ML_1_homework_q1_submission

October 1, 2024

1 Problem 1 of ML1 homework | Problem 2.9 from ACT book

[ ]:

```

```

2 Imports

[2]:

```
import pandas as pd
import statsmodels.api as sm
import matplotlib.pyplot as plt
import numpy as np
```

3 For any random initializations

[3]:

```
np.random.seed(42)
```

[ ]:

```

```

4 Data preview

[4]:

```
# Load the CSV file into a DataFrame
df = pd.read_csv("StockBeta.csv")

pd.set_option('display.max_rows', None)  # Display all rows
pd.set_option('display.max_columns', None)  # Display all columns

display(df)
```

	Date	S_P500	IBM	Apple
0	9/3/13	0.0395	0.0422	0.0039
1	8/1/13	-0.0313	-0.0608	0.0838
2	7/1/13	0.0495	0.0206	0.1412
3	6/3/13	-0.0150	-0.0813	-0.1183
4	5/1/13	0.0208	0.0319	0.0224
5	4/1/13	0.0181	-0.0505	0.0003
6	3/1/13	0.0360	0.0621	0.0028

7	2/1/13	0.0111	-0.0068	-0.0253
8	1/2/13	0.0504	0.0601	-0.1441
9	12/3/12	0.0071	0.0078	-0.0907
10	11/1/12	0.0028	-0.0187	-0.0124
11	10/1/12	-0.0198	-0.0622	-0.1076
12	9/4/12	0.0242	0.0647	0.0028
13	8/1/12	0.0198	-0.0015	0.0939
14	7/2/12	0.0126	0.0020	0.0458
15	6/1/12	0.0396	0.0139	0.0109
16	5/1/12	-0.0627	-0.0646	-0.0107
17	4/2/12	-0.0075	-0.0075	-0.0260
18	3/1/12	0.0313	0.0606	0.1053
19	2/1/12	0.0406	0.0254	0.1883
20	1/3/12	0.0436	0.0474	0.1271
21	12/1/11	0.0085	-0.0219	0.0596
22	11/1/11	-0.0051	0.0223	-0.0558
23	10/3/11	0.1077	0.0558	0.0615
24	9/1/11	-0.0718	0.0172	-0.0091
25	8/1/11	-0.0568	-0.0505	-0.0145
26	7/1/11	-0.0215	0.0601	0.1633
27	6/1/11	-0.0183	0.0155	-0.0349
28	5/2/11	-0.0135	-0.0053	-0.0066
29	4/1/11	0.0285	0.0460	0.0047
30	3/1/11	-0.0010	0.0074	-0.0133
31	2/1/11	0.0320	0.0032	0.0409
32	1/3/11	0.0226	0.1038	0.0520
33	12/1/10	0.0653	0.0375	0.0367
34	11/1/10	-0.0023	-0.0106	0.0338
35	10/1/10	0.0369	0.0706	0.0607
36	9/1/10	0.0876	0.0894	0.1672
37	8/2/10	-0.0474	-0.0363	-0.0550
38	7/1/10	0.0688	0.0399	0.0227
39	6/1/10	-0.0539	-0.0142	-0.0208
40	5/3/10	-0.0820	-0.0241	-0.0161
41	4/1/10	0.0148	0.0059	0.1110
42	3/1/10	0.0588	0.0085	0.1485
43	2/1/10	0.0285	0.0437	0.0654
44	1/4/10	-0.0370	-0.0651	-0.0886
45	12/1/09	0.0178	0.0360	0.0542
46	11/2/09	0.0574	0.0523	0.0605
47	10/1/09	-0.0198	0.0083	0.0170
48	9/1/09	0.0357	0.0132	0.1019
49	8/3/09	0.0336	0.0058	0.0295
50	7/1/09	0.0741	0.1293	0.1472
51	6/1/09	0.0002	-0.0175	0.0488
52	5/1/09	0.0531	0.0351	0.0793
53	4/1/09	0.0939	0.0652	0.1971
54	3/2/09	0.0854	0.0529	0.1770

55	2/2/09	-0.1099	0.0096	-0.0091
56	1/2/09	-0.0857	0.0890	0.0560
57	12/1/08	0.0078	0.0314	-0.0790
58	11/3/08	-0.0748	-0.1174	-0.1387
59	10/1/08	-0.1694	-0.2051	-0.0534
60	9/2/08	-0.0908	-0.0392	-0.3296
61	8/1/08	0.0122	-0.0451	0.0666
62	7/1/08	-0.0099	0.0797	-0.0507
63	6/2/08	-0.0860	-0.0842	-0.1129
64	5/1/08	0.0107	0.0767	0.0851
65	4/1/08	0.0475	0.0483	0.2122
66	3/3/08	-0.0060	0.0113	0.1478
67	2/1/08	-0.0348	0.0671	-0.0763
68	1/2/08	-0.0612	-0.0091	-0.3167
69	12/3/07	-0.0086	0.0277	0.0870
70	11/1/07	-0.0440	-0.0910	-0.0407
71	10/1/07	0.0148	-0.0143	0.2377
72	9/4/07	0.0358	0.0095	0.1083
73	8/1/07	0.0129	0.0582	0.0510
74	7/2/07	-0.0320	0.0514	0.0796
75	6/1/07	-0.0178	-0.0127	0.0070
76	5/1/07	0.0325	0.0471	0.2143
77	4/2/07	0.0433	0.0843	0.0741
78	3/1/07	0.0100	0.0141	0.0981
79	2/1/07	-0.0218	-0.0598	-0.0131
80	1/3/07	0.0141	0.0206	0.0105
81	12/1/06	0.0126	0.0569	-0.0744
82	11/1/06	0.0165	-0.0012	0.1305
83	10/2/06	0.0315	0.1268	0.0532
84	9/1/06	0.0246	0.0120	0.1346
85	8/1/06	0.0213	0.0502	-0.0015
86	7/3/06	0.0051	0.0077	0.1865
87	6/1/06	0.0001	-0.0386	-0.0418
88	5/1/06	-0.0309	-0.0261	-0.1509
89	4/3/06	0.0122	-0.0017	0.1223
90	3/1/06	0.0111	0.0278	-0.0842
91	2/1/06	0.0005	-0.0105	-0.0930
92	1/3/06	0.0255	-0.0110	0.0503
93	12/1/05	-0.0010	-0.0754	0.0600
94	11/1/05	0.0352	0.0883	0.1776
95	10/3/05	-0.0177	0.0208	0.0742
96	9/1/05	0.0069	-0.0050	0.1434
97	8/1/05	-0.0112	-0.0317	0.0993
98	7/1/05	0.0360	0.1247	0.1587
99	6/1/05	-0.0001	-0.0179	-0.0742
100	5/2/05	0.0300	-0.0081	0.1027
101	4/1/05	-0.0201	-0.1642	-0.1347
102	3/1/05	-0.0191	-0.0129	-0.0711

```
103   2/1/05  0.0189 -0.0071  0.1669
```

