question 1

In [400…
```
from math import sqrt, pi
import numpy as np
from scipy.stats import bernoulli, multivariate_normal, norm, invgamma
from tabulate import tabulate
import json
import os
import pandas as pd
import matplotlib as mpl
import seaborn as sns
```

$log(L) = \frac{n}{2}log(2\pi(\sigma)^2) - \frac{\sum x^2}{2(\sigma)^2} + \frac{\sum 2\mu x}{2(\sigma)^2} - \frac{n\mu^2}{2(\sigma)^2} - \frac{(\mu)^2}{200} + 3ln((\sigma)^2)$

$-(\sigma)^2 10$

the first derivative of μ = $\frac{\sum x}{(\sigma)^2} - \frac{n\mu}{((\sigma)^2)} - \frac{\mu}{10} = 0$

the first derivative of $(\sigma)^2$ = $\frac{n}{2(\sigma)^2} + \frac{\sum x^2}{2((\sigma)^2)^2} + \frac{\sum 2x\mu}{2((\sigma)^2)^2} - \frac{n(\mu)^2}{2((\sigma)^2)^2} + \frac{3}{(\sigma)^2} - 10 = 0$

In [300…
```
pos = norm(50,sqrt(10))
xx =pos.rvs(100)
```

In [140…
```
y_sum = np.sum(xx)
ys_sum = np.sum(xx **2)
print(y_sum,ys_sum)
```

```
5111.737683835008 272659.9717441508
```

In [213…
```
ys_sum = np.sum(xx-50.89498)
ys_sum*10/50.89498
```

Out[213]:
```
14.054667550628878
```

we plug $\sum x$ = 5111.737683835008, $\sum x^2$ = 5111.737683835008, n = 100 to the above equation

and then get results μ = 50.89498

var = 14.054667550628878

second derivative of σ^2 = $\frac{n+6}{2((\sigma)^2)^2} - \frac{\sum(y-\mu)^2+20}{2((\sigma)^2)^3}$

then we calculate the second derevative of μ = $\frac{-1}{10} - \frac{n}{((\sigma)^2)}$

derivative of μ and $(\sigma)^2$ = $\frac{n\mu-\sum y}{2((\sigma)^2)^2}$

In [214…
```
mu_est = 50.89498
var_est = 14.054667550628878
```

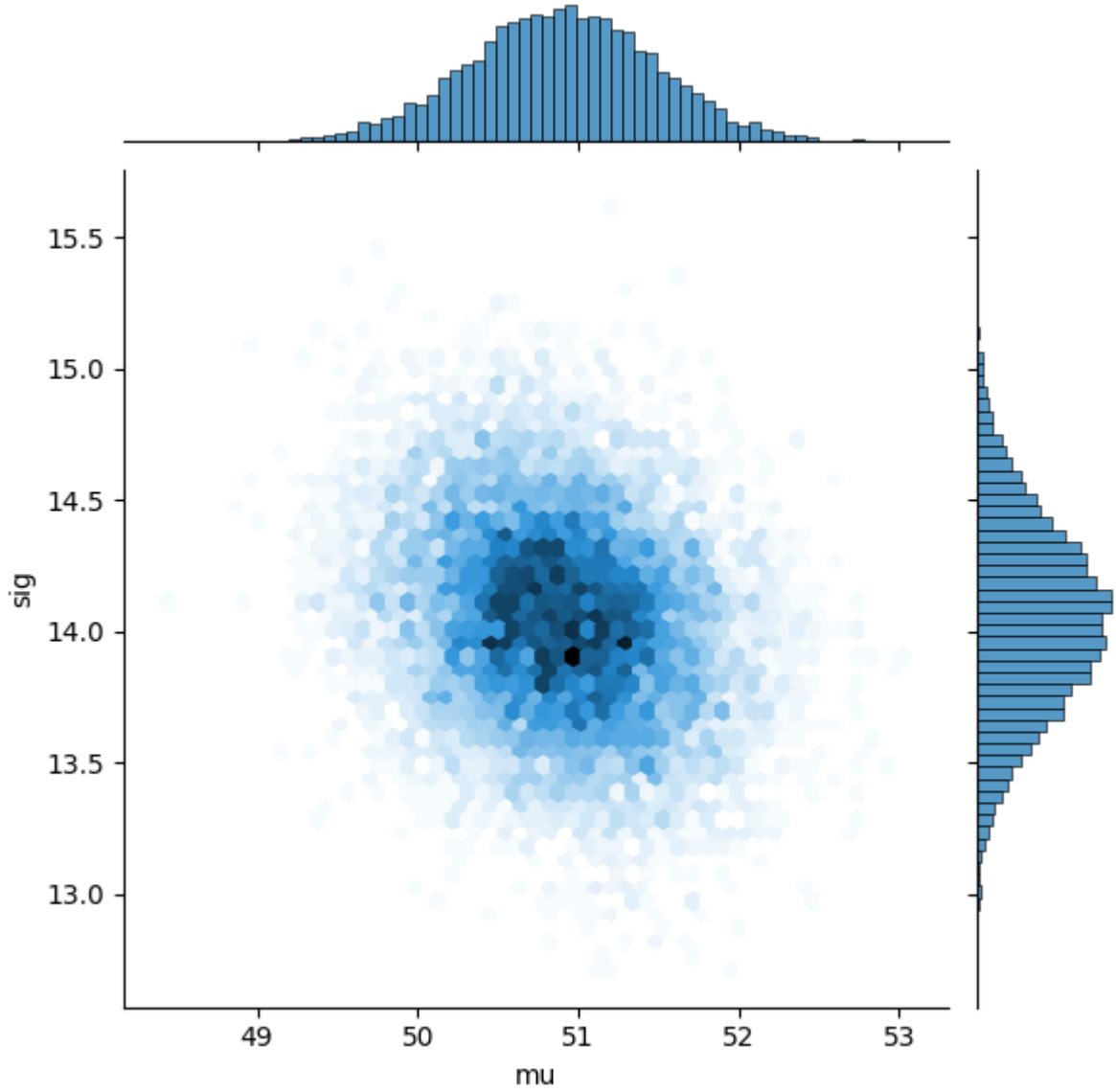
In [270…
```
var1 = (-100/var_est) -(1/10)
var2 = (10*mu_est - y_sum)/(var_est)**2
var3 = var2
var4 = (100 + 6)/(2*(var_est)**2) -(np.sum((xx - 50.89498)**2) + 20)/(2*(var_est)**3)
```

In [271…
```
var = [[-var1,-var2],[-var3,-var4]]
cov = np.inv(var)
```

In [286…
```
mean =[50.89498,14.054667550628878]
pos = multivariate_normal(mean,cov)
map = pos.rvs(10000)
map =pd.DataFrame(map)
map.columns =['mu', 'sig']
```

In [287…
```
sns.jointplot(data = map, x="mu", y="sig", kind="hex")
```

Out[287]: <seaborn.axisgrid.JointGrid at 0x129541840>



In [ ]:
```
### questin 2
```

In [403…
```
m = 0
v = 10
b = 10
a = 1
```

In [406…
```
for n in range(1000):
    m = (100*y_bar)/(100 + (b/a)*(1/10))
    v = sqrt((b/a) /(100 + (b/a)*(1/10)))
    b = (a*(100*(v**2) + np.sum((xx-m) ** 2) + 2*10))/(2 + 100)
```

In [419…
```
print(m, b, v)
```

```
50.417364162874634 8.905816223503864 0.2971060884229268
```

In [411…
```
posmu = norm(50.4, 0.297)
xxmu = pos.rvs(1000)
```

```
possig = invgamma(1,8.9)
xxsig = possig.rvs(1000)
```

In [412…
```
df = pd.DataFrame({'mu': xxmu,
                   'sig':xxsig})
df
```

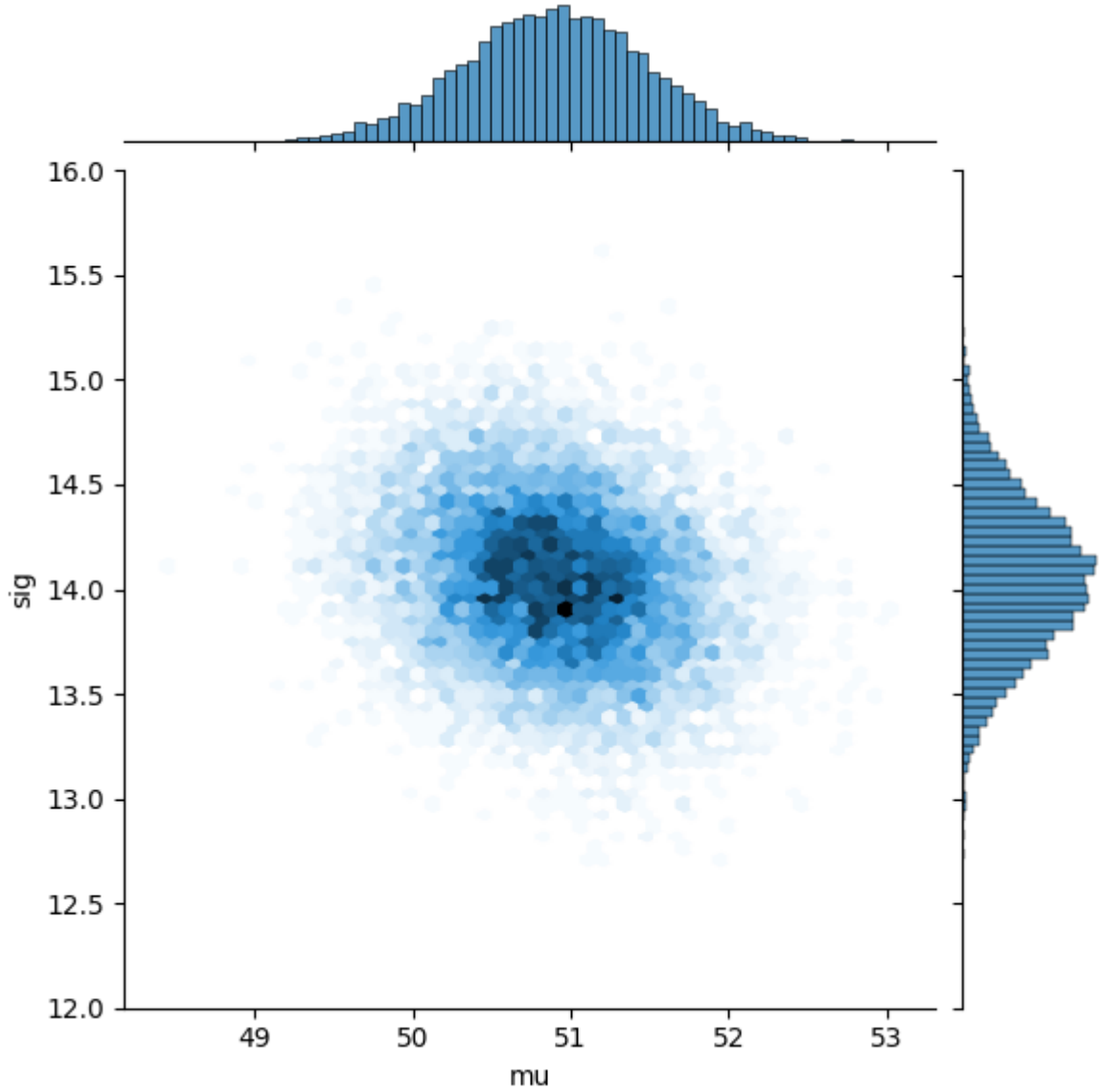
Out[412]:

	mu	sig
0	43.174544	11.143568
1	44.481720	10.560856
2	46.822444	33.109367
3	52.219187	9.036808
4	52.298720	10.698409
…	…	…
995	46.587481	11.403187
996	48.080608	17.899383
997	47.714017	11.388299
998	48.681686	9.161553
999	52.434901	10.690855

1000 rows × 2 columns

In [418…
```
sns.jointplot(data = map, x="mu", y="sig", kind="hex", ylim = (12, 16))
```

Out[418]:
```
<seaborn.axisgrid.JointGrid at 0x18ea1ef50>
```



In [ ]:

question 2

In [18]:
```
N = 200
w = 0.5
c_mean = 0
```

In [149…
```
mean = [0, 0]
c_cov_0 = [[10, 0], [0, 10]]
cov = [[1, 0], [0, 1]]
theta = [5,5]
cov
```

Out[149]:
```
[[1, 0], [0, 1]]
```

In [82]:
```
clutter = multivariate_normal(mean,c_cov_0)
theta = [5,5]
target_dist = multivariate_normal(theta,cov_0)
sample_1 = bernoulli(w).rvs
samples = [(clutter.rvs() if sample_1() else target_dist.rvs())
               for _ in range(N)]
X = samples
```

In [161…
```
# initialise
D = 2
p_mean = [0,0]
```

```
p_var = 100
p_s = (2 * pi * p_var) ** (-D/2)
est_mean = p_mean
est_var = p_var
fn_mean = 0
fn_var = np.infty
```

In [162…
```
pi = 3.1415
```

In [163…
```
def gaus(x, m, v):
    return np.exp(-0.5 * ((x[0] - m[0]) ** 2) * (1 / v)) / ((abs(2 * pi * v)) ** 0.5)
```

In [164…
```
converged = False
```

In [165…
```
f_means = [0,0]
f_vars = 100
f_ss = 1
```

In [ ]:
```
for iteration in range(3):
        if converged:
            break
        for n in range(200):
            v_theta = 1 / ((1/est_var) - (1/fn_var))
            mean_theta = est_mean + (v_theta * (1/fn_var) * (est_mean - fn_mean))
            Zn = ((1 - w) * gaus(X[n], mean_theta, (v_theta + 1))) + (w * gaus(X[n],
            pi_0 = 1 - ((w/Zn) * gaus(X[n],[0,0], 10))
            est_mean = mean_theta + (pi_0 * v_theta * (X[n] - mean_theta) /
                                     (v_theta + 1))
            est_var = v_theta - (pi_0 * (v_theta ** 2) / (v_theta + 1)) + \
                (pi_0 * (1 - pi_0) * (v_theta ** 2) * (((abs(X[n][0] - mean_theta[0])
                 (abs(X[n][1] - mean_theta[1])) ** 2))/ (D * ((v_theta + 1) ** 2))
            if est_var != v_theta:
                fn_var_new = 1 / ((1 / est_var) - (1 / v_theta))
                fn_mean_new = mean_theta + ((fn_var_new + v_theta) * (1 / v_theta) *
                fn_ss_new = Zn / (((2 * pi * abs(fn_var_new)) ** (D / 2)) * gaus(fn_m
                # Check for convergence
                if abs(f_means[0] - fn_mean_new[0]) > tol and \
                    abs(f_means[1] - fn_mean_new[1]) > tol and \
                    abs(f_vars- fn_var_new) > tol and \
                        abs(f_ss- fn_ss_new) > tol:
                    converged = False
                else:
                    converged = True
                    fn_means = fn_mean_new
                    fn_vars = fn_var_new
                    f_ss = fn_ss_new
```

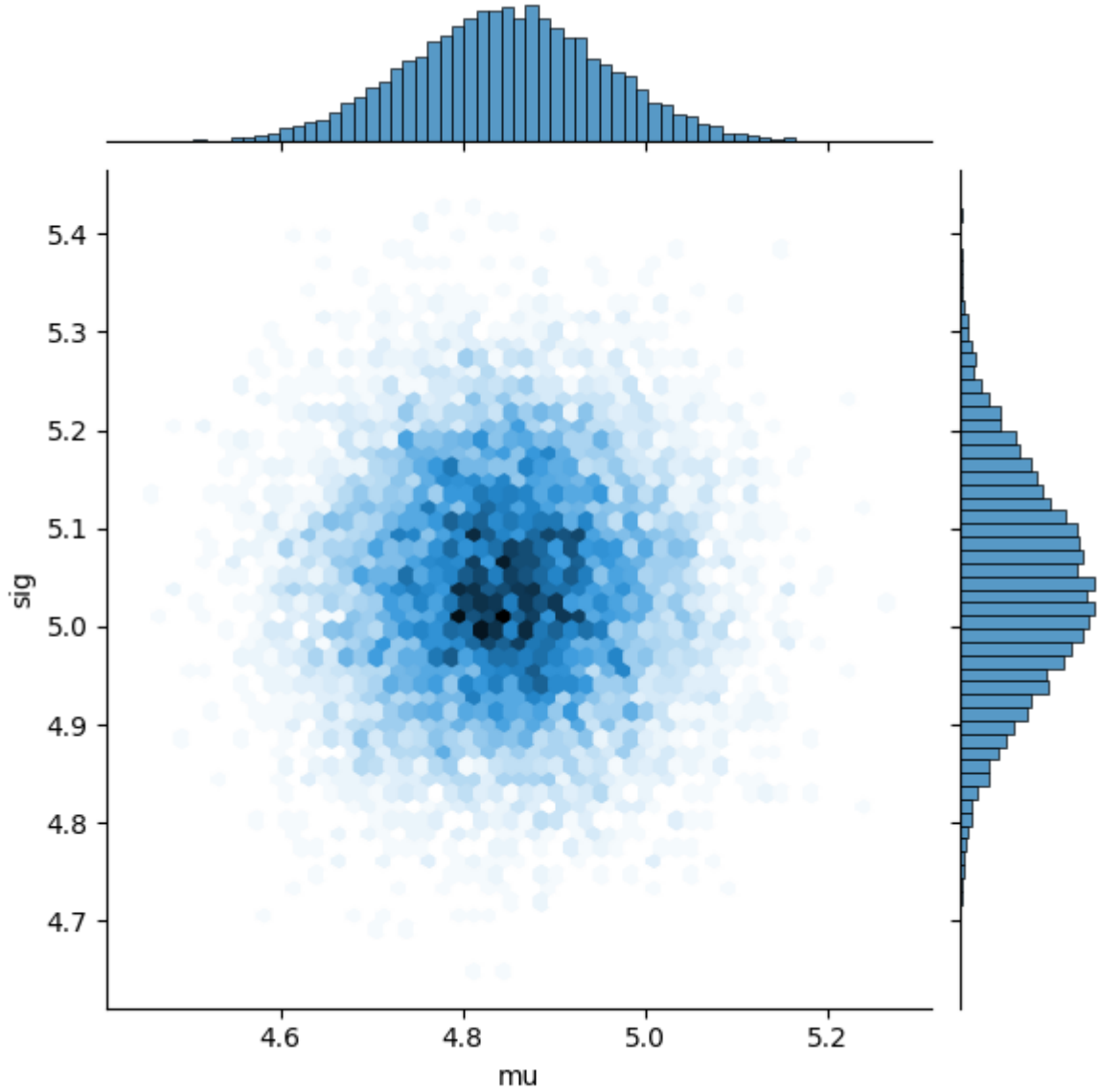
In [319…
```
print(est_var)
```

```
0.011753844927614507
```

In [39]:
```
cov = [[0.011753844927614507, 0],[0, 0.011753844927614507]]
mean =[4.84932703, 5.04245692]
pos = multivariate_normal(mean,cov)
xx = pos.rvs(10000)
xx =pd.DataFrame(xx)
xx.columns =['mu', 'sig']
```

In [40]:
```
sns.jointplot(data=xx, x="mu", y="sig", kind="hex")
```

Out[40]:
```
<seaborn.axisgrid.JointGrid at 0x12792dd80>
```



ADF

In [309…]
```
s = 1
est_mean = [0,0]
v_theta = 100
```

In [ ]:
```
  for n in range(200):
            Zn = ((1 - w) * gaus(X[n], mean_theta, (v_theta + 1))) + (w * gaus(X[n],
            pi_0 = 1 - ((w/Zn) * gaus(X[n],[0,0], 10))
            s = s * Zn
            new_v = v_theta - (pi_0 * (v_theta ** 2) / (v_theta + 1)) + \
                (pi_0 * (1 - pi_0) * (v_theta ** 2) * (((abs(X[n][0] - mean_theta[0])
                 (abs(X[n][1] - mean_theta[1])) ** 2))/ (D * ((v_theta + 1) ** 2))
            mean_theta = mean_theta + (pi_0 * v_theta * (X[n] - mean_theta) /
                                       (v_theta + 1))
            v_theta = new_v
```

In [ ]:
```
print(new_v)
print(mean_theta)
```

In [36]:
```
cov = [[0.011717634739124731, 0],[0, 0.011717634739124731]]
mean =[4.84774685, 5.0458862]
pos = multivariate_normal(mean,cov)
xx = pos.rvs(10000)
xx =pd.DataFrame(xx)
xx.columns =['mu', 'sig']
```

In [38]:
```
sns.jointplot(data=xx, x="mu", y="sig", kind="hex")
```

Out[38]:
```
<seaborn.axisgrid.JointGrid at 0x1279bee00>
```