Analysis of Regular Freight Cost

Scraping the oil price data from the website and define the diesel price function

In [1]:

```
import time
import numpy as np
import pandas as pd
import seaborn as sns
from bs4 import BeautifulSoup
from selenium import webdriver

import operator
import scipy as sp
from numpy import random
from scipy.stats import norm, skew
import matplotlib.pyplot as plt

from sklearn import linear_model
```

In [2]:

```
chrome_obj = webdriver.Chrome()
chrome_obj.get(url='https://www.zuixinyoujia.com/guonei/')
time.sleep(3)#间隔3秒

soup = BeautifulSoup(chrome_obj.page_source, "html.parser")
tables = soup.find_all('table')
oil_price = pd.read_html(str(tables[1]),header=0,flavor='bs4')[0]
print(oil_price.shape)
```

```
(32, 5)
```

In [3]:

```
oil_price.head()
```

Out[3]:

	地区	92号汽油	95号汽油	98号汽油	0号柴油
0	北京	7.79	8.29	9.27	7.50
1	上海	7.75	8.25	9.25	7.43
2	天津	7.78	8.22	9.50	7.45
3	重庆	7.85	8.30	9.34	7.52
4	福建	7.75	8.28	9.06	7.44

In [4]:

```
def diesel_price(location=None):
    die_price = dict(zip(list(oil_price.iloc[:,0]),list(oil_price.iloc[:,4])))
    return die_price.get(location,'')
```

calculate the cost of regular freight

create calculation function

In [5]:

```
#location 位置，mileage 单边里程，cbm 方位
def regular_freight_cost(location=None,mileage=None,cbm=None,**datas):
    datas = {
        12:[0.13,7000,500,700,60000],
        16:[0.14,7000,500,700,65000],
        20:[0.15,8000,500,800,90000],
        30:[0.18,8000,500,800,120000],
        35:[0.19,10000,600,1550,145000],
        45:[0.22,14000,600,2000,155000],
        56:[0.25,17000,800,2933,255500],
    }
    #oil cost
    diesel_price = dict(zip(list(oil_price.iloc[:,0]),list(oil_price.iloc[:,4])))[locati
    oil_cost = round(datas.get(cbm,'')[0]*diesel_price*mileage*2,2)

    #road toll
    road_toll = round(
        pd.read_excel(
            "C:\\Users\\Kieran\\Downloads\\data.xlsx", index_col=0,header=0
        ).loc[location,cbm]*mileage*2,2)

    #salary
    salaries = {
        4000:[50,101],4500:[101,201],4700:[201,301],5000:[301,401],5500:[401,501],
        6000:[501,601],6250:[601,701],6500:[701,801],6750:[801,901],7250:[901,1001],
        7750:[1001,1201],8750:[1201,1401],9750:[1401,1501],10750:[1501,2000],
    }
    for sala, mile in salaries.items():
        minimum = mile[0]
        maximum = mile[1]
        if minimum <= mileage*2 <= maximum:
            salary = round(sala/30,2)

    #insurance
    insurance = round(datas[cbm][1]/360,2)

    #maintenance cost
    maintenance_cost = round(datas[cbm][2]/10000*mileage*2,2)

    #tyre cost
    tyre_cost = round(datas[cbm][3]*6/100000*mileage*2,2)

    #profit
    months = {
        17:[901,3001],
        18:[801,901],
        19:[701,801],
        20:[601,701],
        21:[501,601],
        22:[401,501],
        23:[301,401],
        24:[0,301],
    }
    for month, mile in months.items():
        minimum = mile[0]
        maximum = mile[1]
        if minimum <= mileage*2 <= maximum:
            profit = round(datas[cbm][4]/month/30,2)

    total_cost =  round(sum([oil_cost,road_toll,salary,insurance,maintenance_cost,tyre_c

    return [location,mileage,cbm,oil_cost,road_toll,salary,insurance,maintenance_cost,ty
```

