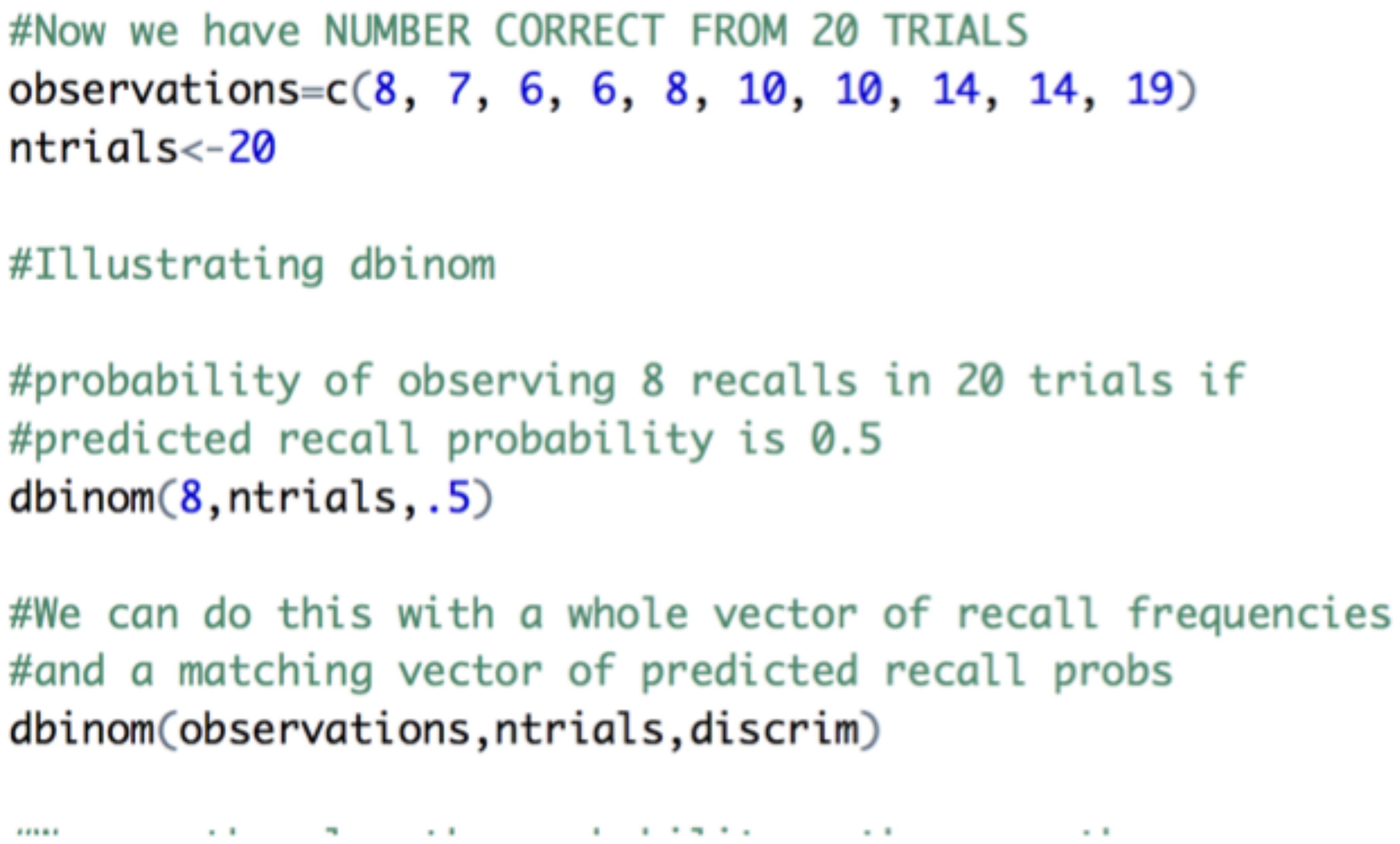


```
#now we can plot the results
dev.off()
#First plot the data (observations)
plot(observations, type="p",xlim=(c(0, 10)),ylim=((c(0,1))),ylab="Prob Correct",

#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="l", col="green", las=1)
```

