Assignment 1

October 14, 2019

1 Assignment 1

```
In [1]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        %matplotlib inline
        import scipy as sp
        import statsmodels.api as sm
        from scipy.stats import norm
        from statsmodels.tsa.arima_process import ArmaProcess
        from statsmodels.graphics.tsaplots import plot_acf
```

1.1 Question 1

```
In [2]: data = pd.read_csv('Portfolios_Formed_on_Size_daily.txt',delim_whitespace = 1)
        data.head()
```

Out[2]:

	Year	Month	Day	1-Dec	2-Dec	3-Dec	4-Dec	5-Dec	6-Dec	7-Dec	\
19260701	1926	7	1	0.58	-0.13	0.68	-0.06	-0.38	-0.07	-0.08	
19260702	1926	7	2	-0.53	-0.40	0.16	0.10	0.29	0.36	0.51	
19260706	1926	7	6	-0.33	0.61	-0.38	0.23	0.68	0.33	0.15	
19260707	1926	7	7	0.28	-0.10	-0.40	-0.54	0.31	0.17	0.19	
19260708	1926	7	8	0.55	-0.90	0.04	0.23	-0.06	0.28	0.17	

	8-Dec	9-Dec	10-Dec	vwretd	ewretd
19260701	-0.06	0.08	0.18	0.12	0.16
19260702	0.41	0.37	0.52	0.46	0.21
19260706	0.33	0.11	0.16	0.18	0.13
19260707	0.16	-0.04	0.13	0.09	-0.11
19260708	0.20	0.20	0.27	0.23	0.40

1.1.1 Q 1(a)

```
In [3]: mean_ret = data.iloc[:,3:].mean()
        print('Percentage of Mean of daily return is:\n{} %'.format(mean_ret.apply(lambda x:fo
```

```
Percentage of Mean of daily return is:
1-Dec     5.114%
```

```
2-Dec     4.898%
3-Dec     5.005%
4-Dec     4.950%
5-Dec     4.796%
6-Dec     4.887%
7-Dec     4.631%
8-Dec     4.604%
9-Dec     4.339%
10-Dec    3.973%
vwretd    4.068%
ewretd    8.280%
dtype: object %
```

```
In [4]: std_ret = data.iloc[:, 3:].std()
        print('Percentage of Stdev of daily return is:\n{} %'.format(std_ret.apply(lambda x:fo
```

```
Percentage of Stdev of daily return is:
1-Dec     128.274%
2-Dec     131.744%
3-Dec     124.357%
4-Dec     120.713%
5-Dec     119.008%
6-Dec     114.785%
7-Dec     114.686%
8-Dec     112.307%
9-Dec     111.161%
10-Dec    107.282%
vwretd    105.892%
ewretd    104.394%
dtype: object %
```

1.1.2 Q 1(b)

```
In [5]: #Calculate on the counts of business days in each interval
        #For there are 22 business days in one month
        import math
        monthly_mean = 22 * mean_ret
        print('Percentage of Mean of monthly return is:\n{} %'.format(monthly_mean.apply(lambda

        monthly_std = math.sqrt(22) * std_ret
        print('Percentage of Stdev of monthly return is:\n{} %'.format(monthly_std.apply(lambda
```

```
Percentage of Mean of monthly return is:
1-Dec     112.504%
2-Dec     107.766%
3-Dec     110.102%
4-Dec     108.909%
```

```
5-Dec     105.513%
6-Dec     107.519%
7-Dec     101.885%
8-Dec     101.291%
9-Dec      95.458%
10-Dec     87.401%
vwretd     89.486%
ewretd    182.165%
dtype: object %
Percentage of Stdev of monthly return is:
1-Dec     601.658%
2-Dec     617.936%
3-Dec     583.285%
4-Dec     566.196%
5-Dec     558.196%
6-Dec     538.390%
7-Dec     537.924%
8-Dec     526.768%
9-Dec     521.392%
10-Dec    503.199%
vwretd    496.676%
ewretd    489.651%
dtype: object %
```

```
In [6]: #For there are 252 business days in one year
        annual_mean = 252 * mean_ret
        print('Percentage of Mean of annual return is:\n{} %'.format(annual_mean.apply(lambda x

        annual_std = math.sqrt(252) * std_ret
        print('Percentage of Stdev of annual return is:\n{} %'.format(annual_std.apply(lambda x
```

```
Percentage of Mean of annual return is:
1-Dec     1288.681%
2-Dec     1234.408%
3-Dec     1261.167%
4-Dec     1247.499%
5-Dec     1208.600%
6-Dec     1231.578%
7-Dec     1167.046%
8-Dec     1160.237%
9-Dec     1093.433%
10-Dec    1001.140%
vwretd    1025.027%
ewretd    2086.617%
dtype: object %
Percentage of Stdev of annual return is:
1-Dec     2036.285%
```

```
2-Dec     2091.379%
3-Dec     1974.101%
4-Dec     1916.265%
5-Dec     1889.190%
6-Dec     1822.157%
7-Dec     1820.580%
8-Dec     1782.825%
9-Dec     1764.629%
10-Dec    1703.056%
vwretd    1680.979%
ewretd    1657.203%
dtype: object %
```

1.1.3 Q 1(c)

```
In [7]: # Compute the means using a simple regression
        data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 24391 entries, 19260701 to 20181231
Data columns (total 15 columns):
Year      24391 non-null int64
Month     24391 non-null int64
Day       24391 non-null int64
1-Dec     24391 non-null float64
2-Dec     24391 non-null float64
3-Dec     24391 non-null float64
4-Dec     24391 non-null float64
5-Dec     24391 non-null float64
6-Dec     24391 non-null float64
7-Dec     24391 non-null float64
8-Dec     24391 non-null float64
9-Dec     24391 non-null float64
10-Dec    24391 non-null float64
vwretd    24391 non-null float64
ewretd    24391 non-null float64
dtypes: float64(12), int64(3)
memory usage: 3.0 MB
```

```
In [8]: coeff = data.drop(columns=['Year','Month','Day'], axis=1)
        coeff.head()
```

```
Out[8]:           1-Dec  2-Dec  3-Dec  4-Dec  5-Dec  6-Dec  7-Dec  8-Dec  9-Dec  \
        19260701   0.58  -0.13   0.68  -0.06  -0.38  -0.07  -0.08  -0.06   0.08
        19260702  -0.53  -0.40   0.16   0.10   0.29   0.36   0.51   0.41   0.37
        19260706  -0.33   0.61  -0.38   0.23   0.68   0.33   0.15   0.33   0.11
        19260707   0.28  -0.10  -0.40  -0.54   0.31   0.17   0.19   0.16  -0.04
```

```
19260708   0.55  -0.90   0.04   0.23  -0.06   0.28   0.17   0.20   0.20
```

	10-Dec	vwretd	ewretd
19260701	0.18	0.12	0.16
19260702	0.52	0.46	0.21
19260706	0.16	0.18	0.13
19260707	0.13	0.09	-0.11
19260708	0.27	0.23	0.40

In [9]:
```
# Take i=0 as an example for summary information
model = sm.OLS(coeff.iloc[:,0], np.ones(len(coeff.index)))
res = model.fit()
res.summary()
```

```
/Users/sfdatabro/anaconda3/lib/python3.7/site-packages/statsmodels/regression/linear_model.py:1
  return self.ess/self.df_model
```

Out[9]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                  1-Dec   R-squared:                       0.000
Model:                            OLS   Adj. R-squared:                  0.000
Method:                 Least Squares   F-statistic:                       nan
Date:                Sat, 12 Oct 2019   Prob (F-statistic):                nan
Time:                        22:15:55   Log-Likelihood:                -40682.
No. Observations:               24391   AIC:                          8.137e+04
Df Residuals:                   24390   BIC:                          8.137e+04
Df Model:                           0
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0511      0.008      6.226      0.000       0.035       0.067
==============================================================================
Omnibus:                    10953.456   Durbin-Watson:                   1.647
Prob(Omnibus):                  0.000   Jarque-Bera (JB):          2060968.076
Skew:                           1.051   Prob(JB):                         0.00
Kurtosis:                      47.983   Cond. No.                         1.00
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly speci
"""
```

In [10]:
```
results = []

 # Generate an literator
```

```
        for j in np.arange(0,12):
            model = sm.OLS(coeff.iloc[:,j], np.ones(len(coeff.index)))
            res = model.fit()
            results.append(res.params[0]*100)

        # Turn the results into a dataframe
        means_ols = pd.DataFrame(columns=coeff.columns)
        means_ols.loc['Mean_OLS Percentage']= results
        means_ols
```

Out[10]:

	1-Dec	2-Dec	3-Dec	4-Dec	5-Dec
Mean_OLS Percentage	5.113812	4.898446	5.004633	4.950392	4.796031

	6-Dec	7-Dec	8-Dec	9-Dec	10-Dec
Mean_OLS Percentage	4.887212	4.631134	4.604116	4.339018	3.972777

	vwretd	ewretd
Mean_OLS Percentage	4.067566	8.280226

In [11]:
```
        # Method 2
        mean_ret_reg = pd.Series([])
        for i in data.columns[3:]:
            model = sm.OLS(data.ix[:,i], np.ones(len(data.index)))
            res = model.fit()
            mean_ret_reg = mean_ret_reg.append(res.params)

        mean_ret_reg.index = data.columns[3:]
        mean_ret_reg
```

```
/Users/sfdatabro/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:4: DeprecationWarn
.ix is deprecated. Please use
.loc for label based indexing or
.iloc for positional indexing

See the documentation here:
http://pandas.pydata.org/pandas-docs/stable/indexing.html#ix-indexer-is-deprecated
  after removing the cwd from sys.path.
```

Out[11]:

```
1-Dec     0.051138
2-Dec     0.048984
3-Dec     0.050046
4-Dec     0.049504
5-Dec     0.047960
6-Dec     0.048872
7-Dec     0.046311
8-Dec     0.046041
9-Dec     0.043390
10-Dec    0.039728
```

```
vwretd    0.040676
ewretd    0.082802
dtype: float64
```

1.1.4 Q 1(d)

```
In [12]: # Split the dataset
         mean_pre = data.loc[data['Year']<=1945,:]
         mean_post = data.loc[data['Year']>1945,:]

         mean_pre.info()
         mean_post.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5814 entries, 19260701 to 19451231
Data columns (total 15 columns):
Year      5814 non-null int64
Month     5814 non-null int64
Day       5814 non-null int64
1-Dec     5814 non-null float64
2-Dec     5814 non-null float64
3-Dec     5814 non-null float64
4-Dec     5814 non-null float64
5-Dec     5814 non-null float64
6-Dec     5814 non-null float64
7-Dec     5814 non-null float64
8-Dec     5814 non-null float64
9-Dec     5814 non-null float64
10-Dec    5814 non-null float64
vwretd    5814 non-null float64
ewretd    5814 non-null float64
dtypes: float64(12), int64(3)
memory usage: 726.8 KB
<class 'pandas.core.frame.DataFrame'>
Int64Index: 18577 entries, 19460102 to 20181231
Data columns (total 15 columns):
Year      18577 non-null int64
Month     18577 non-null int64
Day       18577 non-null int64
1-Dec     18577 non-null float64
2-Dec     18577 non-null float64
3-Dec     18577 non-null float64
4-Dec     18577 non-null float64
5-Dec     18577 non-null float64
6-Dec     18577 non-null float64
7-Dec     18577 non-null float64
8-Dec     18577 non-null float64
9-Dec     18577 non-null float64
```

```
10-Dec    18577 non-null float64
vwretd    18577 non-null float64
ewretd    18577 non-null float64
dtypes: float64(12), int64(3)
memory usage: 2.3 MB
```

```
In [13]: t_mean = pd.Series([])
         pval_mean = pd.Series([])

         for i in range(3,15):
             t,pval = sp.stats.ttest_ind(mean_pre.iloc[:,i],mean_post.iloc[:,i],equal_var = Fa
             t_mean = t_mean.append(pd.Series(t,index=[i]))
             pval_mean = pval_mean.append(pd.Series(pval, index=[i]))

         print('T statistic value:\n',t_mean)
         print('P-value:\n',pval_mean)
```

```
T statistic value:
 3     0.700360
4     0.247870
5    -0.100710
6     0.058821
7    -0.326088
8     0.050282
9    -0.486052
10   -0.471566
11   -0.612983
12   -0.679560
13   -0.580293
14    2.804149
dtype: float64
P-value:
 3     0.483728
4     0.804242
5     0.919784
6     0.953096
7     0.744367
8     0.959899
9     0.626945
10    0.637251
11    0.539907
12    0.496804
13    0.561735
14    0.005059
dtype: float64
```

1.2 Q 2

1.2.1 Q 2 (a) (b)