In [6]:

```
[random.choice(list(oil_price.iloc[:,0])),random.randint(50,1000),random.choice([12,16,2
```

Out[6]:
```
['宁夏', 275, 45]
```

In [7]:
```python
def get_origin_data():
    origin_data = []
    random.seed(5)
    while operator.lt(len(origin_data),1000):
        i = [random.choice(list(oil_price.iloc[:,0])),
             #lower, upper = mu - 3 * sigma, mu + 3 * sigma  # 截断在[μ-3σ, μ+3σ]
             #stats.truncnorm.rvs((lower - mu) / sigma, (upper - mu) / sigma, loc=mu, sc
             round(sp.stats.truncnorm.rvs((0.3 - 3) / 0.9, (5.7 - 3) / 0.9, loc=3, scale
             random.choice([12,16,20,30,35,45,56])
            ]
        origin_data.append(i)
    return origin_data
```

In [8]:
```python
get_origin_data()[0:3]
```

Out[8]:
```
[['重庆', 263, 45], ['广东', 449, 16], ['贵州', 300, 56]]
```

In [9]:
```python
len(get_origin_data())
```

Out[9]:
```
1000
```

In [10]:
```python
def get_total_cost():
    result = []
    origin_data = get_origin_data()
    for i in origin_data:
        total_cost = regular_freight_cost(i[0],i[1],i[2])
        result.append(total_cost)
    return result
```

In [11]:
```python
df = pd.DataFrame(
    get_total_cost(),
    columns=[
        'location','mileage','cbm','oil_cost','road_toll','salary','insurance','maintena
    ]
)
df.head()
```

Out[11]:

	location	mileage	cbm	oil_cost	road_toll	salary	insurance	maintenance_cost	tyre_cost	profit	t
0	重庆	263	45	870.21	347.16	200.00	38.89	31.56	63.12	246.03	
1	广东	449	16	937.87	215.52	225.00	19.44	44.90	37.72	120.37	
2	贵州	300	56	1134.00	648.00	200.00	47.22	48.00	105.59	405.56	
3	黑龙江	445	12	845.77	178.00	225.00	19.44	44.50	37.38	111.11	
4	广西	848	16	1783.17	407.04	358.33	19.44	84.80	71.23	127.45	

In [12]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 12 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   location          1000 non-null   object 
 1   mileage           1000 non-null   int64  
 2   cbm               1000 non-null   int32  
 3   oil_cost          1000 non-null   float64
 4   road_toll         1000 non-null   float64
 5   salary            1000 non-null   float64
 6   insurance         1000 non-null   float64
 7   maintenance_cost  1000 non-null   float64
 8   tyre_cost         1000 non-null   float64
 9   profit            1000 non-null   float64
 10  total_cost        1000 non-null   float64
 11  diesel_price      1000 non-null   float64
dtypes: float64(9), int32(1), int64(1), object(1)
memory usage: 90.0+ KB
```

In [13]:
```
df.describe().T
```

Out[13]:

	count	mean	std	min	25%	50%	75%	max
mileage	1000.0	506.69600	152.594753	75.00	401.000	503.000	609.2500	931.00
cbm	1000.0	30.28800	14.918768	12.00	16.000	30.000	45.0000	56.00
oil_cost	1000.0	1349.44482	511.309631	168.75	981.600	1262.330	1676.2875	3084.35
road_toll	1000.0	444.08302	295.522902	34.80	227.580	346.920	585.5800	1879.44
salary	1000.0	253.89343	44.240973	150.00	225.000	258.330	291.6700	358.33
insurance	1000.0	27.93702	10.061724	19.44	19.440	22.220	38.8900	47.22
maintenance_cost	1000.0	57.72332	20.424924	7.50	43.800	55.400	68.8250	133.92
tyre_cost	1000.0	80.94215	55.799702	7.20	42.505	56.740	108.6600	294.59
profit	1000.0	235.99300	119.731756	83.33	127.450	230.160	284.3100	500.98
total_cost	1000.0	2450.01676	978.220175	472.33	1739.855	2232.845	3025.9600	6026.83
diesel_price	1000.0	7.44953	0.121488	7.25	7.370	7.440	7.5100	7.99

异常值处理(省略)

In [14]:
```
df.loc[:,['location','oil_cost','road_toll','salary','insurance','maintenance_cost','tyr
```

Out[14]:

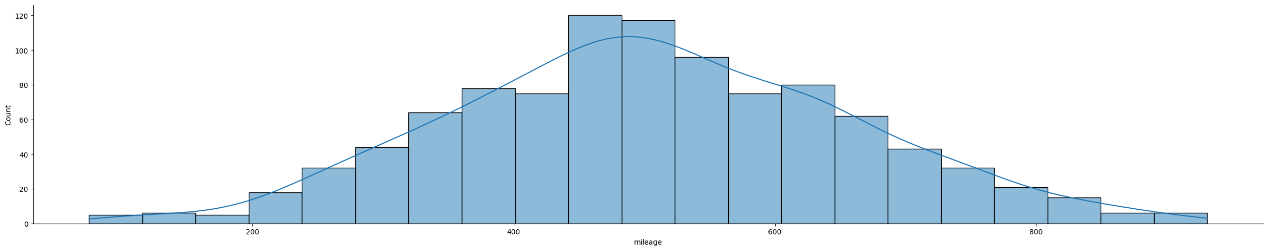
	oil_cost	road_toll	salary	insurance	maintenance_cost	tyre_cost	profit
location							
上海	1443.18	503.37	254.72	30.92	61.22	95.41	275.61
云南	1319.07	411.63	258.33	27.39	58.58	78.92	219.77
内蒙古	1311.55	436.32	251.43	28.17	56.42	79.77	231.32
北京	1359.72	453.52	252.78	29.98	58.11	86.43	253.42
吉林	1243.08	310.28	250.40	26.51	54.67	73.18	213.31

normal distribution test

In [15]:

```
sns.displot(data = df, x='mileage', kde=True, aspect=5 )
print('Kurtosis: %f' % df['mileage'].kurt())
print('Skewness: %f' % df['mileage'].skew())
```

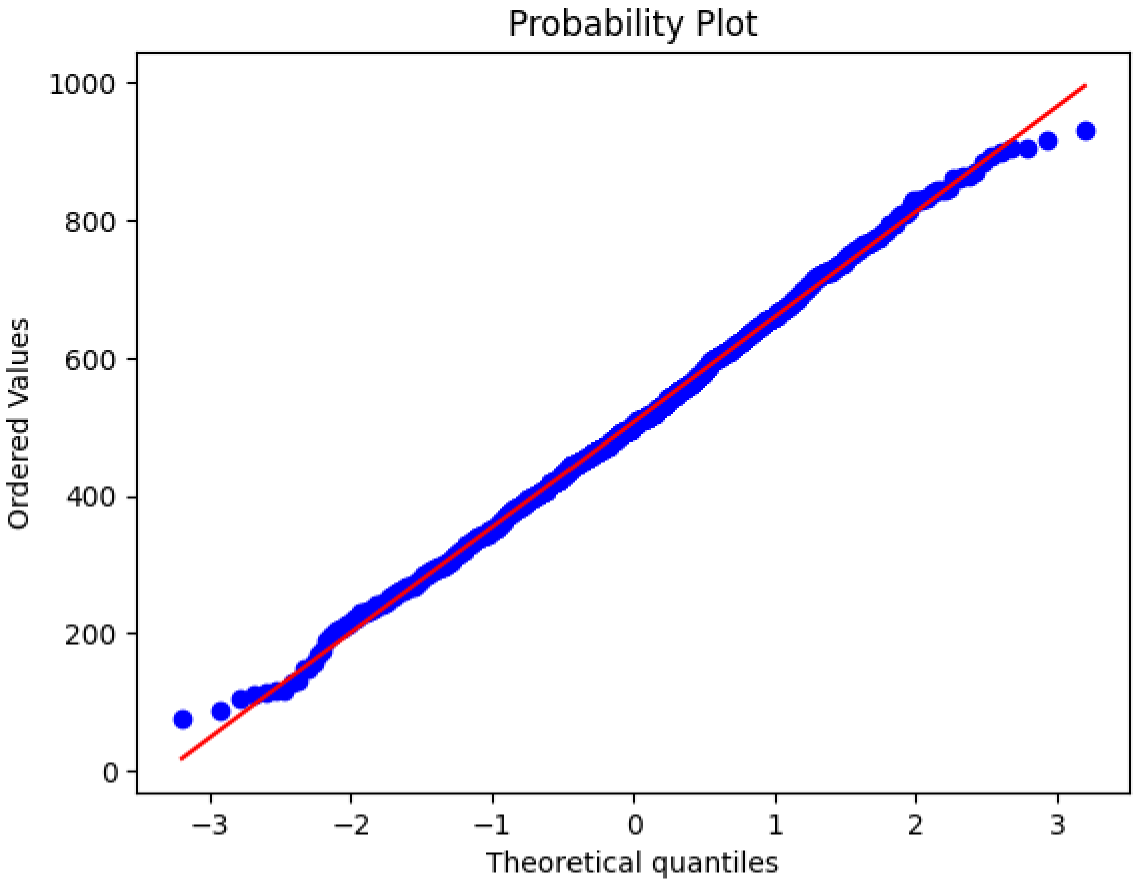
```
Kurtosis: -0.190341
Skewness: 0.070603
```



calculate quantiles for a probability plot, and optionally show the plot.

In [16]:

```
fig = plt.figure()
res = sp.stats.probplot(df['mileage'], plot=plt)
plt.show()
```



Exploratory Data Analysis with Data Visualization

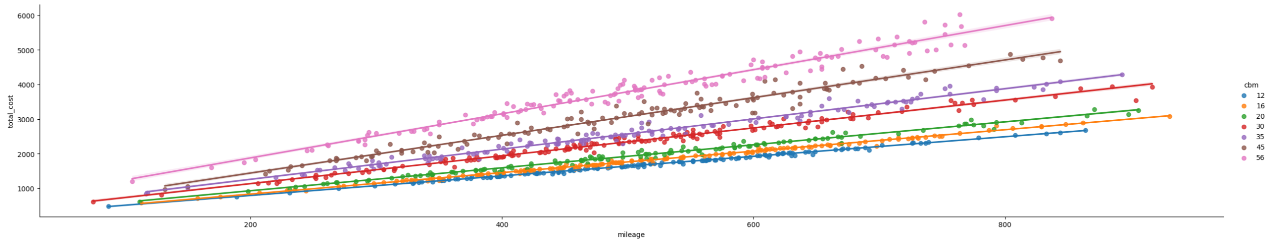
visualize the relationship between mileage and total cost

In [17]:

```
sns.lmplot(data= df, x="mileage", y="total_cost", hue=str("cbm"), aspect=5)
```

Out[17]:

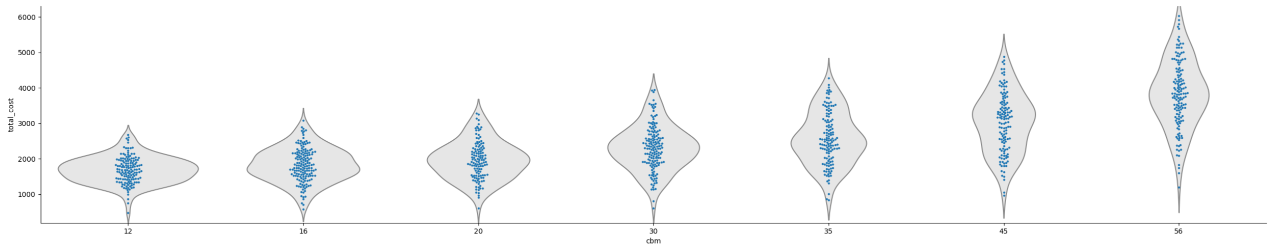
```
<seaborn.axisgrid.FacetGrid at 0x29ca6150d90>
```



visualize the relationship between CBM and total cost

In [18]:
```
sns.catplot(data = df, x="cbm", y="total_cost", kind="violin", color=".9", inner=None, a
sns.swarmplot(data=df, x="cbm", y="total_cost", size=3)
```

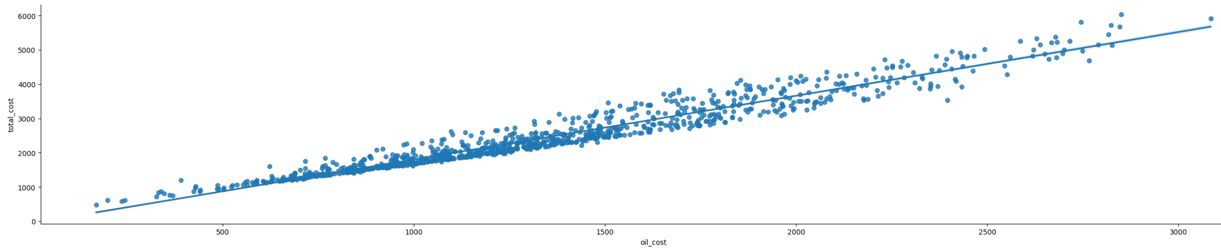
Out[18]:
```
<AxesSubplot: xlabel='cbm', ylabel='total_cost'>
```



visualize the relationship between oil cost and total cost

In [19]:
```
sns.lmplot(data= df, x="oil_cost", y="total_cost", aspect = 5)
```

Out[19]:
```
<seaborn.axisgrid.FacetGrid at 0x29ca63a4650>
```



Multiple Linear Regression

select some features used for regression.

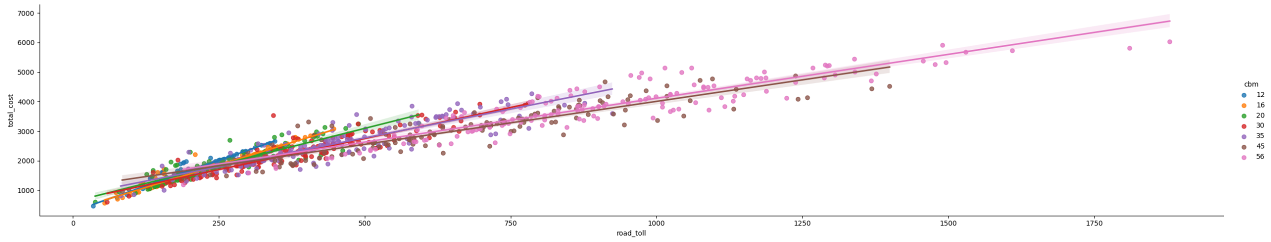
In [20]:
```
cdf = df[['mileage','cbm','diesel_price','road_toll','salary','insurance','maintenance_c
cdf.head()
```

Out[20]:

	mileage	cbm	diesel_price	road_toll	salary	insurance	maintenance_cost	tyre_cost	profit	total_c
0	263	45	7.52	347.16	200.00	38.89	31.56	63.12	246.03	1796
1	449	16	7.46	215.52	225.00	19.44	44.90	37.72	120.37	1600
2	300	56	7.56	648.00	200.00	47.22	48.00	105.59	405.56	2588
3	445	12	7.31	178.00	225.00	19.44	44.50	37.38	111.11	1461
4	848	16	7.51	407.04	358.33	19.44	84.80	71.23	127.45	2851

In [21]:
```
sns.lmplot(data = cdf,x="road_toll", y="total_cost", hue=str("cbm"),aspect = 5)
```

