

Bayesian Time Series Methods: Introductory

Computer Tutorial 1: MATLAB Basics

➤ What is MATLAB?

- MATLAB is an interpreted matrix programming language. MATLAB's interpreter translates statements into machine code one by one.
- We can work in MATLAB in three ways. From the command prompt, writing scripts and writing functions.
- By using M-files we can create a script, which can take no arguments. It is a series of statements that will be executed sequentially. M-files can also be used to create a function that can take and return arguments.

➤ Exercise 1 (a): Basic Matrix Commands

Creating a Matrix

- How to type the following 3-by-3 x matrix into MATLAB?

$$x = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}.$$

- Write a square bracket, separate elements in a row by space or comma, use semicolon to separate rows, and end the matrix with another square bracket.

```
>> x = [1 2 3; 4 5 6; 7 8 9];
```

```
>> disp x; >> disp(x);
```

- The former, prints x in the command window and the latter, prints only the value of x on command window.

➤ Exercise 1 (a cont.): Basic Matrix Commands

Matrix Arithmetic Operators

Operator Example

Meaning

+	$x + y$	Matrix addition is valid if x and y are of the same size.
-	$x - y$	Likewise, matrix subtraction is valid if x and y are of the same size.
*	$x * y$	Matrix multiplication is valid if the number of columns of x is the same as number of rows of y .

➤ Exercise 1 (a cont.): Basic Matrix Commands

Matrix Arithmetic Operators

Command	Meaning
$a = [x, y]$	Concatenates the 3-by-3 matrix x by adding the 3-by-3 matrix y as an additional column.
$b = [x; y]$	Concatenates the 3-by-3 matrix x by adding the 3-by-3 matrix y as an additional row.
$c = x(:, 2)$	Returns second column of x .
$d = y(:, 1)$	Accesses first column in y .
$e = x(2, 3)$	We can access specific elements in each argument. Here the command returns second element (row) in argument (column) 3 of x .

➤ Exercise 1 (b, c, d): Basic Matrix Commands

Matrix Arithmetic Operators

Command	Meaning
$f = x'$	The transpose operator is defined by a single quote. It flips a matrix around its main diagonal and it returns a row vector into a column vector.
$\text{eye}(m)$	Returns an m -by- m identity matrix.
$\text{eye}(m, n)$	Returns an m -by- n matrix.
$i = \text{eye}(\text{size}(x))$	Defines an identity matrix of size x .
$g = \text{plus}(i, x)$	Returns the identity matrix i plus x .
$\text{ginv} = \text{inv}(g)$	Returns inverse of g .

Exercise 2: OLS Estimation Using Artificial Data

- We are interested in the regression model of the form

$$y_i = \beta_1 + \beta_2 x_{i2} + \dots + \beta_k x_{ik} + \varepsilon_i, \text{ for } i = 1, \dots, n.$$

- Let x_i and ε_i be random draws from the $U(0, 1)$ and $N(0, 1)$ distributions, respectively. The above model in matrix notation is

$$\begin{array}{ccccccc} y & = & X & \beta & + & \varepsilon. \\ (n \times 1) & & (n \times k) & (k \times 1) & & (n \times 1) \end{array}$$

- where $X = [\iota, x_2]$, ι being an $n \times 1$ vector of ones.

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, X = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix}, \beta = \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix}.$$

Exercise 2 (a): OLS Estimation Using Artificial Data

- We compute the OLS estimates β from $b = (X'X)^{-1} X'y$.
- Recall that the OLS estimate of the error variance is $s^2 = e'e/(n - k)$, where $e = y - Xb$, and

$$\begin{aligned} e'e &= (y - Xb)'(y - Xb) = y'y - 2b'X'y + b'X'Xb \\ &= y'y - 2b'X'y + b'X'X(X'X)^{-1}X'y \\ &= y'y - b'X'y. \end{aligned}$$

```
Type:    >> bhat = inv(x'*x)*x'*y;
          >> resids = y - x*bhat;
          >> s2 = resids'*resids/(n-2);
```


Exercise 2 (b): OLS Estimation Using Artificial Data

- How well does the estimated model fit the data? Let's attempt to measure this in terms of the proportion of the variation in y explained by the model. We calculate R^2 .

$$\begin{aligned} R^2 &= 1 - \frac{\sum e_i^2}{\sum (y_i - \bar{y})^2} = 1 - \frac{\sum (y_i - \hat{y}_i)^2}{\sum (y_i - \bar{y})^2}, \\ &= 1 - \frac{RSS \leftarrow \text{Residual sum of squares}}{TSS \leftarrow \text{Total sum of squares}}. \end{aligned}$$

- where $\bar{y} = \sum y_i / n$ is the sample mean and \sum denotes $\sum_{i=1}^n$.

```
Type:  >> yhat = x*bhat;
        >> Rsq = 1 - sum((y - yhat).^2)/sum((y - mean(y)).^2);
        >> disp 'The R-squared of the regression is';
        >> disp(Rsq);
```

Exercise 2 (c): OLS Estimation Using Artificial Data

- Recall that $b = \beta + (X'X)^{-1} X'\varepsilon$. The conditional covariance matrix of b is

$$\begin{aligned}\text{var}(b \mid X) &= E[(b - \beta)(b - \beta)' \mid X] \\&= E[(X'X)^{-1} X'\varepsilon\varepsilon'X(X'X)^{-1} \mid X] \\&= (X'X)^{-1} X'E(\varepsilon\varepsilon' \mid X)X(X'X)^{-1} \\&= \sigma^2 (X'X)^{-1} X'X(X'X)^{-1} \text{ since } E(\varepsilon\varepsilon' \mid X) = \sigma^2 I_n \\&= \sigma^2 (X'X)^{-1}.\end{aligned}$$

Type: `varb = s2*inv(x'*x);`
 `disp 'The covariance matrix of the OLS estimator is';`
 `disp(varb);`