```
In [14]: T_AR = pd.DataFrame(columns=data.columns[3:])
         R2 = pd.DataFrame(columns=data.columns[3:])
         coeff_df = pd.DataFrame(columns=data.columns[3:])

         for i in data.columns[3:]:
             reti = data.loc[:,i]
             model = sm.OLS(reti.iloc[1:],sm.add_constant(np.array(reti.iloc[:-1])))
             res = model.fit()
             coeff_df.loc[:,i] = pd.Series(res.params)
             T_AR.loc[:,i] = pd.Series(res.tvalues)
             R2.loc[:,i] = pd.Series(res.rsquared)

         coeff_df
```

Out[14]:

	1-Dec	2-Dec	3-Dec	4-Dec	5-Dec	6-Dec	7-Dec	\
const	0.042100	0.042514	0.042044	0.042217	0.041178	0.041584	0.040143	
x1	0.176411	0.132326	0.159476	0.147371	0.141876	0.149355	0.133427	

	8-Dec	9-Dec	10-Dec	vwretd	ewretd
const	0.041046	0.039563	0.038571	0.037859	0.064993
x1	0.108692	0.088241	0.028987	0.069220	0.215102

```
In [15]: T_AR
```

Out[15]:

	1-Dec	2-Dec	3-Dec	4-Dec	5-Dec	6-Dec	\
const	5.203053	5.080738	5.344011	5.517211	5.454383	5.716560	
x1	27.988295	20.848140	25.227586	23.268308	22.382529	23.588517	

	7-Dec	8-Dec	9-Dec	10-Dec	vwretd	ewretd
const	5.511128	5.736710	5.575553	5.613208	5.592675	9.924417
x1	21.024413	17.074919	13.834190	4.528602	10.835680	34.396714

```
In [16]: R2
```

Out[16]:

	1-Dec	2-Dec	3-Dec	4-Dec	5-Dec	6-Dec	7-Dec	\
0	0.03112	0.01751	0.025432	0.021718	0.020128	0.022306	0.017802	

	8-Dec	9-Dec	10-Dec	vwretd	ewretd
0	0.011814	0.007786	0.00084	0.004791	0.046268

```
In [ ]:
```

1.2.2 Q 2(c)

```
In [ ]:
```

1.3 Q 3

1.3.1 Q 3(a)

In [17]:
```
retvw = data.loc[:,'vwretd']

const = []
x1 = []
T_predict_const = []
T_predict_x1 = []
R2_predict = []
for i in range(3,6):
    reti = data.iloc[:,i]
    model = sm.OLS(retvw[1:],sm.add_constant(np.array(reti[:-1])))
    res = model.fit()
    const.append(res.params[0])
    x1.append(res.params[1])
    T_predict_const.append(res.tvalues[0])
    T_predict_x1.append(res.tvalues[1])
    R2_predict.append(res.rsquared)

rtvw_df = pd.DataFrame(columns=['1-Dec', '2-Dec', '3-Dec'])
rtvw_df.loc['coeff_const'] = const
rtvw_df.loc['coeff_x1'] = x1
rtvw_df.loc['t_predict_const'] = T_predict_const
rtvw_df.loc['t_predict_x1'] = T_predict_x1
rtvw_df.loc['r_squared'] = R2_predict
rtvw_df
```

Out[17]:

	1-Dec	2-Dec	3-Dec
coeff_const	0.039729	0.039487	0.039140
coeff_x1	0.018462	0.024223	0.030643
t_predict_const	5.855936	5.822032	5.771310
t_predict_x1	3.493411	4.708386	5.623433
r_squared	0.000500	0.000908	0.001295

In [18]:
```
#Method2
retvw = data.loc[:,'vwretd']

coeff_predict = pd.DataFrame(columns = data.columns[3:6])
T_predict = pd.DataFrame(columns= data.columns[3:6])
R2_predict = pd.DataFrame(columns= data.columns[3:6])

for i in data.columns[3:6]:
    reti = data.loc[:,i]
    model = sm.OLS(retvw[1:],sm.add_constant(np.array(reti[:-1])))
    res = model.fit()
    coeff_predict.loc[:,i] = pd.Series(res.params)
    T_predict.loc[:,i] = pd.Series(res.tvalues)
    R2_predict.loc[:,i] = pd.Series(res.rsquared)
```

```
In [19]: coeff_predict
```

```
Out[19]:          1-Dec     2-Dec     3-Dec
         const  0.039729  0.039487  0.039140
         x1     0.018462  0.024223  0.030643
```

```
In [20]: T_predict
```

```
Out[20]:          1-Dec     2-Dec     3-Dec
         const  5.855936  5.822032  5.771310
         x1     3.493411  4.708386  5.623433
```

```
In [21]: R2_predict
```

```
Out[21]:     1-Dec     2-Dec     3-Dec
         0  0.0005  0.000908  0.001295
```

```
In [22]: #To explore deeper, we take 10-Decile to conduct one more test.
```

```
In [23]: retvw = data.loc[:,'vwretd']
         reti = data.iloc[:,9]
         model = sm.OLS(retvw[1:],sm.add_constant(np.array(reti[:-1])))
         res = model.fit()
         res.summary()
         coeff = res.params
         t_predict = res.tvalues
         r2_predict = res.rsquared
```

```
In [24]: print('coeff: ',coeff)
         print('t_predict: ', t_predict)
         print('r2_predict', r2_predict)
```

