Assignment 1

Abhishek Prataap Raghuvir

Edwin Jeyakumar

Hridayraj Kartik Modi

Mayuresh Budukh

Yuanhui Jiang

Question 1

1. Visualization between price, horsepower and bodystyle

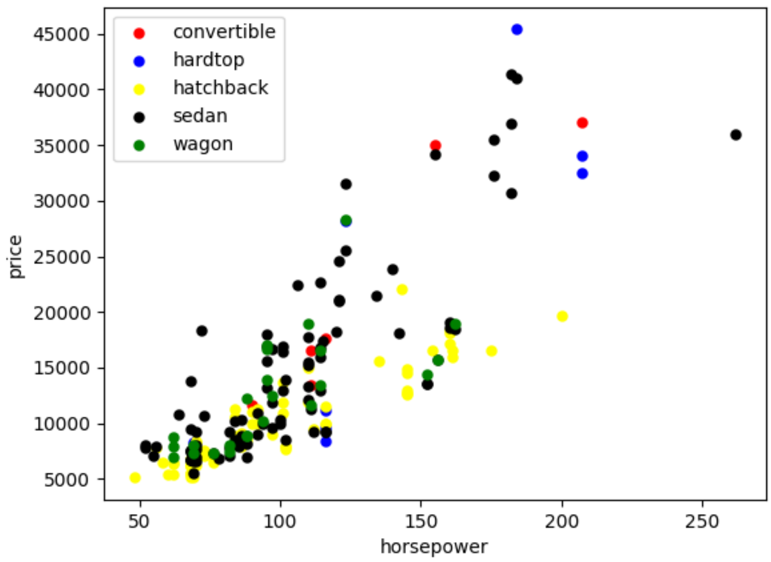
In [ ]:

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

#import dataset
data=pd.read_csv('C:\\Users\\Downloads\\imports-85.csv')

#calculate log and squared price
data['log_price']=np.log(data['price'])
data['squared_price']=data['price']**2
data=data.merge(pd.get_dummies(data['body-style']),how='left',left_index=True,right_index=True)
```

When using price as dependent variable:

In [ ]:

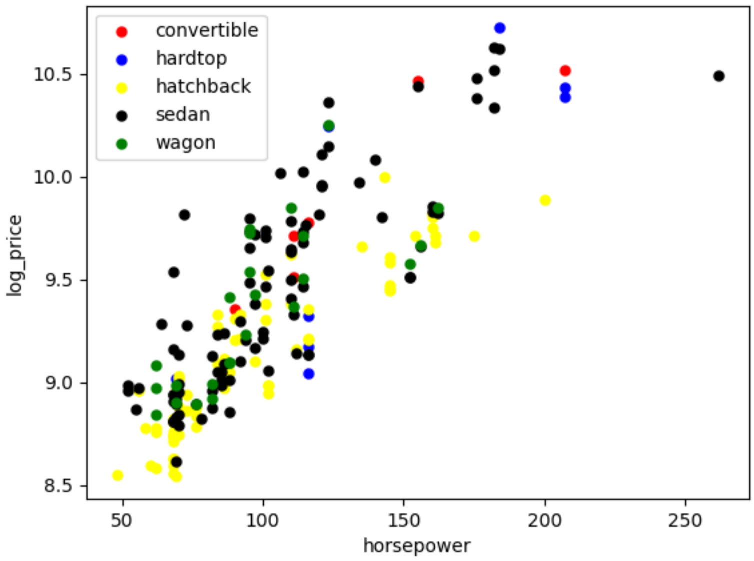
```
#create labeled scatterplot
for bodystyle,color in [('convertible','red'),('hardtop','blue'),('hatchback','yellow'),('sedan','black'),('wagon','green')]:
    temp_data=data[data['body-style']==bodystyle]
    plt.scatter(temp_data['horsepower'],temp_data['price'],c=color,label=bodystyle,linewidths=.05)