PART TWO: OVER TO SIMON

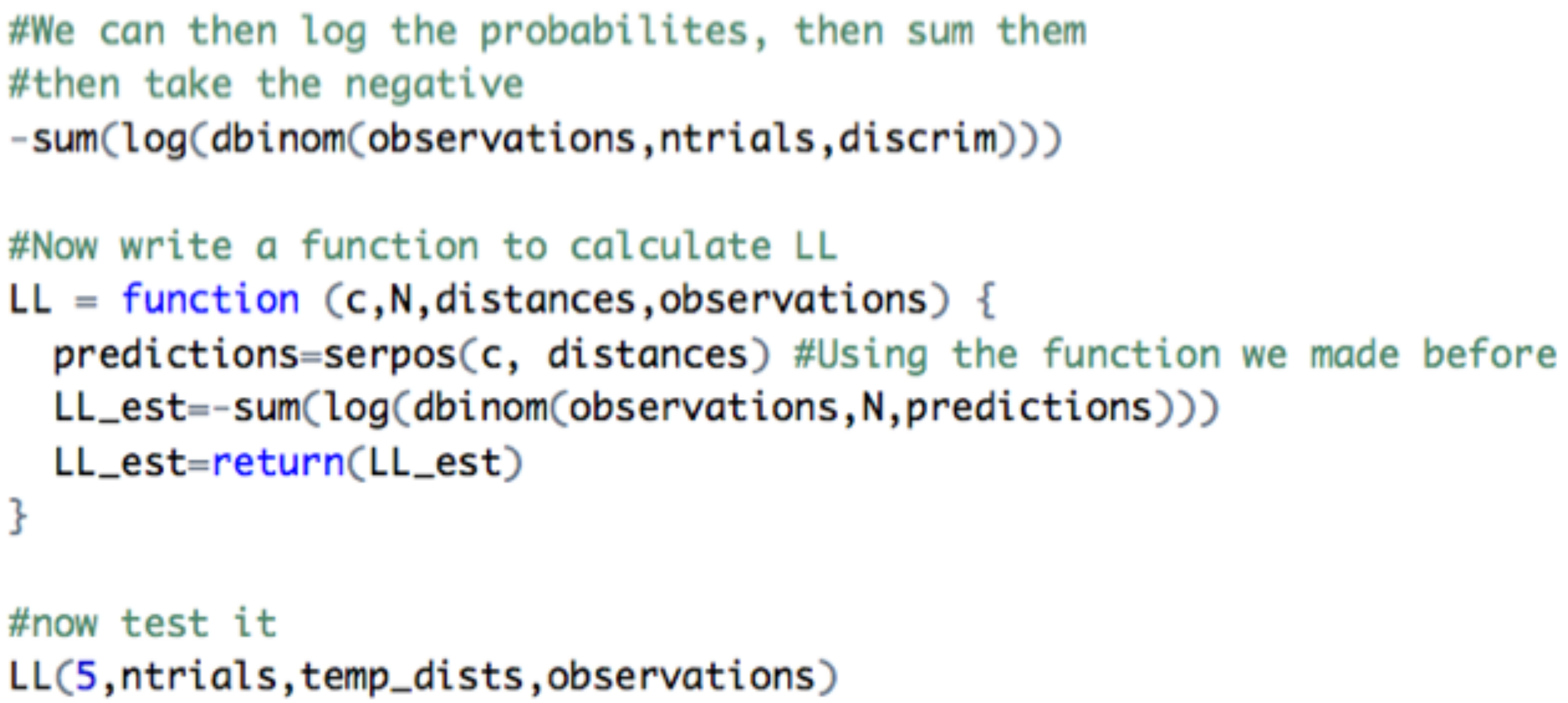
Likelihood

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)

[illegible]
```

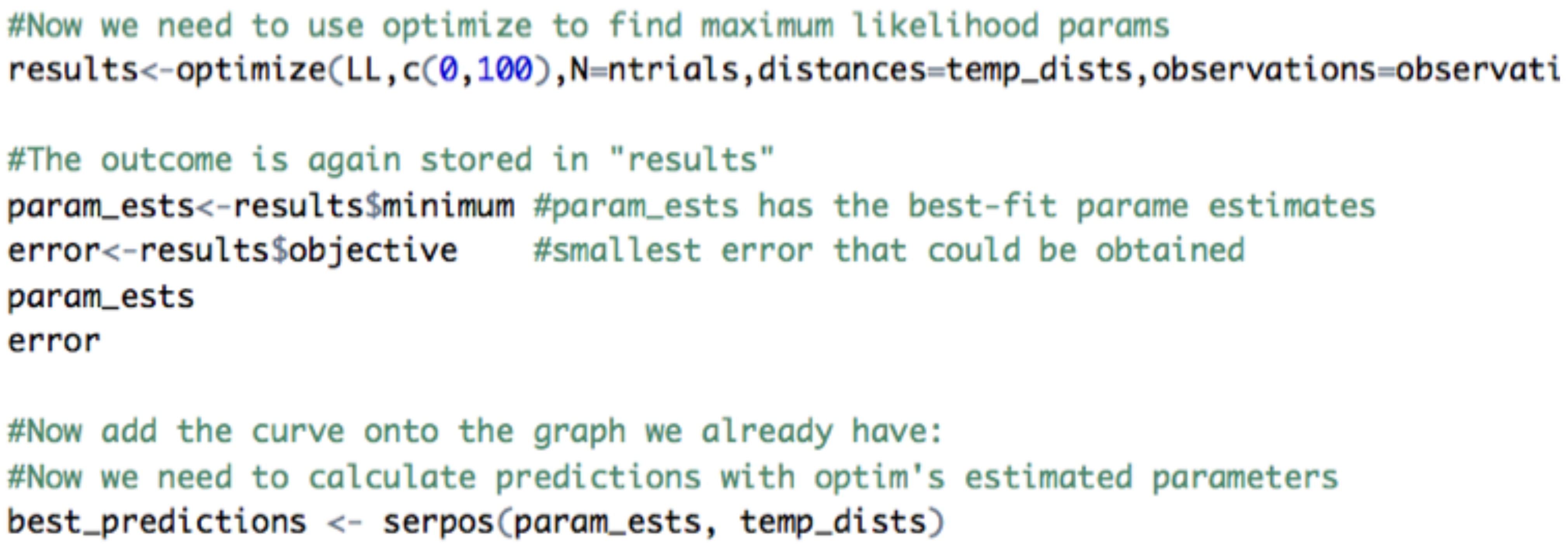
Now: A function to calculate neg LL



```
#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)
```

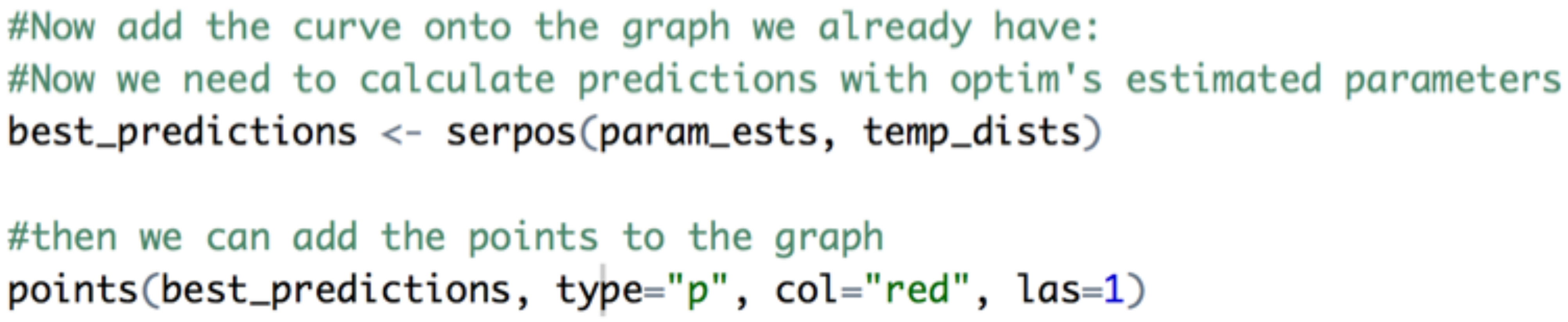
Now we can use `optimize`

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=observati

#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)
```

END

Extending the Model

See textbook listings for more example SIMPLE code

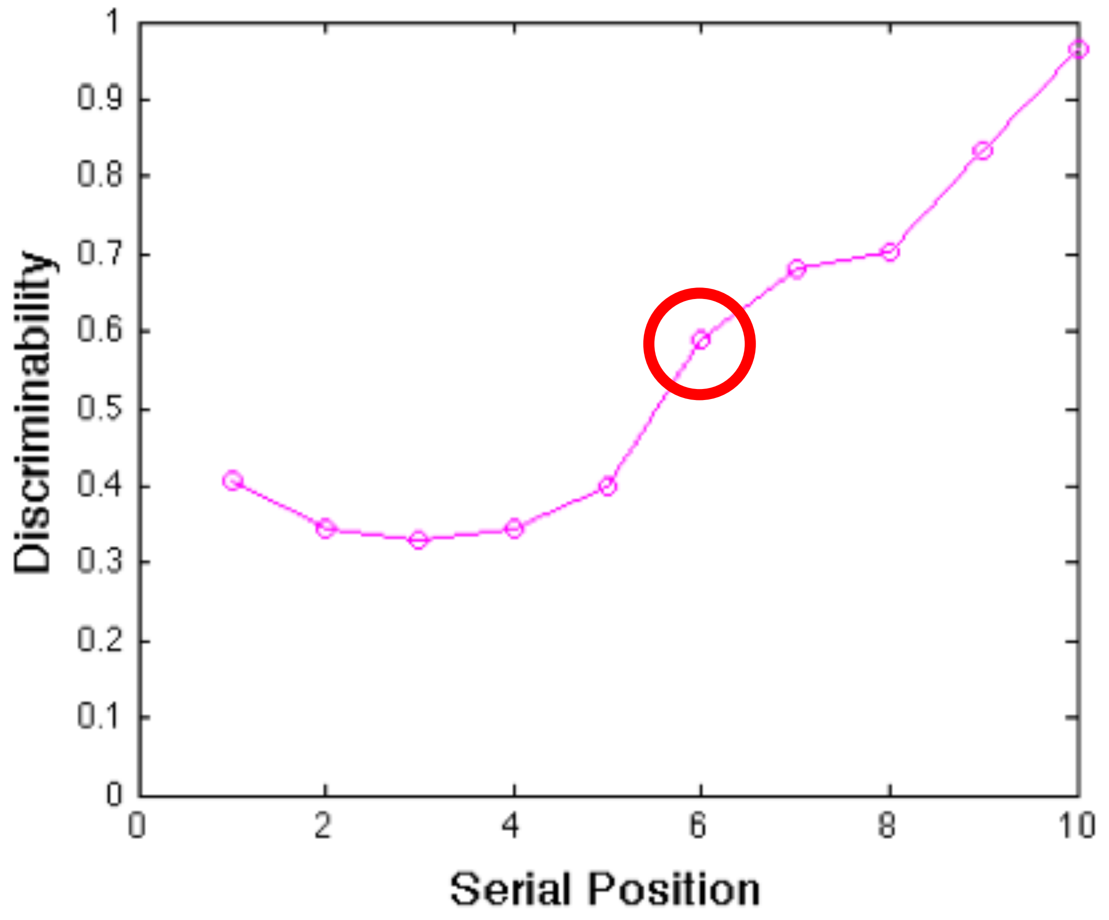
Some relevant experimental evidence

Extensions to model

- Add in additional dimensions to account for confusability effects
- Add in a threshold parameter to account for omission errors
- Take account of changing temporal perspective during recall
- Represent the fact that items have temporal extension (i.e., abandon the assumption that item locations are point sources)

Temporal Isolation Effects

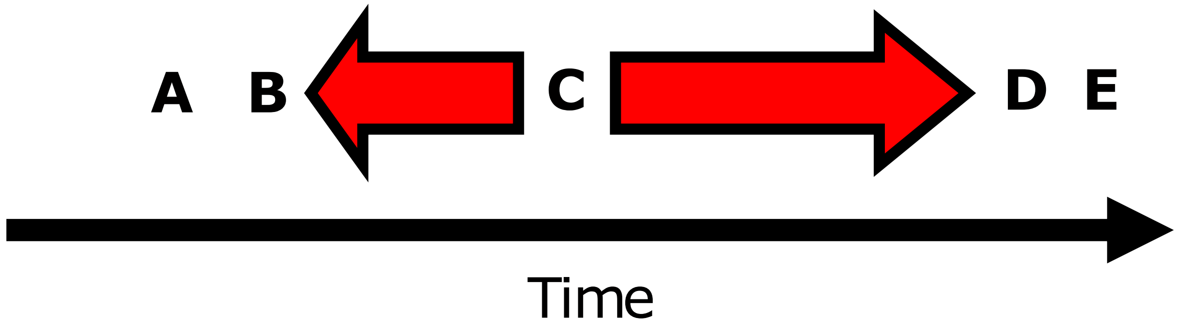
```
dist=[12:-1:8 6 4:-1:1]
```



Evidence for Time-based Encoding

“Temporal distinctive” items are those that occupy relatively isolated locations along the temporal dimension in memory

