Pandas 資料分析 (4)

張家瑋 副教授
國立臺中科技大學資訊工程系

資料轉換與缺失值處理

```
import pandas as pd
import numpy as np
pd.set_option('max_columns', 4, 'max_rows', 10, 'max_colwidth', 12)

fueleco = pd.read_csv('data/vehicles.csv.zip')
fueleco
```

```
c:\users\test\appdata\local\programs\python\python37\lib\site-packag
umns (70,71,72,73,74,76,79) have mixed types.Specify dtype option on
  exec(code_obj, self.user_global_ns, self.user_ns)
```

	barrels08	barrelsA08	...	phevHwy	phevComb
0	15.695714	0.0	...	0	0
1	29.964545	0.0	...	0	0
2	12.207778	0.0	...	0	0
3	29.964545	0.0	...	0	0
4	17.347895	0.0	...	0	0
...	...	...	...	...	...
39096	14.982273	0.0	...	0	0
39097	14.330870	0.0	...	0	0
39098	15.695714	0.0	...	0	0
39099	15.695714	0.0	...	0	0
39100	18.311667	0.0	...	0	0

39101 rows × 83 columns

資料轉換與缺失值處理

```
fueleco.select_dtypes(object).columns
```

```
Index(['drive', 'eng_dscr', 'fuelType', 'fuelType1', 'make', 'model',
       'mpgData', 'trany', 'VClass', 'guzzler', 'trans_dscr', 'tCharger',
       'sCharger', 'atvType', 'fuelType2', 'rangeA', 'evMotor', 'mfrCode',
       'c240Dscr', 'c240bDscr', 'createdOn', 'modifiedOn', 'startStop'],
      dtype='object')
```

```
fueleco.drive.nunique()
```

```
7
```

```
fueleco.drive.sample(5, random_state=42)
```

```
4217     4-Wheel ...
1736     4-Wheel ...
36029    Rear-Whe...
37631    Front-Wh...
1668     Rear-Whe...
Name: drive, dtype: object
```

```
fueleco.drive.isna().sum()
```

```
1189
```

```
fueleco.drive.isna().mean() * 100
```

```
3.0408429451932175
```

資料轉換與缺失值處理

```
fueleco.drive.value_counts()
```

```
Front-Wheel Drive                13653
Rear-Wheel Drive                 13284
4-Wheel or All-Wheel Drive        6648
All-Wheel Drive                   2401
4-Wheel Drive                     1221
2-Wheel Drive                      507
Part-time 4-Wheel Drive            198
Name: drive, dtype: int64
```

```
fueleco.drive.value_counts(dropna=False)
```

```
Front-Wheel Drive                13653
Rear-Wheel Drive                 13284
4-Wheel or All-Wheel Drive        6648
All-Wheel Drive                   2401
4-Wheel Drive                     1221
NaN                               1189
2-Wheel Drive                      507
Part-time 4-Wheel Drive            198
Name: drive, dtype: int64
```

#保留前六項分類

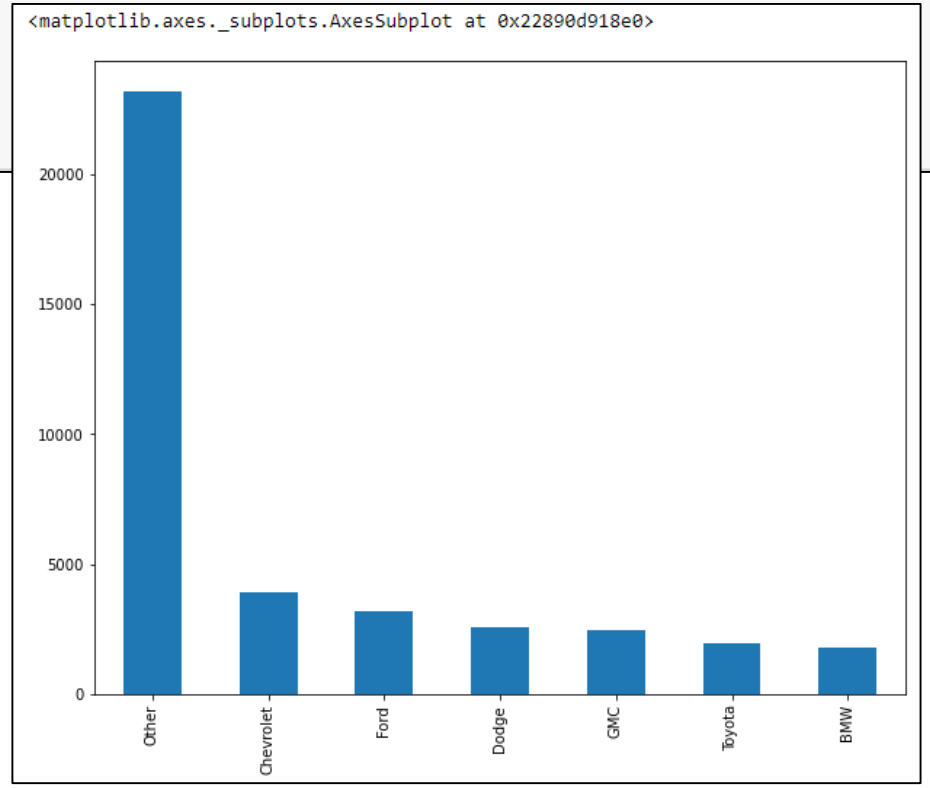
```
top_n = fueleco.make.value_counts().index[:6]
(fueleco
   .assign(make=fueleco.make.where(
              fueleco.make.isin(top_n), 'Other'))
   .make
   .value_counts()
)
```

