

데이터 분석

기술 통계

❖pandas의 기술 통계

- ✓ count, min, max, sum, mean, median, var, std
- ✓ argmin(최소값 위치), argmax, idxmin(최소값 색인), idxmax, quantile(사분위수)
- ✓ describe(요약) – Series로 리턴
- ✓ Cumsum(누적합), cummin, cummax, cumprod
- ✓ Diff(산술 차)
- ✓ Pct_change: 이전 데이터와의 차이로 -1을 대입하면 반대로
- ✓ Unique(): 동일한 값을 제외한 배열 리턴 – Series에만 사용
- ✓ Value_counts(): 도수를 리턴, 기본적으로 내림차순 정렬을 수행하며 sort 속성에 False를 대입하면 정렬하지 않습니다. – Series에만 사용

기술 통계

❖ tdata.csv

번호,성적

1,77

2,85

3,63

4,69

5,82

6,78

7,73

8,87

9,65

10,92

```
=====
count      50.00000
mean       74.90000
std        14.13961
min        37.00000
25%        65.00000
50%        77.00000
75%        85.00000
max        100.00000
Name: 성적, dtype: float64
=====
빈도 수: 77      5
65      4
87      4
73      4
92      4
69      3
76      3
77      2
```

기술 통계

```
from pandas import Series, DataFrame
import pandas as pd
import numpy as np
items = pd.read_csv('tdata.csv', encoding='ms949')
print(items) #데이터 출력
print('=====')
print(items['성적'].describe()) #데이터 요약
print('=====')
print('빈도 수:', items['성적'].value_counts()) #빈도 수
```

기술 통계

❖ 기술 통계

- ✓ 산술 평균: 합의 평균
- ✓ 기하 평균: 평균 비율을 구할 때 사용하는 평균으로 각 데이터의 비율을 곱한 후 제곱근을 구하는 것

Ex) 매출액이 100 인 회사에서 다음 해에 110을 기록했고 그 다음해에 107.8을 기록했을 때 연 평균 성장률은?

첫해 10프로 인상됐고 두번째 해에 2프로 감소 했으므로 $10 - 2/2 \Rightarrow 4$ 프로
 $1.1 * 0.98$ 의 제곱근

- ✓ 조화평균: 2개의 수의 곱에 2를 곱한 후 2개의 수를 더한 값으로 나누는 것으로 속도의 평균을 구할 때 많이 사용

Ex) 동일한 거리를 시속 100km로 한 번 가고 60km로 갔을 때 평균 속도는?

$$2 * 100 * 60 / 100 + 60$$

기술 통계

- ❖ 매출액이 10 인 상태에서 다음에는 11 그리고 그 다음에는 10.78 인 경우의 평균 성장률 계산

평균 성장률?: 0.0400000000000000036

평균 성장률(기하평균): 1.0382677881933928

- ❖ 동일한 거리를 한번은 시속 100으로 한번은 시속 60으로 갔을 때의 평균 속도

평균 속도?: 80.0

평균 속도(조화평균): 75.0

기술 통계

```
from pandas import Series, DataFrame
import pandas as pd
import numpy as np
import math

s = Series([10, 11, 10.78])
print("평균 성장률?:", s.pct_change().mean())
print("평균 성장률(기하평균):", math.sqrt((11/10)*(10.78/11)))
print("평균 속도?:", (100+60) / 2)
print("평균 속도(조화평균):", 2*100*60 / (100+60))
```

기술 통계

❖ 기술 통계

- ✓ 표준화: 모든 값들의 표준 값을 정해서 그 값을 기준으로 차이를 구해서 비교하는 방법
- ✓ 표준 값: $(\text{데이터} - \text{평균}) / \text{표준편차}$
- ✓ 편차 값: $\text{표준값} * 10 + 50$
- ✓ 표준 값들의 평균은 0, 표준편차는 1이다.(편차 값들의 평균은 50, 표준편차는 10이다.)
 - 표준점수 0.0(=편차치 50) 이상은 전체의 50%이다.
 - 표준점수 1.0(=편차치 60) 이상은 전체의 15.866%이다.
 - 표준점수 2.0(=편차치 70) 이상은 전체의 2.275%이다.
 - 표준점수 3.0(=편차치 80) 이상은 전체의 0.13499%이다.
 - 표준점수 4.0(=편차치 90) 이상은 전체의 0.00315%이다.
 - 표준점수 5.0(=편차치 100) 이상은 전체의 0.00002%이다.

기술 통계

```
from pandas import Series, DataFrame
import pandas as pd
import numpy as np
import math

df = pd.read_csv("student.csv", encoding='ms949')
df.index = df['이름']
df = df.drop('이름', axis=1)
kormean, korstd = df['국어'].mean(), df['국어'].std()
engmean, engstd = df['영어'].mean(), df['영어'].std()
matmean, matstd = df['수학'].mean(), df['수학'].std()
```

기술 통계

```
df['국어표준값'] = (df['국어'] - kormean)/korstd  
df['영어표준값'] = (df['영어'] - engmean)/engstd  
df['수학표준값'] = (df['수학'] - matmean)/matstd
```

```
df['국어편차값'] = df['국어표준값'] * 10 + 50  
df['영어편차값'] = df['영어표준값'] * 10 + 50  
df['수학편차값'] = df['수학표준값'] * 10 + 50
```

```
print(df)
```

표본 추출

❖ 표본 추출

- ✓ 구글의 사용자들이 검색하는 질의를 분석하는 경우에 특정 알고리즘이 검색을 개선하는지를 알아보기 위해서 사용자들이 입력한 모든 질의를 분석하는 것은 낭비에 가깝습니다.
- ✓ 위의 경우 특정 기간에 있었던 질의만 분석한다던가 특정 조건을 만족하는 질의만 분석하는 것이 효율적일 것입니다.
- ✓ 이처럼 전체 데이터(모집단) 중 일부를 표본(샘플)으로 추출하는 작업이 데이터 분석에서는 필수입니다.
- ✓ 보통은 훈련 데이터와 테스트 데이터를 80% 와 20%로 분리하여 데이터에 대한 모델링은 훈련 데이터로만 수행하고 모델의 성능은 테스트 데이터로 평가하면 모델의 성능을 가장 적절히 평가할 수 있습니다.
- ✓ 데이터의 분포가 일정하지 않다면 가중치를 이용해서 데이터를 추출하는 부분도 고려해야 합니다.

표본 추출

❖ 표본 추출

- ✓ 단순 임의 추출은 전체 데이터에서 각 데이터를 추출할 확률을 동일하게 하여 표본을 추출하는 방법으로 복원 추출과 비복원 추출이 있습니다.
- ✓ 복원 추출은 한 번 추출된 표본을 다시 선택하는 것이 가능한 경우를 의미하며 비복원 추출은 한 번 추출된 표본은 다시 선택할 수 없는 경우를 의미합니다.
- ✓ 파이썬에서 복원 추출은 `random.random` 함수를 이용하며 비복원 추출은 `random.sample` 함수를 이용합니다.
- ✓ 시퀀스 자료형의 순서를 무작위로 배치하는 `shuffle`(시퀀스 자료형) 함수도 있습니다.
- ✓ `random.random()` 함수는 0 이상 1 미만의 숫자 중에서 하나를 리턴하고 `randrange`(시작위치, 종료위치) 나 `randint` (시작위치, 종료위치) 함수를 이용하면 특정 범위 내의 정수를 리턴합니다.

표본 추출

❖ 표본 추출

- ✓ random.sample 함수는 비복원 추출이 가능한 함수인데 매개변수로는 데이터와 추출할 개수를 넘겨주면 됩니다.

```
import random
li = [10,20,30,40,50]
#복원 추출
for i in range(5):
    print(li[random.randint(0, len(li)-1)], end=' ')
print()
#비복원 추출
print(random.sample(li, k=5))
```

표본 추출

❖ 표본 추출

- ✓ numpy 라이브러리에도 동일한 함수가 존재
- ✓ numpy 라이브러리의 random.choice를 이용하면 가중치를 이용한 데이터 추출도 가능
- ✓ 데이터의 분포가 일정하지 않은 경우 일반 추출을 수행하게 되면 빈도가 적은 데이터가 많이 나올 수 있습니다.
- ✓ choice(a, size=None, replace=True, p=None): a는 배열이고 size는 개수이며 replace는 복원 여부이고 p는 확률

가중치를 고려한 추출

```
arr = ['Hello', 'Hi', 'Good', 'Nice']
```

```
print(np.random.choice(arr, 5, p=[0.5, 0.1, 0.1, 0.3]))
```

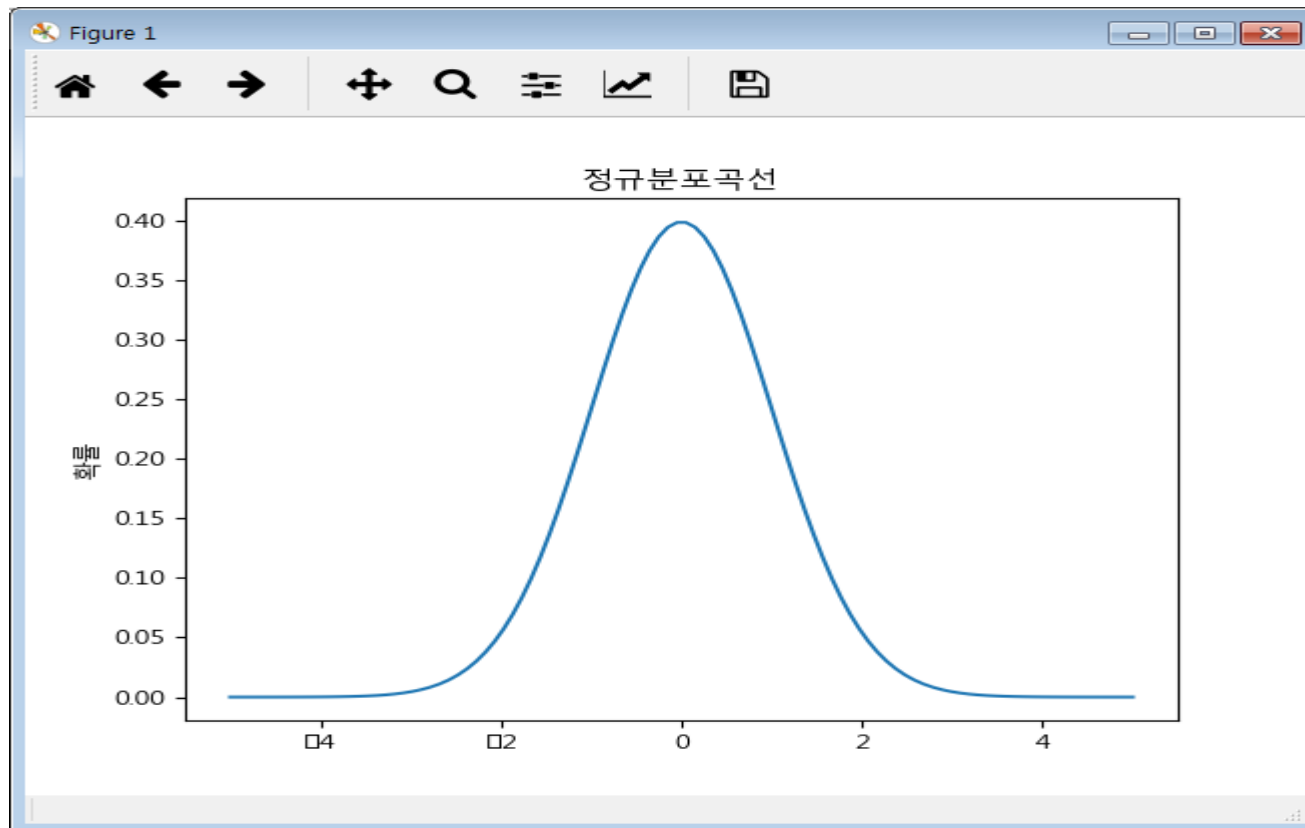
정규 분포

❖ 정규 분포

- ✓ 가우시안 정규 분포(Gaussian normal distribution), 혹은 그냥 간단히 정규 분포라고 부르는 분포는 자연 현상에서 나타나는 숫자를 확률 모형으로 모형화할 때 가장 많이 사용되는 모형
- ✓ 데이터들이 평균 값을 기준으로 좌 우 대칭형으로 분포되어 있는 형태
- ✓ 평균 값을 기준으로 표준편차 1배 안에 전체 데이터의 약 70% 이상이 몰려 있고 약 1.96배 안에 95%이상이 분포된 경우
- ✓ scipy의 stats 서브 패키지에 있는 norm 클래스는 정규 분포에 대한 클래스 인데 loc 인수로 평균을 설정하고 scale 인수로 표준 편차를 설정
- ✓ 시뮬레이션을 통해 샘플을 얻으려면 rvs 메서드를 사용

정규 분포

❖ 평균이 0이고 표준 편차가 1인 정규 분포



정규 분포

❖평균이 0이고 표준 편차가 1인 정규 분포에서 100개의 데이터 샘플링

[-0.18866368 0.33298835 -0.38338861 -0.58630201 -0.0352717 0.83131306 0.45430349
1.00497603 0.88857198 1.16386041 0.98541524 -0.08807121 -0.6247982 0.3660567 -
0.62156248 1.05328769 -0.33809822 -0.74267457
0.68693788 0.56864614 -0.83377739 0.75246509 1.41831302 0.60342366
-2.2525986 0.23723631 -1.23366991 0.84888304 -0.23597921 -0.22991797
-0.55103535 0.15477685 -0.43566131 0.46150938 0.51121478 0.14369775
-1.65575749 1.66535737 -0.97471866 -3.52622788 -0.56315723 0.60597605
0.36105992 -0.24231248 0.04772834 0.09526121 -0.01955942 0.47597838

정규 분포

```
from pandas import Series, DataFrame
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import font_manager, rc
from scipy import stats
import scipy as sp
```

```
font_name =
font_manager.FontProperties(fname="c:/Windows/Fonts/malgun.ttf").get_name()
rc('font', family=font_name)
```

정규 분포

```
mu = 0
std = 1
rv = sp.stats.norm(mu, std)
xx = np.linspace(-5, 5, 100)
plt.plot(xx, rv.pdf(xx))
plt.ylabel("확률")
plt.title("정규분포곡선")
plt.show()
x = rv.rvs(100)
print(x)
```

검정 통계

- ❖ 검정(testing)은 데이터 뒤에 숨어있는 확률 변수의 분포와 모수에 대한 가설의 진위를 정량적(quantitatively)으로 증명하는 작업
- ❖ 통계처리를 할 때 대개는 모집단의 분산이나 평균을 알기가 어려운데 이유는 모집단 자체를 전수조사하기가 어렵기 때문이기도 하고 모집단의 정확한 범위를 알지 못 하는 경우도 있고 그래서 실제 모집단에서 표본 몇 십 개 또는 몇 백 개를 추출해서 그것의 분산(표본분산)이나 평균(표본평균)을 사용해야 하는 경우가 대부분입니다.
- ❖ 표본수가 크지 않을 때 표본분산(표본표준편차)을 사용한 테스트는 t-분포를 이용한다고 해서 T-test라고 합니다.
- ❖ 가설 증명 즉 검정의 기본적인 논리는 다음과 같습니다.
 - ✓ 만약 가설이 맞다면 즉, 모수 값이 특정한 조건을 만족한다면 해당 확률 변수로부터 만들어진 표본(sample) 데이터들은 어떤 규칙을 따르게 된다.

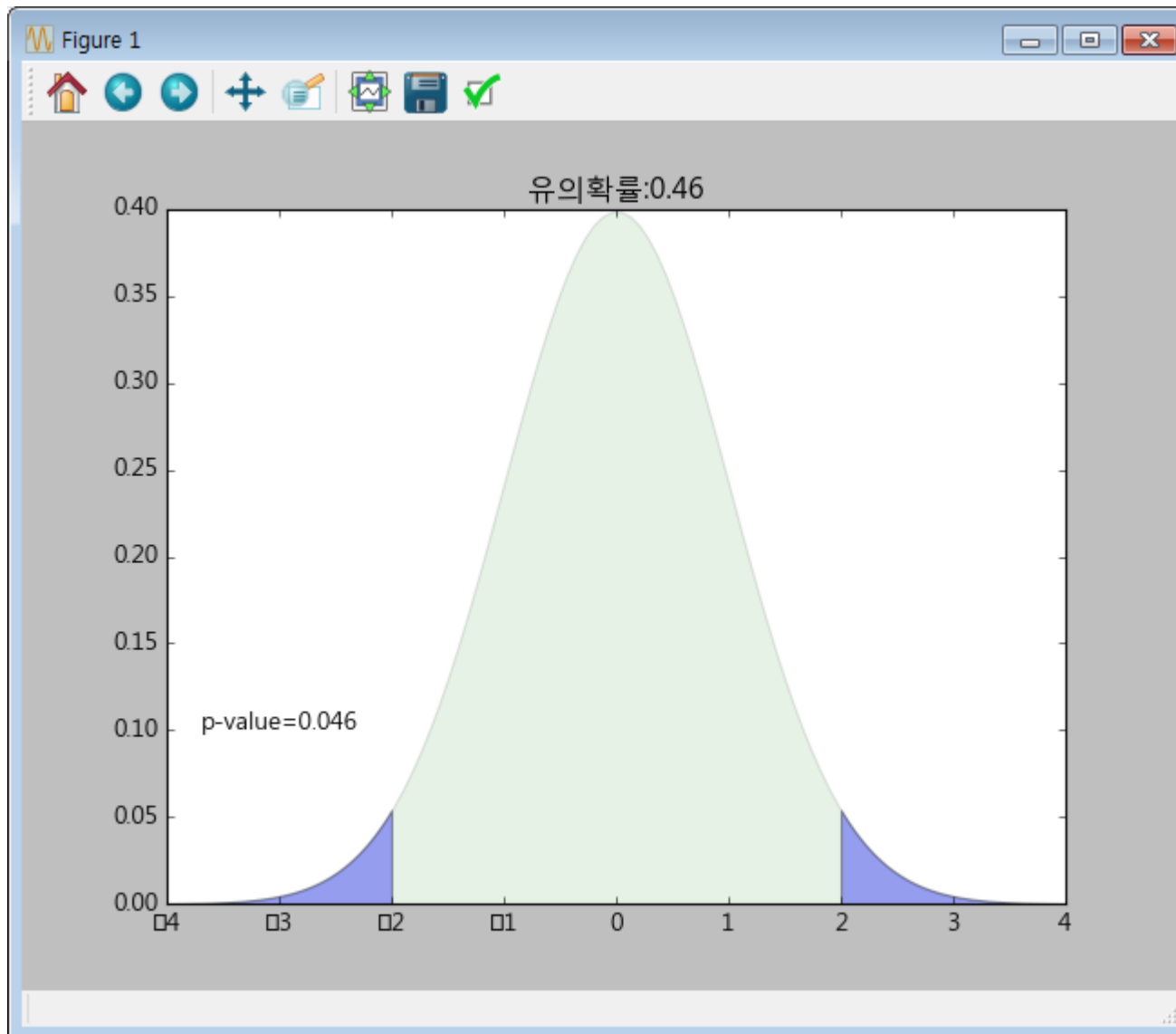
검정 통계

- ✓ 해당 규칙에 따라 표본 데이터 집합에서 어떤 숫자를 계산하면 계산된 숫자는 특정한 확률 분포를 따르게 된다. 이 숫자를 검정 통계치(test statistics)라고 하며 확률 분포를 검정 통계 분포(test statistics distribution)라고 한다. 검정 통계 분포의 종류 및 모수의 값은 처음에 정한 가설에 의해 결정된다. 이렇게 검정 통계 분포를 결정하는 최초의 가설을 귀무 가설(Null hypothesis)이라고 한다.
- ✓ 데이터에 의해서 실제로 계산된 숫자, 즉, 검정 통계치가 해당 검정 통계 분포에서 나올 수 있는 확률을 계산한다. 이를 유의 확률(p-value)라고 한다.
- ✓ 만약 유의 확률이 미리 정한 특정한 기준 값보다 작은 경우를 생각하자. 이 기준 값을 유의 수준(significance level)이라고 하는 데 보통 1% 혹은 5% 정도의 작은 값을 지정한다. 유의 확률이 유의 수준으로 정한 값(예 1%)보다도 작다는 말은 해당 검정 통계 분포에서 이 검정 통계치가 나올 수 있는 확률이 아주 작다는 의미이므로 가장 근본이 되는 가설 즉, 귀무 가설이 틀렸다는 의미이다. 따라서 이 경우에는 귀무 가설을 기각(reject)한다.
- ✓ 만약 유의 확률이 유의 수준보다 크다면 해당 검정 통계 분포에서 이 검정 통계치가 나오는 것이 불가능하지만은 않다는 의미이므로 귀무 가설을 기각할 수 없다. 따라서 이 경우에는 귀무 가설을 채택(accept)한다.

검정 통계

- ❖ 유의수준(significance level)은 보통 1%, 5%, 10% 세 개를 주로 사용하는데 0.01, 0.05, 0.1.... 그 중에서 1%와 5%를 많이 사용합니다.
- ❖ 유의수준 0.05라 함은 두 개 집단의 모평균은 실제 같은데도 잘못해서 귀무 가설을 기각하게 될 확률을 의미합니다.
- ❖ 5%(0.05)의 유의성이란 테스트 결과가 "사실이 아닐 확률"이 5%라는 뜻....또는 "사실일 확률"이 95% 라는 뜻입니다.
- ❖ 유의확률은 소위 p-value라고 하는 것으로 귀무 가설을 기각할 수 있는 최소한의 확률을 의미합니다.
- ❖ 유의확률이 0.009로 도출되었다고 할 때 유의확률은 위의 유의수준을 보다 정확히 계산한 것으로 귀무 가설을 잘못 기각할 확률 0.9% 밖에 안 된다는 겁니다.
- ❖ 여기서는 0.009(0.9%)니까, 5%는 물론 1%의 유의수준에서 귀무 가설을 기각할 수 있다는 뜻입니다.

검정 통계



검정 통계

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import font_manager, rc
from scipy import stats
import scipy as sp

font_name =
font_manager.FontProperties(fname="c:/Windows/Fonts/malgun.ttf").get_name()
rc('font', family=font_name)

xx1 = np.linspace(-4, 4, 100)
xx2 = np.linspace(-4, -2, 100)
xx3 = np.linspace(2, 4, 100)
```


검정 통계

```
plt.fill_between(xx1, sp.stats.norm.pdf(xx1), facecolor='green', alpha=0.1)
plt.fill_between(xx2, sp.stats.norm.pdf(xx2), facecolor='blue', alpha=0.35)
plt.fill_between(xx3, sp.stats.norm.pdf(xx3), facecolor='blue', alpha=0.35)
plt.text(-3, 0.1, "p-value=%5.3f" % (2*sp.stats.norm.cdf(-2)),
horizontalalignment='center')
plt.title("유의확률:0.46")

plt.show()
```

검정 통계

- ❖ 베르누이 검정: 모수 θ 를 가지는 베르누이 분포 확률 변수에 대해서는 전체 시도 횟수 N 번 중 성공한 횟수 n 자체를 검정 통계량으로 사용하는 것으로 자유도 N 과 모수 θ 를 가지는 이항 분포를 따름
- ❖ `1- scipy.stats.binom(시도횟수, 성공확률).cdf(성공횟수-1)`로 유의 확률을 구합니다.
- ❖ 게임에서 내가 이길 확률은 0.3이고 100번 했을 때 30번 이길 유의확률은?
그리고 이것이 가능한 것인지 유의수준 5%로 검정
- ❖ 게임에서 내가 이길 확률은 0.3이고 100번 했을 때 60번 이길 유의확률은?
그리고 이것이 가능한 것인지 유의수준 5%로 검정

검정 통계

```
from scipy import stats
```

```
import scipy as sp
```

```
r = 1-sp.stats.binom(100, 0.3).cdf(30 - 1)
```

```
if r > 0.05:
```

```
    print("p-value가 0.05보다 크므로 정상적인 상황입니다..")
```

```
else:
```

```
    print("p-value가 0.05보다 작으므로 발생할 가능성이 낮은 상황입니다..")
```

```
r = 1-sp.stats.binom(100, 0.3).cdf(60 - 1)
```

```
if r > 0.05:
```

```
    print("p-value가 0.05보다 크므로 정상적인 상황입니다..")
```

```
else:
```

```
    print("p-value가 0.05보다 작으므로 발생할 가능성이 낮은 상황입니다..")
```

검정 통계

- ❖ 단일 표본 t-검정은 정규 분포의 표본에 대해 기댓값을 조사하는 검정방법
- ❖ 데이터의 표준편차나 분산을 아는 경우에 아래 함수를 이용해서 유의확률을 구합니다.

`scipy.stats.t(df=데이터개수-1).cdf(평균/표준편차*데이터개수의 제곱근)`

- ❖ **평균/표준편차*데이터개수의 제곱근**을 검정 통계량이라고 합니다.

검정 통계

❖ tdata.csv

번호,성적

1,77

2,85

3,63

4,69

5,82

6,78

7,73

8,87

9,65

10,92

검정 통계

```
import matplotlib.pyplot as plt
```

```
import pandas as pd
```

```
import numpy as np
```

```
from scipy import stats
```

```
import scipy as sp
```

```
items = pd.read_csv('tdata.csv', encoding='ms949')
```

```
#평균이 75라고 할 수 있는지 유의수준 5%로 검정
```

```
items['성적'] = items['성적']/75-1
```

검정 통계

```
t = items['성적'].mean()/items['성적'].std(ddof=1)*np.sqrt(len(items['성적']))
print("검정통계:", t)
result = 1-sp.stats.t(df=len(items['성적'])-1).cdf(t)
print("유의확률:", result)
if result >= 0.05:
    print("평균은 75라고 할 수 있습니다.")
else:
    print("평균은 75라고 할 수 없습니다.")
```

검정 통계

- ❖ 독립 표본 t-검정(Independent-two-sample t-test)은 간단하게 two sample t-검정이라고도 합니다.
- ❖ 두 개의 독립적인 정규 분포에서 나온 두 개의 데이터 셋을 사용하여 두 정규 분포의 기댓값이 동일한지를 검사합니다.
- ❖ scipy stats 서브패키지의 ttest_ind 함수를 사용
- ❖ 독립 표본 t-검정은 두 정규 분포의 분산 값이 같은 경우와 같지 않은 경우에 사용하는 검정 통계량이 다르기 때문에 equal_var 인수를 사용하여 이를 지정해 주어야 합니다.
- ❖ 서로 다른 10명의 사람에게 수면제1을 복용했을 때의 수면 증가 시간은 [0.7,-1.6,-0.2,-1.2,-0.1,3.4,3.7,0.8,0.0,2.0] 이고 수면제2를 복용했을 때의 수면 증가 시간은 [1.9,0.8,1.1,0.1,-0.1,4.4,5.5,1.6,4.6,3.4] 인 경우 2가지 약 복용 시 수면 증가 시간은 차이가 없는지 유의확률 5%로 검정

검정 통계

```
import numpy as np
from scipy import stats
import scipy as sp

x1 = np.array([0.7,-1.6,-0.2,-1.2,-0.1,3.4,3.7,0.8,0.0,2.0]);
x2 = np.array([1.9,0.8,1.1,0.1,-0.1,4.4,5.5,1.6,4.6,3.4]);
r = sp.stats.ttest_ind(x1, x2, equal_var=True)
print(x1.var())

if r.pvalue >= 0.05:
    print("2가지 약의 평균 수면 증가시간은 같다.")
else:
    print("2가지 약의 평균 수면 증가시간은 다르다.")
```

검정 통계

- ❖ 대응 표본 t-검정은 독립 표본 t-검정을 두 집단의 샘플이 1대1 대응하는 경우에 대해 수정한 것
- ❖ 독립 표본 t-검정과 마찬가지로 두 정규 분포의 기댓값이 같은지 확인하기 위한 검정
- ❖ 예를 들어 어떤 반의 학생들이 특강을 수강하기 전과 수강한 이후에 각각 시험을 본 시험 점수의 경우에는 같은 학생의 두 점수는 대응
- ❖ 이 대응 정보를 알고 있다면 보통의 독립 표본 t-검정에서 발생할 수 있는 샘플간의 차이의 영향을 없앨 수 있기 때문에 특강 수강의 영향을 보다 정확하게 추정할 수 있습니다.
- ❖ scipy stats 서브패키지의 `ttest_rel` 함수를 사용

검정 통계

```
import numpy as np
from scipy import stats
import scipy as sp

x1 = np.array([0.7,-1.6,-0.2,-1.2,-0.1,3.4,3.7,0.8,0.0,2.0]);
x2 = np.array([1.9,0.8,1.1,0.1,-0.1,4.4,5.5,1.6,4.6,3.4]);
r = sp.stats.ttest_rel(x1, x2)
print(x1.var())

if r.pvalue >= 0.05:
    print("2가지 약의 평균 수면 증가시간은 같다.")
else:
    print("2가지 약의 평균 수면 증가시간은 다르다.")
```

상관관계

- ❖ 두 변수 간에 어떤 선형적 관계가 있는지 분석하는 것을 이를 상관 분석 (Correlation Analysis)이라고 합니다.
- ❖ 예를 들면 교육수준과 월 수입 간의 관계
- ❖ 공분산은 결합 분포의 평균을 중심으로 각 자료들이 어떻게 분포되어 있는지를 보여줍니다.

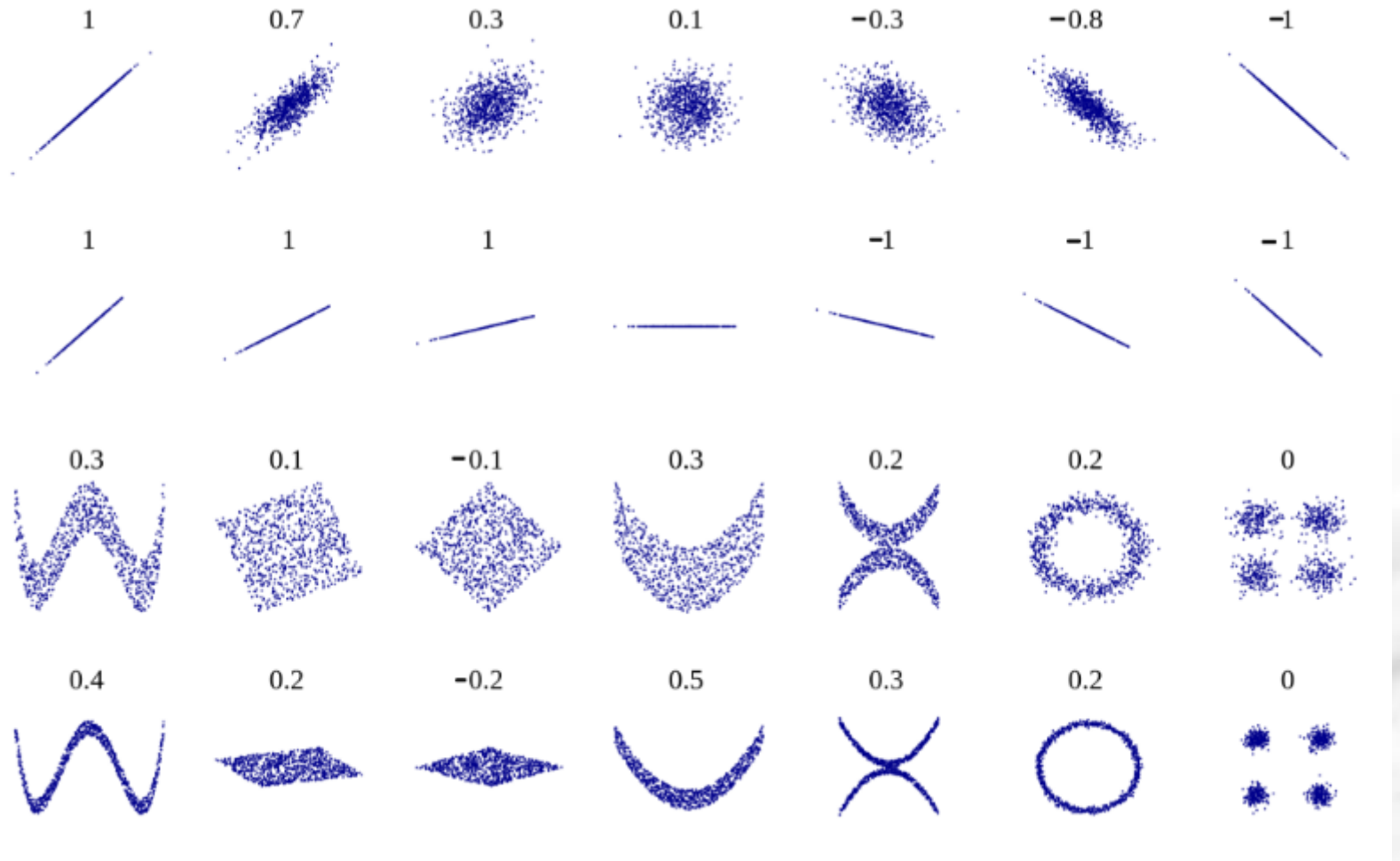
샘플 공분산(sample covariance)은 다음과 같이 정의된다. 여기에서 x_i 와 y_i 는 각각 i 번째의 x 자료와 y 자료의 값을 가리키고, m_x 와 m_y 는 x 자료와 y 자료의 샘플 평균을 가리킨다.

$$s_{xy}^2 = \frac{1}{N} \sum_{i=1}^N (x_i - m_x)(y_i - m_y)$$

상관관계

- ❖ 분포의 크기는 공분산이 아닌 분산만으로도 알 수 있기 때문에 대부분의 경우 자료 분포의 방향성만 분리하여 보는 것이 유용한데 이 때 필요한 것이 상관계수 (correlation coefficient)
- ❖ 값의 범위는 -1에서 1사이입니다.
- ❖ 두 변수 사이에는 강한 양의 상관관계가 존재하면 상관 계수의 값은 1에 가까울 것입니다. 반대로 두 변수 사이에 특별한 상관관계가 존재하지 않으면 상관 계수의 값은 0에 가까울 것이며, 두 변수 사이에 음의 상관관계가 존재한다면 상관 계수의 값은 -1에 가까워집니다.
- ❖ DataFrame 이나 Series 객체의 `corr()`을 이용하면 상관계수를 알아볼 수 있습니다.
- ❖ `cov()`는 공분산

상관관계



상관관계

```
From pandas import Series, DataFrame  
Import pandas as pd  
Import numpy as np  
Import matplotlib.pyplot as plt  
From matplotlib import font_manager, rc  
From scipy import stats  
Import scipy as sp
```

```
Sales = Series([3,5,8,11,13])  
은 = Series([1,2,3,4,5])  
Su = sales.corr(dms)  
print("상관계수:",su)
```

회귀분석

- ❖ 두 변수 사이에 상관관계가 존재한다면 통계적으로 모델을 작성할 수 있으며, 특히 단순하게 일차원적인 선형 모델의 경우라면 다음과 같이 단순한 수식 형태로 표현할 수 있습니다

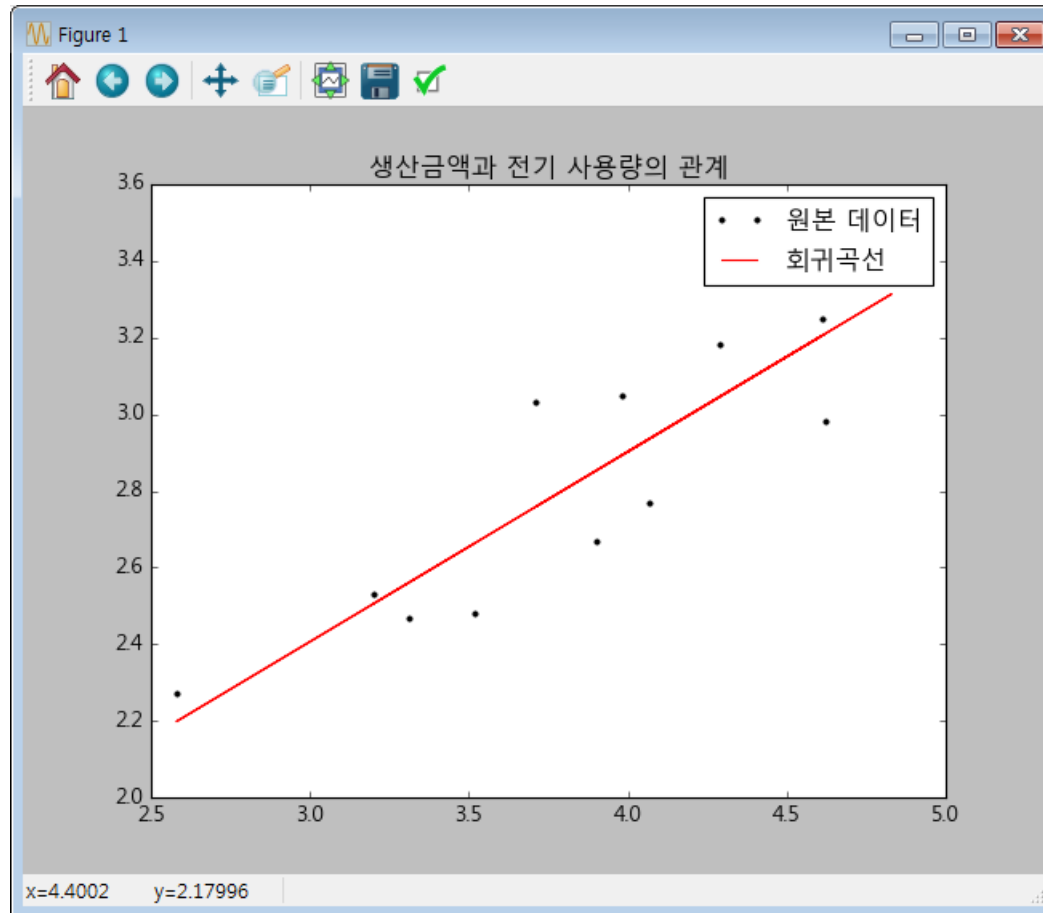
$$Y_i = B_0 + B_1 * X_i + E_i$$

- ❖ 여기서 Y_i 를 종속 변수(dependent variable), X_i 를 독립 변수(independent variable)라고 합니다. 종속 변수와 독립 변수 간의 관계식을 결정하는 B_0 와 B_1 는 데이터로부터 추정할 수 있으며, 이를 각각 절편(intercept), 기울기(slope)라고 합니다. E_i 는 오차항으로 우리가 작성한 모델과 실제 데이터 값과의 차이를 나타냅니다

회귀분석

- ❖ 종속 변수에 영향을 주는 변수가 1개일 경우를 단일 회귀분석이라고 합니다.
- ❖ 단일 회귀 분석의 경우는 scipy 패키지의 stats 모듈의 linregress 함수를 이용합니다.
- ❖ 결과는 float 형으로 선형 모델의 기울기, 절편, 상관 계수, p-value, 에러의 표준 편차가 순차적으로 반환됩니다.
- ❖ 여기서 p-value는 통계학에서 예측 불확실성의 정도를 나타내는 값으로, 일반적으로 0.05 미만일 때가 통계학적으로 유의미

회귀분석



회귀분석

```
import pandas as pd
import numpy as np
from pandas import Series, DataFrame
from scipy import stats
import matplotlib as plt
from matplotlib import font_manager, rc
from scipy import stats, polyval
from pylab import plot, title, show, legend

font_name =
font_manager.FontProperties(fname="c:/Windows/Fonts/malgun.ttf").get_name()
rc('font', family=font_name)

data = pd.read_csv("electronic.csv", encoding="ms949")
```

회귀분석

```
slope, intercept, r_value, p_value, stderr = stats.linregress(data['production'],  
data['quantity'])  
print("기울기:", slope)  
print("절편:", intercept)  
print("상관계수", r_value)  
print("불확실성 정도:", p_value)  
print("생산금액이 4가 되기 위한 전기 사용량:", end=' ')  
print(4 * slope + intercept)
```

회귀분석

```
ry = polyval([slope, intercept], data['production'])  
plot(data['production'], data['quantity'],'k.')  
plot(data['production'], ry,'r')  
title('생산금액과 전기 사용량의 관계')  
legend(['원본 데이터', '회귀곡선'])  
show()
```

삼성전자와 코스닥 지수

❖ 삼성전자 최근 주가 30개 가져오기

0	2135000.0	15	2104000.0
1	2062000.0	16	2072000.0
2	2038000.0	17	2060000.0
3	2014000.0	18	2099000.0
4	2045000.0	19	2089000.0
5	2075000.0	20	2074000.0
6	2078000.0	21	2060000.0
7	2101000.0	22	2075000.0
8	2121000.0	23	2090000.0
9	2095000.0	24	2123000.0
10	2080000.0	25	2128000.0
11	2097000.0	26	2095000.0
12	2080000.0	27	2120000.0
13	2092000.0	28	2092000.0
14	2107000.0	29	2070000.0

삼성전자와 코스닥 지수

```
import urllib
import time
from urllib.request import urlopen
from bs4 import BeautifulSoup
import pandas as pd
import numpy as np
from pandas import DataFrame, Series
```



삼성전자와 코스닥 지수

#,를 없애주는 함수

```
def f(st):
```

```
    ar = st.split(',')
```

```
    k=""
```

```
    for i in ar:
```

```
        k = k + i
```

```
    return k
```

#문자열을 실수로 변경해주는 함수

```
def func(st):
```

```
    return float(st)
```


삼성전자와 코스닥 지수

```
stockItem = '005930'
```

```
datelist = []
```

```
samsunglist = []
```

```
for i in range(1, 4):
```

```
    url = 'http://finance.naver.com/item/sise_day.nhn?code=' + stockItem +  
'&page=' + str(i)
```

```
    html = urlopen(url)
```

```
    source = BeautifulSoup(html.read(), "html.parser")
```

```
    srlists = source.find_all("tr")
```

```
    isCheckNone = None
```

삼성전자와 코스닥 지수

```
time.sleep(1.50)
```

```
for i in range(1, len(srlists) - 1):
```

```
    if (srlists[i].span != isCheckNone):
```

```
        datelist.append(srlists[i].td.text)
```

```
        samsunglist.append(srlists[i].find_all("td", class_="num")[0].text)
```

```
df = Series(samsunglist)
```

```
df = df.map(f)
```

```
df = df.map(func)
```

```
print(df)
```

삼성전자와 코스닥 지수


❖코스피 지수 가져오기

[2173.7399999999998, 2165.04, 2149.1500000000001, 2138.4000000000001, 2148.46, 2145.7600000000002, 2134.8800000000001, 2148.6100000000001, 2128.9099999999999, 2123.8499999999999, 2133.3200000000002, 2151.73, 2152.75, 2160.8499999999999, 2161.0999999999999, 2167.5100000000002, 2160.23, 2164.6399999999999, 2166.98, 2163.3099999999999, 2155.6599999999999, 2168.9499999999998, 2172.7199999999998, 2168.3000000000002, 2178.3800000000001, 2157.0100000000002, 2164.5799999999999, 2150.0799999999999, 2133.0]

삼성전자와 코스닥 지수

```
from pandas_datareader import data, wb
from datetime import datetime
d = datetime.today()
imsi =
data.DataReader("KRX:KOSPI", "google", datetime(2017, 1, 1), datetime(d.year, d.month,
d.day))
imsi = imsi.sort_index(ascending=False)

s = imsi['Close']
li = []
for i in range(0, 29):
    li.append(s[i])
print(li)
```



삼성전자와 코스닥 지수

❖상관관계분석

기울기: $7.99531776094e-05$

절편: 1988.13275028

상관계수 0.143899577799

불확실성 정도: 0.456437811423

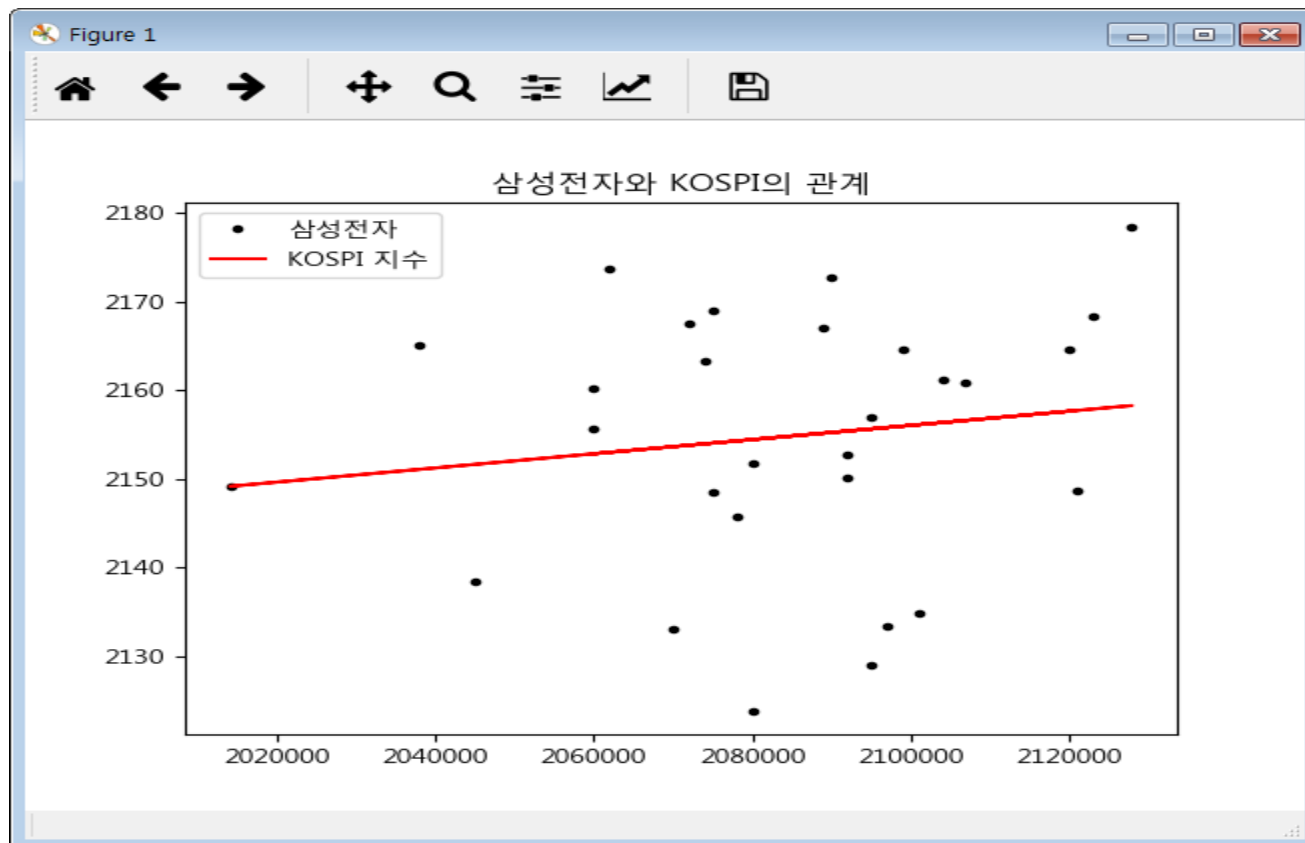


삼성전자와 코스닥 지수

```
from scipy import stats
slope, intercept, r_value, p_value, stderr = stats.linregress(df[1:30], li)
print("기울기:", slope)
print("절편:", intercept)
print("상관계수", r_value)
print("불확실성 정도:", p_value)
print(2060000 * slope + intercept)
```

삼성전자와 코스닥 지수

❖차트 만들기



삼성전자와 코스닥 지수

```
from matplotlib import font_manager, rc
from scipy import polyval
from pylab import plot, title, show, legend
```

```
font_name =
font_manager.FontProperties(fname="c:/Windows/Fonts/malgun.ttf").get_name()
rc('font', family=font_name)
```

```
ry = polyval([slope, intercept], df[1:30])
plot(df[1:30], li,'k.')
plot(df[1:30], ry,'r')
title('삼성전자와 KOSPI의 관계')
legend(['삼성전자', 'KOSPI 지수'])
show()
```


다중회귀분석

- ❖ statsmodels 는 통계 분석을 위한 python 패키지
- ❖ <http://www.statsmodels.org>
- ❖ statsmodels는 다음과 같이 기초 통계, 회귀 분석, 시계열 분석 등 다양한 통계 분석 기능을 제공
- ❖ 기초 통계 (Statistics)
 - ✓ 각종 검정(test) 기능
 - ✓ 커널 밀도 추정
 - ✓ Generalized Method of Moments
- ❖ 회귀 분석 (Linear Regression)
 - ✓ 선형 모형 (Linear Model)
 - ✓ 일반화 선형 모형 (Generalized Linear Model)
 - ✓ 강인 선형 모형 (Robust Linear Model)
 - ✓ 선형 혼합 효과 모형 (Linear Mixed Effects Model)
 - ✓ ANOVA (Analysis of Variance)
 - ✓ Discrete Dependent Variable (Logistic Regression 포함)
 - ✓ 시계열 분석 (Time Series Analysis)

다중회귀분석

❖ 단일 회귀 분석

- ✓ `statsmodels.regression.linear_model.OLS(endog, exog=None)`
- ✓ 파라미터
 - `endog` : 종속 변수. 1차원 배열
 - `exog` : 독립 변수, 2차원 배열.
- ✓ `statsmodels` 의 OLS 클래스는 자동으로 상수 항을 만들어주지 않기 때문에 사용자가 `add_constant` 명령으로 상수 항을 추가해야 한다.
- ✓ 모형 객체가 생성되면 `fit`, `predict` 메서드를 사용하여 추정 및 예측을 실시합니다.
- ✓ 예측 결과는 `RegressionResults` 클래스 객체로 출력되며 `summary` 메서드로 결과 보고서를 볼 수 있습니다.

다중회귀분석

❖ 다중 회귀 분석

- ✓ statsmodels.formula.api 패키지의 `ols(formula = '종속변수 ~ 독립변수[+ 독립변수]', data = 데이터프레임).fit()`을 호출해서 결과를 리턴 받습니다.
- ✓ 결과의 `params` 가 y절편과 각 독립변수 와 의 상관계수를 Series로 리턴
- ✓ 결과의 `pvalues` 가 유의 확률을 Series로 리턴
- ✓ 결과의 `predict()`이 예측 값을 ndarray 타입으로 리턴
- ✓ 결과의 `rsquared`가 반응 변수 변동의 백분율을 리턴하는데 일반적으로 값이 클수록 모형이 데이터를 더 잘 적합시킵니다.
 - 항상 0%에서 100% 사이입니다.
 - R-제곱은 다중 회귀 분석에서 결정 계수 또는 다중 결정 계수라고도 합니다.

다중회귀분석

name	score	iq	academy	game	tv
A	90	140	2	1	0
B	75	125	1	3	3
C	77	120	1	0	4
D	83	135	2	3	2
E	65	105	0	4	4
F	80	123	3	1	1
G	83	132	3	4	1
H	70	115	1	1	3
I	87	128	4	0	0
J	79	131	2	2	3

다중회귀분석

```
import pandas as pd
import numpy as np
from pandas import Series, DataFrame
from scipy import stats
import statsmodels.formula.api as sm

