

## Table of Contents

[Output from REM](#)  
[Confidence as a probability judgment](#)  
[Decision rule](#)  
[Miscalibration](#)  
[How can we model this miscalibration?](#)  
[Dyad Models](#)

## Output from REM

Output from REM is the odds of an item being old as opposed to new:  $\phi$

In some cases, this might be stated as the log odds:  $l = \log(\phi)$ .

### Confidence as a probability judgment

In terms of confidence, let's assume that people are communicating their confidence in terms of the probability of correct. One proposal is that participants directly map their odds into a probability judgment  $p$  so that:

$$p = \frac{\phi}{1 + \phi}$$

### Decision rule

If the agent was making a decision based on  $p$  then they would choose old when  $p > .5$ ; otherwise state new.

Group decision models will mimic the models that use the odds.

### Miscalibration

But, we know that people's confidence judgments are miscalibrated so that often people are overconfident (Koriat, 1980; Lichtenstein et al., 1978; McClelland & Bolger, 1994), though sometimes they are underconfident. When they are overconfident when they report they are, say, 80% confident that an item is old they are only correct 70% of the time. Similarly, when they say they are 20% confident an item is old (i.e., a 80% chance the item is new), the item is old 30% of the time (i.e., 70% of the time it is new).

### How can we model this miscalibration?

Consider the Prelec weighting function. This is used in decision making to capture how people distort probabilities when making decisions, overweighting or underweighting probabilities when evaluating gambles. According to the function the weight  $w$  people use is determined by

$$w = \exp(-\beta(-\log(p))^\alpha)$$

where  $\alpha$  controls the curvature and  $\beta$  controls the cross over point. See below.

```
% Prelec weighting function
w_prelec = @(p,alpha,beta) exp(-beta .* (-log(p)).^alpha);

% Probability grid (avoid 0 and 1)
p = linspace(1e-6, 1-1e-6, 1000);

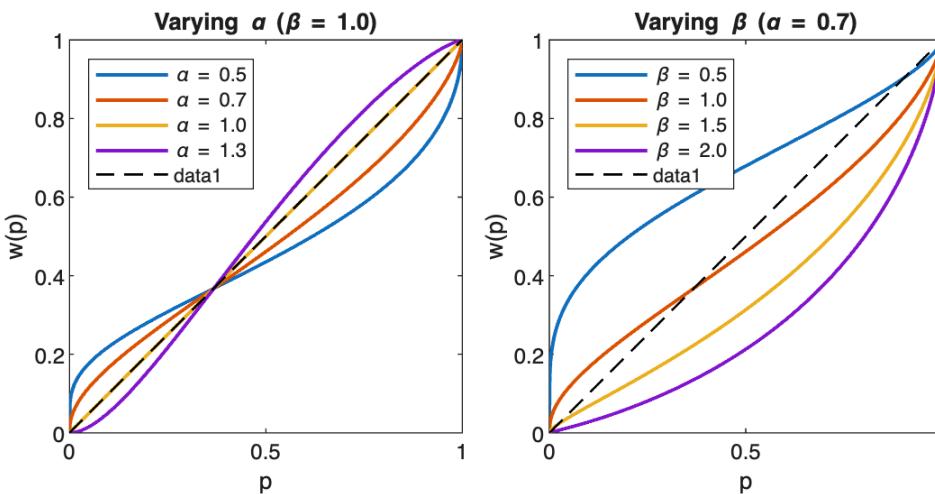
% ----- Parameters -----
beta_const = 1.0;
alpha_vals = [0.5 0.7 1.0 1.3];

alpha_const = 0.7;
beta_vals = [0.5 1.0 1.5 2.0];

% ----- Figure -----
figure;
tiledlayout(1,2, 'Padding', 'compact', 'TileSpacing', 'compact');

% ----- Left panel: vary alpha -----
nexttile;
hold on;
for a = alpha_vals
    plot(p, w_prelec(p,a,beta_const), 'LineWidth', 1.5, ...
        'DisplayName', sprintf('"\alpha = %.1f"', a));
end
plot(p,p,'k--','LineWidth',1); % identity
xlabel('p');
ylabel('w(p)');
title(sprintf('Varying "\alpha" ("\beta = %.1f")', beta_const));
legend('Location','northwest');
axis square;
box on;

% ----- Right panel: vary beta -----
nexttile;
hold on;
for b = beta_vals
    plot(p, w_prelec(p,alpha_const,b), 'LineWidth', 1.5, ...
        'DisplayName', sprintf('"\beta = %.1f"', b));
end
plot(p,p,'k--','LineWidth',1); % identity
xlabel('p');
ylabel('w(p)');
title(sprintf('Varying "\beta" ("\alpha = %.1f")', alpha_const));
legend('Location','northwest');
axis square;
box on;
```



For our purposes, we probably want to modify this function so that the function crosses the identity line at .5,  $w = .5$  when  $p = .5$ . This would allow us to capture the general idea of miscalibration with participants being either over or underconfident. To do this, we can solve the 2 parameter prelec function so that  $w(.5) = .5$ .