➤ Exercise 3 (a): For *loops* and *if* statements

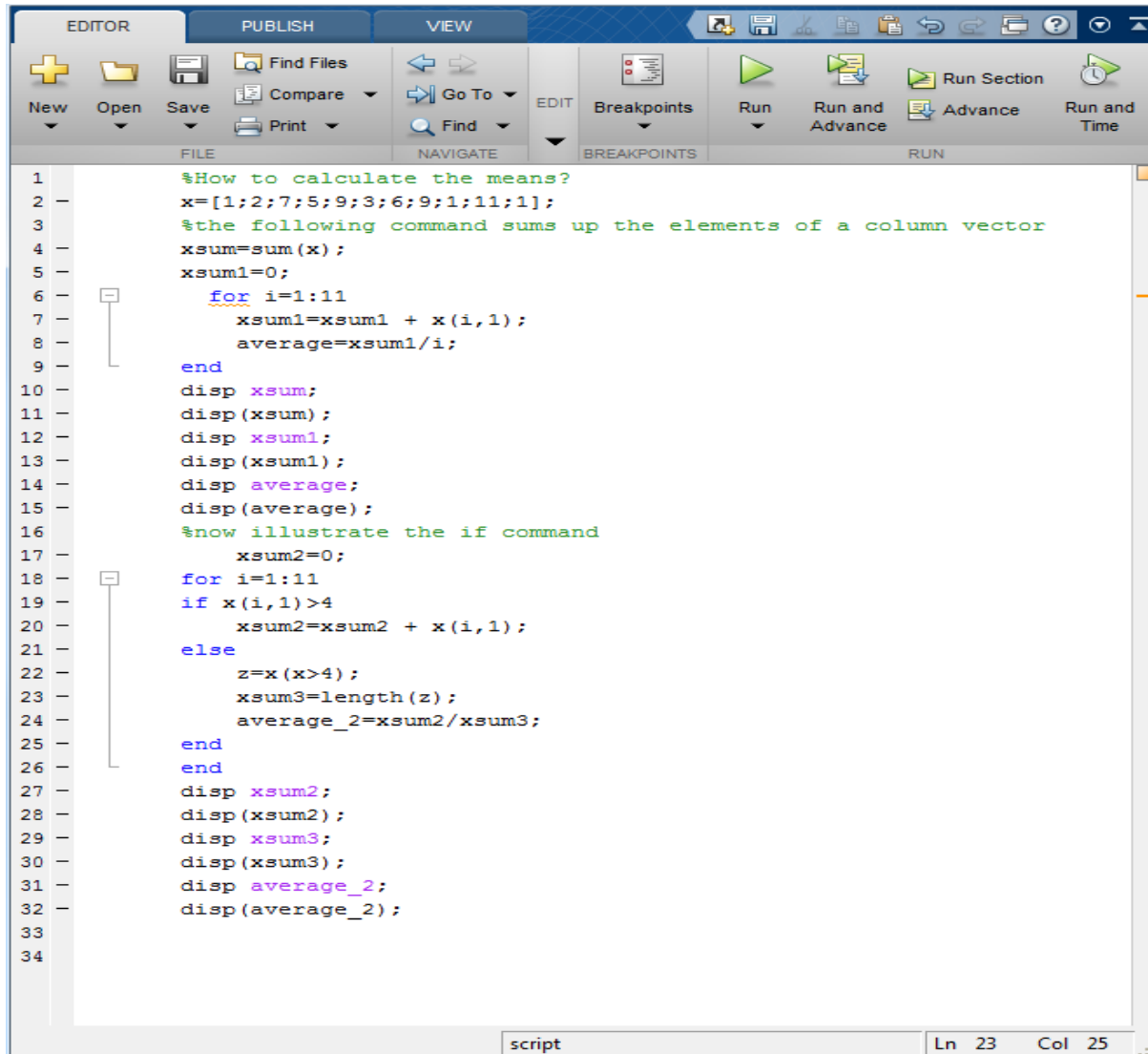
Type: `>> x=[1;2;7;5;9;3;6;9;1;11;1];`

- This command returns an 11-by-1 column vector of x .

`>> xsum=sum(x); xsum1=0;`

- First command sums up the elements of the vector x and the second command returns a zero element.
- The *for* command, iterates over procedure “ $xsum1 = xsum1 + x(i,1)$ ”; incrementing i from 1 to 11 by 1. In other words, it computes the sum of all numbers from 1 to 11 in the procedure.
- The *If* command proceeds as follows, *if* criteria “ $x(i,1) > 4$ ” is true do “ $xsum2=xsum2 + x(i,1)$ ”.

➤ Exercise 3 (b): For *loops* and *if* statements



The image shows a screenshot of the MATLAB Editor window. The interface includes a top toolbar with tabs for EDITOR, PUBLISH, and VIEW. Below the tabs are icons for New, Open, Save, Find Files, Compare, Print, Go To, Find, Breakpoints, Run, Run and Advance, Run Section, Advance, and Run and Time. The main editor area displays a script with the following code:

```
1 %How to calculate the means?
2 x=[1;2;7;5;9;3;6;9;1;11;1];
3 %the following command sums up the elements of a column vector
4 xsum=sum(x);
5 xsum1=0;
6 for i=1:11
7     xsum1=xsum1 + x(i,1);
8     average=xsum1/i;
9 end
10 disp xsum;
11 disp(xsum);
12 disp xsum1;
13 disp(xsum1);
14 disp average;
15 disp(average);
16 %now illustrate the if command
17 xsum2=0;
18 for i=1:11
19     if x(i,1)>4
20         xsum2=xsum2 + x(i,1);
21     else
22         z=x(x>4);
23         xsum3=length(z);
24         average_2=xsum2/xsum3;
25     end
26 end
27 disp xsum2;
28 disp(xsum2);
29 disp xsum3;
30 disp(xsum3);
31 disp average_2;
32 disp(average_2);
33
34
```

The status bar at the bottom indicates the file is named 'script' and the cursor is at line 23, column 25.

➤ Exercise 4: Drawing from Standard Distributions

Uniform Distribution

- To generate a uniform random number, use the MATLAB command “rand”. (type ‘help rand’ in console)

Type: `>> tempp = rand(num_draws, 1);`

- “rand(m , n)” command returns m -by- n matrix of independent uniformly distributed random entries. In the example, it will generate a 10-by-1 column vector of uniformly distributed random numbers.

`>> unifans = [mean(tempp) std(tempp)];`

- “tempp” is an 10-by-1 column vector, “mean(tempp)” returns a row vector of size 1 whose j th entry is the mean of the j th column of “num_draws”. “std(tempp)” returns the standard deviation of “tempp”.

➤ Exercise 4 (cont.): Drawing from Standard Distributions

Normal Distribution

- Obtaining draws from normal random number generator in MATLAB can be done by the function “randn” and its functionality is similar to the function “rand”.

Type: `>> tempp = randn(num_draws, 1);`

- Generates an 10-by-1 vector of normally distributed random numbers.

`>> normans = [mean(tempp) std(tempp)];`

- “tempp” is an 10-by-1 column vector, “mean(tempp)” returns a row vector of size 1 whose j th entry is the mean of the j th column of “num_draws”. “std(tempp)” returns the standard deviation of “tempp”.

➤ Exercise 4 (cont.): Drawing from Standard Distributions

Student's- t Distribution

Type: `>> tempp = trnd(3, num_draws, 1)`

- “trnd” command generates a 10-by-1 column vector of random numbers from Student's t distribution with degrees of freedom set to $\nu = 3$.

➤ Exercise 4 (cont.): Drawing from Standard Distributions

Beta Distribution

- Generate random variables from the Beta distribution with parameters using the command “`betarnd(α_1 , α_2)`” by setting $\alpha_1 = 3$ and $\alpha_2 = 2$.

Type `>> tempp = betarnd(3, 2, num_draws, 1);`

- The command returns a 10-by-1 column vector of random numbers generated from the Beta distribution.

➤ Exercise 4 (cont.): Drawing from Standard Distributions

Exponential Distribution

Type: `>> tempp = exprnd(5, num_draws, 1);`

- This command returns random numbers from exponential distribution with mean parameter set to $\mu = 5$ and number of draws set to a column vector with 10 rows.

➤ Exercise 4 (cont.): Drawing from Standard Distributions

Chi-Square Distribution

- A Chi-square distribution with ν degrees of freedom calls for the use of “chi2rnd2” command. It generates random numbers from the Chi square distribution.

Type: `>> tempp = chi2rnd(3, num_draws, 1);`

- The command returns a 10-by-1 column vector of random numbers.

➤ Exercise 4 (cont.): Drawing from Standard Distributions

Gamma Distribution

- We can generate random numbers from the Gamma distribution with mean μ and degrees of freedom ν as follows

```
>> tempp = gamrnd(4, 2, num_draws,1);
```

- The command returns a 10-by-1 column vector of random numbers generated from the Gamma distribution with $\mu = 4$ and $\nu = 2$.

➤ Exercise 5: Monte Carlo Integration

- Suppose that the posterior density

$$p(\theta | y),$$

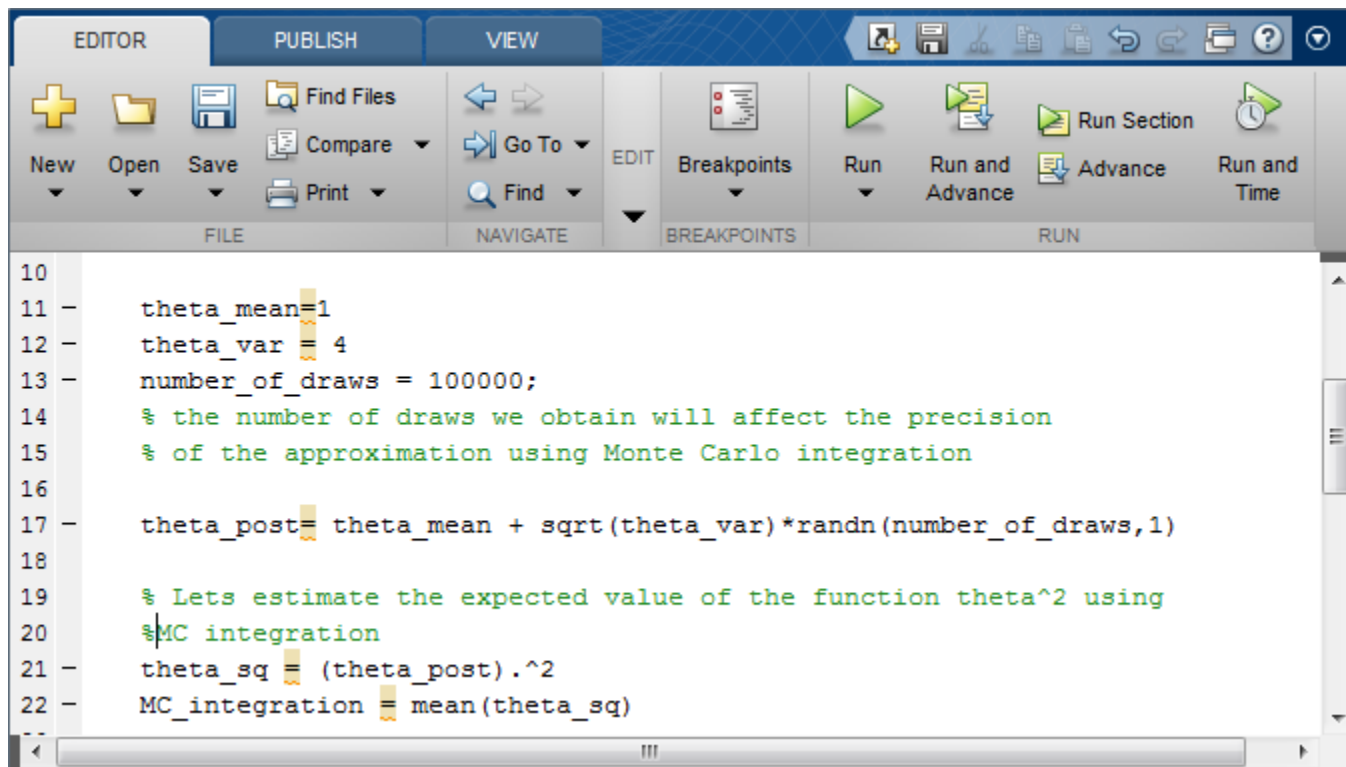
- has a known form. Thus, we can generate R i.i.d. random draws of the parameters, $\theta^{(r)}$, for $r = 1, \dots, R$.
- But we are interested in the parameters with a functional form $g(\theta)$.
- To estimate the $E(g(\theta) | y)$, sampling methods use a sample average as

$$E(g(\theta) | y) = \int g(\theta) P(\theta | y) d\theta \approx \frac{\sum_{r=1}^R g(\theta^{(r)})}{R}.$$

➤ Exercise 5: Monte Carlo Integration

- Lets generate the parameter of interest from the function $g(\theta) = \theta^2$ from a Normal density

$$p(\theta | y) \sim N(1,4).$$

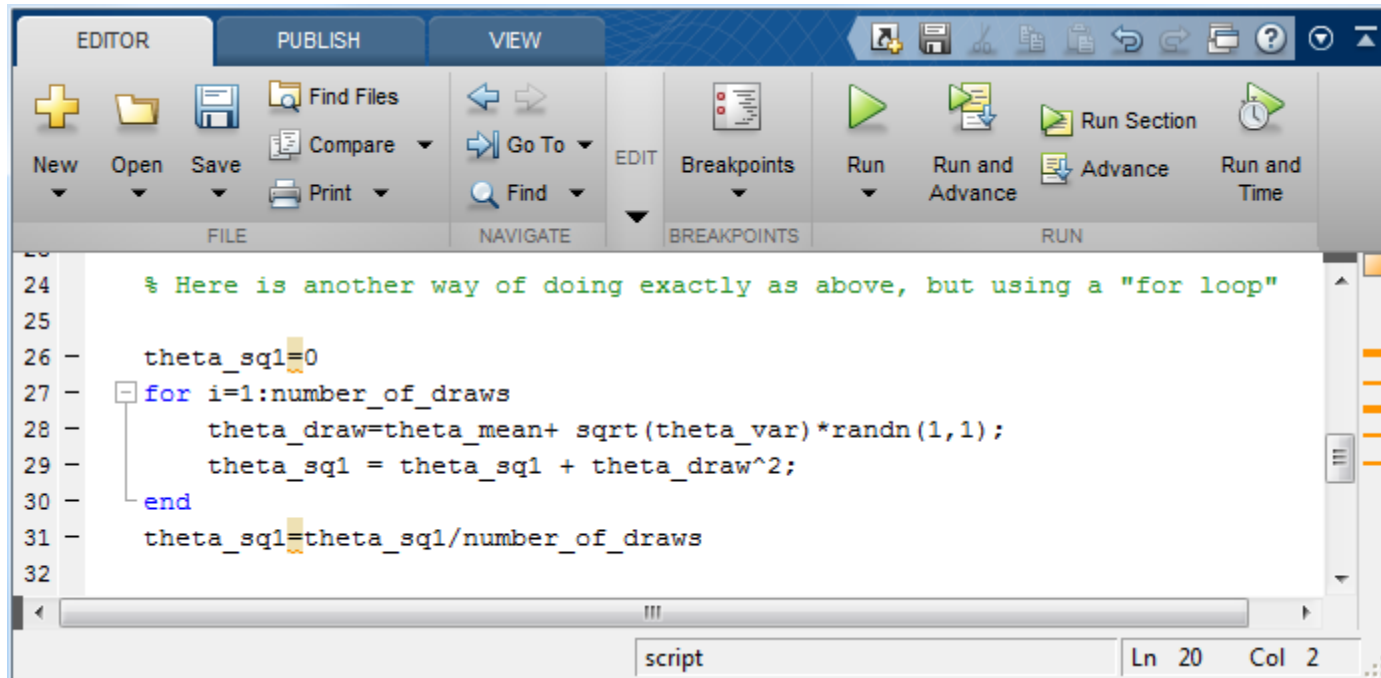


The screenshot shows a MATLAB editor window with the following code:

```
10
11 - theta_mean=1
12 - theta_var = 4
13 - number_of_draws = 100000;
14 % the number of draws we obtain will affect the precision
15 % of the approximation using Monte Carlo integration
16
17 - theta_post= theta_mean + sqrt(theta_var)*randn(number_of_draws,1)
18
19 % Lets estimate the expected value of the function theta^2 using
20 %MC integration
21 - theta_sq = (theta_post).^2
22 - MC_integration = mean(theta_sq)
```

➤ Exercise 5 (cont.): Monte Carlo Integration

- Alternative method using control flow



The screenshot shows the MATLAB script editor interface. The top toolbar includes tabs for EDITOR, PUBLISH, and VIEW. Below these are icons for New, Open, Save, Find Files, Compare, Print, Go To, Find, Breakpoints, Run, Run and Advance, Run Section, and Run and Time. The main editor area displays a script with the following code:

```
24 % Here is another way of doing exactly as above, but using a "for loop"
25
26 theta_sq1=0
27 for i=1:number_of_draws
28     theta_draw=theta_mean+ sqrt(theta_var)*randn(1,1);
29     theta_sq1 = theta_sq1 + theta_draw^2;
30 end
31 theta_sq1=theta_sq1/number_of_draws
32
```

The status bar at the bottom indicates the file is named 'script' and the cursor is at line 20, column 2.

➤ References

- Koop, G. (2003), Bayesian econometrics, Wiley.
- Koop, G., Poirier, D. and Tobias, J. (2007), Bayesian econometric methods. Cambridge: Cambridge University Press.