HELLO WORLD: PROGRAMAR ESTUDOS CLÁSSICOS DE PSICOLOGIA COM MATLAB

INÊS MANUEL BRITO[1], SOFIA FRADE[1] & RADJON RODRIGUES HAQUE[2]
[1]Iscte-Instituto Universitário de Lisboa, [2]Instituto Piaget

Palavras-chave: MATLAB; Programação; Estudos clássicos em Psicologia; *Stroop task*.

OBJETIVO

Este capítulo visa:

(a) Apresentar a plataforma MATLAB, como também, a sua aplicabilidade em investigação em Psicologia;

(b) Apresentar noções básicas de MATLAB;

(c) Demonstrar como ler um *script* de um estudo clássico de Psicologia;

(d) Destacar recursos e suporte disponíveis tanto dentro da aplicação como online, de modo a auxiliar os utilizadores na programação e resolução de problemas.

INTRODUÇÃO

MATLAB é uma plataforma de programação concebida para engenheiros e cientistas, que lhes permite analisar e desenvolver sistemas e produtos. Através da sua linguagem baseada em matrizes (i.e., conjuntos retangulares de números), MATLAB facilita a expressão intuitiva da matemática computacional (The MathWorks Inc., 1994-2024).

Em comparação com as outras linguagens de programação tradicionais (e.g., C/C+, Java), MATLAB é relativamente fácil de utilizar, com diversos gráficos incorporados que facilitam a visualização e o acesso às informações sobre os dados. Por ser uma plataforma tão utilizada, há muito apoio disponível tanto dentro da aplicação (e.g., por meio do botão *Help* disponível no *Toolstrip*) como *online* (e.g., no website oficial do MATLAB). Apesar de ser uma ferramenta paga, os alunos/as e os/as investigadores/as conseguem podem ter acesso através de uma licença do Iscte (para mais informações de como descarregar o MATLAB e como ativar a licença, consultar o seguinte link).

MATLAB em Investigação em Psicologia

Para a investigação em Psicologia, MATLAB tem diversas utilizações, entre elas:

Desenvolvimento de experiências, com a sua execução, controlo de estímulos apresentados aos/às participantes, recolha de respostas e registo de dados. Estas ações são, geralmente, realizadas através de um *script* (i.e., um ficheiro que contém várias linhas sequenciais de comandos e funções) programado pelo investigador/a. Neste capítulo, iremos aprofundar estas questões.

Visualização de dados, com a criação de gráficos e visualizações interativas para apresentar os resultados da investigação de forma clara e compreensível.

Análise de dados experimentais (e.g., estatísticas descritivas, ANOVA).

Processamento de sinais e imagens, em que EEG, EMG e ECG podem filtrar, segmentar, extrair características e analisar padrões.

Quando não temos muito conhecimento em MATLAB, é mais fácil começarmos a **programar com um código existente**. Deste modo, para programar estudos clássicos de Psicologia, uma vez que são estudos bastante conhecidos, há uma maior probabilidade de haver códigos publicados por outros programadores em diversos websites como, por exemplo, o website oficial do MATLAB, que permite partilhar e descarregar o nosso código para ajudar outros; e o GitHub, uma plataforma que possibilita a contribuição em projetos privados e/ou código aberto.

Coding Style

Caso tenham alguma experiência prévia em programação, devem ter conhecimento dos diferentes *coding styles*, como também, das **boas práticas de codificação**. Cada pessoa tem a sua própria forma de escrever, e é sugerido que se tente ter o código mais sucinto e direto possível. Contudo o código mais bonito nem sempre é o melhor, e o mais importante é que seja funcional. Adicionalmente, é sempre um bónus se passarmos menos tempo a escrever o nosso código.

Relativamente às boas práticas de codificação, é essencial o uso de comentários – sinalizados através de "%" (que se encontram a verde) – que expliquem segmentos do código. Deste modo, conseguimos garantir que da próxima vez que estivermos a ler o nosso *script* sabemos o que cada parte do código está a realizar, e que outras pessoas que possam vir a utilizá-lo são capazes de compreender. Outra questão que deve ser acautelada é que o nosso *script* possa ser utilizado por outro computador.

A Figura 1 apresenta duas formas de escrever o mesmo código, sendo que o modo mais aconselhado é o A, que é mais fácil de ler, perceber e corrigir eventuais erros.

Figura 1 | Exemplos de Diferentes Formas de Escrever um Código

A

```
function reviewSubjects(subjectList)
%Function to go through each subject in a list and perform review

    for i = 1:length(subjectList)

        fprintf('Reviewing subject %d',subjectList(i));
        x = reviewData(subjectList(i));

        if x
            fprintf('Review successful\n');
        else
            fprintf('Review unsuccessful\n');
        end

    end

return
```

B

```
function reviewSubjects(subjectList)
%Function to go through each subject in a list and perform review

for i = 1:length(subjectList)

fprintf('Reviewing subject %d',subjectList(i));
x = reviewData(subjectList(i));

if x
fprintf('Review successful\n');
else
fprintf('Review unsuccessful\n');
end

end

return
```

NOÇÕES BÁSICAS DE MATLAB

Quando abrimos a aplicação do MATLAB (ver Figura 2), temos:

Toolstrip localizado na parte superior da janela, que contém ícones para fácil acesso a diversas funções e ferramentas (e.g., abrir arquivos, executar *scripts*, criar gráficos).

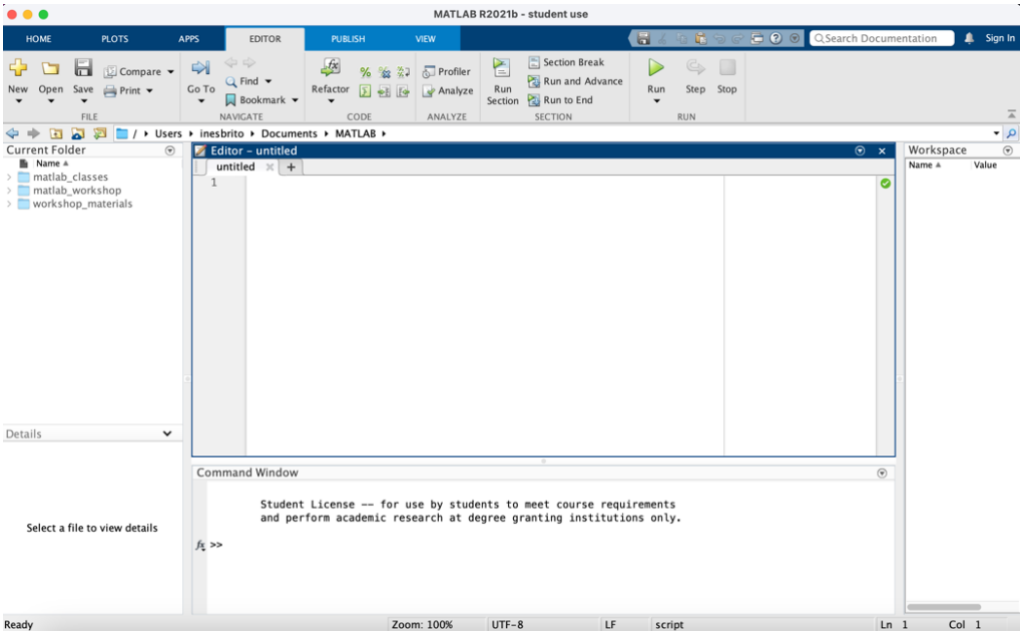
Current Folder, tal como o nome indica, mostra os arquivos e pastas atuais, podendo navegar entre pastas e aceder a arquivos diretamente.

Command Window, onde podemos escrever comandos diretamente e ver os resultados imediatamente, como também, executar cálculos simples, testar funções e pedir ajuda. Esta janela indica-nos os erros do nosso código.

Workspace exibe as variáveis atualmente definidas no nosso *script*, juntamente com os seus valores.

Editor permite escrever, editar e salvar *scripts* de MATLAB.

FIGURA 2 | ABERTURA DA APLICAÇÃO MATLAB



Um **comando** (i.e., *command*) consiste numa diretiva ou instrução utilizada para executar uma ação específica. Essas instruções podem abranger desde operações aritméticas básicas (e.g., +) até manipulações complexas de dados e funções (e.g., *randi* é um comando que cria números aleatórios ou inteiros uniformemente distribuídos). Uma lista de referência das funções básicas do MATLAB pode ser consultada aqui.

Todas as **variáveis** presentes na memória atual são exibidas no *Workspace*, e atribui-se valores às variáveis (e.g., x = 1), como também, operações matemáticas podem ser realizadas com essas variáveis (e.g., y = 2+2). Neste caso, no *Workspace* teríamos duas variáveis diferentes *x* e *y* de valor 1 e 4, correspondente. O MATLAB é bastante sensível a maiúsculas e minúsculas, pelo que *A* e *a* não são a mesma variável. Deste modo, *myvar*, *myVar* e *my_var* representam variáveis distintas.

Para testar se uma condição é verdadeira ou não, utilizámos **operadores lógicos** (ver Tabela 1), em que 0 significa falso e 1 verdadeiro.

TABELA 1 | OPERADORES LÓGICOS BÁSICOS

==	Igual a (diferente de = que é utilizado para definir uma variável)
~=	Não igual a
>	Maior que
<	Menor que
>=	Maior ou igual a
<=	Menor ou igual a
&	E
|	Ou
~	Não

Uma **função** é um segmento de código autónomo que executa uma função específica, isto é refere-se a blocos encapsulados de código que operam em variáveis nos seus próprios espaços de trabalho independentes. De salientar, que todas as variáveis definidas numa função apenas existem dentro dessa própria função, sendo preferível a definição das variáveis no início do *script*.

Estruturas de Repetição

If Conditionals permite executar blocos de código com base em condições específicas. Assim sendo, o bloco de código dentro do "if" apenas é executado se a condição for verdadeira e, caso a condição for falsa e houver um "else", o bloco de código dentro do "else" é executado.

```
if (condicao)
        bloco de código a ser executado se a condição for verdadeira
else
        bloco de código a ser executado se a condição for falsa
end
```

For Loop é uma estrutura que possibilita a repetição de um bloco de código por um número específico de vezes. A variável "i" é utilizada como variável de iteração, e definimos um intervalo de valores (de "inicial" a "final"), sendo que o bloco de código dentro do "for" é executado para cada valor da variável de iteração, com a variável assumindo valores sequencialmente dentro do intervalo especificado.

```
for i = inicial:final
        bloco de código a ser repetido
end
```

While Loop repete um bloco de código enquanto a condição for verdadeira, sendo importante garantir que a condição eventualmente se torne falsa para evitar um loop infinito. Caso isso não aconteça, temos de usar o comando "break".

```
while (condicao)
        bloco de código a ser repetido
end
```

Estudo Clássico em Psicologia: Stroop Task

O ***Stroop Task*** é um fenómeno nomeado após John Ridley Stroop (1935), que mostra que quando existe incongruência entre o nome de uma cor e a cor em que a palavra está impressa (e.g., a

palavra "vermelho" impressa a azul), os/as participantes demoram mais tempo a nomear a cor corretamente (i.e., azul), e tendem a cometer mais erros do que quando existe congruência (e.g., a palavra "vermelho" impressa a vermelho).

EXEMPLO DO *STROOP TASK*

Há aspetos fundamentais na programação desta tarefa experimental, que iremos explicar com mais detalhe. Podem utilizar o seguinte exemplo para se familiarizarem com a plataforma. O *script* completo do *Stroop Task* pode ser consultado aqui.

É importante que a nossa experiência tenha um *slide* inicial com as **instruções** para os/as participantes. Estas têm que ser claras e concisas, de modo a garantir a sua fácil compreensão. Antes de apresentarmos as instruções, é necessário configurar as propriedades da figura onde estas serão exibidas.

```
% Stroop Task with Instructions and Responses on a Black Screen
clc; % Clear command window
clear; % Clear workspace
close all; % Close all figures

% Instructions for the task as a cell array of strings
instructions = {'Welcome to the Stroop Ta  sk!', ...
'In this task, you will see color names written in various colors.', ...
'Your task is to identify the color of the ink, ignoring the word itself.', ...
'For example, if you see the word "red" written in green ink, you should press r for red.', ...
    'Press r for red, g for green, and b for blue.', ...
    'You will have 1 second to respond to each stimulus.', ...
    'Press any key to begin...'};

% Set up figure properties
figure('Color', 'black'); % Create a figure with black background
set(gcf, 'units','normalized','outerposition',[0 0 1 1]); % Maximize figure window; gcf
stands for 'get current figure'. This specifies the units used for setting the figure's
position. This is often used to make figures independent of screen resolution or size.
This specifies the position and size of the figure relative to its parent container (usually
the screen or the desktop). The values [0 0 1 1] mean that the figure will be positioned at
the bottom-left corner of its parent container ([0 0]) and will occupy the full width and
height of the container ([1 1])
axis off; % Turn off axis
```

```
% Clear figure before displaying instructions
clf;
set(gcf, 'Color', 'black'); % Set black background
axis off; % Turn off axis

% Display instructions
for i = 1:numel(instructions) % numel() = returns the number of elements in an array or
the total number of elements in an object's contents
        text(0.5, 1 - i * 0.1, instructions{i}, 'Color', 'white', 'FontSize', 14,
        'HorizontalAlignment', 'center');
end
drawnow; % Force immediate rendering
pause; % Wait for any key press to continue
```

De seguida, temos que **definir as condições**, através da criação de diversas variáveis, que armanezem as respostas dos/as participantes, os seus tempos de reação, o *feedback*, como também, a palavra e a cor apresentada.

```
% Define colors and their corresponding names
colors = {'r', 'g', 'b'}; % Colors represented by letters
color_names = {'Red', 'Green', 'Blue'}; % Color names

% Initialize variables
num_trials = 3; % Number of trials
response = cell(1, num_trials); % Store user responses as cell array with a size of 1 row
and num_trials columns
reaction_time = zeros(1, num_trials); % Store reaction times as numeric array with a size
of 1 row and num_trials columns, filled with zeros
feedback = cell(1, num_trials);
word_presented = cell(1, num_trials);
color_presented = cell(1, num_trials);
```

Cada ***trial*** **(ensaio) da nossa experiência** terá uma imagem de um sinal de mais apresentado a branco (i.e., +), a seleção aleatória de uma cor e nome da cor, a apresentação da mesma, o registo do tempo de reação, assim como, o *feedback* da resposta dos/as participantes. Este processo é repetido para cada *trial* que temos – sendo que, neste caso, corresponde a 3 (num_trials = 3) –, i.e., para cada palavra apresentada.

```
% Start the experiment
for trial = 1:num_trials
        % Clear figure
        clf;
        set(gcf, 'Color', 'black'); % Set black background

        % Display the image of the white plus sign
        img = imread('+.jpg'); % Reads the image
        image(img); % Displays the image
        axis off; % Turn off axis
        axis image; % Maintain aspect ratio

        % Wait for a short duration before proceeding
        pause(1); % Adjust the delay as needed

        % Randomly select a color and a color name
        color_idx = randi(numel(colors)); % Determine a random index to select a color
from the list of colors
        color = colors{color_idx}; % Retrieve the color corresponding to the randomly
chosen index
        color_name_idx = randi(numel(color_names)); % Same thing but for the color
name
        color_name = color_names{color_name_idx};

        % Clear figure
        clf;
        set(gcf, 'Color', 'black'); % Set black background

        % Draw a large black rectangle over the entire figure
        rectangle('Position', [0, 0, 1, 1], 'FaceColor', 'black', 'EdgeColor', 'none'); % [0,
0] specify the (x, y) coordinates of the bottom-left corner of the object[1, 1] specify the
width and height of the object. Here, [1, 1] means the object spans the entire width and
height
        axis off; % Turn off axis

        % Display the color name in the selected color
        text(0.5, 0.5, color_name, 'Color', color, 'FontSize', 50, 'HorizontalAlignment',
        'center', 'VerticalAlignment', 'middle');
```

```
drawnow; % Force immediate rendering

% Record start time
start_time = tic; % tic is a function used to start a stopwatch timer. It marks the current time as the starting point for measuring elapsed time

% Prompt user for response
key = getkey(colors); % getkey is a custom function defined later in the code

% Record reaction time
reaction_time(trial) = toc(start_time); % This function call calculates the elapsed time since the stopwatch timer was started at start_time

% Check if response is correct
if key == color % This conditional statement checks if the participant's response (key) matches the color presented (color) for the current trial
feedback{trial} = 'Correct';
else
feedback{trial} = 'Incorrect';
end

% Clear figure
clf;
set(gcf, 'Color', 'black'); % Set black background

% Draw a large black rectangle over the entire figure
rectangle('Position', [0, 0, 1, 1], 'FaceColor', 'black', 'EdgeColor', 'none');
axis off; % Turn off axis

% Display feedback
text(0.5, 0.5, feedback{trial}, 'Color', 'white', 'FontSize', 50,
'HorizontalAlignment', 'center');
drawnow; % Force immediate rendering
pause(1); % Display feedback for 1 seconds

% Store user response, word presented, and color of the word presented
response{trial} = key;
word_presented{trial} = color_name;
```

```
        color_presented{trial} = color;
    end

    % Close figure
    close;
```

O MATLAB não guarda as respostas dos/as participantes automaticamente, sendo necessário criar um código que remete para o **armazenamento dos dados** num ficheiro de Excel que registe todas as respostas, como também, a palavra e a cor da palavra apresentada, o *feedback* e o tempo de reação.

```
    % Save results to Excel
    filename = ['stroop_results_'];

    % Convert data into a table
    results_table = table(word_presented', color_presented', response', feedback',
    reaction_time', ...
    'VariableNames', {'Word_Presented', 'Color_Presented', 'Response', 'Feedback',
    'Reaction_Time'});

    % Write the table to Excel
    writetable(results_table, filename);
```

A **apresentação das respostas, do *feedback* e os tempos de reação aos/às participantes** é opcional e, muitas das vezes, não é algo comum e/ou aconselhável. Apesar disso, de seguida, encontramos o código que remete para a última fase da nossa experiência.

```
    % Display responses, correct and incorrect answers along with reaction times on a black
    screen
    figure('Color', 'black');
    set(gcf, 'units','normalized','outerposition',[0 0 1 1]); % Maximize figure window
    axis off; % Turn off axis

    text(0.5, 0.9, 'Experiment Finished!', 'Color', 'white', 'FontSize', 18, 'HorizontalAlignment',
    'center'); % These are the coordinates where the text will be positioned. The first value
    (0.5) represents the x-coordinate, and the second value (0.9) represents the y-
    coordinate

    text(0.5, 0.8, 'Responses:', 'Color', 'white', 'FontSize', 16, 'HorizontalAlignment', 'center');
```

```
text(0.5, 0.75, sprintf('%s ', response{:}), 'Color', 'white', 'FontSize', 14, 'HorizontalAlignment', 'center') % This part generates a string representing the responses of the participants. response{:} retrieves all elements of the response cell array and sprintf('%s ', ...) formats them into a single string with a space between each response

text(0.5, 0.6, 'Feedback:', 'Color', 'white', 'FontSize', 16, 'HorizontalAlignment', 'center');
text(0.5, 0.55, sprintf('%s ', feedback{:}), 'Color', 'white', 'FontSize', 14, 'HorizontalAlignment', 'center');

text(0.5, 0.4, 'Reaction Times (s):', 'Color', 'white', 'FontSize', 16, 'HorizontalAlignment', 'center');
text(0.5, 0.35, sprintf('%.2f ', reaction_time), 'Color', 'white', 'FontSize', 14, 'HorizontalAlignment', 'center');
% sprintf('%.2f ', ...) formats them into a single string with a space between each response, with 2 decimal places

pause(10); % Display results for 10 seconds before closing
close;

% Function to get key press
function key = getkey(valid_keys) % This line defines a function named getkey that takes one input argument valid_keys. This function will return the pressed key, which is stored in the variable key
    while true % This starts an infinite loop, which will keep running until a valid key is pressed
        w = waitforbuttonpress;
        if w == 1 % It returns 1 if a key was pressed and 0 if a mouse button was clicked.
            key = lower(get(gcf, 'CurrentCharacter')); % This line retrieves the character corresponding to the pressed key and stores it in the variable key. The function get(gcf, 'CurrentCharacter') retrieves the current character typed in the figure window, and lower() converts it to lowercase
            if ismember(key, valid_keys) % This line checks if the pressed key is included in the list of valid keys specified by the input argument valid_keys.
```

```
                    break; % If the pressed key is valid, the loop is
                terminated using the break statement.
            end
        end
    end
end
```

RECURSOS

Procurar Ajuda

Existe muito **apoio disponível tanto diretamente na aplicação MATLAB** como *online* (e.g., *Google*). A ajuda integrada pode ser acedida a partir da barra de ferramentas (Home -> Help), e ajuda para funções específicas basta escrevermos no *Command Window* "*help* função" (e.g., *help disp*). Além disso, existem diversos vídeos de demonstração incorporados – escrevam "demo" o *Command Window*.

Devemos tirar partido de todas as ferramentas que temos ao nosso dispor, especialmente dos ***chatbots* de Inteligência Artificial** (i.e., ChatGPT e Copilot), que têm a capacidade de verificar e corrigir os erros no nosso código, explicar partes do código que não estejamos a compreender, bem como, criar código de raíz. Para tal, é necessário que tenhamos certos cuidados nas nossas instruções (para mais informações, ver o capítulo de Haque et al., neste volume).

Cursos Online

Caso queiram aprender mais sobre MATLAB, sugerimos os seguintes cursos *online*:

. MATLAB Onramp by MathWorks, que ensina as noções básicas do MATLAB com um tutorial introdutório sobre as funcionalidades e fluxos de trabalho mais utilizados, gratuitamente.

. Coursera – Introduction to Programmimg with MATLAB, disponibilizado pela Universidade de Vanderbilt, é um curso grátis (mas para obterer o certificado, é necessário pagar uma taxa) mais completo para principiantes, com a aprendizagem de conceitos fundamentais de programação informática (e.g., variáveis, funções), como trabalhar com matrizes, como lidar com diversos tipos de dados e entre outras coisas.

. Udemy - MATLAB for Engineers | Go from Zero to Hero, oferece um curso pago para principiantes que ensina a programar MATLAB de raíz, resolver problemas difíceis, plotagem de figuras 2D e 3D, entre outras coisas.

. LinkedIn Learning – MATLAB Essential Training, pretende que, através deste curso pago, sejam capazes de utilizar eficazmente MATLAB para análise numérica, modelação e visualização de dados.

. Skillshare – Learn MATLAB Programming, é um curso pago, que tem como objetivo ensinar os comandos e conceitos mais importantes do MATLAB e como executar *scripts*, não cometendo erros.

Considerações Finais

O MATLAB é uma ferramenta muito útil para investigação em Psicologia, podendo, entre outras coisas, desenvolver experiências. Ao programar em MATLAB, é essencial adotar boas práticas de codificação e utilizar estruturas de repetição e condicionais de maneira eficaz. Para iniciantes, existem diversos recursos disponíveis, como cursos *online* gratuitos e pagos, para aprender e desenvolver as habilidades em MATLAB.

Sobre os/as Autores/as

Inês Brito é estudante finalista do Mestrado de Psicologia Social e das Organizações do Iscte-Instituto Universitário de Lisboa. Realizou o seu estágio curricular no LAPSO-Laboratório de Psicologia no ano letivo de 2023/2024, tendo lecionado um workshop sobre a aplicabilidade do MatLab para a investigação em Psicologia.

Sofia Frade é doutorada em Ciência Cognitiva (2021) pela Universidade de Lisboa. É Gestora Científica do LAPSO-Laboratório de Psicologia, Iscte e Investigadora Integrada do Cis_Iscte, sendo membro do grupo de investigação Behavior, Emotion and Cognition (BEC).

Radjon Haque é estudante finalista do Mestrado de Psicologia Social e das Organizações do Instituto Piaget. Realizou o seu estágio curricular no LAPSO-Laboratório de Psicologia no ano letivo de 2023/2024.

Referências

Stroop, J. R. (1935). Studies of interference in serial verbal reactions. *Journal of Experimental Psychology, 18*, 643-662. https://doi.org/10.1037/h0054651

The MathWorks Inc. (1994-2024). *What is MATLAB?* https://www.mathworks.com/discovery/what-is-matlab.html