```matlab
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%  MATLAB Brush Up Course: Session 4  %%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%   by JOAN MARGALEF   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% DATA IMPORT/EXPORT & PLOTTING

% This session delves into data import, export, and plotting in MATLAB.
% It starts with creating different data vectors and demonstrates various
% plotting techniques, including line plots, scatter plots, histograms,
% and bar plots. The session highlights how to customize plots with titles,
% legends, and axis labels, and how to save plots.
% It covers data export in various formats and demonstrates data import
% from external sources. The practical exercises involve handling World
% Bank data, showcasing MATLAB’s strengths in data manipulation,
% statistical analysis and data plotting.


%% 1. Matrices: Create, Plot, Export, and Import

clc;clear

% 1. CREATE

x = [1950:1:1999];
y = [0.1:0.2:10];
z = rand(1,50);

% 2. PLOT

% Plotting x vs. y
plot(x,y);
plot(x,z);  % it replaces previous figure


% One can use the figure command to have different windows
figure;
plot(x,y);

figure;
plot(x,z);

% Plotting x vs. y and x vs. z overlaid
plot(x,y);
hold on     % Retain the current plot when adding new plots
plot(x,z);
hold off;   % Release the hold to allow for other plots to replace this one


% Plotting several graphs same image
subplot(1, 2, 1); % Divide into a 1x2 grid, and use the 1st section
plot(x, y);

subplot(1, 2, 2); % Use the 2nd section of the 1x2 grid
plot(x, z);



% Line features:

% Adding features with customized plot appearance
figure;
plot(x, y, 'b');
hold on
plot(x, z, 'r--', 'LineWidth', 1.5); % Red thicker dashed line
```

```
% Adding features:
figure;
plot(x,y);
hold on     % Retain the current plot when adding new plots
plot(x,z);
hold off;   % Release the hold to allow for other plots to replace this one
title('Plot of x vs. y and z'); % Sets the title of the plot
subtitle('Matlab Course'); % Adds a subtitle below the main title
legend('Linear Data','Random Data'); % Adds a legend
xlabel('Year'); % Labels the x-axis as 'Year'
ylabel('Values'); % Labels the y-axis as 'Random Value'
xlim([1950 1975]); % Sets the x-axis to display from 1950 to 1975
ylim([min(z) max(z)]); % Dynamically sets the y-axis limits based on z


% Save the current figure to the file
saveas(gcf, 'myPlot.png')  % 'gcf': gets the handle to the current figure


% 3. EXPORT

% Formats: (see https://www.mathworks.com/help/matlab/import export/
% supported-le-formats.html)

% .mat  (if no specified also)
% .dat  can be read by other programs

save vars1   % Save all Variables
save vars2 y % Save Specific Elements
save vars3.dat



% 4. IMPORT
clear;
load vars1
A=importdata('vars3.dat') % structure

% A, a structure array, is a data type that groups (typically related)
% data using data containers called fields. Each field can contain any
% type of data (e.g., "double", "string").

% Access:
% The commands A.x and A.y give you the x and y vectors, respectively.

% Input:
% A.z = 'IDEA' adds field z containing string "IDEA" to structure A.

%% Practice 1.

% 1. Create four vectors of different types and lengths (choose freely)
%    - Create a linearly spaced vector 'x' from 1950 to 1999.
%    - Create a vector 'y' that linearly increases from 0 to 10.
%    - Generate a random vector 'z' with values between 0 and 1.
%    - Create a vector 'w' containing Stand. Normal random values
% 2. Adjust them to the same size (take smallest)
% 3. Plot each vector against 'x' in separate figure windows.
% 4. Plot all vectors against 'x' with legends and axis-labels
% 5. Create a 2x2 subplot grid with titles (use only 3 of them)
% 6. Save all vectors to a '.mat' file
% 7. Clear the workspace, then load the saved data.
```

```
%% 2. Different Types of Plots

% 1. Line Plots:
%    - Used for showing trends over time or ordered categories.
%    - Example: Plotting a time series.

x = 1:10; % Sample x data
y = rand(1, 10); % Sample y data, random values
figure;
plot(x, y); % Creating a line plot
title('Example Line Plot');
xlabel('x');
ylabel('Random Value');

% 2. Histograms:
%    - Ideal for visualizing the distribution of a numerical variable.
%    - Example: Showing the frequency distribution of ages.

data = randn(1000, 1); % Sample data, 1000 random numbers
figure;
histogram(data); % Creating a histogram
title('Example Histogram');
xlabel('Value');
ylabel('Frequency');

% 3. Scatter Plots:
%    - Great for exploring relationships between two continuous variables.
%    - Example: Examining the relationship between height and weight.

x = randn(100, 1); % Random x values
y = randn(100, 1); % Random y values
figure;
scatter(x, y); % Creating a scatter plot
title('Example Scatter Plot');
xlabel('x');
ylabel('y');

% 4. Bar Plots:
%    - Suitable for comparing quantities across different categories.
%    - Example: Comparing average scores of different groups.

categories = {'A', 'B', 'C', 'D'};
values = [10, 20, 15, 25]; % Sample values for each category
figure;
bar(values); % Creating a bar plot
title('Example Bar Plot');
set(gca, 'xticklabel', categories); %'gca' stands for "get current axis"
ylabel('Values');


%% Practice 2.

% 1. Create a line plot for a sine wave till 2pi

% 2. Create a histogram of N(mu=5, sigma=3) with 1000 draws.

% 3. Simulate 100 obs and do a scatter plot of this Data Generating Process
%    wage= 0.5 + 3 * edu + error
%    edu follows a uniform (0,1)
%    error follos N mu=0, sigma=0.5

% 4. The probability of getting a job is 0.4, 0.6, 0.2 when
%    age below 30, 30-60, above 60 resepctively. Create a bar plot.
```

```matlab
%%  3. Data


% 1. MAT Files: (DONE)
% - Best for MATLAB-specific data structures.
% - Advantages: Keeps data types intact, efficient for MATLAB data.
% - Disadvantages: Only for MATLAB, not for data sharing outside MATLAB.

% 2. TABLES from CSV/Excel:  (FOCUS ON THIS)
% - Ideal for mixed-type data.
% - Advantages: Easy data access and manipulation, good for diverse data.
% - Disadvantages: Less efficient for large data, slower than arrays.

% 3. CELL ARRAYS/STRUCTURES:
% - Flexible for varied data types and sizes.
% - Advantages: Extremely versatile for mixed data.
% - Disadvantages: More complex to access and manipulate.



% We will use data from the World Bank
% https://databank.worldbank.org/source/world-development-indicators#

% File WBdata contains Panel Data from 1960-2022 with variables:
% - Country
% - Name
% - Country Code
% - Time
% - Time Code
% - GDP (constant 2015 US$) [NY.GDP.MKTP.KD]
% - Interest rate spread (lending rate minus deposit rate, %) [FR.INR.LNDP]
% - Battle-related deaths (number of people) [VC.BTL.DETH]



%% 3.1. Data: Import

clc;clear;

%There are two ways of importing data
    % 1 MANUALLY: good for not preprocessed data - bad for reloading

    % 2 COMMANDS: preprocessed data (or long commands) - easy to reload


% Import Data Manually: Home, Import Data - and play with VarType

% Importing a CSV file (FOCUS ON THIS)
dataCSV = readtable('WBdata.csv'); % Reads data from CSV file into table

% Importing an Excel file
dataXLS = readtable('WBdata.xlsx'); % Reads data from Excel file into table


% PRE-PROCESSING DATA:


% Missing values are often represented as NaN (Not a Number).
% This is important, it allows MATLAB to
% handle missing data in calculations and analyses without causing
% errors. When you perform operations like sum, mean, or other
% statistical calculations, MATLAB can ignore NaN values.
% Initial Data Inspection

% Display the names of all variables (columns) in the table
variableNames = dataCSV.Properties.VariableNames
```

```matlab
disp(head(dataCSV)); % Display the first few rows of the CSV data

% Rename long column names to simpler ones

dataCSV.Properties.VariableNames{'GDP_constant2015US___NY_GDP_MKTP_KD_'} ...
    = 'GDP';
dataCSV.Properties.VariableNames{'InterestRateSpread_lendingRateMinusDepositRate____FR_INR_LNDP_'} ...
    = 'InterestRateSpread';
dataCSV.Properties.VariableNames{'Battle_relatedDeaths_numberOfPeople__VC_BTL_DETH_'} ...
     = 'BattleDeaths';


% Update the numericColumns array with the new simplified names
str2numColumns = {'InterestRateSpread', 'BattleDeaths'};

% Loop over each specified numeric column
for columnName = str2numColumns
    % Convert the entire column
    dataCSV.(columnName{1}) = cellfun(@(x) str2double(regexprep(x, '\.\.', ...
'NaN')),dataCSV.(columnName{1}), 'UniformOutput', true);

    % cellfun(): Applies a function to each element in a cell structure.
       % str2double(): converts the string output from regexprep to double
          % regexprep(): Replaces '..' with 'NaN'.