plt.xlabel("horsepower")
plt.ylabel("price")
plt.legend()
plt.show()
```



When using log price as dependent variable:

In [ ]:

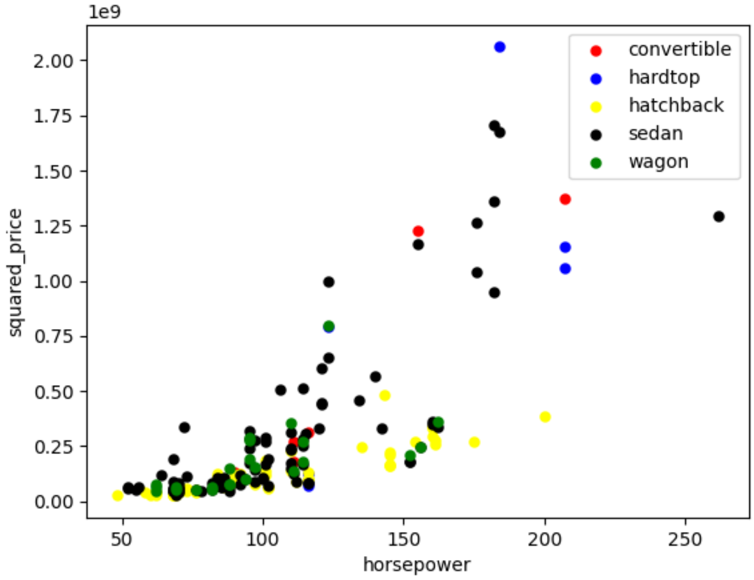
```
for bodystyle,color in [('convertible','red'),('hardtop','blue'),('hatchback','yellow'),('sedan','black'),('wagon','green')]:
    temp_data=data[data['body-style']==bodystyle]
    plt.scatter(temp_data['horsepower'],temp_data['log_price'],c=color,label=bodystyle,linewidths=.05)

plt.xlabel("horsepower")
plt.ylabel("log_price")
plt.legend()
plt.show()
```



When using squared price as dependent variable:

In [ ]:

```
for bodystyle,color in [('convertible','red'),('hardtop','blue'),('hatchback','yellow'),('sedan','black'),('wagon','green')]:
    temp_data=data[data['body-style']==bodystyle]
    plt.scatter(temp_data['horsepower'],temp_data['squared_price'],c=color,label=bodystyle,linewidths=.05)

plt.xlabel("horsepower")
plt.ylabel("squared_price")
plt.legend()
plt.show()
```



From the scatter plots we can see: For convertible, hardtop and sedan bodystyle, their average prices seem to be higher than the other two bodystyle. Hatchback and wagon have the lowest average prices. Thus bodystyle appear to be relevant for prices, regardless of the effect of horsepower.

2. Regress log_price on horsepower with intercept:

In [ ]:

```
data['cons']=1
mod2 = sm.OLS(data['log_price'],data[['horsepower','cons']],missing='drop')
res2=mod2.fit()
print(res2.summary())

#draw regression diagnostic plot
plt.scatter(data['horsepower'],data['log_price'],linewidths=.05)
plt.plot(data['horsepower'][res2.fittedvalues.index],res2.fittedvalues,label='fitted value')
plt.xlabel("horsepower")
plt.ylabel("log_price")
plt.legend()
plt.show()

#or can use seaborn_qqplot package
from seaborn_qqplot import pplot
pplot(data, x="horsepower", y="log_price",kind='qq',display_kws={"identity":False, "fit":True})
```

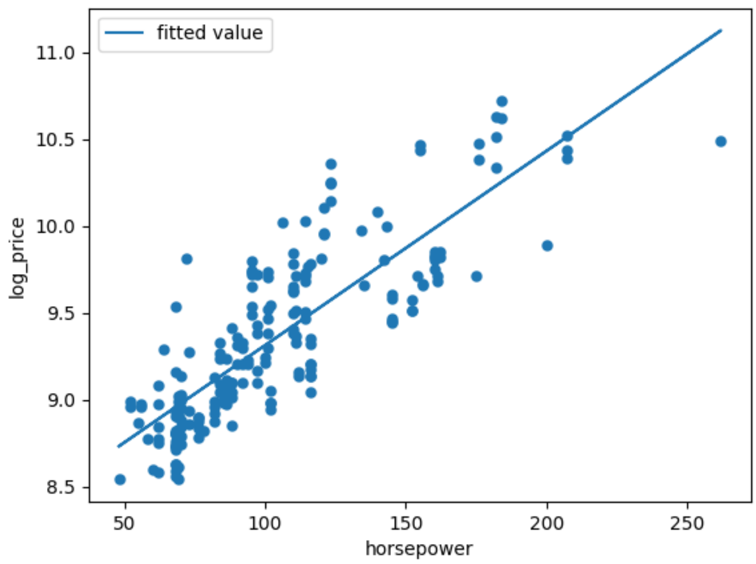
Result:

```
==============================================================================
Dep. Variable:              log_price   R-squared:                       0.694
Model:                            OLS   Adj. R-squared:                  0.693
Method:                 Least Squares   F-statistic:                     446.9
Date:                Mon, 08 Apr 2024   Prob (F-statistic):           1.47e-52
Time:                        18:05:09   Log-Likelihood:                -27.845
No. Observations:                 199   AIC:                             59.69
Df Residuals:                     197   BIC:                             66.28
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
horsepower     0.0112      0.001     21.140      0.000       0.010       0.012
cons           8.1949      0.058    140.772      0.000       8.080       8.310
==============================================================================
Omnibus:                       11.085   Durbin-Watson:                   0.663
Prob(Omnibus):                  0.004   Jarque-Bera (JB):               11.966
Skew:                           0.598   Prob(JB):                      0.00252
Kurtosis:                       2.897   Cond. No.                         323.
==============================================================================
```

Residual diagnostics:

From the regression result, the R-squared is 0.694, which means that the linear model can explain 70% of the dependent variable variance,sum of squared residual(SSR) only count for 30% of the dependent variable variance, showing the fit is good. Also, the p value of F-statistic is 0, meaning horsepower is a significant independent variable for price.

regression diagnostic plot:

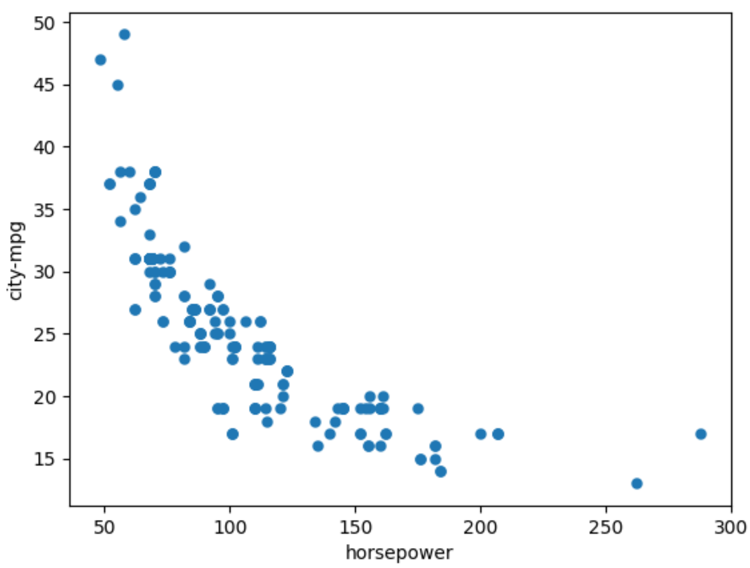


From the regression diagnostic plot we know:
The residuals seems to be closer to zero when fitted value is lower than 10, compared with when fitted value is more than 10. Also, it seems that when horsepower is larger than 125, the average residual is more far away from zero compared with when horsepower lower than 125. And seems the larger horsepower is, the bigger variance the residuals have.

3. Visulization between fuel efficiency and horsepower:

In [ ]:

```
plt.scatter(data['horsepower'],data['city-mpg'],linewidths=.05)
plt.xlabel("horsepower")
plt.ylabel("city-mpg")
plt.show()
```



Regress city-mpg on horsepower:

In [ ]:

```
mod3 = sm.OLS(data['city-mpg'],data[['horsepower','cons']],missing='drop')
res3=mod3.fit()
print(res3.summary())
```

```
Results:
==============================================================================
Dep. Variable:               city-mpg   R-squared:                       0.646
Model:                            OLS   Adj. R-squared:                  0.644
Method:                 Least Squares   F-statistic:                     366.5
Date:                Mon, 08 Apr 2024   Prob (F-statistic):           3.49e-47
Time:                        18:05:11   Log-Likelihood:                -564.37
No. Observations:                 203   AIC:                             1133.
Df Residuals:                     201   BIC:                             1139.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
horsepower    -0.1330      0.007    -19.144      0.000      -0.147      -0.119
cons          39.1031      0.775     50.482      0.000      37.576      40.630
==============================================================================
Omnibus:                       61.567   Durbin-Watson:                   1.404
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              184.768
Skew:                           1.254   Prob(JB):                     7.55e-41
Kurtosis:                       6.945   Cond. No.                         314.
==============================================================================
```

From the regression results we can see: the coefficient of horsepower is negative, meaning a negative relationship between city-mpg and horsepower, which is consistent with the plot. Also, horsepower is statistically significant, it's a significant independent variable for fuel efficiency. R2 is 64.6%, the model can still be improved by adding independent variables(second order variable for example).

Question 2

In [4]:

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import scipy.stats as stats
import statsmodels.api as sm
```

In [5]:

```
pip install seaborn
```

```
Requirement already satisfied: seaborn in /Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages
(0.13.2)
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in /Library/Frameworks/Python.framework/Versions/3.10/lib/python3.1
0/site-packages (from seaborn) (3.5.1)
Requirement already satisfied: pandas>=1.2 in /Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packag
es (from seaborn) (1.3.5)
Requirement already satisfied: numpy!=1.24.0,>=1.20 in /Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/si
te-packages (from seaborn) (1.21.4)
Requirement already satisfied: pyparsing>=2.2.1 in /Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-p
ackages (from matplotlib!=3.6.1,>=3.4->seaborn) (3.0.6)
Requirement already satisfied: fonttools>=4.22.0 in /Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-
packages (from matplotlib!=3.6.1,>=3.4->seaborn) (4.28.5)
Requirement already satisfied: cycler>=0.10 in /Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packa
ges (from matplotlib!=3.6.1,>=3.4->seaborn) (0.11.0)
Requirement already satisfied: packaging>=20.0 in /Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-pa
ckages (from matplotlib!=3.6.1,>=3.4->seaborn) (21.3)
Requirement already satisfied: pillow>=6.2.0 in /Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-pack
ages (from matplotlib!=3.6.1,>=3.4->seaborn) (9.0.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-
packages (from matplotlib!=3.6.1,>=3.4->seaborn) (1.3.2)
Requirement already satisfied: python-dateutil>=2.7 in /Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/si
te-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in /Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packa
ges (from pandas>=1.2->seaborn) (2021.3)
Requirement already satisfied: six>=1.5 in /Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages
(from python-dateutil>=2.7->matplotlib!=3.6.1,>=3.4->seaborn) (1.16.0)

[notice] A new release of pip available: 22.2.2 -> 24.0
[notice] To update, run: pip3 install --upgrade pip
Note: you may need to restart the kernel to use updated packages.
```

In [6]:

```
stock_data = pd.read_csv("/Users/hriday/Downloads/StockRetAcct_DT.csv")
stock_data
```

Out[6]:

	Unnamed: 0	FirmID	year	lnAnnRet	lnRf	MEwt	lnIssue	lnMom	lnME	lnProf	lnEP	lnInv	lnLever	lnROE	
0	1	6	1980	0.363631	0.078944	0.000281	0.031344	0.075355	12.581472	0.201767	0.146411	0.093626	0.696001	0.095294	0.084
1	2	6	1981	-0.290409	0.130199	0.000321	0.044213	0.512652	12.907996	0.215661	0.102555	0.087242	0.709843	0.082180	0.056
2	3	6	1982	0.186630	0.130703	0.000266	-0.068195	-0.220505	12.557775	0.184087	0.119548	0.111663	0.730972	0.079516	0.062
3	4	6	1983	0.489819	0.089830	0.000170	-0.071780	0.046218	12.561954	0.165531	0.115924	-0.033117	0.710885	0.055374	0.076
4	5	10	1991	-0.508005	0.061216	0.000033	0.115204	1.341053	11.565831	0.239788	0.023147	0.300051	0.418764	0.146828	0.374
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
70751	70752	20314	2010	0.200823	0.003067	0.000181	NaN	NaN	14.613427	NaN	NaN	NaN	NaN	NaN	I
70752	70753	20314	2011	0.071530	0.001880	0.000193	NaN	0.269093	14.923732	-0.891749	-0.063006	1.058996	0.623099	-0.556968	0.205
70753	70754	20314	2012	1.232889	0.002083	0.000210	0.215003	-0.080371	15.008085	-1.264313	-0.089005	0.614060	1.158263	-0.700480	0.251
70754	70755	20314	2013	0.804701	0.001553	0.000708	0.260489	1.104453	16.383282	-1.163863	-0.108056	0.445773	2.189972	-1.182673	0.220
70755	70756	20314	2014	0.111068	0.001175	0.001308	0.183487	0.609459	17.213655	0.036069	-0.004005	0.774370	1.277111	-0.104200	0.236

