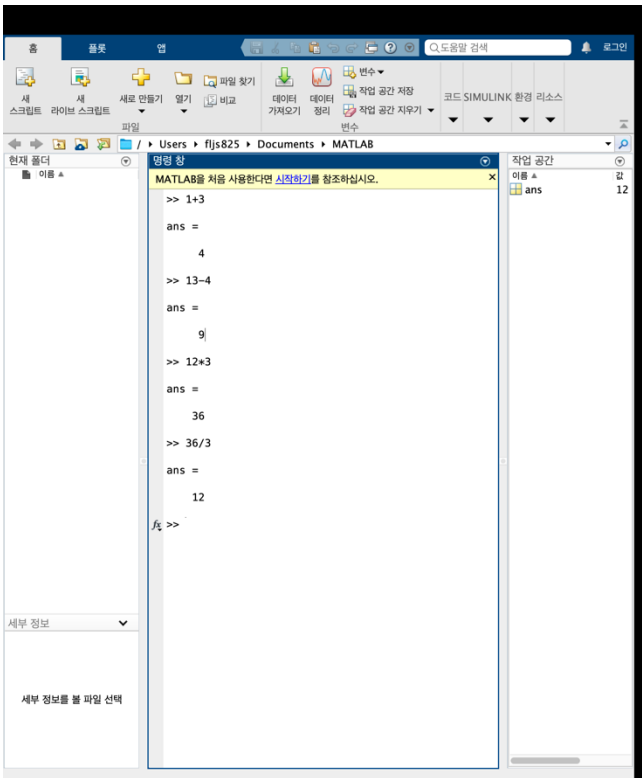


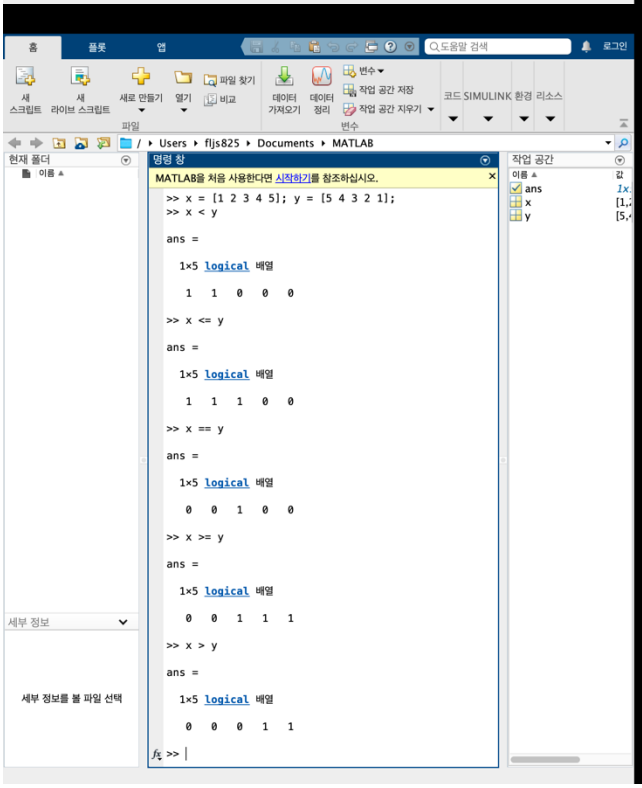
(+), 빼기 (−), 곱하기 (∗), 나누기 (/)가 있다.

`A + B`	A와 B를 더하다
`A - B`	A에서 B를 빼다
`A * B`	A에 B를 곱하다
`A / B`	A를 B로 나누다


```
>> 1+3
 ans = 4
>> 13-4
 ans = 9
>> 12*3
 ans = 36
>> 36/3
 ans = 12
```

관계연산자

같은 크기를 갖는 두 행렬의 서로 대응되는 원소의 크기를 비교하기 위하여 사

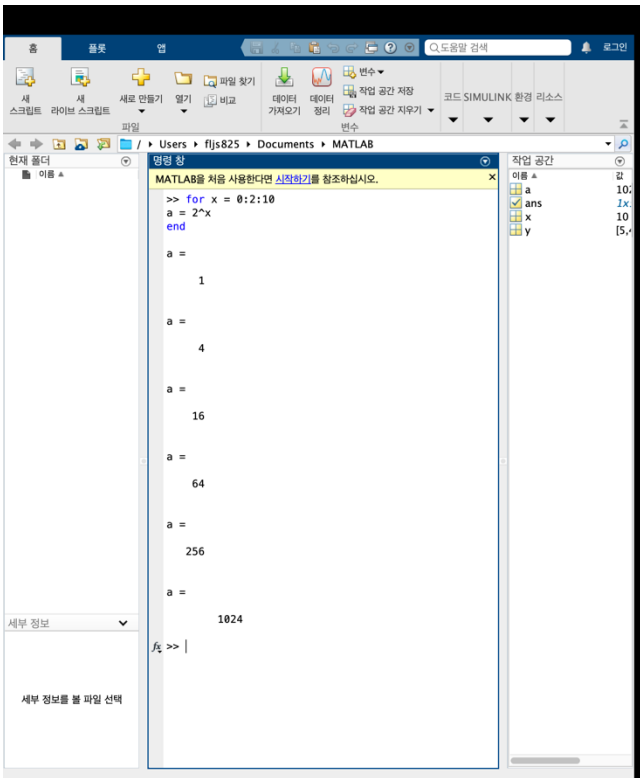



```
>> x = [1 2 3 4 5]; y = [5 4 3 2 1];
>> x < y
  ans =  1     1     0     0     0
>> x <= y
  ans =  1     1     1     0     0
>> x == y
  ans =  0     0     1     0     0
>> x >= y
  ans =  0     0     1     1     1
>> x > y
  ans =  0     0     0     1     1
```

논리연산자

다양한 논리연산자 중 자주 사용하는 몇 가지에 대해서 알아보도록 하자.

`A & B`	A와 B가 둘 다 동시에 참일 때만 참이고, 그 외에는 모두 거짓
`A | B`	A 또는 B 중 적어도 하나만 참이면 참이고, 그 외에는 거짓



제 2 절 기본 구문

MATLAB 기본 구문으로는 for 문, if 문, while 문이 있으며, 각 구문은 모두 end와 짝을 이루어 사용된다.