Such isolation occurs because of recency, but can also be created experimentally



Such items have fewer near neighbours on the temporal dimension

If temporally isolated items are better recalled: Evidence for time-based encoding

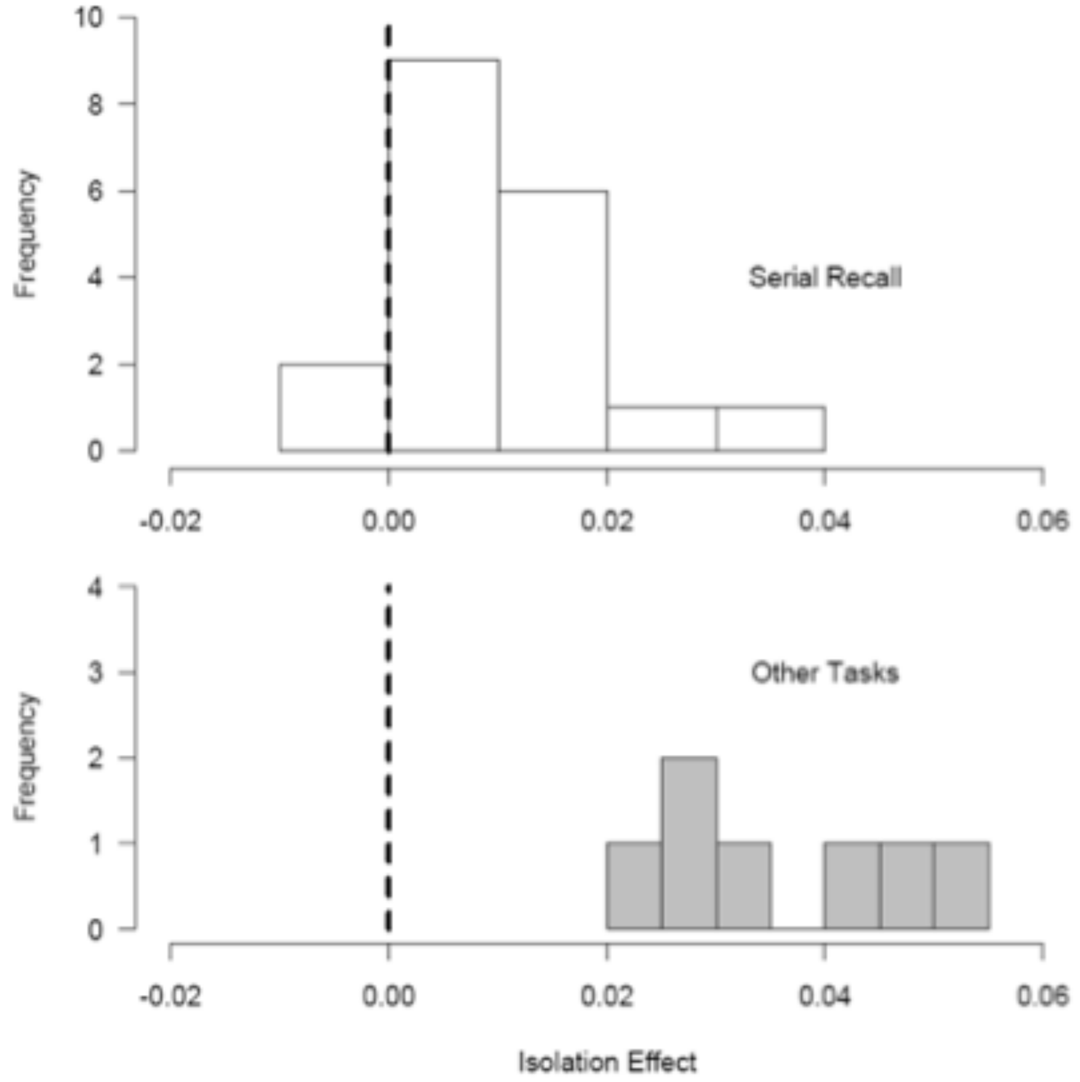
Temporal Isolation Effects

Temporal isolation effects **are** observed in:

- Free recall (Brown, Morin, & Lewandowsky, 2006)
- Recognition (Morin, Brown, & Lewandowsky, 2010)
- Free reconstruction of serial order information (Lewandowsky, Nimmo, & Brown, 2008)
- Forward serial recall in running memory span (Geiger & Lewandowsky, 2008)

Either not observed in forward serial recall, or much smaller

- Forward serial recall (Lewandowsky & Brown, 2005; Lewandowsky, Brown, Wright, & Nimmo, 2006; Lewandowsky, Nimmo, & Brown, 2008; Morin, Brown, & Lewandowsky, 2010; Nimmo & Lewandowsky, 2005; 2006; Parmentier, King, & Dennis, 2006)
- Probed serial recall (Lewandowsky, Brown, Wright, & Nimmo, 2006)



Suggests that:

- Temporal isolation effects occur in all tasks
- They are smaller in forward serial recall

People Often Omit Items

How to transform the item discriminabilities into probabilities of actual recall?

Assume items will only be recalled if their discriminability is above a threshold *t*. Probability of recalling item *i* will be given by:

$$P(R_i \mid D_i) = \frac{1}{1 + e^{-s(D_i - t)}}$$

The parameter *s* determines the “abruptness” of the threshold

Short-term Serial Recall

In serial recall:

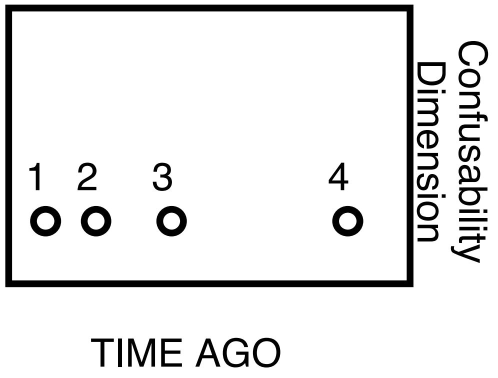
- The same principles govern memory retrieval
- Phonological confusability: Non-temporal dimension
- Need to include other dimensions in the memory space
- ((Extended primacy arises due to changing temporal perspective of recall (like walking away from the line of telegraph poles) [and/or output interference with representation along a positional dimension]))

Phonological Confusability Effects

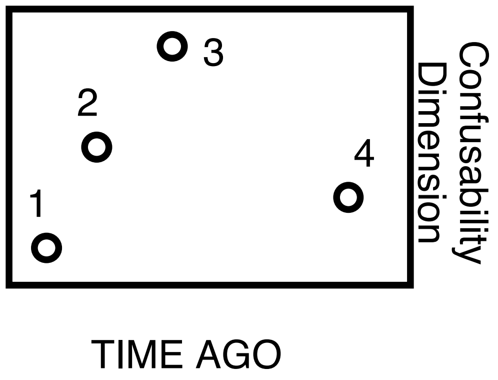
After a short time interval, STM is reduced for confusable items

Items can be represented in the model in terms of position along both a temporal and a "phonological confusability" dimension (should really be multi-dimensional articulatory space but principle is the same)

