

Time Series Econometrics/Econometrics II

PC Tutorial: Introduction to Matlab

Richard Schnorrenberger

richard.schn@stat-econ.uni-kiel.de

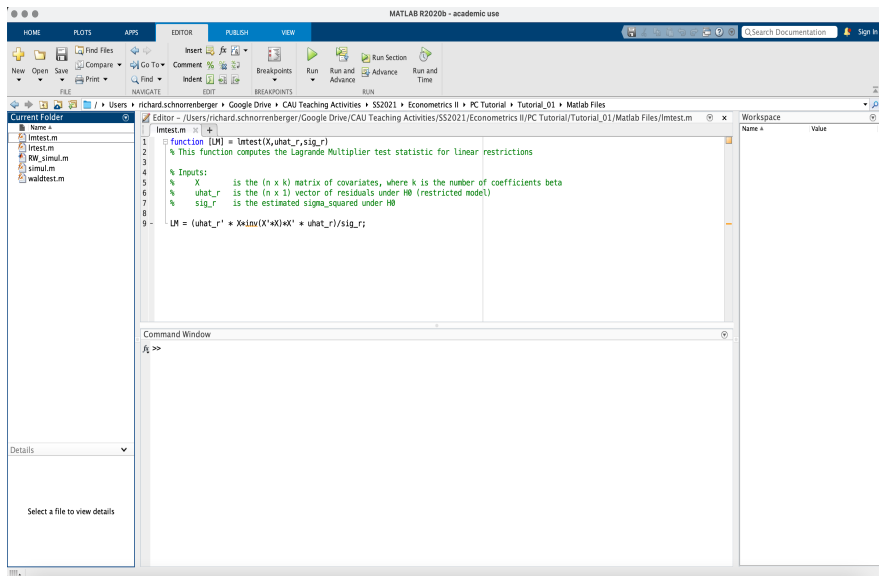
Institute of Statistics and Econometrics
University of Kiel

Summer Semester 2024

Matlab

- ▶ Matlab stands for the Matrix Laboratory.
- ▶ It has a very simple syntax.
- ▶ High performance and runs faster than most statistical software packages.
- ▶ Matlab toolboxes and MathWorks documentation. See, for instance, [MathWorks' documentation on the Econometrics Toolbox](#).
- ▶ Data visualization.

Matlab Interface



Vectors and Matrices

- ▶ % Create a row vector: `a = [1 2 3 4 5]`
- ▶ % Create a column vector: `a = [1 2 3 4 5]'` or `a = [1; 2; 3; 4; 5]`
- ▶ % Create a vector with evenly spaced elements: `t = 0:1:100`
- ▶ % Create matrices: `A = [1 2; 3 4]; B = A'; C = [1 2 3; 5 6 7; 9 10 11]`
- ▶ % Create an $N \times N$ empty matrix: `nan(N)`
- ▶ % Create an $N \times N$ matrix of zeros: `zeros(N)`
- ▶ % Create an $N \times N$ matrix of ones: `ones(N)`
- ▶ % Create an $N \times N$ identity matrix: `eye(N)`
- ▶ % Create diagonal matrix or get the main diagonal elements of matrix:
`d = diag([1 2 3]); diag(d)`
- ▶ % Get the size of the row or column dimension of a matrix `A`: `size(A,1)`, `size(A,2)`
- ▶ % Combine two vectors/matrices `A` and `B`: `C = [A B]`, `C = [A; B]`
- ▶ % Select the entry (i,j) in matrix `A`: `A(i,j)`
- ▶ % Select all entries in row m or column n : `A(m,:)`, `A(:,n)`
- ▶ % Get a selection of elements in row m or column n : `A(m,j1:j2)`, `A(i1:i2,n)`
- ▶ % Select last entry or elements in the last row/column: `A(end)`, `A(end,:)`, `A(:,end)`
- ▶ % Specify elements of a vector/matrix by indexing:
`a(end) = 2`, `X(:,1) = ones(size(X,1),1)`, `A(end,1:2) = zeros(1,2)`

For more commands on vectors (arrays) and matrices see [MathWorks documentation](#).

