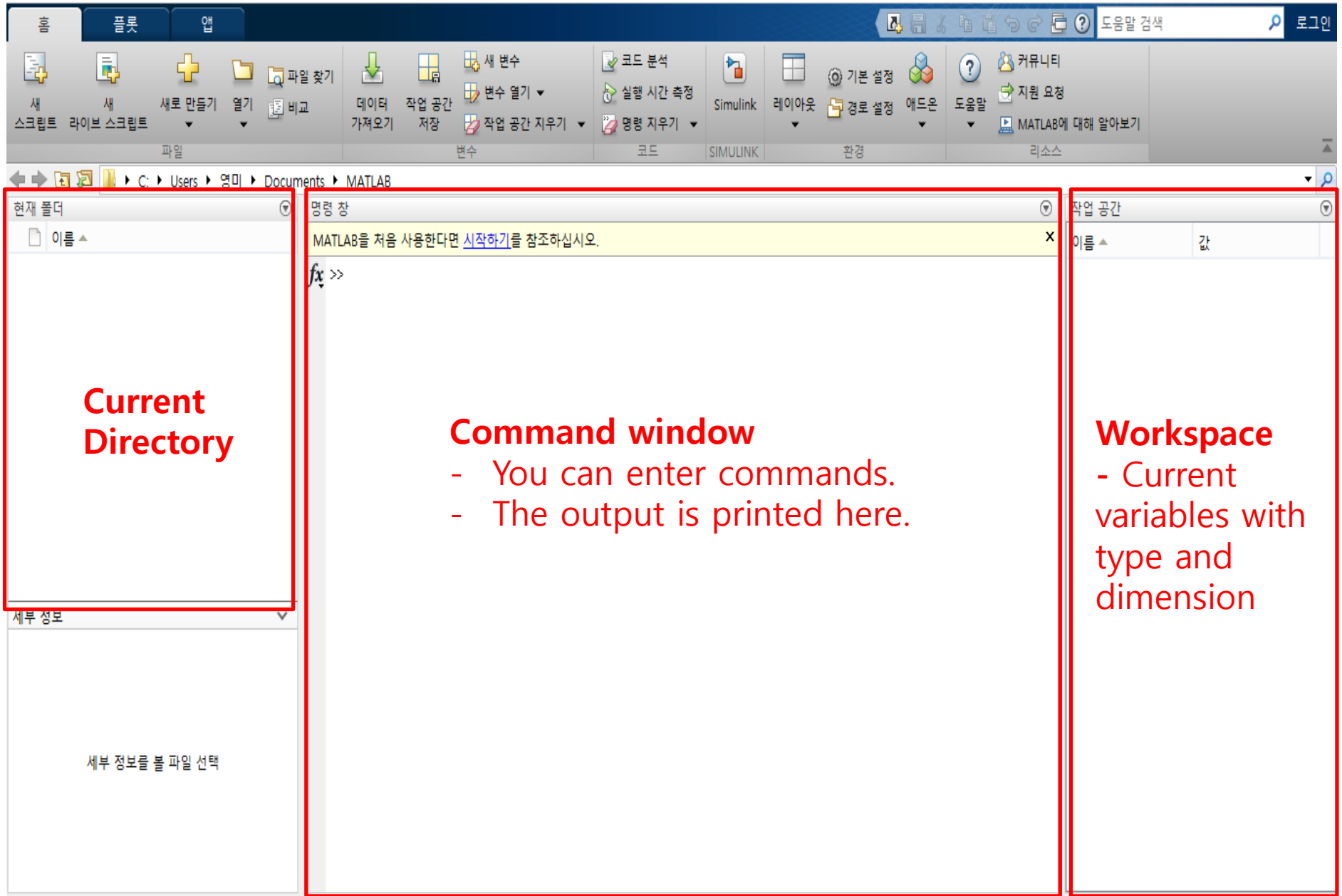
MATLAB Session I

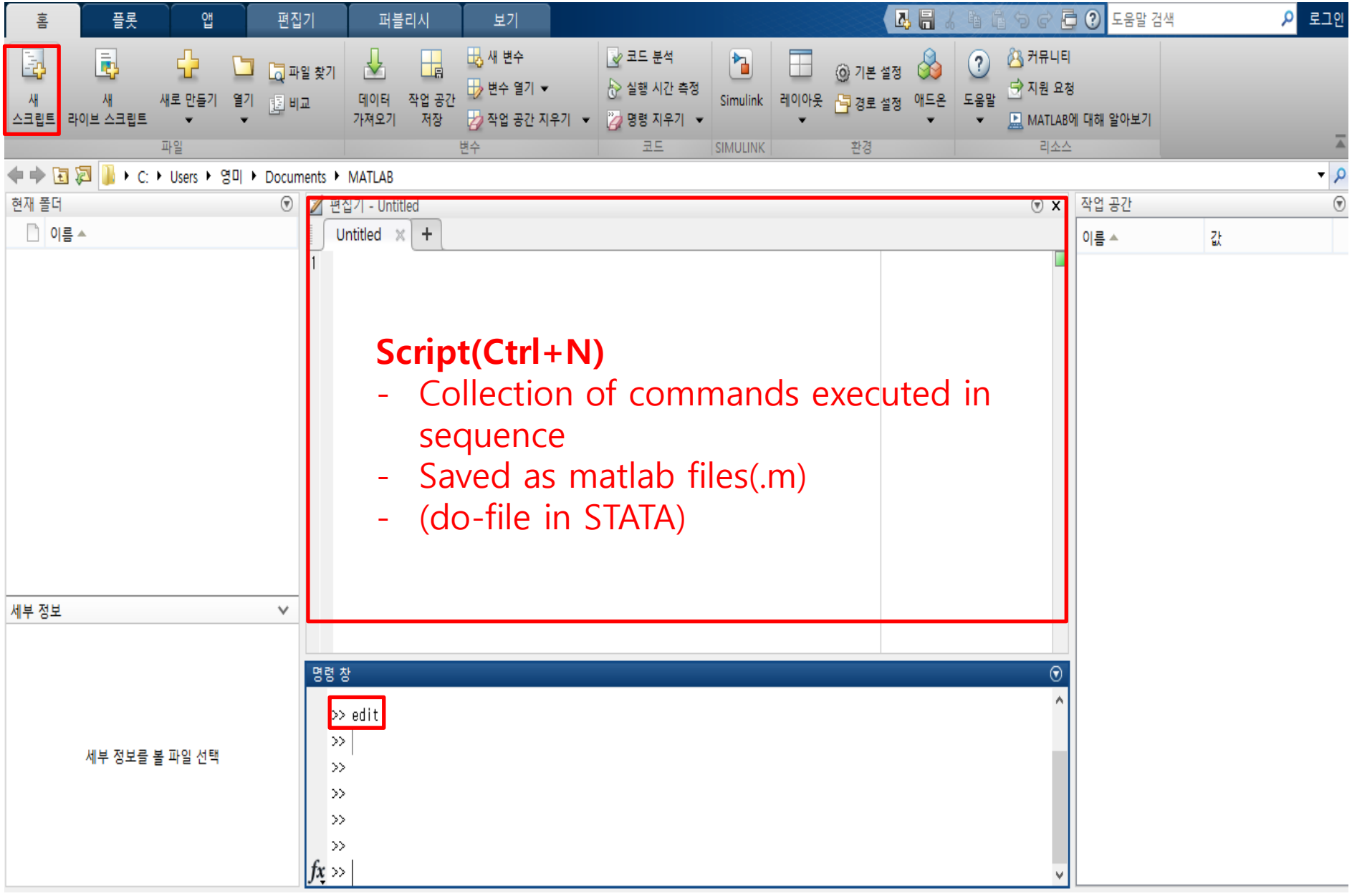
171024 TA Session 4
Yeongmi Jeong

Contents

Layout


홈
플롯
앱
도움말 검색
로그인
새 스크립트
새 라이브 스크립트
새로 만들기
열기
파일 찾기
비교
파일
데이터 가져오기
작업 공간 저장
새 변수
변수 열기
작업 공간 지우기
변수
코드 분석
실행 시간 측정
명령 지우기
코드
Simulink
SIMULINK
레이아웃
기본 설정
경로 설정
애드온
환경
도움말
커뮤니티
지원 요청
MATLAB에 대해 알아보기
리소스
C: ▸ Users ▸ 영미 ▸ Documents ▸ MATLAB
현재 폴더
이름
세부 정보
세부 정보를 볼 파일 선택
명령 창
MATLAB을 처음 사용한다면 시작하기를 참조하십시오.
>>
작업 공간
이름
값
Current Directory
Command window
- You can enter commands.
- The output is printed here.
Workspace
- Current variables with type and dimension