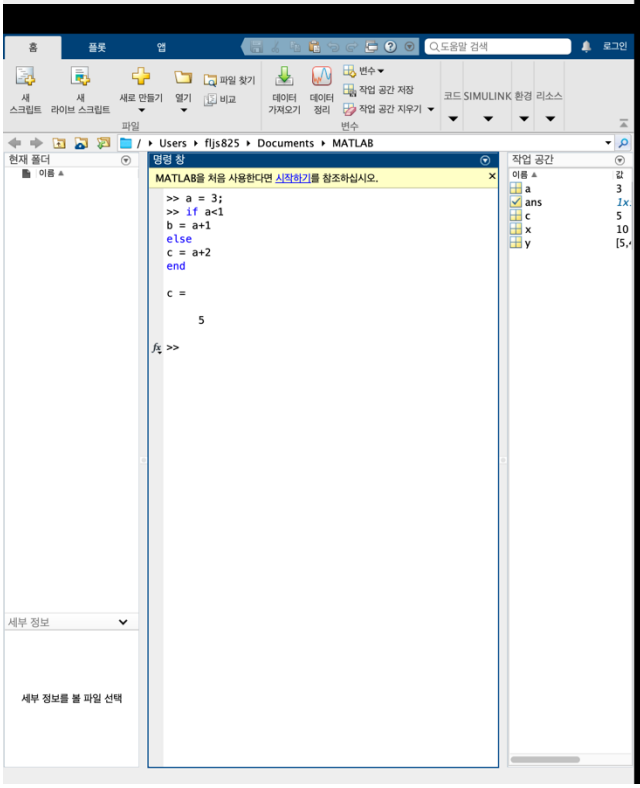
for 문

'for'문은 'end'문과 짝을 이루어 사용된다. 'for'문과 같은 행에 있는 변수의 값을 초깃값부터 증분의 크기만큼 누적시키면서 최종 값에 도달될 때까지 'for'문과 'end'문 사이 문장의 명령을 수행한다. 증분이 '1'인 경우에는 '증분:'을 생략해도 무방하다.

```
for x=0:2:10
  a=2^x
end
```

위의 코드를 보면 변수 x는 0부터 2간격으로 10까지 변화하는 것을 알 수 있다. 즉, 각 x가 차례대로 0, 2, 4, 6, 8, 10의 값이 입력되어 a 값을 출력하게 된다.

```
a = 1
a = 4
a = 16
a = 64
a = 256
a = 1024
```



if else 문

여러 가지 조건에 따라 각각 다른 명령을 실행하고자 할 때, 'if ~ else ~ end'문을 사용한다. 아래 보기처럼 **조건 1**이 참이면 **문장 1**이 수행되고, **조건 1**이 모두 거짓이면, 'if'문을 빠져 나와 **문장 2**가 수행된다.

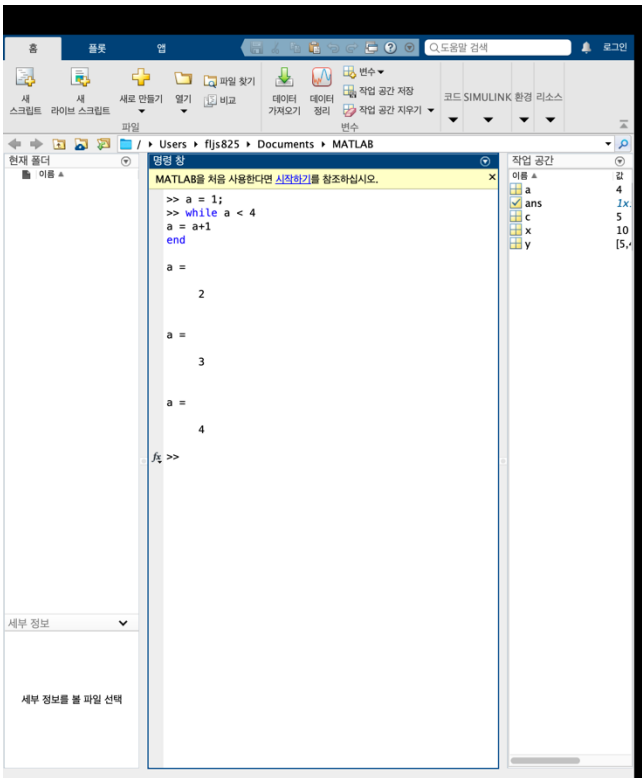
```
a=3;
if a<1
    b=a+1
else
    c=a+2
end
```

위의 코드는 a의 초깃값을 3으로 지정하게 된다. if 구문을 살펴보면 처음 조건은 a가 1보다 작을 경우에 b값을 $a+1$로 출력하게 되며, 그렇지 않을 경우 c값을 $a+2$로 출력하게 되는 것을 알 수 있다. 따라서 $a=3$은 첫 번째 조건을 만족하지 않으므로 결과 값으로 $c=5$값을 출력하는 것을 알 수 있다.

```
c = 5
```

while 문

'while'문은 'end'문과 짝을 이루어 사용된다. 'while'문과 같은 행에 있는 조건이 참이면 'while' 문과 'end'문 사이에 있는 문장의 명령을 반복적으로 수행한다.



while 문

'while'문은 'end'문과 짝을 이루어 사용된다. 'while'문과 같은 행에 있는 조건이 참이면 'while' 문과 'end'문 사이에 있는 문장의 명령을 반복적으로 수행한다.

