ASSIGNMENT 1

1. Create a matrix 3x4 matrix named x that looks like this:

100	1234	1245	1252
1	2	3	4
1	3	5	6

```
x = [100 1234 1245 1252; 1 2 3 4; 1 3 5 6]
```

```
x = 3×4
        100        1234        1245        1252
          1           2           3           4
          1           3           5           6
```

2. Run the below code to create the matrix called x.

Create a 2x2 **table** called `xtable` (all lowercase) that stores only values in the first two rows and first two columns of `X1.`

- Use column names of 'dollar' for column 1 and 'yuan' for column 2.
- Use row names of 'gold' and 'silver'

Row and column names are case sensitive, so use all lowercase letters for both

```
x = [ 1 6 3 1; 7 3 6 0; 1 9 4 1]
```

```
x = 3×4
     1     6     3     1
     7     3     6     0
     1     9     4     1
```

```
x1 = x(1:2, 1:2)
```

```
x1 = 2×2
     1     6
     7     3
```

```
xtable = array2table(x1)
```

xtable = 2×2 table

	x11	x12
1	1	6
2	7	3

```
xtable.Properties.VariableNames = {'dollar','yuan'}
```

xtable = 2×2 table

	dollar	yuan
1	1	6
2	7	3

```
xtable.Properties.RowNames = {'gold','silver'}
```

xtable = 2×2 table

	dollar	yuan
1 gold	1	6
2 silver	7	3

```
xtable
```

xtable = 2×2 table

	dollar	yuan
1 gold	1	6
2 silver	7	3

3. First, create a matrix called `B` with the following form:

1 2 3 4 5

6 7 8 9 10

11 12 13 14 15

Next, define the following three variables:

- A 3x1 column vector called `B1` that stores only the second column of `B`.
- A 1x5 row vector called `B2` that stores only the third row of `B`.
- A 2x3 matrix called `B3` that stores the intersection of columns 2, 3, 4 with rows 1 and 3 (i.e. the values 2, 3, 4, 12, 13 ,14).

```
B = [1 2 3 4 5; 6 7 8 9 10;11 12 13 14 15]
```

```
B = 3×5
     1     2     3     4     5
     6     7     8     9    10
    11    12    13    14    15
```

```
B1 = B(:,2)
```

```
B1 = 3×1
     2
     7
    12
```

```
B2 = B(3,:)
```

```
B2 = 1×5
    11    12    13    14    15
```

```
B3 = B([1,3],[2,3,4])
```

```
B3 = 2×3
     2     3     4
    12    13    14
```

4. Run the `readtable` statement below to view a set of monthly portfolio returns stored in the data table called `portfolio_returns`.

The arithmetic average return of the set $\{r_1, r_2, \ldots, r_T\}$ is the simple mean $\bar{r} = \frac{1}{T}\sum_{t=1}^{T} r_t$

The geometric average return of the set $\{r_1, r_2, \ldots, r_T\}$ is calculated as $\bar{r} = \left(\prod_{t=1}^{T} 1 + r_t\right)^{1/T} - 1$, which represents the average growth rate.

Use Matlab to calculate the arithmetic and geometric mean monthly returns. **You cannot use the functions mean or geomean to answer this question.**

Assign the arithmetic average to a variable named `meanA`.

Assign the geometric average to a variable named `meanG`.

```
portfolio_returns = readtable('means_calc.xlsx')
```

portfolio_returns = 12×2 table

	Month	Portfolio_Rets
1	'Jan'	0.1958
2	'Feb'	0.3109
3	'Mar'	0.0495
4	'Apr'	0.2241
5	'May'	0.2524
6	'Jun'	0.1894
7	'Jul'	-0.1691
8	'Aug'	0.0479
9	'Sep'	-0.2802
10	'Oct'	0.0384
11	'Nov'	-0.0430
12	'Dec'	0.0958

```
returns = portfolio_returns.Portfolio_Rets
```

```
returns = 12×1
    0.1958
    0.3109
    0.0495
    0.2241
    0.2524
    0.1894
   -0.1691
    0.0479
   -0.2802
    0.0384
```