[5]:
```
df.columns
```

[5]:
```
Index(['Date', 'S_P500', 'IBM', 'Apple'], dtype='object')
```

[ ]:
```

```

5 Simple sanity checks

[6]:
```
df.isnull().sum()
```

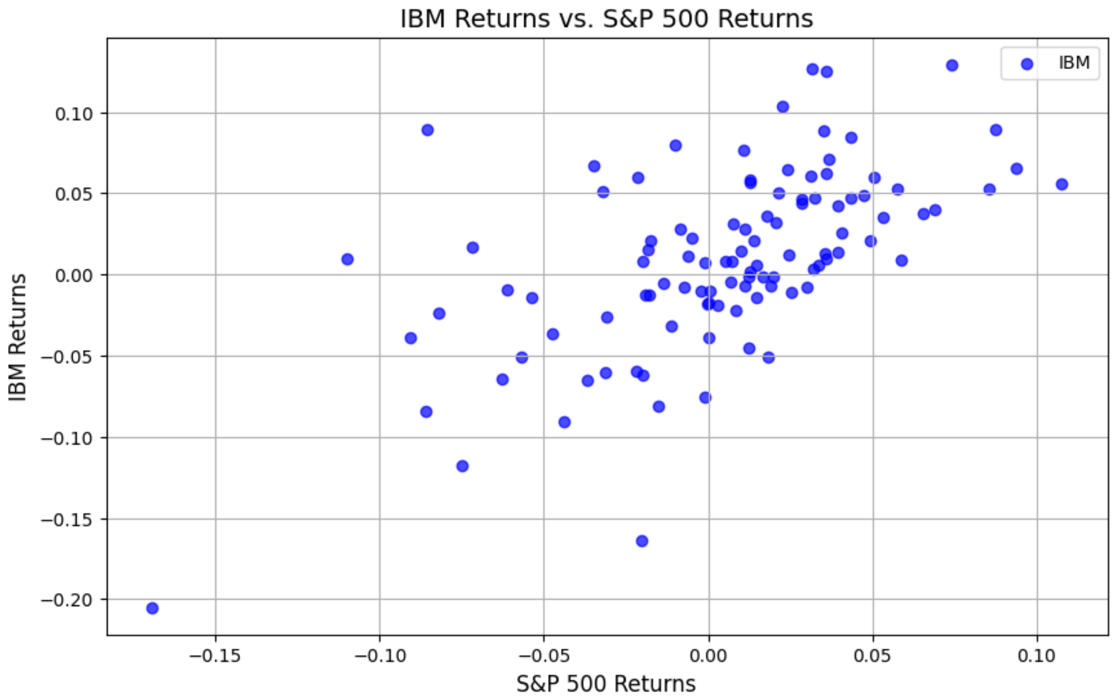
[6]:
```
Date      0
S_P500    0
IBM       0
Apple     0
dtype: int64
```

[ ]:
```

```

6 Plot the scatter plots

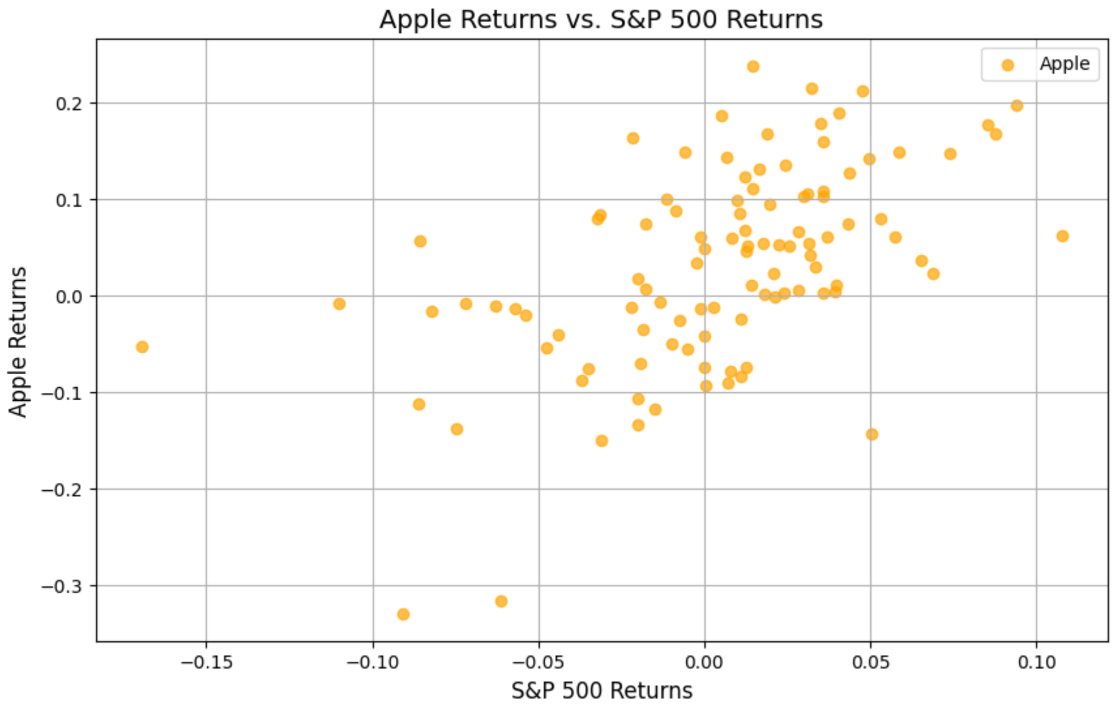
6.0.1 IBM vs S&P-500

[8]:
```
# Create scatter plot for IBM
plt.figure(figsize=(10, 6))
plt.scatter(df['S_P500'], df['IBM'], color='blue', label='IBM', alpha=0.7)
plt.title('IBM Returns vs. S&P 500 Returns', fontsize=14)
plt.xlabel('S&P 500 Returns', fontsize=12)
plt.ylabel('IBM Returns', fontsize=12)
plt.legend()
plt.grid(True)
plt.show()
```



6.0.2 Apple vs S&P-500

[9]:
```
# Create scatter plot for Apple
plt.figure(figsize=(10, 6))
plt.scatter(df['S_P500'], df['Apple'], color='orange', label='Apple', alpha=0.7)
plt.title('Apple Returns vs. S&P 500 Returns', fontsize=14)
plt.xlabel('S&P 500 Returns', fontsize=12)
plt.ylabel('Apple Returns', fontsize=12)
plt.legend()
plt.grid(True)
plt.show()
```



[ ]:

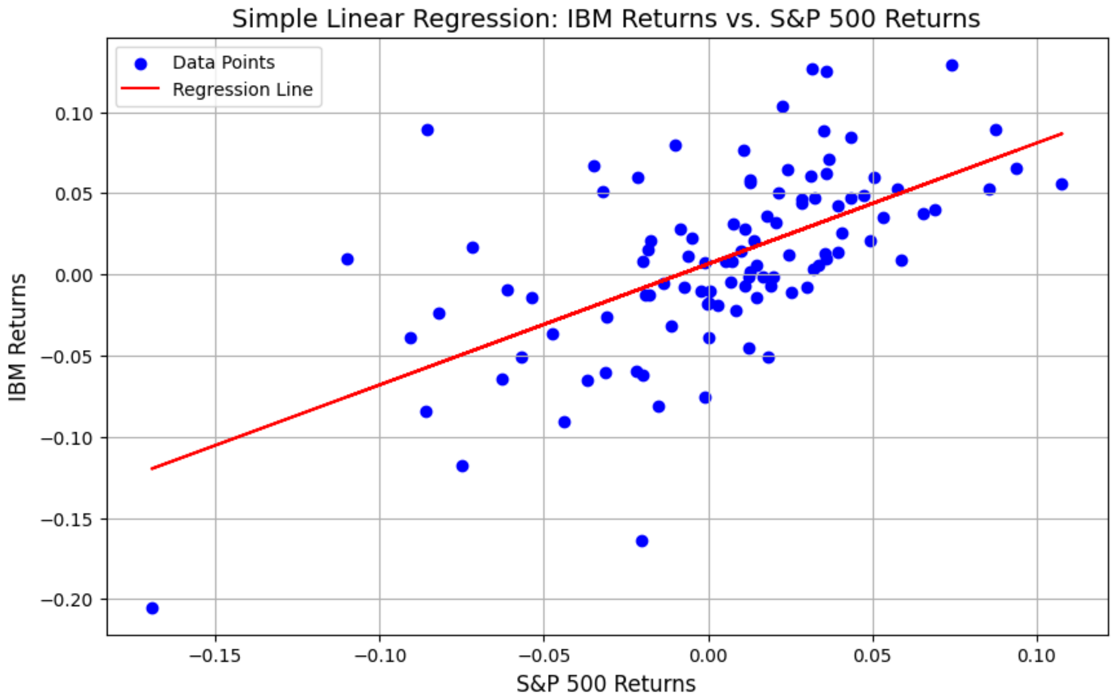
7 Building simple linear regression model for IBM against S&P-500

[10]:
```
# Define the independent variable (S&P 500 returns) and dependent variable (IBM␣
↪returns)
X_0 = df['S_P500']
y_0 = df['IBM']

# Add a constant, initialized to 1, to the independent variable. This will be␣
↪the intercept.
X_0 = sm.add_constant(X_0)
```

[12]:
```
# Fit the linear regression model
model = sm.OLS(y_0, X_0).fit()

# Print the summary table of the regression model
print(model.summary())

# Optional: Plot the regression line
plt.figure(figsize=(10, 6))
plt.scatter(df['S_P500'], df['IBM'], color='blue', label='Data Points')
```

```python
plt.plot(df['S_P500'], model.predict(X_0), color='red', label='Regression Line')
plt.title('Simple Linear Regression: IBM Returns vs. S&P 500 Returns',
 ↪fontsize=14)
plt.xlabel('S&P 500 Returns', fontsize=12)
plt.ylabel('IBM Returns', fontsize=12)
plt.legend()
plt.grid(True)
plt.show()
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    IBM   R-squared:                       0.357
Model:                            OLS   Adj. R-squared:                  0.351
Method:                 Least Squares   F-statistic:                     56.63
Date:                Mon, 30 Sep 2024   Prob (F-statistic):           2.15e-11
Time:                        03:19:44   Log-Likelihood:                 176.46
No. Observations:                 104   AIC:                            -348.9
Df Residuals:                     102   BIC:                            -343.6
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0064      0.004      1.454      0.149      -0.002       0.015
S_P500         0.7448      0.099      7.525      0.000       0.548       0.941
==============================================================================
Omnibus:                        6.373   Durbin-Watson:                   2.006
Prob(Omnibus):                  0.041   Jarque-Bera (JB):                9.404
Skew:                           0.194   Prob(JB):                      0.00908
Kurtosis:                       4.421   Cond. No.                         22.5
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```



7.0.1 The value for Beta relating returns on S&P-500 to the returns on IBM

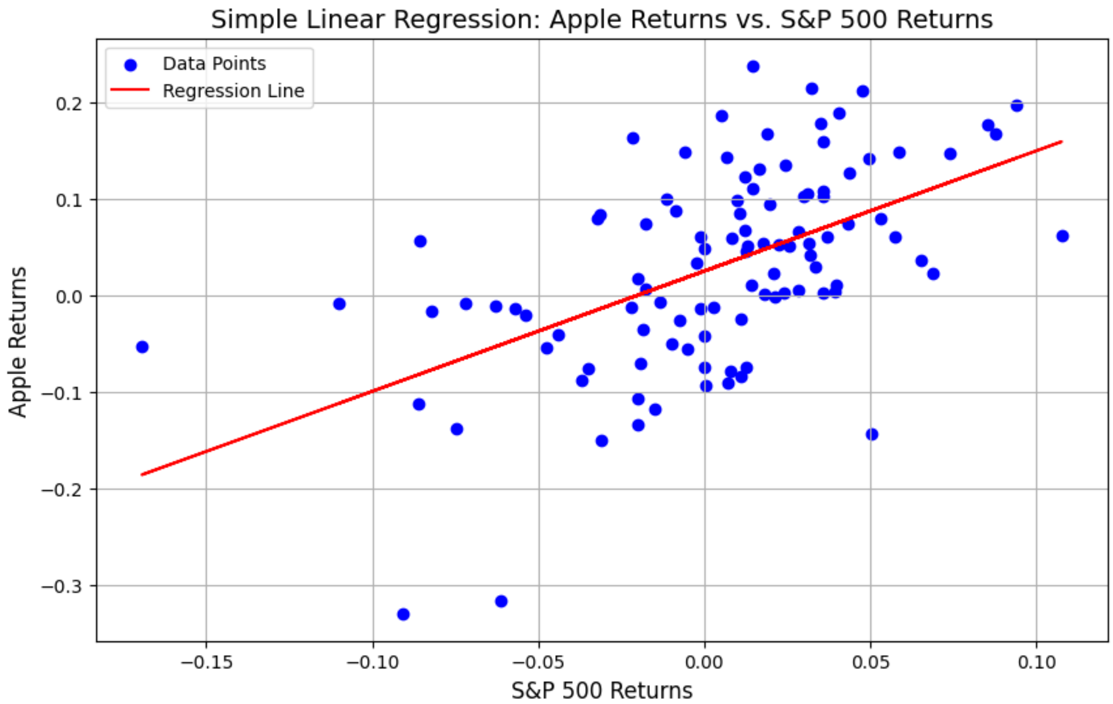
-> 0.7448

[13]:
```
# Define the independent variable (S&P 500 returns) and dependent variable (IBM␣
↪returns)
X_1 = df['S_P500']
y_1 = df['Apple']