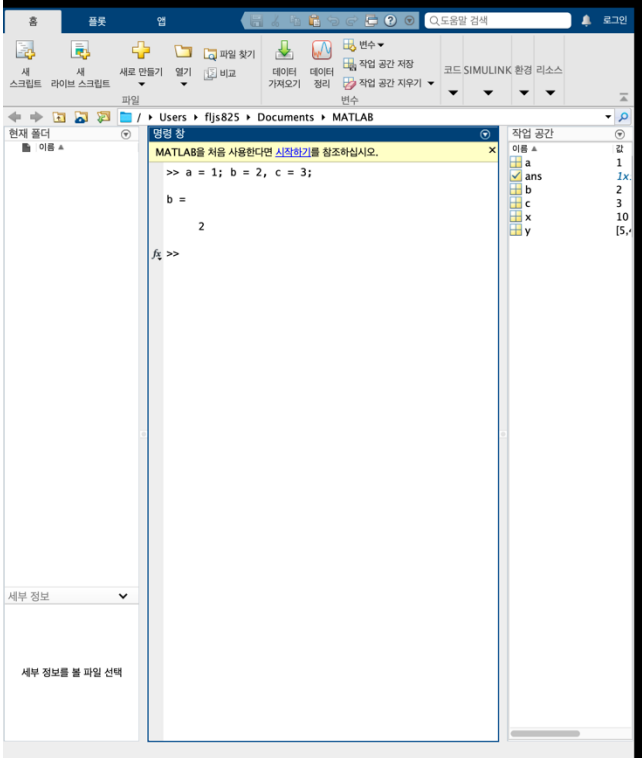

```
a=1;
while a<4
    a=a+1
end
```

위의 코드를 보면, 초기 a는 1로 지정하는 것을 볼 수 있다. while 구문의 첫 번째 조건을 보면, a가 4보다 작을때, 해당 구문 $(a = a + 1)$을 실행하게 된다. 즉, a가 1부터 증가하다가 4가 되는 순간 while문이 끝나게 된다.

```
a =  2
a =  3
a =  4
```

clear, clc, clf

clear는 작업공간(workspace)에 있는 모든 변수의 값을 지워주는 역할을 한다.



clear, clc, clf

clear는 작업공간(workspace)에 있는 모든 변수의 값을 지워주는 역할을 한다. clc는 명령창(command window)를 지워주는 역할을 하며, clf는 그림창(figure)을 지워주는 역할을 한다.

세미콜론 ;

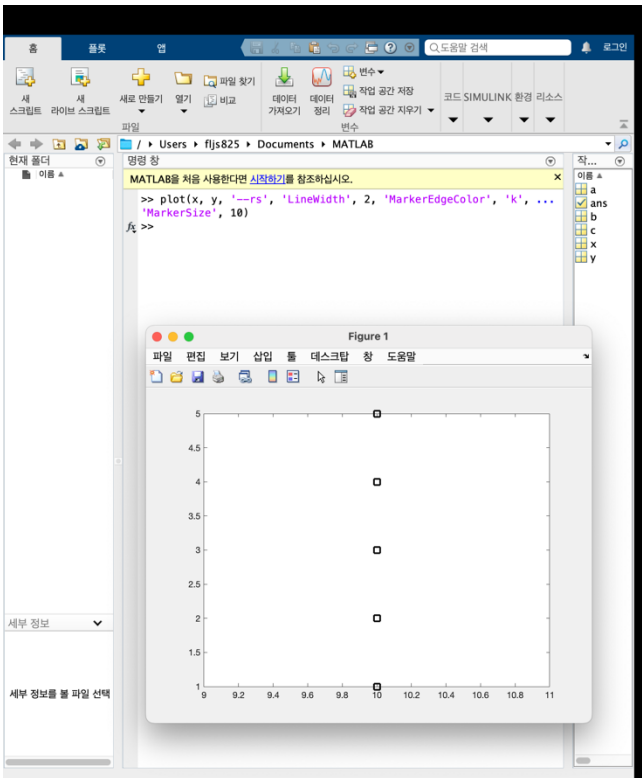
세미콜론을 입력하고 실행하면 명령창(commend window)에는 결과가 출력되지 않고, 작업공간(workspace)에만 값이 저장된다.


```
a=1; b=2, c=3;
b=2
```

명령문 연속 ...

...은 줄을 넘겨주는 역할을 한다. 명령어가 너무 길어 다음 줄로 넘기고 싶을 때 사용한다. 편집기에서뿐만 아니라 작업창에서도 사용가능하다. 주의 할 점은 ' '의 중간을 ...으로 줄 바꿈을 할 경우 오류가 발생한다. 따라서, ' '다음에 ...을 사용해야한다.

```
plot(x,y,'--rs','LineWidth',2,'MarkerEdgeColor','k', ...
```




```
a=1; b=2, c=3;
b=2
```

명령문 연속 ...

...은 줄을 넘겨주는 역할을 한다. 명령어가 너무 길어 다음 줄로 넘기고 싶을 때 사용한다. 편집기에서뿐만 아니라 작업창에서도 사용가능하다. 주의 할 점은 ' '의 중간을 ...으로 줄 바꿈을 할 경우 오류가 발생한다. 따라서, ' '다음에 ...을 사용해야한다.

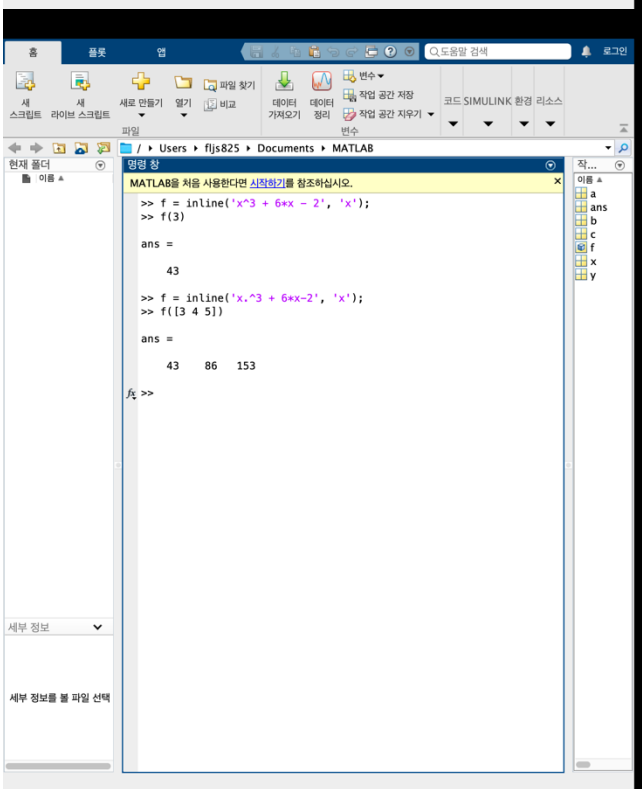
```
plot(x,y,'--rs','LineWidth',2,'MarkerEdgeColor','k', ...
      'MarkerSize',10)
```

inline

inline을 이용하면 간편하게 함수를 정의할 수 있다.

```
f = inline('x^3+6*x-2','x');
f(3)
ans = 43
```

벡터값을 입력으로 받고 벡터값을 출력하는 inline 함수는 다음과 같다.



MATLAB_기본명령어.pdf
11/20페이지

```
      'MarkerSize',10)
```

inline

inline을 이용하면 간편하게 함수를 정의할 수 있다.

