Gholap Sourabh Somnath
EE20BTECH11047

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import emcee
import corner
import dynesty
import scipy.stats as stats
```

Question 1

Download the SPT fgas data from http://iith.ac.in/~shantanud/fgas_spt.txt. Fit the data to f0(1 + f1z) where f0 and f1 are unknown constants. Determine the best fit values of f0 and f1 including 68% and 90% credible intervals using emcee and corner.py . The priors on f0 and f1 should be 0 < f0 < 0.5 and −0.5 < f1 < 0.5. (30 pts)

```
data = pd.read_csv('SPATfgas.txt',sep=' ')
data=pd.DataFrame(data)
data.head()
```

	#z	fgas	fgas_error	ignore
0	0.2777	0.096610	0.014883	0.0
1	0.2786	0.113973	0.015304	0.0
2	0.2836	0.141139	0.019905	0.0
3	0.2950	0.113488	0.017296	0.0
4	0.2960	0.063003	0.011192	0.0

```
z = data['#z']
fgas = data['fgas']
fgas_error = data['fgas_error']

def log_likelihood(theta,z,fgas,fgas_error):
    f0,f1 = theta
    model_fit = f0*(1+f1*z)
    sigma = fgas_error
    return -0.5*np.sum(np.log(2 * np.pi * sigma ** 2) + pow((fgas -
model_fit) ,2) / pow(sigma,2))

def log_prior(theta):
    f0,f1 = theta
    if 0<f0<0.5 and -0.5<f1<0.5:
        return 0
    return -np.inf

def log_posterior(theta,z,fgas,fgas_error):
    log_prior_value = log_prior(theta)
    if not np.isfinite(log_prior_value):
        return -np.inf
    return log_prior_value + log_likelihood(theta,z,fgas,fgas_error)


no_of_walkers = 50
```

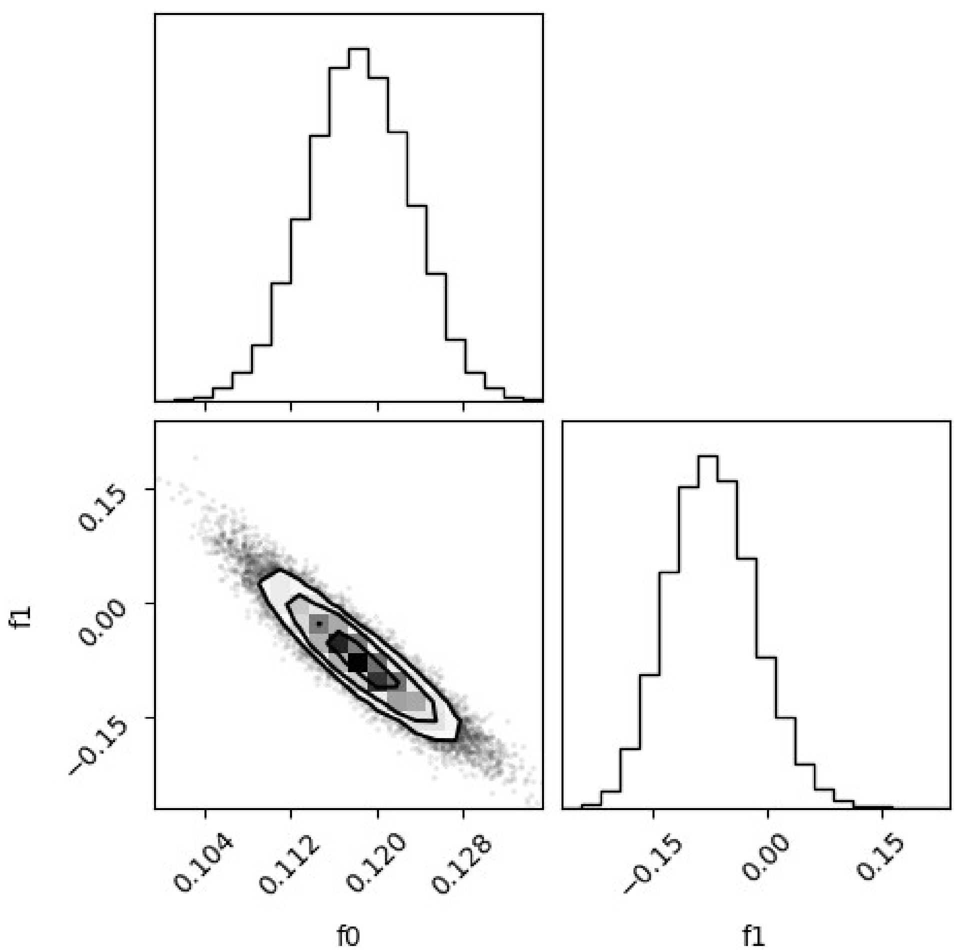
```
intial_guess =
np.array([np.random.uniform(0.0,0.5,no_of_walkers),np.random.uniform(0
.,0.5,no_of_walkers)]).T