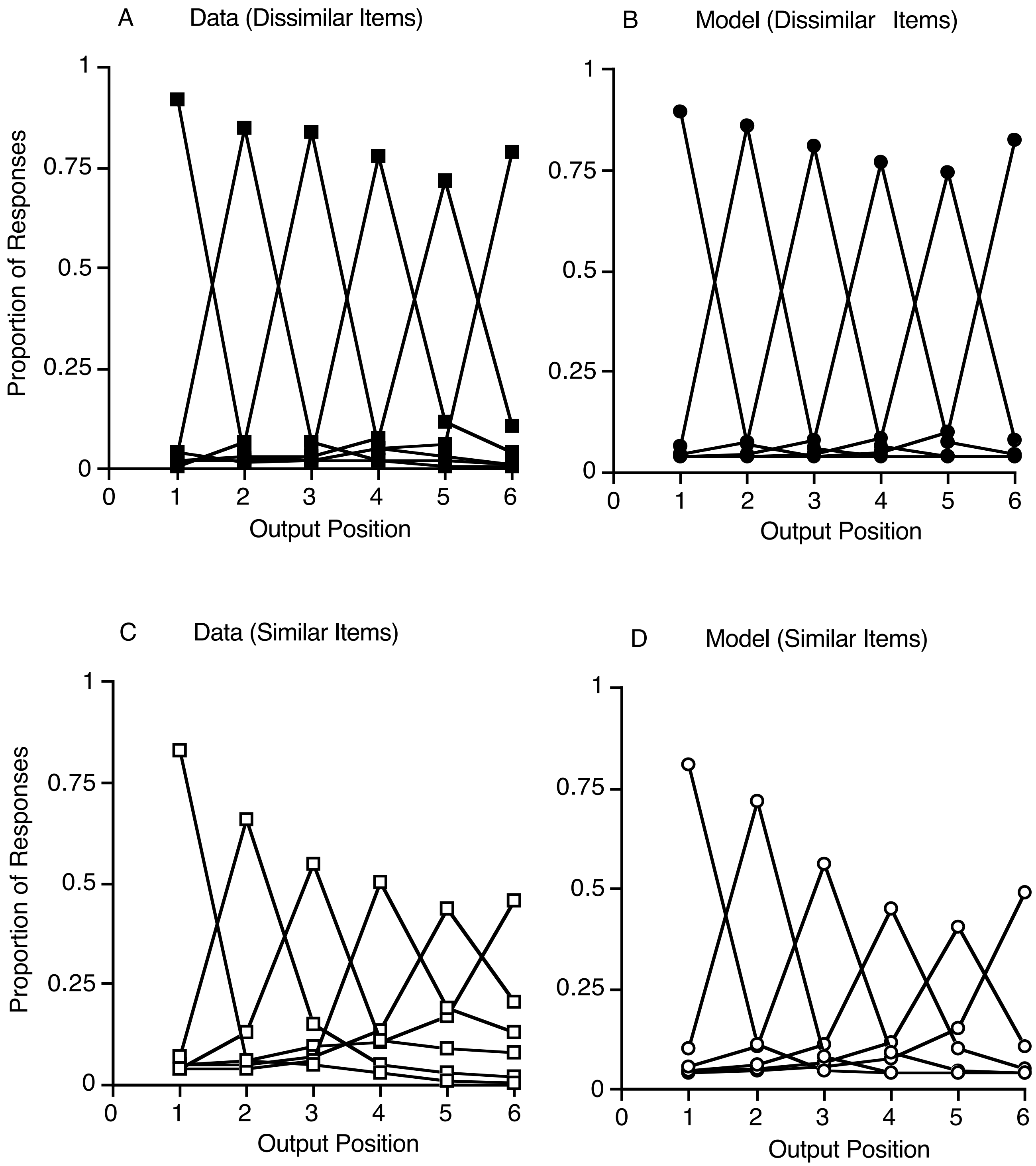
Confusable list:



Non-confusable list:



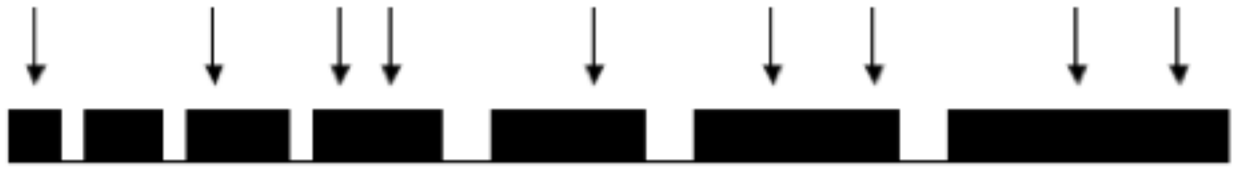
(Other dimensions important; not included here)


A
Data (Dissimilar Items)
B
Model (Dissimilar Items)
C
Data (Similar Items)
D
Model (Similar Items)
Proportion of Responses
Output Position
0
0.25
0.5
0.75
1
0
1
2
3
4
5
6

Temporal Extension

SIMPLE makes the simplifying assumption that items occupy point locations along the temporal dimension.

In fact: Items have temporal extension – occupy segments



Current versions of the model assume that any given point along the time-line may be cued, and this cue "activates" items in the same was as assumed by SIMPLE

Relation Between Time and Memory

Is memory retrieval like discriminating temporal intervals?

If the temporal distances (times from the point of recall) of items are hard to distinguish, the items will be difficult to retrieve

The ease of identifying the serial position of an item in memory should reflect the confusability of that item's temporal distance

A purely *temporal* task (identifying tones with different durations) should enable prediction of *memory* performance (Brown, Vousden, & McCormack, 2009)

Memory for Order *vs*. Timing

Present seven items at a rate of three per second (fast enough to prevent rehearsal; items visually presented and read aloud)