#使用 where()更改 make 欄的內容：若不是前六項分類則改為 Other

```
Other        23211
Chevrolet     3900
Ford          3208
Dodge         2557
GMC           2442
Toyota        1976
BMW           1807
Name: make, dtype: int64
```

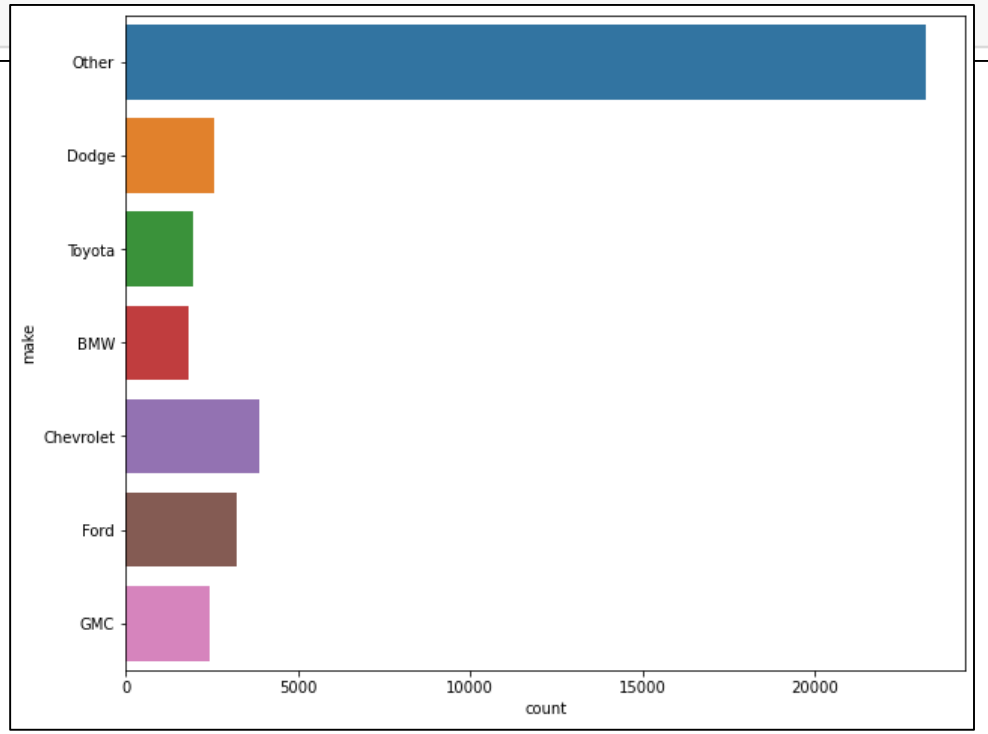
資料轉換與缺失值處理

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots(figsize=(10, 8))
top_n = fueleco.make.value_counts().index[:6]
(fueleco
   .assign(make=fueleco.make.where(
               fueleco.make.isin(top_n),
               'Other'))
   .make
   .value_counts()
   .plot.bar(ax=ax)
)
```