.5 =  $\exp(-\beta(-\log(p))^{\alpha})$

$\beta = (\log(2))^{(1-\alpha)}$

Note log is a natural log.

So that

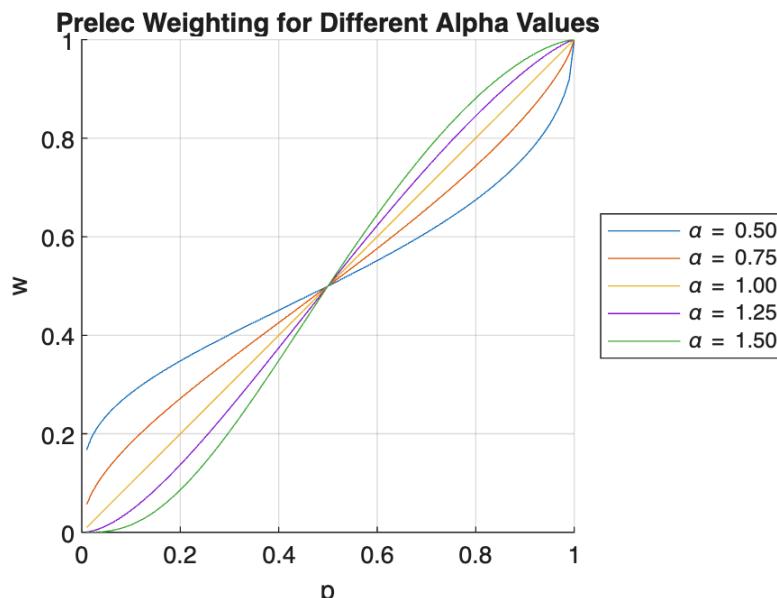
$w = \exp[-(\log(2))^{(1-\alpha)} \times (-\log(p))^{\alpha}]$

Here is the function.

```
function w = w_prelec_fixed_half(p, alpha)
% Prelec weighting with fixed point w(0.5)=0.5
% Single free parameter alpha
if nargin < 2, alpha = 0.7; end
epsv = 1e-12;
p = min(max(p, epsv), 1-epsv);
beta = ((log(2))^(1-alpha));
w = exp(-beta .* (-log(p)).^alpha);
end
```

Visualize our modified prelec function for different alpha (i.e., miscalibration) parameters:

```
p = .01:.01:1;
alpha = [.5,.75,1.0,1.25,1.5];
figure;
hold on;
for i = 1:length(alpha)
    w_values = w_prelec_fixed_half(p, alpha(i));
    plot(p, w_values, 'DisplayName', sprintf('\alpha = %.2f', alpha(i)));
end
axis square
hold off;
xlabel('p');
ylabel('w');
title('Prelec Weighting for Different Alpha Values');
legend show;
legend('Location', 'eastoutside');
grid on;
```



When  $\alpha > 1$ , the agent is overconfident in old/new recognition confidence judgments.

When  $\alpha = 1$  the agent is perfectly calibrated.

When  $\alpha < 1$  the agent is underconfident in old/new recognition confidence judgments.

## Dyad Models

1. Replace UW Model with two agents that are overconfident  $\alpha > 1$ . Note it may be necessary to sweep across different levels of  $\alpha$  to get a feel for predictions.
2. Replace UW Model with two agents that are underconfident  $\alpha < 1$ .
3. Replace Defer-to-Max with two agents that are overconfident.
4. Replace Defer-to-Max with two agents that are underconfident.

In the 4 models above we can examine how varying encoding ability between 2 agents impacts dyadic performance.

But, there is another dimension we can also examine: how does different levels of miscalibration impact dyadic performance. So what happens when we set agent A at  $\alpha_A = 1.2$ , but then sweep across different values of  $\alpha$  for B?