```
coeff:  const    0.038033
x1       0.057044
dtype: float64
t_predict:  const    5.615216
x1       9.666526
dtype: float64
r2_predict 0.0038168392419561936
```

1.3.2 Q 3(b)

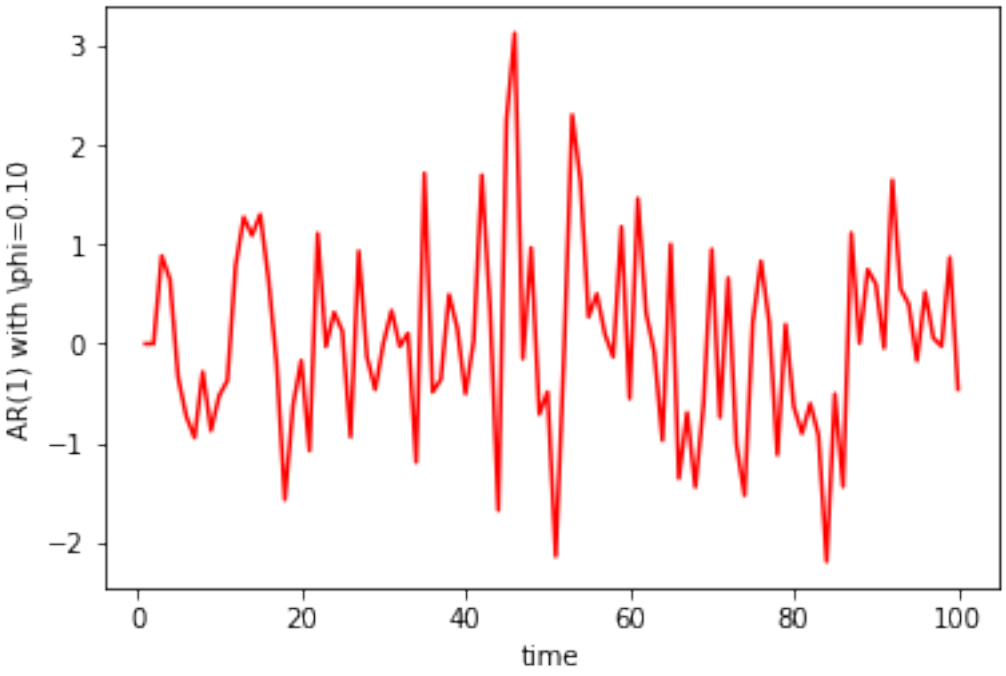
1.4 Q 4

1.4.1 Q 4 (a)(b)(c)

```
In [25]: np.random.seed(5435576)
         T = 100
         t = np.arange(1,T+1)
```

```
eps = np.random.normal(0,1,T)
Y = np.zeros((T,1))
phi=0.1
for i in np.arange(1,len(Y)-1):
    Y[i+1] = Y[i]*phi+eps[i+1]

fig = plt.figure()
plt.plot(t,Y, color='red')
plt.xlabel('time')
plt.ylabel('AR(1) with \phi=%4.2f' %(phi))
plt.show()
```

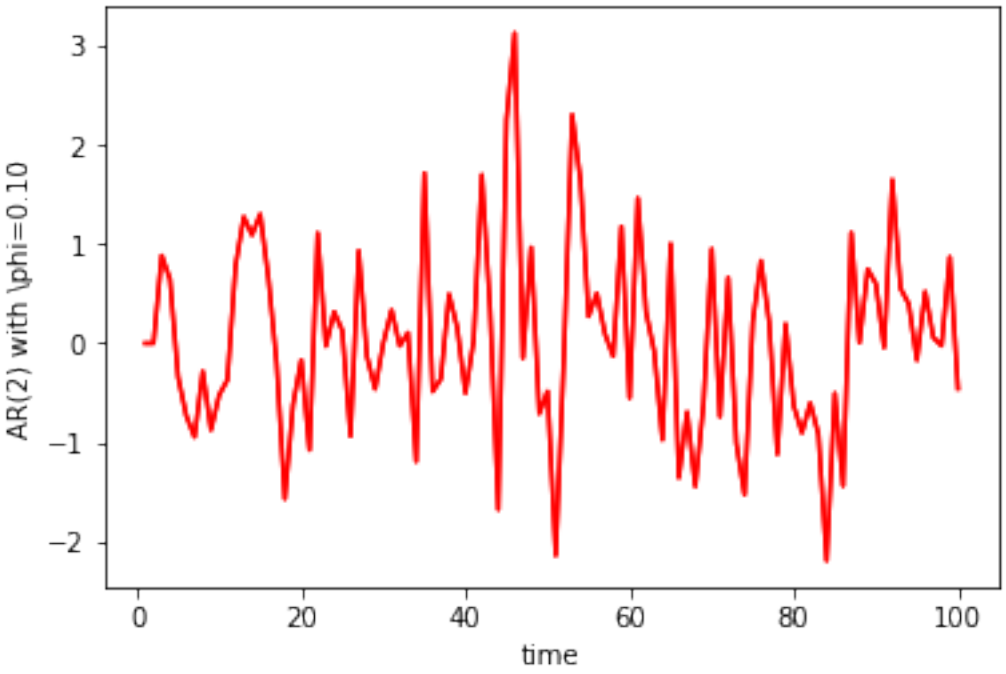


1.4.2 Q 4 (d)(e)

```
In [26]: np.random.seed(5435576)
         T = 100
         t = np.arange(1,T+1)
         eps_ar2 = np.random.randn(T,2)
         Y = np.zeros((T,2))
         phi=0.1
         for i in np.arange(1,len(Y)-1):
             Y[i+1] = Y[i]*phi+eps[i+1]

         fig = plt.figure()
```

```
plt.plot(t,Y, color='red')
plt.xlabel('time')
plt.ylabel('AR(2) with \phi=%4.2f' %(phi))
plt.show()
```



1.4.3 Q 4 (f)

```
In [27]: np.random.seed(5435576)
         T=100
         Sims=1000
         eps = np.random.randn(T,Sims)
         Y = np.zeros((T+1,Sims))
         phi=0.1
         for i in np.arange(0,len(Y)-1):
             Y[i+1,:] = Y[i,:] * phi +eps[i,:]

         Y = Y[1:,]
```

1.4.4 Q 4 (g)(h)(i)(j)(k)

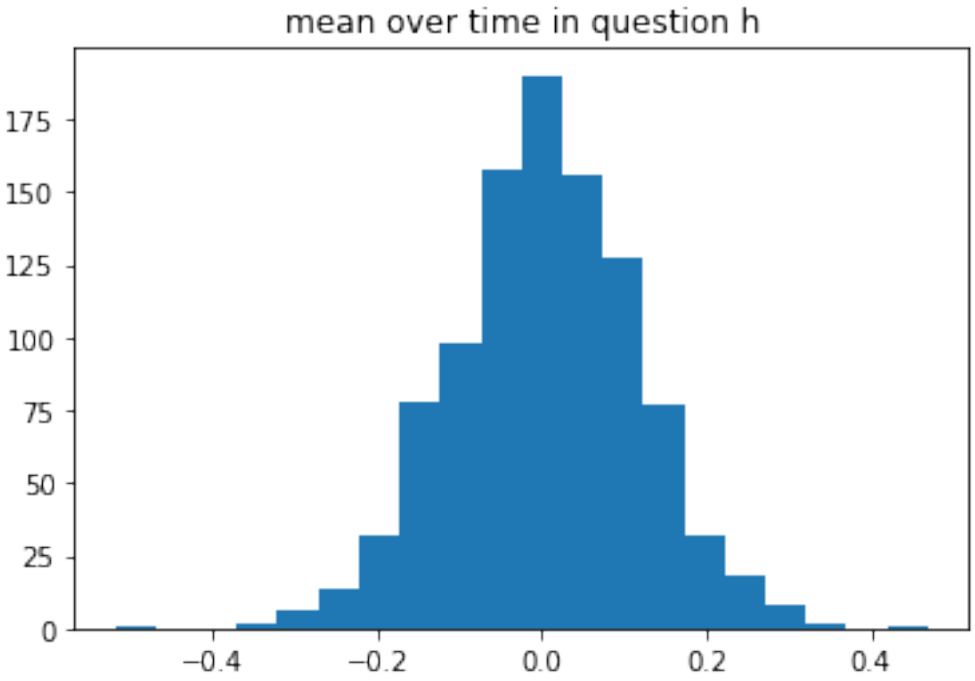
```
In [28]: # Theoretically, E(Yt) should be 0 and Var(Yt) should be 1.
```

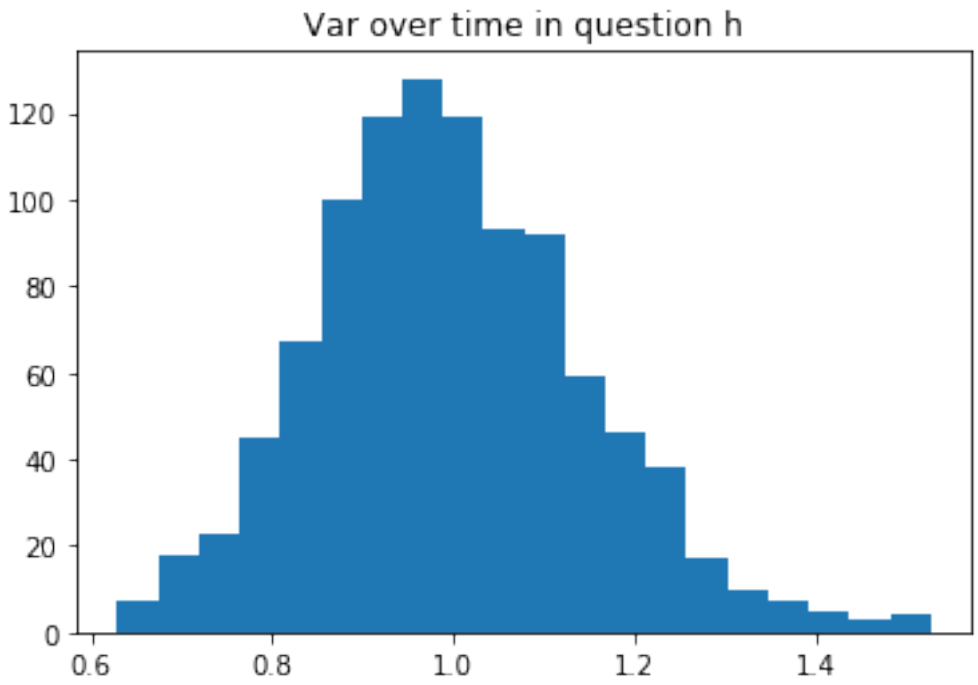
```
In [48]: mu_ts = np.mean(Y,0)
         var_ts = np.var(Y,0)
```

```
fig = plt.figure()
plt.hist(mu_ts,bins=20)
plt.title('mean over time in question h')

fig = plt.figure()
plt.hist(var_ts,bins=20)
plt.title('Var over time in question h')
```

Out[48]: Text(0.5, 1.0, 'Var over time in question h')



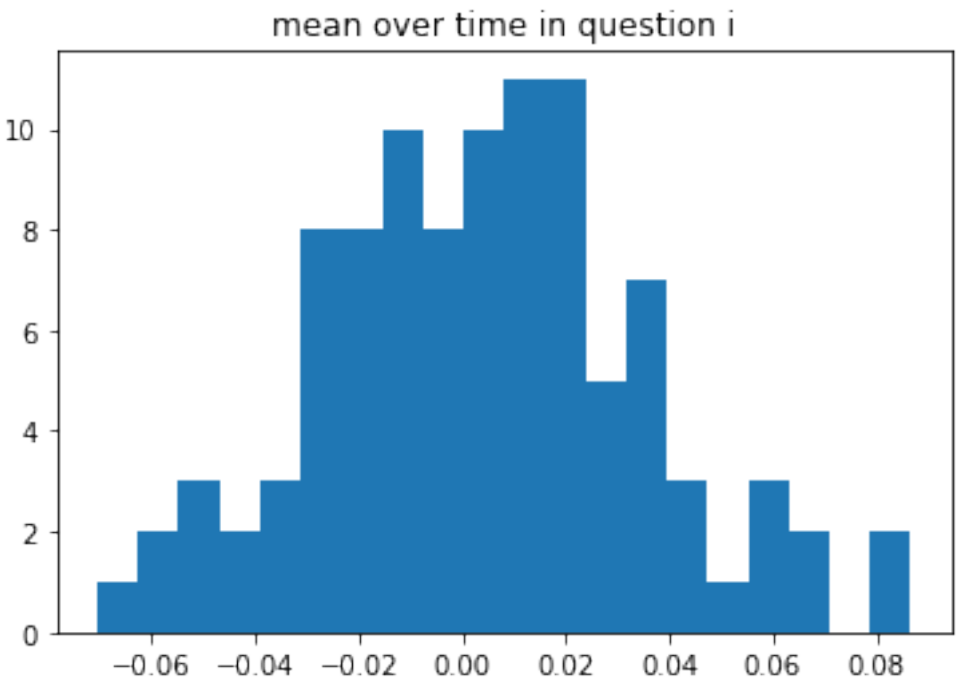


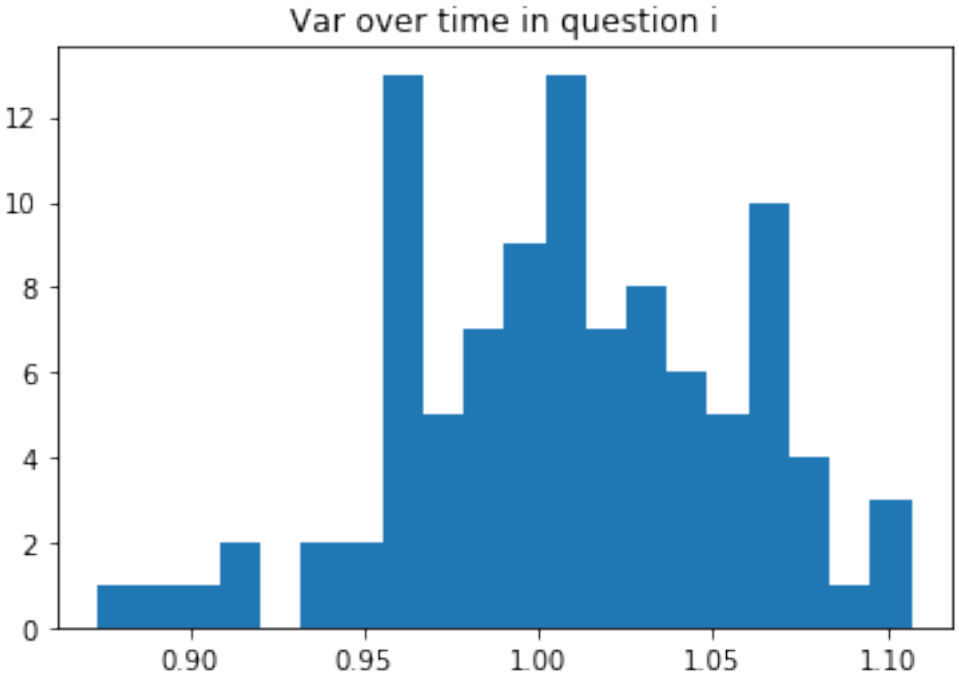

```
In [49]: mu_cs = np.mean(Y,1)
         var_cs = np.var(Y,1)

         fig = plt.figure()
         plt.hist(mu_cs,bins=20)
         plt.title('mean over time in question i')

         fig = plt.figure()
         plt.hist(var_cs,bins=20)
         plt.title('Var over time in question i')

Out[49]: Text(0.5, 1.0, 'Var over time in question i')
```

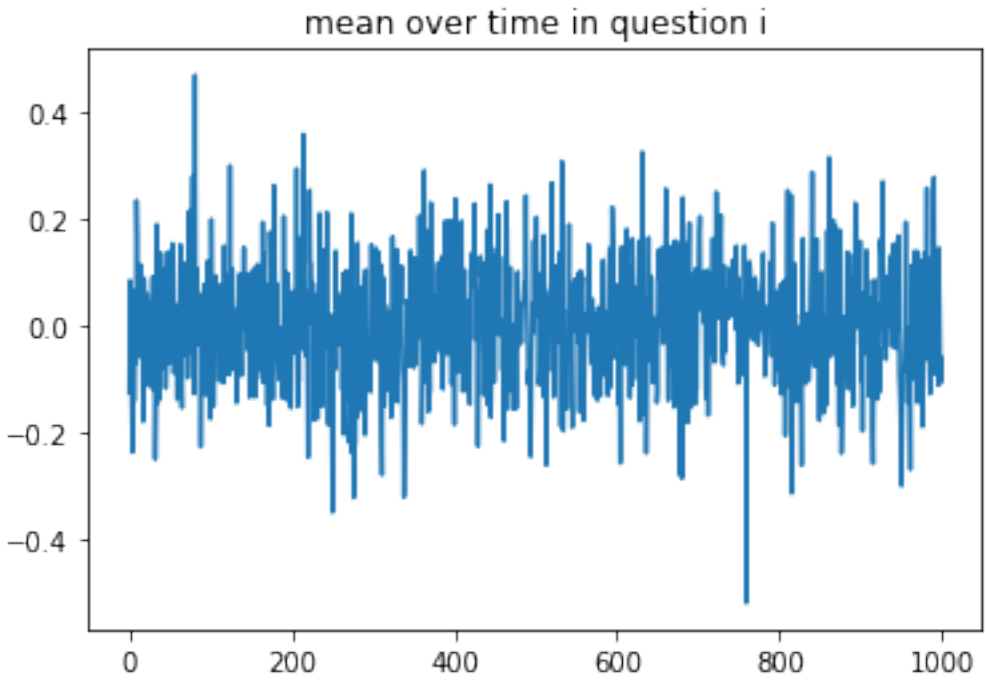

mean over time in question i
0
2
4
6
8
10
−0.06
−0.04
−0.02
0.00
0.02
0.04
0.06
0.08


Var over time in question i
0
2
4
6
8
10
12
0.90
0.95
1.00
1.05
1.10

1.4.5 Q 4 (l)

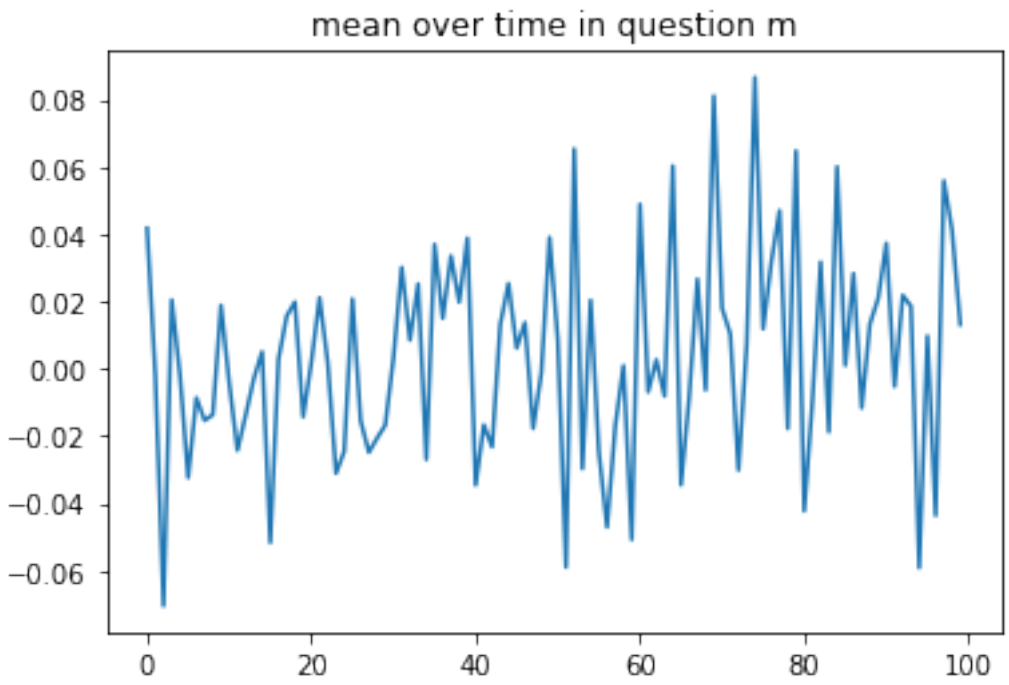
```
In [50]: fig = plt.figure()
         plt.plot(mu_ts)
         plt.title('mean over time in question i')

