ECO423A: Financial Economics (Assignment-2)

Group Members:

1. Sachin Angural (160597)
2. Anurag Chanani (160138)
3. Akash Yadav (160071)
4. Mohit Sheoran (150418)
5. Ramesh Patel (170551)

In [1]:
```
import pandas as pd
import numpy as np
import scipy.stats as st
```

In [2]:
```
data = pd.read_excel(r'/home/sachin/Downloads/ass2.xlsx')
data.columns = ['Date', 'Jet_Kero_Price',  'USD_AUD', 'AUD_USD']
data.set_index("Date",inplace=True)
```

In [3]:
```
data.describe()
```

Out[3]:

	Jet_Kero_Price	USD_AUD	AUD_USD
count	301.000000	301.000000	301.000000
mean	454.042027	0.765262	1.351388
std	318.957676	0.139718	0.252114
min	105.000000	0.487400	0.923446
25%	181.000000	0.674600	1.186662
50%	298.200000	0.757900	1.319435
75%	680.250000	0.842700	1.482360
max	1406.050000	1.082900	2.051703

In [4]:
```
data.head()
```

Out[4]:

	Jet_Kero_Price	USD_AUD	AUD_USD
Date			
1990-09-04	320.0	0.8146	1.227596
1990-10-04	481.5	0.8307	1.203804
1990-11-04	385.0	0.7780	1.285347
1990-12-04	328.5	0.7760	1.288660
1991-01-04	255.0	0.7782	1.285017

Question 1

In [5]:
```
price_Jetfuel_sept = 500.9
USDperAUD_sept = 0.6972
Number_of_flights_sept = 522
Fixed_cost = 11*1e6
Earning_sept = ((40000 - (7000 + (price_Jetfuel_sept*13)))*Number_of_flights_sept - Fixed_cost)/ USDperAUD_sept

print("Answer to Question 1:")
print("The Earnings before tax for JetSafe for the month of September 2015 is : {:0.4f} million AUD".format(Earning_sept/1e6))
```

```
Answer to Question 1:
The Earnings before tax for JetSafe for the month of September 2015 is : 4.0546 million AUD
```

Question 2

In [6]:
```
j = np.array(data.Jet_Kero_Price)
k = np.array(data.AUD_USD)
j_ = np.diff(j)
k_ = np.diff(k)
jet_ret = j_/j[:-1]
AUD_USD_ret = k_/k[:-1]
```

In [7]:
```
std_jet = np.std(jet_ret)*100
std_AUD_USD = np.std(AUD_USD_ret)*100
mean_jet = np.mean(jet_ret)*100
mean_AUD_USD = np.mean(AUD_USD_ret)*100
geomean_jet = (np.prod((jet_ret[:]+1))**(12/301)-1)*100
geomean_aud_usd = (np.prod((AUD_USD_ret[:]+1))**(12/301)-1)*100
```

In [8]:
```
print("Answers to Question 2 :")
print("Volatility per month for Jet kerosene returns in USD : {:0.4f}%".format(std_jet))
print("Volatility per annum for Jet kerosene returns in USD : {:0.4f}%".format(std_jet*(12)**0.5))
print("Mean per month for Jet kerosene returns in USD : {:0.4f}%".format(mean_jet))
print("Mean per annum for Jet kerosene returns in USD : {:0.4f}%  (Calculated simply)".format(mean_jet*(12)))
print("Geometric average rate of return for Jet kerosene returns in USD : {:0.4f}% (Calculated compoundedly)".format(geomean_jet))
print("Volatility per month for AUD/USD returns : {:0.4f}%".format(std_AUD_USD))
print("Volatility per annum for AUD/USD returns : {:0.4f}%".format(std_AUD_USD*(12)**0.5))
print("Mean per month for AUD/USD returns : {:0.4f}%".format(mean_AUD_USD))
print("Mean per annum for AUD/USD returns : {:0.4f}% (Calculated simply)".format(mean_AUD_USD*(12)))
print("Geometric average rate of return for AUD/USD returns : {:0.4f}% (Calculated compoundedly)".format(geomean_aud_usd))
```

```
Answers to Question 2 :
Volatility per month for Jet kerosene returns in USD : 10.1114%
Volatility per annum for Jet kerosene returns in USD : 35.0271%
Mean per month for Jet kerosene returns in USD : 0.6661%
Mean per annum for Jet kerosene returns in USD : 7.9934%  (Calculated simply)
Geometric average rate of return for Jet kerosene returns in USD : 1.8024% (Calculated compoundedly)
Volatility per month for AUD/USD returns : 3.3926%
Volatility per annum for AUD/USD returns : 11.7522%
Mean per month for AUD/USD returns : 0.1088%
Mean per annum for AUD/USD returns : 1.3052% (Calculated simply)
Geometric average rate of return for AUD/USD returns : 0.6224% (Calculated compoundedly)
```