⋮

```
n = length(returns)
```

```
n =
12
```

```
meanA = sum(returns)/n
```

```
meanA =
0.0760
```

```
product_returns = prod(1 + returns)
```

```
product_returns =
2.0470
```

```
meanG = product_returns^(1/n) - 1
```

```
meanG =
0.0615
```

5. Run the `readtable` command below to see monthly returns on an active portfolio and its benchmark.

Let r_{pt} and r_{bt} be the portfolio and benchmark returns at time *t*, respectively. The average active return on the portfolio over the interval $[0, T]$ is calculated as $\overline{r}_a = (\overline{r}_p - \overline{r}_b)$ where $\overline{r}$ denotes the geometric mean. Note that this difference between geometric means is not the same as the geometric mean of the differences:

$$\overline{r}_a \neq \left(\prod_{t=1}^{T} r_{bt} - r_{pt}\right)^{1/T} - 1$$

Use Matlab to calculate the average active return as the geometric mean return on the portfolio less the geometric mean return on the benchmark. **Your solution cannot use the geomean function.**

- Assign the value of the geometric mean portfolio return to `mean_Rp`.
- Assign the value of the geometric mean benchmark return to `mean_Rb`.
- Assign the value of the geometric mean active return to `mean_Ra`.

```
ret_table = readtable('RpRb_1.xlsx')
```

ret_table = 12×3 table

	Month	Portfolio_Rets	Benchmark_Rets
1	'Jan'	0.1958	0.1945
2	'Feb'	0.3109	0.2792
3	'Mar'	0.0495	0.0710
4	'Apr'	0.2241	0.2380
5	'May'	0.2524	0.2854

	Month	Portfolio_Rets	Benchmark_Rets
6	'Jun'	0.1894	0.1953
7	'Jul'	-0.1691	-0.1550
8	'Aug'	0.0479	0.0248
9	'Sep'	-0.2802	-0.2431
10	'Oct'	0.0384	0.0858
11	'Nov'	-0.0430	-0.0506
12	'Dec'	0.0958	0.0819

```
Rp = ret_table.Portfolio_Rets
```

```
Rp = 12×1
    0.1958
    0.3109
    0.0495
    0.2241
    0.2524
    0.1894
   -0.1691
    0.0479
   -0.2802
    0.0384
      ⋮
```

```
Rb = ret_table.Benchmark_Rets
```

```
Rb = 12×1
    0.1945
    0.2792
    0.0710
    0.2380
    0.2854
    0.1953
   -0.1550
    0.0248
   -0.2431
    0.0858
      ⋮
```

```
mean_Rp = prod(1+ Rp)^(1/length(Rp))-1
```

```
mean_Rp =
0.0615
```

```
annualized_Rp = (1+mean_Rp)^(12)-1
```

```
annualized_Rp =
1.0470
```

```
mean_Rb = prod(1+ Rb)^(1/length(Rb))-1
```

```
mean_Rb =
0.0709
```

```
annualized_Rb = (1+mean_Rb)^(12)-1
```

```
annualized_Rb =
1.2755
```

```
mean_Ra = mean_Rp - mean_Rb
```

```
mean_Ra =
-0.0094
```

6. The tracking error of a portfolio return is the standard deviation of the active return. If r_{pt} and r_{bt} are the portfolio return at time *t*, respectively, then $r_{at} = r_{pt} - r_{bt}$ is the active return at time *t*, and the tracking error of the portfolio is calculated as:

$TE_p = \sqrt{\frac{1}{T-1}\sum_{t=1}^{T}(r_{at} - \bar{r}_a)^2}$ where $\bar{r}_a = \frac{1}{T}\sum_{t=1}^{T} r_{at}$ is the arithmetic mean active return.

The information ratio of the portfolio is the ratio of the average active return to its tracking error:

$IR_p = \frac{\text{average active return}}{TE_p}$, where the average active return is calculated as the difference between the geometric mean portfolio return and the geometric mean benchmark return

Run the `readtable` command below to see a set of monthly returns on an active portfolio and its benchmark stored in the table `rets`.