資料轉換與缺失值處理

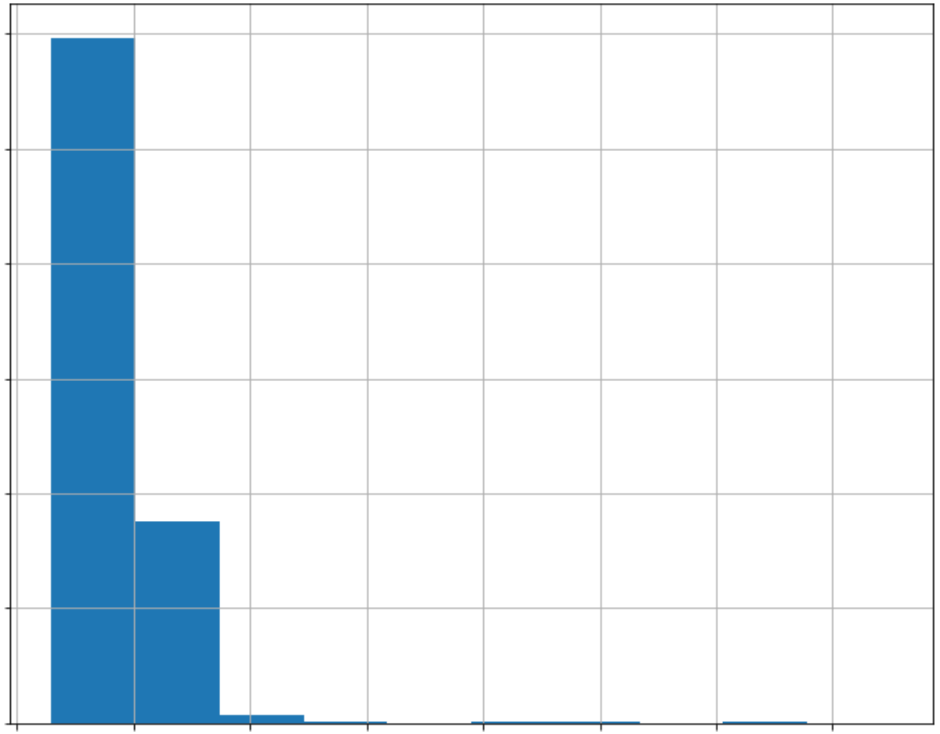
```
import seaborn as sns
fig, ax = plt.subplots(figsize=(10, 8))
top_n = fueleco.make.value_counts().index[:6]
sns.countplot(y='make',
    data= (fueleco
        .assign(make=fueleco.make.where(
                fueleco.make.isin(top_n),
                'Other'))
  )
)
```



檢視連續資料的分布狀況

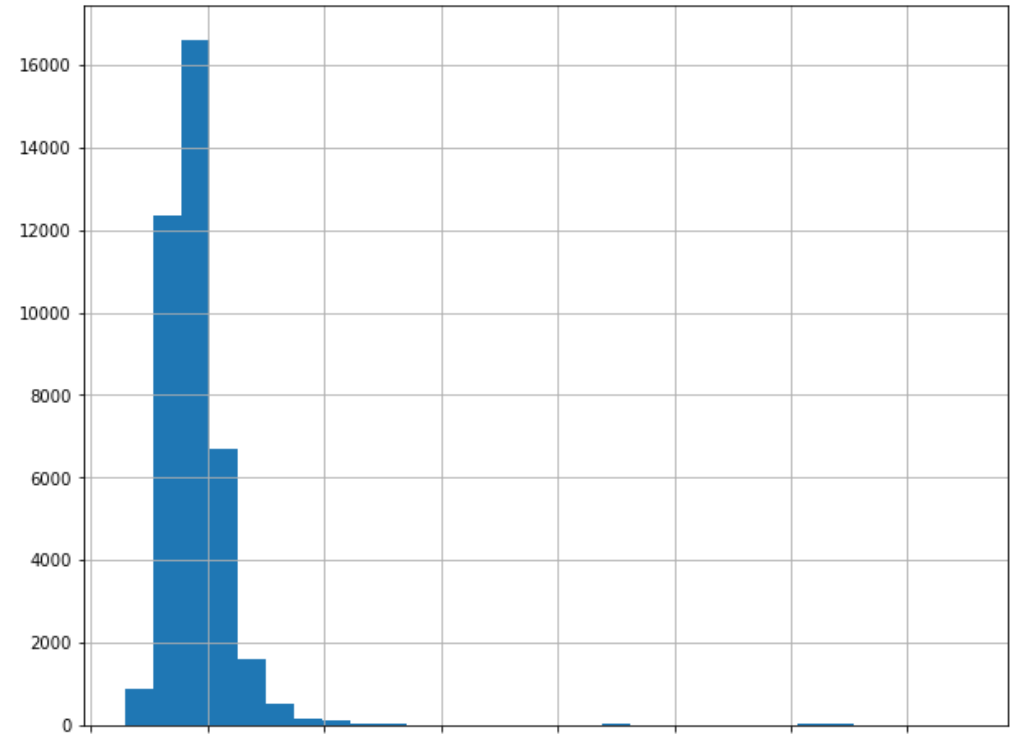
```
import matplotlib.pyplot as plt
fig, ax = plt.subplots(figsize=(10, 8))
fueleco.city08.hist(ax=ax)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x228968ec3d0>
```