Question 3 & 4

In [9]:
```
z_val_jet = (-23-(mean_jet))/(std_jet)
norm_prob = st.norm.sf(abs(z_val_jet))

emp_prob = (jet_ret*100 < -23).sum()/len(jet_ret)
```

In [10]:
```
print("Answers to Question 3 & 4 :")
print("Probability that in any given month jet kerosene returns in USD will be lower than -23%, assuming normality : {:0.4f}".format(norm_prob))
print("Corresponding z-value : {:0.4f}".format(z_val_jet))
print("Actual probability that in any given month jet kerosene returns in USD will be lower than -23%, empirically : {:0.4f}".format(emp_prob))
```

```
Answers to Question 3 & 4 :
Probability that in any given month jet kerosene returns in USD will be lower than -23%, assuming normality : 0.0096
Corresponding z-value : -2.3405
Actual probability that in any given month jet kerosene returns in USD will be lower than -23%, empirically : 0.0100
```

Implications

1. The normality assumption gives a smaller value of probablity of returns lower than -23% compared to empirical value, though the values are very close.
2. This reflects the fact that the returns may follow Normal distribution with mean & standard deviation as calculated above.

Question 5

In [11]:
```
corr_coeff = np.corrcoef(jet_ret, AUD_USD_ret)[0][1]
print("Answers to Question 5 :")
print("The correlation coefficient between jet kerosene returns and AUD/USD returns is : {:0.4f}".format(corr_coeff))
```

```
Answers to Question 5 :
The correlation coefficient between jet kerosene returns and AUD/USD returns is : -0.2929
```

Implications

1. The correlation coefficient is a measure of linear relationship between the variables.
2. The negative value shows a negative relationship i.e. the variables move in opposite directions.
3. However, this relationship is not very strong in our case as the value is significantly smaller than 1.

Question 6

In [12]:
```
std_jetfuel = .1
std_AUDinUSD = .035
rho = 0.3
covar = rho*std_jetfuel*std_AUDinUSD
covar_mat = [[std_jetfuel**2, covar],[covar, std_AUDinUSD**2]]
mean = [0,0]
```

In [13]:
```
def simulate(n):
    return np.random.multivariate_normal(mean, covar_mat, [12,n])
```

In [14]:
```
def Simulator(initial_price_jetfuel, initial_AUDinUSD, initial_flights,n=1):
    s = simulate(n)
    Flight_deviation = 0.8
    Earning, price_jetfuel, price_AUDinUSD,Number_of_flights =[0]*13, [0]*13, [0]*13, [0]*13
    price_jetfuel[0] = np.array([initial_price_jetfuel]*n)
    price_AUDinUSD[0] = np.array([initial_AUDinUSD]*n)
    Number_of_flights[0] = np.array([initial_flights]*n)
    Earning[0] = np.array([Earning_sept]*n)

    for i in range(1,13):
        log_ret_jetfuel = s[i-1,:,0]
        price_jetfuel[i] = price_jetfuel[i-1]*np.exp(log_ret_jetfuel)

        log_ret_AUDinUSD = s[i-1,:,1]
        price_AUDinUSD[i] = price_AUDinUSD[i-1]*np.exp(log_ret_AUDinUSD)

        Number_of_flights[i] = Number_of_flights[i-1]*np.exp(log_ret_AUDinUSD*Flight_deviation)

        Earning[i]= ((40000 - (7000 + (price_jetfuel[i]*13)))*Number_of_flights[i]-Fixed_cost) * price_AUDinUSD[i]

    return np.array(price_jetfuel), np.array(Earning), np.array(Number_of_flights), np.array(price_AUDinUSD)
```