sampler =
emcee.EnsembleSampler(no_of_walkers,2,log_posterior,args=(z,fgas,fgas_
error))
sampler.run_mcmc(intial_guess,5000,progress = True)
chain = sampler.chain[: ,2000: , : ].reshape((-1,2))
```

```
100%|██████████| 5000/5000 [02:58<00:00, 28.00it/s]
```

For 68% credible region :

```
flat_samples68 = sampler.get_chain(discard=200,thin=15,flat=True)
labels=['f0','f1']
fig = corner.corner(flat_samples68,
labels=labels,title_quantiles=[0.16,0.5,0.84])
```



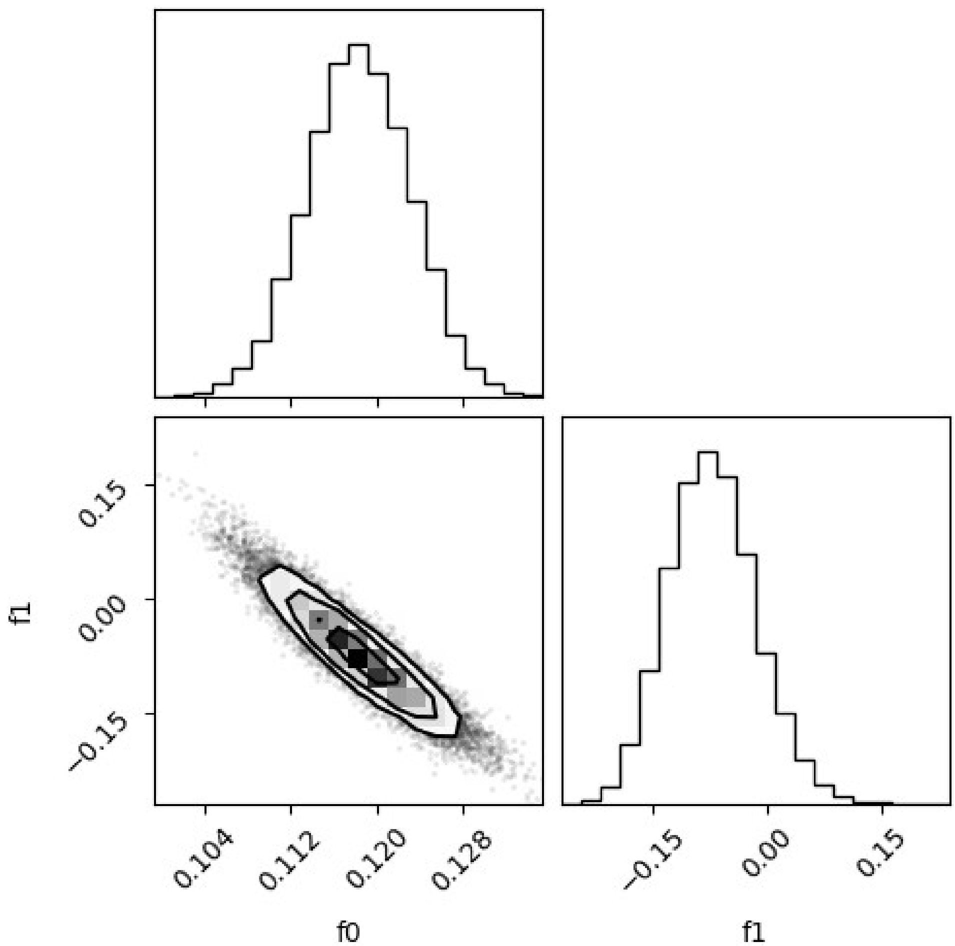

```
f0 = corner.quantile(flat_samples68, [0.16,0.84], weights=None)[0]
f1 = corner.quantile(flat_samples68, [0.16,0.84], weights=None)[1]
f0 = round(f0,3)
```

```
f1 = round(f1,3)
print('f0 =',f0)
print('f1 =',f1)
print(f"Data is fitted to {f0}*(1+{f1}*z)")
```

```
f0 = -0.1
f1 = 0.121
Data is fitted to -0.1*(1+0.121*z)
```

For 90% credible region

```
flat_samples90 = sampler.get_chain(discard=200,thin=15,flat=True)
labels=['f0','f1']
fig = corner.corner(flat_samples90,
labels=labels,title_quantiles=[0.05,0.5,0.95])
```




```
f0 = corner.quantile(flat_samples68, [0.05,0.95], weights=None)[0]
f1 = corner.quantile(flat_samples68, [0.05,0.95], weights=None)[1]
f0 = round(f0,3)
f1 = round(f1,3)
print('f0 =',f0)
```

```
print('f1 =',f1)
print(f"Data is fitted to {f0}*(1+{f1}*z)")
```

```
f0 = -0.143
f1 = 0.125
Data is fitted to -0.143*(1+0.125*z)
```

Question 2

Calculate the Bayes factor for the linear and quadratic model for the example given on fifth blog article of the Pythonic Perambulations Series using dynesty or Nestle. Do the values agree with what's on the blog (obtained by integrating the emcee samples).? (30 points)

```
data2 = np.array([[ 0.42,  0.72,  0.  ,  0.3 ,  0.15,
                   0.09,  0.19,  0.35,  0.4 ,  0.54,
                   0.42,  0.69,  0.2 ,  0.88,  0.03,
                   0.67,  0.42,  0.56,  0.14,  0.2  ],
                 [ 0.33,  0.41, -0.22,  0.01, -0.05,
                  -0.05, -0.12,  0.26,  0.29,  0.39,
                   0.31,  0.42, -0.01,  0.58, -0.2 ,
                   0.52,  0.15,  0.32, -0.13, -0.09 ],
                 [ 0.1 ,  0.1 ,  0.1 ,  0.1 ,  0.1 ,
                   0.1 ,  0.1 ,  0.1 ,  0.1 ,  0.1 ,
                   0.1 ,  0.1 ,  0.1 ,  0.1 ,  0.1 ,
                   0.1 ,  0.1 ,  0.1 ,  0.1 ,  0.1  ]])
x,y,sigma_y = data2

def log_likelihood_lin(theta):
    m, b, log_f = theta
    model = m * x + b
    sigma2 = sigma_y**2 + model**2 * np.exp(2 * log_f)
    return -0.5 * np.sum((y - model) ** 2 / sigma2 + np.log(sigma2))

def log_prior_lin(theta):
    m_prior, b_prior, lf_prior = theta
    m = 5.5 * m_prior - 5.
    b = 10. * b_prior
    lnf = 11. * lf_prior - 10.
    return m, b, lnf

def log_likelihood_quad(theta):
    a, b, c, log_f = theta
    model = a*x**2 + b*x + c
    sigma2 = sigma_y**2 + model**2 * np.exp(2 * log_f)
    return -0.5 * np.sum((y - model) ** 2 / sigma2 + np.log(sigma2))

def log_prior_quad(theta):
    a_prior, b_prior, c_prior, lf_prior = theta
    a = 5.5 * a_prior - 5.
```