```
import matplotlib.pyplot as plt
fig, ax = plt.subplots(figsize=(10, 8))
fueleco.city08.hist(ax=ax, bins=30)
```

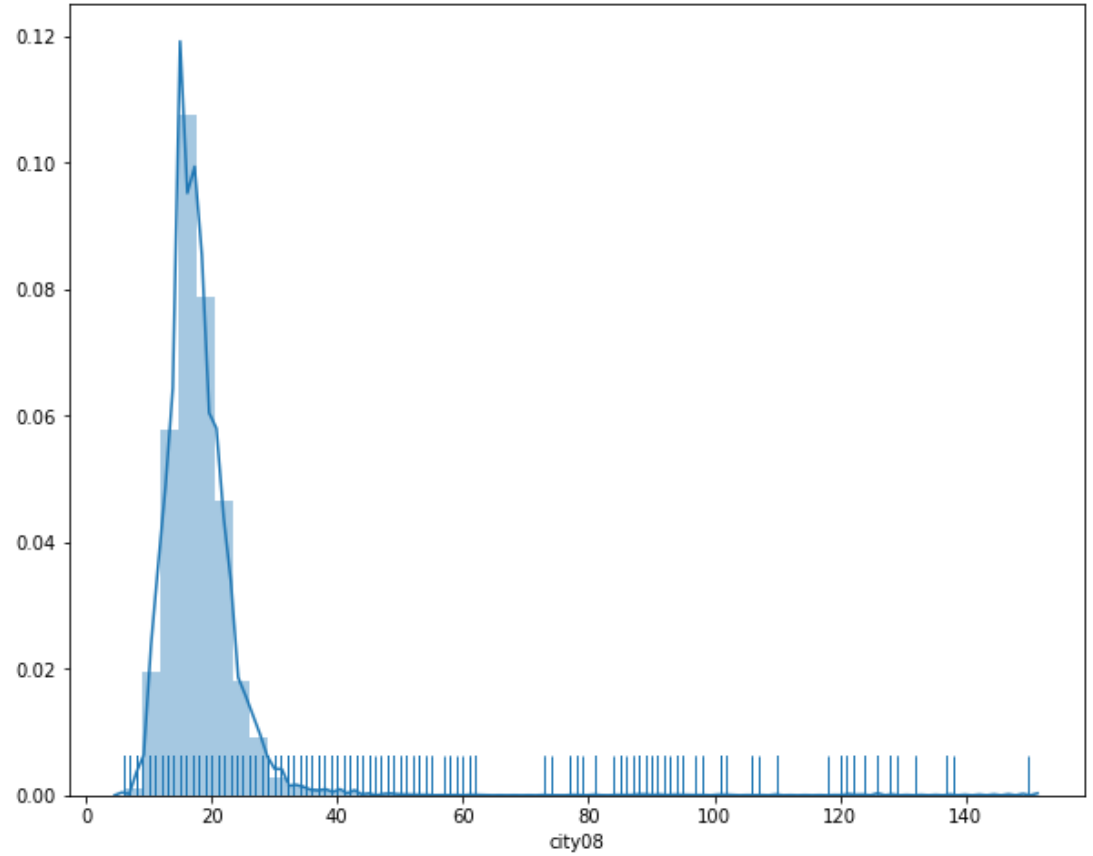
```
<matplotlib.axes._subplots.AxesSubplot at 0x2289579bcd0>
```



檢視連續資料的分布狀況

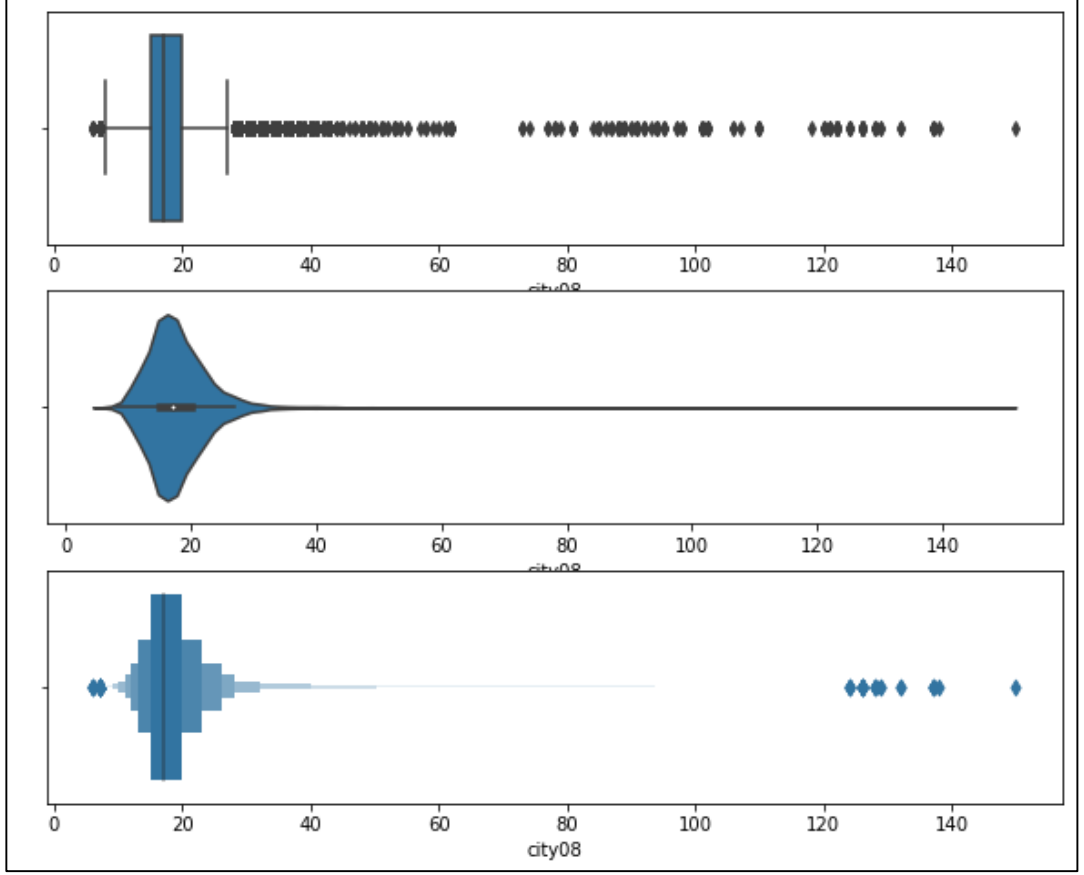
```
fig, ax = plt.subplots(figsize=(10, 8))
sns.distplot(fueleco.city08, rug=True, ax=ax)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x228950910d0>
```