In [15]:
```
Jet_price_sept = data.iloc[-1].Jet_Kero_Price
AUDinUSD_sept = data.iloc[-1].AUD_USD
Flights_sept = 522
Number_of_simulations = 10000
sim_jet_price, sim_earning, sim_flights, sim_AUD_USD = Simulator(Jet_price_sept, AUDinUSD_sept, Flights_sept, Number_of_simulations)
```

In [16]:
```
months = ["Sept 2015","Oct 2015","Nov 2015","Dec 2015","Jan 2016","Feb 2016","Mar 2016","Apr 2016","May 2016","Jun 2016","Jul 2016"
        ,"Aug 2016","Sep 2016"]
sim_jet_price = pd.DataFrame(sim_jet_price,index=months).round(5)
sim_earning = pd.DataFrame(sim_earning/1e6,index=months).round(5)
sim_AUD_USD = pd.DataFrame(sim_AUD_USD,index=months).round(5)
sim_flights = pd.DataFrame(sim_flights,index=months).round(5)
```

In [17]:
```
sim_jet_price = sim_jet_price.T.rolling(10).mean().iloc[9::10].T
sim_flights = sim_flights.T.rolling(10).mean().iloc[9::10].T
sim_earning = sim_earning.T.rolling(10).mean().iloc[9::10].T
sim_AUD_USD = sim_AUD_USD.T.rolling(10).mean().iloc[9::10].T
```

In [18]:
```
print("Displaying simulated JetFuel Prices for 10 simualtions:")
display(sim_jet_price.iloc[1:,:10])
sim_jet_price[1:].T.describe().round(4)
```

Displaying simulated JetFuel Prices for 10 simualtions:

	9	19	29	39	49	59	69	79	89	99
Oct 2015	510.270793	503.930045	518.728937	481.124308	514.625942	498.895678	513.431696	494.393298	501.656721	521.285723
Nov 2015	521.307218	487.796489	517.847670	480.926182	496.923288	490.671640	522.890480	485.642082	491.042769	502.321547
Dec 2015	515.582035	483.690013	518.967537	493.047514	490.700976	478.336683	575.081001	495.294095	495.509276	505.670976
Jan 2016	513.171614	471.291431	545.084647	481.667704	505.944733	494.011118	566.159157	499.060508	527.216681	531.094041
Feb 2016	508.212230	475.394448	561.285894	495.231307	514.259294	485.158804	585.529165	510.668066	513.172811	538.256259
Mar 2016	526.639150	468.764668	567.223988	531.499747	507.808797	499.114947	577.307602	499.587675	537.273298	532.315444
Apr 2016	525.512499	452.668086	585.199721	530.029437	526.094217	504.716616	575.433259	490.860319	526.980038	527.507893
May 2016	532.178206	455.277475	586.637904	504.016200	508.379136	506.267581	599.009135	513.388018	515.921595	526.393301
Jun 2016	518.897373	465.054133	601.055807	520.774058	539.387581	514.183707	573.439522	507.030172	516.699779	526.324835
Jul 2016	514.705107	439.299711	599.560578	511.907308	532.614035	507.001429	549.422498	493.757888	507.312730	568.365577
Aug 2016	496.890335	430.684946	564.617877	520.023072	503.730110	519.371213	556.500981	498.691135	509.209635	559.116819
Sep 2016	512.817805	417.040673	578.301894	539.613197	502.498814	538.208764	565.994775	526.383904	535.947034	567.216087

Out[18]:

	Oct 2015	Nov 2015	Dec 2015	Jan 2016	Feb 2016	Mar 2016	Apr 2016	May 2016	Jun 2016	Jul 2016	Aug 2016	Sep 2016
count	1000.0000	1000.0000	1000.0000	1000.0000	1000.0000	1000.0000	1000.0000	1000.0000	1000.0000	1000.0000	1000.0000	1000.0000
mean	503.5626	505.8377	508.7222	511.1780	514.0190	515.2166	516.7622	518.8909	520.4710	523.2463	525.4453	527.3085
std	15.7067	22.1796	28.5571	33.0670	36.3429	39.9519	43.5541	47.7425	50.1869	53.1437	55.9536	59.7361
min	454.6411	446.2458	428.1701	411.5284	401.9581	374.7367	365.9355	368.8854	378.4849	380.8320	363.1678	344.4930
25%	492.6684	489.4692	488.9002	487.0847	488.6734	486.9974	486.3524	485.6018	485.0536	486.2761	487.6741	486.9581
50%	503.4859	504.7928	505.9276	509.8666	512.2143	513.0905	515.8919	516.9873	518.1388	519.3937	521.1468	521.4674
75%	513.6522	521.5656	527.1537	532.2458	537.1820	540.9092	544.1238	549.0214	549.0705	555.4308	561.5238	565.7849
max	550.7374	581.9053	609.2180	644.4029	682.2648	682.6444	662.4153	673.5608	677.7301	693.7005	742.2918	748.4062

In [19]:
```
print("Displaying simulated AUD per USD rates for 10 simualtions:")
display(sim_AUD_USD.iloc[1:,:10])
sim_AUD_USD[1:].T.describe().round(4)
```

Displaying simulated AUD per USD rates for 10 simualtions:

	9	19	29	39	49	59	69	79	89	99
Oct 2015	1.413984	1.437097	1.414129	1.421647	1.452427	1.445566	1.454903	1.424870	1.439391	1.444909
Nov 2015	1.425757	1.445522	1.384914	1.427631	1.448230	1.446692	1.471151	1.407417	1.429965	1.423357
Dec 2015	1.434601	1.467568	1.391710	1.416152	1.451153	1.437233	1.447394	1.398683	1.423456	1.438808
Jan 2016	1.432340	1.462500	1.402358	1.414029	1.453468	1.455691	1.448211	1.372093	1.409935	1.451243
Feb 2016	1.430097	1.471562	1.394892	1.396999	1.476736	1.437571	1.464442	1.376584	1.375502	1.447386
Mar 2016	1.422993	1.466416	1.415690	1.399632	1.451965	1.442577	1.463663	1.391168	1.386184	1.451159
Apr 2016	1.431333	1.462832	1.437836	1.406228	1.454772	1.443289	1.488819	1.415173	1.392612	1.447575
May 2016	1.416755	1.457017	1.431896	1.390526	1.427166	1.452464	1.486482	1.427896	1.391062	1.459229
Jun 2016	1.386290	1.445572	1.434603	1.399985	1.431612	1.450890	1.491396	1.444917	1.390789	1.440912
Jul 2016	1.376303	1.414515	1.423788	1.407855	1.412497	1.452955	1.486131	1.452220	1.393302	1.474477
Aug 2016	1.369300	1.410281	1.428666	1.407455	1.391242	1.469338	1.475372	1.439758	1.366477	1.488603
Sep 2016	1.364628	1.407061	1.442479	1.401732	1.393825	1.480432	1.480898	1.432846	1.367484	1.494533

Out[19]:

	Oct 2015	Nov 2015	Dec 2015	Jan 2016	Feb 2016	Mar 2016	Apr 2016	May 2016	Jun 2016	Jul 2016	Aug 2016	Sep 2016
count	1000.0000	1000.0000	1000.0000	1000.0000	1000.0000	1000.0000	1000.0000	1000.0000	1000.0000	1000.0000	1000.0000	1000.0000
mean	1.4360	1.4366	1.4380	1.4387	1.4390	1.4393	1.4400	1.4408	1.4411	1.4419	1.4431	1.4444
std	0.0158	0.0231	0.0283	0.0323	0.0360	0.0390	0.0423	0.0450	0.0477	0.0502	0.0536	0.0561
min	1.3841	1.3637	1.3549	1.3312	1.3190	1.3064	1.2963	1.2859	1.2630	1.2424	1.2637	1.2672
25%	1.4253	1.4215	1.4200	1.4176	1.4157	1.4143	1.4121	1.4102	1.4091	1.4092	1.4062	1.4045
50%	1.4359	1.4369	1.4374	1.4385	1.4400	1.4392	1.4394	1.4395	1.4389	1.4393	1.4428	1.4448
75%	1.4463	1.4512	1.4575	1.4599	1.4642	1.4649	1.4682	1.4716	1.4733	1.4742	1.4787	1.4810
max	1.4903	1.5166	1.5222	1.5411	1.5620	1.6008	1.5970	1.6212	1.6126	1.6209	1.6487	1.6203