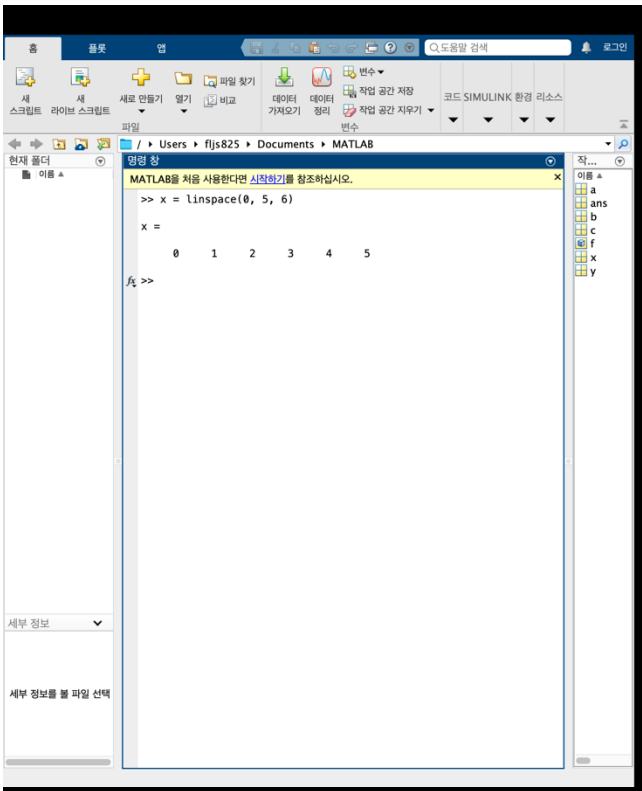
```
f = inline('x^3+6*x-2','x');
f(3)
ans = 43
```

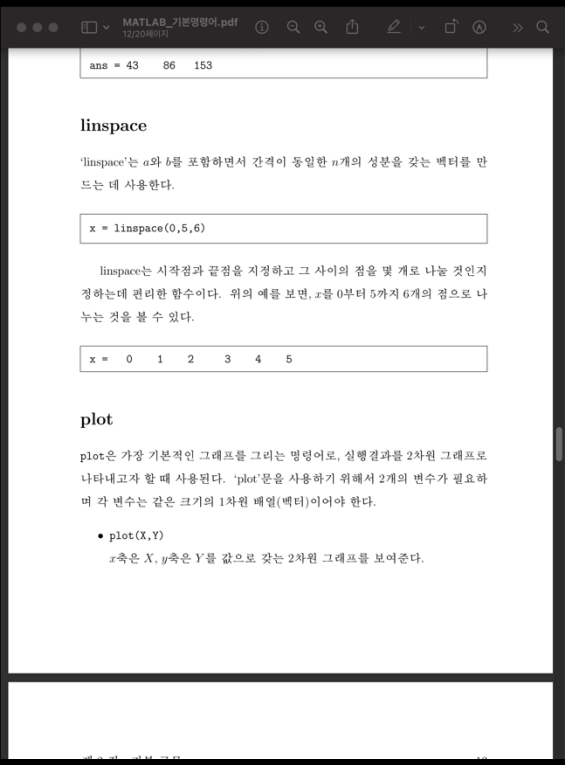
벡터값을 입력으로 받고 벡터값을 출력하는 inline 함수는 다음과 같다.


```
f = inline('x.^3+6*x-2','x');
f([3 4 5])
ans = 43    86    153
```

linspace

'linspace'는 a와 b를 포함하면서 간격이 동일한 n개의 성분을 갖는 벡터를 만드는 데 사용한다.





linspace

'linspace'는 a와 b를 포함하면서 간격이 동일한 n개의 성분을 갖는 벡터를 만드는 데 사용한다.

```
x = linspace(0,5,6)
```

linspace는 시작점과 끝점을 지정하고 그 사이의 점을 몇 개로 나눌 것인지 정하는데 편리한 함수이다. 위의 예를 보면, x를 0부터 5까지 6개의 점으로 나누는 것을 볼 수 있다.

```
x =  0  1  2  3  4  5
```

plot

plot은 가장 기본적인 그래프를 그리는 명령어로, 실행결과를 2차원 그래프로 나타내고자 할 때 사용된다. 'plot'문을 사용하기 위해서 2개의 변수가 필요하며 각 변수는 같은 크기의 1차원 배열(벡터)이어야 한다.

- `plot(X,Y)`
  x축은 X, y축은 Y를 값으로 갖는 2차원 그래프를 보여준다.

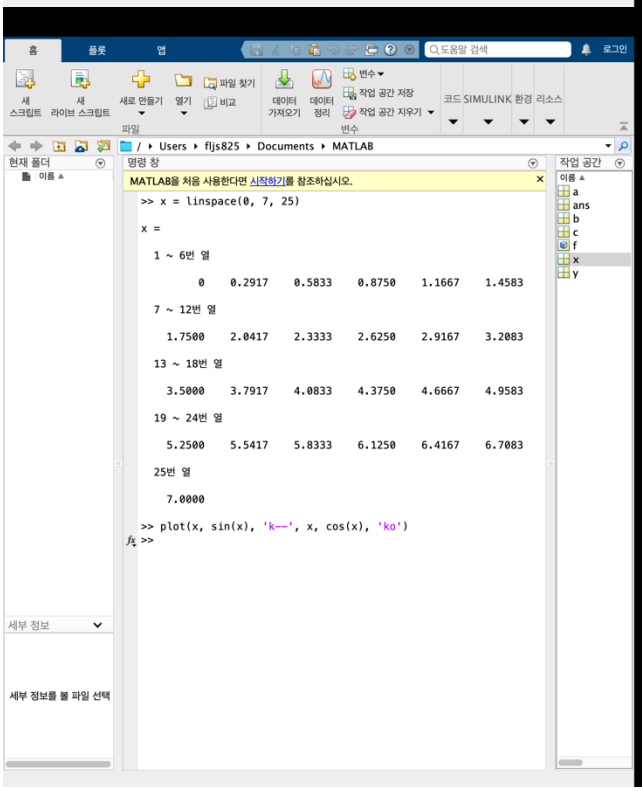


표 1.1: plot 명령어의 옵션

색상		모양		라인	
b	Blue	.	Point	-	Solid
g	Green	o	Circle	:	Dotted
r	Red	x	x mark	-.	Dashdot
c	Cyan	+	Plus	--	Dashed
m	Magenta	*	Star	(none)	No line
y	Yellow	s	Square		
k	Black	d	Diamond		
w	white	v	Triangle(down)		
		^	Triangle(up)		

예를 들어, `plot(x,sin(x),'k--',x,cos(x),'ko')`를 실행하면, [그림 1.1]을 얻게 된다. 이는 y 축의 값을 $\sin(x)$, $\cos(x)$로 하는 두 개의 그래프를 나타내며, 첫 번째 $\sin(x)$는 검은색 점선으로, $\cos(x)$는 검은색 원으로 표현된다.