1. Calculate the average active return and assign it to the variable `portfolio_Ra`.
2. Calculate the tracking error associated with the active portfolio and assign it to the variable `portfolio_TE`.
3. Calculate the information ratio associated with the active portfolio and assign it to the variable `portfolio_IR`.

```
rets = readtable('RpRb_1.xlsx')
```

rets = 12×3 table

	Month	Portfolio_Rets	Benchmark_Rets
1	'Jan'	0.1958	0.1945
2	'Feb'	0.3109	0.2792
3	'Mar'	0.0495	0.0710
4	'Apr'	0.2241	0.2380
5	'May'	0.2524	0.2854
6	'Jun'	0.1894	0.1953
7	'Jul'	-0.1691	-0.1550
8	'Aug'	0.0479	0.0248
9	'Sep'	-0.2802	-0.2431

	Month	Portfolio_Rets	Benchmark_Rets
10	'Oct'	0.0384	0.0858
11	'Nov'	-0.0430	-0.0506
12	'Dec'	0.0958	0.0819

```
T              = size(rets,1);
Rp             = prod(1+rets{:,2})^(1/T)-1
```

```
Rp =
0.0615
```

```
Rb             = prod(1+rets{:,3})^(1/T)-1
```

```
Rb =
0.0709
```

```
portfolio_Ra = Rp - Rb
```

```
portfolio_Ra =
-0.0094
```

```
active_rets     = rets{:,2}-rets{:,3}
```

```
active_rets = 12×1
    0.0013
    0.0317
   -0.0215
   -0.0139
   -0.0330
   -0.0059
   -0.0141
    0.0230
   -0.0372
   -0.0474
      ⋮
```

```
avg_active_ret  = sum(active_rets)/T
```

```
avg_active_ret =
-0.0080
```

```
sqrd_deviations = (active_rets - avg_active_ret).^2
```

```
sqrd_deviations = 12×1
    0.0001
    0.0016
    0.0002
    0.0000
    0.0006
    0.0000
    0.0000
    0.0010
    0.0009
    0.0016
```

⋮

```
portfolio_TE    = sqrt(sum(sqrd_deviations)/(T-1))
```

```
portfolio_TE =
0.0246
```

```
% An easier way to calculate TE is to use the standard deviation function.
portfolio_TE   = std(active_rets)
```

```
portfolio_TE =
0.0246
```

```
portfolio_IR = portfolio_Ra/portfolio_TE
```

```
portfolio_IR =
-0.3827
```

7. Run the below `readtable` command to create a table that stores monthly total returns on Oracle, a benchmark portfolio (CRSP-VW index), and the risk-free rate. The risk-free rate has already been deannualized to a monthly basis.

Use this data to estimate the linear regression $r_{it} - r_{ft} = \widehat{\alpha}_i + \widehat{\beta}_i(r_{bt} - r_{ft}) + e_{it}$ to calculate Oracle's alpha and beta.

- Store the estimate of beta to the scalar `ORCL_beta`.
- Store the estimate of alpha to the scalar `ORCL_alpha`.
- Store the t-statistic for Oracle's alpha as the scalar `alpha_tstat`.

```
data = readtable('CAPM_table.csv')
```

data = 60×4 table

	DATE	ORCL_RET	MKT_RET	RF
1	20160129	-0.0019	-0.0576	1.0000e-04
2	20160229	0.0129	-6.0000e-04	2.0000e-04
3	20160331	0.1123	0.0698	2.0000e-04
4	20160429	-0.0220	0.0093	1.0000e-04
5	20160531	0.0085	0.0179	1.0000e-04
6	20160630	0.0182	-3.0000e-04	2.0000e-04
7	20160729	0.0064	0.0397	2.0000e-04
8	20160831	0.0044	0.0052	2.0000e-04
9	20160930	-0.0471	0.0027	2.0000e-04
10	20161031	-0.0181	-0.0200	2.0000e-04
11	20161130	0.0461	0.0487	1.0000e-04

