Computational Tools for Macroeconomics using MATLAB

Week 3 – Data Input/Output & Plotting

Cristiano Cantore

Sapienza University of Rome

Recap — Week 2

- **Conditionals:** `if/elseif/else`, Booleans, short-circuit logic.
- **Loops:** `for`, `while`; preallocation; basic timing with `tic`/`toc`.
- **Functions:** inputs/outputs, help text, input validation; `dbstop` / `dbstep`.
- **Good practices:** comments, naming, folder structure, saving outputs.
- Questions from the homework? Any sticking points to revisit?

Week 3 — Goals & Roadmap

Today:

- **Data Types:** `table`, `timetable`, `struct`, `cell`, `numeric`, `string`, `datetime`.
- **Import/Export:** CSV, Excel, MAT (`readtable`, `writetable`, `load`/`save`).
- **Data containers:** `table` → `timetable`; quarterly dates; sorting.
- **Cleaning/Transforms:** handle missing values; `log`, differences, YoY growth.
- **Plotting basics:** `plot`, labels, titles, legends; saving figures (`saveas`, `savefig`).
- **Outputs:** export cleaned data to CSV for reports/reproducibility.

Week 3 Learning Outcomes

By the end of this week, students will be able to:

1. Understand and use different data types in MATLAB.
2. Import and export data between MATLAB and common formats (CSV, Excel, MAT).
3. Organize data using arrays, tables, and timetables.
4. Perform simple data cleaning and transformations.
5. Produce basic 2D plots and customise them.
6. Save figures and outputs for reports.

MATLAB Data Types: Numeric Arrays, Strings, Cells and NaNs

- The most common type in MATLAB is the **numeric array**:
  - `double` (default): floating point numbers.
  - `single`: lower precision (rare in economics).
  - `int8`, `int16`, `int32`, `int64`: integers.
  - **Examples:**

```
A = [1 2 3];            % 1x3 double array
x = 5;                  % scalar double
M = rand(3);            % 3x3 double matrix
```

- **Strings** (`string` or `char`):
  - Text data.

```
s1 = "GDP";         % string
s2 = 'Inflation'; % char
```

- **Cell arrays** (`{}`):
  - Store different types and sizes together.
  - Useful when mixing text and numbers.

```
C = {'Italy', 2024, 2.5};
```

- **NaN** = "Not a Number" (missing/invalid).

MATLAB Data Types: Tables, Timetables, Datetime

- **Tables**:
  - Organise data by variables (columns with names).
  - Similar to spreadsheets or Stata data frames.
- **Timetables**:
  - A table with a **time dimension**.
  - Best for time series: easy lag/lead operations.
- **Datetime**:
  - Special class for dates and times.
  - Works seamlessly with timetables.

```
d = datetime(2024,1,1);    % Jan 1, 2024
```

MATLAB Data Types: Structs

- **Structs**:
  - Containers with named fields (like a record or dictionary).

```
S.Country = 'Italy';
S.Year    = 2024;
S.GDP     = 2.5;
```

Reading Data: Modern MATLAB Functions

- **CSV / Excel:**
  - `T = readtable('gdp_quarterly.csv');`
  - `X = readmatrix('gdp_quarterly.xlsx');`
- **MAT-files (binary, MATLAB native):**
  - `load('mydata.mat')` → loads all variables in file
- Automatically detects headers, data types, missing values.
- `head(T)` or `T(1:5,:)` to preview the imported table.

Writing Data

- **CSV / Excel:**
  - `writetable(T,'cleaned_gdp.csv');`
  - `writematrix(X,'cleaned_gdp.xlsx');`
- **MAT-files:**
  - `save('mydata.mat','T','X');`
- Portability: prefer CSV for sharing; MAT for efficiency.

Demo: Import Quarterly GDP

```
% CSV
T = readtable('gdp_quarterly.csv');
head(T)

% Excel
X = readmatrix('gdp_quarterly.xlsx');

% Check variable types
summary(T)
```

- Inspect headers, variable names, and missing values.
- Use `summary(T)` to see datatypes (numeric, string, datetime).
- Try using readmatrix with .cvs and readtable with .xlsx. What do you notice?

Export Cleaned Table

```
% Suppose we computed GDP growth rates
T.growth = [NaN; diff(log(T.GDP))*100];

% Write to new CSV
writetable(T,'gdp_cleaned.csv');
```

- Always export cleaned/transformed data for reproducibility.
- Check the output file in Excel/Notepad.

Formatted Text Output (Optional)

```
% Example: write results to log
fid = fopen('results.log','w');
fprintf(fid,'Mean GDP growth: %.2f%%\n', mean(T.growth,'omitnan'
fprintf(fid,'Sample size: %d\n', height(T));
fclose(fid);
```

- Use `fprintf` for logging results or exporting formatted tables.
- Handy for reproducible reporting.

Importing Data from Online Sources

- Many statistical agencies and central banks provide **APIs** or **direct CSV links**.
- MATLAB can read data directly from a URL with functions like:
  - `readtable('https://...')`
  - `webread('https://...')`
- Advantages:
  - Always the latest version of the dataset.
  - No need to manually download CSV/Excel each time.
  - Facilitates reproducibility and automation.
- Common providers: FRED, World Bank, IMF, Eurostat, ECB.

Example: Download Nominal GDP from FRED

```
clear all; close all; clc;

% FRED: Quarterly Nominal GDP (SAAR, billions USD)
fred_url = 'https://fred.stlouisfed.org/graph/fredgraph.csv?id=GDP';
T = readtable(fred_url);        % columns: DATE, GDP

% Parse
dates_q = datetime(T.observation_date,'InputFormat','yyyy-MM-dd');
gdp_q   = T.GDP;   % nominal, SAAR, billions

% Plot
figure;
plot(dates_q, gdp_q, '-o','LineWidth',1.25); grid on;
xlabel('Year'); ylabel('GDP (US$, billions, SAAR)');
title('Nominal GDP (FRED GDP series) -- Quarterly');

% Save to CSV / Excel
Time = dates_q; GDP = gdp_q;
writetable(table(Time,GDP), 'gdp_quarterly.csv');
writetable(table(Time,GDP), 'gdp_quarterly.xlsx');
```

Example: Download Nominal GDP from World Bank (API)

```
%% WORLD BANK: Annual Nominal GDP (current US$)
% Country: set ISO3 code (e.g., 'ITA', 'USA', 'FRA', 'DEU', 'GBR', 'ESP',
% 'PRT') for all countries 'all'
country = 'USA';

% Build API URL (per_page large to get full history, JSON format)
url = ['https://api.worldbank.org/v2/country/' country ...
       '/indicator/NY.GDP.MKTP.CD?format=json&per_page=20000'];

% Read JSON
raw = webread(url);           % raw is a 1x2 cell array: {metadata, data}
data = raw{2};                % second cell contains an array of structs

%  Preallocate memory for the data
countries = cell(length(data), 1);
values = cell(length(data), 1);
periods = cell(length(data), 1);
countries_code = cell(length(data), 1);


% Extract GDP values and years from the data
for i = 1:length(data)
    if isempty(data(i).value)              % GDP value
        values{i} = NaN;                   % Missing data = NaN
    else
    values{i} = data(i).value;        % Extract GDP values
    end
    periods{i} = data(i).date;     % Extract corresponding years (strings)
    countries{i} = data(i).country.value;  % Country name
    countries_code{i} = data(i).countryiso3code;  % Country name
end

years=str2double(periods); %convert periods to numerical values
gdpValues=cell2mat(values); %convert values to numerical values
```

Online Data: Direct CVS download vs. JSON APIs

FRED (CSV download)

- Direct link to CSV file.
- Use `readtable('URL')` to import.
- Data ready in table format.
- Example from FRED (gdp_quarterly.csv).
- Very convenient for quick access.

World Bank (JSON API)

- Query an API endpoint.
- Use `webread('URL')` to download JSON.
- Parse metadata and values from struct arrays.
- Example from World Bank (wb_nominal_gdp.mat).
- More flexible but requires extra parsing.

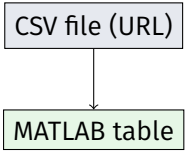
Key takeaway:

- CSV = simplest, “ready-to-use”.
- JSON APIs = more general, but need parsing.
- Both are reproducible and automate downloading the latest data.

Online Data: FRED vs. World Bank APIs

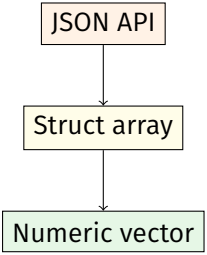
FRED (CSV download)

- Direct link to CSV file.
- Use `readtable('URL')` to import.
- Data ready in table format.



World Bank (JSON API)

- Query API endpoint.
- Use `webread('URL')` to get JSON.
- Parse struct array into numeric vector.



Key takeaway: CSV = quick and simple; JSON = flexible but needs parsing.

Finding Data URLs and APIs

- **FRED (Federal Reserve Economic Data)**
  - Search at: https://fred.stlouisfed.org/
  - Each series page has a "Download" button → choose CSV.
  - Example: GDP series → .../fredgraph.csv?id=GDP.
- **World Bank Open Data**
  - Explore indicators: https://data.worldbank.org/.
  - API documentation: https://datahelpdesk.worldbank.org/knowledgebase/articles/889392-api-documentation.
  - Format: /country/ISO3/indicator/CODE?format=json.
  - Example: US GDP → /country/USA/indicator/NY.GDP.MKTP.CD.
- **IMF (International Monetary Fund)**
  - Data portal: https://data.imf.org/.
  - API documentation: https://data.imf.org/api/documentation.
  - Example datasets: IFS (International Financial Statistics).
- **General tips:**
  - Look for "Download CSV/Excel" buttons → copy link address.
  - Check if APIs provide JSON/CSV endpoints for reproducibility.
  - Always document the URL + date retrieved in your script.