Layout

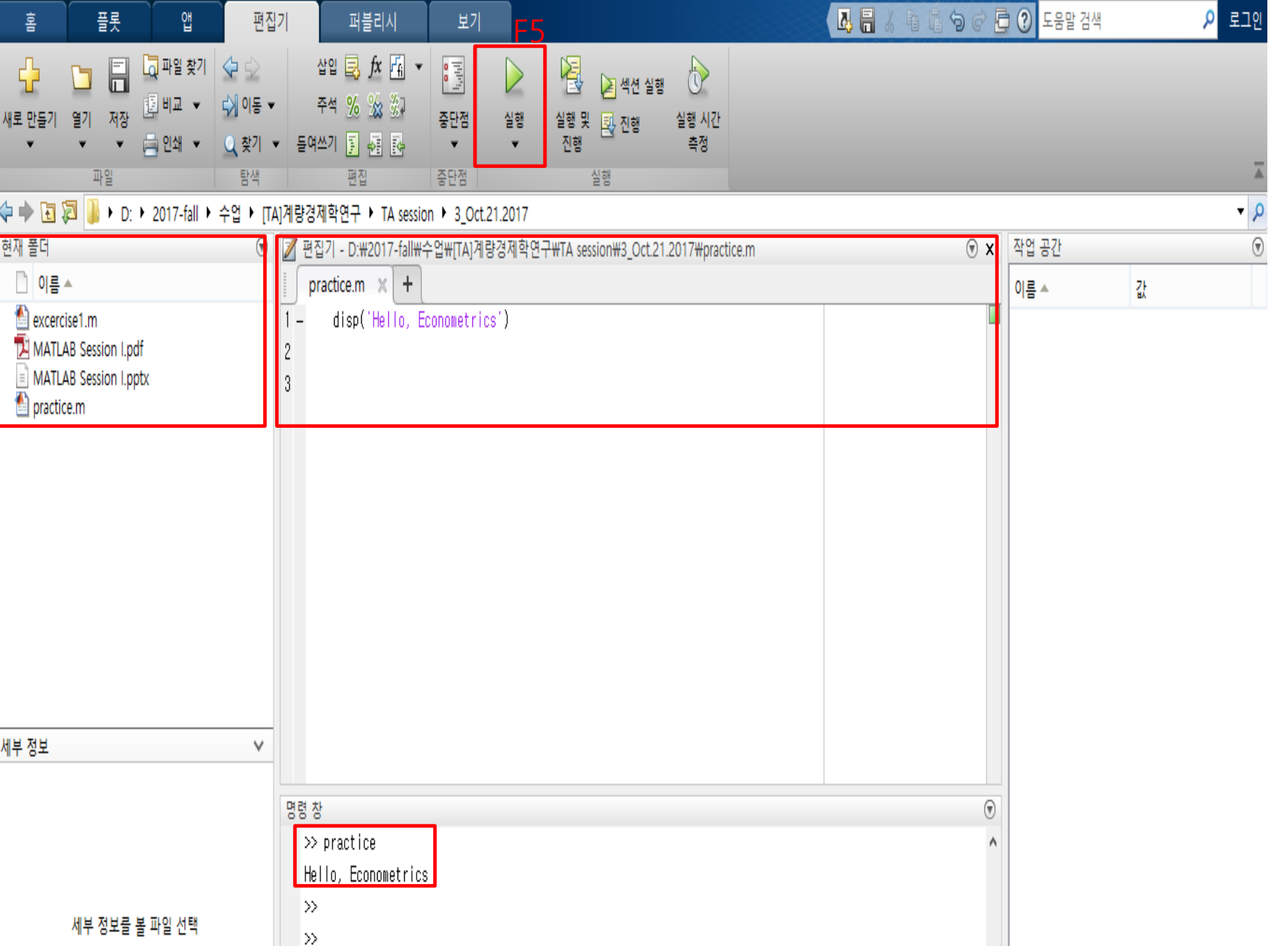

홈
플롯
앱
편집기
퍼블리시
보기
도움말 검색
로그인
새 스크립트
새 라이브 스크립트
새로 만들기
열기
파일 찾기
비교
데이터 가져오기
작업 공간 저장
새 변수
변수 열기
작업 공간 지우기
코드 분석
실행 시간 측정
명령 지우기
Simulink
레이아웃
기본 설정
경로 설정
애드온
도움말
커뮤니티
지원 요청
MATLAB에 대해 알아보기
파일
변수
코드
SIMULINK
환경
리소스
C: ▸ Users ▸ 영미 ▸ Documents ▸ MATLAB
현재 폴더
이름
세부 정보
세부 정보를 볼 파일 선택
편집기 - Untitled
Untitled
1
Script(Ctrl+N)
- Collection of commands executed in sequence
- Saved as matlab files(.m)
- (do-file in STATA)
명령 창
>> edit
작업 공간
이름
값

Execution/ Stop

- **Execution**
- (Command window) Enter commands to the command window
- (Script) Click the execution button to execute all command in a script
- (Script) F5 to execute all commands in a script

- **Stop**
- Let the cursor be on the command window. Try Ctrl+c.