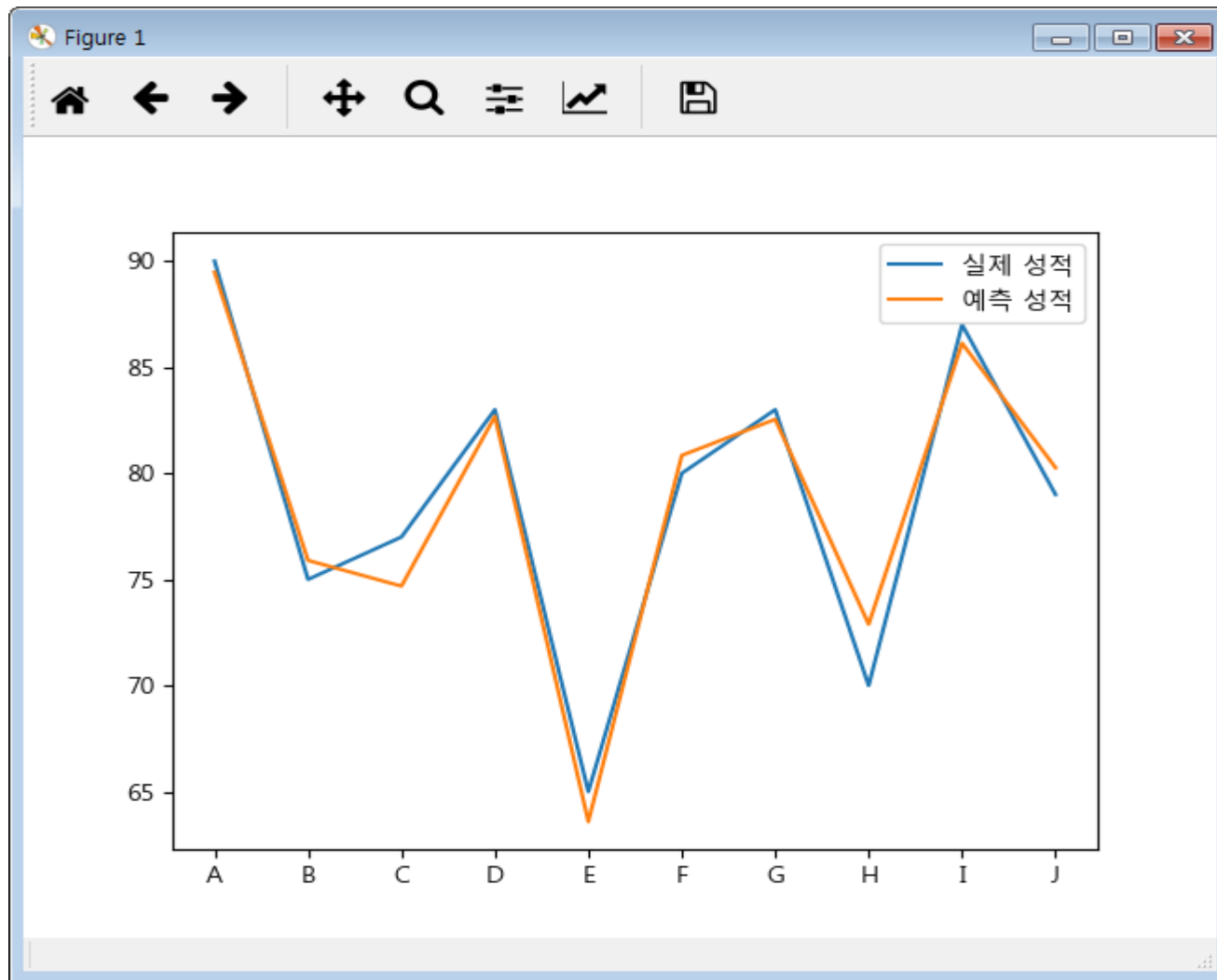
df = pd.read_csv("score.csv", encoding="ms949")
result = sm.ols(formula = 'score ~ iq + academy + game + tv', data = df).fit()
print('상관계수 : ', result.params)
print('Pvalue :', result.pvalues)
print('Predicted values', result.predict())
print('Rsquared : ', result.rsquared)
```

다중회귀분석

#IQ가 130이고 학원을 3개 다니고 게임을 2시간하고 TV를 1시간 보는 학생의 예측 점수는?

```
y = result.params.Intercept + 130*result.params.iq + 3*result.params.academy +  
2*result.params.game+1*result.params.tv  
print("예측점수:" , y)
```

다중회귀분석



다중회귀분석

```
import pandas as pd
import numpy as np
from pandas import Series, DataFrame
from scipy import stats
import statsmodels.formula.api as sm
import matplotlib.pyplot as plt
from matplotlib import font_manager, rc
font_name =
font_manager.FontProperties(fname="c:/Windows/Fonts/malgun.ttf").get_name()
rc('font', family=font_name)
```


다중회귀분석

```
df = pd.read_csv("score.csv", encoding="ms949")
result = sm.ols(formula = 'score ~ iq + academy + game + tv', data = df).fit()
plt.figure()
plt.plot(df['score'], label='실제 성적')
plt.plot(result.predict(), label='예측 성적')
plt.xticks(range(0,10,1),df['name'])
plt.legend();
plt.show()
```

다중회귀분석

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
4	3	최정	3루수	SK	120	434	75	137	18	0	28	83	24	64	109	10	0.316
5	4	박석민	3루수	삼성	117	396	61	126	24	0	18	76	4	54	73	13	0.318
6	5	손아섭	우익수	롯데	128	498	83	172	23	4	11	69	36	64	88	9	0.345
7	6	정성훈	3루수	LG	121	407	64	127	22	1	9	62	13	59	58	12	0.312
8	7	배영섭	중견수	삼성	113	393	66	116	16	4	2	38	23	52	62	9	0.295
9	8	나지완	좌익수	KIA	125	435	57	125	18	0	21	96	7	62	111	12	0.287
10	9	박정권	1루수	SK	110	363	56	106	20	1	18	70	4	64	79	5	0.292
11	10	박용택	중견수	LG	125	476	79	156	22	4	7	67	13	52	71	5	0.328
12	11	이진영	우익수	LG	106	368	41	121	26	1	3	62	6	37	42	6	0.329
13	12	최진행	좌익수	한화	106	367	39	110	27	0	8	53	0	50	86	18	0.3
14	13	강정호	유격수	넥센	126	450	67	131	21	1	22	96	15	68	109	18	0.291
15	14	민병헌	우익수	두산	119	383	71	122	21	7	9	65	27	40	62	7	0.319
16	15	이병규	우익수	LG	98	374	39	130	19	3	5	74	2	22	40	10	0.348
17	16	김현수	좌익수	두산	122	434	63	131	23	1	16	90	2	62	71	6	0.302
18	17	홍성흔	1루수	두산	127	469	61	140	21	0	15	72	5	57	93	15	0.299
19	18	김종호	우익수	NC	128	465	72	129	12	7	0	22	50	57	100	11	0.277
20	19	이용규	중견수	KIA	100	390	74	115	20	1	2	22	21	44	37	4	0.295
21	20	김강민	중견수	SK	105	352	39	106	22	3	10	55	10	36	54	10	0.301
22	21	이종욱	중견수	두산	110	401	77	123	23	6	6	52	30	38	57	6	0.307
23	22	정근우	2루수	SK	112	407	64	114	19	3	9	35	28	50	48	6	0.28
24	23	전준우	중견수	롯데	128	455	65	125	25	1	7	66	19	65	77	15	0.275
25	24	최형우	좌익수	삼성	128	511	80	156	28	0	29	98	2	47	91	5	0.305
26	25	가미호	포수	롯데	105	377	48	77	13	0	11	57	1	60	87	5	0.235

baseball

준비

100 %

다중회귀분석

```
import pandas as pd
import numpy as np
from pandas import Series, DataFrame
from scipy import stats
import statsmodels.formula.api as sm

df = pd.read_csv("baseball.csv", encoding="ms949")
df = df[["타수", "득점", "2루타", "3루타", "홈런", "삼진", "타율"]]
df['run'] = df['득점']/df['타수']
df['double'] = df['2루타']/df['타수']
df['triple'] = df['3루타']/df['타수']
df['homerun'] = df['홈런']/df['타수']
df['strikeout'] = df['삼진']/df['타수']
df['avg'] = df['타율']
print(df)
```

다중회귀분석

#타율과 2루타, 3루타, 홈런, 삼진 등이 득점 확률과의 회귀분석

```
result = sm.ols(formula = 'run ~ avg+strikeout+homerun+double+triple', data =  
df).fit()
```

```
print('상관계수 : ', result.params)
```

```
print('Pvalue :', result.pvalues)
```

```
print('Predicted values', result.predict())
```

```
print('Rsquared : ', result.rsquared)
```