Matrix Operations

Let A and B be matrices with $A \in M(m \times n)$ and $B \in M(r \times s)$ and c be a scalar.

| Matrix Operation | Matlab Command | Condition |
|-----------------------------------|---|----------------|
| add scalar | $A+c$ | |
| add matrix | $A+B$ | $m = r, n = s$ |
| subtract scalar | $A-c$ | |
| subtract matrix | $A-B$ | $m = r, n = s$ |
| multiply scalar | $c*A$ | |
| matrix multiplication | $A*B$ | $n = r$ |
| element-wise multiplication | $A.*B$ | $m=r, m = s$ |
| to the power | A^c | $m = n$ |
| element-wise power | $A.^c$ | |
| transpose | A' | |
| invert | $\text{inv}(A)$ | A invertible |
| invert faster and efficiently (a) | $\text{eye}(\text{size}(A,1))/A$ | A invertible |
| invert faster and efficiently (b) | $A \backslash \text{eye}(\text{size}(A,1))$ | A invertible |
| matrix "division" | $A*\text{inv}(B) = A/B = B \backslash A$ | B invertible |

Control Structures:

Control structures such as `for`, `while`, and `if-else` are implemented through *loops* and *conditionals* as follows:

```
for i = 1:100 (loop counter)
    Statements
end
```

```
if x > y (condition)
    Statements
end
```

```
while x == 0 (condition)
    Statements
end
```

```
if x > y (condition)
    Statements
else if x == y (condition)
    Statements
else
    Statements
end
```

Conditional and Logical Operators:

Conditional Operators

| Operator | Relation | Application |
|--------------------|------------------|------------------------|
| <code>==</code> | Equality | <code>x == y</code> |
| <code>></code> | Larger than | <code>x > y</code> |
| <code><</code> | Smaller than | <code>x < y</code> |
| <code>>=</code> | Larger or equal | <code>x >= y</code> |
| <code><=</code> | Smaller or equal | <code>x <= y</code> |
| <code>~=</code> | Different | <code>x ~= y</code> |

Logical Operators

| Operator | Condition | Application |
|--------------------|-----------------|--------------------------------------|
| <code>&</code> | And | <code>x > 0 & y > 0</code> |
| <code>—</code> | Or | <code>x > 0 — y > 0</code> |
| <code>~</code> | Not | <code>~ (x > 0)</code> |
| <code>all</code> | All elements... | <code>all(x > 0)</code> |
| <code>any</code> | Any element... | <code>any(x > 0)</code> |

Functions

- ▶ You can write your own functions in Matlab.
- ▶ They are constructed as follows:

```
function [output1, output2, ...] = functionname(input1, input2, ...)
    Statements...
```

- ▶ For instance, create the Matlab function to compute the Wald test statistic for linear restrictions:

```
function [W] = waldtest(X,R,r,theta)
% X is the  $n \times k$  matrix of covariates
% R is the  $q \times k$  matrix of restrictions
% r is the  $q \times 1$  vector of restricted/hypothesized values
% theta is the  $(k + 1) \times 1$  vector of parameter estimates
n = size(x,1); % sample size
k = size(x,2); % number of covariates (including intercept)
sig = theta(end); % estimate for sigma
V = sig*inv(x'*x); % covariance matrix based on conditional expectations
W = (R*theta-r)'*inv(R*(V./n)*R')*(R*theta-r); % compute Wald test statistic
```


Loading Data into Matlab

The screenshot shows the MATLAB R2019a interface. The 'Import Data' button in the 'EDITOR' tab is circled in blue. Below it, the 'Import data from file' dialog box is open, showing the file 'german_yields.xls' selected. The 'Range' is set to 'A2:N361' and 'Output Type' is 'Table'. The 'Import Selection' button is highlighted with a green checkmark.