	DATE	ORCL_RET	MKT_RET	RF
12	20161230	-0.0433	0.0185	3.0000e-04
13	20170131	0.0471	0.0198	4.0000e-04
14	20170228	0.0618	0.0361	4.0000e-04
15	20170331	0.0474	0.0020	3.0000e-04
16	20170428	0.0121	0.0114	5.0000e-04
17	20170531	0.0096	0.0112	6.0000e-04
18	20170630	0.1046	0.0084	6.0000e-04
19	20170731	-3.9900e-04	0.0194	7.0000e-04
20	20170831	0.0080	0.0025	9.0000e-04
21	20170929	-0.0393	0.0260	9.0000e-04
22	20171031	0.0567	0.0234	9.0000e-04
23	20171130	-0.0361	0.0320	8.0000e-04
24	20171229	-0.0363	0.0115	9.0000e-04
25	20180131	0.0952	0.0569	0.0011
26	20180228	-0.0178	-0.0354	0.0011
27	20180329	-0.0971	-0.0223	0.0012
28	20180430	0.0024	0.0043	0.0014
29	20180531	0.0230	0.0279	0.0014
30	20180629	-0.0569	0.0062	0.0014
31	20180731	0.0865	0.0335	0.0016
32	20180831	0.0189	0.0360	0.0016
33	20180928	0.0613	0.0021	0.0015
34	20181031	-0.0491	-0.0749	0.0019
35	20181130	-0.0016	0.0187	0.0018
36	20181231	-0.0740	-0.0936	0.0019
37	20190131	0.1167	0.0862	0.0021
38	20190228	0.0378	0.0358	0.0018
39	20190329	0.0303	0.0129	0.0019
40	20190430	0.0346	0.0417	0.0021
41	20190531	-0.0855	-0.0673	0.0021
42	20190628	0.1259	0.0711	0.0018
43	20190731	-0.0075	0.0138	0.0019
44	20190830	-0.0753	-0.0242	0.0016

	DATE	ORCL_RET	MKT_RET	RF
45	20190930	0.0570	0.0161	0.0018
46	20191031	-0.0055	0.0221	0.0015
47	20191129	0.0303	0.0399	0.0012
48	20191231	-0.0563	0.0291	0.0014
49	20200131	-0.0055	2.0000e-04	0.0013
50	20200228	-0.0570	-0.0801	0.0012
51	20200331	-0.0228	-0.1326	0.0012
52	20200430	0.1010	0.1365	0
53	20200529	0.0151	0.0559	1.0000e-04
54	20200630	0.0279	0.0247	1.0000e-04
55	20200731	0.0076	0.0578	1.0000e-04
56	20200831	0.0319	0.0764	1.0000e-04
57	20200930	0.0433	-0.0362	1.0000e-04
58	20201030	-0.0561	-0.0209	1.0000e-04
59	20201130	0.0287	0.1248	1.0000e-04
60	20201231	0.1208	0.0464	1.0000e-04

```
ORCL_Ret = data.ORCL_RET; % Oracle monthly returns
MKT_RET = data.MKT_RET;   % Benchmark monthly returns
RF = data.RF;             % Risk-free rate
ORCL_Excess = ORCL_Ret - RF;    % Excess return for Oracle
Benchmark_Excess = MKT_RET - RF; % Excess return for Benchmark

% Perform linear regression
lm = fitlm(Benchmark_Excess, ORCL_Excess, 'Intercept', true);

% Extract alpha, beta, and alpha's t-statistic
ORCL_alpha = lm.Coefficients.Estimate(1);       % Intercept (alpha)
ORCL_beta = lm.Coefficients.Estimate(2);        % Slope (beta)
alpha_tstat = lm.Coefficients.tStat(1);         % t-stat for alpha

% Display results
disp(['Oracle Alpha: ', num2str(ORCL_alpha)]);
```

```
Oracle Alpha: 0.0024086
```

```
disp(['Oracle Beta: ', num2str(ORCL_beta)]);
```

```
Oracle Beta: 0.70066
```

```
disp(['Alpha t-statistic: ', num2str(alpha_tstat)]);
```

```
Alpha t-statistic: 0.41867
```