       % 'UniformOutput',true: ensures output is uniform array (not cell)
end



% If we want to work with matrices, and not tables, in MATLAB, a matrix
% cannot contain both numeric and string values.

% One alternative is spliting the data

numeric = table2array(dataCSV(1 : end, [3 5 6 7]));
names = table2array(dataCSV(1 : end, [1 2 4]));


% Having data splitted in matrices is very useful, since you can easily
% apply functions to them.
```

```
%% 3.2. Data: Analysis

% A. Data as Matrices.

    mean(numeric(:,3))
    mean(numeric(:,3), 'omitnan')
    max(numeric(:,3),[], 'omitnan')  % special syntax

    unique(names(:,2)) % List of different Values
    length( unique(names(:,2)) ) % Count the number of unique countries

    % Create new dummy variable for large interest rates
    numeric(:, 5) = numeric(:, 3) > 5; % Creates logical (1 true, 0 false)

    % It takes NaN as 0! This is bad. How to circunvent this problem:
    numeric(:, 5) = NaN;
    notNaNIndices = ~isnan(numeric(:, 3)); % Find indices is not NaN
    numeric(notNaNIndices, 5) = numeric(notNaNIndices, 3) > 5;%1true,0false



% B. Data as a Table

    mean(dataCSV.InterestRateSpread, 'omitnan')

    % Other functions
    max(dataCSV.InterestRateSpread, [], 'omitnan')
    min(dataCSV.InterestRateSpread, [], 'omitnan')
    std(dataCSV.InterestRateSpread, 'omitnan')


    dataCSV.LargeIR = dataCSV.InterestRateSpread > 5;

    % Same Problem
    dataCSV.LargeIR = NaN(height(dataCSV), 1);
    notNaNIndices = ~isnan(dataCSV.InterestRateSpread (:));
    dataCSV.LargeIR(notNaNIndices) = ...
                                 dataCSV.InterestRateSpread(notNaNIndices) > 5;


% Refering to Specific Countries

% To refere to a subset of data refering to a specific country, is good to
% use index.


% A. Data as Matrices.

    % Virgin Islands are coded as British Virgin Islands
    index = find(contains (names , 'Virgin Islands')) % Contains
    index = find(strcmp (names, 'Virgin Islands')) % Exaclty

    mean(numeric(index, 3), 'omitnan')
    corrcoef(numeric(index, 2), numeric(index, 3), 'Rows', 'complete')

    index = find(strcmp (names, 'Trinidad and Tobago')) % Exactly the one


% B. Data as a Table
    index = find(contains(dataCSV.CountryName, 'Trinidad and Tobago'));
    index = find(strcmp (dataCSV.CountryName, 'Trinidad and Tobago')) % Exactly


    mean(dataCSV.InterestRateSpread(index), 'omitnan')
    corrcoef(dataCSV.GDP(index), dataCSV.InterestRateSpread(index), ...
        'Rows', 'complete')
```

```
%% 3.3. Data: Plotting

% Assuming the GDP data is in the 3rd column of the 'numeric' matrix
% and the corresponding time (years) is in the 1st column


% A. Data as Matrices.
    % Plotting GDP over time
    figure;
    plot(numeric(:, 1), numeric(:, 2));
    title('GDP Evolution Over Time (Matrix)');
    xlabel('Year');
    ylabel('GDP');


    % Plotting GDP over time for a country
    figure;
    plot(numeric(index, 1), numeric(index, 2));
    title('GDP Evolution Over Time (Matrix)');
    xlabel('Year');
    ylabel('GDP');



% B. Data as a Table

    % Plotting GDP over time
    figure;
    plot(dataCSV.Time, dataCSV.GDP);
    title('GDP Evolution Over Time (Table)');
    xlabel('Year');
    ylabel('GDP');

    figure;
    plot(dataCSV.Time(index), dataCSV.GDP(index));
    title('GDP Evolution Over Time (Table)');
    xlabel('Year');
    ylabel('GDP');





%% Practice 3.

% Use Matrice or Table approach:
% Using WB data, calculate statistics related to battle-related deaths,
% and plot the evolution of it time for specific countries.

% Tasks:
% 1. Calculate mean, max, and standard deviation of battle-related deaths
%    for Central African Republic

% 2. Create a variable called war, that takes value 1 if there are more
% than 1000 deaths (for all the Data)

% 3. Explore correlations between Interest Rate and war (for all the Data)

% 4. Plot the evolution over time if battle related deahts
%    (for Central African Republic)
```