# Add a constant to the independent variable
X_1 = sm.add_constant(X_1)
```

[31]:
```
# Fit the linear regression model
model = sm.OLS(y_1, X_1).fit()

# Print the summary table of the regression model
print(model.summary())

# Optional: Plot the regression line
plt.figure(figsize=(10, 6))
plt.scatter(df['S_P500'], df['Apple'], color='blue', label='Data Points')
plt.plot(df['S_P500'], model.predict(X_1), color='red', label='Regression Line')
plt.title('Simple Linear Regression: Apple Returns vs. S&P 500 Returns',␣
↪fontsize=14)
plt.xlabel('S&P 500 Returns', fontsize=12)
```

```python
plt.ylabel('Apple Returns', fontsize=12)
plt.legend()
plt.grid(True)
plt.show()
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Apple   R-squared:                       0.290
Model:                            OLS   Adj. R-squared:                  0.283
Method:                 Least Squares   F-statistic:                     41.60
Date:                Mon, 30 Sep 2024   Prob (F-statistic):           3.80e-09
Time:                        03:33:19   Log-Likelihood:                 107.01
No. Observations:                 104   AIC:                            -210.0
Df Residuals:                     102   BIC:                            -204.7
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0249      0.009      2.889      0.005       0.008       0.042
S_P500         1.2449      0.193      6.450      0.000       0.862       1.628
==============================================================================
Omnibus:                        3.933   Durbin-Watson:                   1.852
Prob(Omnibus):                  0.140   Jarque-Bera (JB):                3.274
Skew:                          -0.390   Prob(JB):                        0.195
Kurtosis:                       3.385   Cond. No.                         22.5
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```



7.0.2 The value for Beta relating returns on S&P-500 to the returns on Apple -> 1.2449

[ ]:

[ ]:

[ ]:

8 Calculating the sample standard deviations

[32]:
```
stock_2_sample_std_dev = dict()

for col in df.columns[1:]:
    # Calculate standard deviation
    std_dev = df[col].std()   # Sample standard deviation (N-1) . default ddof␣
 ↪is 1. so total obs. (N) - 1.
    std_dev_population = df[col].std(ddof=0)   # Population standard deviation␣
 ↪(N)

    print("\n")

    print(f"Sample Standard Deviation {col}:", std_dev)
```

```
    stock_2_sample_std_dev[col] = std_dev
```

```
Sample Standard Deviation S_P500: 0.04457852558188211


Sample Standard Deviation IBM: 0.055571051134871506


Sample Standard Deviation Apple: 0.10310404422155045
```

9 Getting the correlation matrix

[36]:
```
correlation_matrix_returns = df[['S_P500', 'IBM', 'Apple']].corr()
print(correlation_matrix_returns)
print(f"\nCorr matrix data type is:- {type(correlation_matrix_returns)}")
```

```
          S_P500       IBM     Apple
S_P500  1.000000  0.597478  0.538232
IBM     0.597478  1.000000  0.414725
Apple   0.538232  0.414725  1.000000

Corr matrix data type is:- <class 'pandas.core.frame.DataFrame'>
```

[41]:
```
r_sp500_ibm = correlation_matrix_returns.iloc[0]['IBM']
print(f"Corr between S&P-500 and IBM {r_sp500_ibm}")
r_sp500_apple = correlation_matrix_returns.iloc[0]['Apple']
print(f"\nCorr between S&P-500 and Apple {r_sp500_apple}")
```

```
Corr between S&P-500 and IBM 0.5974779352335425

Corr between S&P-500 and Apple 0.5382316734516045
```

[43]:
```
s_apple = stock_2_sample_std_dev["Apple"]
s_ibm = stock_2_sample_std_dev["IBM"]
s_sp500 = stock_2_sample_std_dev["S_P500"]
print( "sample std dev s&p-500:" , s_sp500)
print( "\nsample std dev IBM:" , s_ibm)
print( "\nsample std dev Apple:", s_apple)
```

```
sample std dev s&p-500: 0.04457852558188211

sample std dev IBM: 0.055571051134871506

sample std dev Apple: 0.10310404422155045
```

[ ]:

```

```

10 Calculating r*(sx/sy)

[22]:

```
r_sp500_ibm = correlation_matrix_returns.iloc[0]['IBM']
print(f"Corr between S&P-500 and IBM {r_sp500_ibm}")
```

```
0.5974779352335425
```

[39]:

```
r_sp500_apple = correlation_matrix_returns.iloc[0]['Apple']
print(f"Corr between S&P-500 and IBM {r_sp500_apple}")
```

```
Corr between S&P-500 and IBM 0.5382316734516045
```

[ ]:

```

```

[ ]:

```

```

[25]:

```
r__s_sp500__ibm = r_sp500_ibm * ( s_ibm / s_sp500 )
r__s_sp500__ibm
```

[25]: 0.7448087718790547

[ ]:

```

```

[26]:

```
r__s_sp500__apple = r_sp500_apple * ( s_apple / s_sp500 )
r__s_sp500__apple
```

[26]: 1.244856386267461

[ ]:

```

```

10.0.1 Looking at the metrics together

[47]:

```
r_sp500_ibm = correlation_matrix_returns.iloc[0]['IBM']
print(f"Corr between S&P-500 and IBM {r_sp500_ibm}")
r_sp500_apple = correlation_matrix_returns.iloc[0]['Apple']
print(f"\nCorr between S&P-500 and Apple {r_sp500_apple}")

s_apple = stock_2_sample_std_dev["Apple"]
s_ibm = stock_2_sample_std_dev["IBM"]
s_sp500 = stock_2_sample_std_dev["S_P500"]
print("\n\n----------")
print( "\n\n\nsample std dev s&p-500:" , s_sp500)
print( "\nsample std dev IBM:" , s_ibm)
print( "\nsample std dev Apple:", s_apple)

print("\n\n----------")
```

```
print("r*sx/sy for s&p-500 and IBM", r__s_sp500__ibm)
print("\nr*sx/sy for s&p-500 and Apple", r__s_sp500__apple)
```

```
Corr between S&P-500 and IBM 0.5974779352335425

Corr between S&P-500 and Apple 0.5382316734516045


----------



sample std dev s&p-500: 0.04457852558188211

sample std dev IBM: 0.05557105113487150 6

sample std dev Apple: 0.10310404422155045


----------
r*sx/sy for s&p-500 and IBM 0.7448087718790547

r*sx/sy for s&p-500 and Apple 1.244856386267461
```

[48]:
```
0.05557105113487150 6/0.04457852558188211 , 0.10310404422155045/0.
 ↪04457852558188211
```

[48]:
```
(1.2465879122179189, 2.312863489219003)
```

The expected return on APPLE in relation to returns on S&P-500 can be said to be

a function of the ratio of ->

- **Sample standard deviation of APPLE returns (spread of APPLE returns data) and the same for S&P-500,**

  i.e., how many units of std deviation in 'APPLE stock returns' for every one unit std deviation in 'S&P-500 stock returns'.