- **Display characters** on the command window: disp('statement')


홈
플롯
앱
편집기
퍼블리시
보기
F5
도움말 검색
로그인
새로 만들기
열기
저장
파일 찾기
비교
인쇄
이동
찾기
삽입
주석
들여쓰기
중단점
실행
실행 및 진행
섹션 실행
진행
실행 시간 측정
파일
탐색
편집
중단점
실행
D: ▸ 2017-fall ▸ 수업 ▸ [TA]계량경제학연구 ▸ TA session ▸ 3_Oct.21.2017
현재 폴더
이름
excercise1.m
MATLAB Session I.pdf
MATLAB Session I.pptx
practice.m
편집기 - D:₩2017-fall₩수업₩[TA]계량경제학연구₩TA session₩3_Oct.21.2017₩practice.m
practice.m
1 - disp('Hello, Econometrics')
2
3
작업 공간
이름
값
세부 정보
세부 정보를 볼 파일 선택
명령 창
>> practice
Hello, Econometrics
>>
>>

Help command

- **Help keyword**

- Displays the help text for the functionality specified by keyword on the command window.

- **Doc keyword**

- Displays documentation for the functionality specified by keyword.

명령 창

```
>> help regression
regression - Linear regression

    This MATLAB function takes these arguments, Target matrix or cell array data
    with a total of N matrix rows Output matrix or cell array data of the same size

    [r,m,b] = regression(t,y)
    [r,m,b] = regression(t,y,'one')

    참고 항목 confusion, plotregression

    regression에 대한 함수 도움말 문서 페이지

fx >>
```

Documentation

최신 문서 검색

목차 닫기

- Documentation Home
- Neural Network Toolbox
- Function Approximation and Clustering
- Function Approximation and Nonlinear Regression
- Neural Network Toolbox
- Function Approximation and Clustering
- Pattern Recognition
- Neural Network Toolbox
- Functions

regression

ON THIS PAGE

- Description
- Examples
- See Also

regression

Linear regression

Syntax

```
[r,m,b] = regression(t,y)
[r,m,b] = regression(t,y,'one')
```

Description

`[r,m,b] = regression(t,y)` takes these arguments,

t	Target matrix or cell array data with a total of N matrix rows
y	Output matrix or cell array data of the same size

and returns these outputs,

r	Regression values for each of the N matrix rows
m	Slope of regression fit for each of the N matrix rows
b	Offset of regression fit for each of the N matrix rows

`[r,m,b] = regression(t,y,'one')` combines all matrix rows before regressing, and returns single scalar regression, slope, and offset values.

Examples

Fit Regression Model and Plot Fitted Values versus Targets

Train a feedforward network, then calculate and plot the regression between its targets and outputs.

```
[x,t] = simplefit_dataset;
net = feedforwardnet(20);
net = train(net,x,t);
y = net(x);
```

Contents

Variables

- **Types**
- 64 bit **double**: a=2
- 16 bit character: a='Hello, Econometrics'
- **1 bit logical, true(1) or false(0)**: a=(1>0)

- **Declaration**
- '=' is the sign for assignment, does not mean 'equal'
- No need to initialize variable types
- To suppress output, end the line with a semicolon

- **Names**
- Case sensitive(Var1, var1).
- First character must be a letter.
- Do not use built-in variables
  i: imaginary number
  pi:3.1415…
  ans: the last assigned value
  inf, -inf: positive and negative infinity
  NaN: not a number

명령 창

```
>> a=2;
>> a='Hello, Econometrics'

a =

    'Hello, Econometrics'

>> a=(1>0)

a =

  logical

   1

>> a=(1<0)

a =

  logical

   0
```

작업 공간

이름	값
a	0.0000 + 1.0000i
b	3.1416
c	Inf
d	NaN
var1	1
Var1	2

Vectors/ Matrix

- **Row vectors**
  : comma or space separated values between bracket
  a=[1 2 3 4], b=[1,2,3,4]

- **Column vectors**
  : semicolon separated values between bracket
  c=[1;2;3;4]

- **Matrices**
  : A=[1 2; 3 4] or
  a=[1 2] b=[3 4] and A=[a; b] or
  a=[1; 3] b=[2; 4] and A=[a, b]

- **Diagonal matrices**
  : A=diag([1 2 3])

```
>> A=diag([1 2 3])

A =

     1     0     0
     0     2     0
     0     0     3
```

명령 창

```
>> a=[1 2 3 4]

a =

     1     2     3     4

>> b=[1; 2; 3; 4]

b =

     1
     2
     3
     4
```

```
>> a=[1 2]

a =

     1     2

>> b=[3 4]

b =

     3     4

>> A=[a;b]

A =

     1     2
     3     4
```

Vectors/ Matrix

- i th element of a vector: **a(i)**
- (i, j) th element of a matrix: **A(i,j)**
- i th row of a matrix: **A(i,:)**
- j th row of a matrix: **A(:,j)**
- i, i+1, …, j rows of a matrix: **A(i:j, :)**
- i, i+1, …, j columns of a matrix: **A(:, i:j)**

명령 창

```
>> A=[1 2 3; 4 5 6]

A =

     1     2     3
     4     5     6

>> A(2,2)

ans =

     5

>> A(2,:)

ans =

     4     5     6

>> A(1, 2:3)

ans =

     2     3
```

Vectors/ Matrix

- **Combining matrices**(: dimension must be matched)
  r_combined=[A; B] or
  c_combined=[A B]

- **Edit matrix components**
  A(i,j)=value;
  A(i,:)=vector;
  A(:,j)=vector;

- **Eliminating components**
  A(i,:)=[ ];
  A(:,j)=[ ];

명령 창

```
>> A=[1 2 3; 4 5 6]

A =

     1     2     3
     4     5     6

>> B=[7 8 9]

B =

     7     8     9

>> C=[A;B]

C =

     1     2     3
     4     5     6
     7     8     9
```

```
>> C(1,1)=10;
>> C

C =

    10     2     3
     4     5     6
     7     8     9

>> C(:,1)=[];
>> C

C =

     2     3
     5     6
     8     9
```

Scalar Operations

(Command window acts as a calculator)

- **Arithmetic operations**: 2+3, 2-3, 2*3, 2/3

- **Multiplication is not implicitly given**
  : (2+3)*2 works but (2+3)2 gives an error

- **Exponential**: 2^3

- **Built-in functions**:
  sqrt(2)
  log(2)
  cos(pi/2), sin(pi/2), tan(pi/2)
  exp(2)
  round(1.5), ceil(1.5), floor(1.5)
  abs(-1)

Vector/Matrix Operations

- Standard **matrix addition/subtraction/multiplication** work.
  The **dimension** must be matched!

명령 창

```
>> A=[1 0;0 1];
>> B=[1 2; 3 4];
>> A+B

ans =

     2     2
     3     5
```

명령 창

```
>> A=[1 0; 0 1];
>> B=[1 2; 3 4];
>> A*B

ans =

     1     2
     3     4
```

- Dot enables **element-wise operation**.: A.+B A.-B A.*B A./B
  The **dimension** must be matched!

명령 창

```
>> A=[1 0; 0 1];
>> B=[1 2; 3 4];
>> A.*B

ans =

     1     0
     0     4
```

명령 창

```
>> A=[1 0; 0 1];
>> B=[1 2; 3 4];
>> A./B

ans =

    1.0000         0
         0    0.2500
```

Vector/Matrix Operations

- Built-in functions work on matrices: exp(A), log(A), sqrt(A), etc.

명령 창

```
>> clear
>> B=[1 2; 3 4];
>> exp(B)

ans =

    2.7183    7.3891
   20.0855   54.5982
```

명령 창

```
>> B=[1 2; 3 4];
>> log(B)

ans =

         0    0.6931
    1.0986    1.3863
```

명령 창

```
>> B=[1 2; 3 4];
>> sqrt(B)

ans =

    1.0000    1.4142
    1.7321    2.0000
```

Vector/Matrix Functions

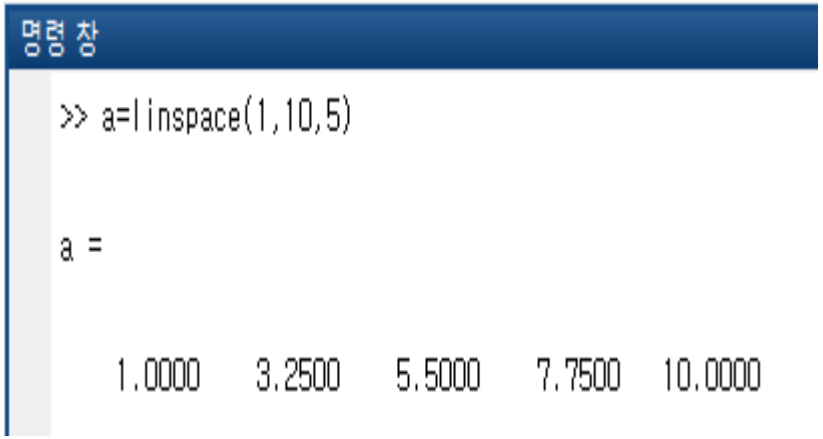
- Transpose: transpose(A), A'
- Sum of each column, of each row, & of all elements
  : sum(A), sum(A,2), & sum(A(:))
- Product of each column, of each row, & of all elements
  : prod(A), prod(A,2), & prod(A(:))
- Minimum of each column, of each row, & of all elements
  : min(A), min(A,[],2), & min(A(:))
- Maximum of all column, of each row, & of all elements
  : max(A), max(A,[],2), & max(A(:))
- Dimension of matrix
  - Number of rows: n=size(A,1)
  - Number of columns: k=size(A,2)
  - Together: [n, k]=size(A)

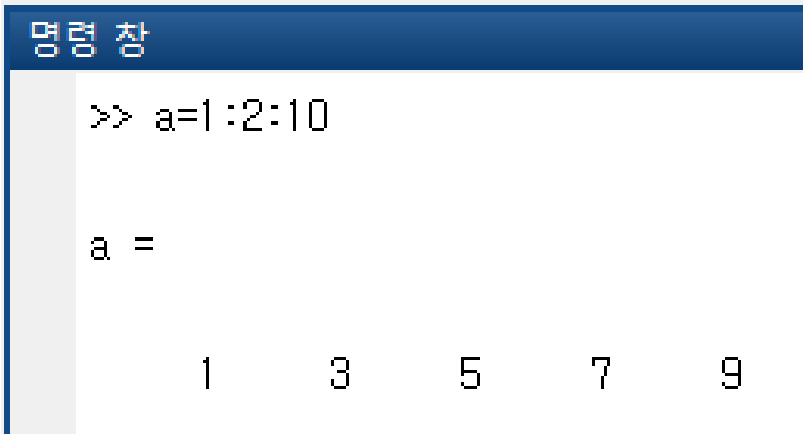
Automatic Initializations

- A matrix of ones
  : **ones(n,k)** where n is the # of rows, and k is the # of columns

- A matrix of zeros
  : **zeros(n,k)**

- Arithmetic sequence
  - Specifying the size of sequence: **linspace(first, last, n)**
  - Specifying the increment: **first:n:last**





Contents

Basic Statistic Functions

- **Generating random variables**
  : **random('distribution', parameter1, parameter2, ..., [n1, n2, ...])**
  eg. random('normal', 0, 1, n, k): n by k random matrix of N(0,1)
  eg. random('poisson', 5, n, k, r): n by k by r 3-d matrix of Poisson(5)

  - short-cut for uniform(0,1): **rand(n1,n2, ...)**
  - short-cut for N(0,1): **randn(n1, n2, ...)**

- **Inverse distribution function**
  : **icdf('distribution', probability, parameter1, parameter2, ...)**
  eg. Icdf('normal', 0.975, 0, 1): 0.975 quantile of N(0,1)

- **Mean**: mean(A), mean(A,2), and mean(A(:))

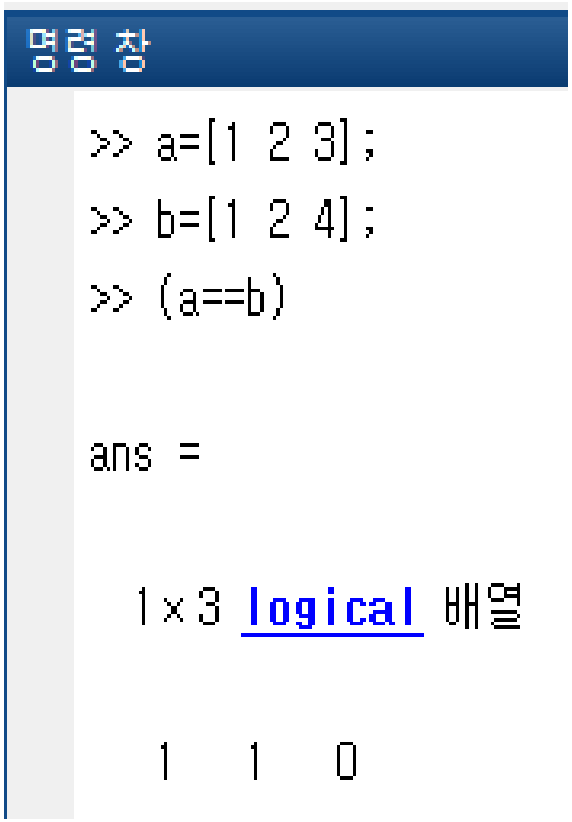
- **Variance**: var(A), var(A,2), var(A(:))

- **Standard deviation**: std(A), std(A,2), std(A(:))

Contents

Relational Operators

- **Standard relational operators**
  - equal ==
  - not equal ~=
  - greater than >
  - less than <
  - greater than or equal to >=
  - less than or equal to <=
- **Logical operators**
  - and &
  - or |



If/ else/ elseif

- 'if' executes a group of statements when the condition is true.
- 'else' or 'elseif' blocks are optional. The statements execute only if the condition in the 'if' are false.
- An 'if' block can include multiple 'elseif' block.

```
% if/elseif/else example

heights=[130 155 180];
if heights(1)==max(heights)
    disp('The first person is the tallest');
elseif heights(2)==max(heights)
    disp('The second person is the tallest');
else disp('The last person is the tallest');
```

```
명령 창
    The last person is the tallest
```

For loop

- '**for**' executes statements specified number of times.

  - n=n+1; is implicitly embedded in the end of the commands.
  - Loop variable is scalar within the command block.
    Loop variable can be defined by a vector.
    eg. a=0:2:10
    eg. a=[0 2 4 6 8 10]
  - To specify certain element of a vector in the command block,
    the vector MUST be declared ahead of the loop.

```
% generating Fibonacci sequence

seq=zeros(1,10);
seq(2)=1;

for a=3:10
    seq(a)=seq(a-1)+seq(a-2)
end

seq
```

명령 창

```
seq =

     0     1     1     2     3     5     8    13    21    34
```

While loop

- '**while**' repeats the execution of a group of statements while the condition is true.
  - no need to specify the number of iteration unlike the for loop.
  - caution: infinite loop!

```
% computing a factorial

n=5;
fac=1;
while n>1
    fac=fac*n;
    n=n-1;
    % unlike the 'for-loop', the increment
    % or decrement must be specified.
    % Otherwise, infinite loop occurs.
end
```

명령 창

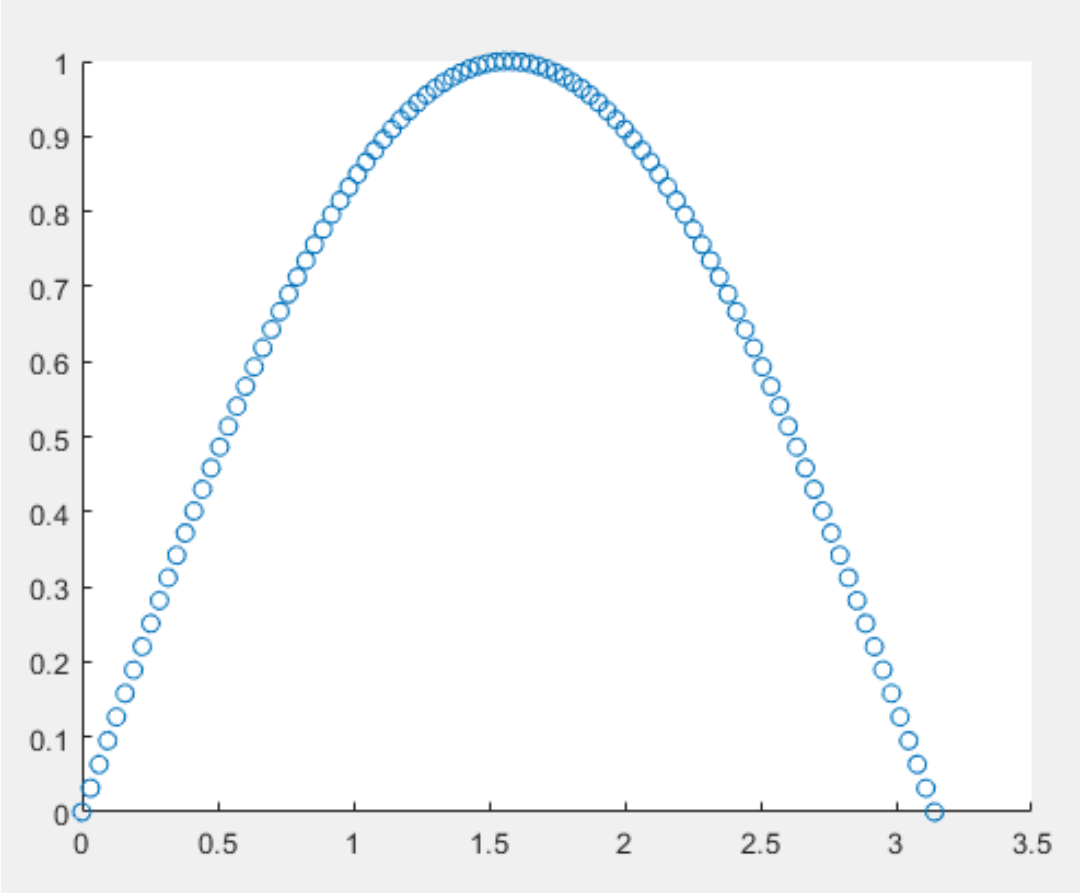
```
fac =

   120
```

Contents

Scatter Plot

- Scatter plot
  : **scatter(x,y)** where x and y are vectors with the same dimension

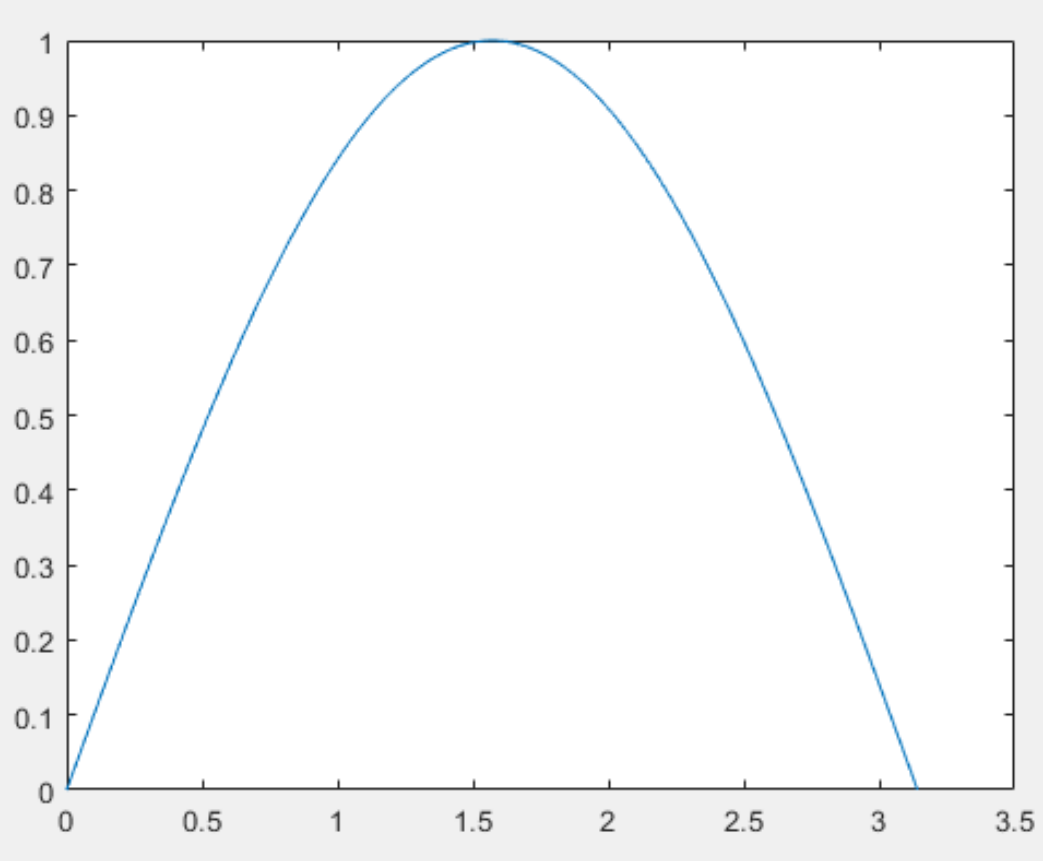
```
x=linspace(0, pi, 100);
y=sin(x);
scatter(x,y);
```



2D Line Plot

- Line plot
  : **plot(x,y)** where x and y are vectors with the same dimension

```
x=linspace(0, pi, 100);
y=sin(x);
plot(x,y);
```



- There are various options for the plot function. See the help documents.