```
fig, axs = plt.subplots(nrows=3, figsize=(10, 8))
sns.boxplot(fueleco.city08, ax=axs[0])
sns.violinplot(fueleco.city08, ax=axs[1])
sns.boxenplot(fueleco.city08, ax=axs[2])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x22897d8f700>
```



檢視連續資料的分布狀況

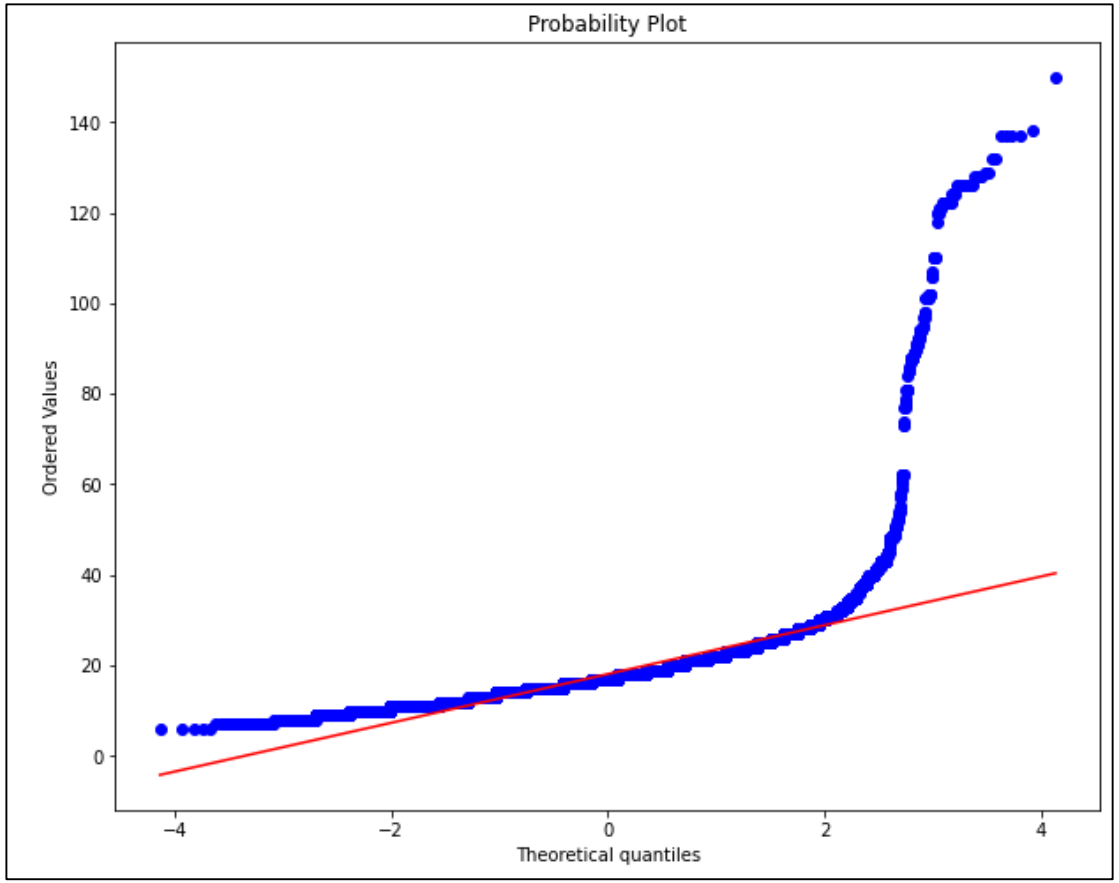
```
from scipy import stats
stats.kstest(fueleco.city08, cdf='norm')#檢查是否常態分佈
```

```
KstestResult(statistic=0.9999999990134123, pvalue=0.0)
```

#pvalue 小於 0.05 代表不是常態分佈

