

Contents

- [Step 1: Import CSV Data](#)
- [Step 2: R Bridge Implementation](#)
- [Step 3: MDTF Formulation to Calculate Preference Dynamics](#)
- [Helper Functions](#)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Main Simulation Script for BoundingOverwatch Project with R Integration
% Randomly 10 trials is used to validate predictions
% DFT Parameters:
% phi1 - sensitivity to attribute differences (typically 0.5-2)
% phi2 - memory/feedback strength (0-1)
% tau - decision time steps (integer > 0)
% error_sd - noise standard deviation (sigma_epsilon)
% beta - attribute weights from R estimation
% w - attention weights (default [0.5;0.5])
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clc;
clear all;
```

Step 1: Import CSV Data

(reference apolloMain_5 amd apolloMain_6 as example for data manipulation) biasData = readtable('user_choices.csv'); % Replace with the path to your data file disp('User bias data imported successfully.');

taskChoice_Data = readtable('user_choices.csv'); % Replace with the path to your data file disp('User task choice data imported successfully.');

```
robotChoice_Data = readtable('G:\My Drive\myResearch\Research Experimentation\Apollo\apollo\data\Bounding_Overwatch_Data\FileRescue\HumanData_Bounding_Overwatch_FIX
% Convert all column headers to lowercase
robotChoice_Data.Properties.VariableNames = lower(robotChoice_Data.Properties.VariableNames);
disp('User robot choice data imported successfully.');
```

% Randomly select 10 rows (or all rows if fewer than 10)

```
numRows = height(robotChoice_Data);
randomIndices = randperm(numRows, min(10, numRows));
robotChoice_Data = robotChoice_Data(randomIndices, :);
```

% Extract robot state attributes dynamically

```
robot_states = struct();
attributeSuffixes = {'traversability', 'visibility'}; % No leading underscores
for i = 1:3
    for attr = attributeSuffixes
        csvColName = sprintf('robot%d_%s', i, attr{1}); % Matches CSV column names
        structFieldName = attr{1}; % Valid field name
        if ismember(csvColName, robotChoice_Data.Properties.VariableNames)
            robot_states.([ 'robot' num2str(i) ]). (structFieldName) = robotChoice_Data.(csvColName);
        else
            warning('Missing attribute column: %s', csvColName);
            robot_states.([ 'robot' num2str(i) ]). (structFieldName) = NaN(height(robotChoice_Data), 1);
        end
    end
end
end
```

% Extract choice data and other metadata

```
choices = robotChoice_Data.choice;
participant_ids = robotChoice_Data.id;
stake_types = robotChoice_Data.stakes;
time_spent = robotChoice_Data.timeelapsed;
```

User robot choice data imported successfully.

Step 2: R Bridge Implementation

```
disp('Initializing R bridge...');
```

% Configure paths

```
rscript_path = 'C:\Program Files\R\R-4.4.2\bin\x64\Rscript.exe';
r_script = 'G:\My Drive\myResearch\Research Experimentation\Apollo\apollo\example\DFT_Bounding_Overwatch.R';
csvFile = 'G:\My Drive\myResearch\Research Experimentation\Apollo\apollo\data\Bounding_Overwatch_Data\FileRescue\HumanData_Bounding_Overwatch_FIXED.csv';
outputDir = 'G:\My Drive\myResearch\Research Experimentation\Apollo\apollo\Output_BoundingOverwatch';
```

% Verify installations

```
if ~isfile(rscript_path)
    error('Rscript.exe not found at: %s', rscript_path);
elseif ~isfile(r_script)
    error('R script not found at: %s', r_script);
elseif ~isfile(csvFile)
    error('Input CSV not found at: %s', csvFile);
elseif ~isfolder(outputDir)
    warning('Output folder does not exist, creating: %s', outputDir);
    mkdir(outputDir);
```

```

end

% Execute R with JSON output
try
    % Use proper argument formatting
    cmd = sprintf(['"%s" "%s" ', ...
        '-i "%s" -o "%s"', ...
        rscript_path, r_script, csvFile, outputDir]);

    [status,result] = system(cmd);

    if status == 0
        % Handle output path (whether directory or file)
        if isfolder(outputDir)
            jsonFile = fullfile(outputDir, 'DFT_output.json');
        else
            jsonFile = outputDir;
        end