```
Out[21]: <seaborn.axisgrid.FacetGrid at 0x29ca6036b10>
```

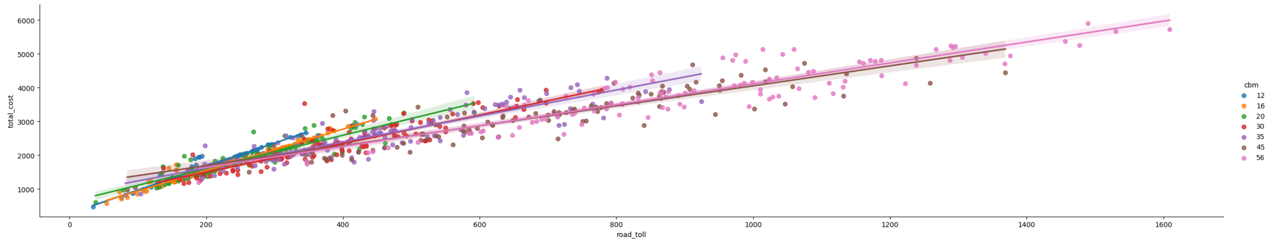


creating train and test dataset

```
In [22]: msk = np.random.rand(len(df)) < 0.8
         train = cdf[msk]
         test = cdf[~msk]
```

```
In [23]: sns.lmplot(data = train,x="road_toll", y="total_cost", hue=str("cbm"),aspect = 5)
```

```
Out[23]: <seaborn.axisgrid.FacetGrid at 0x29ca6165490>
```



multiple regression model

```
In [24]: regr = linear_model.LinearRegression()
         x = np.asanyarray(train[['mileage','cbm','diesel_price','road_toll','salary','insurance'
         y = np.asanyarray(train[['total_cost']])
         regr.fit (x, y)
         # The coefficients
         print ('Coefficients: ', regr.coef_)
```

```
Coefficients:  [[   3.30058353   12.65139818 146.63232009    1.15036136    1.47969607
   -20.61546602 -19.34267008    9.42210967    1.83258147]]
```

prediction

```
In [25]: y_hat= regr.predict(test[['mileage','cbm','diesel_price','road_toll','salary','insurance
         x = np.asanyarray(test[['mileage','cbm','diesel_price','road_toll','salary','insurance',
         y = np.asanyarray(test[['total_cost']])
         print("Residual sum of squares: %.2f"
               % np.mean((y_hat - y) ** 2))

         # Explained variance score: 1 is perfect prediction
         print('Variance score: %.2f' % regr.score(x, y))
```

```
Residual sum of squares: 1329.91
Variance score: 1.00
```

```
C:\Users\Kieran\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\base.p
y:402: UserWarning: X has feature names, but LinearRegression was fitted without feature
names
  warnings.warn(
```

model2

In [26]:

```
regr = linear_model.LinearRegression()
x = np.asanyarray(train[['mileage','cbm','diesel_price','road_toll']])
y = np.asanyarray(train[['total_cost']])
regr.fit (x, y)
print ('Coefficients: ', regr.coef_)
y_= regr.predict(test[['mileage','cbm','diesel_price','road_toll']])
x = np.asanyarray(test[['mileage','cbm','diesel_price','road_toll']])
y = np.asanyarray(test[['total_cost']])
print("Residual sum of squares: %.2f"% np.mean((y_ - y) ** 2))
print('Variance score: %.2f' % regr.score(x, y))
```

```
Coefficients:  [[ 2.8241511  23.54219737 90.03861889  1.59239958]]
Residual sum of squares: 9891.49
Variance score: 0.99
```

```
C:\Users\Kieran\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\base.p
y:402: UserWarning: X has feature names, but LinearRegression was fitted without feature
names
  warnings.warn(
```

In [ ]: