데이터 만들기

1

확률변수 만들기, 외부 txt, csv. excel 데이터 읽어온 후 pandas 데이터 만들기(Series, DataFrame)

난수생성&PDF

scipy 모듈 scipy.stats.분포함수명

```
import scipy.stats as st
```

scipy.stats.분포함수명.rvs(모수, size=)

- 설정한 확률분포함수 따르는 데이터(난수) 생성 $generate\ x_i \sim f(x)$

scipy.stats.분포함수명.pdf (cdf, ppf) (x, arg, 모수)

- 확률분포함수 $f(x)$, 누적분포함수 $F(x)$, 역누적분포함수(백분위 값) $F^{-1}(x)$
- logpad, logcdf 사용 가능

scipy.stats.분포함수명.moment(n=, x, 모수)

- n차 적률 구하기

scipy.stats.분포함수명.stats(arg, ,모수, moments='arg2')

- arg2 : m, v, s, k
- stats 대신 mean, median, var, std 사용 가능
- interval(alpha, arg, 모수) : 백분위(alpha %) 누적 백분위 값

scipy.stats.분포함수명.fit(data)

- 데이터가 설정된 확률분포를 따른다는 가정에서 모수를 추정함

주요 분포함수 모수

이산 균일분포(discrete uniform) randint(low= , high=)

(low, high-1) 사이의 정수의 균일분포

이항분포(binomial) binom(n= , p=) - n번 베르누이(성공확률 p) 시행 중 성공회수

$f(k) = \binom{n}{k} p^k (1-p)^{n-k}$, $k = 0,1,2,\ldots,n$ 평균=np, 분산= npq

포아송분포 poisson(mu=) - 평균이 μ인 관심 사건의 발생 건수

$f(k) = \exp(-\mu)\frac{\mu^k}{k!}$, $k = 0,1,2,\ldots,$ 평균= λ, 분산=λ

음이항분포(negative binomial) nbinom(n= , p=) - 성공확률 p인 베르누이 시행에서 n번의 성공까지 실패회수

$f(k) = \binom{k+n-1}{n-1} p^n (1-p)^k$, $k = 0,1,2,\ldots,$ 평균= $\frac{n}{p}$, 분산= $\frac{n(1-p)}{p^2}$

초기하분포(hyper-geometric) hypergeom(M= , n= , N=) 개체가 2진형(성공, 실패)으로 구성된 모집단(총 개체수 M, 성공개체수 n)으로부터 N개 표본을 추출할 때 성공개체 수

$$f(k) = \frac{\binom{n}{k}}{\binom{M}{N}}\binom{M-n}{N-k},\ k = 0,1,2,\ldots,min(N,n),\ 평균= \frac{Nn}{M}$$

균일분포 uniform(loc= , scale=)

$$f(x) = \frac{1}{scale}, loc < x < loc + scale$$

정규분포 norm(loc= , scale=)

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{(x-\mu)^2}{2\sigma^2}},\ -\infty < x < \infty\ 평균= \mu,\ 분산\ \sigma^2$$

감마분포 gamma(a(k)= , scale(θ)=)

$$f(x) = \frac{1}{\Gamma(k)\theta^k}x^{k-1}e^{-\frac{x}{\theta}},\ 0 < x,\ 평균= k\theta,\ 분산\ k\theta^2$$

와이블분포

exponweib.pdf(x, a, c)=a * c * (1-exp(-x**c))**(a-1) * exp(-x**c)*x**(c-1)

$$f(x;k,\lambda;\alpha) = \alpha\frac{k}{\lambda}\left[\frac{x}{\lambda}\right]^{k-1}\left[1-e^{-(x/\lambda)^k}\right]^{\alpha-1}e^{-(x/\lambda)^k}$$

in Python 사례

감마분포($\alpha = 2, \beta = 0.5$) 500개 데이터 생성

```
from scipy.stats import gamma
a = 2; s =0.5
y=gamma.rvs(a=a,scale=s, size=500)  #Gamma(2,0.5) size=500
```

$data\ y_i \sim\ Gamma(\alpha = 2,\ \beta = 0.5),\ i = 1,2,...,500$

난수 생성된 데이터 기초통계량

```
st.describe(y)  #elelmentary statistics
```

```
DescribeResult(nobs=1000, minmax=(0.013393060445211555, 4.4837821476271404), mean=1.02
8986889010037, variance=0.49501838881339916, skewness=1.1960879646017015, kurtosis=1.74
43735928218542)
```

기초통계량 저장 및 출력

```
n, rgn, m, var, sk, kur = st.describe(y)
print('min= %10.4f, mean= %10.4f,  varince=%10.4f' % (rgn[0], mean, var))
```

```
min=     0.0171, mean=     1.0000,  varince=     0.5140
```

감마분포($\alpha = 2, \beta = 0.5$) 모집단 통계량 구하기

```
n, rgn, m, var, sk, kur = st.describe(y)
print('min= %10.4f, mean= %10.4f,  varince=%10.4f' % (rgn[0], mean, var))
```

```
min=     0.0171, mean=     1.0000,  varince=    0.5140
```

```
mean, var, skew, kurt = st.gamma.stats(a=a,scale=s, moments='mvsk')
print(mean, var)
```

```
1.0 0.5
```

```
st.gamma.moment(n=2, a=a,scale=s)  #2nd Moment
```

```
1.5
```

```
st.gamma.moment(n=1, a=a,scale=s)  #1nd Moment
```

```
1.0
```

모집단 감마분포 기초통계량 구하기.

$Gamma(\alpha = 2, \beta = 0.5)$ 분포의 중위값=0.7, 표준편차는 $\sqrt{0.514}$이다.

```
st.gamma.median(a=a,scale=s)
```

```
0.8391734950083306
```

```
st.gamma.std(a=a,scale=s)
```

```
0.7071067811865476
```

(생성) 데이터 분포함수 모수 구하기

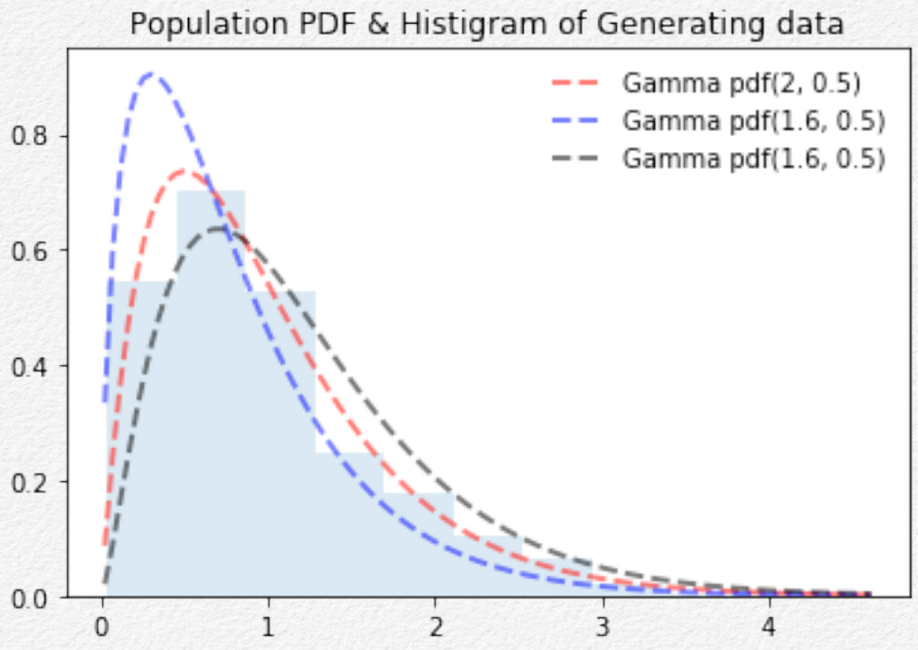
$Gamma(\alpha = 2, \beta = 0.5)$로부터 생성된 데이터를 감마분포함수에 적합시켜 모수를 추정하면 1.882(모집단은 2이다), 0.536(모집단은 0.5이었음)이다. 난수 생성이므로 모집단의 모수와는 동일하지 않다. 아마 이런 난수 생성을 무한히 하여 평균을 구하면 2, 0.5에 근사할 것이다.

```
a, loc, scale=st.gamma.fit(y)
print('alpha(shape)= %6.3f, scale= %6.3f, ' % (a, scale))
```

```
alpha(shape)=  1.882, scale=  0.536,
```