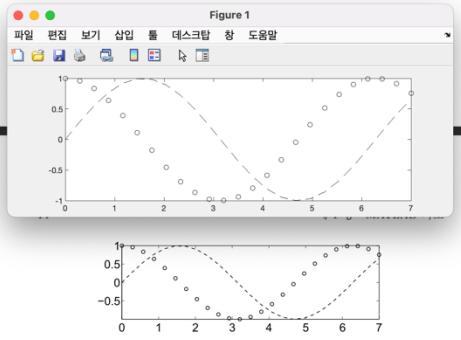
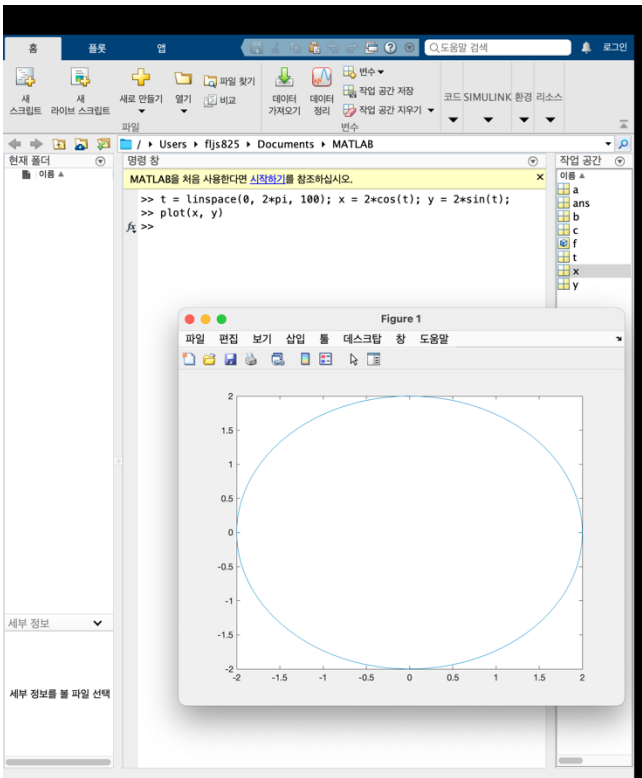


그림 1.1: plot문 옵션을 이용한 프로그램 실행결과

- `title`
  그래프의 제목 넣기



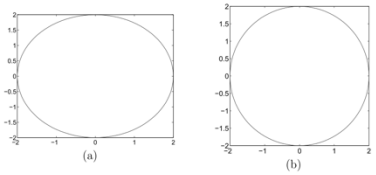
- `grid`
  생성된 그래프에 격자 넣기

- `text`
  지정한 위치에 원하는 글 넣기

- `axis([xmin xmax ymin ymax])`
  그래프의 x축과 y축의 크기를 조절하기

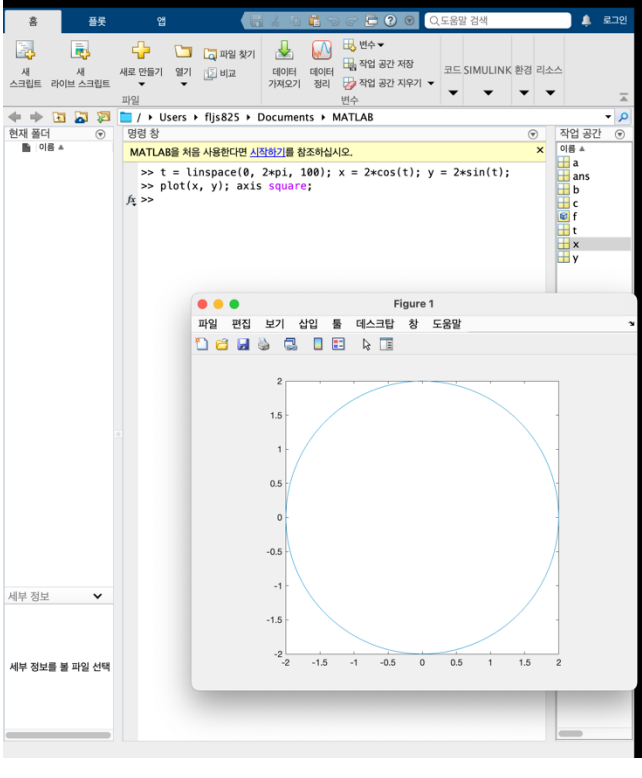
그림을 그릴 때, MATLAB은 자동으로 좌표축을 설정한다. 생성된 그래프의 전체적인 크기 비율은 각각의 좌표축에 대한 aspect ratio에 따라 다르게 나온다. 예를 들어 반지름이 2인 원을 만들어 보자.


```
t=linspace(0,2*pi,100); x=2*cos(t); y=2*sin(t);
plot(x,y)
```

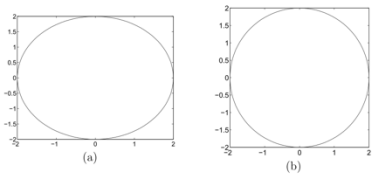


(a) (b)

그림 1.2: (a) 좌표계가 default인 경우. (b) 좌표계가 square인 경우.




```
t=linspace(0,2*pi,100); x=2*cos(t); y=2*sin(t);
plot(x,y)
```