Out[50]: Text(0.5, 1.0, 'mean over time in question i')
```




```
In [51]: fig = plt.figure()
         plt.plot(mu_cs)
         plt.title('mean over time in question m')

Out[51]: Text(0.5, 1.0, 'mean over time in question m')
```



In [ ]:

1.5 Q 5

In [31]:
```
# Downloaded the most recent data from 1926-07-01 to 2019-08-30
data = pd.read_csv('Portfolios_Formed_on_ME_daily.csv',header=0)
data.columns = ['Date', '1-Dec', '2-Dec', '3-Dec', '4-Dec', '5-Dec', '6-Dec',
       '7-Dec', '8-Dec', '9-Dec', '10-Dec']
data.head()
```

Out[31]:

	Date	1-Dec	2-Dec	3-Dec	4-Dec	5-Dec	6-Dec	7-Dec	8-Dec	9-Dec	\
0	19260701	0.58	-0.13	0.68	-0.06	-0.38	-0.07	-0.08	-0.06	0.08	
1	19260702	-0.53	-0.40	0.16	0.10	0.29	0.36	0.51	0.41	0.37	
2	19260706	-0.33	0.61	-0.38	0.23	0.68	0.33	0.15	0.33	0.11	
3	19260707	0.28	-0.10	-0.40	-0.54	0.31	0.17	0.19	0.16	-0.04	
4	19260708	0.55	-0.90	0.04	0.23	-0.06	0.28	0.17	0.20	0.20	

	10-Dec
0	0.18
1	0.52
2	0.16
3	0.13
4	0.27

In [32]:
```
#Compute the daily mean
mean_ret = data.iloc[:,1:].mean()
```

```
        print('Percentage of Mean of daily return is:\n{} %'.format(mean_ret.apply(lambda x:f
        std_ret = data.iloc[:,1:].std()
        print('Percentage of Stdev of daily return is:\n{} %'.format(std_ret.apply(lambda x:f
```

```
Percentage of Mean of daily return is:
1-Dec     5.093%
2-Dec     4.890%
3-Dec     5.005%
4-Dec     4.949%
5-Dec     4.810%
6-Dec     4.930%
7-Dec     4.673%
8-Dec     4.640%
9-Dec     4.386%
10-Dec    4.014%
dtype: object %
Percentage of Stdev of daily return is:
1-Dec     128.118%
2-Dec     131.642%
3-Dec     124.279%
4-Dec     120.686%
5-Dec     118.988%
6-Dec     114.760%
7-Dec     114.634%
8-Dec     112.191%
9-Dec     111.031%
10-Dec    107.170%
dtype: object %
```

```
In [33]: # Compute the monthly return
         monthly_mean = 22 * mean_ret
         print('Percentage of Mean of monthly return is:\n{} %'.format(monthly_mean.apply(lambd

         monthly_std = math.sqrt(22) * std_ret
         print('Percentage of Stdev of monthly return is:\n{} %'.format(monthly_std.apply(lambd

         # Compute the annual return
         annual_mean = 252 * mean_ret
         print('Percentage of Mean of annual return is:\n{} %'.format(annual_mean.apply(lambda

         annual_std = math.sqrt(252) * std_ret
         print('Percentage of Stdev of annual return is:\n{} %'.format(annual_std.apply(lambda
```

```
Percentage of Mean of monthly return is:
1-Dec     112.050%
2-Dec     107.571%
3-Dec     110.106%
```

```
4-Dec     108.879%
5-Dec     105.809%
6-Dec     108.454%
7-Dec     102.814%
8-Dec     102.078%
9-Dec      96.481%
10-Dec     88.306%
dtype: object %
Percentage of Stdev of monthly return is:
1-Dec     600.925%
2-Dec     617.455%
3-Dec     582.918%
4-Dec     566.067%
5-Dec     558.104%
6-Dec     538.273%
7-Dec     537.682%
8-Dec     526.223%
9-Dec     520.781%
10-Dec    502.672%
dtype: object %
Percentage of Mean of annual return is:
1-Dec     1283.487%
2-Dec     1232.172%
3-Dec     1261.211%
4-Dec     1247.163%
5-Dec     1211.999%
6-Dec     1242.290%
7-Dec     1177.686%
8-Dec     1169.262%
9-Dec     1105.151%
10-Dec    1011.509%
dtype: object %
Percentage of Stdev of annual return is:
1-Dec     2033.804%
2-Dec     2089.750%
3-Dec     1972.861%
4-Dec     1915.829%
5-Dec     1888.880%
6-Dec     1821.761%
7-Dec     1819.760%
8-Dec     1780.979%
9-Dec     1762.561%
10-Dec    1701.271%
dtype: object %
```

```
In [34]: results = []
```

```
        # Generate an literator
        for j in np.arange(1,11):
            model = sm.OLS(data.iloc[:,j], np.ones(len(data.index)))
            res = model.fit()
            results.append(res.params[0]*100)

        # Turn the results into a dataframe
        means_ols = pd.DataFrame(columns=data.columns[1:])
        means_ols.loc['Mean_OLS Percentage']= results
        means_ols
```

Out[34]:

	1-Dec	2-Dec	3-Dec	4-Dec	5-Dec	6-Dec	7-Dec	8-Dec	9-Dec	10-Dec
Mean_OLS Percentage	5.093204	4.889572	5.004805	4.949061	4.80952	4.92972	4.673358	4.639928	4.385521	4.013926

```
In [35]: year_l = []
         for i in data['Date']:
             i = str(i)
             year_l.append(i[:4])

         data['Year'] = year_l
```

```
In [36]: data.head()
```

Out[36]:

	Date	1-Dec	2-Dec	3-Dec	4-Dec	5-Dec	6-Dec	7-Dec	8-Dec	9-Dec	10-Dec	Year
0	19260701	0.58	-0.13	0.68	-0.06	-0.38	-0.07	-0.08	-0.06	0.08	0.18	1926
1	19260702	-0.53	-0.40	0.16	0.10	0.29	0.36	0.51	0.41	0.37	0.52	1926
2	19260706	-0.33	0.61	-0.38	0.23	0.68	0.33	0.15	0.33	0.11	0.16	1926
3	19260707	0.28	-0.10	-0.40	-0.54	0.31	0.17	0.19	0.16	-0.04	0.13	1926
4	19260708	0.55	-0.90	0.04	0.23	-0.06	0.28	0.17	0.20	0.20	0.27	1926

```
In [41]: pd.to_numeric(data['Year'])
```

```
Out[41]: 0        1926
         1        1926
         2        1926
         3        1926
         4        1926
         5        1926
         6        1926
```

```
7       1926
8       1926
9       1926
10      1926
11      1926
12      1926
13      1926
14      1926
15      1926
16      1926
17      1926
18      1926
19      1926
20      1926
21      1926
22      1926
23      1926
24      1926
25      1926
26      1926
27      1926
28      1926
29      1926
        ...
24529   2019
24530   2019
24531   2019
24532   2019
24533   2019
24534   2019
24535   2019
24536   2019
24537   2019
24538   2019
24539   2019
24540   2019
24541   2019
24542   2019
24543   2019
24544   2019
24545   2019
24546   2019
24547   2019
24548   2019
24549   2019
24550   2019
24551   2019
24552   2019
```

```
24553    2019
24554    2019
24555    2019
24556    2019
24557    2019
24558    2019
Name: Year, Length: 24559, dtype: int64
```

```
In [43]: # Split the dataset
         mean_pre = data.loc[data['Year']<='1945',:]
         mean_post = data.loc[data['Year']>'1945',:]

         mean_pre.info()
         mean_post.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5814 entries, 0 to 5813
Data columns (total 12 columns):
Date      5814 non-null int64
1-Dec     5814 non-null float64
2-Dec     5814 non-null float64
3-Dec     5814 non-null float64
4-Dec     5814 non-null float64
5-Dec     5814 non-null float64
6-Dec     5814 non-null float64
7-Dec     5814 non-null float64
8-Dec     5814 non-null float64
9-Dec     5814 non-null float64
10-Dec    5814 non-null float64
Year      5814 non-null object
dtypes: float64(10), int64(1), object(1)
memory usage: 590.5+ KB
<class 'pandas.core.frame.DataFrame'>
Int64Index: 18745 entries, 5814 to 24558
Data columns (total 12 columns):
Date      18745 non-null int64
1-Dec     18745 non-null float64
2-Dec     18745 non-null float64
3-Dec     18745 non-null float64
4-Dec     18745 non-null float64
5-Dec     18745 non-null float64
6-Dec     18745 non-null float64
7-Dec     18745 non-null float64
8-Dec     18745 non-null float64
9-Dec     18745 non-null float64
10-Dec    18745 non-null float64
Year      18745 non-null object
dtypes: float64(10), int64(1), object(1)
```

```
memory usage: 1.9+ MB
```

```
In [44]: t_mean = pd.Series([])
         pval_mean = pd.Series([])

         for i in range(1,11):
             t,pval = sp.stats.ttest_ind(mean_pre.iloc[:,i],mean_post.iloc[:,i],equal_var = Fa
             t_mean = t_mean.append(pd.Series(t,index=[i]))
             pval_mean = pval_mean.append(pd.Series(pval, index=[i]))

         print('T statistic value:\n',t_mean)
         print('P-value:\n',pval_mean)
```

```
T statistic value:
 1     0.708559
2     0.251793
3    -0.100625
4     0.059465
5    -0.333314
6     0.025017
7    -0.510227
8    -0.492878
9    -0.640578
10   -0.706171
dtype: float64
P-value:
 1     0.478624
2     0.801209
3     0.919851
4     0.952583
5     0.738907
6     0.980042
7     0.609908
8     0.622114
9     0.521817
10    0.480103
dtype: float64
```

```
In [45]: T_AR = pd.DataFrame(columns=data.columns[1:11])
         R2 = pd.DataFrame(columns=data.columns[1:11])
         coeff_df = pd.DataFrame(columns=data.columns[1:11])

         for i in data.columns[1:11]:
             reti = data.loc[:,i]
             model = sm.OLS(reti.iloc[1:],sm.add_constant(np.array(reti.iloc[:-1])))
             res = model.fit()
```

```
        coeff_df.loc[:,i] = pd.Series(res.params)
        T_AR.loc[:,i] = pd.Series(res.tvalues)
        R2.loc[:,i] = pd.Series(res.rsquared)

    coeff_df
```

Out[45]:

	1-Dec	2-Dec	3-Dec	4-Dec	5-Dec	6-Dec	7-Dec	8-Dec	9-Dec	10-Dec
const	0.041967	0.042476	0.042101	0.042270	0.041350	0.041999	0.040550	0.041391	0.040014	0.038998
x1	0.175608	0.131397	0.158247	0.145953	0.140597	0.148155	0.132432	0.108057	0.087539	0.028284

In [46]:
```
T_AR
```

Out[46]:

	1-Dec	2-Dec	3-Dec	4-Dec	5-Dec	6-Dec	7-Dec	8-Dec	9-Dec	10-Dec
const	5.209997	5.096951	5.372001	5.543253	5.495847	5.793580	5.587762	5.810397	5.664740	5.700602
x1	27.952884	20.770403	25.114424	23.118822	22.253086	23.475427	20.936915	17.032632	13.770466	4.433925

In [47]:
```
R2
```

Out[47]:

	1-Dec	2-Dec	3-Dec	4-Dec	5-Dec	6-Dec	7-Dec	8-Dec	9-Dec	10-Dec
0	0.030838	0.017265	0.025042	0.021302	0.019768	0.02195	0.017538	0.011676	0.007663	0.0008

In [ ]:
```
```