        % Parse JSON output
        if exist(jsonFile, 'file')
            jsonText = fileread(jsonFile);
            params = jsondecode(jsonText);

            % Extract parameters with validation
            %Boundedphi1, phi2 parameters
            phi1 = max(0, validateParam(params, 'phi1', 0.5)); % Ensure non-negative
            phi2 = min(max(0, validateParam(params, 'phi2', 0.8)), 1); % Constrain 0-1

            %Raw phi1, phi2 parameters
            %phi1 = validateParam(params, 'phi1', 0.5);
            %phi2 = validateParam(params, 'phi2', 0.8);
            tau = 1 + exp(validateParam(params, 'timesteps', 0.5));
            error_sd = min(max(0.1, validateParam(params, 'error_sd', 0.1)), 1); % still clip here

            % Extract attribute weights
            beta_weights = [
                params.b_attr1;
                params.b_attr2;
                params.b_attr3;
                params.b_attr4
            ];

            % Get initial preferences from ASCs
            initial_P = [
                validateParam(params, 'asc_1', 0);
                validateParam(params, 'asc_2', 0);
                validateParam(params, 'asc_3', 0);
            ];

            disp('Estimated Parameters:');
            disp(['phi1: ', num2str(phi1)]);
            disp(['phi2: ', num2str(phi2)]);
            disp(['tau: ', num2str(tau)]);
            disp(['error_sd: ', num2str(error_sd)]);
            disp('Initial Preferences (from ASCs):');
            disp(initial_P);
        else
            error('R output file not found');
        end
    else
        error('R execution failed: %s', result);
    end
catch ME
    disp('Error during R execution:');
    disp(getReport(ME, 'extended'));
    [phi1, phi2, tau, error_sd] = getFallbackParams();
    beta_weights = [0.3; 0.2; 0.4; 0.5]; % Default weights
    initial_P = zeros(3,1); % Neutral initial preferences
end

```

Initializing R bridge...

Step 3: MDFT Formulation to Calculate Preference Dynamics

(MDFT calculations based on estimated parameters) Create M matrix from current trial's attributes C11-C14 are consequence attributes for Robot 1 C21-C24 are consequence attributes for Robot 2 C31-C34 are consequence attributes for Robot 3

```

for current_trial = 1:height(robotChoice_Data)
    num_attributes = 4;

    M = [
        robotChoice_Data.c11(current_trial), robotChoice_Data.c12(current_trial), robotChoice_Data.c13(current_trial), robotChoice_Data.c14(current_trial);
        robotChoice_Data.c21(current_trial), robotChoice_Data.c22(current_trial), robotChoice_Data.c23(current_trial), robotChoice_Data.c24(current_trial);

```

```
robotChoice_Data.c31(current_trial), robotChoice_Data.c32(current_trial), robotChoice_Data.c33(current_trial), robotChoice_Data.c34(current_trial)
];

M = M / 100;          % Convert all values from 0-100 to 0.00-1.00 scale

attributes = {'C1 - Easy Nav, Low Exposure', 'C2 - Hard Nav, Low Exposure', 'C3 - Easy Nav, High Exposure', 'C4 - Hard Nav, High Exposure'};
beta = beta_weights ./ sum(abs(beta_weights));
beta = beta';

[E_P, V_P, choice_probs, P_tau] = calculateDFTdynamics(...
    phi1, phi2, tau, error_sd, beta, M, initial_P);

% Display results for the trial
disp('=== Trial Analysis ===');
disp(['Trial: ', num2str(current_trial)]);
disp(['Participant: ', num2str(participant_ids(current_trial))]);
disp(['Actual Choice: Robot ', num2str(choices(current_trial))]);

disp('M matrix (alternatives x attributes):');
disp(array2table(M, ...
    'RowNames', {'Robot1','Robot2','Robot3'}, ...
    'VariableNames', attributes));

disp('DFT Results:');
disp(['E_P: ', num2str(E_P), '%.2f ']);
disp(['Choice probabilities: ', num2str(choice_probs), '%.3f ']);
[~, predicted_choice] = max(choice_probs);
disp(['Predicted choice: Robot ', num2str(predicted_choice)]);
disp(['Actual choice: Robot ', num2str(choices(current_trial))]);
disp(' ');

if predicted_choice == choices(current_trial)
    disp('✓ Prediction matches actual choice');
else
    disp('X Prediction differs from actual choice');
end

% Plot evolution
figure;
plot(0:tau, P_tau);
xlabel('Preference Step (\tau)');
ylabel('Preference Strength');
legend({'Robot1','Robot2','Robot3'});
title(sprintf('Preference Evolution (Trial %d)', current_trial));
grid on;
end

%{
%% Step 4: Output Results
disp('Saving results to CSV...');
output_table = table(E_P, V_P, P_tau(end,:), ...
    'VariableNames', {'ExpectedPreference', 'VariancePreference', 'FinalPreferences'});
writetable(output_table, 'results.csv');
disp('Results saved successfully!');
%}
```

```
=== Trial Analysis ===
Trial: 1
Participant: 181700
Actual Choice: Robot 3
M matrix (alternatives x attributes):
```

	C1 - Easy Nav, Low Exposure	C2 - Hard Nav, Low Exposure	C3 - Easy Nav, High Exposure	C4 - Hard Nav, High Exposure
Robot1	0.00904	0.00585	0.0041	0.0009
Robot2	0.01117	0.00685	0.00544	0.00112
Robot3	0.01182	0.00773	0.00528	0.00118

```
DFT Results:
E_P: 3.19 -1.57 -1.62
Choice probabilities: 0.984 0.008 0.008
Predicted choice: Robot 1
Actual choice: Robot 3

X Prediction differs from actual choice
=== Trial Analysis ===
Trial: 2
Participant: 181700
Actual Choice: Robot 1
M matrix (alternatives x attributes):
```

	C1 - Easy Nav, Low Exposure	C2 - Hard Nav, Low Exposure	C3 - Easy Nav, High Exposure	C4 - Hard Nav, High Exposure
Robot1	0.0168	0.00974	0.00874	0.00168
Robot2	0.02141	0.01278	0.01078	0.00214
Robot3	0.01873	0.01054	0.01006	0.00187

DFT Results:

E_P: 4.11 -4.98 0.87
Choice probabilities: 0.962 0.000 0.038
Predicted choice: Robot 1
Actual choice: Robot 1

✓ Prediction matches actual choice

=== Trial Analysis ===

Trial: 3

Participant: 181700

Actual Choice: Robot 3

M matrix (alternatives × attributes):

	C1 - Easy Nav, Low Exposure	C2 - Hard Nav, Low Exposure	C3 - Easy Nav, High Exposure	C4 - Hard Nav, High Exposure
Robot1	0.01114	0.00682	0.00544	0.00111
Robot2	0.01687	0.0112	0.00736	0.00169
Robot3	0.00666	0.00458	0.00274	0.00067

DFT Results:

E_P: 1.28 -9.86 8.57
Choice probabilities: 0.001 0.000 0.999
Predicted choice: Robot 3
Actual choice: Robot 3

✓ Prediction matches actual choice

=== Trial Analysis ===

Trial: 4

Participant: 214504

Actual Choice: Robot 2

M matrix (alternatives × attributes):

	C1 - Easy Nav, Low Exposure	C2 - Hard Nav, Low Exposure	C3 - Easy Nav, High Exposure	C4 - Hard Nav, High Exposure
Robot1	0.0079	0.00477	0.00391	0.00079
Robot2	0.01316	0.00868	0.00579	0.00132
Robot3	0.01295	0.00803	0.00622	0.0013

DFT Results:

E_P: 6.30 -3.99 -2.31
Choice probabilities: 1.000 0.000 0.000
Predicted choice: Robot 1
Actual choice: Robot 2

X Prediction differs from actual choice

=== Trial Analysis ===

Trial: 5

Participant: 214504

Actual Choice: Robot 1

M matrix (alternatives × attributes):

	C1 - Easy Nav, Low Exposure	C2 - Hard Nav, Low Exposure	C3 - Easy Nav, High Exposure	C4 - Hard Nav, High Exposure
Robot1	0.0074	0.00427	0.00387	0.00074
Robot2	0.00713	0.00423	0.00362	0.00071
Robot3	0.01182	0.00779	0.00522	0.00118

DFT Results:

E_P: 2.88 1.98 -4.86
Choice probabilities: 0.712 0.288 0.000
Predicted choice: Robot 1
Actual choice: Robot 1

✓ Prediction matches actual choice

=== Trial Analysis ===

Trial: 6

Participant: 181700

Actual Choice: Robot 1

M matrix (alternatives × attributes):

	C1 - Easy Nav, Low Exposure	C2 - Hard Nav, Low Exposure	C3 - Easy Nav, High Exposure	C4 - Hard Nav, High Exposure
Robot1	0.0205	0.01187	0.01069	0.00205
Robot2	0.02545	0.01479	0.01321	0.00255
Robot3	0.0201	0.01101	0.01109	0.00201

DFT Results:

E_P: 2.93 -6.61 3.68
Choice probabilities: 0.323 0.000 0.677
Predicted choice: Robot 3
Actual choice: Robot 1

X Prediction differs from actual choice

=== Trial Analysis ===

Trial: 7

Participant: 181700

Actual Choice: Robot 1

M matrix (alternatives × attributes):

C1 - Easy Nav, Low Exposure

C2 - Hard Nav, Low Exposure

C3 - Easy Nav, High Exposure

C4 - Hard Nav, High Exposure

Robot1	0.01532	0.00824	0.00861	0.00153
Robot2	0.01597	0.00867	0.0089	0.0016
Robot3	0.01789	0.01104	0.00864	0.00179

DFT Results:

E_P: 2.38 -0.02 -2.36

Choice probabilities: 0.909 0.083 0.008

Predicted choice: Robot 1

Actual choice: Robot 1

✓ Prediction matches actual choice

=== Trial Analysis ===

Trial: 8

Participant: 214504

Actual Choice: Robot 3

M matrix (alternatives × attributes):

C1 - Easy Nav, Low Exposure

C2 - Hard Nav, Low Exposure

C3 - Easy Nav, High Exposure

C4 - Hard Nav, High Exposure

Robot1	0.01168	0.00689	0.00596	0.00117
Robot2	0.01106	0.00694	0.00522	0.00111
Robot3	0.0113	0.00651	0.00592	0.00113

DFT Results:

E_P: -0.06 -0.44 0.50

Choice probabilities: 0.291 0.199 0.510

Predicted choice: Robot 3

Actual choice: Robot 3

✓ Prediction matches actual choice

=== Trial Analysis ===

Trial: 9

Participant: 181700

Actual Choice: Robot 3

M matrix (alternatives × attributes):

C1 - Easy Nav, Low Exposure

C2 - Hard Nav, Low Exposure

C3 - Easy Nav, High Exposure

C4 - Hard Nav, High Exposure

Robot1	0.02599	0.01525	0.01334	0.0026
Robot2	0.02289	0.01237	0.01281	0.00229
Robot3	0.01694	0.0089	0.00974	0.00169

DFT Results:

E_P: -6.47 -2.31 8.79

Choice probabilities: 0.000 0.000 1.000

Predicted choice: Robot 3

Actual choice: Robot 3

✓ Prediction matches actual choice

=== Trial Analysis ===

Trial: 10

Participant: 181700

Actual Choice: Robot 3

M matrix (alternatives × attributes):

C1 - Easy Nav, Low Exposure

C2 - Hard Nav, Low Exposure

C3 - Easy Nav, High Exposure

C4 - Hard Nav, High Exposure

Robot1	0.01988	0.01075	0.01111	0.00199
Robot2	0.02631	0.01583	0.01311	0.00263
Robot3	0.01479	0.00791	0.00835	0.00148

DFT Results:

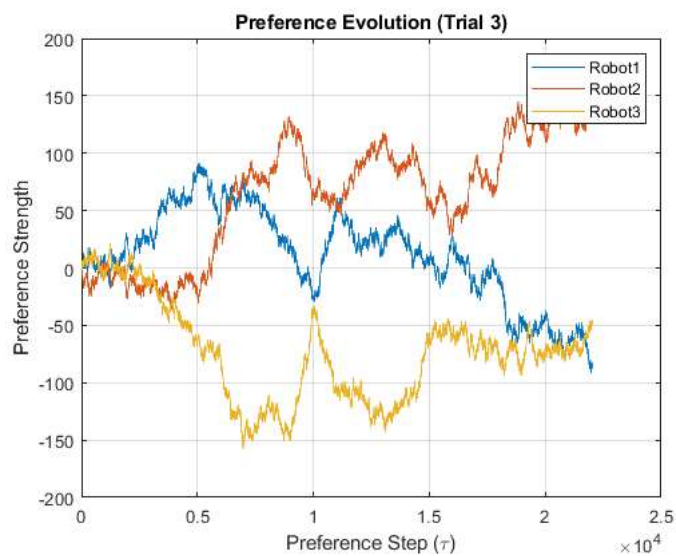
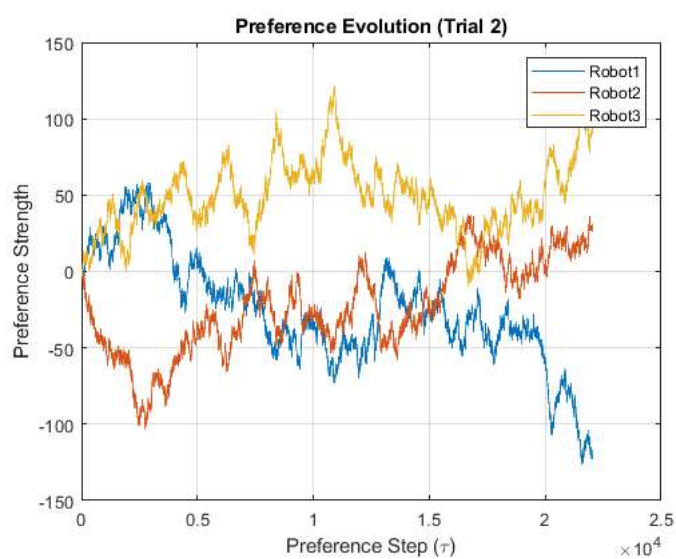
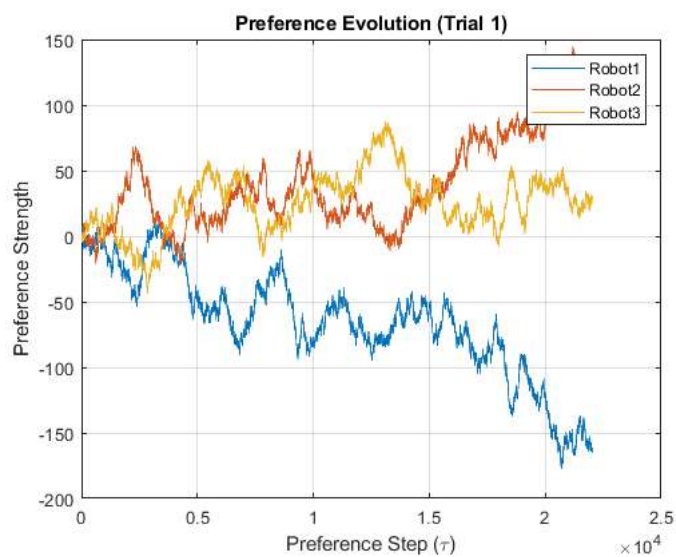
E_P: 1.36 -11.07 9.71

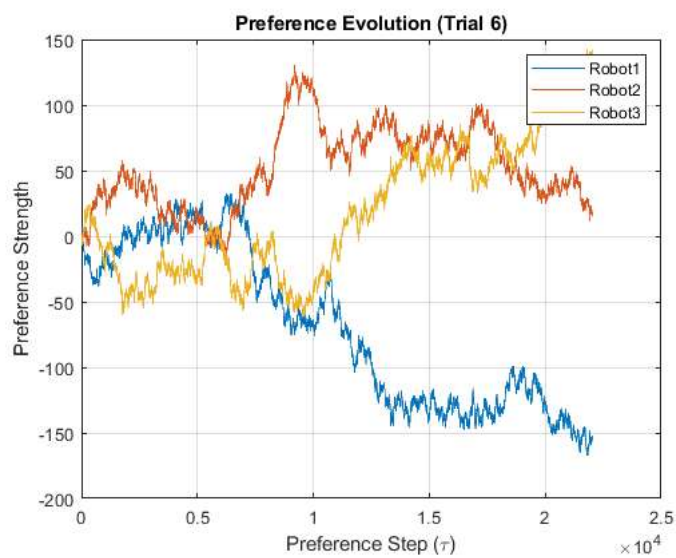
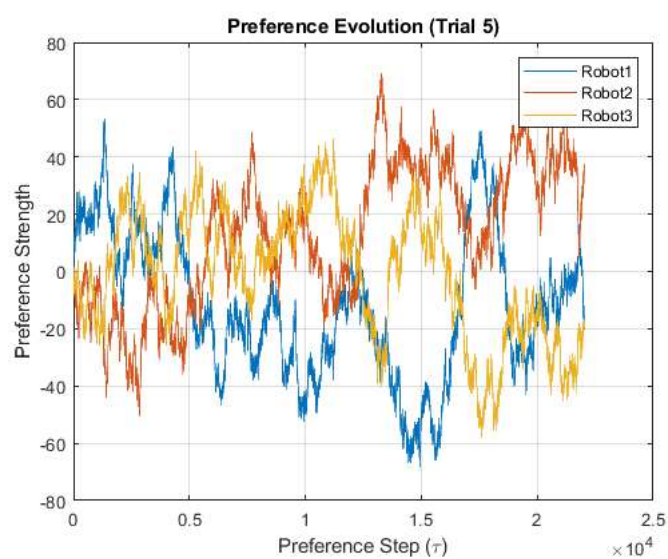
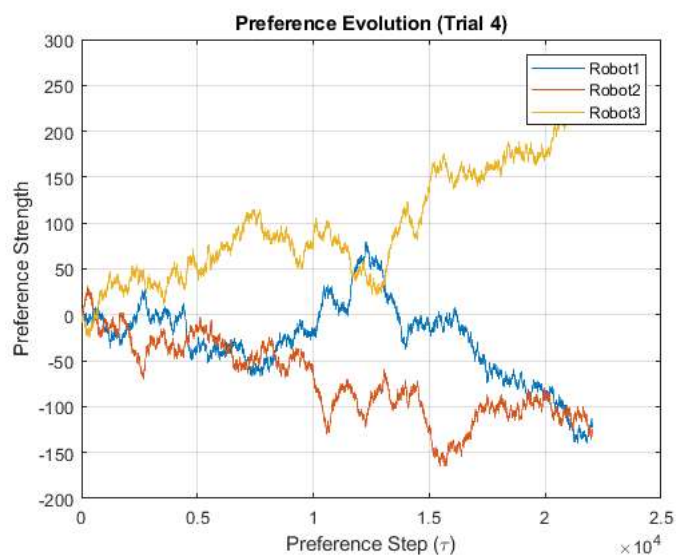
Choice probabilities: 0.000 0.000 1.000

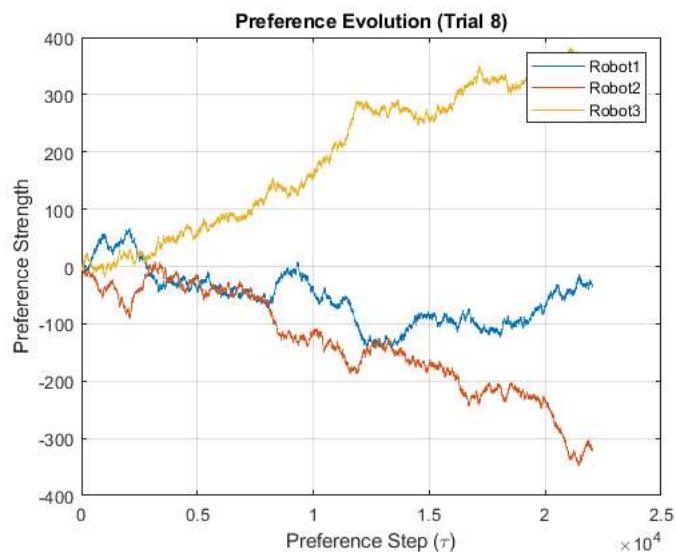
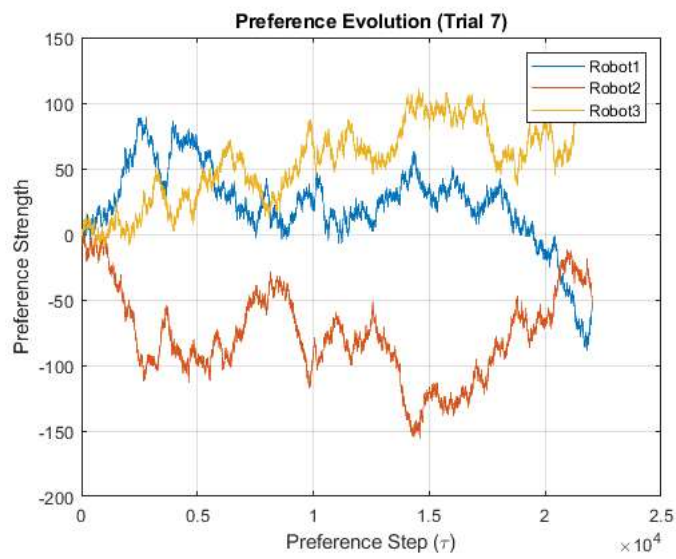
Predicted choice: Robot 3

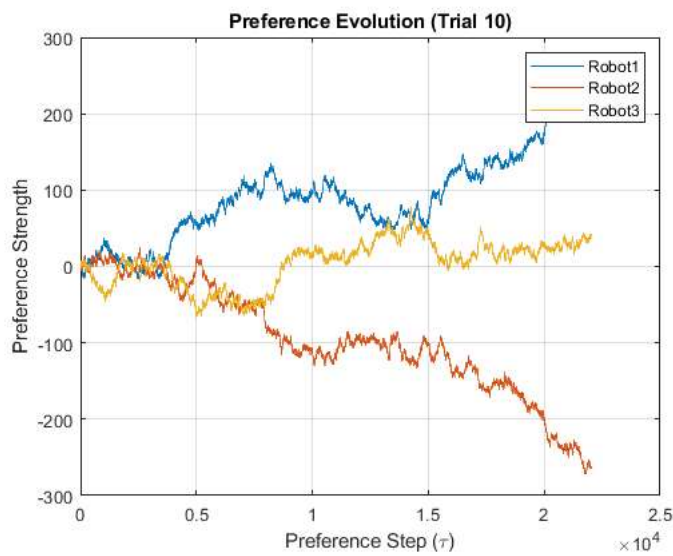
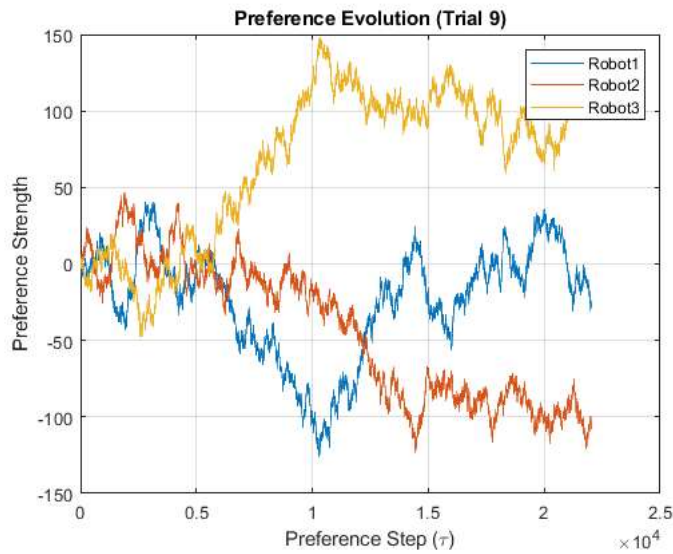
Actual choice: Robot 3

✓ Prediction matches actual choice









Helper Functions

```
function param = validateParam(params, name, default)
    if isfield(params, name) && isnumeric(params.(name))
        param = params.(name);
    else
        warning('Using default for %s', name);
        param = default;
    end
end

function [phi1, phi2, tau, error_sd] = getFallbackParams()
    phi1 = 0.5 + 0.1*randn();
    phi2 = 0.8 + 0.1*randn();
    tau = 10 + randi(5);
    error_sd = 0.1 + 0.05*rand();
    warning('Using randomized default parameters');
end
```

Estimated Parameters:

phi1: -0.2493

phi2: 0.1799

tau: 22027.4658

error_sd: 1

Initial Preferences (from ASCs):

0.1118 -1.1965 0