From `table` to `timetable`

```
% Assume T has variables: Time (datetime), GDP
head(T)

% Convert to timetable indexed by Time
TT = table2timetable(T, 'RowTimes', 'Time');

% Optional: drop duplicate time column if present, then sort
if any(strcmp('Time', TT.Properties.VariableNames))
    TT.Time = [];
end
TT = sortrows(TT);   % ensure chronological order

head(TT)
```

- `timetable`: time becomes the row index — ideal for time ops.
- Keep `Time` as `datetime` for quarters/dates.

Transforms: `log` and Differences

```
% Levels to logs
TT.GDP_log = log(TT.GDP);

% QoQ growth (approx, in %): 100 * diff(log(GDP))
TT.GDP_qoq = [NaN; 100 * diff(TT.GDP_log)];
```

- Log-diff ≈ percentage growth.
- Exact growth in levels: `100*(GDP(t)/GDP(t-1) - 1)`.

Handle Missing Values

```
% Quick diagnostics
summary(TT)

% Remove rows with any missing entries
TT_nomiss = rmmissing(TT);

% Or fill missing values (choose a method)
TT_fill = fillmissing(TT,'linear');% 'previous','next','constant
```

- Use `rmmissing` when gaps are rare; `fillmissing` for short runs.
- Always note your choice (reproducibility).

YoY Growth from Quarterly Data (no toolboxes)

```
% Exact YoY growth (%, needs 4-quarter lag)
TT.GDP_yoy = NaN(height(TT),1);
TT.GDP_yoy(5:end) = 100 * ( ...
    TT.GDP(5:end) ./ TT.GDP(1:end-4) - 1 );

% Inspect alignment
head(TT(:, {'GDP','GDP_qoq','GDP_yoy'}))
```

- First 4 entries are `NaN` by construction (insufficient lags).
- Indexing approach avoids toolbox dependencies.

Save an Intermediate Snapshot (`.mat`)

```
% Checkpoint for reproducibility
save('gdp_quarterly_clean.mat','TT');

% Later: reload and continue
clear TT
load('gdp_quarterly_clean.mat','TT');
whos TT
```

- Use `.mat` for fast reloads in class/homework.
- Also export a CSV for sharing outside MATLAB.

Build Quarterly `datetime` from Year/Quarter

```
% If your table has numeric Year and Quarter columns:
month = (T.Quarter - 1) * 3 + 1;   % 1,4,7,10
T.Time = datetime(T.Year, month, 1);    % first day of quarter
TT = table2timetable(T, 'RowTimes','Time');
```

- Constructing `datetime` ensures robust time indexing.
- Works even when original dates are missing or text-only.

Line Plots: `plot`, Labels, Title, Legend

```
% Assume TT has variables: GDP (level), GDP_yoy (%), and row tim
figure('Color','w');
plot(TT.Time, TT.GDP, '-o', 'LineWidth', 1.25);
grid on; box on;

xlabel('Time');
ylabel('GDP (billions)');
title('GDP Level -- Quarterly');
legend({'GDP'}, 'Location','best');
```

- Use `grid on` and `box on` for readability.
- `legend` clarifies multiple series (use descriptive labels).

Customising Lines, Markers, Limits

```
figure('Color','w');
plot(TT.Time, TT.GDP, '--s', 'MarkerSize', 5, 'LineWidth', 1.25)
grid on; box on;

xlabel('Time'); ylabel('GDP (billions)');
title('GDP Level -- Style Demo');

% Axis limits and ticks (optional)
xlim([TT.Time(1) TT.Time(end)]);
% ylim([min(TT.GDP)*0.95, max(TT.GDP)*1.05]);

% Add a horizontal reference line (e.g. long-run mean)
hold on; yline(mean(TT.GDP,'omitnan'), ':', 'Mean GDP');
hold off;
```

- Line style (`'--'`), marker (`'s'`), width, and size.
- `xlim`/`ylim` to focus the view; `yline` for references.