```
from scipy import stats
fig, ax = plt.subplots(figsize=(10, 8))
stats.probplot(fueleco.city08, plot=ax)
```

```
((array([-4.1352692 , -3.92687024, -3.81314873, ...,  3.81314873,
         3.92687024,  4.1352692 ]),
  array([  6,   6,   6, ..., 137, 138, 150], dtype=int64)),
 (5.385946629915974, 18.077798521776934, 0.772587941459713))
```



比較連續欄位間的關聯性

```
fueleco.city08.cov(fueleco.highway08)
```

```
46.33326023673624
```

#cov() 計算共變異數

```
fueleco.city08.cov(fueleco.comb08)
```

```
47.419946678190776
```

```
fueleco.city08.cov(fueleco.cylinders)
```

```
-5.931560263764768
```

```
fueleco.city08.corr(fueleco.highway08)
```

```
0.932494506228495
```

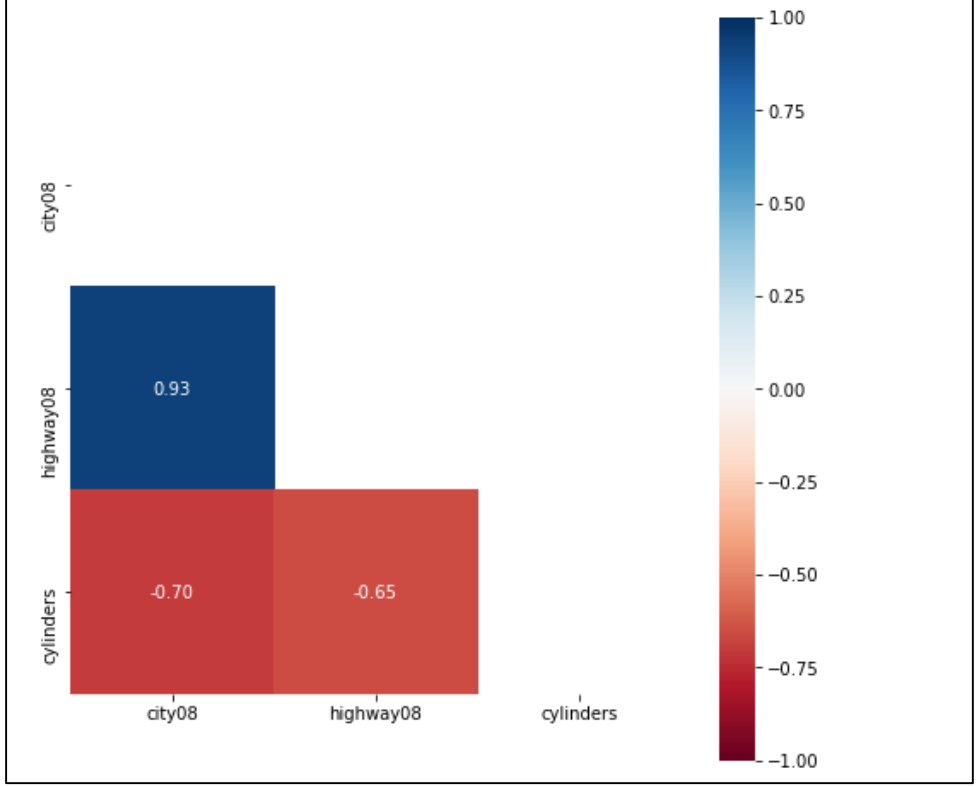
#corr() 計算皮爾森相關性