  This ratio is higher for the APPLE stock as compared to IBM's.

  Multiplying this ratio with the correlation coefficient (which are almost comparable for the 2), yields a coefficient

Hw1 Question(2/3)

2024-09-27

```r
auto <- read.table("auto.txt", sep = "", header = T)
remove <- which(is.na(auto),arr.ind = T)[,1]
auto <- auto[-remove,]
auto$origin <-  factor((auto$origin), labels = c("US", "Europe", "Japan"))
lm1 <- lm(mpg ~ horsepower, data = auto)
summary(lm1)
```

```
##
## Call:
## lm(formula = mpg ~ horsepower, data = auto)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.5710  -3.2592  -0.3435   2.7630  16.9240
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 39.935861   0.717499   55.66   <2e-16 ***
## horsepower  -0.157845   0.006446  -24.49   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.906 on 390 degrees of freedom
## Multiple R-squared:  0.6059, Adjusted R-squared:  0.6049
## F-statistic: 599.7 on 1 and 390 DF,  p-value: < 2.2e-16
```

2(a): The estimated regression equation is $y = -0.158 * horsepower + 39.936$ 2(b): The slope tells us that on average, one unit increase of the horse power is associated with a decrease 0.158 of mpg. 2(d): The residual standard error tells me the standard error of the estimated regression equation is 4.906. 2(e): The adjusted R-squared is 0.6049, with p-value of the slope less than 2e-16, there is a statistically significant relationship between mpg and horsepower 2(f): 60.59% of variation in mpg is explained by using linear function of horsepower. 2(g): The answer is $39.935861 - 0.157845 * 98 = 24.467$ 2(h): The equation is $(B_0 + B_1 x_0) \pm t_{0.025} * \sigma_e)$ and is equal to (14.821, 34.113) 2(i): We calculate it using predict in R, which also gives us the standard error of the mean predictor

```r
predict(lm1, data.frame(horsepower = 98), interval = "prediction", level = 0.95, se.fit = T)
```

```
## $fit
##        fit     lwr      upr
## 1 24.46708 14.8094 34.12476
##
## $se.fit
## [1] 0.2512623
##
## $df
## [1] 390
```

```
##
## $residual.scale
## [1] 4.905757
```

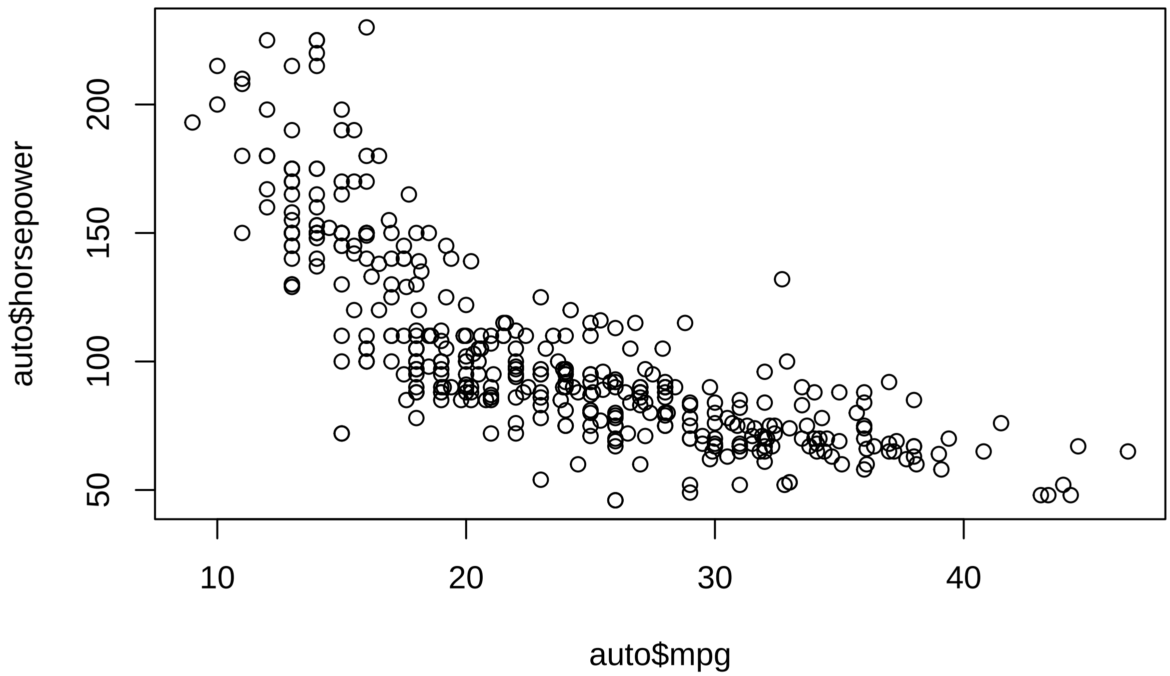
2(j):

```
c(-0.157845 - 0.006446*qt(0.95, 390), -0.157845+0.006446*qt(0.95,390))
```