MLE 추정량 보기

```
import matplotlib.pyplot as plt
import numpy as np
fig, ax = plt.subplots(1, 1)

x = np.linspace(gamma.ppf(0.001,a=a,scale=s), gamma.ppf(0.999,a=a,scale=s), 100)
ax.set_title('Population PDF & Histigram of Generating data')
ax.plot(x, gamma.pdf(x,a=a,scale=s), 'r--', lw=2, alpha=0.6, label='Gamma pdf('+str(a)+', '+str(s)+')')
ax.plot(x, gamma.pdf(x,a=0.8*a,scale=s), 'b--', lw=2, alpha=0.6, label='Gamma pdf('+str(0.8*a)+', '+str(s)+')')
ax.plot(x, gamma.pdf(x,a=1.2*a,scale=s), 'k--', lw=2, alpha=0.6, label='Gamma pdf('+str(0.8*a)+', '+str(s)+')')

ax.hist(y, density=True, histtype='stepfilled', alpha=0.2)
ax.legend(loc='best', frameon=False)
plt.show()
```



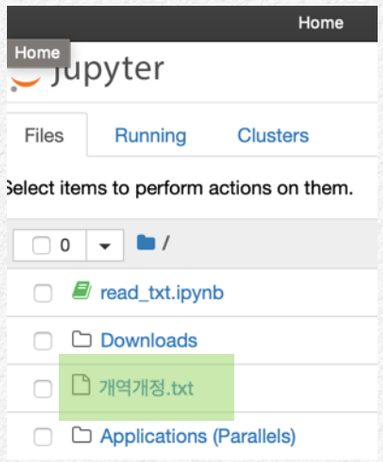
Python Code [copy & paste]

```
import scipy.stats as st
import numpy as np
a = 2; s =0.5
y=st.gamma.rvs(a=a,scale=s, size=500)  #Gamma(2,0.5) size=500

st.describe(y)   #elelmentary statistics

n, rgn, m, var, sk, kur = st.describe(y)
print('min= %10.4f, mean= %10.4f,  varince=%10.4f' % (rgn[0], mean,
var))

mean, var, skew, kurt = st.gamma.stats(a=a,scale=s, moments='mvsk')
print(mean, var)

st.gamma.moment(n=2, a=a,scale=s)  #2nd Moment

st.gamma.moment(n=1, a=a,scale=s)  #1nd Moment

st.gamma.median(a=a,scale=s)

st.gamma.std(a=a,scale=s)

a, loc, scale=st.gamma.fit(y)
print('alpha(shape)= %6.3f, scale= %6.3f, ' % (a, scale))



import matplotlib.pyplot as plt
import numpy as np
fig, ax = plt.subplots(1, 1)

x = np.linspace(gamma.ppf(0.001,a=a,scale=s),
gamma.ppf(0.999,a=a,scale=s), 100)
ax.set_title('Population PDF & Histigram of Generating data')
ax.plot(x, gamma.pdf(x,a=a,scale=s), 'r--', lw=2, alpha=0.6,
label='Gamma pdf('+str(a)+', '+str(s)+')')
ax.plot(x, gamma.pdf(x,a=0.8*a,scale=s), 'b--', lw=2, alpha=0.6,
label='Gamma pdf('+str(0.8*a)+', '+str(s)+')')
ax.plot(x, gamma.pdf(x,a=1.2*a,scale=s), 'k--', lw=2, alpha=0.6,
label='Gamma pdf('+str(0.8*a)+', '+str(s)+')')

ax.hist(y, density=True, histtype='stepfilled', alpha=0.2)
ax.legend(loc='best', frameon=False)
plt.show()

plt.subplots(1, 1)
```