```
fueleco.city08.corr(fueleco.cylinders)
```

```
-0.7016548423827895
```

```
import seaborn as sns
fig, ax = plt.subplots(figsize=(8,8))
corr = fueleco[['city08', 'highway08', 'cylinders']].corr()
mask = np.zeros_like(corr, dtype=np.bool)
mask[np.triu_indices_from(mask)] = True
sns.heatmap(corr, mask=mask,
            fmt='.2f', annot=True, ax=ax, cmap='RdBu', vmin=-1, vmax=1,
            square=True)
```

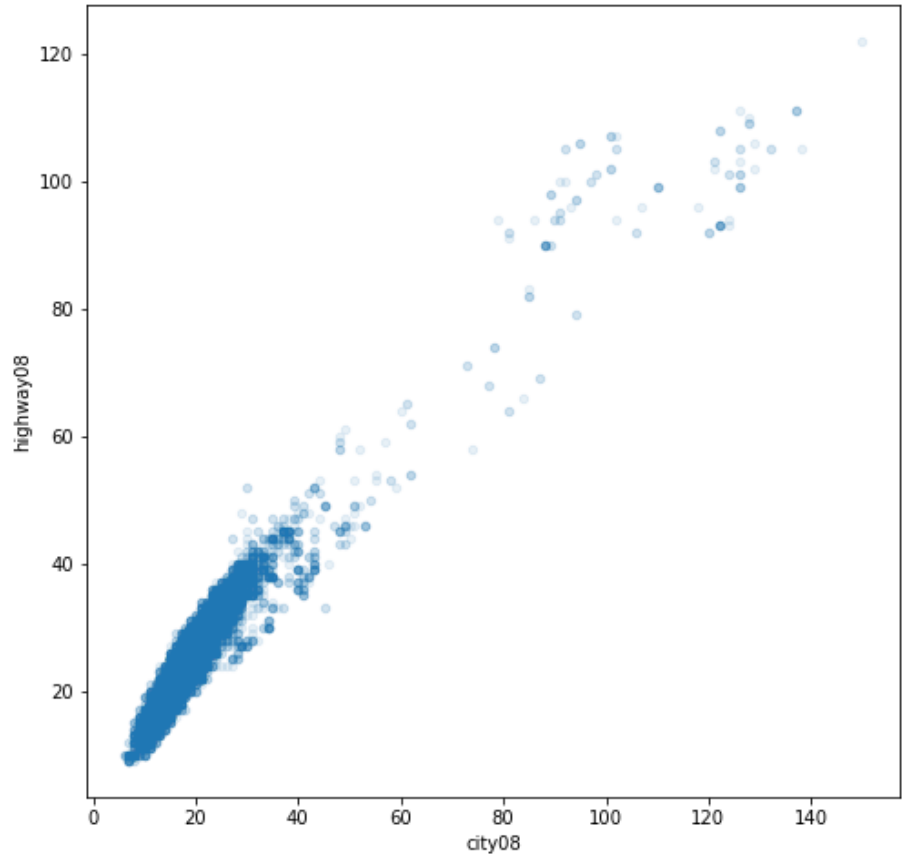
```
<matplotlib.axes._subplots.AxesSubplot at 0x228992a3eb0>
```



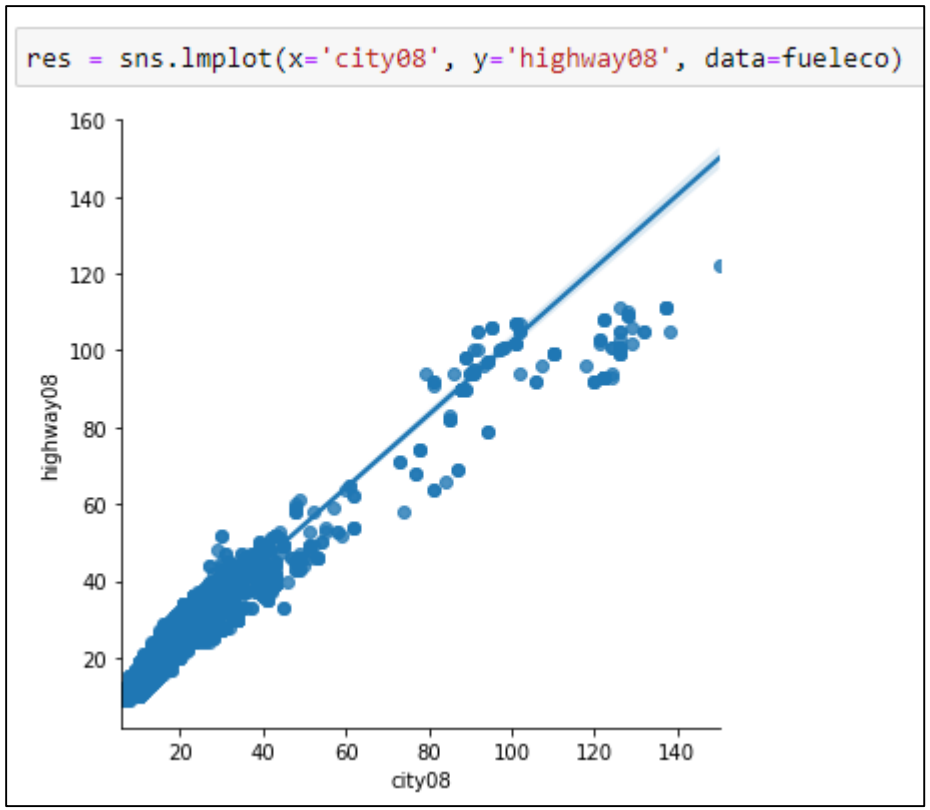
比較連續欄位間的關聯性

```
fig, ax = plt.subplots(figsize=(8,8))
fueleco.plot.scatter(x='city08', y='highway08', alpha=.1, ax=ax)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x228993189a0>
```




```
res = sns.lmplot(x='city08', y='highway08', data=fueleco)
```



Python 和 Pandas 日期工具的區別

```
import pandas as pd
import numpy as np
pd.set_option('max_columns', 4, 'max_rows', 10,'max_colwidth', 12)
```

```
import datetime
date = datetime.date(year=2022, month=6, day=7)
time = datetime.time(hour=12, minute=30, second=19, microsecond=463198)
dt = datetime.datetime(year=2022, month=6, day=7, hour=12, minute=30, second=19,
                       microsecond=463198)
print(f'date is {date}')
```

```
date is 2022-06-07
```

```
print(f'time is {time}')
```

```
time is 12:30:19.463198
```

```
print(f'datetime is {dt}')
```

```
datetime is 2022-06-07 12:30:19.463198
```

```
td = datetime.timedelta(weeks=2, days=5, hours=10,
                        minutes=20, seconds=6.73,
                        milliseconds=99, microseconds=8)
td
```

```
datetime.timedelta(days=19, seconds=37206, microseconds=829008)
```

```
print(f'new date is {date+td}')
```

```
new date is 2022-06-26
```

```
print(f'new datetime is {dt+td}')
```

```
new datetime is 2022-06-26 22:50:26.292206
```

```
# time + td #error is existed because no date.
```

Python 和 Pandas 日期工具的區別

```
pd.Timestamp(year=2021, month=12, day=21, hour=5,
             minute=10, second=8, microsecond=99)
```

```
Timestamp('2021-12-21 05:10:08.000099')
```

```
pd.Timestamp('2016/1/10')
```

```
Timestamp('2016-01-10 00:00:00')
```

```
pd.Timestamp('2014-5/10')
```

```
Timestamp('2014-05-10 00:00:00')
```

```
pd.Timestamp('Jan 3, 2019 20:45.56')
```

```
Timestamp('2019-01-03 20:45:33')
```

```
pd.Timestamp('2016-01-05T05:34:43.123456789')
```

```
Timestamp('2016-01-05 05:34:43.123456789')
```

```
pd.to_datetime('2015-5-13')
```

```
Timestamp('2015-05-13 00:00:00')
```

```
pd.to_datetime('2015-13-5', dayfirst=True)
```

```
Timestamp('2015-05-13 00:00:00')
```

```
pd.to_datetime('Start Date: Sep 30, 2017 Start Time: 1:30 pm',
               format='Start Date: %b %d, %Y Start Time: %I:%M %p')
```

```
Timestamp('2017-09-30 13:30:00')
```

```
pd.to_datetime(100, unit='D', origin='2013-1-1')
```

```
Timestamp('2013-04-11 00:00:00')
```

Python 和 Pandas 日期工具的區別

```
pd.Timestamp(500)
```

```
Timestamp('1970-01-01 00:00:00.000000500')
```

```
pd.Timestamp(5000, unit='D')
```

```
Timestamp('1983-09-10 00:00:00')
```

```
s = pd.Series([10, 100, 1000, 10000])
pd.to_datetime(s, unit='D')
```

```
0   1970-01-11
1   1970-04-11
2   1972-09-27
3   1997-05-19
dtype: datetime64[ns]
```

```
s = pd.Series(['12-5-2015', '14-1-2013', '20/12/2017', '40/23/2017'])
pd.to_datetime(s, dayfirst=True, errors='coerce')
```

```
0   2015-05-12
1   2013-01-14
2   2017-12-20
3          NaT
dtype: datetime64[ns]
```

Python 和 Pandas 日期工具的區別

```
s = pd.Series([10, 100])
pd.to_timedelta(s, unit='s')
```

```
0   00:00:10
1   00:01:40
dtype: timedelta64[ns]
```

```
pd.Timedelta('12 days 5 hours 3 minutes') * 2
```

```
Timedelta('24 days 10:06:00')
```

```
(pd.Timestamp('1/1/2022') + pd.Timedelta('12 days 5 hours 3 minutes') * 2)
```

```
Timestamp('2022-01-25 10:06:00')
```

```
td1 = pd.to_timedelta([10, 100], unit='s')
td2 = pd.to_timedelta(['3 hours', '4 hours'])
td1 + td2
```

```
TimedeltaIndex(['03:00:10', '04:01:40'], dtype='timedelta64[ns]', freq=None)
```

```
ts = pd.Timestamp('2021-10-1 4:23:23.9')
ts.ceil('h')
```

```
Timestamp('2021-10-01 05:00:00')
```

```
ts.year, ts.month, ts.day, ts.hour, ts.minute, ts.second
```

```
(2021, 10, 1, 4, 23, 23)
```

```
ts.dayofweek, ts.dayofyear, ts.daysinmonth
```

```
(4, 274, 31)
```

```
ts.to_pydatetime()
```

```
datetime.datetime(2021, 10, 1, 4, 23, 23, 900000)
```

```
td = pd.Timedelta(125.8723, unit='h')
td
```

```
Timedelta('5 days 05:52:20.280000')
```

```
td.round('min')
```

```
Timedelta('5 days 05:52:00')
```

```
td.components
```

```
Components(days=5, hours=5, minutes=52, seconds=20, milliseconds=280, microseconds=0, nanoseconds=0)
```

```
td.total_seconds()
```

```
453140.28
```