In [20]:
```
print("Displaying simulated number of flights for 10 simualtions:")
display(sim_flights.iloc[1:,:10])
sim_flights[1:].T.describe().round(4)
```

Displaying simulated number of flights for 10 simualtions:

	9	19	29	39	49	59	69	79	89	99
Oct 2015	516.035218	522.748817	516.098184	518.269517	527.188808	525.241346	527.938025	519.231398	523.416015	525.033145
Nov 2015	519.478478	525.153400	507.539253	519.974262	525.831827	525.517856	532.605478	514.061856	520.540771	518.719735
Dec 2015	522.023840	531.491736	509.478139	516.624903	526.526834	522.698894	525.656693	511.414235	518.472182	523.079448
Jan 2016	521.232145	529.973859	512.577941	515.929558	527.191923	528.049197	525.913260	503.688676	514.565279	526.649116
Feb 2016	520.564133	532.573062	510.390363	510.790578	533.894026	522.712566	530.518401	504.933485	504.435483	525.409671
Mar 2016	518.533469	531.013132	516.474497	511.275677	526.744311	524.094800	530.293268	509.187757	507.492990	526.484568
Apr 2016	520.884301	529.863681	522.877341	513.152369	527.450607	524.226438	537.631976	516.180234	509.064912	525.499196
May 2016	516.470296	528.205412	520.982432	508.549279	519.266333	526.752476	536.944649	519.696885	508.670252	528.862955
Jun 2016	507.470045	524.811476	521.588565	511.318847	520.523189	526.249355	538.340288	524.583451	508.655562	523.502910
Jul 2016	504.487750	515.683335	518.442990	513.679158	514.806044	526.904553	536.638625	526.689345	509.326649	533.278984
Aug 2016	502.489285	514.234054	519.749328	513.508214	508.669521	531.418839	533.511518	523.000644	501.491655	537.356871
Sep 2016	501.252196	513.217442	523.778220	511.730828	509.342926	534.512285	535.132764	521.011332	501.670389	538.966100

Out[20]:

	Oct 2015	Nov 2015	Dec 2015	Jan 2016	Feb 2016	Mar 2016	Apr 2016	May 2016	Jun 2016	Jul 2016	Aug 2016	Sep 2016
count	1000.0000	1000.0000	1000.0000	1000.0000	1000.0000	1000.0000	1000.0000	1000.0000	1000.0000	1000.0000	1000.0000	1000.0000
mean	522.4462	522.5768	522.9170	523.0757	523.1083	523.1559	523.3074	523.4953	523.5274	523.7054	523.9979	524.3228
std	4.6002	6.7246	8.2221	9.4020	10.4694	11.3430	12.2990	13.0645	13.8479	14.5792	15.5351	16.2590
min	507.3141	501.2620	498.6318	491.6518	487.9350	484.2274	481.1765	478.0494	471.1629	464.9832	471.4018	472.4876
25%	519.3354	518.1778	517.7143	516.9564	516.3533	515.9077	515.2039	514.6764	514.1694	514.2411	513.4420	512.7461
50%	522.4185	522.6541	522.7557	523.0372	523.3842	523.1791	523.2219	523.0564	523.0199	523.1532	523.8679	524.4399
75%	525.4294	526.8212	528.5975	529.1911	530.3474	530.5540	531.5225	532.4189	532.7523	533.1549	534.4100	535.1244
max	538.1980	545.6789	547.2957	552.4380	558.6610	569.7143	568.6614	575.3932	572.9522	574.9490	582.7305	575.3032

In [21]:
```
print("Displaying simulated Earnings per month (in million AUD) for 10 simualtions:")
display(sim_earning.iloc[1:,:10])
sim_earning[1:].T.describe().round(4)
```

Displaying simulated Earnings per month (in million AUD) for 10 simualtions:

	9	19	29	39	49	59	69	79	89	99
Oct 2015	3.684424	4.080383	3.618509	4.060932	4.175986	4.245162	4.232817	3.996340	4.113071	4.009693
Nov 2015	3.741219	4.389118	3.234010	4.149306	4.282155	4.349006	4.366395	3.846449	4.123991	3.920099
Dec 2015	3.922642	4.760010	3.313819	3.869177	4.460121	4.361792	3.488360	3.644141	4.051775	4.111250
Jan 2016	3.927150	4.822639	3.238685	3.958139	4.337935	4.431531	3.589980	3.231562	3.541945	4.076890
Feb 2016	3.965562	4.908710	3.013763	3.539820	4.547875	4.267461	3.579126	3.202798	3.243862	3.996810
Mar 2016	3.668644	4.904056	3.255121	3.204239	4.251442	4.216634	3.660762	3.471601	3.160376	4.143743
Apr 2016	3.814914	5.031343	3.358933	3.295902	4.081628	4.163515	4.023209	3.910416	3.369871	4.128682
May 2016	3.554091	4.905489	3.264916	3.360584	3.933058	4.295017	3.739563	3.939015	3.557284	4.294346
Jun 2016	3.281304	4.633268	3.122934	3.286857	3.699980	4.069627	4.135604	4.213388	3.505972	3.997579
Jul 2016	3.194001	4.470337	3.121171	3.488995	3.525965	4.177712	4.318263	4.463897	3.729087	4.023891
Aug 2016	3.290578	4.564544	3.549572	3.367428	3.561411	4.238795	4.039580	4.323510	3.440489	4.284073
Sep 2016	3.035262	4.725358	3.587804	3.126140	3.654489	4.263685	3.971090	3.991841	3.379854	4.319122

Out[21]:

	Oct 2015	Nov 2015	Dec 2015	Jan 2016	Feb 2016	Mar 2016	Apr 2016	May 2016	Jun 2016	Jul 2016	Aug 2016	Sep 2016
count	1000.0000	1000.0000	1000.0000	1000.0000	1000.0000	1000.0000	1000.0000	1000.0000	1000.0000	1000.0000	1000.0000	1000.0000
mean	4.0603	4.0558	4.0541	4.0472	4.0324	4.0314	4.0339	4.0314	4.0270	4.0192	4.0213	4.0268
std	0.2313	0.3443	0.4150	0.4869	0.5488	0.5889	0.6447	0.6885	0.7279	0.7696	0.8178	0.8566
min	3.3932	3.0991	2.8291	2.3170	2.2004	2.1613	2.1222	1.9351	1.7109	1.6248	1.3114	0.7452
25%	3.9040	3.8284	3.7943	3.7238	3.6494	3.6443	3.6165	3.5667	3.5441	3.4973	3.4861	3.4443
50%	4.0562	4.0620	4.0500	4.0350	4.0114	4.0199	4.0125	4.0119	4.0006	4.0130	3.9865	3.9823
75%	4.2072	4.2852	4.2990	4.3564	4.3940	4.4266	4.4388	4.4905	4.5202	4.5103	4.5330	4.5498
max	4.9038	5.2768	5.6252	5.9523	5.9832	6.1021	6.0207	6.2651	6.3055	6.4291	6.7636	6.7947

In [22]:
```
earning_params = sim_earning[1:].sum().describe(percentiles=[0.05,.25,.5,.75,.95])
print(earning_params)
```

```
count    1000.000000
mean       48.440762
std         6.221194
min        29.851586
5%         38.354025
25%        44.325849
50%        48.187243
75%        52.371854
95%        59.094867
max        68.314811
dtype: float64
```

In [23]:
```
EaR = earning_params["50%"]-earning_params["5%"]
print("Earnings at Risk (with 95**% c**onfidence) : **{:0.2f}** (in Million AUD)".format(EaR))
```

Earnings at Risk (with 95% confidence) : 9.83 (in Million AUD)

Question 7

In [24]:
```
Oct_jetfuel = sim_jet_price.loc["Oct 2015"]
Oct_earning = sim_earning.loc["Oct 2015"]
Oct_AUD_USD = sim_AUD_USD.loc["Oct 2015"]
```

In [25]:
```
Rho_jet_earn = np.corrcoef(Oct_earning,Oct_jetfuel)[0,1]
Rho_AUD_USD_earn = np.corrcoef(Oct_earning,Oct_AUD_USD)[0,1]

print("The correlation between earnings and jet kerosene from the simulated values is **{:0.5f}**".format(Rho_jet_earn))
print("The correlation between earnings and  AUD/USD from the simulated values is **{:0.5f}**".format(Rho_AUD_USD_earn))
```

The correlation between earnings and jet kerosene from the simulated values is -0.40279
The correlation between earnings and AUD/USD from the simulated values is 0.76815

In [26]:
```
h_star_jet = Rho_jet_earn*(Oct_earning.std()/Oct_jetfuel.std())
h_star_USD_AUD = Rho_AUD_USD_earn*(Oct_earning.std()/Oct_AUD_USD.std())

print("The minimum variance hedge for jet kerosene is : **{:0.5f}**".format(h_star_jet))
print("The minimum variance hedge for currency(AUD/USD) is : **{:0.5f}**".format(h_star_USD_AUD))
```

The minimum variance hedge for jet kerosene is : -0.00593
The minimum variance hedge for currency(AUD/USD) is : 11.24017

In [27]:
```
from sklearn.linear_model import LinearRegression
lr_jet = LinearRegression()
lr_AUD_USD = LinearRegression()
linreg = LinearRegression()
```

In [28]:
```
X_jet = [[a] for a in Oct_jetfuel]
X_AUD_USD = [[a] for a in Oct_AUD_USD]
X = [[a,b] for a,b in zip(Oct_jetfuel,Oct_AUD_USD)]
Y = Oct_earning
```

In [29]:
```
lr_jet.fit(X_jet,Y)

print("The coefficient is : {:0.5f}".format(lr_jet.coef_[0]))
print("The intercept for the fit is : {:0.5f}".format(lr_jet.intercept_))
print("The R-Squared Score for the fit is : {:0.5f}".format(lr_jet.score(X_jet,Y)))
```

```
The coefficient is : -0.00593
The intercept for the fit is : 7.04758
The R-Squared Score for the fit is : 0.16224
```

In [30]:
```
lr_AUD_USD.fit(X_AUD_USD,Y)

print("The coefficient is : {:0.5f}".format(lr_AUD_USD.coef_[0]))
print("The intercept for the fit is : {:0.5f}".format(lr_AUD_USD.intercept_))
print("The R-Squared Score for the fit is : {:0.5f}".format(lr_AUD_USD.score(X_AUD_USD,Y)))
```

```
The coefficient is : 11.24017
The intercept for the fit is : -12.08078
The R-Squared Score for the fit is : 0.59005
```

The following models of the OLS regression is employed here to find the optimal/minimum hedge ratio h*, which is the slope of equation:

Earnings = intercept + h * currency rate + ε

Earnings = intercept + h * jet fuel price + ε

where, ε is the error term from OLS estimation. The coefficient obtained using this model is nearly equal to the theoretically found out minimum hedge ratio which validates the regression method.

In [31]:
```
linreg.fit(X,Y)

print("The coefficients are : {:0.5f} & {:0.5f}".format(linreg.coef_[0],linreg.coef_[1]))
print("The intercept for the fit is : {:0.5f}".format(linreg.intercept_))
print("The R-Squared Score for the fit is : {:0.5f}".format(linreg.score(X,Y)))
```

```
The coefficients are : -0.00979 & 13.91548
The intercept for the fit is : -10.99461
The R-Squared Score for the fit is : 0.99813
```

As we know that, the R-square of the OLS model indicates the hedging effectiveness, so the larger the R-square value, the more effective the hedging would be using that independent varible. So, here in our case, the hedging effectiveness is larger for the case of currency rate compared to fuel prices and thus hedging would be contributed more by the currency prices. Also, we see that optimal hedge ratio for jet fuel prices is negative. The optimal number of contracts can be found out using the optimal hedge ratio as calculated above depending upon the commodity exchange market. Hence, the optimal strategy for hedging the market risk would be shorting the positions of jet fuel contracts and going long on positions of currency contracts.