데이터_만들기

텍스트 (내부) 데이터 불러오기

위치 : Home Folder에 있음



open() 함수

- 'rt' - 텍스트모드로 읽음
- 반드시 읽은 후 데이터를 close할 것을 권한다.
- 읽은 데이터는 'str' 형식으로 저장된다.

```
txt=open('개역개정.txt','rt')
dtxt=txt.read()   #read file into str type
txt.close()   #close '개역개정.txt'
print(type(dtxt), len(dtxt))
```

```
<class 'str'> 1836672
```

- 인덱스 0에서 39까지 저장된 내용이다.

```
dtxt[0:40]   #40 strings form p=0 to p0sition 39
```

```
'\ufeff태초에 하나님이 천지를 창조하시니라\n땅이 혼돈하고 공허하며 흑암이 깊음'
```

단어 단위 나누어 저장

- split() 함수에 의해 공백으로 분리되어 리스트 형식으로 저장된다.
- 총 473,696개 단어가 있다. 그 중 20개만 일부 저장하였다.

```
word=dtxt.split()   #split str by blank into list type
type(word)
```

```
list
```

```
len(word)   # size of list
```

```
473696
```

```
word_sub=word[0:20]   # subset of list word from 0 to 19
word_sub
```

```
['\ufeff태초에',
 '하나님이',
 '천지를',
 '창조하시니라',
 '땅이',
 '혼돈하고',
 '공허하며',
 '흑암이',
```

지정 단어 빈도 구하기 count()

- 리스트 전체에 대하여 count()함수 사용하면 ' '안에 설정된 단어와 일치하는 단어 개수를 출력한다. --> word_sub 리스트 안에 있는 20개 값(단어) 중 '하나님'과 일치하는 단어(값)는 없으므로 0, '하나님이' 단어는 2개이다.
- 만약 리스트 안의 개체를 개별적으로 나누면, 그 값 안에서 ' '안에 설정된 단어가 있으면 카운트 된다. --> **k in word_sub** = 리스트 안의 개별 값(단어)가 차례로 k에 대체되고 그 k가 '하나님' 단어가 있으면 카운트 된다.

```
len(word_sub)   # size of subset list
```

```
20
```

```
word_sub.count('하나님')   #exact word
```

```
0
```

```
word_sub.count('하나님이')   #exact word
```

```
2
```

```
num=0
for k in word_sub:   #by individual
    num=num + k.count('하나님')
num
```

```
3
```

```
word_sub
```

```
['\ufeff태초에',
 '하나님이',
 '천지를',
 '창조하시니라',
 '땅이',
 '혼돈하고',
 '공허하며',
 '흑암이',
 '깊음',
 '위에',
 '있고',
 '하나님의',
 '영은',
 '수면',
 '위에',
 '운행하시니라',
 '하나님이',
 '이르시되',
 '빛이',
 '있으라']
```

```
num=0
for k in word_sub:   #by individual
    num=num + k.count('이')
num
```

```
6
```

=> 단어 중 '이'가 있으면 카운트 된다.

문장(\n) 단위로 나누어 저장하기

```
tline=dtxt.splitlines()   #split by lines
type(tline)
```

```
list
```

```
tline[0:3]   #0~2nd lines
```

```
['\ufeff태초에 하나님이 천지를 창조하시니라',
 '땅이 혼돈하고 공허하며 흑암이 깊음 위에 있고 하나님의 영은 수면 위에 운행하시니라',
 '하나님이 이르시되 빛이 있으라 하시니 빛이 있었고']
```

```
tline.count('하나님')
```

```
0
```

```
num=0
for k in tline[0:3]:
    num=num + k.count('하나님')
num
```

```
3
```