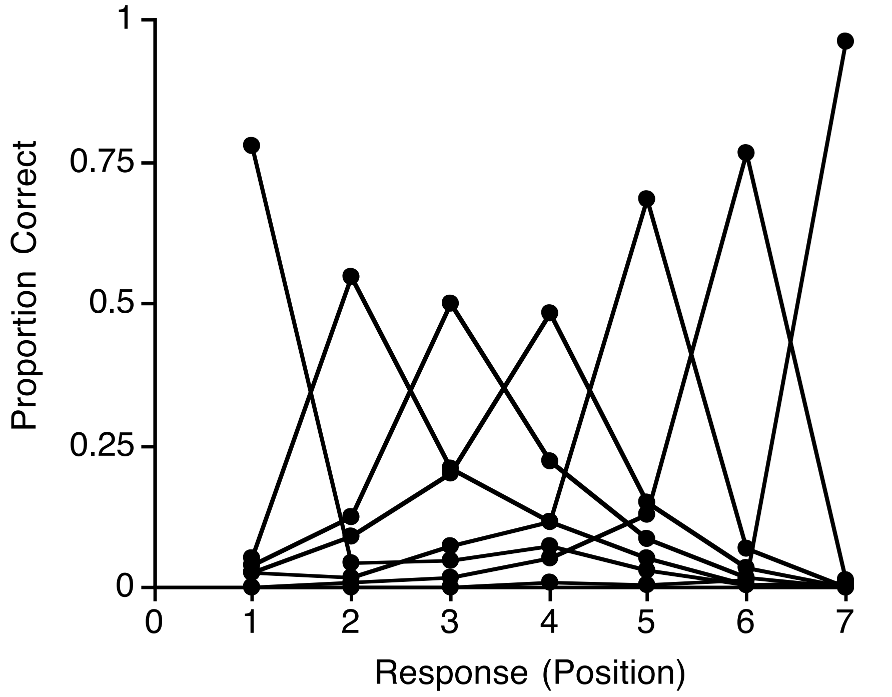
Immediately after each list, present an item from the list. Participants must respond with the item's position

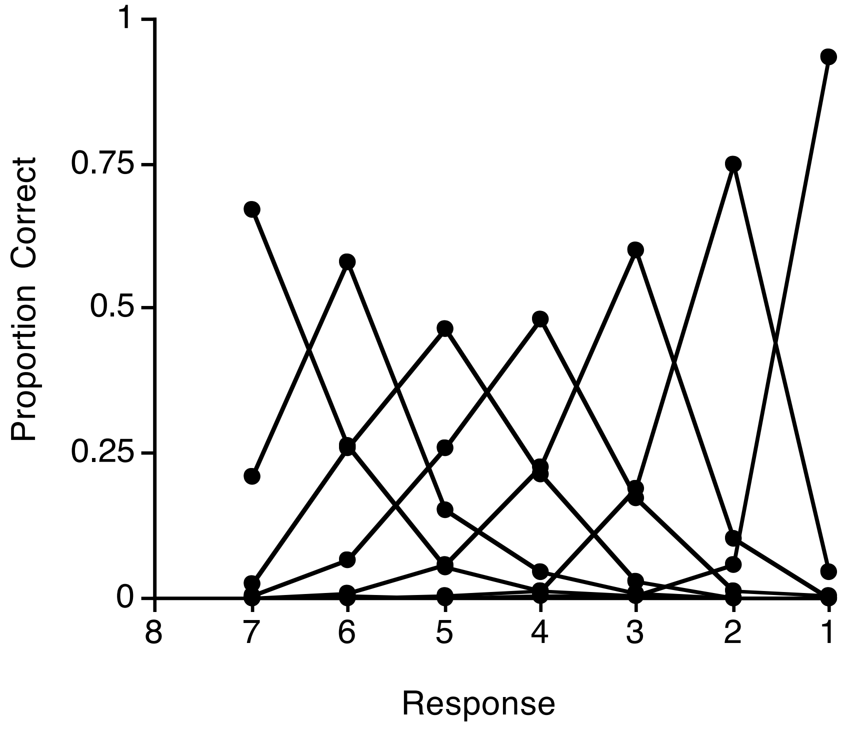
- Temporal distances of items are .33 s, .67 s, ………… 2.33 s

Examine absolute identification of tones; durations equal to the temporal distances of the items in memory.

Seven tone durations:

- 0.33 0.67 1.0 1.33 1.67 2.0 2.33 s
- Play durations to S with labels 1 to 7
- Then present durations to S in random order; S must respond with the label for each one


Proportion Correct
Response (Position)


Proportion Correct
Response