Multi-Series: Levels vs. Growth

```
% Option A: two panels (clear separation)
figure('Color','w');

subplot(2,1,1);
plot(TT.Time, TT.GDP, '-o', 'LineWidth', 1.2); grid on; box on;
ylabel('GDP (billions)'); title('GDP Level');

subplot(2,1,2);
plot(TT.Time, TT.GDP_yoy, '-s', 'LineWidth', 1.2); grid on; box on;
ylabel('YoY growth (%)'); xlabel('Time');
yline(0,':'); legend({'YoY'},'Location','best');

% Option B: dual y-axes (compact, but use carefully)
%{
figure('Color','w');
yyaxis left
plot(TT.Time, TT.GDP, '-o', 'LineWidth', 1.2); ylabel('GDP (billions)');
yyaxis right
plot(TT.Time, TT.GDP_yoy, '-s', 'LineWidth', 1.2); ylabel('YoY growth (%)');
grid on; box on; xlabel('Time'); title('GDP: Level (LHS) and YoY (RHS)');
legend({'GDP','YoY'},'Location','best');
%}
```

- Prefer **two panels** for clarity in teaching.
- `yyaxis` is concise but can be confusing—label clearly.

Annotate Recessions (Placeholder Band) & Save Figures

```
% Recession band placeholder (example dates)
t0 = datetime(2008,1,1); t1 = datetime(2009,6,1);

figure('Color','w');
plot(TT.Time, TT.GDP, 'LineWidth', 1.25); grid on; box on; hold on;

% Shade a time window with patch (behind the line)
yl = ylim;
patch([t0 t1 t1 t0], [yl(1) yl(1) yl(2) yl(2)], ...
        [0.9 0.9 0.9], 'EdgeColor','none', 'FaceAlpha',0.5);
uistack(findobj(gca,'Type','line'),'top'); % keep line on top

xlabel('Time'); ylabel('GDP (billions)');
title('GDP with Recession Band (Illustrative)');
hold off;

% Save outputs
saveas(gcf, 'week3_gdp_level.png');
savefig(gcf, 'week3_gdp_level.fig');
```

- Use `patch` to shade periods; set `FaceAlpha` for transparency.
- Always **save** plots as PNG (reports) and FIG (MATLAB editing).

Challenge

Task: Work with a quarterly GDP series that contains missing values.

- Load a provided CSV with and without missing observations.
- Build timetables from the tables.
- Diagnose the data: structure, missing patterns, check if the two time series are aligned.
- Fill missing values with a linear interpolation and a previous value.
- Compute **YoY growth** from the cleaned series.
- Compare with the **full (no-missing)** series and plot both YoY series.

Compare approaches:

- `rmmissing` vs `fillmissing('linear')` (or `'previous'`).
- Discuss pros/cons for growth-rate calculations.

Data & Deliverables

Input:

- `gdp_quarterly_missing.csv` (uploaded in the repo).
- `gdp_quarterly.csv` is the full dataset already used.

Deliverables:

- `week3_challenge_starter.m`.
- Figure comparing YoY growth: `week3_yoy_compare.png` and `.fig`.
- Cleaned data export: `gdp_quarterly_clean.csv`.

Reminder: YoY % growth (quarterly data)

$$\mathrm{YoY}(t) = 100 \times \left(\frac{\mathrm{GDP}(t)}{\mathrm{GDP}(t-4)} - 1 \right).$$

Homework — Wolrd Bank: Global Real GDP & Growth

Goal: Use the same series as the example in class to build a country panel, compute growth, and compare groups.

Tasks

1. Use the WB api to download **real GDP levels** (data code `NY.GDP.MKTP.KD`) for **as many countries as available**. You can do this either by using the API (harder) or by downloading the CSV from the website (easier).
2. Inspect the data structure and extract Real GDP values and years for each country. Make sure to handle properly strings and convert them to numerical values where applicable.
3. End up with a table with the following info: (`country` and/or `iso3`, `year`, `value`).
4. Compute Logs and **annual real GDP growth** for each country. Hint: you have to deal with missing values (NaN). Do not fill missing values just take logs of poisitve values for Real GDP (Hint: us an if statement).

Homework — Wolrd Bank: Global Real GDP & Growth

Goal: Use the same series as the example in class to build a country panel, compute growth, and compare groups.

Tasks

5. At this stage verify that you get the same results as in homework 1 by plotting the growth of the US Real GDP.
6. Select some countries of your choice and plot their Real GDP on the same graph. Add a legend to the plot. Save the figure as `week3_rgdp_comparison.png`.
7. Finally plot the growth rates for the selected countries in different subplots on the same figure. Try to also add a line for the average growth rate for each country. Save the figure as `week3_rgdp_growth.png`.
8. Export the table with the panel of selected countries with the growth rates to CSV.

Deliverables

- `week3_homework_solution.m` (main script) and any helper functions.
- `clean_rgdp_panel.csv`; `week3_rgdp_comparison.png`, `week3_rgdp_growth.png`.

Next Week

Next week: **Block B – Numerical Tools & Data Handling**
Week 4 – Matrix Algebra for Economics