70756 rows × 17 columns

In [7]:

```
stock_data['Excess Return']=np.exp(stock_data.lnAnnRet) - np.exp(stock_data.lnRf)
stock_data['lnIssue'] = stock_data['lnIssue'] + np.random.normal(0,1/100,len(stock_data['lnIssue']))
```

In [8]:

```
stock_data['decile_portfolio']=pd.qcut(stock_data['lnIssue'], 10,labels=np.arange(1, 11, 1))
stock_data_cleaned = stock_data[stock_data['decile_portfolio'].notna()]
stock_data['decile_portfolio']
```

Out[8]:

```
0         5
1         6
2         2
3         2
4         7
       ...
70751    NaN
70752    NaN
70753      9
70754      9
70755      8
Name: decile_portfolio, Length: 70756, dtype: category
Categories (10, int64): [1 < 2 < 3 < 4 ... 7 < 8 < 9 < 10]
```

In [9]:

```
decile_portfolio_data = pd.DataFrame(stock_data_cleaned.groupby(['year','decile_portfolio']).mean()['Excess Return']).reset_index(level =
# We have the mean excess return for each desile portfolio for each year now find the mean across the whole timeframe
decile_portfolio_final = decile_portfolio_data.groupby('decile_portfolio').mean()['Excess Return']
```

In [10]:

```
#1st Question:
decile_portfolio_final
```

Out[10]:

```
decile_portfolio
1     0.123128
2     0.107318
3     0.095070
4     0.086524
5     0.099442
6     0.111739
7     0.109371
8     0.093081
9     0.094850
10    0.050659
Name: Excess Return, dtype: float64
```

In [11]:

```
#2nd Question:
# Plot the portfolios
plt.figure()
ax=sns.regplot (x      = np.arange(0,10),
                y      = decile_portfolio_final,
                scatter_kws={"color": "black"},
                line_kws={"color": "purple"},
                fit_reg=True,
                order = 5
                )
ax.set(ylim=(0.07,0.15),title='Issuance bins vs. Excess Returns')
```

```
g: Polyfit may be poorly conditioned
  boot_dist.append(f(*sample, **func_kwargs))
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages/seaborn/algorithms.py:100: RankWarnin
g: Polyfit may be poorly conditioned
  boot_dist.append(f(*sample, **func_kwargs))
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages/seaborn/algorithms.py:100: RankWarnin
g: Polyfit may be poorly conditioned
  boot_dist.append(f(*sample, **func_kwargs))
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages/seaborn/algorithms.py:100: RankWarnin
g: Polyfit may be poorly conditioned
  boot_dist.append(f(*sample, **func_kwargs))
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages/seaborn/algorithms.py:100: RankWarnin
g: Polyfit may be poorly conditioned
  boot_dist.append(f(*sample, **func_kwargs))
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages/seaborn/algorithms.py:100: RankWarnin
g: Polyfit may be poorly conditioned
  boot_dist.append(f(*sample, **func_kwargs))
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages/seaborn/algorithms.py:100: RankWarnin
g: Polyfit may be poorly conditioned
  boot_dist.append(f(*sample, **func_kwargs))
```

The observed pattern is nonlinear, demonstrating that low issuance tends to result in higher returns, while higher issuance leads to lower returns, as anticipated. This observation holds significant implications. When a company issues new shares, it diminishes the ownership percentage of current shareholders. This dilution can directly impact earnings per share (EPS), resulting in reduced stock prices.

In [12]:

```
#3rd Question:
conditions = [(stock_data['decile_portfolio'] == 0),(stock_data['decile_portfolio'] == 9)]
values = [-1, 1]

stock_data['transformed_issuance'] = np.select(conditions, values, default=0)
```

In [13]:

```
stock_data_cleaned_transform = stock_data[stock_data['decile_portfolio'].notna()]
```

In [16]:

```
# Step 1: Run time-series regressions
ts_results = []
for date, group in stock_data_cleaned_transform.groupby('year'):
    X = sm.add_constant(group[['transformed_issuance']])
    y = group['Excess Return']
    model = sm.OLS(y, X).fit()
    ts_results.append({'date': date, 'const': model.params[0], 'coeff_issuance': model.params[1],'coeff_pvalue':round(model.pvalues[1],3)

ts_results_df = pd.DataFrame(ts_results)
```

In [17]:

```
lamda_hat = ts_results_df['coeff_issuance'].mean()
tstat = lamda_hat/(ts_results_df['coeff_issuance'].std()/np.sqrt(len(ts_results_df)))
p_value = stats.t.sf(abs(tstat), len(ts_results_df)) * 2
```

In [18]:

```
print(" Lamda Hat for FAMA-Macbeth Model :",lamda_hat)
print(" T-Statistic : ",tstat)
print(" P-Value : ",p_value)
```

```
 Lamda Hat for FAMA-Macbeth Model : 0.0001885210692128678
 T-Statistic :  0.011866050004478322
 P-Value :  0.990599865425727
```

In this context, the p-value holds statistical significance. The negative coefficient coupled with its statistical significance indicates that stocks exhibiting extreme issuance characteristics, such as those in Decile 10, typically yield lower expected returns compared to Decile 1 stocks. Consequently, a strategy may involve shorting Decile 10 stocks while going long on Decile 1 stocks. No position is taken on the remaining 80% of stocks falling within other deciles.

Question 3

In [3]:

```
import pandas as pd

# Load the dataset
stock_data = pd.read_csv("StockRetAcct_DT.csv")

stock_data.head()
```

Out[3]:

	Unnamed: 0	FirmID	year	lnAnnRet	lnRf	MEwt	lnIssue	lnMom	lnME	lnProf	lnEP	lnInv	lnLever	lnROE	rv
0	1	6	1980	0.363631	0.078944	0.000281	0.031344	0.075355	12.581472	0.201767	0.146411	0.093626	0.696001	0.095294	0.084134
1	2	6	1981	-0.290409	0.130199	0.000321	0.044213	0.512652	12.907996	0.215661	0.102555	0.087242	0.709843	0.082180	0.056381
2	3	6	1982	0.186630	0.130703	0.000266	-0.068195	-0.220505	12.557775	0.184087	0.119548	0.111663	0.730972	0.079516	0.062072
3	4	6	1983	0.489819	0.089830	0.000170	-0.071780	0.046218	12.561954	0.165531	0.115924	-0.033117	0.710885	0.055374	0.076955
4	5	10	1991	-0.508005	0.061216	0.000033	0.115204	1.341053	11.565831	0.239788	0.023147	0.300051	0.418764	0.146828	0.374368

In [4]:

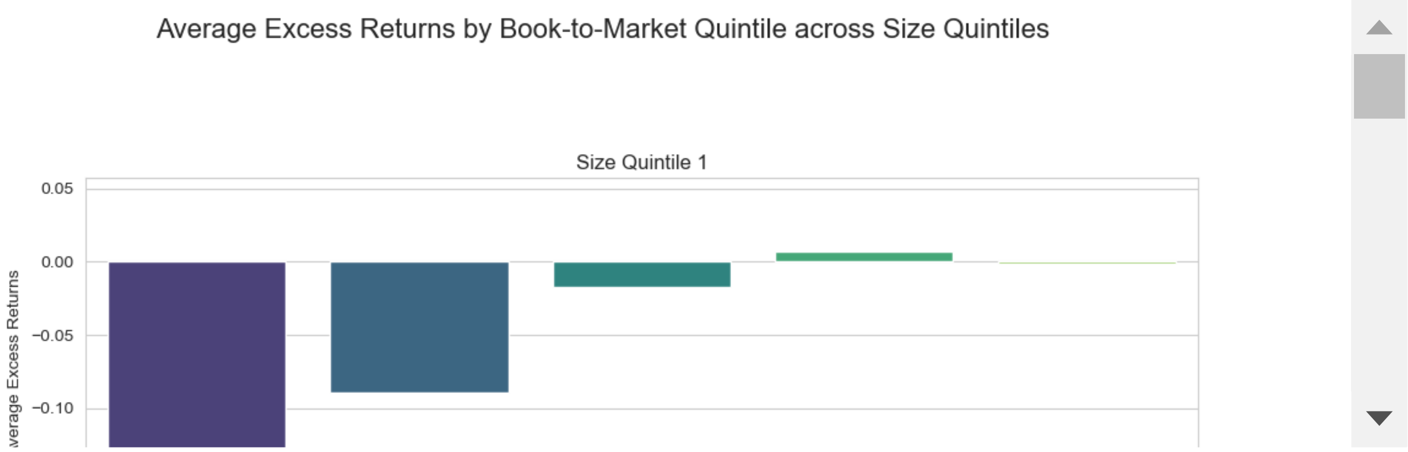
```
# Calculating excess returns
stock_data['ExcessReturns'] = stock_data['lnAnnRet'] - stock_data['lnRf']

# Creating quintiles within each year for lnBM and lnME
stock_data['BMQuintile'] = stock_data.groupby('year')['lnBM'].transform(lambda x: pd.qcut(x, 5, labels=False) + 1)
stock_data['MEQuintile'] = stock_data.groupby('year')['lnME'].transform(lambda x: pd.qcut(x, 5, labels=False) + 1)

# Dropping rows where quintiles could not be calculated (due to NaN values in lnBM or lnME)
stock_data_cleaned = stock_data.dropna(subset=['BMQuintile', 'MEQuintile'])

# Preparing the data for plotting
plot_data = []

for size_q in range(1, 6):
    size_data = stock_data_cleaned[stock_data_cleaned['MEQuintile'] == size_q]
    bm_return_avg = size_data.groupby('BMQuintile')['ExcessReturns'].mean()
    plot_data.append(bm_return_avg.reset_index())

plot_data
```

Out[4]:

```
[   BMQuintile  ExcessReturns
 0         1.0      -0.144999
 1         2.0      -0.088989
 2         3.0      -0.017390
 3         4.0       0.007045
 4         5.0      -0.001545,
    BMQuintile  ExcessReturns
 0         1.0      -0.137862
 1         2.0      -0.052131
 2         3.0      -0.001154
 3         4.0       0.027832
 4         5.0      -0.025816,
    BMQuintile  ExcessReturns
 0         1.0      -0.102001
 1         2.0      -0.029774
 2         3.0       0.010626
 3         4.0       0.030003
 4         5.0       0.014907,
    BMQuintile  ExcessReturns
 0         1.0      -0.042966
 1         2.0      -0.008414
 2         3.0       0.021801
 3         4.0       0.036108
 4         5.0       0.029016,
    BMQuintile  ExcessReturns
 0         1.0      -0.026501
 1         2.0       0.018100
 2         3.0       0.031934
 3         4.0       0.045096
 4         5.0       0.034601]
```

In [5]:

```
import matplotlib.pyplot as plt
import seaborn as sns

# Set the style of seaborn
sns.set_style('whitegrid')

# Plotting
fig, axes = plt.subplots(5, 1, figsize=(10, 20), sharex=True)
fig.suptitle('Average Excess Returns by Book-to-Market Quintile across Size Quintiles', fontsize=16)

for i, ax in enumerate(axes.flat):
    sns.barplot(x='BMQuintile', y='ExcessReturns', data=plot_data[i], ax=ax, palette='viridis')
    ax.set_title(f'Size Quintile {i+1}')
    ax.set_xlabel('Book-to-Market Quintile')
    ax.set_ylabel('Average Excess Returns')
    ax.set_ylim([plot_data[i]['ExcessReturns'].min() - 0.05, plot_data[i]['ExcessReturns'].max() + 0.05])

plt.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.show()
```



From these plots, we observe a general trend that within each size quintile, the relationship between book-to-market quintiles and average excess returns does not strictly follow a linear pattern. While in some size quintiles, there's a noticeable increase in average excess returns as we move to higher book-to-market quintiles, the pattern varies, indicating that the assumption of conditional linearity may not fully capture the relationship between expected returns, book-to-market ratio, and size.

Considering these visual insights, it might be worthwhile to explore models that can accommodate non-linear relationships or interactions between variables in more complex ways, such as quadratic terms or using machine learning models