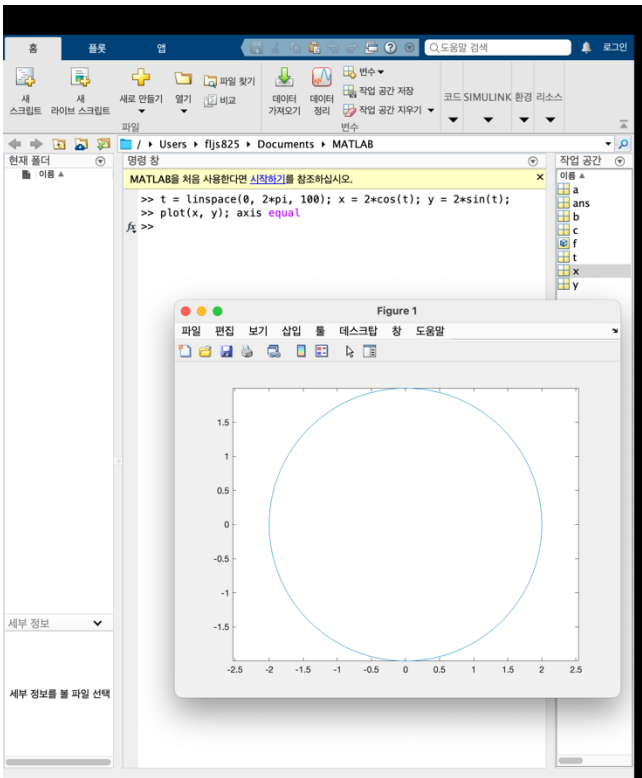
(a) (b)

그림 1.2: (a) 좌표계가 default인 경우. (b) 좌표계가 square인 경우.

그림 1.2(a)은 원의 모양보다는 오히려 타원의 모양을 갖고 있다. 그러므로 원래 원하는 원의 모양을 갖고 싶으면 aspect ratio를 바꾸어 주어야 한다. Aspect ratio의 옵션을 "square"로 하면, Figure window의 모양을 직사각형 모양의 좌표계가 아닌 정사각형의 모양의 좌표계로 바꾼다. 그림 1.2(b)는 그림 1.2(a)보다 훨씬 더 원하는 원의 모양을 갖고 있다. 그러나 이것은 어디까지나 Figure window를 정사각형 모양으로 바꾸어 준 결과로 나타난 것이다. 결국 각 좌표축의 units는 무시한 결과이다.

```
t=linspace(0,2*pi,100); x=2*cos(t); y=2*sin(t);
plot(x,y); axis square;
```

만일 실제 사물의 크기 비율(aspect ratio)와 같게 하려면, 각각의 좌표축에 대한 units는 동등한 크기를 가져야 한다. 이때 이용되는 aspect ratio로

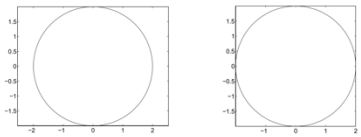


에 대한 units는 동등한 크기를 가져야 한다. 이때 이용되는 aspect ratio로

는 "equal"과 "image"가 있다. 해당 그래프에 대한 데이터의 범위까지만 display하는 경우가 "image"에 해당한다 (그림 1.3 참조).

```
t=linspace(0,2*pi,100); x=2*cos(t); y=2*sin(t);
plot(x,y); axis equal
```

```
t=linspace(0,2*pi,100); x=2*cos(t); y=2*sin(t);
plot(x,y); axis image;
```

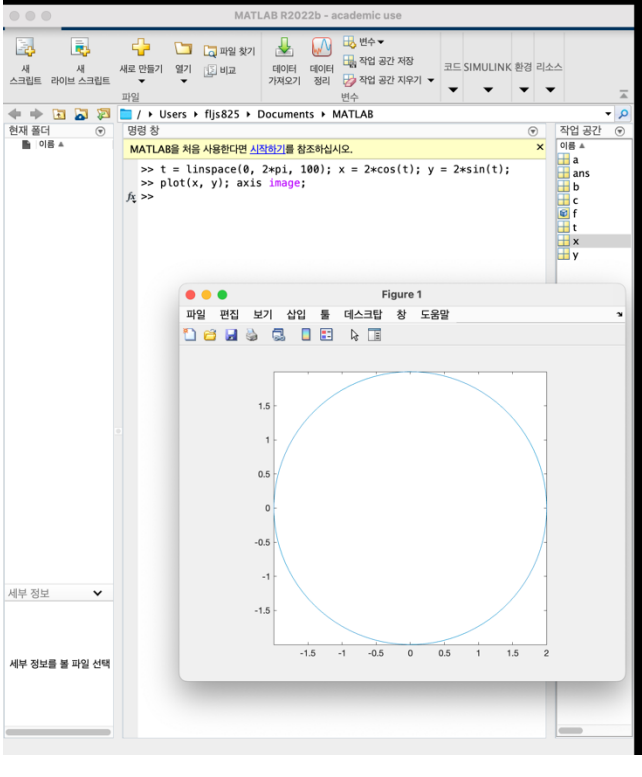


(a) (b)

그림 1.3: (a) 좌표계가 equal인 경우. (b) 좌표계가 image인 경우.

ones

각 원소 값들이 1인 전방행렬을 생성한다.



MATLAB_기본명령어.pdf
16/20페이지

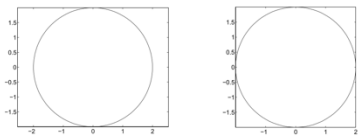
에 대한 units는 동등한 크기를 가져야 한다. 이때 이용되는 aspect ratio로

16 제 1 장 MATLAB 기초

는 "equal"과 "image"가 있다. 해당 그래프에 대한 데이터의 범위까지만 display하는 경우가 "image"에 해당한다 (그림 1.3 참조).

```
t=linspace(0,2*pi,100); x=2*cos(t); y=2*sin(t);
plot(x,y); axis equal
```

```
t=linspace(0,2*pi,100); x=2*cos(t); y=2*sin(t);
plot(x,y); axis image;
```

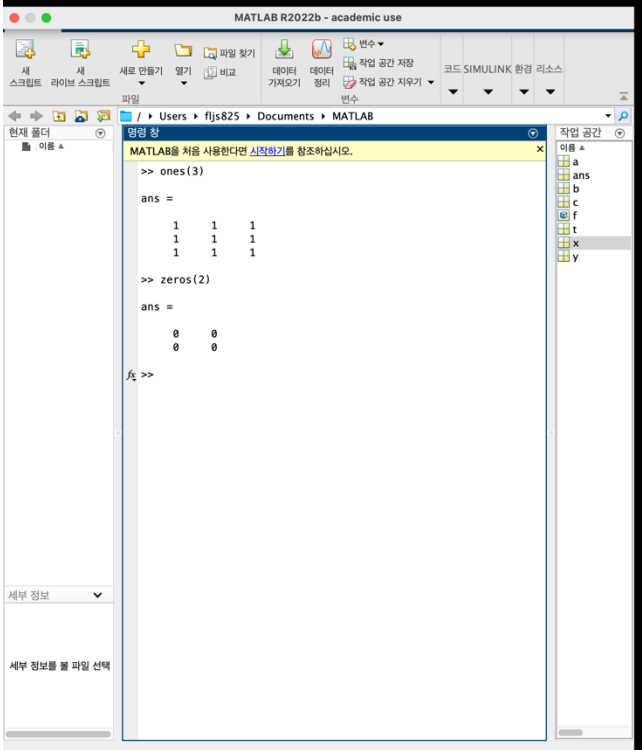


(a) (b)

그림 1.3: (a) 좌표계가 equal인 경우. (b) 좌표계가 image인 경우.

ones

각 원소 값들이 1인 전방행렬을 생성한다.



ones

각 원소 값들이 1인 정방행렬을 생성한다.

```
>> ones(3)
ans = 1     1     1
```

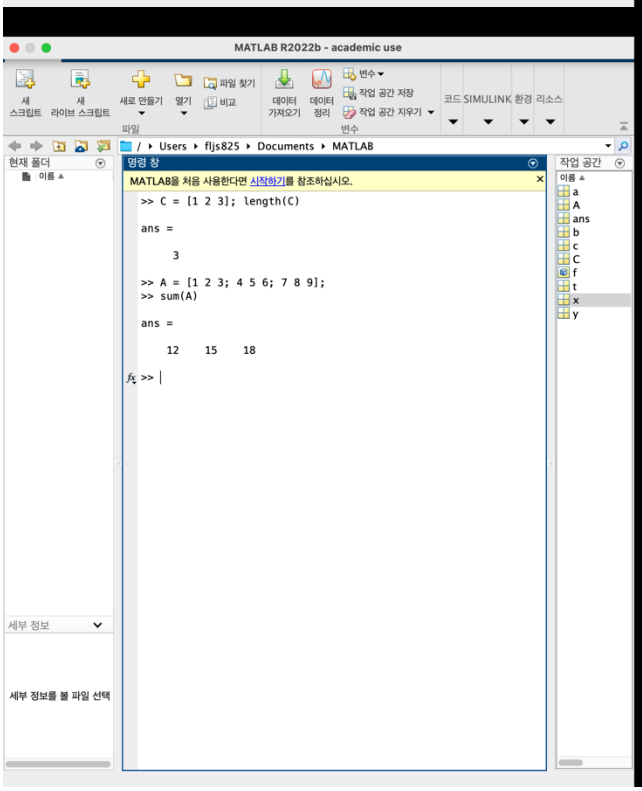


```
     1     1     1
     1     1     1
```

zeros

각 원소 값들이 0인 정방행렬을 생성한다.

```
>> zeros(2)
ans = 0     0
      0     0
```



length

length로 벡터의 길이를 알 수 있다.

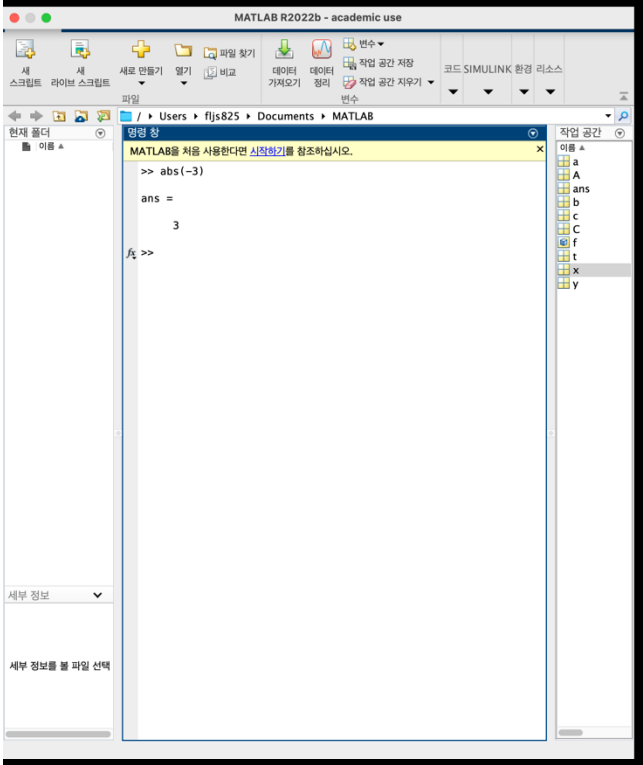
```
>> C=[1 2 3]; length(C)
ans = 3
```


sum

행렬 또는 벡터의 원소들의 합을 구할 수 있다. 벡터의 경우 전체를 더해주게 되며, 행렬의 경우 각 열의 합을 계산하게 된다.

```
>> A=[1 2 3; 4 5 6; 7 8 9];
>> sum(A)
ans = 12    15    18
```

abs



행렬 또는 벡터의 원소들의 합을 구할 수 있다. 벡터의 경우 전체를 더해주게 되며, 행렬의 경우 각 열의 합을 계산하게 된다.

```
>> A=[1 2 3; 4 5 6; 7 8 9];
>> sum(A)
ans = 12    15    18
```

