Assignment 7

Raghav Juyal
EP20BTECH11018

Imports

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import emcee
from scipy import optimize
from sklearn.neighbors import KernelDensity
import nestle
from astropy import stats as stats_astropy
from astroML import stats as stats_astroML
```

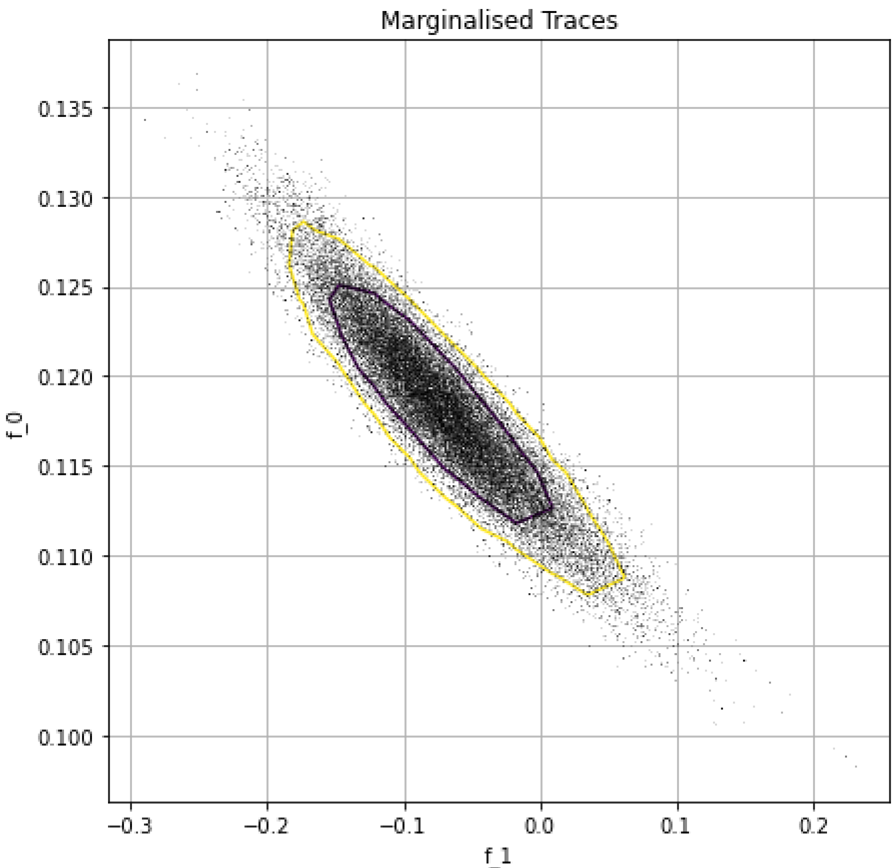
Question 1

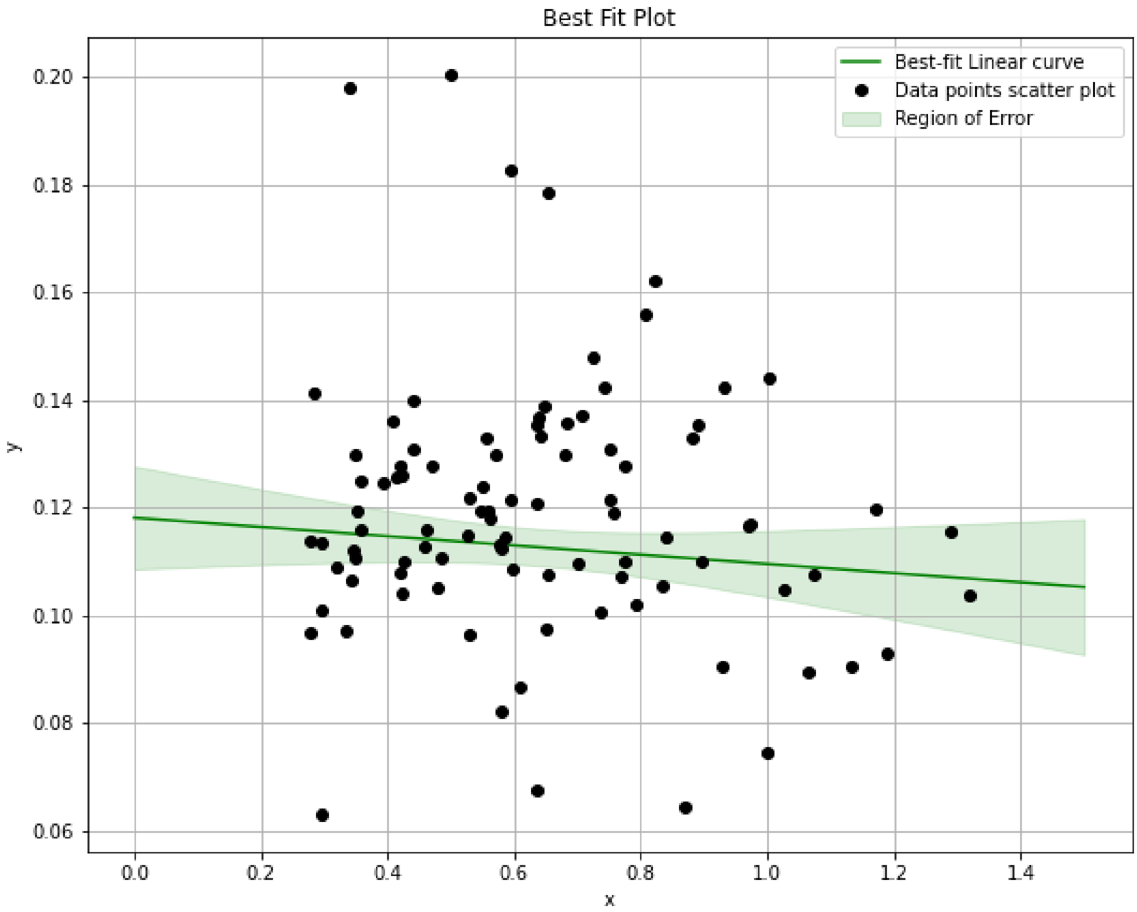
Download the SPT fgas data from http://iith.ac.in/~shantanud/fgas_spt.txt. Fit the data to f0(1 + f1z) where f0 and f1 are unknown constants. Determine the best fit values of f0 and f1 including 68% and 90% credible intervals using emcee and corner.py.
The priors on f0 and f1 should be 0 < f0 < 0.5 and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

In [ ]:
```
# extracting data from the website
data = np.loadtxt('https://iith.ac.in/~shantanud/fgas_spt.txt')
z = data[:, 0]
fgas = data[:, 1]
fgas_error = data[:, 2]