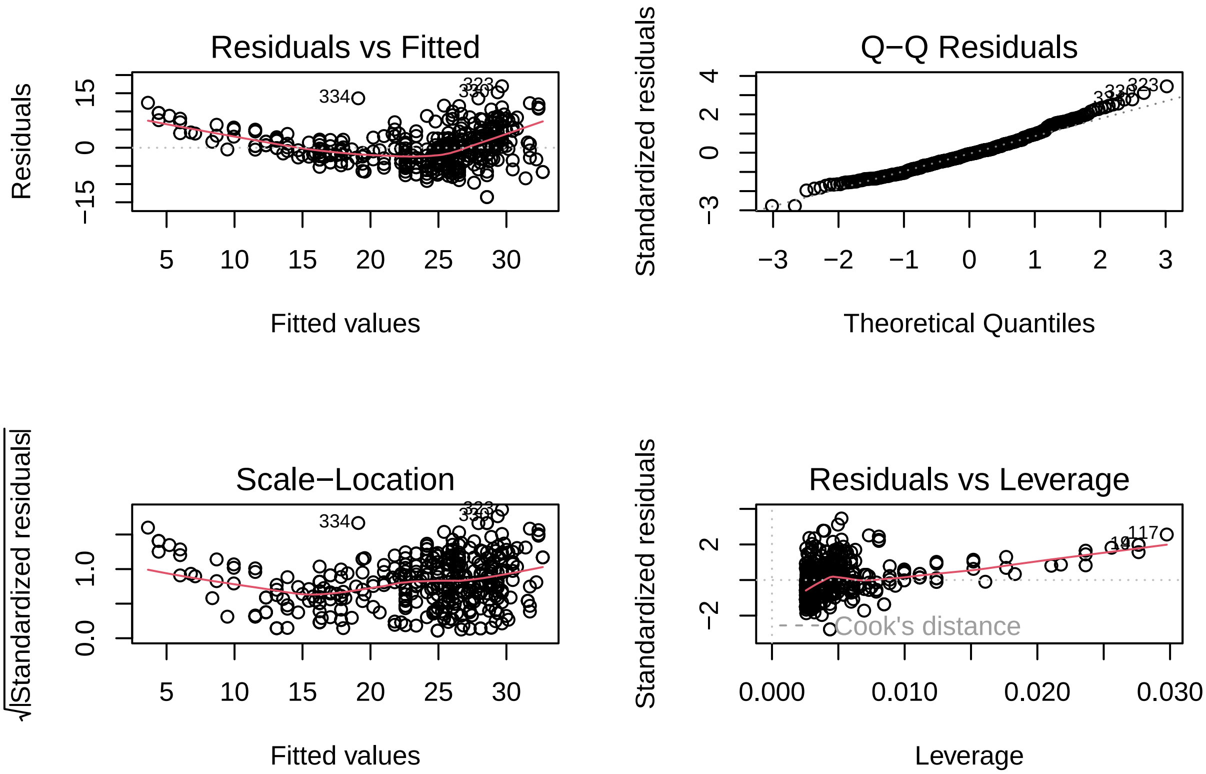
```
## [1] -0.168473 -0.147217
```

2(k):

```
plot(auto$mpg, auto$horsepower)
```




```
par(mfrow = c(2,2))
plot(lm1)
```



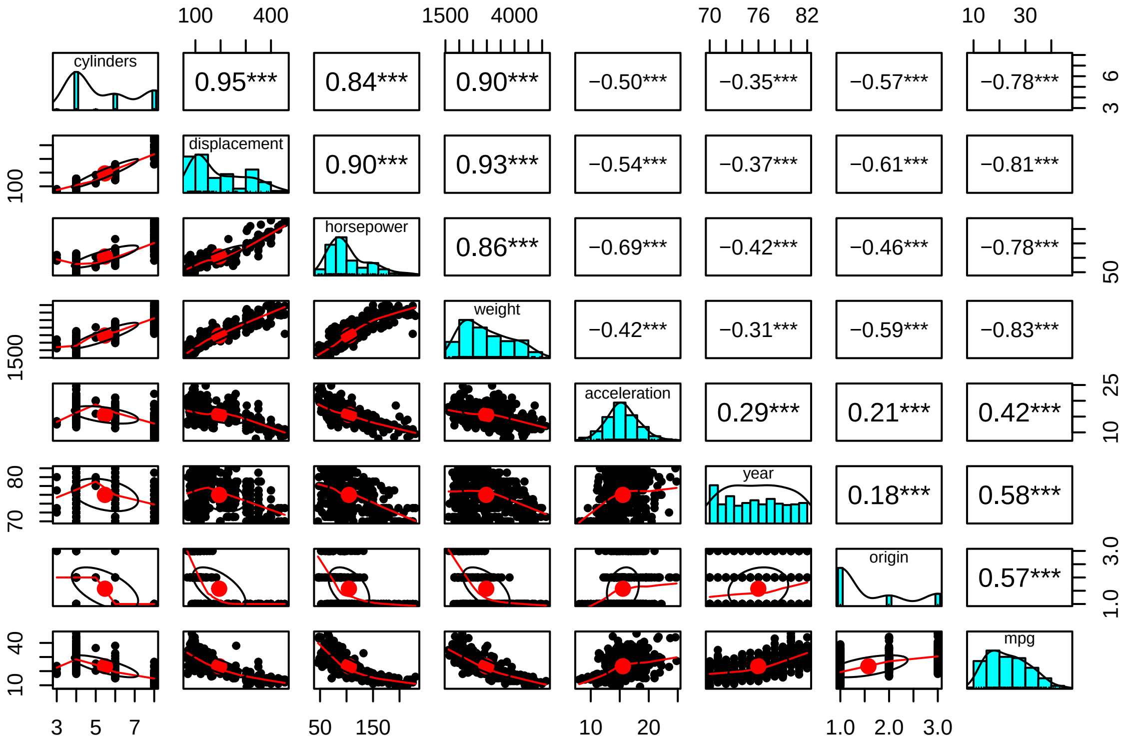
From the scatterplot, we can see that this is a negative and nonlinear relationship. The possible relationship seemes to be logarithmic. By looking at the Residual vs Fitted plot, we can see that the residuals follow a curved line, indicating nonlinear relationship between predictor and response variable. The Q-Q residuals plot seems to work well, indicating the normal assumptions are met.

problem 3

```
library(psych)
```

```
## Warning: package 'psych' was built under R version 4.3.3
```

```
pairs.panels(auto[,c(2:8,1)], stars=T, density=T) # part a
```




```r
round(cor(auto[,1:7], use="pair"),4)    # part b
```

```
##                  mpg cylinders displacement horsepower  weight acceleration
## mpg           1.0000   -0.7776      -0.8051    -0.7784 -0.8322       0.4233
## cylinders    -0.7776    1.0000       0.9508     0.8430  0.8975      -0.5047
## displacement -0.8051    0.9508       1.0000     0.8973  0.9330      -0.5438
## horsepower   -0.7784    0.8430       0.8973     1.0000  0.8645      -0.6892
## weight       -0.8322    0.8975       0.9330     0.8645  1.0000      -0.4168
## acceleration  0.4233   -0.5047      -0.5438    -0.6892 -0.4168       1.0000
## year          0.5805   -0.3456      -0.3699    -0.4164 -0.3091       0.2903
##                 year
## mpg           0.5805
## cylinders    -0.3456
## displacement -0.3699
## horsepower   -0.4164
## weight       -0.3091
## acceleration  0.2903
## year          1.0000
```

```r
fit = lm(mpg~., auto[,1:8])      # part c
summary(fit)
```

```
##
## Call:
## lm(formula = mpg ~ ., data = auto[, 1:8])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.0095 -2.0785 -0.0982  1.9856 13.3608
##
## Coefficients:
```

```
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.795e+01  4.677e+00  -3.839 0.000145 ***
## cylinders    -4.897e-01  3.212e-01  -1.524 0.128215
## displacement  2.398e-02  7.653e-03   3.133 0.001863 **
## horsepower   -1.818e-02  1.371e-02  -1.326 0.185488
## weight       -6.710e-03  6.551e-04 -10.243  < 2e-16 ***
## acceleration  7.910e-02  9.822e-02   0.805 0.421101
## year          7.770e-01  5.178e-02  15.005  < 2e-16 ***
## originEurope  2.630e+00  5.664e-01   4.643 4.72e-06 ***
## originJapan   2.853e+00  5.527e-01   5.162 3.93e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.307 on 383 degrees of freedom
## Multiple R-squared:  0.8242, Adjusted R-squared:  0.8205
## F-statistic: 224.5 on 8 and 383 DF,  p-value: < 2.2e-16
```

```r
# Assuming 'auto' is your dataset
library(psych)

auto$origin <-  factor((auto$origin), labels = c("US", "Europe", "Japan"))

# Fit the multiple linear regression model
fit = lm(mpg ~ cylinders + displacement + horsepower + weight + acceleration + year + origin, data = au

# Summary of the regression model
summary_fit = summary(fit)

# (c) F-test results
f_statistic = summary_fit$fstatistic
f_p_value = pf(f_statistic[1], f_statistic[2], f_statistic[3], lower.tail = FALSE)

cat("(c) F-statistic:", f_statistic[1], "\n")
```

```
## (c) F-statistic: 224.4507
```

```r
cat("   p-value:", f_p_value, "\n\n")
```

```
##    p-value: 1.789724e-139
```

```r
# (d) Statistically significant predictors
cat("(d) Statistically significant predictors:\n")
```

```
## (d) Statistically significant predictors:
```

```r
significant_predictors = summary_fit$coefficients[summary_fit$coefficients[,4] < 0.05, ]
print(significant_predictors)
```

```
##                  Estimate   Std. Error    t value     Pr(>|t|)
## (Intercept)  -17.954602067 4.6769339310  -3.838969 1.445124e-04
## displacement   0.023978644 0.0076532690   3.133124 1.862685e-03
## weight        -0.006710384 0.0006551331 -10.242779 6.375633e-22
## year           0.777026939 0.0517840867  15.005130 2.332943e-40
## originEurope   2.630002360 0.5664146647   4.643246 4.720373e-06
## originJapan    2.853228228 0.5527363020   5.162006 3.933208e-07
```

```r
cat("\n")
```

```r
# (e) and (f) Slope coefficients for year and displacement
year_coef = summary_fit$coefficients["year", ]
displacement_coef = summary_fit$coefficients["displacement", ]

cat("(e) Slope coefficient for year:", year_coef[1], "\n")
```

```
## (e) Slope coefficient for year: 0.7770269
```

```r
cat("   p-value:", year_coef[4], "\n\n")
```

```
##    p-value: 2.332943e-40
```

```r
cat("(f) Slope coefficient for displacement:", displacement_coef[1], "\n")
```

```
## (f) Slope coefficient for displacement: 0.02397864
```

```r
cat("   p-value:", displacement_coef[4], "\n")
```

```
##    p-value: 0.001862685
```

a) Among all predictors, Cylinders, displacement, horsepower, and weight show relatively strong negative relationships with mpg. As these variables increase, mpg tends to decrease. While acceleration, year, and origin show relatively weak positive relationships with mpg. As these variables increase, mpg tends to increase as well.

b) The correlation between mpg and displacement is -0.80. This strong negative correlation indicates that as displacement increases, mpg tends to decrease significantly. Since all of the correlation coefficients are marked with three asterisks (***) between predictors and mpg, it indicates all variables have significant relationships with mpg.

c) Since the F-test is 224.4507, which is highly significant, it suggests that there is a statistically significant relationship between the set of predictors and the mpg response variable.

d) All predictors have p-values less than the significance level of 0.05, indicating that they all have a statistically significant relationship with the response variable mpg. Among these predictors, year has the most significant relationship since it has the lowest p-value and highest t-value.

e) The positive slope of 0.7770269 suggests that for each additional year (as cars get newer), the mpg increases by approximately 0.7770269 units, holding all other variables constant.

f) The positive slope of 0.02397864 suggests that for each unit increase in displacement, the mpg slightly increases by approximately 0.02397864 units, holding all other variables constant.

PROBLEM 4 of ML-1, Homework-1

GIVEN :→

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \text{ and } A = \begin{bmatrix} a_{11} & a_{12} \\ a_{12} & a_{22} \end{bmatrix}$$

⇒ EVALUATING THE EXPRESSION :→

$$x^T \cdot A \cdot x$$

$$\Rightarrow \begin{bmatrix} x_1 & x_2 \end{bmatrix}_{1\times2} \cdot \begin{bmatrix} a_{11} & a_{12} \\ a_{12} & a_{22} \end{bmatrix}_{2\times2} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}_{2\times1}$$

$$= \begin{bmatrix} a_{11} \times x_1 + a_{12} \times x_2 & a_{12} \times x_1 + a_{22} \times x_2 \end{bmatrix}_{1\times2} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}_{2\times2}$$

$$= \left(a_{11} \times x_1^2 + a_{12} \times x_2 \times x_1\right) + \left(a_{12} \times x_1 \times x_2 + a_{22} \times x_2^2\right)$$

$$\cancel{= 2 \times a_{11} \times x_1 + 2 \times a_{12} \times x_1 \times x_2 + 2 \times a_{22} \times x_2}$$

$$\cancel{+ a_{12} \times x_1 + a_{12} \times x_2}$$

Taking partial derivative of the above expression with respect to 'x'.

classmate

Date

Page

$$\Rightarrow \frac{\partial (x^T \cdot A \cdot x)}{\partial x}$$

$\Rightarrow$ Taking partial derivative w.r.t. x_1

$$\Rightarrow \frac{\partial (a_{11} \times x_1^2 + 2 \times a_{12} \times x_1 \times x_2 + a_{22} \times x_2^2)}{\partial x_1}$$

$$= 2 \times a_{11} \times x_1 + 2 \times a_{12} \times x_2 + 0 \cancel{= 2\times}$$

$\Rightarrow$ w.r.t. x_2,

$$\frac{\partial (a_{11} \times x_1^2 + 2 \times a_{12} \times x_1 \times x_2 + a_{22} \times x_2^2)}{\partial x_2}$$

$$= 0 + 2 \times a_{12} \times x_1 + 2 \times a_{22} \times x_2$$

$\Rightarrow$ Evaluating the expression $2 \times A \cdot x$

$$\Rightarrow 2 \times \begin{bmatrix} a_{11} & a_{12} \\ a_{12} & a_{22} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$= 2 \times \begin{bmatrix} a_{11} \times x + a_{12} \times x_2 \\ a_{12} \times x_1 + a_{22} \times x_2 \end{bmatrix}$$

, which are the same as the expressions derived via the partial derivatives.

$$\Rightarrow \begin{matrix} 2 \times (a_{11} \times x_1 + a_{12} \times x_2) \\ 2 \times (a_{12} \times x_1 + a_{22} \times x_2) \end{matrix}$$

5.(a) $n \times 2$ matrix has the form $\begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix}$

(b) $X^T X = \begin{bmatrix} 1 & \cdots & 1 \\ x_1 & \cdots & x_n \end{bmatrix} \begin{bmatrix} 1 & x_1 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix} = \begin{bmatrix} n & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} x_i^2 \end{bmatrix}$

(c) Yes, it is symmetric

(d)

$$A = \begin{bmatrix} 1 & \cdots & 1 \\ x_{11} & \cdots & x_{1n} \\ x_{21} & \cdots & x_{2n} \\ \vdots & & \vdots \\ x_{p1} & \cdots & x_{pn} \end{bmatrix} \begin{bmatrix} 1 & x_{11} & x_{21} & \cdots & x_{p1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_{1n} & x_{2n} & \cdots & x_{pn} \end{bmatrix}$$

$$A = \begin{bmatrix} n & \sum_i x_{1i} & & \\ \sum_i x_{1i} & \sum_{i=1}^{n} x_{1i}^2 & & \\ & & \ddots & \\ & & & \sum_{i=1}^{n} x_{pi}^2 \end{bmatrix}$$

we wish to show $A_{ij} = A_{ji}$, for all $i \neq j$

$A_{ij} = X^T(i,:) \cdot X(:,j)$, $A_{ji} = X^T(j,:)\, X(:,i)$

Notice that $X^T(i,:) = X(:,i)$ Row i of X^T = Column i of X

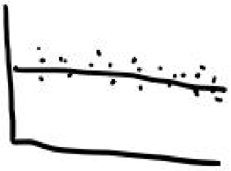
By definition of a Transpose of a matrix

$X(:,j) = X^T(j,:)$ by same reason. Since it is a dot product, where we sum the multiple of the vector, $A_{ij} = A_{ji}$ for all $i \neq j$

Question_6-HW1

Monday, September 30, 2024 10:20 AM

a) implication: horizontal line $y=\alpha$



because $E(e_i)=0$

so mean $e_i=0$

if $\beta=0$ $y_i=\alpha+e_i$

b) $y_i=\alpha+e_i$ derivative= minimize

least square estimator minimizes SSR

$S(a)=\sum_{i=1}^{n}(y_i-a)^2 \to$ SSR

$$\frac{d}{da}=\sum_{i=1}^{n}(y_i-a)^2=-2/2\sum_{i=1}^{n}(y_i-a)=0/2$$

$$\sum_{i=1}^{n}y_i-na=0 \quad \sum_{i=1}^{n}y_i=na$$

$$\boxed{a=\frac{1}{n}\sum_{i=1}^{n}y_i=\overline{y}}$$

c) $a=1/n\sum_{i=1}^{n}y_i=\frac{1}{n}\sum_{i=1}^{n}(\alpha+e_i)$

take expectation to see if unbiased

$$E(a)=\frac{1}{n}\sum_{i=1}^{n}E(\alpha+e_i)=\frac{1}{n}\sum_{i=1}^{n}(\alpha+E(e_i)$$

$$E(e_i)=0\to\boxed{E(a)=\frac{1}{n}\sum_{i=1}^{n}\alpha=\alpha}$$

d) $V(a)=V(\frac{1}{n}\sum_{i=1}^{n}(\alpha+e_i))$

$$V(\alpha)=0\to V(\frac{1}{n}\sum_{i=1}^{n}e_i)=\frac{1}{n^2}\sum_{i=1}^{n}\sigma^2\to\boxed{\frac{\sigma^2}{n}}$$

e) The estimates are normally distributed by the Central Limit Theorem, which states that with enough n values (a lot of them) $\overline{Y}$ follows a normal distribution, as long as they are independent and identically distributed.

f) i) unbiased $\rightarrow E(\hat{\alpha}) = \alpha$

$\hookrightarrow E\left(\sum_{i=1}^{n} c_i \underline{y_i}\right) = \sum_{i=1}^{n} c_i \underline{E(y_i)} = \sum_{i=1}^{n} c_i \left(\underline{\alpha^{\nearrow 1} + E(e_i)^{\nearrow 0}}\right) = \alpha \sum_{i=1}^{n} c_i = 1$

implies sum of $C = 1$

ii) $\sum_{i=1}^{n} d_i / n = 0 \quad \hat{\alpha} = \sum_{i=1}^{n} c_i y_i$

$d_i = c_i - 1/n$

$E\left(\sum_{i=1}^{n} c_i y_i\right) = \alpha \quad \sum_{i=1}^{n} c_i \alpha = \alpha \rightarrow \sum_{i=1}^{n} c_i = 1 \rightarrow c_i = 1/n + d_i$

$\sum_{i=1}^{n} \left(\frac{1}{n} + d_i\right) = 1 \rightarrow \sum_{i=1}^{n} \frac{1}{n} + \sum_{i=1}^{n} d_i = 1$

$\sum_{i=1}^{n} \frac{1}{n} = \frac{n}{n} = 1 \quad \sum_{i=1}^{n} d_i = 0$

$\boxed{\sum_{i=1}^{n} d_i / n = 1/n \sum_{i=1}^{n} d_i = 0}$

iii) $y_i = \alpha + e_i$

$V(\hat{\alpha}) = V\left(\sum_{i=1}^{n} c_i y_i\right) \rightarrow V\left(\sum_{i=1}^{n} c_i (\alpha + e_i)\right)$

$\hat{\alpha} = \alpha \underbrace{\sum_{i}^{n} c_i}_{1} + \sum_{i=1}^{n} c_i e_i$

$V\left(\sum_{i=1}^{n} c_i e_i\right) \rightarrow \sum_{i=1}^{n} c_i^2 V(e_i)^{\nearrow \sigma^2}$

$\sigma^2 \sum_{i=1}^{n} \underline{(d_i + 1/n)^2} = \underline{\cancel{d_i^2}^{=0} + \cancel{2 d_i / n}^{=0} + 1/n^2}$

$\sigma^2 \sum_{i=1}^{n} 1/n^2 = \sigma^2 \cdot \cancel{n} \cdot 1/n^2 = \frac{\sigma^2}{n}$

$\boxed{\text{variance is minimized when } V(\hat{\alpha}) = \frac{\sigma^2}{n}}$