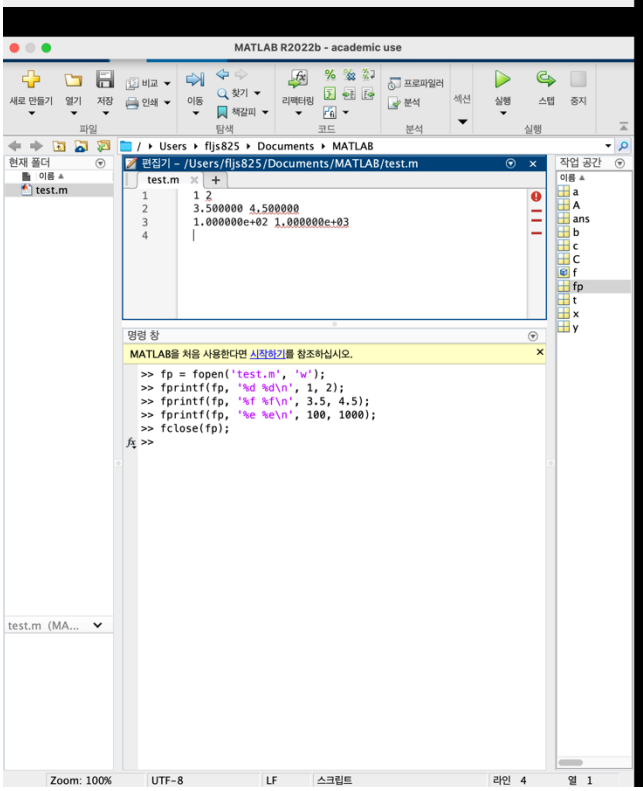
abs

abs(a)를 사용하면 a의 절댓값을 얻을 수 있다.

```
>> abs(-3)
 ans = 3
```

fprintf

파일을 입출력할 때 사용하는 fprintf 함수에 대해서 알아보자. 이 함수의 옵션은 %d, %f, %e가 있는데, 이는 형식 변환 문자로서 %d는 부호가 있는 10진수를 출력하고, %f는 부동 소수점으로 출력하며, %e는 e의 거듭제곱 형태로 출력을 한다.



파일을 입출력할 때 사용하는 fprintf 함수에 대해서 알아보자. 이 함수의 옵션은 %d, %f, %e가 있는데, 이는 형식 변환 문자로서 %d는 부호가 있는 10진수를 출력하고, %f는 부동 소수점으로 출력하며, %e는 e의 거듭제곱 형태로 출력을 한다.

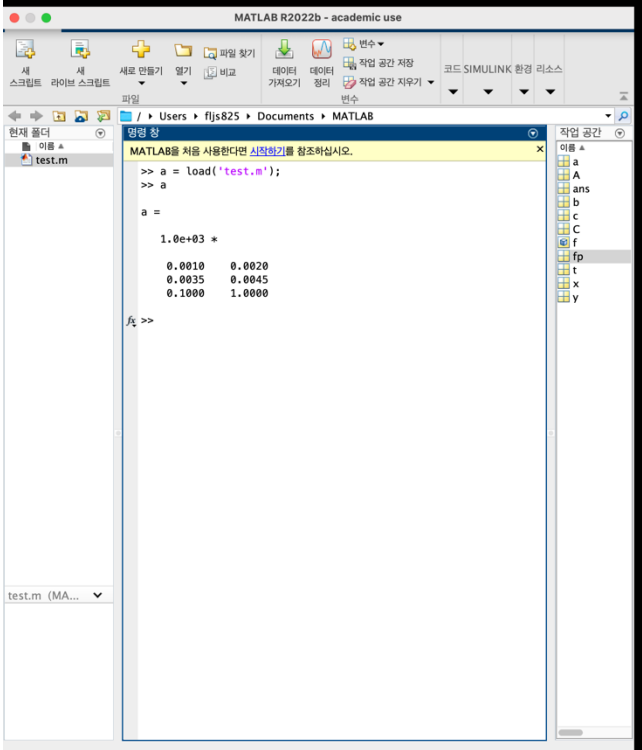

```
fp = fopen('test.m','w'); %test.m란 파일을 쓰기용으로 생성
fprintf(fp, '%d %d\n', 1, 2);        %파일에 1 2 쓰기
fprintf(fp, '%f %f\n', 3.5, 4.5);    %파일에 3.5 4.5 쓰기
fprintf(fp, '%e %e\n', 100, 1000);   %파일에 100 1000 쓰기
fclose(fp); %파일 close
```

```
1 2
3.500000 4.500000
1.000000e+002 1.000000e+003
```

load

load 함수는 파일에 저장된 데이터를 가져올 수 있다. 단, 파일에 문자가 들어있으면 안 된다.

```
a = load('test.m');
```




```
1 2
3.500000 4.500000
1.000000e+002 1.000000e+003
```

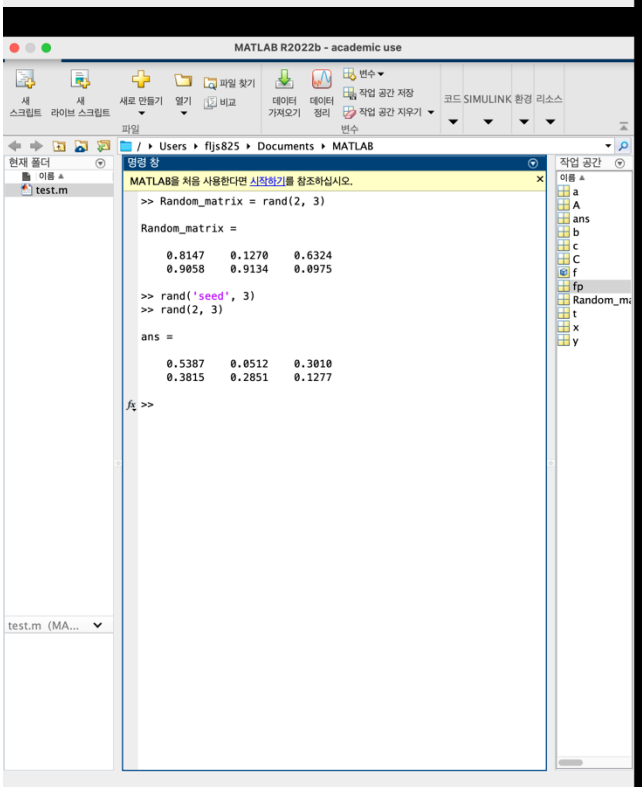
load

load 함수는 파일에 저장된 데이터를 가져올 수 있다. 단, 파일에 문자가 들어 있으면 안 된다.

```
a = load('test.m');
```

```
a =  1.0e+003 *
    0.0010    0.0020
    0.0035    0.0045
    0.1000    1.0000
```


rand() 함수



20 제 1 장 MATLAB 기초

rand() 함수

MATLAB의 rand(M,N) 명령어는 0과 1사이의 무작위의 수로 이루어진 M×N 행렬을 생성한다.

```
Random_matrix = rand(2,3)
```

```
Random_matrix =
    0.5359    0.8579    0.5402
    0.0354    0.6158    0.6218
```

따라서, rand을 사용할 때마다 다른 결과를 얻는다. 그러나 rand로 생성되는 값을 임의로 똑같이 출력할 수가 있다. rand(M,N)을 하기 전에 seed를 정의해 주면 된다.

```
rand('seed',3)
rand(2,3)
```

```
ans = 0.5387    0.0512    0.3010
      0.3815    0.2851    0.1277
```