# utility function for log prior
def log_prior(theta):
    m, b = theta
    if -0.5 < m < 0.5 and 0 < b < 0.5:
        return 0.0
    return -np.inf

# utility function for log likelihood
def log_llihood(theta, x, y, y_error):
    m, b = theta
    y_model = b*(1+m*x)
    return -0.5 * np.sum(((y_model - y)/y_error)**2)

# function for log posterior
def log_posterior(theta, x, y, y_error):
    log_pr = log_prior(theta)
    if not np.isfinite(log_pr):
        return -np.inf
    return log_pr + log_llihood(theta, x, y, y_error)
```

```
# MCMC walkers
nwalkers = 50
# number of final posterior samples
nsamples = 1000
# burn period
nburn = 1000
# number of MCMC nsteps
nsteps = nburn + nsamples

# initial guesses
guesses = np.array([np.random.uniform(-0.5, 0.5, nwalkers), np.random.uniform(0.
# number of dimensions
ndims = guesses.shape[1]

sampler = emcee.EnsembleSampler(nwalkers, ndims, log_posterior, args=[z,fgas,fga
sampler.run_mcmc(guesses, nsteps)
samples = sampler.chain[:, nburn:, :].reshape((-1, ndims))

# function for calculating sigma level
def sigma_level(t1, t2, nbins=20):
    L, xbins, ybins = np.histogram2d(t1, t2, nbins)
    L[L == 0] = 1E-16

    shape = L.shape
    L = L.ravel()

    i_sort = np.argsort(L)[::-1]
    i_unsort = np.argsort(i_sort)
    # cumulative sum
    L_cumsum = L[i_sort].cumsum()
    L_cumsum /= L_cumsum[-1]

    sigma = L_cumsum[i_unsort].reshape(shape)

    xbins = 0.5 * (xbins[1:] + xbins[:-1])
    ybins = 0.5 * (ybins[1:] + ybins[:-1])
    return xbins, ybins, sigma

# plotting marginalised traces
trace = samples.T
xbins, ybins, sigma = sigma_level(trace[0],trace[1])

plt.figure(figsize=(7,7))
plt.contour(xbins, ybins, sigma.T, levels=[0.68,0.90])
plt.plot(trace[0], trace[1], ',k', alpha=0.15)

plt.title("Marginalised Traces")
plt.xlabel('f_1')
plt.ylabel('f_0')
plt.grid()
plt.show()

# plotting best fit curves
x1 = np.linspace(0, 1.5, 2000)
m, b = trace[:2]
y1 = b[:,None]*(1+m[:,None]*x1)
# for error region bounds
bound1 = y1.mean(0)-2*y1.std(0)
bound2 = y1.mean(0)+2*y1.std(0)
```

```
plt.figure(figsize=(10,8))
plt.plot(x1, y1.mean(0), label='Best-fit Linear curve',color='g')
plt.plot(z, fgas, 'ok', label='Data points scatter plot')
plt.fill_between(x1, bound1, bound2, alpha=0.15, color = 'g', label='Region of E

plt.xlabel('x')
plt.ylabel('y')
plt.title("Best Fit Plot")
plt.grid()
plt.legend()
plt.show()
```





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)?

In [ ]:
```
data = 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.
                 [0.33,  0.41, -0.22,  0.01, -0.05, -0.05, -0.12,  0.26,  0.29,
                  0.31,  0.42, -0.01,  0.58, -0.2, 0.52,  0.15,  0.32, -0.13, -0
                 [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]])

# function for polynomial fit
def polynomial_fit(theta, x):
    return sum(t * x ** n for (n, t) in enumerate(theta))

# function for log likelihood
def log_llihood(theta, data=data):
    x, y, sigma_y = data
    yM = polynomial_fit(theta, x)
    return -0.5 * np.sum(np.log(2 * np.pi * sigma_y ** 2) + (y - yM) ** 2 / sign

# function for prior
def prior(theta):
    return 200*theta - 100
```

```
# printing required things
x, y, sigma_y = data
result_lin = nestle.sample(log_llihood, prior, 2)
print(result_lin.summary())
result_quad = nestle.sample(log_llihood, prior, 3)
print()
print(result_quad.summary())
```

```
niter: 1575
ncall: 2745
nsamples: 1675
logz:  7.121 +/-  0.373
h: 13.878

niter: 2037
ncall: 4861
nsamples: 2137
logz:  3.369 +/-  0.425
h: 18.094
```

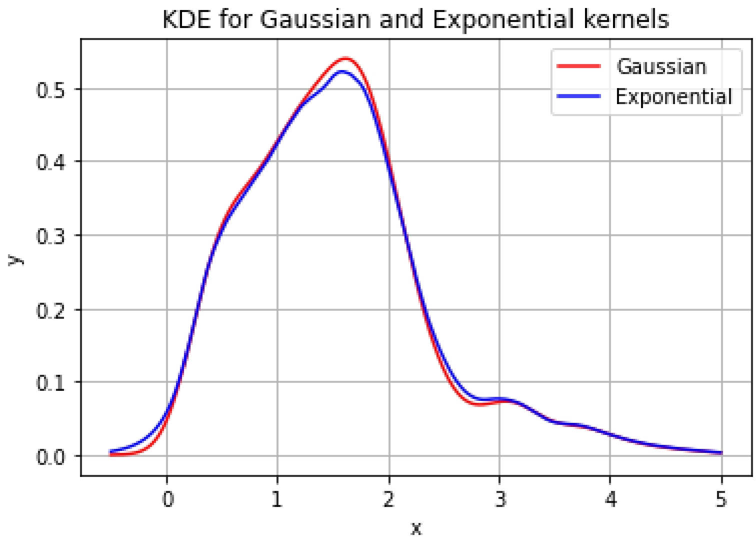
The values do not match with what is on the blog.

Question 3

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.

In [ ]:

```
# importing the input data
dataframe = pd.read_csv('SDSS_quasar.dat',sep = '\s+')
data2 = dataframe['z']
data = data2.to_numpy()

# points on x-axis
x_1 = np.linspace(-0.5, 5, 1000)

# KDE estimate using gaussian kernel
log_pdf_gaus = KernelDensity(kernel='gaussian', bandwidth=0.2).fit(data[:,np.new
pdf_g = np.exp(log_pdf_gaus)

# KDE estimate using exponential kernel
log_pdf_exp = KernelDensity(kernel='exponential', bandwidth=0.2).fit(data[:,np.r
pdf_exp = np.exp(log_pdf_exp)

# plot
plt.plot(x_1, pdf_g, 'r', label = 'Gaussian')
plt.plot(x_1, pdf_exp, 'b', label = 'Exponential')
plt.title('KDE for Gaussian and Exponential kernels')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.grid()
plt.show()
```


KDE for Gaussian and Exponential kernels
Gaussian
Exponential
0.5
0.4
0.3
0.2
0.1
0.0
y
0
1
2
3
4
5
x