텍스트 (url) 데이터 불러오기

- 한글 url 사이트 데이터 불러오는 경우 크롬에서 주소 복사하기를 권한다.(강추)

```
import urllib.request
url='http://203.247.53.31/Big_Data/data/%EA%B0%9C%EC%97%AD%EA'
request = urllib.request.Request(url)
response = urllib.request.urlopen(request)
dtxt=response.read().decode('utf-8')
```

```
type(dtxt)
```

```
str
```

```
word=dtxt.split()
word.count('하나님')
```

```
998
```

```
num=0
for k in word:
    num=num+k.count('하나님')
print("하나님 사용 비율 = {:.2%}".format(num/len(word)))
```

```
하나님 사용 비율 = 0.88%
```

csv 데이터 불러오기

http://wolfpack.hnu.ac.kr/Stat_Notes/adv_stat/LinearModel/data/SMSA.csv [예제 데이터]

- 한글 깨짐 방지를 위한 옵션 : encoding='ms949' (영어로만 데이터의 경우 이 옵션을 사용하지 않아도 된다.)

```
import pandas as pd
filename='http://wolfpack.hnu.ac.kr/Stat_Notes/adv_stat/LinearModel/data/SMSA.csv'
ds=pd.read_csv(filename, encoding='ms949')
```

```
ds.head(3)
```

	city	Mortality	JanTemp	JulyTemp	RelHum	Rain	Education
0	Akron, OH	921.87	27	71	59	36	11.4
1	Albany-Schenectady-	997.87	23	72	57	35	11.0

```
ds.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 60 entries, 0 to 59
Data columns (total 15 columns):
city          60 non-null object
Mortality     60 non-null float64
```

```
ds.columns
```

```
Index(['city', 'Mortality', 'JanTemp', 'JulyTemp', 'RelHum', 'Rain',
       'Education', 'PopDensity', 'NonWhite', 'WC', 'pop/house', 'income',
       'HCPot', 'NOxPot', 'SO2Pot'],
      dtype='object')
```

excel 데이터 불러오기

http://203.247.53.31/Stat_Notes/example_data/LPGA_2008.xls

```
ImportError: Install xlrd >= 1.0.0 for Excel support
```

이런 오류가 나오는 경우 xlrd 모듈을 설치가 필요. `pip3 install xlrd`

```
ds.columns
```

```
Index(['city', 'Mortality', 'JanTemp', 'JulyTemp', 'RelHum', 'Rain',
       'Education', 'PopDensity', 'NonWhite', 'WC', 'pop/house', 'income',
       'HCPot', 'NOxPot', 'SO2Pot'],
      dtype='object')
```

```
import pandas as pd
filename='http://203.247.53.31/Stat_Notes/example_data/LPGA_2008.xls'
ds_excel=pd.read_excel(filename, encoding='ms949')
```

```
ds_excel.columns
```

```
Index(['Player', 'Driver_Dist', 'Fairway_pcnt', 'Green_Regulation', 'Avg_Putt',
       'Sand_num', 'Sand_saves', 'Money_round', 'Total_round', 'Total_money'],
      dtype='object')
```

```
ds_excel[0:3]
```

	Player	Driver_Dist	Fairway_pcnt	Green_Regulation	Avg_Putt	Sand_num	Sand_saves	Money_round	Total_round	Tota
0	Ochoa, Lorena	269.3	66.4	71.6	28.31	0.94	50.0	40635	68	
1	Creamer, Paula	246.3	73.6	70.5	28.38	0.58	47.1	20727	88	
	Tseng									

SAS data 불러오기

http://203.247.53.31/Stat_Notes/example_data/cars.sas7bdat

SAS는 url 형식인 아닌 다운 후 Home 폴더에 저장하여 읽어오는 코딩

```
import pandas as pd
from sas7bdat import *
foo = SAS7BDAT('cars.sas7bdat', encoding='ms949')
ds_sas = foo.to_data_frame()
```

```
ds_sas.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 428 entries, 0 to 427
Data columns (total 15 columns):
Make          428 non-null object
Model         428 non-null object
```

```
ds_sas.head(3)
```

	Make	Model	Type	Origin	DriveTrain	MSRP	Invoice	EngineSize	Cylinders	Horsepower	MPG_City
0	Acura	MDX	SUV	Asia	All	36945.0	33337.0	3.5	6.0	265.0	17.0
1	Acura	RSX Type S 2dr	Sedan	Asia	Front	23820.0	21761.0	2.0	4.0	200.0	24.0
2	Acura	TSX 4dr	Sedan	Asia	Front	26990.0	24647.0	2.4	4.0	200.0	22.0

데이터 저장하기

csv 형식 저장

```
import pandas as pd
ds_sas.to_csv('골프데이터.csv', encoding='ms949')
```

	A	B	C	D	E	F	G	H
		Make	Model	Type	Origin	DriveTrain	MSRP	Invoice
0		Acura	MDX	SUV	Asia	All	36945	33337
1		Acura	RSX Type S 2	Sedan	Asia	Front	23820	21761
2		Acura	TSX 4dr	Sedan	Asia	Front	26990	24647
3		Acura	TL 4dr	Sedan	Asia	Front	33195	30299

- 첫열에 pandas 데이터프레임의 index 없이 저장하기

```
import pandas as pd
ds_sas.to_csv('골프데이터.csv', index = False, encoding='ms949')
```

A	B	C	D	E	F	G	H
Make	Model	Type	Origin	DriveTrain	MSRP	Invoice	EngineSize
Acura	MDX	SUV	Asia	All	36945	33337	3.5
Acura	RSX Type S 2	Sedan	Asia	Front	23820	21761	2
Acura	TSX 4dr	Sedan	Asia	Front	26990	24647	2.4

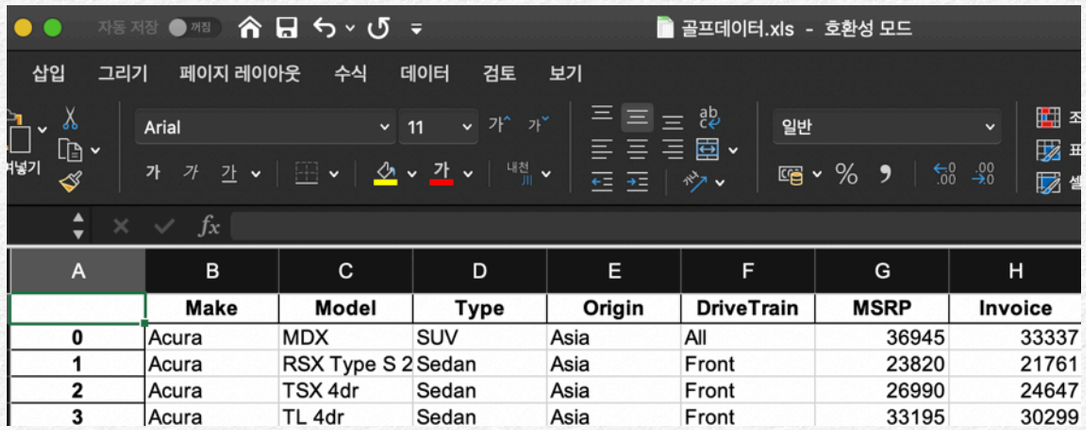
excel 파일 저장

```
ModuleNotFoundError: No module named 'xlwt'
```

오류 발생이 나면

`pip3 install xlwt` 모듈 설치 필요

```
import pandas as pd
ds_sas.to_excel('골프데이터.xls', encoding='ms949')
```



A	B	C	D	E	F	G	H
	Make	Model	Type	Origin	DriveTrain	MSRP	Invoice
0	Acura	MDX	SUV	Asia	All	36945	33337
1	Acura	RSX Type S 2	Sedan	Asia	Front	23820	21761
2	Acura	TSX 4dr	Sedan	Asia	Front	26990	24647
3	Acura	TL 4dr	Sedan	Asia	Front	33195	30299