The 'Import Selection' dialog box shows the following data:

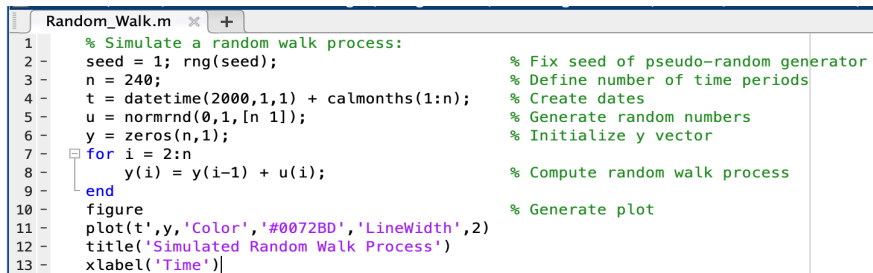
| germanyields | | | | | | | | | | |
|--------------|----------|----------|----------|----------|----------|----------|----------|----------|----------|------------|
| | Maturity | VarName2 | VarName3 | VarName4 | VarName5 | VarName6 | VarName7 | VarName8 | VarName9 | VarName... |
| 1 | 1990-01 | 8.1600 | 8.1360 | 8.1170 | 8.1340 | 8.1410 | 8.1270 | 8.0940 | 8.0480 | 7.9930 |
| 2 | 1990-02 | 8.7000 | 8.7630 | 8.8550 | 8.9160 | 8.9500 | 8.9620 | 8.9540 | 8.9280 | 8.8890 |
| 3 | 1990-03 | 8.4100 | 8.6450 | 8.8180 | 8.8500 | 8.8320 | 8.7910 | 8.7380 | 8.6790 | 8.6180 |
| 4 | 1990-04 | 8.5100 | 8.7440 | 8.9230 | 9.0070 | 9.0480 | 9.0630 | 9.0600 | 9.0450 | 9.0220 |
| 5 | 1990-05 | 8.4100 | 8.6440 | 8.8410 | 8.9360 | 8.9820 | 8.9960 | 8.9880 | 8.9660 | 8.9340 |
| 6 | 1990-06 | 8.3900 | 8.5500 | 8.7190 | 8.8170 | 8.8730 | 8.8970 | 8.8990 | 8.8840 | 8.8570 |
| 7 | 1990-07 | 8.2600 | 8.4450 | 8.5610 | 8.6110 | 8.6380 | 8.6510 | 8.6530 | 8.6470 | 8.6340 |
| 8 | 1990-08 | 8.7300 | 8.8140 | 8.9220 | 8.9970 | 9.0480 | 9.0810 | 9.0970 | 9.0990 | 9.0900 |
| 9 | 1990-09 | 8.7000 | 8.7910 | 8.9580 | 9.0860 | 9.1780 | 9.2380 | 9.2730 | 9.2860 | 9.2810 |

The 'Command Window' at the bottom shows the command:

```
fg >> load german_yields1992.mat
```

Plots

- ▶ Most useful plot functions: `plot(x,y)`, `scatter(x,y)`, `plotmatrix(x)`, `histogram(x,nbins)`.
- ▶ For basic plotting functions see the [YouTube video](#). For details on input arguments of the plot function (line style, color, title,...), see the [function documentation](#).
- ▶ For detailed information on all types of Matlab plots, see [MathWorks online documentation](#).
- ▶ To create sequences of dates and time, see [MathWorks online documentation](#).
- ▶ Example:



```

1 % Simulate a random walk process:
2 seed = 1; rng(seed); % Fix seed of pseudo-random generator
3 n = 240; % Define number of time periods
4 t = datetime(2000,1,1) + calmonths(1:n); % Create dates
5 u = normrnd(0,1,[n 1]); % Generate random numbers
6 y = zeros(n,1); % Initialize y vector
7 for i = 2:n
8     y(i) = y(i-1) + u(i); % Compute random walk process
9 end
10 figure % Generate plot
11 plot(t',y,'Color','#0072BD','LineWidth',2)
12 title('Simulated Random Walk Process')
13 xlabel('Time')
  
```

Matlab Plot (Time Series Example)