```
    b = 10. * b_prior
    c = 10. * c_prior
    lnf = 11. * lf_prior -10
    return a, b, c, lnf
```

```
sampler_dynesty_lin =
dynesty.DynamicNestedSampler(log_likelihood_lin,log_prior_lin,ndim=3,b
ound='multi',sample='rwalk')
sampler_dynesty_lin.run_nested()
dres_linear = sampler_dynesty_lin.results
logz_linear = dres_linear.logz[-1]
z_linear = np.exp(logz_linear)
```

```
17154it [00:40, 419.08it/s, batch: 6 | bound: 25 | nc: 1 | ncall:
348637 | eff(%):  4.770 | loglstar: 20.520 < 26.433 < 25.407 | logz:
14.087 +/-  0.093 | stop:  0.874]
```

```
sampler_dynesty_quad =
dynesty.DynamicNestedSampler(log_likelihood_quad,log_prior_quad,ndim=4
,bound='multi',sample='rwalk')
sampler_dynesty_quad.run_nested()
dres_quad = sampler_dynesty_quad.results
logz_quad = dres_quad.logz[-1]
z_quad = np.exp(logz_quad)
```

```
18217it [00:41, 436.01it/s, batch: 6 | bound: 20 | nc: 1 | ncall:
389126 | eff(%):  4.547 | loglstar: 25.131 < 30.860 < 29.287 | logz:
15.675 +/-  0.105 | stop:  0.900]
```

```
Bayes_factor = z_quad/z_linear
print("Bayes factor is :",Bayes_factor)
```

```
Bayes factor is : 4.664715957014887
```

Bayes factor according to blog is 2.36

Question3

Download the SDSS quasar dataset from http://astrostatistics.psu.edu/datasets/SDSS_quasar.dat. Plot the KDE estimate of the quasar redshift distribution (the column with the title z) using a Gaussian and also an exponential kernel (with bandwidth=0.2) from -0.5 to 5.5. (20 points) (Hint: Look at the KDE help page in scikit-learn or use the corresponding functions in astroML module by looking at source code of astroML figures 6.3 and 6.4

```
from sklearn.neighbors import KernelDensity
```

```
data3 = np.genfromtxt('SDSS_quasar.dat.txt',dtype =str)
useful_data = data3[1:,3].astype(np.float64)
gau_dist = stats.norm(np.mean(useful_data),np.std(useful_data))
x = np.linspace(-0.5,5.5,100)
gau_pdf = gau_dist.pdf(x)
```

```
kernel_density_ex =
KernelDensity(kernel='exponential',bandwidth=0.2).fit(useful_data.resh
ape(-1,1))
exp_kernel_density_ex =
np.exp(kernel_density_ex.score_samples(x.reshape(-1,1)))

kernel_density_gau =
KernelDensity(kernel='gaussian',bandwidth=0.2).fit(useful_data.reshape
(-1,1))
exp_kernel_density_gau =
np.exp(kernel_density_gau.score_samples(x.reshape(-1,1)))

plt.figure(figsize=(12,6))
plt.plot(x, gau_pdf)
plt.plot(x, exp_kernel_density_ex)
plt.plot(x, exp_kernel_density_gau)
plt.title('KDE estimate of quassar redshift distribution')
plt.xlabel('x')
plt.ylabel('y')
plt.grid()
plt.legend(['Input','exponential kernal','Gaussian kernel'])
plt.show()
```