In [267]:

```python
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from scipy.stats import norm
pd.set_option('display.max_colwidth', None)
from scipy.optimize import minimize
from scipy.optimize import fsolve
from scipy.integrate import quad
import scipy.integrate as integrate
import scipy.special as special
import math
```

In [342]:

```python
xls = pd.ExcelFile('Dataset_ Exam 2.xlsx')
df = pd.read_excel(xls, 'Question_2 Estimation Dataset')
df.columns = df.columns.map(str)
```

In [343]:

```python
df.head(2)
```

Out[343]:

| | t | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 0.202074 | 0.189858 | 0.217837 | 0.208467 | 0.193658 | 0.198569 | 0.189084 | 0.209849 | 0.204434 |
| **1** | 1 | 0.202074 | 0.202074 | 0.202074 | 0.202074 | 0.202074 | 0.202074 | 0.202074 | 0.202074 | 0.202074 |

2 rows × 21 columns

## Q2 (c)

https://stackoverflow.com/questions/8739227/how-to-solve-a-pair-of-nonlinear-equations-using-python
(https://stackoverflow.com/questions/8739227/how-to-solve-a-pair-of-nonlinear-equations-using-python)
https://stackoverflow.com/questions/41687908/python-nsolve-solve-triple-of-equations
(https://stackoverflow.com/questions/41687908/python-nsolve-solve-triple-of-equations)

In [344]:

```python
signals_component = {}
for i in range(20):
    signals_component[i+1] =np.array(df[str(i+1)].dropna())
```

In [345]:

```python
phi = np.mean([v[0] for k,v in signals_component.items()])
ld = phi
D2 = 21
phi
```

Out[345]:

```
0.20115499998656788
```

In [346]:

```python
f_times_dict = {k:len(v)-1 for k,v in signals_component.items()}
f_times_list = np.array([len(v)-1 for k,v in signals_component.items()])
n=20
```

In [351]:

```
def equations(p):
    c, a = p
    return (n/c+(1/a)*np.sum([(1-c/ti)/ti for ti in f_times_list]),
            -n/(2*a)+1/(2*a*a)*np.sum([(1-c/ti)**2 for ti in f_times_lis
t]))

c, a =  fsolve(equations, (1, 1),maxfev=100000)
c,a
```

Out[351]:

```
(170.70136796254243, 0.20218223173679964)
```

In [352]:

```
mu0 = (D2-ld)/c
sigma0_2 = a*mu0**2
print(mu0,sigma0_2)
```

```
0.12184345824678683 0.003001562701220904
```

In [353]:

```
thetas = []
sigmas = []
phis = []
for j in range(20):
    i = j+1
    length = len(signals_component[i])
    X = np.array(range(length)).reshape((-1,1))
    y = signals_component[i]
    reg = LinearRegression().fit(X, y)
    thetas.append(reg.coef_)
    phis.append(reg.intercept_)
    if j:
        sigmas.append(np.sum(np.array([y[x]-reg.predict(np.array([[x]]))
for x in range(1,length)])**2)/(j*(length-1)))
```

In [354]:

```
sigma_2 = np.mean(sigmas)
```

In [355]:

```
phi,sigma_2,mu0,sigma0_2
```

Out[355]:

```
(0.20115499998656788,
 0.28904353668711674,
 0.12184345824678683,
 0.003001562701220904)
```

## Q2 (d)

In [356]:

```
df2 = pd.read_excel(xls, 'Question 2 Prediction Dataset')
df2.columns = df2.columns.map(str)
```

In [357]:

```
D2 = 21
```

In [358]:

```
df_pred = {}
for col in df2.columns[1:]:
    df_pred[col] = np.array(df2[col].dropna())
```

In [359]:

```
def mu_p(S):
    length = len(df_pred[S])
    return (sigma0_2*np.sum([(x-phi)*t for t,x in enumerate(df_pred[S
])])+mu0*sigma_2)/ \
        (sigma0_2*np.sum(np.arange(length)**2)+sigma_2)
def sigma_p(S):
    length = len(df_pred[S])
    return (sigma_2*sigma0_2)/(sigma0_2*np.sum(np.arange(length)**2)+sig
ma_2)
def g(t,S):
    length = len(df_pred[S])
    return (mu_p(S)*(t+length-1)+phi-D2)/np.sqrt(sigma_p(S)*(t+length-1)
**2+sigma_2)
def F_T(t,S):
    return (norm.cdf(g(t,S))-norm.cdf(g(0,S)))/(1-norm.cdf(g(0,S)))
```

In [360]:

```
F_T(521.579,'21')
```

Out[360]:

```
0.5029152065869491
```

```
In [361]: for j in range(21,31):
              i = str(j)
              def R_(t):
                  return F_T(t,i)-0.5
              def R(t):
                  return 1-F_T(t,i)
              soln = fsolve(R_, [10])
              print(i,soln)
              print(F_T(soln,i))
              print(integrate.quad(R, 0, 10000)[0])
```

```
21 [521.43031]
[0.5]
521.5795166045941
22 [61.51160685]
[0.5]
61.517269531481794
23 [151.06243708]
[0.5]
151.07444379163186
24 [172.06208475]
[0.5]
172.07284336303266
25 [93.03722004]
[0.5]
93.0420935439947
26 [97.78390989]
[0.5]
97.7889045963119
27 [135.3130576]
[0.5]
135.31987697263912
28 [109.28061199]
[0.5]
109.28902218155073
29 [13.94950154]
[0.5]
14.105182176041094
30 [39.55214782]
[0.5]
39.55522292785549
```

## Q2 (e)

In [335]:

```
for j in range(21,31):
    length = len(df_pred[str(j)])
    X = np.array(range(length)).reshape((-1,1))
    y = df_pred[str(j)]
    reg = LinearRegression().fit(X, y)
    mu0 = reg.coef_

    i = str(j)
    phi = df_pred[i][0]
    def R_(t):
        return F_T(t,i)-0.5
    def R(t):
        return 1-F_T(t,i)
    soln = fsolve(R_, [10])
    print(i,soln)
    print(F_T(soln,i))
    print(integrate.quad(R, 0, 10000)[0])
phi = 21
mu0 = (D2-ld)/c
sigma0_2 = a*mu0**2
```

```
21 [524.97389045]
[0.5]
525.1257143862829
22 [61.68449423]
[0.5]
61.69017914293935
23 [150.80813423]
[0.5]
150.82010244864986
24 [171.95504664]
[0.5]
171.9657927526233
25 [92.86077895]
[0.5]
92.86564047841111
26 [97.46025122]
[0.5]
97.46522428612107
27 [135.74933923]
[0.5]
135.75619242745955
28 [108.93570563]
[0.5]
108.9440708382618
29 [13.94149394]
[0.5]
14.097446100242196
30 [39.54294548]
[0.5]
39.54601995867078
```

## Q3

In [362]:

```
df3 = pd.read_excel(xls, 'Question 3 Repair & Inventory')
df3.columns = df3.columns.map(str)
```

In [363]:

```
df3 = df3.drop(columns='t')
df3 = np.array(df3['9'])
df_pred['9'] = df3
```

In [364]:

```
df_pred.keys()
```

Out[364]:

```
dict_keys(['21', '22', '23', '24', '25', '26', '27', '28', '29', '30',
'9'])
```

In [446]:

```
i = '9'

def R_(t):
    return F_T(t,i)-0.5
def R(t):
    return 1-F_T(t,i)
soln = fsolve(R_, [100])
```

In [378]:

```
cp = 50
cf = 100
def Cr(tr):
    return (cp*(1-F_T(tr,i))+cf*F_T(tr,i))/integrate.quad(lambda x:1-F_T
(x,i),0,tr)[0]
```

In [389]:

```
Cr(170)
```

Out[389]:

```
0.3011970157961235
```

In [393]:

```
res = minimize(Cr,[1])
res
```

Out[393]:

```
      fun: 0.301163111631661
 hess_inv: array([[2932.56255828]])
      jac: array([-6.77630305e-06])
  message: 'Optimization terminated successfully.'
     nfev: 46
      nit: 21
     njev: 23
   status: 0
  success: True
        x: array([170.43503861])
```

In [396]:

```
opt_tr = res.x[0]
```

In [442]:

```
kh = 0.5
ks = 200
L = 7
def Co(to,tr=res.x[0]):
    return (ks*integrate.quad(lambda x:F_T(x,i),to,to+L)[0]+kh*integrate
.quad(lambda x:1-F_T(x,i),to+L,tr)[0])/ \
            (integrate.quad(lambda x:F_T(x,i),to,to+L)[0]+integrate.quad
(lambda x:1-F_T(x,i),0,tr)[0])
```

In [439]:

```
res2 = minimize(Co,[1])
```

In [405]:

```
res2
```

Out[405]:

```
      fun: 0.03404403447655391
 hess_inv: array([[1201.06519859]])
      jac: array([-1.96322799e-06])
  message: 'Optimization terminated successfully.'
     nfev: 46
      nit: 5
     njev: 23
   status: 0
  success: True
        x: array([154.87290223])
```

In [423]:

```
def C_combined(var):
    to,tr = var
    return (ks*integrate.quad(lambda x:F_T(x,i),to,to+L)[0]+kh*integrate
.quad(lambda x:1-F_T(x,i),to+L,tr)[0])/ \
            (integrate.quad(lambda x:F_T(x,i),to,to+L)[0]+integrate.quad
(lambda x:1-F_T(x,i),0,tr)[0]) \
            +(cp*(1-F_T(tr,i))+cf*F_T(tr,i))/integrate.quad(lambda x:1-F
_T(x,i),0,tr)[0]
```

In [424]:

```
cons = ({'type': 'ineq', 'fun': lambda x:  x[1] - x[0] - 7})
res3 = minimize(C_combined,(1,1),constraints=cons)
```

In [425]:

```
res3
```

Out[425]:

```
     fun: 0.3182976090392948
     jac: array([-0.00129464,  0.00129038])
 message: 'Optimization terminated successfully'
    nfev: 92
     nit: 29
    njev: 29
  status: 0
 success: True
       x: array([153.08898943, 160.08898943])
```

In [445]:

```
def total_cost(to,tr):
    return Co(to,tr)+Cr(tr)
print(total_cost(*res3.x))
print(total_cost(*res2.x,*res.x))
```

```
0.3182976090392948
0.3352071461082149
```

In [434]:

```
print(*list(res3.x))
```

```
153.08898943255065 160.0889894325496
```

In [ ]: